Variational Quantum Algorithm Parameter Tuning with Estimation of Distribution Algorithms

Vicente P. Soloviev, Pedro Larrañaga, Concha Bielza
Department of Artificial Intelligence, Universidad Politécnica de Madrid, Spain
(vicente.perez.soloviev, pedro.larranaga, mcbielza)@upm.es

2023 IEEE Congress on Evolutionary Computation (CEC) | 979-8-3503-1458-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/CEC53210.2023.10254138

Abstract—**Variational quantum algorithms (VQAs) are hybrid approaches between classical and quantum computation, where a classical optimizer proposes parameter configurations for a quantum parametric circuit which is iteratively sampled. The overall performance of the algorithm depends on how the classical optimizer tunes the parameters of the quantum circuit. Several gradient-free and gradient-based approaches have been proposed in the literature to face this task. Estimation of distribution algorithms (EDAs) are a type of evolutionary algorithms where a probabilistic model is updated and sampled at each generation to optimize a cost function. EDAs have shown to be able to achieve good solutions in a reasonable computation time for different optimization problems, and thus, we believe that this algorithm can be a good option to overcome VQAs challenges such as the Barren plateaus phenomenon. In this paper, we study the use of three different EDAs, characterized by different probabilistic model complexities, to tune the parameters of two different VQAs to solver the Max Cut problem and to a VQA to simulate the behaviour of a molecule. Three EDA variants are compared to some state-of-the-art optimizers widely used for this task. Our results show statistical significant improvement of the EDA variants compared to different optimizers, and identify the VQAs characteristics that best fit to each EDA type. We also perform an analysis of the main EDAs hyper-parameters.**

Index Terms—**Quantum optimization, variational quantum algorithms, estimation of distribution algorithm, gradient-free approach, gradient-based approach**

I. Introduction

In the last decades there has been a noticeable increase in the use and research in quantum computing technologies, which have been applied to a wide range of problems [1]. Quantum computing has demonstrated to be a way of saving energy consumption and outperforming classical computation in some specific problems, such as optimization or simulation. This is leading to the noisy intermediate-scale quantum (NISQ) era, which is characterized by being limited by the number of qubits of the devices and the presence of quantum noise in the systems.

Three different research tendencies can be identified in the area of quantum optimization: (i) quantum-inspired optimization [2], [3], which uses classical computers to simulate the principles of quantum mechanics; (ii) quantum adiabatic computation [4], which consists of encoding the optimization problem to be solved into a set of Hamiltonians that represent an energy function, in which the ground state is desired to be found; and (iii) variational quantum algorithms (VQAs) [5], which use quantum parametric circuits (henceforth named as *ansatz)* to optimize a given problem.

VQAs are hybrid approaches between quantum and classical computation, which have been applied to a wide range of problems [6] [7] [8] for different advantages, such as their resilience to the presence of quantum noise in quantum computers [9]. Due to the common characteristics with the classical heuristics, these algorithms are also known as quantum heuristics. VQAs involve an *ansatz* that represents a quantum state, which is used as a sampling engine for new solutions. Iteratively, a classical optimizer proposes new *ansatz* parameter configurations, which represent configurations of angle rotations in the qubits of the system. Thus, the overall performance of the approach heavily depends on how the classical optimizer tunes the *ansatz* parameters. Some characteristics of this optimization problem are the increase of parameters depending on the VQA, the computational time invested, or the presence of Barren plateaus [10], a phenomenon present in VQAs where gradients are vanished exponentially with the number of qubits of the system. Some of these characteristics are analyzed in this work. An optimum parameter configuration usually leads to a correctly optimization of the cost function. This task has been researched deeply in the last decades from different perspectives [11], such as population-based algorithms [12].

Estimation of distribution algorithms (EDAs) [13] are a type of evolutionary algorithms where a probabilistic model is updated iteratively with the best individuals of previous generations. Depending on the complexity and type of probabilistic model, EDAs can be classified into univariate/multivariate or parametric/semiparametric/nonparametric approaches. Firstly, a univariate EDA, such as the univariate marginal distribution algorithm (UMDA) [14], embeds a probabilistic model which does not contemplate dependencies between the variables, while a multivariate EDA allows dependencies between variables by using complex probabilistic models as Bayesian networks (BNs) [15], such as the estimation of Gaussian networks algorithm (EGNA) [16] [17]. Secondly, for continuous optimization a joint Gaussian probability distribution is commonly assumed in the variables to be optimized, which leads to a parametric EDA, such as EGNA, or some works which use mixtures of distributions [18] [19]. Some nonparametric approaches have been proposed in the literature using kernel

This work was partially supported by the Spanish Ministry of Science and Innovation through the PID2019-109247GB-I00, TED2021-131310B-I00 and RTC2019-006871-7 projects. Author 1 was supported by the predoctoral grant FPI PRE2020-094828 from the Spanish Ministry of Science and Innovation.

density estimation (KDE) [20] [21], but no further work was researched in this line due to its heavy computational burden. Semiparametric EDAs (SPEDAs) [22] allow KDE variables to co-exist with Gaussian variables, and they decide if the conditional density of a variable given its parent variables in the BN is Gaussian or KDE. Some recent works propose quantum evolutionary algorithms [23] [24].

In this work, we compare the use of different EDAs to face the VQAs *ansatz* parameter tuning. The results show a univariate (UMDA), a multivariate parametric (EGNA), and a multivariate semiparametric EDA (SPEDA) compared to the state-of-the-art classical optimizers used for this task, in two different VQAs, analyzing the quality of the solutions found and their computational runtime. Although the use of (the simplest) UMDA for a specific VQA *ansatz* parameter tuning has been previously researched [25] with promising results, the aim of this paper is to provide a deeper comparison of different EDA complexities and identify the general VQA characteristics that better fit to each EDA analyzed. We extend the previous work by analysing the three EDA variants in different optimization scenarios, and performing an analysis of the main EDA hyper-parameters.

The remainder of the paper is organized as follows. Section II and Section III review some theoretical background about the VQAs and EDAs, respectively. Section IV reports the experimental results comparing different EDA complexities to other state-of-the-art optimizers. Section V rounds the paper off with some conclusions and future work on the topic.

