Quantum-Inspired Estimation Of Distribution Algorithm To Solve The Travelling Salesman Problem

Vicente P. Soloviev
Computational Intelligence Group
Universidad Politécnica de Madrid
Madrid, Spain
vicente.perez.soloviev@fi.upm.es

Concha Bielza
Computational Intelligence Group
Universidad Politécnica de Madrid
Madrid, Spain
mcbielza@fi.upm.es

Pedro Larrañaga
Computational Intelligence Group
Universidad Politécnica de Madrid
Madrid, Spain
pedro.larranaga@fi.upm.es

***Abstract*—A novel Quantum-Inspired Estimation of Distribution Algorithm (QIEDA) is proposed to solve the Travelling Salesman Problem (TSP). The QIEDA uses a modified version of the W state quantum circuits to sample new solutions during the algorithm runtime. The algorithm behaviour is compared with other state-of-the-art population-based algorithms. QIEDA convergence is faster than other algorithms, and the obtained solutions improve as the size of the problem increases. Moreover, we show that quantum noise enhances the search of an optimal solution. Because quantum computers differ from each other, partly due to the topology that distributes the qubits, the computational cost of executing the QIEDA in different topologies is analyzed and an ideal topology is proposed for the TSP solved with the QIEDA.**

***Index Terms*—quantum computing, quantum machine learning, travelling salesman problem, estimation of distribution algorithm**

I. Introduction

Developing efficient and effective heuristic approaches for solving optimization problems has become the focus of metaheuristic research [1]. Deciding which algorithm is better to solve a problem is guided by three factors: (i) the quality of the obtained solutions, (ii) the computational cost, which is the amount of resources dedicated to executing the algorithm, and (iii) the running time, which is the time that elapses during the execution of the algorithm.

TSP is a combinatorial optimization problem widely studied in the literature in different research areas. In the Quantum Computing (QC) area [2], [3] several techniques such as adiabatic QC [4], and quantum approximate optimization algorithms [5] are used to solve it. In the heuristic optimization and the machine learning area, several metaheuristics such as Estimation of Distribution Algorithms (EDAs) [6], genetic algorithms [7], deep learning [8] and reinforcement learning [9] have been applied to approach the TSP.

This work has been partially supported by the Spanish Ministry of Science and Innovation through the PID2019-109247GB-I00 and RTC2019-006871-7 projects, and by Repsol through the "Batch Reinforcement Learning" project.

The TSP is a well known NP-hard problem. [10]. The problem corresponds to finding the shortest Hamiltonian cycle in a graph defined by $G = (V, E)$ of n cities represented as vertices V in the graph and edges E between the nodes to represent the interconnections. Each edge between vertices i and j is associated with a given cost denoted by d_{ij}. Thus, the TSP consists of finding a permutation π of n cities that minimizes the function of the total cost of visiting the n cities and going back to the starting point:

$$C(\pi) = \sum_{i=1}^{n-1} \left(d_{\pi(i),\pi(i+1)}\right) + d_{\pi(n),\pi(1)} \tag{1}$$

Evolutionary algorithms (EAs) are bio-inspired metaheuristics based on the natural evolution. There are many types of EAs: genetic algorithms (GAs) [11], evolutionary strategies [12], evolutionary programming [13], genetic programming [14] and estimation of distribution algorithms (EDAs) [15]. EDAs are a type of EAs which reproduce new individuals by simulation from a probabilistic model built from the best individuals of the previous generation.

In the last years, the capability of quantum computers to diminish the execution times and to solve very complex problems has gained a lot of interest. Quantum Computing (QC) [2] is based on the principles of quantum theory, which explains the behaviour of the interactions between atomic particles. Speed computation improvement is based on quantum mechanics phenomena such as superposition and entanglement.

Several well-known companies are working on developing quantum computers to run quantum algorithms. The main distinctions among the different quantum devices are the qubits quantum noise, the quantum topology which distributes the interrelations among the qubits, and the technology used to build the device. Different configurations may lead to different results due to quantum noise or decoherence times [3] influenced by the redistribution of the qubits to execute the algorithm in the desired topology.

Previous studies on quantum machine learning [16]–[18] have focused on exploiting the benefits of the QC to improve

the performance of the optimization algorithms. With respect to EAs, quantum machine learning revolves around three main research areas [19]: (i) quantum-inspired evolutionary algorithms (QIEA), which take advantage of the concepts and principles of quantum mechanics to improve the classic EAs; (ii) evolutionary-designed quantum algorithms, which develop new quantum algorithms implemented by genetic programming; and (iii) quantum evolutionary algorithms, which develop new evolutionary algorithms to be executed in quantum devices.

In this paper, we will focus on QIEA. Since the seminal paper [20] where principles of QC were used in the reproduction step of the EA, other works have applied different modifications to improve the results, or accommodate the algorithm to specific optimization problems. A matrix quantum individual representation is used in [21], [22] to solve ordering optimization problems. A novel quantum-inspired algorithm is proposed by [23] motivated by the colony behavior of the leafcutter ants. Other works propose different modifications to these algorithms in order to improve the smoothing and efficiency of the exploration [24], and compare them to the EDAs behaviour [25]. However, these works cannot be executed in a real circuit model-based quantum computer without being adapted, as mentioned in [26]. Implementing a QIEA in a quantum computer requires a hybrid implementation between quantum programming, for the reproduction step, and the classical programming, for the rest of the EA steps. Running these algorithms in a quantum computer also includes considering the quantum noise present in the quantum computers.

In this work, we present a new quantum-inspired approach for solving the TSP. The Quantum-Inspired Estimation of Distribution Algorithm (QIEDA) is a population-based algorithm based on QC for the reproduction of new solutions during runtime. The solutions obtained by the QIEDA are competitive with other state-of-the-art population-based approaches in terms of convergence and quality of solutions found. Also, we measure the impact of running the QIEDA in different quantum computers with non-identical topologies, and analyse which would be the ideal quantum topology to solve the TSP. The novelty of this algorithm, compared to those in the literature, is that it can be executed without adaptations in a real quantum computer based on the circuit model programming. A parameterized quantum circuit is used during the reproduction step in order to sample new solutions. The experiments were executed simulating the real IBM quantum computers.

The paper is organized as follows. Section II explains the implementation of the QIEDA approach. Section III presents an empirical comparison of the QIEDA with other optimization algorithms and a benchmarking of the computational cost for the TSP. Finally, Section IV concludes the work and proposes future research lines.

