IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 9, 2021, accepted February 22, 2021, date of publication February 26, 2021, date of current version March 15, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3062790*

# Quantum-Inspired Genetic Algorithm for Resource-Constrained Project-Scheduling

**HATEM M. H. SAAD[1,2], RIPON K. CHAKRABORTTY[1], (Member, IEEE), SABER ELSAYED[3], (Member, IEEE), AND MICHAEL J. RYAN[1], (Senior Member, IEEE)**

[1]Capability Systems Centre, School of Engineering and Information Technology, University of New South Wales at Canberra, Canberra, ACT 2612, Australia
[2]Department of Physics, Faculty of Science, Al-Azhar University, Cairo 11651, Egypt
[3]School of Engineering and Information Technology, University of New South Wales at Canberra, Canberra, ACT 2612, Australia

Corresponding author: Hatem M. H. Saad (hatemmhsaad@gmail.com)

This work was supported by the Capability Systems Centre, School of Engineering and Information Technology, University of New South Wales at Canberra.

**ABSTRACT** The Resource-Constrained Project-Scheduling Problem (RCPSP) is an NP-hard problem which can be found in many research domains. The optimal solution of the RCPSP problems requires a balance between exploration/exploitation and diversification/intensification. With this in mind, quantum-inspired evolutionary algorithms' ability to improve the population and quality of solutions, this work investigates the performance of a quantum-inspired genetic algorithm (QIGA), which has been adapted to work with RCPSPs. The proposed QIGA possesses the same structure as classical genetic algorithms, but the initial and updated populations are implemented using quantum gates and quantum superposition, bearing in mind the adaptation of such operators to fit with discrete problems. This work aims to solve RCPSPs using the QIGA to investigate the influence of the various quantum parameters in the proposed algorithm, such as the quantum population size, different options for the quantum gates and re-combination to use in the QIGA. The well-known PSPLIB benchmark instances of J30, J60 and J120 activities are used to test the effectiveness and performance of the proposed QIGA. It is apparent from the results that quantum mutation, quantum crossover and representation of quantum superposition using quantum gates enhances population diversity. The QIGA is also found to outperform many other evolutionary algorithms.

**INDEX TERMS** Quantum-inspired evolutionary algorithm, resource-constrained project-scheduling problems, NP-hard problems, genetic algorithm, quantum gates, quantum superposition.

## I. INTRODUCTION

Over recent years, researchers have faced difficulties in solving NP-hard problems, such as the Resources-Constrained Project-Scheduling Problem (RCPSP) [1], [2], leading them to propose various innovative algorithms. Numerous optimisation algorithms have been developed for solving such problems, aimed at minimising the duration time (makespan) while satisfying the precedence and resource constraints. The classical optimisation approaches (i.e., exact methods), such as integer programming [2], branch-and-bound (B&B) algorithms [3], and dynamic programming [4], have been found to be impractical in a large number of cases (i.e., they could not solve problems with more than 60 activities) [5].

Researchers have therefore been trying to overcome the solving complexity of the RCPSP with a large number of activity instances by introducing different approaches, such as the use of heuristics, meta-heuristics and hybrid versions. Although solutions from such algorithms were found to be valid, they were still unable to guarantee optimal solutions, mainly when varying the complexity of the instances. More explanations and details in this regard can be found in [5]–[9]. Not surprisingly, due to this vital shortcoming, researchers are still proposing new algorithms to improve the solution quality.

In the last two decades, some promising algorithms have been inspired by quantum-mechanics concepts, such as standing waves, interference, coherence, qubits, quantum gates and superposition of states [10]. These quantum concepts have been combined with the concepts of evolutionary computing to produce quantum-inspired evolutionary algorithms [11]. Such algorithms were found to be powerful in solving complex combinatorial and numerical optimisation problems, and have performed better than their conventional counterparts. The first quantum-inspired genetic algorithm (QIGA) was proposed by Narayanan, and Moore [12] and showed the

The associate editor coordinating the review of this manuscript and approving it for publication was Francisco Rafael Marques Lima.

feasibility of solving the Travelling-Salesperson Problem. Han and Kim [13] proposed a genetic quantum algorithm (GQA), then improved it to use parallel calculations [14]. They also proposed another QIEA [11] to solve knapsack problems which was found to perform quite well, even with only one qubit individual. They explained quantum-computational concepts using quantum mechanics, adding explanations of interference, probabilistic quantum representation, qubit individuals and quantum chromosomes. They also introduced quantum migration and local and global migration. This was followed by the use of a numerical real coded representation in implementing QIEA for benchmark optimisation problems by Alfares *et al.* [15] and Cruz *et al.* [16], leading to a reduction in the memory required, better convergence time and better results than other algorithms.

There are several advantages of the quantum-inspired meta-heuristics that motivated us to propose a quantum-inspired genetic algorithm (QIGA) to solve the RCPSP. **(1)** Exponential quantum parallelism using quantum gates can reduce the search time and improve computational performance by computing with all values of a particular variable concurrently; this also improves the quality of the solution [10], [17]. **(2)** The representation of individuals in a population using quantum superposition and quantum gates leads to (a) more population diversity; (b) a stronger search capability; (c) better convergence speed and accuracy; and (d) effective escape from local optima. The small number of individuals means the algorithm can easily explore the search space for a global solution even with only one element [10], [11], [17]. **(3)** There is a balance between diversification/intensification and exploration/exploitation [18].

Wang *et al.* [19] introduced further improvements to the QIGA with the introduction of a simple determination of the quantum rotation angle using an auto-adjust rotation angle; quantum mutation using quantum NOT gates; and a quantum disaster operation by initialising new chromosomes randomly. The QIGA has been effective in solving many complex problems such as: (1) the Travelling-Purchaser Problem, in which the QIGA outperformed traditional GAs in terms of solution quality and computation time [20]; (2) the Antenna-Positioning Problem (APP), an NP-hard binary optimisation problem, in which the QIGA outperformed both PBIL and GAs in many benchmarks [21]; (3) real-time scheduling tasks in a multiprocessor environment, in which the QIGA outperformed its classical counterparts in scheduling accuracy and time in a multi-objective sense [22]; (4) stochastic job shop scheduling [23]; and (5) the Double Digest Problem, a fundamental problem in bioinformatics [24].

The success of the QIGA algorithm in solving the above problems motivated us to examine the adaptation of QIGAs for solving the RCPSP. Our main contributions in this paper can be summarised as follows. **(1)** A new QIGA is derived for solving the RCPSP with the implementation of a local search to accelerate the convergence rate, quantum mutation, single-point and two-point quantum crossover, and various quantum gates such as the quantum rotation gate, the Hadamard gate and the quantum NOT gate. To our knowledge, this has been previously proposed in the RCPSP context. **(2)** The paper analyses the effect of different quantum parameters, such as quantum rotation gates, rotation angles, and quantum mutation and quantum crossover. **(3)** The proposed QIGA is tested by solving the well-known PSPLIB benchmark data sets [25], [26]), with 30, 60 and 120 activities, containing 480, 480 and 600 problems, respectively. **(4)** Comparative experiments are carried out, and the QIGA found to outperform other evolutionary algorithms.

The paper is organised as follows. A brief review of the meta-heuristics used to solve the RCPSP and of quantum-inspired evolutionary algorithms is presented in Section II. Section III explains the mathematical formulation of an RCPSP. The theoretical background of quantum computing is reviewed in Section IV. This is followed by adapting a QIGA to solve the RCPSP in Section V. In Sections VI–IX, we present the experimental results and discussion, with conclusions presented in Section X.

## II. LITERATURE REVIEW

Recently, meta-heuristic algorithms and QIEAs have produced promising results in solving NP-hard problems. In tackling RCPSPs, meta-heuristic algorithms demonstrated improvements in the solutions for most of the problems, but the optimal solutions for many activities are still not guaranteed because of the stochastic nature of these algorithms. On the other hand, the advantages and quality of the results demonstrated by the QIEAs have attracted researchers' attention. QIEAs offered better results in solving NP-hard problems, such as the Travelling-Salesperson Problem and knapsack problems [17].

In this section, we present a brief review of the existing solutions for both types of algorithms.

### A. RECENT META-HEURISTIC APPROACHES FOR THE RCPSP

Blum and Roli [27] have summarised the ways of differentiating and classifying the meta-heuristic algorithms as follows. **(1)** A classification based on the origin of the algorithm: ones inspired by nature such as genetic algorithms (GA) citezhang2008genetic, Particle-Swarm optimisation (PSO) [28] and Ant-Colony optimisation (ACO) [29]; and non-natural algorithms such as the Tabu Search and Iterated Local Search [30]. **(2)** A classification based on the number of solutions used at the same time: algorithms that consist of multiple points in the search space and are evolutionary, called *population-based* meta-heuristics; and algorithms that work on a single solution, called *trajectory algorithms* [5]. **(3)** A classification based on how the objective function is used: *dynamic algorithms* that modify the objective function during the search to escape from local minima; and *static algorithms* that do not modify the objective function [27]. **(4)** A classification based on the neighbourhood structures:

one single neighbourhood structure versus a set of neighbourhood structures [31]. **(5)** A classification based on memory usage: the use of long-term memory (memory-usage algorithms) or short-term memory (memory-less algorithms) [27].

Recently, Pellerin *et al.* [5] classified the meta-heuristics for solving RCPSPs into three classes: **(1)** population-based meta-heuristics; **(2)** local search meta-heuristics; and **(3)** learning meta-heuristics. Of these, population-based meta-heuristics (such as PSO [28], ACO [29], differential evolution (DE) [32] and GA [33]) have shown a good performance in solving RCPSPs. Researchers have worked to obtain the best compromise between these algorithms to produce high-quality solutions within reasonable computational times, bearing in mind their adaptability, simplicity of execution and accuracy. These efforts have resulted in various kinds of meta-heuristic methods and also hybrid meta-heuristic methods for solving RCPSPs. For a more detailed discussion, readers are referred to [5], [9]. These algorithms are usually evaluated according to objective metrics such as the execution time and the solution quality, and tested using a set of benchmark functions such as J30, J60 and J120 from the PSPLIB. The meta-heuristic algorithms have performed well, particularly in reducing the chance of becoming trapped in local optima. In addition, meta-heuristics have found solutions that were close to optimal, but the stochastic search characteristics in these algorithms mean that an optimal solution could not be guaranteed. Not surprisingly, researchers are still proposing new algorithms to improve the quality of the solution.

### B. QUANTUM-INSPIRED EVOLUTIONARY ALGORITHMS

The first construction of a quantum-mechanical model for quantum computing was proposed early in 1980 by Benioff [34], and Feynman [35], [36], who explained the possibility of simulating quantum principles for computation using quantum physics. Quantum algorithms were then developed and formalised by Deutsch in 1985 [37], Shor in 1994 [38] and Grover in 1996 [39]. Explanations of the quantum-computational concepts of superposition, qubits, quantum gates and quantum entanglement can be found in [40]–[42].

In recent years, many quantum-inspired meta-heuristics have been developed for solving different complex problems. The quantum-inspired acromyrmex evolutionary algorithm (QIAEA) [43] outperformed classical GA for different complex systems of 15 benchmark optimisation functions each of which was characterised by two-dimensional minimisation problems. Other quantum-inspired meta-heuristics are the quantum-inspired Particle Swarm Optimisation (QIPSO) [44] and Firefly Algorithm with Particle Swarm Optimization (QIFAPSO) [45]. These algorithms were able to solve NP-problems such as the 0–1 knapsack problem and Travelling-Salesperson Problem better than many traditional algorithms [17]. Other applications of the QIPSO were in the design of a superconducting magnetic-energy-storage system [46], and in solving J30 of the RCPSP [47]; it was effective in reducing the makespan. Another quantum-inspired meta-heuristics, the quantum-inspired differential-evolution algorithm (QIDEA), was better than the classical methods in solving problems such as the N-Queens Problem [48] and the deep-belief network [49]. The success of such algorithms motivated researchers to make further modifications; for example, a hybridisation of quantum-inspired algorithms with classical algorithms (e.g. [50]–[56]). One such hybrid algorithm [53] showed its effectiveness in solving the multi-objective flow shop scheduling problem (FSSP). For a more detailed discussion on the quantum-inspired meta-heuristics, readers are referred to [10], [17], [42].