II. Variational quantum algorithms

VQAs are iterative algorithms in which, in each iteration, a classical optimizer proposes new parameter configurations $\boldsymbol{\theta} \in [0, 2\pi]^n$ for the *ansatz*, which is then measured to obtain new sets of solutions, where n is the number of qubits of the quantum system. The parameters represent angle rotations in the *ansatz* of one and two-qubits rotation gates. For each parameter configuration ($\boldsymbol{\theta}$) proposed by the classical optimizer, the *ansatz* is measured N times, and the expectation value ($E(Z)$) among the solutions Z is computed as follows:

$$E(Z) = \sum_{z \in Z} C(z) N_z, \tag{1}$$

where z is each different solution in Z, $C(z)$ is the cost of the solution evaluated in the classical cost function to be optimized, N_z is the number of occurrences of z in Z, and $N = \sum_{z \in Z} N_z$. Considering $E(Z)$ and Z, the classical optimizer proposes new sets of parameters, and iterates until the convergence criterion is met.

Then, VQAs do not minimize the classical cost function, but minimize the expectation value among the solutions sampled from the *ansatz*. Minimizing $E(Z)$ will lead to finding the ground state of the quantum state $\psi(\boldsymbol{\theta})$, which will consequently lead to the classical cost function minimization. The VQA workflow is shown in Fig. 1.

In the literature, there exists a plethora of VQA proposals; however, the quantum approximate optimization algorithm

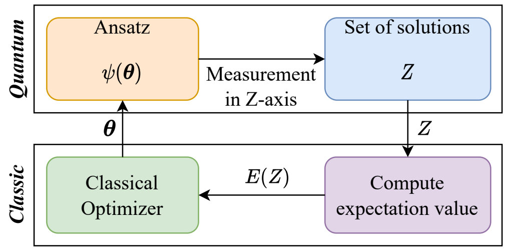

Fig. 1. VQA workflow where, iteratively, the classical optimizer proposes a new set of parameters ($\boldsymbol{\theta}$) for the *ansatz* $\psi(\boldsymbol{\theta})$, which is then measured (Z), and the expectation value is computed ($E(Z)$) Note that when measurement is performed, each qubit is measured in the Z-axis.

(QAOA) [26] and the variational quantum eigensolver (VQE) [27] have shown to be quite resilient to quantum noises present in NISQ devices and to achieve competitive results in a wide range of problems [7]. The main difference between both approaches is the *ansatz* design, which, in the case of QAOA is specific for each problem instance, while VQE uses a pre-designed *ansatz* for every given problem. In this work, a study will be conducted on each of the two VQAs.

A. QAOA

QAOA was originally proposed by Farhi, Goldstone and Gutmann [26] for solving combinatorial optimization problems. The QAOA *ansatz* is composed of $p \in \mathbb{N}$ layers, which internally builds two sequential operators: (i) the cost operator $U(H_C, \gamma)$ parameterized by $\gamma \in [0, 2\pi]$, which is built specifically for each problem instance and encodes the classical cost function through a combination of single and two-qubit rotation gates,

$$U(H_C, \gamma) = e^{-i\gamma H_C} = \prod_{\alpha=1}^{m} e^{-i\gamma C_\alpha}, \tag{2}$$

where C_α is the cost function to be minimized, and m the number of clauses that define the classical cost function to be optimized; and (ii) the mixed operator $U(H_B, \beta)$ parameterized by $\beta \in [0, 2\pi]$, which represents a rotation in the X-axis in each qubit (σ_j^x),

$$U(H_B, \beta) = e^{-i\beta H_B} = \prod_{j=1}^{n} e^{-i\beta \sigma_j^x}, \tag{3}$$

where n is the number of qubits of the quantum system.

Thus, an *ansatz* composed by p layers has $2p$ parameters to be optimized $\boldsymbol{\theta} = (\gamma_1, \beta_1, \ldots, \gamma_p, \beta_p)$, as shown in the *ansatz* example in Fig. 2. The quantum state represented by the QAOA *ansatz* is,

$$\psi(\boldsymbol{\gamma}, \boldsymbol{\beta}) = U(H_B, \beta_p) U(H_C, \gamma_p) \cdots U(H_B, \beta_1) U(H_C, \gamma_1) \langle s \rangle, \tag{4}$$

where $p \geq 1$, $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$, and $\langle s \rangle$ is the superposition state over the computational quantum states.

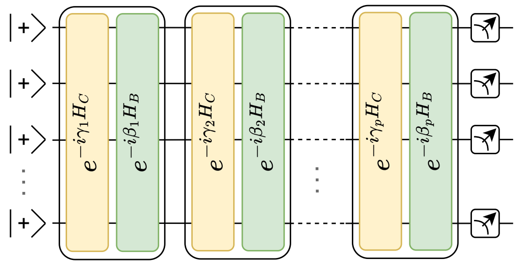

Fig. 2. An *ansatz* with p layers and $2p$ parameters to be tuned. The quantum circuit starts from a superposition state over all the possible computational states, then applies p layers, and measures the qubits in the Z-axis.

B. VQE

The VQE was originally proposed by Peruzzo et al. [27]. In contrast to QAOA, the VQE *ansatz* is not designed specifically for each problem instance. There exist several pre-designed *ansatz* which are used independently of the optimization problem to be solved, and usually involve a larger number of parameters to be optimized. The parameters are tuned following the workflow defined in Fig. 1. In this work, we will focus on the TwoLocal *ansatz*[1], shown in Fig. 3, one of the most widely-used *ansatz* in the literature.

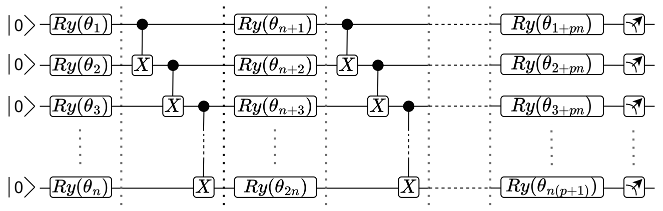

Fig. 3. TwoLocal *ansatz* used for the VQE with $p \in \mathbb{N}$ layers, n qubits and pn parameters. Each layer applied a rotation gate in the Y-axis over each qubit. Between each layer, a linear entanglement is applied in the system, where each qubit i is connected through a two-qubit rotation gate in the X-axis to the qubit $i+1$.

The TwoLocal *ansatz* is composed by p layers, which have a single-qubit rotation gate in the Y-axis for each qubit in the system, parameterized by an independent parameter per quantum gate. Between each layer, the *ansatz* builds a linear entanglement in which the qubit i is connected with the qubit $i+1$ through a two-qubit rotation gate in the X-axis. Thus, an *ansatz* composed by p layers has pn parameters to be optimized $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_{pn})$.