II. QIEDA

Depending on the relationships between the variables that are involved in the problem to be solved, the EDA approaches can be classified into three main groups: (i) univariate EDAs [27], which do not contemplate dependencies between variables; (ii) bivariate EDAs [28], which only consider pairwise dependencies; and (iii) multivariate EDAs, which do not restrict the relations among the variables, such as the estimation of Bayesian network algorithm (EBNA) [29] which uses Bayesian networks (BNs) [30], [31] to model the relationships among the variables.

QIEDA is a multivariate EDA which uses probabilistic logic sampling (PLS) [32] to generate new solutions from a BN. The algorithm samples new solutions taking advantage from the QC principles. The QIEDA implementation is a hybrid implementation between classic and quantum implementations. Qiskit (0.16.0) and IBM platforms [33] have been used to implement the quantum part.

A. Representation

In each iteration of the evolutionary algorithm, QIEDA performs a sampling process to obtain a set of solutions. Before describing the QIEDA implementation, the basics of QC are addressed briefly.

In the classical computation the minimum unit of information found in a computer is a classical bit. It can be set to 1 or 0 to perform different operations with other bits. In QC, the smallest unit of information is the quantum bit (qubit) [2]. It can be manipulated to be in the $|0\rangle$ or $|1\rangle$ states, or in a superposition of both. When a qubit measurement is performed, the probabilities make the qubit to collapse to one of the two states, $|0\rangle$ or $|1\rangle$. The qubits states can be represented following Dirac's notation as,

$$|\Psi_1\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2}$$

where α and β are complex numbers that represent the probability amplitudes of the corresponding states and $|\alpha|^2+|\beta|^2= 1$. $|\alpha|^2$ and $|\beta|^2$ are the probabilities of the qubit to be in the $|0\rangle$ and $|1\rangle$ states, respectively.

A 2-qubit is represented as $|\Psi_2\rangle = \delta_1\,|00\rangle + \delta_2\,|01\rangle + \delta_3\,|10\rangle + \delta_4\,|11\rangle$ with probability amplitudes $|\delta_1|^2, |\delta_2|^2, |\delta_3|^2, |\delta_4|^2$ respectively, such that $|\delta_1|^2+|\delta_2|^2+|\delta_3|^2+|\delta_4|^2= 1$.

In general, an n-qubit is represented as,

$$\begin{aligned}|\Psi_n\rangle =&\delta_1\,|00\ldots00\rangle + \delta_2\,|00\ldots01\rangle + \delta_3\,|00\ldots10\rangle + \\ &\delta_4\,|00\ldots11\rangle + \cdots + \delta_{2^n-1}\,|11\ldots10\rangle + \\ &\delta_{2^n}\,|11\ldots11\rangle\end{aligned} \tag{3}$$

with probability amplitudes $|\delta_1|^2, \cdots, |\delta_{2^n}|^2$ ($\sum_{i=1}^{2^n}|\delta_i|^2= 1$).

A qubit can be represented geometrically as a sphere, where the north and south poles represent the $|0\rangle$ and $|1\rangle$ states respectively, and a superposition of both states is represented as a point in the sphere surface named the Bloch sphere [2]. Several operations can be defined on the qubit to modify its state before a measurement is performed. These operations are carried out with quantum gates over the qubit. Some relevant gates used in this approach are the $RY(\phi)$ and $X(\phi)$ gates,

which perform a rotation in the Y and X axis of ϕ radians, respectively, in the Bloch sphere, among others.

There are a lot of ways to encode the TSP solutions [7]. In this paper, a matrix representation is used. Assume a set of cities $U = \{1, 2, \ldots, n\}$ of the TSP. The vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is a solution for the problem where $x_i \in U$, and i denotes the city ordering. Every x_i assumes one of the elements of U with a certain probability. The sum of probabilities of the elements in U must be equal to one. For example, for a TSP problem of size $n = 5$, assume for the first city to be visited (x_1) a vector of probabilities for the elements of U being observed $(0.1, 0.2, 0.1, 0.5, 0.1)$. Thus, the element x_1 takes the fourth city in U with 0.5 probability as it is the city with the highest probability, and once it is fixed, this city cannot be selected again by the rest of the elements of $\boldsymbol{x}$.

When adapting this codification to quantum individuals, each of the elements of U is translated as a pure state of an *n*-qubit. Then, every x_i assumes a different pure state with a certain probability. Each of the pure states that form the *n*-qubit have only one qubit in the $|1\rangle$ state. The position of the $|1\rangle$ state will be the city that represents that specific state. For example, in an *n*-qubit of size $n = 5$, the only desired pure states are $10000, 01000, 00100, 00010, 00001$ which represent the cities $1, 2, 3, 4, 5$ respectively. Thus, a universe of size n is represented by an *n*-qubit with n different states.

The *n*-qubit that represents the desired states to our problem codification is,

$$\begin{aligned} |\Psi\rangle = &\gamma_1 |10\ldots00\rangle + \gamma_2 |01\ldots00\rangle + \cdots + \\ &+ \gamma_{n-1} |00\ldots10\rangle + \gamma_n |00\ldots01\rangle , \end{aligned} \tag{4}$$

where $|\gamma_i|^2$ are the probabilities of each of the desired states and $\sum_{i=1}^{n} |\gamma_i|^2 = 1$. Note that the rest of $2^n - n$ states, which are not desired to be observed, are set to probability amplitudes equal to zero.

Therefore, using the matrix quantum codification, each solution of the TSP is represented as,

$$\boldsymbol{Q} = \begin{bmatrix} q_{11} & \cdots & q_{1n} \\ \vdots & \ddots & \vdots \\ q_{n1} & \cdots & q_{nn} \end{bmatrix}, q_{ij} \in \{0, 1\},$$

where rows accounts for city ordering while columns correspond to the position in U, and q_{ij} is equal to 1 whenever a city j is visited in the ordering i position. Thus,

$$\sum_{j=1}^{n} q_{ij} = 1, \forall i \in 1, \ldots, n. \tag{5}$$

An example of a valid solution for a TSP of size $n = 5$ is,

$$\boldsymbol{Q_1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

which corresponds to the following ordered city path [5, 3, 4, 1, 2].

B. Algorithm

Algorithm 1 describes the QIEDA algorithm.