## III. RCPSP: MATHEMATICAL MODEL

RCPSPs have been extensively investigated since their introduction in 1969 [57]; they are NP-hard combinatorial optimisation problems in the strong sense. Due to the importance of RCPSPs in real-world applications, huge efforts have been made to solve them [5], [58].

Each project in the standard version of an RCPSP comprises a set of activities $\{j_0, j_1, \ldots, j_n, j_{n+1}\}$ that cannot be interrupted and are performed sequentially according to precedence and resource constraints. An activity is prevented from starting before its predecessors have finished; this is known as a precedence constraint. The main objective of an RCPSP is to minimise the makespan (the completion time of a project). Duration of activities can be represented as $\{d_{j_0}, d_{j_1}, \ldots, d_{j_n}, d_{j_{n+1}}\}$. Here, we focus on solving a single-mode RCPSP using resources $R = \{R_1, \ldots, R_k, \ldots, R_K\}$, $\forall k = \{1, 2, \ldots, K\}$; each activity $j$ demands a resource $r_{kj}$ for each period. The first and last activities, $j_0$ and $j_{n+1}$ are dummies, with their durations and resources set to zero, i.e. $d_{j_0} = d_{j_{n+1}} = 0$ and $r_{kj_0} = r_{kj_{n+1}} = 0$.

On applying the precedence and resource constraints, the RCPSP can be summarised as follows.

1) The duration of the project (or the makespan) is to be minimised by minimising the finishing time of the last dummy activity, $FT_{j_{n+1}}$:

$$\text{Minimise } FT_{j_{n+1}}. \tag{1}$$

2) A successor starting time is always greater than or equal to the maximum finishing time of its predecessors. This precedence constraint can be written as

$$FT_j \leq FT_{j+1} - d_j, \quad j = \{j_1, j_2, \ldots, j_n\}. \tag{2}$$

3) The amount of resources $r_k$ required by activity $j$ must be less than the available $R_k$ for a resource type $k$. At any time $t$, for a set of activities $J(t)$, the resource constraints can be implemented as

$$\sum_{j \in J(t)} r_{kj} \leq R_k, \quad \forall k = \{1, 2, \ldots, K\}. \tag{3}$$

4) The finishing times of activities $j = \{j_1, j_2, \ldots, j_n\}$ must be non-negative:

$$FT_j \geq 0. \tag{4}$$

## IV. BASICS OF QUANTUM COMPUTING

In conventional computations, information is recorded at a macroscopic level and represented using two logical basis states; a "0" or "1". These states form a bit of information coded in the current flowing in a circuit: closed for "0" and open for "1". However, quantum computations are represented at a microscopic level using quantum states. For example, an atom, according to quantum mechanics, can be in the ground state or in an excited state, depending on the absorbed energy. Inspired by quantum mechanics, this allows information to be represented using Dirac notation: in the ground state as $|0\rangle$ and in the excited state as $|1\rangle$ [59]. These are the two basis states that form a qubit [59]. The qubit stores the quantum information; every qubit results from the superposition of two or more quantum basis states.

Again from the quantum-mechanical point of view, the qubit $|\psi\rangle$ is an element of a Hilbert space in which the quantum atom is located [17]. The square of the wave function (qubit) is the probability of locating the particle somewhere. A particular superposition is produced by solving the linear time-dependent Schrödinger equation

$$i\hbar\frac{d}{dt}|\psi\rangle = H|\psi\rangle. \tag{5}$$

Thus, the linear superposition of the individual quantum states

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \quad \text{and} \quad |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$$

can be written as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{bmatrix}\alpha\\\beta\end{bmatrix}. \tag{6}$$

In addition to the $|0\rangle$ and $|1\rangle$ basis states, the probability of the basis states leads to producing mixed quantum states [11], [24]. $\alpha$ and $\beta$ are complex numbers that assign the probability amplitudes of the quantum atom for finding a qubit in states $|0\rangle$ and $|1\rangle$, respectively. The probability amplitudes in quantum computing provide a relationship between the qubits and the results of observations of the quantum system [17].

### A. QUBIT REPRESENTATION

The representation $\begin{bmatrix}\alpha\\\beta\end{bmatrix}$ forms the $i$th qubit; if there are $n$ possible quantum states for the quantum atom, there will be $n$ complex probabilities for this atom at that location. The probabilities must satisfy the normalisation condition $|\alpha|^2 + |\beta|^2 = 1$. One important aspect of the quantum-measurement process is that the probability of outcome $|0\rangle$ is $|\alpha|^2$ and of $|1\rangle$ is $|\beta|^2$. This is the concept of a qubit measurement giving a single classical bit of information, which collapses from the quantum state to one of the two classical states $\begin{bmatrix}1\\0\end{bmatrix}$ or $\begin{bmatrix}0\\1\end{bmatrix}$.

The difference between the memory in classical and quantum computation is that $n$ bits in a classical computation require $2^n$ binary states to represent them; however, the $n$ qubits require only $n$ quantum states.

The eigenvalues of the Schrödinger equation are transformed using the unitary transformation operator $U$, which is the solution of the Schrödinger equation:

$$|\psi\rangle_{\text{output}} = U\,|\psi\rangle_{\text{input}}\,. \tag{7}$$

The unitary transformation can be implemented and decomposed using various reversible *quantum gates* around the Bloch sphere to manipulate the quantum superposition [60]. The quantum states of the qubits cover the whole Bloch sphere. Using the spherical-coordinate system $(r, \theta, \varphi)$; Figure 1), any quantum state on the Bloch sphere can be written as

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle\,, \tag{8}$$

where the angle $\theta \in [0, \pi]$ corresponds to latitude on the Bloch sphere and $\varphi \in [0, 2\pi]$ to longitude.

### B. QUANTUM GATES

There are several important quantum gates that can be used to perform the transformation reversibly on a single qubit [17], [42]. For example, if we want to perform the identity transformation, no change, on a qubit, we use the identity gate $I = \begin{bmatrix}1&0\\0&1\end{bmatrix}$. If we want to rotate a qubit around the Bloch sphere's $x$, $y$ or $z$ axis, we use the respective Pauli gates

$$X = \begin{bmatrix}0&1\\1&0\end{bmatrix}, \quad Y = \begin{bmatrix}0&-i\\i&0\end{bmatrix}, \quad Z = \begin{bmatrix}1&0\\0&-1\end{bmatrix}.$$

The Pauli $X$ gate is equivalent to the classical NOT gate [17].

One of the most important transformation quantum gates is the Hadamard gate [17], which can act on a single qubit or multiple qubits: $H = \frac{1}{\sqrt{2}}\begin{bmatrix}1&1\\1&-1\end{bmatrix}$. We apply the Hadamard gate to the qubit $|0\rangle$ by multiplying the matrix $\begin{bmatrix}1\\0\end{bmatrix}$ by $H$:

$$\begin{aligned} H\,|0\rangle &= \frac{1}{\sqrt{2}}\begin{bmatrix}1&1\\1&-1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} \\ &= \frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\end{bmatrix} + \frac{1}{\sqrt{2}}\begin{bmatrix}0\\1\end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}. \end{aligned} \tag{9}$$

This means, the measurement of qubit $|0\rangle$ using the Hadamard gate leads to the superposition of the quantum states $|0\rangle$ and $|1\rangle$ in the same proportion. Similarly, the Hadamard gate maps $|1\rangle$ to $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$, along the negative $x$ axis of the Bloch sphere, also having equal proportions of $|0\rangle$ and $|1\rangle$. The representation of a quantum state on the Bloch sphere is shown in Figure 1.

Rotation around the Bloch sphere $x$, $y$ and $z$ axes can be generated by exponentiation of the Pauli matrices, known as the rotation gates:

$$R_x(\theta) = \begin{bmatrix}\cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right)\\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right)\end{bmatrix} \tag{10}$$

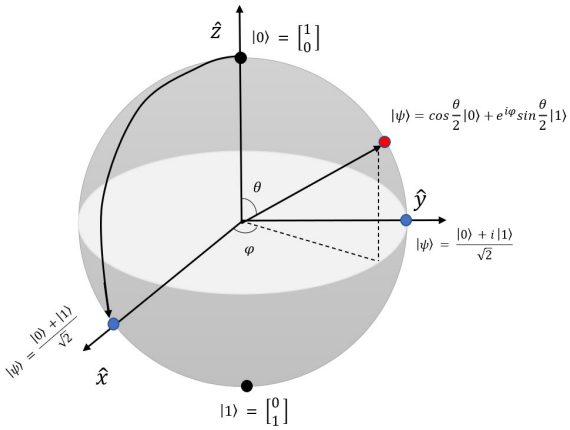



**FIGURE 1.** Geometrical representation on the Bloch sphere of the qubit $|\psi\rangle$. The basis qubits $|0\rangle$ and $|1\rangle$ are represented on the $z$ axis as the north and south poles. The superposition of quantum states can be represented using the rotation gates with angles $\theta \in [0, \pi]$ and $\varphi \in [0, 2\pi]$.

$$R_y(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{11}$$

$$R_z(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix} \tag{12}$$

These rotation quantum gates, used in the unitary transformation, increase or decrease the amplitude of the qubit along the $x$, $y$ or $z$ axis, respectively.

## V. THE PROPOSED QIGA FOR RCPSPs

The procedure of the proposed QIGA is similar to that of a classical GA but with the implementation of quantum concepts. As shown in Algorithm 1, QIGA consists of the initialisation of the quantum population, evaluation according to the fitness function which requires the conversion of binary values to their integer representations, selection, quantum crossover and quantum-mutation operators. To accelerate the convergence rate, a local search is also utilised. The evolutionary process is continued until a stopping criterion is met, that is the maximum number of fitness evaluations is reached, or the algorithm has converged to the best-known solution. Each component of the proposed QIGA is described in more detail in the following sub-sections.

### A. INITIALISATION

Similar to any population-based algorithm, the QIGA needs to start with a set of solutions, each of which is of size $n+2$ variables, (i.e. number of decision variables). The first and last activities are dummies, with their durations and resources set to zero.

As mentioned previously, a qubit in the quantum chromosomes is defined as $\begin{bmatrix}\alpha\\ \beta\end{bmatrix}$. The initial population of quantum

**Algorithm 1** Quantum Genetic Algorithm for RCPSPs

**Set** $g \leftarrow 0$; $FES \leftarrow 0$; $FES_{max} \leftarrow 50,000$; $QPS$.
**Initialise** $Q = \{\overrightarrow{Q}_1, \overrightarrow{Q}_2, \ldots, \overrightarrow{Q}_{QPS}\}$ (see Section V-A).
**Convert** $Q$ to binary $X_B = \{\overrightarrow{X}_{B,1}, \overrightarrow{X}_{B,2}, \ldots, \overrightarrow{X}_{B,QPS}\}$.
**Convert** $X_B$ to its discrete-based representation $X_D$.
**Repair** $X_D$ to satisfy all problems constraints; update $X_B$ and $Q$, if needed.
**Evaluate** $X_D$ and update $FES$.
To every solution in $X_D$, apply **local search** (see Section V-D) and update $FES$, $X_B$ and $Q$.
**while** $FES < FES_{max}$ *and optimal solution is not found* **do**
  $g \leftarrow g+1$;
  Generate new values for $Q$;
  Apply **crossover** and **mutation** operators to generate new $Q$ ($Q_{new}$) of size $QPS$;
  **Convert** $Q_{new}$ to its binary-based counterpart $X_{B,new}$;
  **Convert** $X_{B,new}$ to its discrete-based representation ($X_{D,new}$);
  **Repair** $X_{D,new}$ to satisfy all problems constraints and update $X_{B,new}$ and $Q_{new}$, if needed;
  **Evaluate** $X_D$ and update $FES$;
  Apply **local search** to $X_{D,_{new}}$ and update $FES$, $X_B$ and $Q$;
  Update $Q$, $X_B$, $X_D$ at $g+1$
**end**

chromosomes at generation $g=1$ with length $n+2$, is defined as

$$\overrightarrow{Q}_i = \left\{q_0, q_1, \ldots, q_j, \ldots, q_{n+1}\right\}, \quad \forall i = \{1, 2, \ldots, QPS\} \tag{13}$$

where each qubit chromosome is of length $m$ and can be written as

$$q_j = \begin{bmatrix} \alpha_1 & \alpha_2 & q\cdots & \alpha_m \\ \beta_1 & \beta_2 & \cdots & \beta_m \end{bmatrix} \tag{14}$$

To perform the quantum superposition in the initialisation, three variations with different quantum gates are used.