C. Parameter tuning

The *ansatz* parameter tuning consists of a continuous optimization where the parameters represent angle rotations of the qubits, and thus, are restricted to $[0, 2\pi]$. However, depending on the type of VQA to be used, and the number of layers that compose the circuit, the number of parameters vary substantially. Additionally, it is necessary to decide whether to prioritise the algorithm runtime, or the quality of the solutions given by the optimiser.

Considering the previous optimization requirements, we list the following state-of-the-art optimizers grouped according to whether they are gradient-based or not [28], which will be analyzed in the comparison performed in this work.

Some gradient-based optimizers include:

- ADAM [29]: stochastic version of the gradient descent.
- Conjugate gradient method (CG) [30]: designed for systems of linear equations whose matrices are symmetric and positive-definite.
- Gradient descent [31]: a first-order optimization algorithm, commonly used in deep learning to optimize the loss function.
- Limited-memory Broyden–Fletcher–Goldfarb-Shanno Bound algorithm (L-BFGS-B) [32]: limited memory-based and designed for non-linear optimization problems.
- Sequential least squares programming (SLSQP) [33]: finds a local search direction by solving the second-order local approximation of the objective function.

Some gradient-free optimizers include:

- Constrained optimization by linear approximation (COBYLA) [34]: uses linear approximations of the objective functions, and is mostly used when the derivative of the objective function is unknown.
- Simultaneous perturbation stochastic approximation (SPSA) [35]: suited for large-scale population optimizations.

III. Estimation of distribution algorithms

The EDA baseline is shown in Algorithm 1, where E is the cost function (Eq. 1) to be minimized. EDAs are a type of evolutionary algorithms where a probabilistic model is used as the sampling engine for new solutions (line 6). This probabilistic model is updated at each iteration considering promising solutions selected from previous generations (lines 4–5). The algorithm iterates until the stopping criteria is met.

Algorithm 1 EDA baseline

```
Input: Population size $\mu$, selection ratio $\delta$ and cost function $E$
Output: Best individual $\boldsymbol{z}'$ and cost found $E(\boldsymbol{z}')$
1: $G_0 \leftarrow \mu$ individuals randomly sampled
2: for $t = 1, 2, ...$ until stopping criterion is met do
3:     Evaluate $G_{t-1}$ according to $E(\cdot)$
4:     $G^S_{t-1} \leftarrow$ Select $\lfloor \delta N \rfloor$ individuals from $G_{t-1}$
5:     $f_{t-1}(\cdot) \leftarrow$ Learn a probabilistic model from $G^S_{t-1}$
6:     $G_t \leftarrow$ Sample $\mu$ individuals from $f_{t-1}(\cdot)$
7: end for
```

Each individual of each generation is a parameter configuration $\boldsymbol{\theta}$ to be evaluated in the *ansatz*. Then, the quantum circuit with the specific configuration is measured N times, and $E(\cdot)$ is computed (line 3). Considering all the proposed configurations in the same generation, and the respective

[1]https://qiskit.org/documentation/stubs/qiskit.circuit.library.TwoLocal.html

computed expectation value, the probabilistic model is updated (line 5). In this paper, we will focus on UMDA, EGNA and SPEDA. All the algorithms are considered to have converged if after 25 iterations no improvement is identified (stopping criterion); otherwise, the algorithm will run the maximum number of iterations defined by the user.

A. *UMDA*

UMDA assumes a Gaussian distribution over each independent variable, where each variable refers to each parameter of the *ansatz*. Then, the algorithm, in each iteration, selects the best individuals and fits an independent Gaussian distribution over each variable. To generate a new solution, each independent Gaussian is sampled once. The main advantage of this algorithm is its simplicity, which on the other hand is a disadvantage, since it is very prone to fall into local optima because of ignoring dependencies between variables.

B. *EGNA*

To address the UMDA shortcomings, EGNA is able to contemplate dependencies between variables, with the use of Gaussian Bayesian networks (GBNs). In each iteration, the structure and parameters of a GBN are learned, where the joint probability distribution of the variables is assumed to be linearly Gaussian. Thus, marginal and conditional distributions are also Gaussians. Although there exist other multivariate variants that model such dependencies, EGNA is one of the most widely used approaches due to its speed and the capability of generalization. Nevertheless, some works have found that EGNA might fall into local optima and revisit previously explored areas of the search space [22] [36].

C. *SPEDA*

Because some distributions cannot be approximated using Gaussians, the SPEDA allows Gaussian variables and KDE variables to coexist. SPEDA decides whether the conditional density of a variable given its parents in the graphical structure representing the probabilistic model will be Gaussian or not. This type of algorithm is able to find solutions not achievable by other types of EDAs, in addition to minimizing the number of iterations needed for convergence [22]. Due to the complexity of the probabilistic model embedded by SPEDA, the computation time was shown to be a drawback in some experiments.

IV. Experimental results

In this section we show some numerical results when comparing different EDA complexities to some state-of-the-art gradient-free and -based algorithms. Note that all the implemented experiments for this study have been coded in Python using Qiskit-0.21.2 [37] and EDAspy-1.0.2[2,3] Python packages and is available at a GitHub repository[4]. Different optimizers have been executed to tune the parameters of the QAOA *ansatz* and the VQE TwoLocal *ansatz*, both to optimize the same benchmark, an instance of the well-known MaxCut benchmark for 10 variables and 10 qubits ($n = 10$). Moreover, Section IV-C shows some experimental results using different optimizers for tuning the parameters of an VQE *ansatz* used to simulate a molecule. Section IV-D analyzes the hyper-parameter tuning of the EDA approach.