Algorithm 1 QIEDA pseudocode

1: $\boldsymbol{p(Q_0)} \leftarrow$ Initialize matrix of statistics
2: $G_0 \leftarrow$ Generate N individuals from $\boldsymbol{p(Q_0)}$
3: **for** $i = 1, 2, ...$ until a stopping criterion is met **do**
4: $\quad G_{l-1} \leftarrow$ Evaluate the individuals with cost function C
5: $\quad G_{l-1}^{Se} \leftarrow$ Select $Se < N$ individuals from G_{l-1}
6: $\quad \boldsymbol{p(Q_i)} \leftarrow$ Update the matrix of statistics from G_{l-1}^{Se}
7: $\quad G_l \leftarrow$ Generate new generation from $\boldsymbol{p(Q_i)}$
8: **end for**=0

The QIEDA is initialized with a matrix of statistics (step 1) which specifies the relative frequency of appearance of each city in each ordering position, among the best solutions selected from the previous generation. In each iteration, the algorithm generates some new individuals from the matrix of statistics (step 2). Then the individuals are evaluated according to the cost function in Eq. 1 we desire to optimize (step 4), and the best individuals of the generation are selected (step 5) in order to improve the next generation cost. The matrix of statistics is updated (step 6) in each iteration with the selected individuals.

1) Initialization: The algorithm must be initialized with a matrix of statistics which specifies the probability of each position to take the value 1 when sampled,

$$\boldsymbol{p(Q)} = \begin{bmatrix} \boldsymbol{p(q_1)} \\ \vdots \\ \boldsymbol{p(q_n)} \end{bmatrix} = \begin{bmatrix} \gamma_{11} & \cdots & \gamma_{1n} \\ \vdots & \ddots & \vdots \\ \gamma_{n1} & \cdots & \gamma_{nn} \end{bmatrix} \tag{6}$$

The probabilities of each row verify the restriction of Eq. (5), and correspond to the probabilities of each of the pure states described in Eq. (4). This matrix is initialized so that no solution is favored beforehand and it is updated during the algorithm runtime until convergence is reached,

$$p(\boldsymbol{Q_0}) = \begin{bmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & \ddots & \vdots \\ \frac{1}{n} & \cdots & \frac{1}{n} \end{bmatrix}$$

When the QIEDA converges, the matrix of statistics should be a matrix of 1s and 0s (Eq. 5), which is the optimum solution found by the algorithm for the proposed TSP.

2) Individuals Generation: As described in Section II-A, the individuals are coded as matrix of ones and zeros where each row i specifies the city visited in the i position of the city ordering. When a city is visited, the following rows must update the probabilities to do not allow visiting this city again. The sampling process can be seen as a sequence of dependent steps in which each step means to sample a row i, and update rows $i+1, i+2, \cdots, n$ according to the i sampling outcome.

Consider a row i of an individual to be sampled in the QIEDA sampling method, expressed as $[q_{i1}, \ldots, q_{in}]$. The row sampling consists of a random selection among the pure states

defined by Eq. (4) with probabilities to be selected $\boldsymbol{p(q_i)} = [\gamma_{i1}, \ldots, \gamma_{in}]$ defined in the row i of the matrix of statistics, Eq. (6), of the corresponding algorithm iteration. Assume that the pure state c has been selected. Then, the following rows of the matrix of statistics are updated with zero probability for state c ($\gamma'_{hc} = 0$, for $h > i$), as the city cannot be visited more than once. Also, the remaining probabilities are updated as,

$$\gamma'_{hj} = \frac{\gamma_{hj}}{1 - \gamma_{ic}} \tag{7}$$

for $h > i, j \neq c$, where γ_{ic} is the previous probability value of row i and column $j = c$, that is replaced by zero. For example, assume a row which takes the values $[0.2, 0.3, 0.4, 0.1]$, and the last probability must be replaced by zero. Then, the row would result in $(\frac{0.2}{1-0.1}, \frac{0.3}{1-0.1}, \frac{0.4}{1-0.1}, 0)$.

The generation reproduction is arranged as a tree. An example of the tree building is shown in Fig. 1. Starting from the root node, we sample the first individual row from $\boldsymbol{p(q_1)}$. As a generation consists of N individuals, the algorithm performs N samplings that will be distributed among the possible pure states according to $\boldsymbol{p(q_1)}$. For example, in Fig. 1, N is distributed by approximately 33% for each node. This is the first level of the tree, whose maximum width is the number of pure states defined by $\boldsymbol{p(q_1)}$. It might occur that N is distributed in such a way that a node is not sampled, and thus, the first level width would be lower than expected. After updating the following rows, probabilities of rows 2 to n, the next level of the tree is built. Each node of level 1 unfolds its possible solutions to level 2 depending on the updated $\boldsymbol{p(q_2)}$. If a node is not sampled, it is not unfolded in the next level. Each node has an independent matrix of statistics depending on the parent node. The samplings of each node for the same level are independent, and thus, can be sampled in parallel. This process continues until reaching the leaf nodes (level n), where the sampled individuals are obtained. Total tree width varies depending on how N is distributed along the tree nodes. The maximum tree width is $N!$.

If we consider the tree as a directed acyclic graph in which each level nodes depend on the previous level nodes, the tree is a BN. There are a great amount of methods for sampling BNs [30]. In this paper, the process followed is similar to the probabilistic logic sampling (PLS), in which each node depends on its parents. PLS defines an ancestral ordering of nodes, which in this case is the city (row) ordering.

Algorithm 2 shows the adapted PLS for reproduction. We define a node as a structure with: (i) updated statistics depending on its parents, (ii) number of samplings (quantum circuit shots) to distribute among child nodes, (iii) level of the node, and (iv) the W state quantum circuit used for sampling.

To sample the individuals rows, the W state quantum circuits have been implemented [34]. The W state is an entangled quantum state [2] of n qubits, in which all possible pure states have one of the qubits in the $|1\rangle$ state, while all other ones are

Algorithm 2 PLS pseudocode

Input Matrix of statistics, N
Output Set of individuals

0: **struct** NODE
0: $stats \leftarrow$ Statistics to sample W state circuit
0: $size \leftarrow$ Number of shots from the quantum circuit
0: $level \leftarrow$ Level in the tree structure
0: $wstate \leftarrow W$ state circuit with $stats$ probabilities
0: **end struct**

1: $root \leftarrow Node(stats[0], size = N, level = 0)$
2: $leaf \leftarrow [root]$
3: **for** $node$ in $leaf$ **do**
4: $sols \leftarrow$ Sample $node.wstate$ $node.size$ times
5: **for** sol in $unique(sols)$ **do**
6: $level_{sol} \leftarrow node.level + 1$
7: $size_{sol} \leftarrow$ Count samplings of sol in $sols$
8: $node_{sol} \leftarrow Node(stats[level_{sol}], level_{sol}, size_{sol})$
9: $node_{sol}.stats \leftarrow$ Update statistics for row $level_{sol}$ given $node_{sol}$
10: $leaf.add(node_{sol}) \leftarrow$ Add node to leaf nodes list
11: **end for**
12: $leaf.remove(node) \leftarrow$ Remove node from leaf nodes list
13: **end for**=0

in the $|0\rangle$ state,

$$|W_n\rangle = \frac{1}{\sqrt{n}}(|10\ldots00\rangle + |01\ldots00\rangle + ...+ \\ + |00\ldots10\rangle + |00\ldots01\rangle) \tag{8}$$