**Option 1:** Initialise the population of the amplitudes of the quantum chromosomes of length $n+2$, i.e. $j = 0, 1, 2, \ldots, n+1$, weighting the linear superposition of all possible states with the same probability amplitude by multiplying by $\frac{1}{\sqrt{2}}$:

$$\begin{bmatrix}\alpha'_m\\ \beta'_m\end{bmatrix} = \frac{1}{\sqrt{2}}|\psi\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}\alpha_m\\ \beta_m\end{bmatrix} = \begin{bmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{bmatrix}. \tag{15}$$

This can be achieved by multiplying $|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$ by the Hadamard quantum-gate matrix.

**Option 2:** Implement the Hadamard quantum-gate matrix for the amplitudes of all qubits in the chromosomes:

$$H|\psi\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}\alpha_m\\ \beta_m\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}\alpha_m + \beta_m\\ \alpha_m - \beta_m\end{bmatrix} \tag{16}$$

**Option 3:** Implement the Hadamard gate and rotation gate for the amplitudes of all qubits in the chromosomes, with the Bloch-sphere angle $\theta$ obtained in the range $(0, \pi]$.

$$\begin{bmatrix} \alpha'_m \\ \beta'_m \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \times \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \alpha_m \\ \beta_m \end{bmatrix} \quad (17)$$

If $\|\alpha|^2 < 0.5$, the corresponding binary value is set to 0; otherwise to 1. This means a binary population of classical chromosomes $X_i$ is produced, such that

$$\overrightarrow{X}_{B,i} = \{x_0, x_1, \ldots, x_{n+1}\}, \quad \forall i = \{1, 2, \ldots, QPS\}. \quad (18)$$

### B. FITNESS EVALUATION AND SELECTION OPERATOR

Evaluating and selecting the individuals in the quantum selection procedures is the same as in the classical procedures. The solutions are evaluated through the fitness function, based on Eq. (1), by minimising the makespan until the optimal or a near-optimal solution is reached. Because RCPSPs are integer-based problems, each $\overrightarrow{X}_{B,i}$ is converted to its discrete representation ($\overrightarrow{X}_{D,i}$) population. This is done by the default, and well-known, binary-to-discrete conversion approach. To avoid having redundant activities, which may appear due to this conversion, all discrete values generated are sorted in ascending order. The numbering of those activities is considered the new schedule ($\overrightarrow{X}_{D,i}$).

As the newly generated $\overrightarrow{X}_{D,i}$ may be infeasible, we apply a repair method to ensure that both the resource and precedence constraints are satisfied. This is done through the serial schedule generation method [9]. This repairing step may require updating the corresponding $\overrightarrow{Q}_i$ and $\overrightarrow{X}_{B,i}$.

Then, each $\overrightarrow{X}_{D,i}$ is evaluated, based on Eq. (1). To clarify, the fitness function of $\overrightarrow{X}_{D,i}$ represents the latest finish time among all activities.

### C. QUANTUM RECOMBINATION

It is worthy to highlight that no selection operation, before applying the crossover operator, is used. The reason for this is to maintain diversity, especially, when a small population size (i.e., 10) is used. Having said, elitism is still, that is, the best solution in each generation is survived to the next one.

The quantum population is updated using quantum gates, quantum crossover and quantum mutations which, in turn, results in the production of their corresponding binary and discrete solutions.

- **Updated population options (UPO):** The quantum population at generation $g$ ($Q_g$) is updated to produce the next generation $Q(g+1)$ using the same techniques as in the initialisation step, applying different quantum gates such as rotation gates and Hadamard gates.

$$\begin{bmatrix} \alpha_m^{g+1} \\ \beta_m^{g+1} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} \alpha_m^g \\ \beta_m^g \end{bmatrix} \quad \text{or} \quad (19)$$

$$\begin{bmatrix} \alpha_m^{g+1} \\ \beta_m^{g+1} \end{bmatrix} = H \begin{bmatrix} \alpha_m^g \\ \beta_m^g \end{bmatrix} \quad \text{or} \quad (20)$$

$$\begin{bmatrix} \alpha_m^{g+1} \\ \beta_m^{g+1} \end{bmatrix} = R_y(\theta) H \begin{bmatrix} \alpha_m^g \\ \beta_m^g \end{bmatrix}. \quad (21)$$

- **Quantum crossover:** We have examined single-point and two-point quantum crossovers. To use either crossover operator, two quantum individuals $\overrightarrow{Q}_1$ and $\overrightarrow{Q}_2$ are randomly selected to produce $\overrightarrow{Q}_{new,1}$ and $\overrightarrow{Q}_{new,2}$, such that:

*Single-point quantum crossover*

Parents

$$Q_i = \left[\begin{array}{cc|c|cc} \alpha_1^i & \alpha_2^i & \ldots & \alpha_h^i & \alpha_m^i \\ \beta_1^i & \beta_2^i & \ldots & \beta_h^i & \beta_m^i \end{array}\right] \quad (22)$$

$$Q_j = \left[\begin{array}{cc|c|cc} \alpha_1^j & \alpha_2^j & \ldots & \alpha_h^j & \alpha_m^j \\ \beta_1^j & \beta_2^j & \ldots & \beta_h^j & \beta_m^j \end{array}\right] \quad (23)$$

Offspring

$$Q_{new,i} = \left[\begin{array}{cc|c|cc} \alpha_1^i & \alpha_2^i & \ldots & \alpha_h^j & \alpha_m^j \\ \beta_1^i & \beta_2^i & \ldots & \beta_h^j & \beta_m^j \end{array}\right] \quad (24)$$

$$Q_{new,j} = \left[\begin{array}{cc|c|cc} \alpha_1^j & \alpha_2^j & \ldots & \alpha_h^i & \alpha_m^i \\ \beta_1^j & \beta_2^j & \ldots & \beta_h^i & \beta_m^i \end{array}\right] \quad (25)$$

*Two-point quantum crossover*

Parents

$$Q_i = \left[\begin{array}{cc|c|c|c} \alpha_1^i & \alpha_2^i & \ldots & \alpha_h^i & \ldots & \alpha_m^i \\ \beta_1^i & \beta_2^i & \ldots & \beta_h^i & \ldots & \beta_m^i \end{array}\right] \quad (26)$$

$$Q_j = \left[\begin{array}{cc|c|c|c} \alpha_1^j & \alpha_2^j & \ldots & \alpha_h^j & \ldots & \alpha_m^j \\ \beta_1^j & \beta_2^j & \ldots & \beta_h^j & \ldots & \beta_m^j \end{array}\right] \quad (27)$$

Offspring

$$Q_{new,i} = \left[\begin{array}{cc|c|c|c} \alpha_1^i & \alpha_2^i & \ldots & \alpha_h^j & \ldots & \alpha_m^i \\ \beta_1^i & \beta_2^i & \ldots & \beta_h^j & \ldots & \beta_m^i \end{array}\right] \quad (28)$$

$$Q_{new,j} = \left[\begin{array}{cc|c|c|c} \alpha_1^j & \alpha_2^j & \ldots & \alpha_h^i & \ldots & \alpha_m^j \\ \beta_1^j & \beta_2^j & \ldots & \beta_h^i & \ldots & \beta_m^j \end{array}\right] \quad (29)$$

This process continues until all new QPS quantum solutions are generated.

- **Quantum mutation:** The quantum-mutation process is inspired by the classical GA. The exchange between two different qubit positions in each quantum chromosome is performed using a quantum NOT gate:

$$Q = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \quad (30)$$

With a low mutation rate. The quantum mutation can be performed as

$$Q\begin{bmatrix}\alpha_m^{g+1}\\ \beta_m^{g+1}\end{bmatrix}=\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}\alpha_m^{g+1}\\ \beta_m^{g+1}\end{bmatrix}=\begin{bmatrix}\beta_m^{g+1}\\ \alpha_m^{g+1}\end{bmatrix}. \quad (31)$$

### D. LOCAL SEARCH AND SELECTION

Once the new solutions are generated using the above steps, each solution is converted to its discrete representation and hence repaired to satisfy all the constraints. Once the new population is generated, their fitness values are computed, and a local search is adopted to enhance its value.

The local search used here is similar to that in [9]. First, all the activities are sorted in descending order of finishing time. Then, each activity is shifted to the right as far as possible, bearing in mind the resource and successor constraints. After that, all the activities are shifted to left by sorting the starting times. Subsequently, the corresponding quantum and discrete vectors are updated.

Finally, the new (quantum, binary and discrete) solutions replace the old ones, with the best solution from the previous generation surviving to the next one.

### E. TIME COMPLEXITY

Given the steps in Algorithm 1, the time complexity of each component of QIGA is as follows.

1) The initialisation of each quantum chromosome has a time complexity of $O(2.n.m)$. This results in a time complexity equivalent to $O(2.QPS.m.n)$ overall populations. By focusing only on the dominant terms and removing any constants, the time complexity for the initialisation step is $O(m.n)$.
2) Mapping solutions from quantum to binary representation will have the same time complexity as the previous step, i.e., $O(m.n)$
3) Similarly, the time complexity for mapping solutions to discrete-based ones is $O(m.n)$. Besides, for each solution, removing any redundant activities which may appear due to this conversion requires sorting the discrete values in ascending order, then numbering those activities. This extra step has a time complexity of $O(n\log(n))$ for sorting plus $O(n)$ for the numbering activities after sorting. Hence, by focusing on the dominant term among those three values $\big(O(n\log(n))+O(n)+O(m.n)\big)$, the time complexity for this step, per solution, is $O(mn)$. Note constants are ignored.
4) The previous step may result in infeasible solutions; making sure that each activity satisfies the precedence and resource constraints result in time complexity of $O(K.n^2)$
5) The time complexity of local search, per solution, is $O(2.K.n^2) \approx O(K.n^2)$.
6) The fitness function has $O(n)$ complexity.
7) The evolution of solutions per generation is $O(QPS.m.n)+O(QPS.m.n)$ (for crossover) + $O(QPS.m.n)$ (for mutation), in addition to $O(QPS.m.n) + O(QPS.m.n) + O(QPS.K.n^2) + O(QPS.K.n^2)+O(n)$ for the remaining steps. By focusing on the dominant terms and removing any constants. the evolution step, per generation, has $O(K.n^2)$ complexity. For $G$ generations, the time complexity is $O(G.K.n^2)$.

By summing up all seven steps, focusing on the dominant terms and removing any constants, we can claim that the overall complexity, over $G$ generations, is $O(G.K.n^2)$.

## VI. EXPERIMENTAL WORK

The experimental work was carried out on the PSPLIB benchmark instances of the RCPSP using J30, J60 and J120 activities containing 480, 480 and 600 problems, respectively, each of which has four resource types [25], [26]. The problems in the J30, J60 and J120 activities possess different complexity levels, selected according to three complexity factors [25]. **(1)** Resource Factor ($RF$) measures the average proportion of resources needed for each job: $RF = 1$ when all resources are needed for each job; $RF = 0$ when no resources are needed. **(2)** Network Complexity ($NC$) measures the average number of non-redundant precedence for each activity. **(3)** Resource Strength ($RS$) measures the proportion of resource usage and availability: $RS = 1$ for a problem with unconstrained resources; $RS = 0$ highly constrained resources.