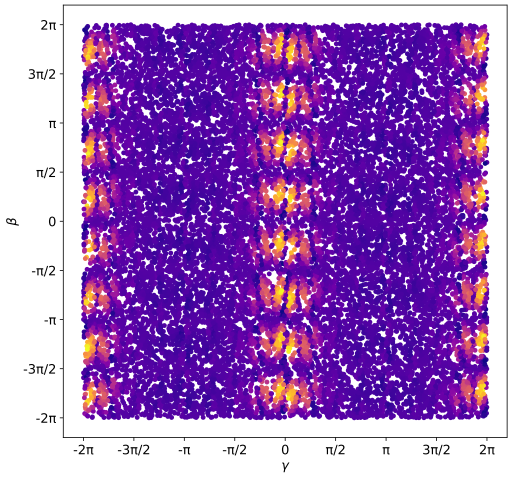

Fig. 4. Optimization landscape where Y- and X-axis represent, respectively, β and γ QAOA *ansatz* parameters for $p = 1$ layer, and the lighted regions over the purple background represent the optimum parameters that minimize the expectation value (Eq. 1). Although the parameter tuning has been restricted to $\boldsymbol{\theta} \in [0, 2\pi]^n$, the landscape of $\boldsymbol{\theta} \in [-2\pi, 2\pi]^n$ is displayed to show that -2π is equivalent to 2π.

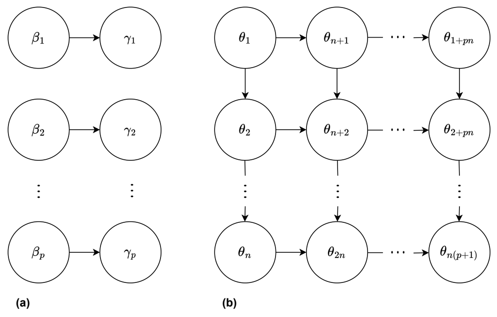

Fig. 5. GBN structures designed for QAOA (a) and VQE TwoLocal (b) *ansatz* parameter tuning embedded by EGNA. Each node represents a parameter in the *ansatz*, and an arc between two nodes represents a linear Gaussian dependency between both variables.

All the algorithms have been configured to a maximum of 100 iterations and the number of shots for the *ansatz* is $t = 1000$. In the case of the EDAs, the selection ratio has been set to $\delta = 0.5$ for all the variants, the population size is $\mu = 60$, and the EDA is considered to have converged if after 20 generations, the algorithm has not improved the best solution found. Note that all the EDAs have been implemented as elitist approaches, where the best individuals of each iteration remain in the future, so that the algorithm never worsens the best results found in previous iterations, and also SPEDA's archive-based feature has been omitted to perform a fair comparison with the other EDAs ($l = 1$, where l is the archive length).

[2]https://github.com/VicentePerezSoloviev/EDAspy
[3]https://pypi.org/project/EDAspy/
[4]https://github.com/VicentePerezSoloviev/EDA_QAOA

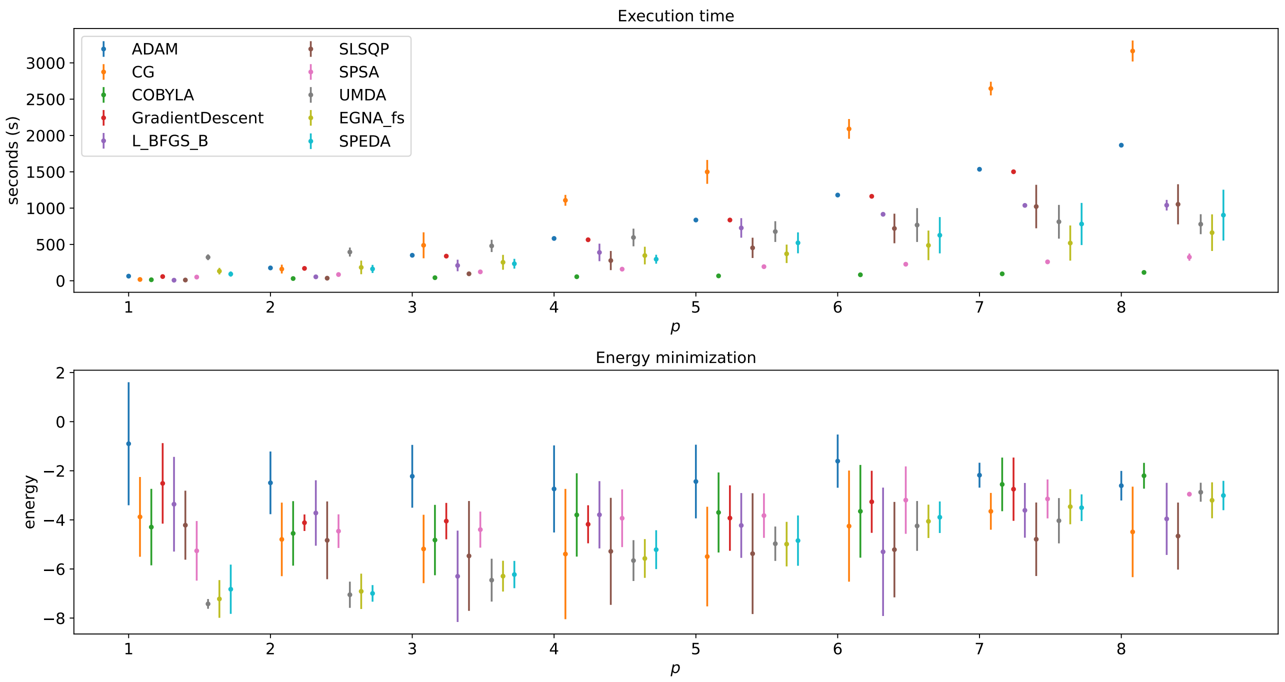

Fig. 6. Comparison of the different EDA variants with the other state-of-the-art optimizers for the QAOA *ansatz* parameter tuning. The top panel shows the mean computation time and standard deviation after running each algorithm 25 independent times for different number of layers $p \in \{1, \dots, 8\}$. The bottom panel shows the mean best expectation value achieved (Eq. 1) and standard deviation after running each algorithm 25 independent times for different number of layers $p \in \{1, \dots, 8\}$.

Different works in the literature [38] [39] have shown that the QAOA *ansatz* parameters in the same layer are dependent, and there exist different optimum configurations for γ_i and β_i for which the expectation value (Eq. 1) is minimum. Fig. 4 shows an example for the MaxCut problem instance, with $p = 1$, where the optimization landscape is shown. Lighted areas over the purple background represent the optimum parameters for the QAOA *ansatz*, which our EDA approach should find.

Due to this dependency between γ_i and β_i in the same layer, in the following experiments the GBN structure embedded by the EGNA approach has been fixed. An example of the embedded GBN in the EGNA approach to tune QAOA *ansatz* parameters with p layers is shown in Fig. 5(a), where each node represents a parameter in the QAOA *ansatz*, and γ_i and β_i are connected for each layer. Thus, the runtime of EGNA is expected to be reduced, as the structure of the GBN is not learned, and only the parameters of the model are updated according to the provided data.