The general W state of Eq. (8) has been modified in order to be able to set different probabilities to the different qubits. Thus, we can apply this approach to sample solutions according to Eq. (4). The circuit building is described in Algorithm 3. The process is divided into two main parts: (i) probability redistribution, in which, the desired probabilities of the pure states that constitute the W state are translated to rotations in the qubit Y axis with RY gates and controlled-RY gates (CRY), and (ii) state reshuffling, to ensure that the number of qubits in the $|1\rangle$ state is only one, with controlled-X gates (CX) and X axis rotation gates (X).

An example of a circuit of size $n = 5$ in which all pure states have the same probabilities is shown in Fig. 2.

The influence of noise in the W state circuit samplings is shown in Fig. 3. Panel (a) shows the histogram of the results obtained executing the quantum circuit in a quantum simulator without noise, while panel (b) shows the results obtained executing the circuit in a real quantum computer. Both experiments were run 1000 times with probabilities $[\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}]$. Note that in Fig. 3(b) some not desired solutions are sampled with a small probability due to the quantum computer noise. Solutions must be filtered and the probability distribution

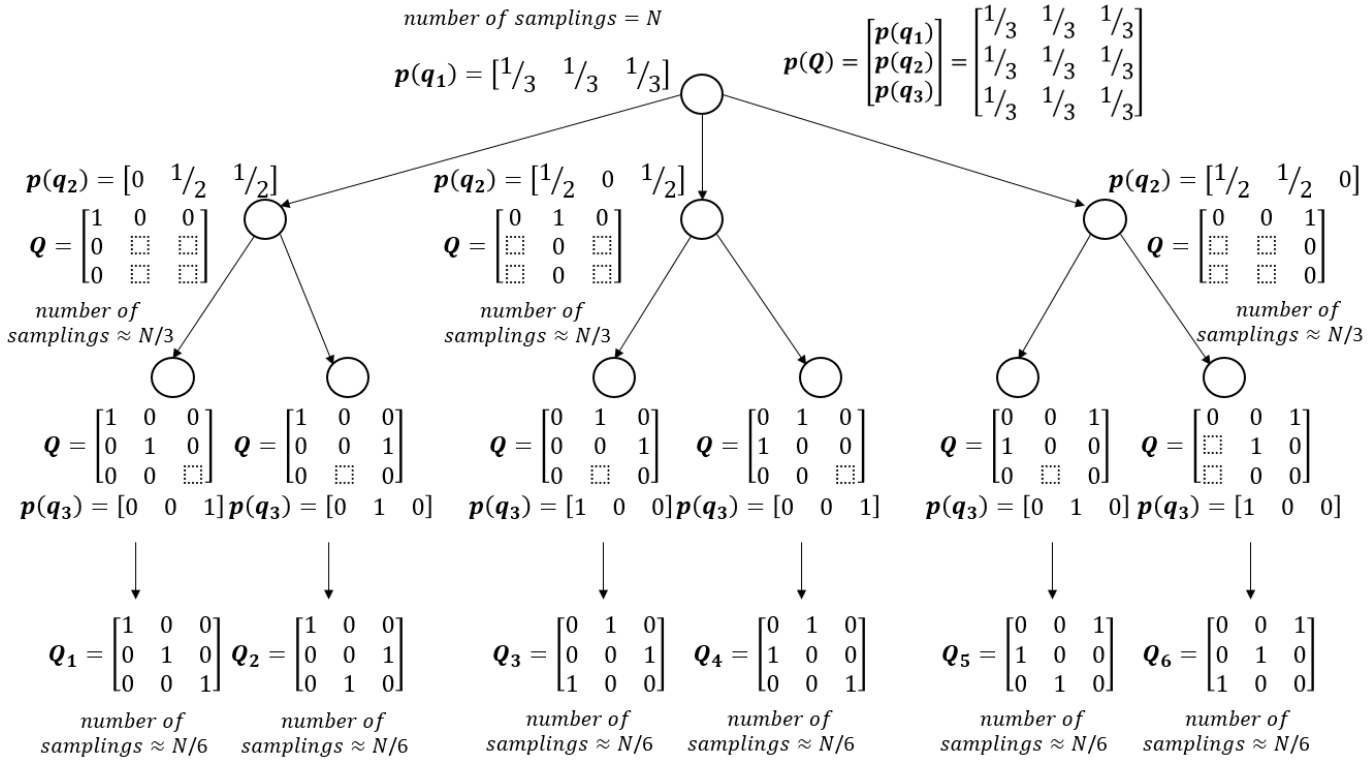

Fig. 1. Example of the quantum individuals sampling of size $n = 3$. An initial matrix of statistics $\boldsymbol{p(Q)}$ is updated in each node depending on its parents samplings. In the leaf nodes, the final individuals are obtained. The algorithm samples N individuals. Each node distributes its number of samples among its child nodes.

Algorithm 3 W state modified building

Input Vector of statistics
Output W state quantum circuit

$stats \leftarrow [\gamma_1, \ldots, \gamma_n]$
$rot_1 = 2cos^{-1}(\sqrt{\gamma_1})$
$wstate \leftarrow$ Quantum circuit with n qubits $q_1, \ldots, q_n$
$wstate \leftarrow$ RY in q_1 of rot_1 rad.
for $i = 2 \ldots n$ **do**
  $amp = \sqrt{1 - \sum_{j=1}^{i} \gamma_i}$
  $rot_i = 2cos^{-1}(\frac{(\gamma_i)^{1/2}}{amp})$
  $wstate \leftarrow CRY$ (rot_i rad.) target q_i and control q_{i-1}
end for
for $i = n, \ldots, 1$ **do**
  $wstate \leftarrow CX$ with target i and control $i - 1$
end for
$wstate \leftarrow X$ gate q_1 =0

among the valid solutions normalized. Valid solutions are the pure states defined following Eq. (4).

III. Results

In Section III-A the QIEDA performance is compared with other optimization algorithms, and in Section III-B the QIEDA is executed for different quantum computing topologies to analyze the computational cost. All the experiments were run using some quantum computer simulators available in the Qiskit software [33]. The ideal simulator and some real devices simulators such as Johannesburg or Tokyo were used to run the experiments.