To generate the instances and problems in the J30 and J60 activities, $NC \in \{1.5, 1.8, 2.1\}$, $RF \in \{0.25, 0.5, 0.75, 1\}$ and $Rs \in \{0.2, 0.5, 0.7, 1\}$. For the J120, the same parameter levels were set for $NC$ and $RF$ but $RS \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ [61].

We evaluated the solution by recording the deviations of each problem in 15 runs with three different fitness evaluations, 1000, 5000 and 50, 000. The deviation $Dev_p$ for the $P$ problems in the activities J30, J60 and J120 in each generation was determined by comparing the best fitness values $F_{best,p}$ with the corresponding lower bounds $LB_P$, as follows:

$$Dev_p = \frac{F_{best,p} - LB_P}{LB_P} \quad \forall p = 1, 2, \ldots, P \quad (32)$$

The average deviation over all three activities, J30, J60 and J120, was determined as

$$\overline{Dev}_{Activity} = \frac{100}{P}\sum_{p=1}^{P} Dev_p \ \%. \quad (33)$$

## VII. PARAMETRIC ANALYSIS

To determine the best combination of the quantum parameters to use in QIGA, we designed experiments using the Taguchi method [32].

The quantum parameters used were set as follows: quantum population sizes QPS = 10, 20, 30; and quantum mutation rates QMR = 0.05, 0.1, 0.2. The initialisation population and updated population options were defined using the quantum-gate options (Section V-A); rotation angles RA = $\pi/18$, $\pi/6$, $\pi/3$; and quantum crossover QC single-point and two-point. The Taguchi method produced

**TABLE 1.** Orthogonal table of 27 combinations of the quantum parameters from the Taguchi method.

| Exp. | Quantum parameter | | | | | Mean makespan (6 problems) | | |
|---|---|---|---|---|---|---|---|---|
| | QPS | RA | MR | IPO | UPO | J30 | J60 | J120 |
| 1 | 10 | | 0.05 | Option 1 | Option 1 | 60.34968912 | 69.63894858 | 93.45249412 |
| 2 | 10 | $\frac{\pi}{18}$ | 0.05 | Option 1 | Option 2 | 60.34968912 | 64.99431885 | 93.45249412 |
| 3 | 10 | | 0.05 | Option 1 | Option 3 | 60.34906250 | 64.99331433 | 93.45268257 |
| 4 | 10 | | 0.1 | Option 2 | Option 1 | 60.34910645 | 64.99307361 | 93.45302863 |
| 5 | 10 | $\frac{\pi}{6}$ | 0.1 | Option 2 | Option 2 | 60.34910645 | 64.99307361 | 93.45302863 |
| 6 | 10 | | 0.1 | Option 2 | Option 3 | 60.35073927 | 64.99239647 | 93.45404359 |
| 7 | 10 | | 0.2 | Option 3 | Option 1 | 60.34938930 | 64.99322936 | 93.45233948 |
| 8 | 10 | $\frac{\pi}{3}$ | 0.2 | Option 3 | Option 2 | 60.34938930 | 64.99322936 | 93.45233948 |
| 9 | 10 | | 0.2 | Option 3 | Option 3 | 60.34911089 | 64.99384153 | 93.45224111 |
| 10 | 20 | | 0.1 | Option 3 | Option 1 | 60.34896671 | 64.99377312 | 93.45285005 |
| 11 | 20 | $\frac{\pi}{18}$ | 0.1 | Option 3 | Option 2 | 60.34896671 | 64.99377312 | 93.45285005 |
| 12 | 20 | | 0.1 | Option 3 | Option 3 | 60.34824434 | 64.99393285 | 93.45419961 |
| 13 | 20 | | 0.2 | Option 1 | Option 1 | 60.34915381 | 64.99393724 | 93.4532943 |
| 14 | 20 | $\frac{\pi}{6}$ | 0.2 | Option 1 | Option 2 | 60.34915381 | 64.99393724 | 93.4532943 |
| 15 | 20 | | 0.2 | Option 1 | Option 3 | 60.34946360 | 64.99175802 | 93.45478593 |
| 16 | 20 | | 0.05 | Option 2 | Option 1 | 60.34867936 | 64.99414353 | 93.45322048 |
| 17 | 20 | $\frac{\pi}{3}$ | 0.05 | Option 2 | Option 2 | 60.34867936 | 64.99414353 | 93.45322048 |
| 18 | 20 | | 0.05 | Option 2 | Option 3 | 60.35258420 | 64.99455456 | 93.45263729 |
| 19 | 30 | | 0.2 | Option 2 | Option 1 | 60.34894080 | 64.99399791 | 93.45173129 |
| 20 | 30 | $\frac{\pi}{18}$ | 0.2 | Option 2 | Option 2 | 60.34894080 | 64.99399791 | 93.45173129 |
| 21 | 30 | | 0.2 | Option 2 | Option 3 | 60.34950653 | 64.99351656 | 93.45247877 |
| 22 | 30 | | 0.05 | Option 3 | Option 1 | 60.34932394 | 64.99342259 | 93.45342528 |
| 23 | 30 | $\frac{\pi}{6}$ | 0.05 | Option 3 | Option 2 | 60.34932394 | 64.99342259 | 93.45342528 |
| 24 | 30 | | 0.05 | Option 3 | Option 3 | 60.35166572 | 64.99405711 | 93.45360293 |
| 25 | 30 | | 0.1 | Option 1 | Option 1 | 60.34890586 | 64.99339636 | 93.45292156 |
| 26 | 30 | $\frac{\pi}{3}$ | 0.1 | Option 1 | Option 2 | 60.34890586 | 64.99339636 | 93.45292156 |
| 27 | 30 | | 0.1 | Option 1 | Option 3 | 60.35096127 | 64.99357151 | 93.4527676 |

27 different combinations (Table 1). Then, QIGA was tested with each combination of parameters on six test problems, 11, 121, 161, 281, 351, 391. These are the first problems in instances 2, 13, 17, 29, 36 and 40, respectively, and were selected based on their complexity as explained in [61].

Note that all experiments were run on a PC with a Core i7-4790 CPU3.60GHz, 16GB RAM with a 64-bit operating system and Windows 7 using MATLAB (R2020b) with some of its functions converted to C++ code by the MATLAB coder tool.

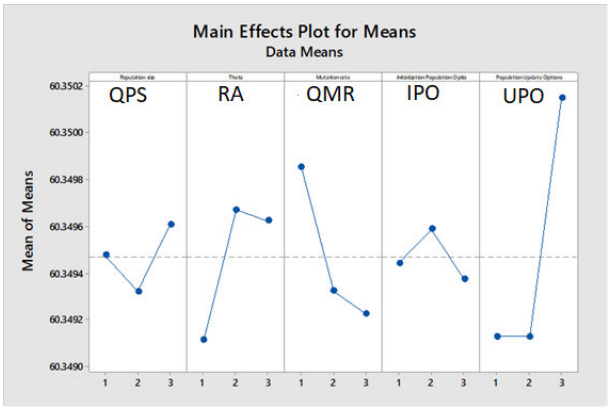
The 27 combinations of QIGA in Table 1 were tested for J30, J60 and J120, and produced 27 mean makespan values. These were used to produce the main effects plots of the quantum parameters for J30, J60 and J120 (Figure 2).

**J30:** The response figure (Figure 2a) shows the trend in each quantum parameter. For J30, the optimal combination produced by the Taguchi method was: QPS 20; RA $\theta = \pi/18$; QMR 0.2; initialisation population option (IPO): Option 3; updated population options (UPO): Option 1 or Option 2, and two-point quantum crossover. Figure 2a reveals that the makespan decreased when QMR increased. The QPS was initialised using Option 3, which comprised a Hadamard gate and rotation gate with small rotation angle. However, it was updated using Option 1, in which all possible states have the same probability amplitude (all multiplied by $1/\sqrt{2}$). As seen in Figure 2a, the makespan decreased when the rotation angle decreased. Based on the analysis, we conclude that the final QIGA worked best for solving J30 when QPS $=$ 20, IPO $=$ Option 3, UPO $=$ Option 1 and two-point quantum crossover at $\theta = \pi/18$ or smaller angles, and QMR $=$ 0.2 or larger.

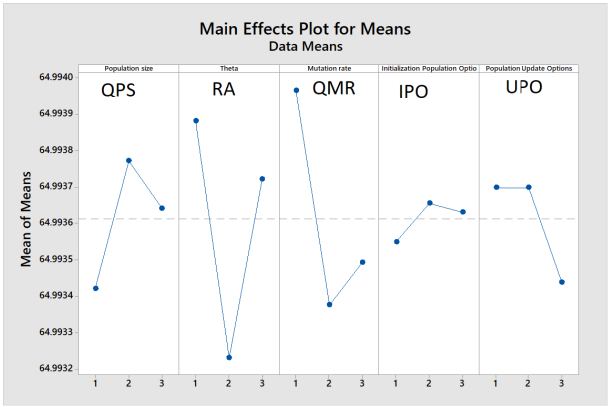
**J60:** The trends in the quantum parameters for the six problems selected from the activities in J60 are shown in Figure 2b. The optimal combinations of quantum parameters in the QIGA produced by the Taguchi method were: QPS $=$ 10; RA $\theta = \pi/6$; QMR $=$ 0.1; IPO $=$ Option 1; UPO $=$ Option 3; and single-point quantum crossover. Comparing these with the optimal combinations for J30, the QPS was smaller for J60. Besides, the rotation angle for the lowest makespan in J60, $\pi/6$, was larger than the $\pi/18$ and $\pi/36$ for J30. For J60, the QPS was initialised using Option 1 and updated using Option 3 with the higher rotation angle.

**J120:** The trends in the quantum parameters for the J120 are shown in Figure 2c. The optimal combinations of quantum parameters in the QIGA produced by the Taguchi method for solving J120 were: QPS $=$ 10; QMR $=$ 0.2; IPO $=$ Option 2; and UPO $=$ Option 1 or Option 2; and single-point quantum crossover. This means the quantum population was initialised using a Hadamard gate, then updated using either a Hadamard gate or Option 1 with the same probability amplitudes. We can also see in Figure 2c that the mean makespans increased when using the quantum rotation gate; the rotation gate and rotation angle are therefore not the preferred option in solving J120, either in the initialisation or updated population.

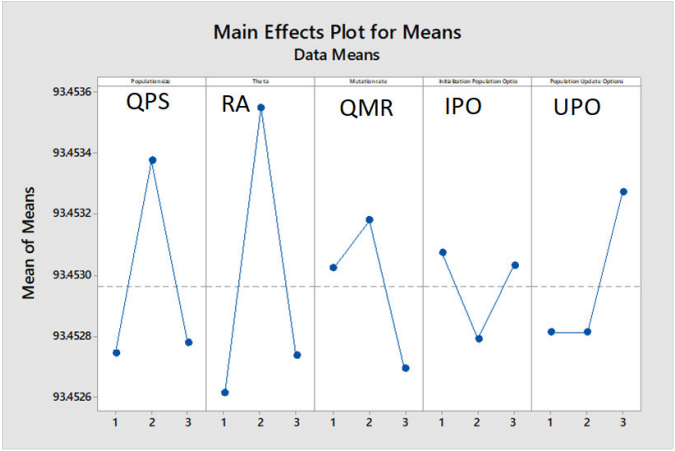
Comparing with other meta-heuristics, one of the main advantages of using quantum-inspired genetic algorithms for solving the RCPSP was to increase the population diversity




(a) Taguchi Method for J30.




(b) Taguchi Method for J60.




(c) Taguchi Method for J120.