Following the same strategy, a fixed structure has been proposed for the VQE *ansatz* parameter tuning using EGNA. In this case, as in each layer a quantum parametric gate is implemented for each qubit, we have designed a structure in which each parameter $\boldsymbol{\theta}_i$ is connected to parameter $\boldsymbol{\theta}_{i+1}$ in the same layer, justified by the subsequent entanglement of the qubits following the same sequence, between each layer of the *ansatz*, as shown in Fig. 3. In addition, the parameters of the quantum parametric gates in each qubit are sequentially connected. An example is shown in Fig. 5(b), where each node represents a parameter in the *ansatz* with p layers, n qubits and pn parameters.

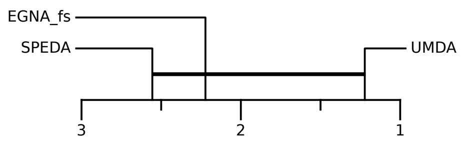

Fig. 7. Critical difference diagram using Friedman tests to reject the null hypothesis of equal expected value, and a post-hoc analysis based on the Wilcoxon-Holm method. The horizontal black line connects the EDA variants that do not have a significant difference in the QAOA *ansatz* parameter tuning in terms of expectation value minimization using the three EDA variants.

In the case of SPEDA, we have decided not to restrict the topology, since the BN structure learning algorithm embedded by SPEDA also learns the node types (Gaussian or KDE).

A. *QAOA ansatz parameter tuning*

Fig. 6 shows a comparison of the computation time and expectation value minimization for different optimizers and the three EDAs: UMDA, EGNA with the fixed structure (EGNA_fs) and SPEDA, for the QAOA parameter tuning.

Fig. 6 (top) shows how EGNA_fs reduces the computation time notably. Note that, when $p < 6$ the UMDA takes a notably larger computation time to converge compared to SPEDA and EGNA_fs, which is probably due to the number of iterations needed during runtime. SPEDA and EGNA_fs take a similar computation time up to $p < 5$. For larger p, EGNA_fs improves runtime.

No statistical significant differences have been found between the three EDA approaches in terms of expectation value

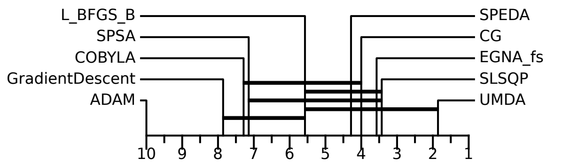

Fig. 8. Critical difference diagram for the QAOA *ansatz* parameter tuning in terms of expectation value minimization comparing the different optimizers including the EDA variants.

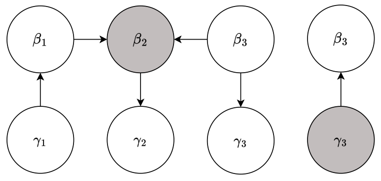

Fig. 9. BN structure where the common arcs found in the BNs of the last iterations of 10 different SPEDA runs for the QAOA *ansatz* parameter tuning with $p = 4$ are represented. White and grey nodes represent the Gaussian and KDE nodes, respectively.

minimization, as shown in the critical difference diagram [40] in Fig. 7. Comparing the EDA variants against the other algorithms, it is shown how the EDAs are the algorithms which achieve the smallest expectation values when $p < 5$. For more layers, the EDAs provide competitive results compared to their competitors, being only beaten by some gradient-based algorithms (CG, L_BFGS_B and SLSQP), see Fig 6 (bottom).

Analyzing the computation time, when $p > 5$, the EDA variants offer a computation time advantage compared to most of their competitors. Nevertheless, when $p < 6$, the computation time is slightly larger or equal than the rest of their competitors. It is worth noting that the computation time of CG is no longer competitive when $p > 3$.

Fig. 8 shows a critical difference diagram where the statistical significant differences are identified in terms of expectation value minimization. The three EDA variants are grouped with other three gradient-based approaches (CG, L_BFGS_B and SLSQP) as the best optimizers for the QAOA *ansatz* parameter tuning for $p \in \{1, 2, \dots, 8\}$, and UMDA improves the performance over the other gradient-free optimizers.

From this analysis we conjecture that EGNA_fs improves the expectation value minimization compared to its competitors, achieving a competitive computation time, when $p \leq 5$. When $p > 5$, EGNA_fs reaches similar results compared to its competitors but offering one of the best computation times.

One of the advantages of EDAs is the interpretability of the algorithm. Due to the use of BNs, it is possible to infer dependencies between variables, which may be of interest when analyzing the problem to be solved. It is expected that the EDA, in the last iterations of the runtime, will find the optimal structure that represents the landscape of the cost function. Fig. 9 shows a BN structure presenting the common arcs found in the BNs of the last iterations of 10 different SPEDA runs for the QAOA *ansatz* parameter tuning with $p = 4$, where white and grey nodes represent the Gaussian and KDE nodes, respectively. Comparing the BN structure with the one fixed for the EGNA_fs approach in Fig. 5, it can be observed the common arcs between nodes β_i and γ_i in each layer, although some arcs have been reversed, which verifies that the structure fixed for the EGNA_fs approach is consistent with the findings. Some spurious arcs have also been identified, such as $\beta_1 \rightarrow \beta_2$ and $\beta_3 \rightarrow \beta_2$.

B. VQE ansatz parameter tuning

Fig. 10 shows a comparison of the computation time and expectation value minimization for different optimizers and the three EDAs: UMDA, EGNA with the fixed structure (EGNA_fs) and SPEDA for the VQE *ansatz* parameter tuning.

No significant differences were found between EGNA_fs and SPEDA; however, a significant improvement is found for the case of UMDA for $p \in \{1, \dots, 9\}$, as shown in the critical difference diagram in Fig. 11. Analyzing the computation time, it can be observed that UMDA, for all p, needs a larger runtime than the other EDA variants, and as p increases, a greater difference is found. For $p \geq 6$ the difference of computation time compared to EGNA_fs and SPEDA is noticeably larger, while the minimization of the expectation value no longer has a statistically significant advantage, as shown in Fig. 12, where all the EDA variants are grouped together. From this analysis we conjecture that UMDA is a competitive optimizer for TwoLocal *ansatz* parameter tuning for low values of p. For larger values, EGNA_fs and SPEDA outperform UMDA, if a trade-off between computation time and expectation value minimization is desired. Moreover, it is worth mentioning the low standard deviation of the expectation value achieved with EGNA_fs and SPEDA compared to its competitors.