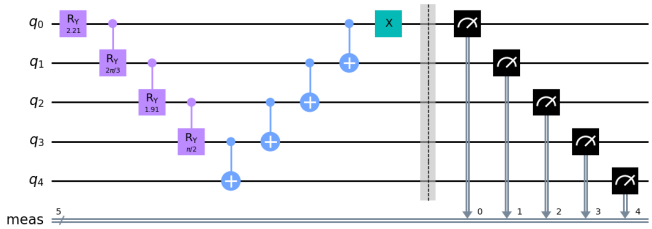

Fig. 2. Example of a W state circuit for $n = 5$. The probabilities are set to $[\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}]$. Purple gates are RY and CRY gates, blue ones are CX gates, green one is X gate, and black ones are the measurement gates.

A. Algorithm Performance

The performance of the QIEDA has been compared to other well known state-of-the-art population-based algorithms: classic genetic algorithm (GA) [35], a binary adaptation of the particle swarm optimization (PSO) [36], ant colony optimization (ACO) [37], and a non-quantum estimation of distribution algorithm (EDA). The non-quantum EDA is a modification of the QIEDA approach in which, the solutions are not sampled from a quantum circuit. New solutions are

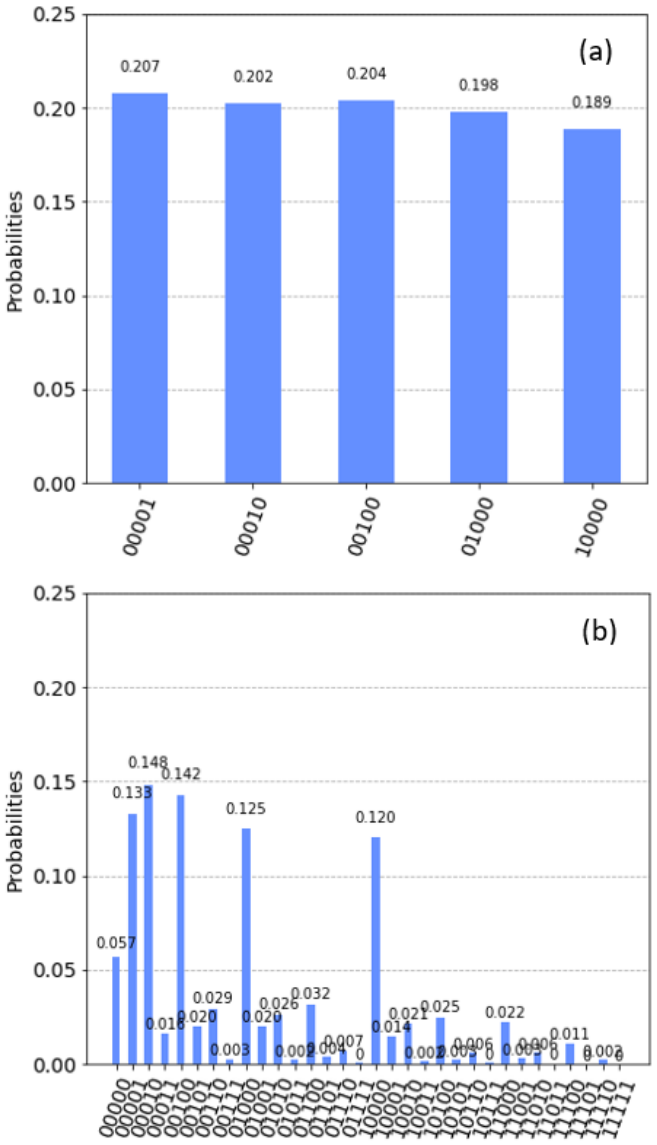

Fig. 3. Probability distribution of the W state circuit sampling for $n = 5$ without (a) and with (b) noise. The quantum circuit was run 1000 times for each scenario.

sampled from a categorical distribution using a binary random number generator. Its pseudocode is similar to that described in the quantum-inspired evolutionary algorithms [19], in which the number of qubits is not a limitation. See Appendix A for details of the algorithms.

The QIEDA is executed with the ideal simulator without quantum noise and in a quantum simulator with the Johannesburg quantum computer noise. The latter is the most realistic simulation as it is executed simulating the behaviour of a real quantum device. Johannesburg quantum device was chosen as the case study since it was observed that the quantum device selection does not influence the results obtained by the QIEDA.

Different datasets have been used [38] to simulate different

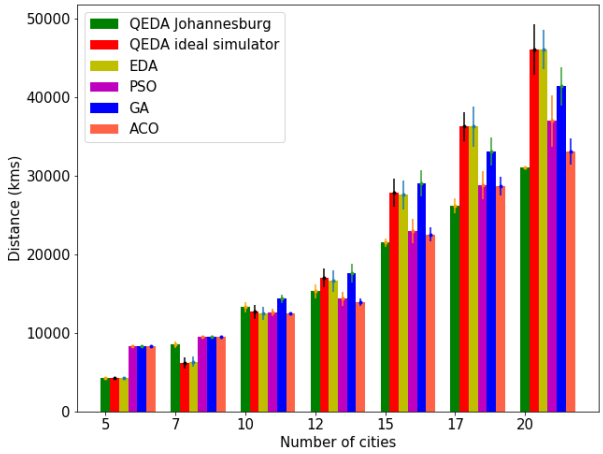

Fig. 4. Mean best cost for the TSP for different number of cities and different algorithms: QIEDA executed in the Johannesburg quantum simulator with noise, QIEDA executed in a quantum simulator without quantum noise, the non-quantum estimation of distribution algorithm (EDA), particle swarm optimization (PSO), genetic algorithm (GA), and the ant colony optimization (ACO).

TSP sizes. The experiments have been carried out for sizes $n = [5, 7, 10, 12, 15, 17, 20]$. The Qiskit simulators only allow to simulate up to 25 qubits. Thus, the aim of this analysis is to study the trend of the results in order to determine the performance of the algorithm.

Figure 4 shows an analysis of the optimum solutions obtained by the QIEDA compared to other population-based algorithms. The algorithms were run 100 times and the figure plots the mean and the deviations of the executions. Note that the QIEDA executed in the Johannesburg quantum computer achieves competitive results compared to other optimization algorithms. In nearly all the experiments, QIEDA achieves the best results on average compared to others, with a small deviation. In order to verify if the results are statistically significant some tests were run. After checking that the data fit a normal distribution, we calculated the p-value with an analysis of variance (ANOVA) [39], and the Student's t-test [40] of the QIEDA results compared to the other algorithms for each of the experiments. A significance level of 0.05 was set. ANOVA p-values are shown in Table I, and all t-tests yielded p-values lower than 0.05, so we can reject the null hypothesis of equal means. The QIEDA executed in the ideal simulator does not reproduce any noise, so the performance of the algorithm is similar to the non-quantum EDA. Note that the results obtained by both algorithms are very similar.