**FIGURE 2.** **Trends in the quantum parameters in the QIGA for solving the RCPSP, tuned using the Taguchi Method and determined using the first problem in the instances 2, 13, 17, 29, 36 and 40 from the J30, J60 and J120.**

**TABLE 2.** **Average deviation values from optimal makespan obtained from QIGA for J30 with different values of the mutation rate. S:single-point crossover; D two-point crossover. The smallest deviations and times for solving each problem are highlighted in bold.**

| No. | Combinations | | | | | | Schedules | | | Time (s) |
|---|---|---|---|---|---|---|---|---|---|---|
| | QPS | RA | QMR | IPO | UPO | QC | 1000 | 5000 | 50,000 | |
| 1 | 20 | $\frac{\pi}{36}$ | 0.2 | Option 3 | Option 1 | S | 0.24 | 0.13 | **0.06** | 1.1702 |
| 2 | | | 0.2 | | | D | 0.22 | 0.14 | **0.06** | 1.1052 |
| 3 | | $\frac{\pi}{18}$ | 0.2 | | | S | 0.22 | 0.15 | 0.09 | 1.3960 |
| 4 | | | 0.2 | | | D | 0.21 | **0.12** | 0.07 | 1.1068 |
| 5 | | | 0.3 | | | S | 0.21 | 0.13 | **0.06** | **1.0018** |
| 6 | | | 0.3 | | | D | **0.20** | 0.14 | 0.08 | 1.2125 |
| 7 | | | 0.4 | | | S | 0.21 | 0.14 | **0.06** | 1.1665 |
| 8 | | | 0.4 | | | D | 0.23 | 0.14 | 0.08 | 1.1045 |

by using quantum gates and quantum superposition in initialisation and updated population and using the Not-quantum gate in the mutation. Using the same probability amplitude by using Eqs. (19) or (20) in the initialisation population or updated population might lead to lower diversity than when we use the rotation gate or Hadamard gate in Eq. (21). However, it was found that other algorithmic components, such as the mutation operator, might mitigate this issue. This was clear from the results obtained for J30 and J120 when we used Eqs. (19) or (20) with a MR of 0.3 (see Figure 2a and 2c).

## VIII. DISCUSSION

In this section, we discuss the influence of the quantum parameters such as the QPS, RA, QMR and QC on the average deviation values and the average computational times of the QIGA for the J30, J60 and J120 at three different fitness evaluations, 1000, 5000 and 50, 000, with 15 runs carried out.

**J30:** The average deviations of the optimal combination for QIGA at 1000, 5000 and 50, 000 schedules were 0.21, 0.12 and 0.07, respectively. The effects of varying QMR and RA on the average deviation values and the average computational times for J30 are listed in Table 2. The smallest

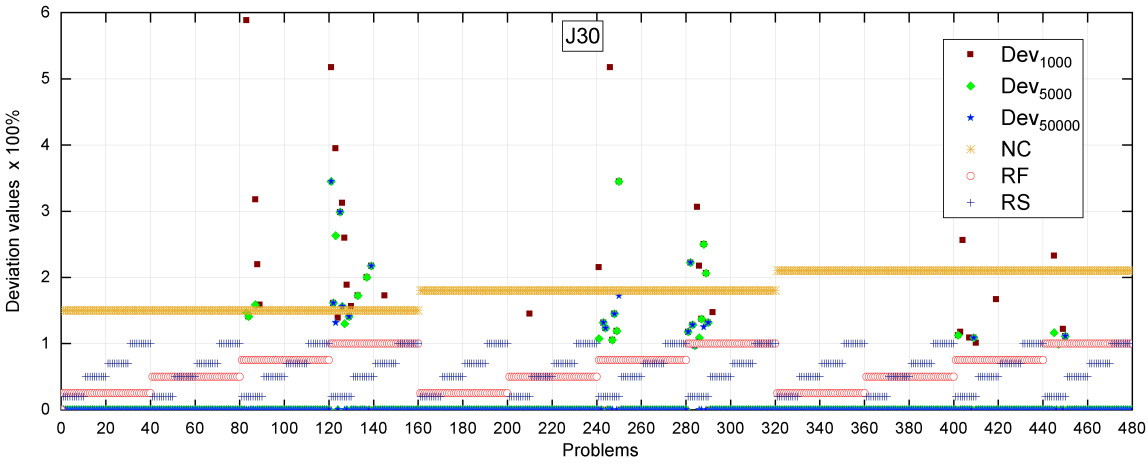



**FIGURE 3.** Deviation values (×100%) from the lower bound of each problem in the J30 at three different fitness evaluations, 1000, 5000 and 50, 000.

**TABLE 3.** Average deviation values from optimal makespan obtained from QIGA for J60 with different values of the mutation rate. S: single-point crossover; D two-point crossover. The smallest deviations and times for solving each problem are highlighted in bold.

| No. | Combinations | | | | | | Schedules | | | Time (s) |
|---|---|---|---|---|---|---|---|---|---|---|
| | QPS | RA | QMR | IPO | UPO | QC | 1000 | 5000 | 50,000 | |
| 1 | 2 | | 0.1 | | | S | 12.44 | 12.16 | 11.84 | 11.4716 |
| 2 | 2 | | 0.1 | | | D | 12.45 | 12.17 | 11.86 | 11.7808 |
| 3 | 10 | | 0.1 | | | S | **12.36** | **12.10** | **11.80** | **11.4268** |
| 4 | 10 | $\frac{\pi}{6}$ | 0.1 | Option 1 | Option 3 | D | 12.40 | 12.16 | 11.86 | 11.4982 |
| 5 | 20 | | 0.1 | | | S | 12.43 | 12.16 | 11.86 | 13.1482 |
| 6 | 20 | | 0.1 | | | D | 12.46 | 12.19 | 11.88 | 12.2454 |
| 7 | 50 | | 0.1 | | | S | 12.43 | 12.16 | 11.86 | 13.4587 |
| 8 | 50 | | 0.1 | | | D | **12.36** | 12.15 | 11.86 | 13.1883 |
| 9 | 100 | | 0.1 | | | S | 12.38 | 12.11 | **11.77** | 16.9392 |

deviations and times per problem are highlighted in bold. An increase in QMR and a decrease in the RA affected the quality of the solutions. The smallest deviation in the 50, 000 fitness evaluations was 0.06, with the average computational time per problem 1.0018 seconds when QMR was set to 0.3 and the single-point crossover operator used. The smallest deviation after 1000 fitness evaluations was 0.20, which was obtained with a QMR of 0.3 and a two-point-point crossover. The decrease in RA to the smaller $\theta = \pi/36$ positively affected the deviation values at 50, 000 schedules, for both the single- and two-point quantum crossovers, although the variant with RA $\theta = \pi/36$ was worst in 1000 schedules. QIGA obtained the smallest deviation (0.12) with 5000 fitness evaluations when QMR was set to 0.2 and a two-point crossover used.

The deviation values for the 480 problems of J30 using the QIGA are shown in Figure 3. QIGA was able to converge quickly to the best-known solutions for the majority of the test problems. For some problems (83, 87, 89, 145, 210, 240, 246, 292, 403, 407, 410, 419 and 444), it did not reach the best-known solution at 1000 fitness evaluations but was able to obtain it at the higher fitness evaluations. On the other hand, the problems for which QIGA, in solving J30, could not obtain the optimum value were 122, 125, 128, 139, 243, 244, 247, 281, 283, 290, 409 and 450. These problems had small resource strength, RS = 0.2, except for problem 139 for which RS = 0.5. In addition, these problems had resource factor RF = 1, except for problems 243, 244, 247 and 409, for which RF = 0.7. That means the deviations were high for those problems characterised by highly constrained resources and employing all resources for each job.

**J60:** The average deviations for J60, obtained using the optimal combination RA $\theta = \pi/6$, QMR = 0.1, IPO = Option 1 and UPO = Option 3, with a single-point crossover were 12.36, 12.10 and 11.80 at 1000, 5000 and 50, 000 fitness evaluations, respectively. The average computational time per problem was 11.4268 seconds. The results of varying QPS to 2, 10, 30, 50 and 100 are listed in Table 3. The smallest deviations and times for solving each problem are highlighted in bold. No large variation in the deviations was noted on increasing the QPS. However, we can note that the average deviation at QPS = 100, 50, 000 fitness evaluations and a single-point crossover decreased to 11.77, but the time per problem increased to 16.9392 seconds.

The deviation values for the 480 problems of J60 obtained using the QIGA are shown in Figure 4. Most of the problems with the small value RS = 0.2 had high in deviations, inconsistent with those for J30. The QIGA obtained good-quality

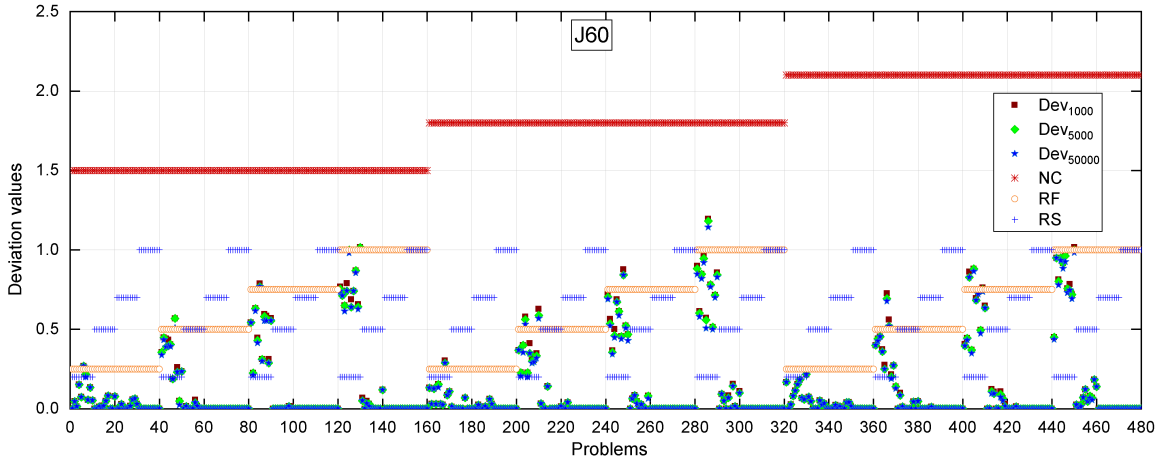



**FIGURE 4.** Deviation values from the lower bound of each problem in the J60 at three different fitness evaluations, 1000, 5000 and 50, 000.

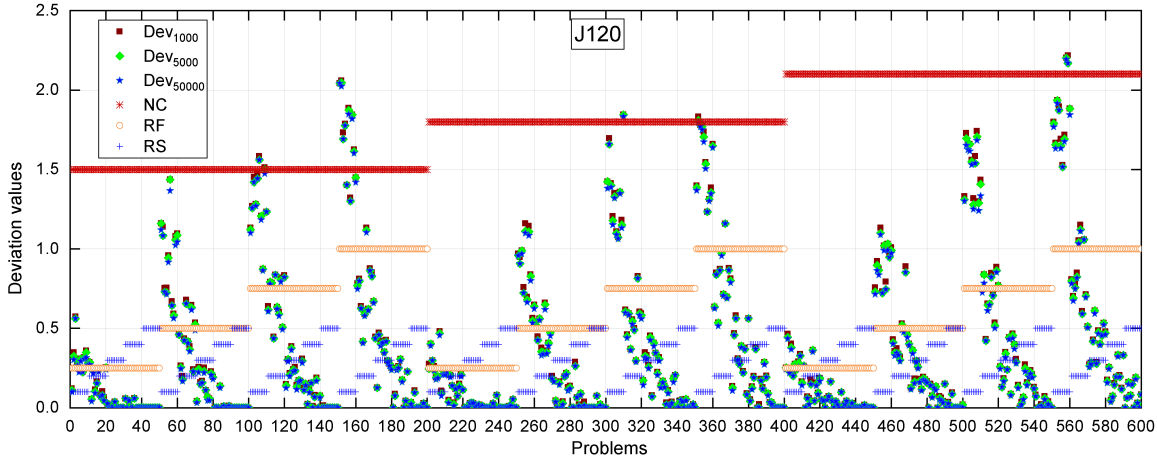



**FIGURE 5.** Deviation values from the lower bound of each problem in the J120 at three different fitness evaluations, 1000, 5000 and 50, 000.