The computation time difference of UMDA compared to the other EDA variants has been identified in both the QAOA and VQE *ansatz* parameter tuning. Fig. 13 shows the number of cost function evaluations of the three EDA variants, for different values of p in the VQE TwoLocal *ansatz* case. Note that each cost function evaluation involves an *ansatz* parameter configuration and measuring the quantum circuit N times. The figure shows how SPEDA is the EDA variant which needs fewer evaluations to converge to a solution that has no significant difference with EGNA_fs and UMDA, for $p \geq 6$. Note that UMDA is the algorithm which needs more evaluations for convergence, and the difference compared to EGNA_fs and SPEDA increases with p. Although the number of evaluations of SPEDA is lower than those of EGNA, the computation time has shown to be slightly higher in Fig. 6 due to the probabilistic model complexity embedded by SPEDA and its structure learning, which was omitted in the case of EGNA_fs. SPEDA estimates some variables using KDE, exploring several areas of the search space in parallel [22], so the cost function evaluations is likely to be reduced, which seems to be happening in this case resulting in a competitive computation time compared to EGNA_fs and UMDA.

Fig. 14 shows a critical difference diagram to identify the significant differences in the results shown in Fig. 10 for the

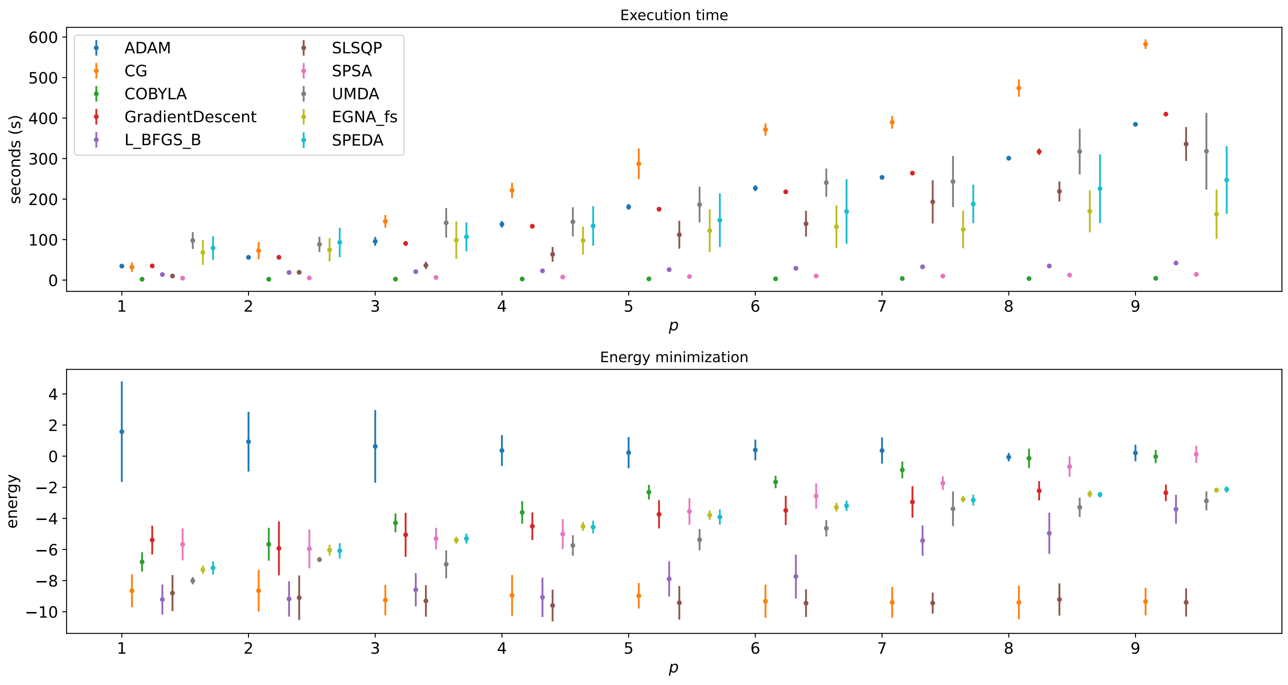

Fig. 10. Comparison of the different EDA variants with the other state-of-the-art optimizers for the VQE TwoLocal *ansatz* parameter tuning. The top panel shows the mean computation time and standard deviation after running each algorithm 25 independent times for different number of layers $p \in \{1, \dots, 9\}$. The bottom panel shows the mean best expectation value achieved (Eq. 1) and standard deviation after running each algorithm 25 independent times for different number of layers $p \in \{1, \dots, 9\}$.

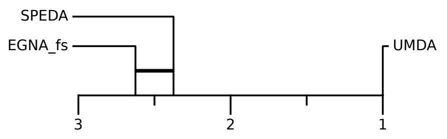

Fig. 11. Critical difference diagram for the VQE TwoLocal *ansatz* parameter tuning in terms of expectation value minimization using three EDA variants.

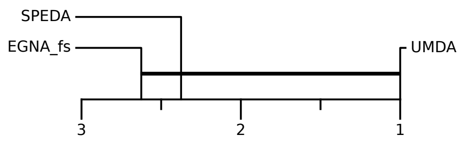

Fig. 12. Critical difference diagram for the VQE TwoLocal *ansatz* parameter tuning ($p \geq 6$) in terms of expectation value minimization using the three EDA variants.

expectation value minimization and $p \in \{1, \dots, 9\}$. The EDA variants are beaten by three gradient-based approaches: CG, L_BFGS_B and SLSQP, which also had a good performance in the case of QAOA. However, it is worth highlighting the computation time demand of CG for $p \geq 1$ and SLSQP for $p \geq 6$, which is improved by EGNA_fs and SPEDA for $p \geq 6$. For a large number of layers ($p \geq 8$) the expectation value minimization of L_BFGS_B is slightly better compared to the EDA variants, and its computation time is one of the best in the overall comparison. Despite the fact that the gradient-free optimizers achieve a low computation time compared to their competitors, the mean expectation value is always worse than that found by the different EDA variants.