TABLE I
ANOVA TEST P-VALUES

n=5	n=7	n=10	n=12	n=15	n=17	n=20
6.1e-56	1.9e-23	0.021	8.8e-09	3.7e-18	1.4e-21	2e-23

It is also noteworthy that when increasing the TSP size,

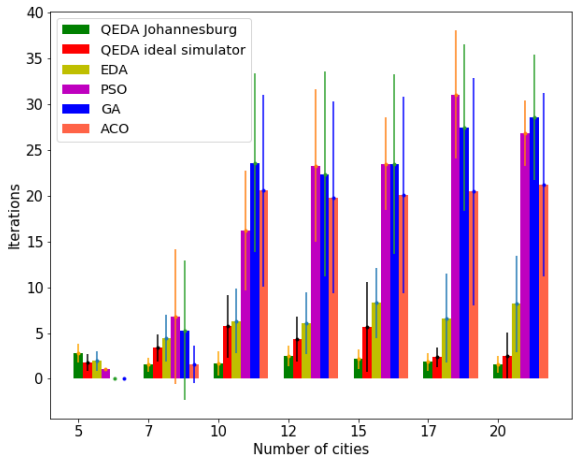

Fig. 5. Mean convergence for the TSP for different number of cities and different algorithms: QIEDA executed in the Johannesburg quantum simulator with noise, QIEDA executed in a quantum simulator without quantum noise, the non-quantum estimation of distribution algorithm (EDA), particle swarm optimization (PSO), genetic algorithm (GA), and the ant colony optimization (ACO).

the QIEDA performance improves compared to the other algorithms. The trend shows that, for larger (and therefore more complex) TSPs, the QIEDA will find much better solutions than the other population-based optimizers.

Figure 5 shows an analysis of the convergence of the QIEDA. The figure shows the mean and the deviation of 100 executions of the algorithm. The algorithm is compared to the same population-based algorithms in the same conditions as before. The EDA approaches have better convergence compared to other algorithms. Note that, regardless of the problem size, the QIEDA convergence remains approximately constant and with a small value compared to the other algorithms.

Despite the fact that the QIEDA approach was not able to be executed for larger TSP sizes, we can observe the trend of the results. As far as this behavior is concerned, the QIEDA approach achieves competitive results compared to the other algorithms. Moreover, the convergence of the algorithm remains constant independently of the size of the problem. Our results suggest that intrinsic quantum noise has the ability to enhance the convergence and the cost of the TSP as noise modifies the search space in such a way that it prevents the algorithm from falling into local optimal solutions.

B. Analysis Of Computing Topologies

There exists a large variety of quantum computers among which the main differences are the quantum topology (Fig. 6) which distributes the qubits and their relations. Two qubits A and B can only operate with each other only if there is a connection between both qubits. Otherwise, the system needs to swap with other qubits until A and B are connected.

To run a quantum circuit in a specific quantum computer some aspects must be analyzed:

- Available quantum gates. The circuit might have some quantum gates not available in the desired computer, so it must be adapted to execute it.
- Available number of qubits.
- Topology. The qubits of each quantum computer are distributed following different topologies. The circuit must adapt so the number of swap operations is minimized.

Executing a circuit without optimizing the number of swap operations increases the computational cost. Minimizing the number of quantum gates in the circuit decreases the computational cost. The aim of this benchmark is to find the ideal topology to execute the designed approach. Finding the topology which minimizes the computational cost of each executed W state circuit, would mean to reduce the computational cost of the QIEDA approach. To that end, we study the circuit depth that refers to the number of time steps (time complexity) required for the quantum operations making up the circuit to run on the quantum hardware [41].

The analysis and adaptation of the circuit to the specific quantum computer is named transpilation. During the transpilation, the circuit can be optimized to minimize the swap operation. Table II shows an analysis of the circuit depth with and without optimization for different topologies (Fig. 6). All the selected quantum computers have the same available quantum gates ($id, u1, u2, u3$ and cx), so the differences between the adapted circuits for each quantum computer are the number of swap operations added to be able to execute the circuit. The larger the depth, the larger the computational cost to execute the circuit in a quantum topology. Each swap operation is a combination of three sequential controlled-x gates between the two qubits to be swapped.

Note in Table II that for any topology, optimization improves the circuit depth as expected due to the minimization of the swap operations.

When, due to the quantum computer topology and the characteristics of the problem, swap operations must be carried out to use all the qubits of the quantum computer, the depth increases considerably in comparison with other topologies, as for example for *Almaden, Singapore* and *Boebligen* for $n = 20$ (depth of 170). For the same problem size, in a topology which does not imply doing swap operations, such as *Johannesburg* or *Poughkeepsie*, the depth decreases to less than a half. For larger problems than the executed in the experiments (for example $N = 50$), the best option would be to use the *Manhattan* topology due to the number of qubits, 65, and the multiple qubits distributions available without using swap operations. However, using such a large quantum computer for considerably smaller problems is a brute solution.

The topology of some computers limits the number of used qubits in order to do not increase considerably the depth of the circuit. As a solution, we have designed an ideal topology for the problem we are solving. In the W state circuit we are using the qubits interaction: $q_0 - q_1, q_1 - q_2, \ldots, q_{n-1} - q_n, q_n - q_{n-1}, q_{n-1} - q_{n-2}, \ldots, q_1 - q_0$. Thus, the ideal topology is a chain qubits distribution, named as *chain_backend* in Table II. An example is shown in Fig. 7. In the benchmark, for the