**TABLE 4.** Average deviation values from optimal makespan obtained from QIGA for J120 with different values of the mutation rate. S: single-point crossover; D two-point crossover. The smallest deviations and times for solving each problem are highlighted in bold.

| No. | Combinations | | | | | | Schedules | | | Time (s) |
|---|---|---|---|---|---|---|---|---|---|---|
| | QPS | RA | QMR | IPO | UPO | QC | 1000 | 5000 | 50,000 | |
| 1 | 10 | Null | 0.2 | Option 2 | Option 1 | S | **36.30** | **36.02** | **35.08** | **138.7** |
| 2 | 10 | Null | 0.2 | Option 2 | Option 1 | D | 36.34 | 36.08 | 35.13 | 139.1 |

solutions but not optimal in the problems with small RS at 1000 schedules; the solutions improved slightly at 50, 000 schedules.

**J120:** The average deviations for J120, obtained using the optimal combination QPS = 10; QMR = 0.2; IPO = Option 2; and UPO = Option 1 or Option 2, with single-point quantum crossover were 36.30, 36.02 and 35.08 at 1000, 5000 and 50, 000 fitness evaluations, respectively. The average computational time per problem was 138.7 seconds.

The deviation values of the 600 problems for J120 obtained using the QIGA are shown in Figure 5. The quality of the solution depended on RS: deviations were large at smaller RS but improved at high values. Overall, for the problems J30, J60 and J120, the QIGA found difficulties in solving the problems with small RS and high RF.

## IX. COMPARISON WITH STATE-OF-THE-ART ALGORITHMS

In this section, we investigate the performance of the QIGA in attaining good solutions compared to the recent techniques of meta-heuristics and hybrid meta-heuristics. These techniques have been discussed in detail in the articles cited in Tables 5–7. It is worth noting that this paper aims to investigate the performance of the quantum-inspired genetic

**TABLE 5.** Comparison of the average deviation values for J30 obtained using the QIGA and some other genetic and hybrid algorithms. The smallest deviations are highlighted in bold.

| Algorithm | Schedule | | |
|---|---|---|---|
| | 1000 | 5000 | 50,000 |
| QIGA (**this study**) | 0.20 | 0.12 | 0.06 |
| TS-MODE (2020) [32] | 0.06 | 0.01 | 0.00 |
| Memetic algorithm (2020) [62] | – | 0.00 | 0.00 |
| Sequential(SS(FBI)) (2018) [63] | 0.10 | 0.02 | 0.00 |
| QIPSO (2017) [47] | 0.47 | – | — |
| COA (EA(GA(LS) + DEA(LS))) (2017) [9] | **0.04** | **0.00** | **0.00** |
| GA -part (ELDRA) (2017) [33] | 0.17 | 0.07 | 0.01 |
| H(SABC-PSO) (2016) [64] | 0.24 | 0.13 | – |
| H(EABC-PSO) (2016) [64] | 0.28 | 0.15 | – |
| MAOA (2015) [65] | 0.17 | 0.06 | 0.01 |
| Specialist(PSO-HH) (2014) [28] | 0.26 | 0.04 | 0.01 |
| PSO+TCO (2013) [66] | 0.27 | 0.14 | – |
| GRASP(FBI) + SS(FBI) (2013) [67] | 0.57 | 0.39 | 0.23 |
| HGA (2012) [68] | 1.66 | 0.78 | – |
| DPSO – FBI (2012) [69] | 0.36 | 0.14 | 0.05 |
| DEA(LS) (2011) [70] | 1.21 | 0.73 | 0.27 |
| GANS(LS) (2011) [71] | 1.83 | 1.27 | 0.71 |
| HGA (Neurogenetic)-FBI (2011) [72] | 0.13 | 0.10 | – |
| JPSO (2011) [73] | 0.29 | 0.14 | 0.04 |
| EA(LS) (2010) [74] | 0.38 | 0.17 | – |
| GA (2009) [75] | 0.33 | 0.19 | 0.12 |
| GAPSP (2009) [76] | 0.06 | 0.02 | 0.01 |
| GA (2008) [77] | 0.36 | 0.23 | – |
| HGA (2008) [78] | 0.27 | 0.06 | 0.02 |

**TABLE 6.** Comparison between the average deviation values for J60 obtained using the QIGA and some other genetic and hybrid algorithms. The smallest deviations are highlighted in bold.

| Algorithm | Schedule | | |
|---|---|---|---|
| | 1000 | 5000 | 50,000 |
| QIGA (**this study**) | 12.36 | 12.10 | 11.77 |
| TS-MODE (2020) [32] | 11.90 | 11.21 | 10.629 |
| Memetic algorithm (2020) [62] | – | 10.72 | 10.55 |
| Sequential(SS(FBI)) (2018) [63] | 11.38 | 10.93 | 10.58 |
| COA (EA(GA(LS) + DEA(LS))) (2017) [9] | **11.30** | 10.77 | 10.58 |
| GA -part (ELDRA) (2017) [33] | 11.61 | 11.08 | 10.71 |
| SABC-PSO (2016) [64] | 12.14 | 11.90 | – |
| EABC-PSO (2016) [64] | 12.53 | 11.98 | – |
| MAOA (2015) [65] | 11.67 | 10.84 | 10.64 |
| Specialist(PSO-HH) (2014) [28] | 11.74 | 11.13 | 10.68 |
| PSO+TCO (2013) [66] | 12.53 | 11.03 | – |
| GRASP-FBI-SS (2013) [67] | 12.88 | 12.42 | 11.96 |
| PSO(FBI) (2013) [79] | 12.02 | 11.33 | 10.79 |
| SFMA(LS) (2013) [80] | 14.58 | 13.06 | – |
| DPSO – FBI (2012) [69] | 11.56 | 11.00 | 10.68 |
| GA(FBI) (2011) [81] | 13.03 | 12.73 | 12.23 |
| GA-priority rule (2011) [73] | 13.30 | 12.74 | 12.26 |
| GANS(LS) (2011) [71] | 11.35 | **10.53** | **10.52** |
| JPSO (2011) [73] | 12.03 | 11.43 | 11.00 |
| Hybrid GA (Neurogenetic)-FBI (2011) [72] | 11.51 | 11.29 | – |
| BCO (2010) [82] | 12.75 | 11.48 | 11.18 |
| EA(LS) (2010) [74] | 12.60 | 11.67 | – |
| GAPSP (2009) [76] | 11.72 | 11.04 | 10.67 |
| Hybrid GA (2008) [78] | 11.56 | 11.10 | 10.7 |

algorithm in solving RCPSPs, not to design new versions of it. We compare the average deviation values from the QIGA and the other techniques. This comparison was done at three fitness evaluations levels, 1000, 5000 and 50, 000. A comparative summary for J30, J60 and J120, is shown in Tables 5–7. The results for the other algorithms are taken from the original articles, except for the GA-priority rule, for which the results were taken from [73].

**TABLE 7.** Comparison between the average deviation values for J120 obtained using the QIGA and some other genetic and hybrid algorithms. The smallest deviations are highlighted in bold.

| Algorithm | Schedule | | |
|---|---|---|---|
| | 1000 | 5000 | 50,000 |
| QIGA (**this study**) | 36.30 | 36.02 | 35.08 |
| TS-MODE (2020) [32] | 34.4 | 32.86 | 30.59 |
| Memetic Algorithm (2020) [62] | – | 32.76 | 31.12 |
| Sequential(SS(FBI)) (2018) [63] | 34.01 | **32.52** | 31.16 |
| COA (EA(GA(LS) + DEA(LS))) (2017) [9] | 34.04 | 32.90 | 31.22 |
| GA-part (ELDRA) (2017) [33] | 34.59 | 33.36 | 31.81 |
| SABC-PSO (2016) [64] | 36.15 | 35.28 | – |
| EABC-PSO (2016) [64] | 37.28 | 36.24 | – |
| MAOA (2015) [65] | **33.8** | 32.64 | **31.02** |
| DEA (2015) [83] | 35.15 | 33.67 | 33.14 |
| Specialist(PSO-HH) (2014) [28] | 35.20 | 32.59 | 31.2 |
| GRASP-FBI-SS (2013) [67] | 38.16 | 37.30 | 36.32 |
| PSO+TCO (2013) [66] | 34.87 | 34.01 | – |
| PSO(FBI) (2013) [79] | 36.77 | 35.16 | 32.89 |
| DPSO – FBI (2012) [69] | 34.95 | 33.34 | 32.19 |
| BCO-GA(FBI) (2012) [84] | 35.86 | 34.37 | 33.14 |
| GA-priority rule (2011) [73] | 39.93 | 38.49 | 36.51 |
| JPSO (2011) [73] | 35.71 | 33.88 | 32.89 |
| Hybrid GA (Neurogenetic)-FBI (2011) [72] | 34.65 | 34.15 | – |
| BCO (2010) [82] | 36.29 | 34.18 | 33.69 |
| EA(LS) (2010) [74] | – | 34.39 | – |
| GA(LS) (2009) [76] | 35.87 | 33.03 | 31.44 |
| Hybrid GA (2008) [78] | 34.07 | 32.54 | 31.24 |
| GA – forw.–backw. (2001) [85] | 39.36 | 35.57 | – |

**TABLE 8.** Comparison between the mean rank of the QIGA and the state-of-the-art algorithms obtained using the Friedman test. The smallest mean rank is highlighted in bold.

| Algorithm | Mean Rank |
|---|---|
| QIGA | 6.89 |
| TS-MODE | **3.33** |
| Sequential(SS(FBI)) | **2.28** |
| COA | **2.28** |
| GA-part | 5.06 |
| MAOA | 3.39 |
| PSO-HH | 4.78 |
| GRASP(FBI) + SS(FBI) | 8.00 |
| GA-priority rule | 9.00 |

For J30, the quality of the solutions obtained by the QIGA was competitive with many of the other algorithms and better than some GAs, HGAs and hybrid meta-heuristics, but the QIGA was outperformed by some other hybrid meta-heuristics, one of which, COA [9], is still ranked first in quality for solving J30. COA was formed from two evolutionary algorithms (a genetic algorithm and differential evolution). In addition, the quality of the solution obtained by the QIGA for J30 was better than the average deviation value obtained by QIPSO. For J60 and J120, the average deviations obtained by the QIGA were again competitive with the other algorithms and better than some.

To evaluate the performance of the proposed QIGA in solving RCPSP, we utilised the Friedman test to calculate the relative ranks of the state-of-the-art algorithms [86]. This test is a ranking procedure based on a non-parametric test. The relative ranks of the state-of-the-art algorithms and the QIGA for solving the RCPSP obtained using the Friedman test are shown in Table 8. As per the Friedman test,

Sequential(SS(FBI)) [63], COA [9], and TS-MODE [32] were the first three (least is the best, as this is a minimisation problem).

Then, we performed a Wilcoxon signed test [86] to see the difference between the rank of the QIGA and the rank of those first three algorithms. The P-value of the Wilcoxon signed test of the three algorithms compared with the QIGA was found to be 0.008, which is less than 0.05. Thus, those first three competing algorithms significantly outperformed the proposed QIGA. Despite such statistically significant differences, the merit of this paper and thus the proposed algorithm is undeniable. Although the proposed QIGA could not outperform many competing heuristic and meta-heuristic approaches, it will unarguably open-up a new avenue for further research on stand-alone and/or hybrid quantum-inspired meta-heuristics to deal with complex and larger combinatorial optimisation problems, such as RCPSP, the Travelling-Salesperson Problem and the job shop scheduling problem.

Therefore to summarise, QIGA was outperformed by some hybrid meta-heuristics but was competitive with many of the other algorithms and better than some GAs, HGAs and hybrid meta-heuristics. However, there is still room for further improvements, which is left for future work.