From this analysis we conjecture that EGNA_fs and SPEDA are competitive optimizers for large number of layers ($p \geq 5$) if a trade-off between computation time and expectation value minimization is desired, where its principal competitor is L_BFGS_B. SPEDA is a better approach to minimize the resources demand, as it needs less quantum circuit measurements compared to EGNA_fs. For lower values ($p < 6$), UMDA improves the expectation values achieved by SPEDA and EGNA_fs, being one of the best optimizers in the overall comparison, although its computation time is worse in general.

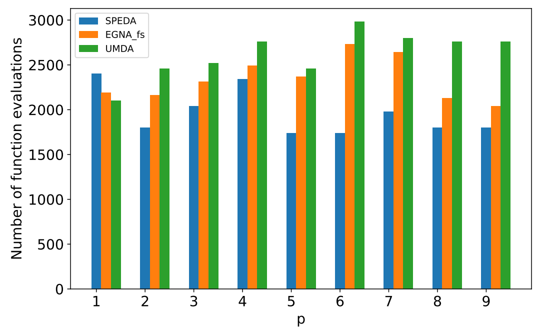

Fig. 13. Comparison of the mean number of cost function evaluations for the VQE TwoLocal *ansatz* parameter tuning after executing each EDA variant 25 independent times. Each cost function evaluation involves a new *ansatz* parameter configuration and measuring the quantum circuit N times.

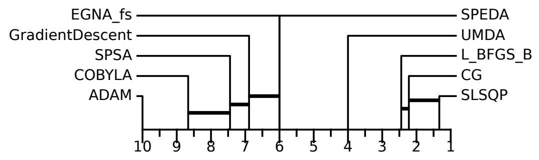

Fig. 14. Critical difference diagram for the VQE TwoLocal *ansatz* parameter tuning in terms of expectation value minimization comparing the different optimizers including the EDA variants.

TABLE I
MEAN EXPECTATION VALUE (EQ. 1) ACHIEVED BY DIFFERENT OPTIMIZERS FOR DIFFERENT VALUES OF $\omega \in [0, 1]$, WHERE BEST VALUES ARE HIGHLIGHTED IN BLUE.

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
ADAM	-1.12	-1.06	-1.09	-1.07	-1.09	-1.09	-1.08	-1.09	-1.09
AQGD	-1.43	-1.23	-1.16	-1.10	-1.09	-1.08	-1.08	-1.08	-1.08
CG	-1.03	-1.06	-1.10	-1.09	-1.08	-1.09	-1.09	-1.09	-1.09
COBYLA	-1.42	-1.27	-1.18	-1.13	-1.11	-1.09	-1.09	-1.09	-1.09
EGNA_fs	-1.44	-1.26	-1.17	-1.13	-1.11	-1.10	-1.09	-1.09	-1.09
GradientDescent	-1.22	-1.08	-1.08	-1.09	-1.09	-1.08	-1.09	-1.09	-1.09
L_BFGS_B	-1.11	-1.07	-1.08	-1.09	-1.09	-1.09	-1.08	-1.09	-1.09
SLSQP	-1.08	-1.06	-1.08	-1.07	-1.09	-1.09	-1.09	-1.09	-1.09
SPEDA	-1.45	-1.26	-1.18	-1.13	-1.11	-1.10	-1.09	-1.09	-1.09
SPSA	-1.42	-1.25	-1.17	-1.12	-1.10	-1.09	-1.09	-1.09	-1.09
UMDA	-1.43	-1.26	-1.17	-1.13	-1.11	-1.10	-1.09	-1.09	-1.09

C. *Molecule simulation with parametric quantum noise*

Simulating molecules behaviour in the area of quantum chemistry has gained a lot of attention in the last years due to its advantage compared to the classical computation [27]. In this section, the VQE TwoLocal *ansatz* is used to simulate the hydrogen molecule (H_2), where the objective is to find the ground state of the Hamiltonian that defines the molecule. Furthermore, this simulation has been carried out considering different intensities of a simulated quantum noise channel. The depolarized noise [41], parameterised by $\omega \in [0, 1]$ has been used, where $\omega = 0$ implies no quantum noise and $\omega = 1$ implies the maximum noise.

Table I shows the mean expectation value (Eq. 1) achieved by the different optimizers for different values of ω. Note that the COBYLA and EDA variants are the optimizers which more resilience offer to quantum noise, being both able to outperform the results of their competitors. AQGD is the worst performing optimizer in general. Note that, for high noise intensities ($\omega \geq 0.7$), all the algorithms tend to converge to the same solutions, but for small ones ($\omega \rightarrow 0$), a greater difference is noted between the results of the EDA variants and the rest of competitors.

D. *EDA hyper-parameter tuning*

The results shown in Section IV-A and Section IV-B show that the higher the number of layers (p), the higher the expectation value $E(\cdot)$ is obtained being not able to reach the values found in the cases of lower number of layers. This factor is directly correlated to the population size (μ) defined for the algorithm. In this section, we will comment on the relationship between the hyper-parameters p and μ for the specific case of SPEDA for QAOA, since UMDA and EGNA have an equivalent behaviour in this aspect for both *ansatz*.

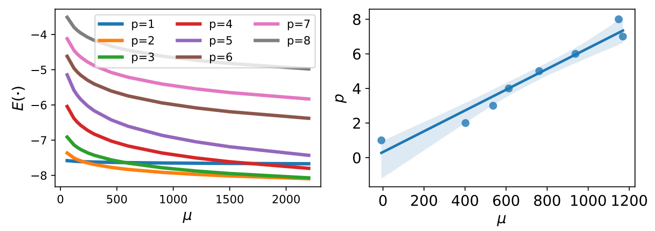

Fig. 15. Left panel shows the mean expectation value tendency for different values of p and μ. Right panel shows the linear dependency between the number of layers and the optimum values of μ found. Both panels analyze the QAOA and SPEDA case.

Fig. 15 (left) shows that regardless of the number of layers (p) the expectation value is reduced by increasing the population size (μ). Identifying the sweet spot between minimizing $E(\cdot)$ but also μ, will lead to find the optimum value of μ. The higher the μ values, the more evaluations are needed, and thus, a higher computation time is required. Following a similar procedure as in the elbow method for clustering, we identify the most interesting population sizes for each p in Fig. 15 (right). A linear dependency is observed between both parameters although this is an approximation.