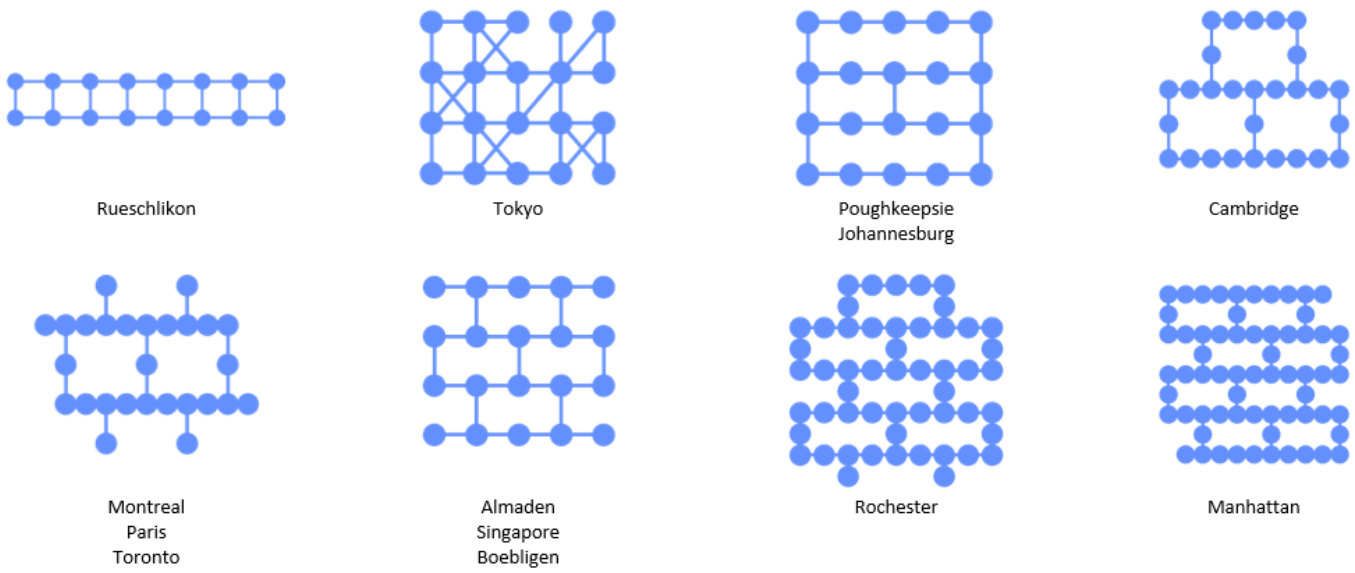

Fig. 6. Quantum computers topologies.

TABLE II
BENCHMARK. DEPTH ANALYSIS OF THE W STATE CIRCUIT FOR DIFFERENT TOPOLOGIES

Simulator	n_qubits	Without optimization				With optimization				
		10	15	20	50	10	15	20	50	60
Rueschlikon	16	62	98	-	-	56	98	-	-	-
Tokyo	20	57	90	145	-	36	56	122	-	-
Almaden	20	48	90	126	-	36	56	170	-	-
Johannesburg	20	60	102	139	-	36	56	76	-	-
Cambridge	28	72	104	184	-	36	56	76	-	-
Manhattan	65	36	116	145	449	36	56	76	196	581
Montreal	27	78	134	196	-	36	56	76	-	-
Rochester	53	69	104	181	481	36	56	76	667	-
chain_backend	-	36	56	76	196	36	56	76	196	236

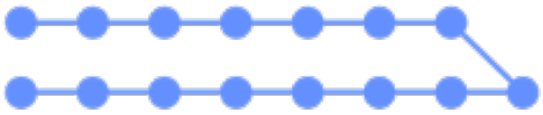

Fig. 7. Chain topology designed as the ideal topology for the QIEDA approach for $n = 15$.

problem of size 60, the chain topology improves *Manhattan* topology by a factor of 2.5.

Note that in Table II, the modified version of the W state circuits depth grows linearly with the size of the problem for the *chain_backend* topology, so the implemented metaheuristic is considered to be efficient [26].

IV. CONCLUSIONS

This work presented a new quantum-inspired EDA approach to solve the TSP problem. The QIEDA uses a modified version of the W state quantum circuits to adapt the PLS process to sample new solutions during the algorithm runtime.

The results obtained by the QIEDA were analysed in terms of convergence and optimum solution found. The algorithm behaviour was compared to other state-of-the-art population-based algorithms. The QIEDA number of iterations until convergence remains constant with increasing number of cities of the TSP, and is smaller than other algorithms. The solutions obtained by the QIEDA are competitive with the other algorithms, and the observed trend justifies the use of quantum computers to solve the TSP. The presence of quantum noise in the reproduction step of the algorithm improves its performance compared to others without noise. The QIEDA computational cost is also analysed by a benchmark. We have shown that the algorithm results are independent of the topology of the quantum computer chosen to be executed on. However, the topology is critical for the execution time due to the swap operations carried out by the quantum computers. We have proposed an ideal topology to solve the TSP although other technologies as IonQ [42] uses all-to-all connected qubits configurations that may be as faster as the one proposed here. However, this technology does not allow all the quantum gates that the W state circuit involves and further studies should be

carried out to adapt our approach.

Future work should include the implementation of a W state quantum circuit able to sample the full individual matrix instead of executing a W state circuit per matrix row. Also, the QIEDA approach could be generalized for other combinatorial optimization problems.

APPENDIX

The hyper-parameters of the population-based algorithms are shown in Table III. Despite the fact their tunning is out of the scope of this work, a previous analysis was done to achieve good solutions for the experiments, and compare them fairly with the QIEDA. For ACO, α is the relative importance of the pheromone and β is the relative importance of the trigger factor. As observed in [37], $\alpha < \beta$ achieves better results. rho is the evaporation rate of global pheromone which is equal to the evaporation rate of the local pheromone. For PSO, α is the cognitive parameter to control the exploitation component and β is the social parameter to control the exploration component of the algorithm. For GA, α is the population percentage considered as elite selection, and the mutation rate m_r is the parameter which influences how the individuals are modified in the mutation phase of the algorithm. For EDA, α is the population percentage considered as elite selection. N and Gen are the population size and the number of iterations, respectively.

TABLE III
EXPERIMENTS SETUP

	N	Gen	α	β	m_r	rho
ACO	50	40	0.4	0.6	-	0.5
PSO	50	40	0.55	0.45	-	-
GA	50	40	0.5	-	0.04	-
EDA	50	40	0.5	-	-	-

ACKNOWLEDGMENTS

The authors would like to thank A. Gomez from Centro de Supercomputación de Galicia (CESGA) for helpful discussions.

CODE AVAILABILITY

The code will be uploaded to **EDAspy** Python package. The package can be found at https://github.com/VicentePerezSoloviev/EDAspy and downloaded from Pypi.

REFERENCES

[1] K. Hussain, M. N. M. Salleh, S. Cheng, and Y. Shi, "Metaheuristic research: A comprehensive survey," *Artificial Intelligence Review*, vol. 52, no. 4, pp. 2191–2233, 2019.

[2] M. A. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2002.

[3] G. Benenti, G. Casati, and G. Strini, *Principles of Quantum Computation and Information-Volume II: Basic Tools and Special Topics*. World Scientific Publishing Company, 2007.