## X. CONCLUSION AND FUTURE WORK

Although many heuristics, meta-heuristic and hybrid meta-heuristic algorithms have been developed for solving the RCPSP, no single algorithm has proved to be the best for all problems, and new algorithms are still being proposed to improve the quality of the solution. In this paper, we investigated the possibility of adapting the quantum-inspired genetic algorithm QIGA) to solve RCPSPs. The algorithm utilised some concepts from evolutionary computation and quantum mechanics (i.e., qubits, quantum gates and quantum superposition).

We analysed the effects of different quantum parameters in the QIGA, such as quantum population size, quantum mutation rate, and single-point and two-point quantum crossover, initialised the quantum population using quantum rotation gates with different rotation angles and the Hadamard gate. In addition, a local search was used to accelerate the convergence rate and to enhance performance. PSPLIB benchmark instances of the RCPSP using J30, J60 and J120 were used to evaluate QIGA. We determined the best combination of the quantum parameters to use in the QIGA. Furthermore, we determined the effects of its operators and parameters.

QIGA was found competitive with other algorithms and better than some, but still not the best. This opens new research directions for quantum-based algorithms for RCPSPs, for instance, quantum interference crossover, evolving the quantum gates using recent and advanced evolutionary algorithms. Adopting the proposed framework with other population-based approaches is a possible future direction. The hybridisation of multiple meta-heuristics with the quantum-inspired approach is another interesting future direction.

Also in the future work, we can use some other computational intelligence algorithms in solving the RCPSP, did not use before in solving the RCPSP, such as monarch butterfly optimization (MBO), earthworm optimization algorithm (EWA), elephant herding optimization (EHO), moth search (MS) algorithm, Slime mould algorithm (SMA), and Harris hawks optimization (HHO).

## REFERENCES

[1] J. Blazewicz, J. K. Lenstra, and A. H. G. R. Kan, "Scheduling subject to resource constraints: Classification and complexity," *Discrete Appl. Math.*, vol. 5, no. 1, pp. 11–24, Jan. 1983.
[2] R. Čorić, M. Dumić, and D. Jakobović, "Complexity comparison of integer programming and genetic algorithms for resource constrained scheduling problems," in *Proc. 40th Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, May 2017, pp. 1182–1188.
[3] A. Jalilvand, S. Khanmohammadi, and F. Shabaninia, "Scheduling of sequence-dependant jobs on parallel multiprocessor systems using a branch and bound-based Petri net," in *Proc. IEEE Symp. Emerg. Technol.*, Sep. 2005, pp. 334–339.
[4] R. Klein, "Bidirectional planning: Improving priority rule-based heuristics for scheduling resource-constrained projects," *Eur. J. Oper. Res.*, vol. 127, no. 3, pp. 619–638, Dec. 2000.
[5] R. Pellerin, N. Perrier, and F. Berthaut, "A survey of hybrid metaheuristics for the resource-constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 280, no. 2, pp. 395–416, Jan. 2020.
[6] R. K. Chakrabortty, R. A. Sarker, and D. L. Essam, "Resource constrained project scheduling with uncertain activity durations," *Comput. Ind. Eng.*, vol. 112, pp. 537–550, Oct. 2017.
[7] S. B. Issa and Y. Tu, "A survey in the resource-constrained project and multi-project scheduling problems," *J. Project Manage.*, pp. 117–138, 2020.
[8] Y. Liu, R. Li, and H. Liu, "Heuristic optimization for robust resource-constrained flexible project scheduling problem," *IEEE Access*, vol. 8, pp. 142269–142281, 2020.
[9] S. Elsayed, R. Sarker, T. Ray, and C. C. Coello, "Consolidated optimization algorithm for resource-constrained project scheduling problems," *Inf. Sci.*, vols. 418–419, pp. 346–362, Dec. 2017.
[10] G. Zhang, "Quantum-inspired evolutionary algorithms: A survey and empirical study," *J. Heuristics*, vol. 17, no. 3, pp. 303–351, Jun. 2011.
[11] K.-H. Han and J.-H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *IEEE Trans. Evol. Comput.*, vol. 6, no. 6, pp. 580–593, Dec. 2002.
[12] A. Narayanan and M. Moore, "Quantum-inspired genetic algorithms," in *Proc. IEEE Int. Conf. Evol. Comput.*, May 1996, pp. 61–66.
[13] K.-H. Han and J.-H. Kim, "Genetic quantum algorithm and its application to combinatorial optimization problem," in *Proc. Congr. Evol. Comput.*, vol. 2, Jul. 2000, pp. 1354–1360.
[14] K.-H. Han, K.-H. Park, C.-H. Lee, and J.-H. Kim, "Parallel quantum-inspired genetic algorithm for combinatorial optimization problem," in *Proc. Congr. Evol. Comput.*, vol. 2, May 2001, pp. 1422–1429.
[15] F. S. Alfares and I. I. Esat, "Real-coded quantum inspired evolution algorithm applied to engineering optimization problems," in *Proc. 2nd Int. Symp. Leveraging Appl. Formal Methods, Verification Validation (isola)*, Nov. 2006, pp. 169–176.
[16] A. V. A. da Cruz, M. M. B. R. Vellasco, and M. C. Pacheco, "Quantum-inspired evolutionary algorithm for numerical optimization," in *Hybrid Evolutionary Algorithms*. Berlin, Germany: Springer, 2007, pp. 19–37, doi: 10.1007/978-3-540-73297-6_2.
[17] O. H. Montiel Ross, "A review of quantum-inspired metaheuristics: Going from classical computers to real quantum computers," *IEEE Access*, vol. 8, pp. 814–838, 2020.
[18] S. Koppaka and A. R. Hota, "Superior exploration-exploitation balance with quantum-inspired Hadamard walks," in *Proc. 12th Annu. Conf. Comp Genetic Evol. Comput. (GECCO)*, 2010, pp. 2093–2094.
[19] H. Wang, J. Liu, J. Zhi, and C. Fu, "The improvement of quantum genetic algorithm and its application on function optimization," *Math. Problems Eng.*, vol. 2013, May 2013, Art. no. 730749.

[20] K. Pradhan, S. Basu, K. Thakur, S. Maity, and M. Maiti, "Imprecise modified solid green traveling purchaser problem for substitute items using quantum-inspired genetic algorithm," *Comput. Ind. Eng.*, vol. 147, Sep. 2020, Art. no. 106578.

[21] Z. A. E. M. Dahi, C. Mezioud, and A. Draa, "A quantum-inspired genetic algorithm for solving the antenna positioning problem," *Swarm Evol. Comput.*, vol. 31, pp. 24–63, Dec. 2016.

[22] 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 Comput. Sci.*, vol. 131, pp. 591–599, 2018.

[23] J. Gu, X. Gu, and M. Gu, "A novel parallel quantum genetic algorithm for stochastic job shop scheduling," *J. Math. Anal. Appl.*, vol. 355, no. 1, pp. 63–81, Jul. 2009.

[24] J. Suo, L. Gu, Y. Pan, S. Yang, and X. Hu, "Quantum inspired genetic algorithm for double digest problem," *IEEE Access*, vol. 8, pp. 72910–72916, 2020.

[25] R. Kolisch and A. Sprecher, "PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program," *Eur. J. Oper. Res.*, vol. 96, no. 1, pp. 205–216, 1997.

[26] R. Kolisch, C. Schwindt, and A. Sprecher, "Benchmark instances for project scheduling problems," in *Project Scheduling*. Boston, MA, USA: Springer, 1999, pp. 197–212, doi: 10.1007/978-1-4615-5533-9_9.

[27] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. surveys (CSUR)*, vol. 35, no. 3, pp. 268–308, 2003.

[28] G. Koulinas, L. Kotsikas, and K. Anagnostopoulos, "A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem," *Inf. Sci.*, vol. 277, pp. 680–693, Sep. 2014.

[29] P. B. Myszkowski, M. E. Skowroński, Ł. P. Olech, and K. Oślizło, "Hybrid ant colony optimization in solving multi-skill resource-constrained project scheduling problem," *Soft Comput.*, vol. 19, no. 12, pp. 3599–3619, Dec. 2015.

[30] T. Servranckx and M. Vanhoucke, "A tabu search procedure for the resource-constrained project scheduling problem with alternative subgraphs," *Eur. J. Oper. Res.*, vol. 273, no. 3, pp. 841–860, Mar. 2019.

[31] M. Palpant, C. Artigues, and P. Michelon, "LSSPER: Solving the resource-constrained project scheduling problem with large neighbourhood search," *Ann. Operations Res.*, vol. 131, nos. 1–4, pp. 237–257, Oct. 2004.

[32] K. M. Sallam, R. K. Chakrabortty, and M. J. Ryan, "A two-stage multi-operator differential evolution algorithm for solving resource constrained project scheduling problems," *Future Gener. Comput. Syst.*, vol. 108, pp. 432–444, Jul. 2020.

[33] R. Zamani, "An evolutionary implicit enumeration procedure for solving the resource-constrained project scheduling problem," *Int. Trans. Oper. Res.*, vol. 24, no. 6, pp. 1525–1547, Nov. 2017.

[34] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by turing machines," *J. Stat. Phys.*, vol. 22, no. 5, pp. 563–591, May 1980.

[35] R. P. Feynman, "Simulating physics with computers," *Int. J. Theor. Phys.*, vol. 21, nos. 6–7, pp. 467–488, Jun. 1982.

[36] R. P. Feynman, "Quantum mechanical computers," *Found. Phys.*, vol. 16, no. 6, pp. 507–531, 1986.

[37] D. Deutsch, "Quantum theory, the Church–Turing principle and the universal quantum computer," *Proc. Roy. Soc. London. A. Math. Phys. Sci.*, vol. 400, no. 1818, pp. 97–117, 1985.

[38] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, 1994, pp. 124–134.

[39] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annu. ACM Symp. Theory Comput. (STOC)*, 1996, pp. 212–219.

[40] V. Vedral and M. B. Plenio, "Basics of quantum computation," *Prog. Quantum Electron.*, vol. 22, no. 1, pp. 1–39, Jan. 1998.

[41] E. Rieffel and W. Polak, "An introduction to quantum computing for non-physicists," *ACM Comput. Surv.*, vol. 32, no. 3, pp. 300–335, Sep. 2000.

[42] R. Lahoz-Beltra, "Quantum genetic algorithms for computer scientists," *Computers*, vol. 5, no. 4, p. 24, Oct. 2016.

[43] O. Montiel, Y. Rubio, C. Olvera, and A. Rivera, "Quantum-inspired acromyrmex evolutionary algorithm," *Sci. Rep.*, vol. 9, no. 1, Dec. 2019, Art. no. 012181.

[44] Y. Wang, X.-Y. Feng, Y.-X. Huang, D.-B. Pu, W.-G. Zhou, Y.-C. Liang, and C.-G. Zhou, "A novel quantum swarm evolutionary algorithm and its applications," *Neurocomputing*, vol. 70, nos. 4–6, pp. 633–640, Jan. 2007.

[45] D. Zouache, F. Nouioua, and A. Moussaoui, "Quantum-inspired firefly algorithm with particle swarm optimization for discrete optimization problems," *Soft Comput.*, vol. 20, no. 7, pp. 2781–2799, Jul. 2016.

[46] S. Tu, O. U. Rehman, S. U. Rehman, S. Ullah, M. Waqas, and R. Zhu, "A novel quantum inspired particle swarm optimization algorithm for electromagnetic applications," *IEEE Access*, vol. 8, pp. 21909–21916, 2020.

[47] R. Sharma, R. Bangroo, M. Kumar, and N. Kumar, "A model for resource constraint project scheduling problem using quantum inspired PSO," in *Proc. Int. Conf. Next Gener. Comput. Technol.* Singapore: Springer, 2017, pp. 75–87, doi: 10.1007/978-981-10-8657-1_6.

[48] A. Draa, S. Meshoul, H. Talbi, and M. Batouche, "A quantum-inspired differential evolution algorithm for solving the N-queens problem," *Neural Netw.*, vol. 1, no. 2, pp. 21–27, 2011.

[49] W. Deng, H. Liu, J. Xu, H. Zhao, and Y. Song, "An improved quantum-inspired differential evolution algorithm for deep belief network," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 10, pp. 7319–7327, Oct. 2020.

[50] W. Deng, J. Xu, X.-Z. Gao, and H. Zhao, "An enhanced MSIQDE algorithm with novel multiple strategies for global optimization problems," *IEEE Trans. Syst., Man, Cybern. Syst.*, early access, Nov. 4, 2020, doi: 10.1109/TSMC.2020.3030792.

[51] W. Deng, J. Xu, H. Zhao, and Y. Song, "A novel gate resource allocation method using improved PSO-based QEA," *IEEE Trans. Intell. Transp. Syst.*, early access, Oct. 1, 2020, doi: 10.1109/TITS.2020.3025796.

[52] L. Wang, F. Tang, and H. Wu, "Hybrid genetic algorithm based on quantum computing for numerical optimization and parameter estimation," *Appl. Math. Comput.*, vol. 171, no. 2, pp. 1141–1156, Dec. 2005.

[53] B.-B. Li and L. Wang, "A hybrid quantum-inspired genetic algorithm for multiobjective flow shop scheduling," *IEEE Trans. Syst., Man Cybern., B (Cybern.)*, vol. 37, no. 3, pp. 576–591, Jun. 2007.

[54] 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," *Appl. Soft Comput.*, vol. 53, pp. 296–307, Apr. 2017.

[55] S. Ding, Z. Jin, and Q. Yang, "Evolving quantum circuits at the gate level with a hybrid quantum-inspired evolutionary algorithm," *Soft Comput.*, vol. 12, no. 11, pp. 1059–1072, Sep. 2008.

[56] Q. Yang and S. Ding, "Methodology and case study of hybrid quantum-inspired evolutionary algorithm for numerical optimization," in *Proc. 3rd Int. Conf. Natural Comput. (ICNC)*, vol. 5, Aug. 2007, pp. 634–638.

[57] A. A. B. Pritsker, L. J. Waiters, and P. M. Wolfe, "Multiproject scheduling with limited resources: A zero-one programming approach," *Manage. Sci.*, vol. 16, no. 1, pp. 93–108, Sep. 1969.

[58] S. Hartmann and D. Briskorn, "A survey of variants and extensions of the resource-constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 207, no. 1, pp. 1–14, Nov. 2010.

[59] B. Schumacher, "Quantum coding," *Phys. Rev. A, Gen. Phys.*, vol. 51, no. 4, p. 2738, 1995.

[60] C.-R. Wie, "Bloch sphere model for two-qubit pure states," 2014, *arXiv:1403.8069*. [Online]. Available: http://arxiv.org/abs/1403.8069

[61] I. Ali, "Evolutionary algorithms for resource constrained project scheduling," Univ. New South Wales, Canberra, NSW, Australia, Tech. Rep., 2016, p. 63. [Online]. Available: http://handle.unsw.edu.au/1959.4/56420, doi: 10.26190/5dc4f1a1f31c9.

[62] H. F. Rahman, R. K. Chakrabortty, and M. J. Ryan, "Memetic algorithm for solving resource constrained project scheduling problems," *Autom. Construction*, vol. 111, Mar. 2020, Art. no. 103052.

[63] F. Berthaut, R. Pellerin, A. Hajji, and N. Perrier, "A path relinking-based scatter search for the resource-constrained project scheduling problem," *Int. J. Project Organisation Manage.*, vol. 10, no. 1, pp. 1–36, 2018.

[64] Q. Jia and Y. Guo, "Hybridization of ABC and PSO algorithms for improved solutions of RCPSP," *J. Chin. Inst. Engineers*, vol. 39, no. 6, pp. 727–734, Aug. 2016.

[65] X.-L. Zheng and L. Wang, "A multi-agent optimization algorithm for resource constrained project scheduling problem," *Expert Syst. Appl.*, vol. 42, nos. 15–16, pp. 6039–6049, Sep. 2015.

[66] V. Zeighami, R. Akbari, and K. Ziarati, "Development of a method based on particle swarm optimization to solve resource constrained project scheduling problem," *Scientia Iranica*, vol. 20, no. 6, pp. 2123–2137, 2013.

[67] J. C. Rivera, L. F. Moreno, V. F. J. Díaz, S. G. E. Peña, "A hybrid heuristic algorithm for solving the resource constrained project scheduling problem (RCPSP)," *Revista EIA*, vol. 20, pp. 87–100, Dec. 2013.

[68] J.-L. Kim, "Hybrid genetic algorithm parameter effects for optimization of construction resource allocation problem," in *Proc. Construct. Res. Congr.*, May 2012, pp. 1560–1569.
[69] J. Czogalla and A. Fink, "A hybrid particle swarm algorithm for resource-constrained project scheduling," in *Hybrid Algorithms for Service, Computing and Manufacturing Systems: Routing and Scheduling Solutions*. Hershey, PA, USA: IGI Global, 2012, pp. 137–157.
[70] L. Wang, Y. Xu, and C. Fang, "A hybrid algorithm based on simplex search and differential evolution for resource-constrained project scheduling problem," in *Proc. Int. Conf. Intell. Comput.* Berlin, Germany: Springer, 2011, pp. 568–575, doi: 10.1007/978-3-642-24728-6_77.
[71] S. Proon and M. Jin, "A genetic algorithm with neighborhood search for the resource-constrained project scheduling problem," *Nav. Res. Logistics (NRL)*, vol. 58, no. 2, pp. 73–82, Mar. 2011.
[72] A. Agarwal, S. Colak, and S. Erenguc, "A neurogenetic approach for the resource-constrained project scheduling problem," *Comput. Oper. Res.*, vol. 38, no. 1, pp. 44–50, Jan. 2011.
[73] R.-M. Chen, "Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem," *Expert Syst. Appl.*, vol. 38, no. 6, pp. 7102–7111, Jun. 2011.
[74] Z.-J. Chen and C.-C. Chyu, "An evolutionary algorithm with multi-local search for the resource-constrained project scheduling problem," *Intell. Inf. Manage.*, vol. 2, no. 3, pp. 220–226, 2010, doi: 10.4236/iim.2012.23026.
[75] M. Deiranlou and F. Jolai, "A new efficient genetic algorithm for project scheduling under resource constrains," *World Appl. Sci. J.*, vol. 7, no. 8, pp. 987–997, 2009.
[76] J. J. M. Mendes, J. F. Gonçalves, and M. G. C. Resende, "A random key based genetic algorithm for the resource constrained project scheduling problem," *Comput. Oper. Res.*, vol. 36, no. 1, pp. 92–109, Jan. 2009.
[77] H. Zhang, H. Xu, and W. Peng, "A genetic algorithm for solving RCPSP," in *Proc. Int. Symp. Comput. Sci. Comput. Technol.*, vol. 2, 2008, pp. 246–249.
[78] V. Valls, F. Ballestín, and S. Quintanilla, "A hybrid genetic algorithm for the resource-constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 185, no. 2, pp. 495–508, Mar. 2008.
[79] M. M. Nasiri, "A pseudo particle swarm optimization for the RCPSP," *Int. J. Adv. Manuf. Technol.*, vol. 65, nos. 5–8, pp. 909–918, Mar. 2013.
[80] L. Wang and J. Liu, "A scale-free based memetic algorithm for resource-constrained project scheduling problems," in *Proc. Int. Conf. Intell. Data Eng. Automated Learn.* Berlin, Germany: Springer, 2013, pp. 202–209, doi: 10.1007/978-3-642-41278-3_25.
[81] F. Chen, W. Ling, and M. Chundi, "Disjunctive arc based genetic algorithm for the resource-constrained project scheduling problem," *Chin. Sci. Papers Online*, vol. 65, nos. 1–8, pp. 1–8, 2011.
[82] Y.-J. Shi, F.-Z. Qu, W. Chen, and B. Li, "An artificial bee colony with random key for resource-constrained project scheduling," in *Life System Modeling and Intelligent Computing*. Berlin, Germany: Springer, 2010, pp. 148–157, doi: 10.1007/978-3-642-15597-0_17.
[83] N. Rahmani, V. Zeighami, and R. Akbari, "A study on the performance of differential search algorithm for single mode resource constrained project scheduling problem," *Decis. Sci. Lett.*, vol. 4, no. 4, pp. 537–550, 2015.
[84] V. Zeighami, R. Akbari, I. Akbari, and Y. Biletskiy, "An ABC-Genetic method to solve resource constrained project scheduling problem," *Artif. Intell. Res. (AIR) J., SCIEDU, Canada*, vol. 1, no. 2, pp. 185–197, 2012.
[85] J. Alcaraz and C. Maroto, "A robust genetic algorithm for resource allocation in project scheduling," *Ann. Oper. Res.*, vol. 102, nos. 1–4, pp. 83–109, 2001.
[86] D. W. Zimmerman and B. D. Zumbo, "Relative power of the Wilcoxon test, the Friedman test, and repeated-measures ANOVA on ranks," *J. Exp. Educ.*, vol. 62, no. 1, pp. 75–86, Jul. 1993.

**HATEM M. H. SAAD** received the B.Sc. and M.Sc. degrees in physics from the Department of Physics, Faculty of Science, AL-Azhar University, Cairo, Egypt, and the Ph.D. degree in physics from the School of Science, University of New South Wales (UNSW) at Canberra, Australia, in 2020. He is currently a Research Associate with the Capability Systems Centre (CSC), School of Engineering and Information Technology (SEIT), UNSW Canberra. His research interests include quantum-inspired evolutionary algorithms, material science, solid-state physics, magnetic and electric properties, and magnetic structure.

**RIPON K. CHAKRABORTTY** (Member, IEEE) received the B.Sc. and M.Sc. degrees in industrial and production engineering and the Ph.D. degree from the Bangladesh University of Engineering and Technology, in 2009, 2013, and 2017, respectively. He is currently a Lecturer of System Engineering and Project Management and also the program coordinators for Master of Decision Analytics and Master of Engineering Science with the School of Engineering and Information Technology, University of New South Wales (UNSW) at Canberra, Australia. He is also the Group Leader of Cross-Disciplinary Optimisation under Capability Context Research Team. He has written two book chapters and over 90 technical journal and conference papers. His research interests include wide range of topics in operations research, project management, supply chain management, artificial intelligence, cyber-physical systems, and information systems management. His research program has been funded by many organisations, such as the Department of Defence-Commonwealth Government, Australia.

**SABER ELSAYED** (Member, IEEE) received the Ph.D. degree in computer science from the University of New South Wales (UNSW) at Canberra, Australia, in 2012. He is currently a Senior Lecturer with the School of Engineering and Information Technology (SEIT), UNSW Canberra. His research interests include single and multi-objective optimisation, static and dynamic optimisation, large-scale optimisation, project scheduling, swarm guidance, and cyber-security using computational intelligence. He was the winner of different competitions organised in top evolutionary computation conferences. He was a member of the organising committee of many international conferences. He is also a member of the IEEE Computational Intelligence Society Summer School subcommittees. From 2019 to 2020, he was the Chair of the IEEE Computational Intelligence Society (ACT Chapter).

**MICHAEL J. RYAN** (Senior Member, IEEE) received the bachelor's, master's, and Ph.D. degrees in engineering. He is currently a Professor and the Director of the Capability Systems Centre (CSC), University of New South Wales (UNSW) at Canberra. He has over 35 years of experience in communications engineering, systems engineering, project management, and management. Since joining UNSW Canberra, he has lectured in a range of subjects, including communications and information systems, systems engineering, requirements engineering, and project management, and he regularly consults in those fields. He is the author/coauthor of 13 books, three book chapters, and over a 300 refereed journal and conference papers. He is also a Fellow of Engineers Australia (FIEAust), a Chartered Professional Engineer (CPEng) of electrical and ITEE colleges, a Fellow of the International Council on Systems Engineering (INCOSE), and a Fellow of the Institute of Managers and Leaders (FIML).