[4] T. D. Kieu, "The travelling salesman problem and adiabatic quantum computation: An algorithm," *Quantum Information Processing*, vol. 18, no. 3, p. 90, 2019.

[5] Y. Ruan, S. Marsh, X. Xue, Z. Liu, and J. Wang, "The quantum approximate algorithm for solving traveling salesman problem," *Computers, Materials and Continua*, vol. 63, no. 3, pp. 1237–1247, 2020.

[6] V. Robles, P. De Miguel, and P. Larrañaga, "Solving the traveling salesman problem with EDAs," in *Estimation of Distribution Algorithms*. Springer, 2002, pp. 211–229.

[7] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic algorithms for the travelling salesman problem: A review of representations and operators," *Artificial Intelligence Review*, vol. 13, no. 2, pp. 129–170, 1999.

[8] S. Miki, D. Yamamoto, and H. Ebara, "Applying deep learning and reinforcement learning to traveling salesman problem," in *2018 International Conference on Computing, Electronics & Communications Engineering*. IEEE, 2018, pp. 65–70.

[9] R. Zhang, A. Prokhorchuk, and J. Dauwels, "Deep reinforcement learning for traveling salesman problem with time windows and rejections," in *2020 International Joint Conference on Neural Networks*. IEEE, 2020, pp. 1–8.

[10] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[11] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. University of Michigan Press, 1975.

[12] I. Rechenberg, *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Frommann-Holzboog, 1973.

[13] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*. Wiley, 1966.

[14] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: The MIT Press, 1992.

[15] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2001.

[16] M. Schuld, *Supervised Learning with Quantum Computers*. Springer, 2018.

[17] H.-K. Lau, R. Pooser, G. Siopsis, and C. Weedbrook, "Quantum machine learning over infinite dimensions," *Physical Review Letters*, vol. 118, no. 8, p. 080501, 2017.

[18] N. Wiebe, D. Braun, and S. Lloyd, "Quantum algorithm for data fitting," *Physical Review Letters*, vol. 109, no. 5, p. 050505, 2012.

[19] G. Zhang, "Quantum-inspired evolutionary algorithms: a survey and empirical study," *Journal of Heuristics*, vol. 17, no. 3, pp. 303–351, 2011.

[20] K.-H. Han and J.-H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 580–593, 2002.

[21] L. R. Silveira, R. Tanscheit, and M. Vellasco, "Quantum-inspired genetic algorithms applied to ordering combinatorial optimization problems," in *2012 IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–7.

[22] L. R. da Silveira, R. Tanscheit, and M. M. Vellasco, "Quantum inspired evolutionary algorithm for ordering problems," *Expert Systems with Applications*, vol. 67, pp. 71–83, 2017.

[23] O. Montiel, Y. Rubio, C. Olvera, and A. Rivera, "Quantum-inspired acromyrmex evolutionary algorithm," *Scientific Reports*, vol. 9, no. 1, pp. 1–10, 2019.

[24] M. D. Platel, S. Schliebs, and N. Kasabov, "Quantum-inspired evolutionary algorithm: A multimodel EDA," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 6, pp. 1218–1232, 2008.

[25] ——, "A versatile quantum-inspired evolutionary algorithm," in *2007 IEEE Congress on Evolutionary Computation*. IEEE, 2007, pp. 423–430.

[26] O. H. M. Ross, "A review of quantum-inspired metaheuristics: Going from classical computers to real quantum computers," *IEEE Access*, vol. 8, pp. 814–838, 2019.

[27] H. Mühlenbein and G. Paass, "From recombination of genes to the estimation of distributions I. Binary parameters," in *International Conference on Parallel Problem Solving from Nature*. Springer, 1996, pp. 178–187.

[28] J. S. De Bonet, C. L. Isbell Jr, and P. A. Viola, "MIMIC: Finding optima by estimating probability densities," in *Advances in Neural Information Processing Systems*, 1997, pp. 424–430.

[29] R. Etxeberria and P. Larrañaga, "Global optimization using Bayesian networks," in *Proc. 2nd Symposium on Artificial Intelligence (CIMAF-99)*, 1999, pp. 332–339.

[30] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. The MIT press, 2009.
[31] P. Larrañaga, H. Karshenas, C. Bielza, and R. Santana, "A review on probabilistic graphical models in evolutionary computation," *Journal of Heuristics*, vol. 18, no. 5, pp. 795–819, 2012.
[32] M. Henrion, "Propagating uncertainty in Bayesian networks by probabilistic logic sampling," in *Uncertainty in Artificial Intelligence*. Elsevier, 1988, vol. 5, pp. 149–163.
[33] G. Aleksandrowicz, T. Alexander, P. Barkoutsos, L. Bello, Y. Ben-Haim, D. Bucher, F. Cabrera-Hernández, J. Carballo-Franquis, A. Chen, C. Chen *et al.*, "Qiskit: An open-source framework for quantum computing," 2019.
[34] D. Cruz, R. Fournier, F. Gremion, A. Jeannerot, K. Komagata, T. Tosic, J. Thiesbrummel, C. L. Chan, N. Macris, M.-A. Dupertuis *et al.*, "Efficient quantum algorithms for GHZ and W States, and implementation on the IBM Quantum Computer," *Advanced Quantum Technologies*, vol. 2, no. 5-6, p. 1970031, 2019.
[35] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine Learning*, vol. 3, no. 2–3, p. 95–99, 1988.
[36] S. K. Hadia, A. H. Joshi, C. K. Patel, and Y. P. Kosta, "Solving city routing issue with particle swarm optimization," *International Journal of Computer Applications*, vol. 47, no. 15, 2012.
[37] H. Yu, "Optimized Ant Colony Algorithm by Local Pheromone Update," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 12, 2014.
[38] Natural Sciences and Engineering Research Council of Canada and Department of Combinatorics and Optimization at the University of Waterloo, "Travelling salesman problem datasets," 2017. [Online]. Available: http://www.math.uwaterloo.ca/tsp/world/countries.html
[39] J. Kaufmann and A. Schering, "Analysis of variance anova," *Wiley StatsRef: Statistics Reference Online*, 2014.
[40] D. Kalpić, N. Hlupić, and M. Lovrić, "Student's t-tests," *International Encyclopedia of Statistical Science*, 2011.
[41] L. Gyongyosi, "Quantum state optimization and computational pathway evaluation for gate-model quantum computers," *Scientific Reports*, vol. 10, no. 1, pp. 1–12, 2020.
[42] IonQ, https://ionq.com/, [Accessed January 4 (2021)].