V. CONCLUSIONS

In this work, a deep study on the use of different variants of EDAs for the VQA *ansatz* parameter tuning has been performed, where the algorithms have been compared to other state-of-the-art gradient-based and gradient-free alternatives widely used. The UMDA, EGNA with a fixed structure (EGNA_fs) and SPEDA algorithms were tested to tune the parameters of the QAOA and VQE *ansatz*.

In the case of QAOA, the three EDA alternatives offer similar solutions in terms of expectation value minimization, but EGNA_fs is the one which needs less computation time for convergence. The results have been validated to test the null hypothesis of equal mean results versus different means among the algorithms.

In the case of VQE, UMDA offers a statistically significant advantage regardless of the number of layers, but requires a longer computational time. However, for large values of p, this advantage is not statistically significant, and SPEDA achieves a notable improvement in computation time compared to the other algorithms, offering competitive expectation value results and needing the least cost function evaluations, compared to EGNA_fs and UMDA.

While it is not the objective of this study to find the best EDA variant for the VQAs *ansatz* parameter tuning, we have found that all three EDA variants perform better than other gradient-free algorithms, and achieve competitive solutions with gradient-based ones. UMDA stands out for the quality of the solutions found, although it suffers from computational speed. EGNA_fs is the fastest in terms of computation time, but SPEDA uses the fewest number of cost function evaluations. Moreover, the three EDA variants, together with COBYLA, have shown a better quantum noise resilience compared to the other competitors for the case of molecule simulation.

As future steps in this research line we propose to study (i) the different EDA variants for VQAs in which other types of quantum noise is present; (ii) the presence of Barren Plateau in the different scenarios; and (iii) the learned graph for larger number of layers.

Acknowledgment

We would like to thank CITIUS and CESGA for access to the computers to run the experiments and IBM Quantum Researchers Program for access to advanced services.

Code Availability

The code of UMDA, EGNA and SPEDA is available in the EDAspy Python package, which can be found at https://github.com/VicentePerezSoloviev/EDAspy and downloaded from Pypi. Additionally, UMDA approach is already available as an optimizer in Qiskit [37] library, and SPEDA and EGNA approaches will be merged soon.

References

[1] L. Gyongyosi and S. Imre, "A survey on quantum computing technology," *Computer Science Review*, vol. 31, pp. 51–71, 2019.

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

[3] V. P. Soloviev, C. Bielza, and P. Larrañaga, "Quantum-Inspired Estimation of Distribution Algorithm to solve the travelling salesman problem," in *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2021, pp. 416–425.

[4] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Reviews of Modern Physics*, vol. 90, no. 1, p. 015002, 2018.

[5] M. Cerezo, A. Arrasmith, R. Babbush *et al.*, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[6] L. Zhu, H. L. Tang, G. S. Barron, F. Calderon-Vargas, N. J. Mayhall, E. Barnes, and S. E. Economou, "Adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer," *Physical Review Research*, vol. 4, no. 3, p. 033029, 2022.

[7] V. P. Soloviev, C. Bielza, and P. Larrañaga, "Quantum approximate optimization algorithm for Bayesian network structure learning," *Quantum Information Processing*, vol. 22, no. 19, 2023.

[8] S. Wei, H. Li, and G. Long, "A full quantum eigensolver for quantum chemistry simulations," *Research*, vol. 2020, 2020.

[9] R. Shaydulin and Y. Alexeev, "Evaluating quantum approximate optimization algorithm: A case study," in *2019 Tenth International Green and Sustainable Computing Conference*. IEEE, 2019, pp. 1–6.

[10] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature Communications*, vol. 9, no. 1, pp. 1–6, 2018.

[11] X. Bonet-Monroig, H. Wang, D. Vermetten *et al.*, "Performance comparison of optimization methods on variational quantum algorithms," *arXiv preprint arXiv:2111.13454*, 2021.

[12] Z. Beheshti and S. M. H. Shamsuddin, "A review of population-based meta-heuristic algorithms," *International Journal of Advanced Software Computing Applications*, vol. 5, no. 1, pp. 1–35, 2013.

[13] P. Larrañaga and J. A. e. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Springer, 2001.

[14] H. Mühlenbein, J. Bendisch, and H.-M. Voigt, "From recombination of genes to the estimation of distributions II. Continuous parameters," in *International Conference on Parallel Problem Solving from Nature*. Springer, 1996, pp. 188–197.

[15] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[16] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña, "Optimization in continuous domains by learning and simulation of Gaussian networks," *Proceedings of the Genetic and Evolutionary Computation Congress*, 2000.

[17] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña, "Optimization by learning and simulation of Bayesian and Gaussian networks," *Technical Report, Department of Computer Science and Artificial Intelligence, University of the Basque Country*, 1999.

[18] M. Gallagher, M. R. Frean, and T. Downs, "Real-valued evolutionary optimization using a flexible probability density estimator." in *GECCO*, vol. 99, 1999, pp. 840–846.

[19] P. A. Bosman and D. Thierens, "Advancing continuous ideas with mixture distributions and factorization selection metrics," *Optimization by Building and Using Probabilistic Models (OBUPM) 2001*, p. 208–212, 2001.

[20] N. Luo and F. Qian, "Evolutionary algorithm using kernel density estimation model in continuous domain," in *2009 7th Asian Control Conference*. IEEE, 2009, pp. 1526–1531.

[21] P. A. Bosman and D. Thierens, "Ideas based on the normal kernels probability density function," Utrecht University, Technical Report UU-CS-2000-11, 2000.

[22] V. P. Soloviev, C. Bielza, and P. Larrañaga, "Semiparametric estimation of distribution algorithms for continuous optimization," *(submitted)*, 2022.

[23] G. Acampora, R. Schiattarella, and A. Vitiello, "Using quantum amplitude amplification in genetic algorithms," *Expert Systems with Applications*, vol. 209, p. 118203, 2022.

[24] G. Acampora and A. Vitiello, "Implementing evolutionary optimization on actual quantum processors," *Information Sciences*, vol. 575, pp. 542–562, 2021.

[25] V. P. Soloviev, P. Larrañaga, and C. Bielza, "Quantum parametric circuit optimization with estimation of distribution algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022, pp. 2247–2250.

[26] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

[27] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, no. 1, pp. 1–7, 2014.

[28] K. Bharti, A. Cervera-Lierta, T. H. Kyaw *et al.*, "Noisy intermediate-scale quantum algorithms," *Reviews of Modern Physics*, vol. 94, no. 1, p. 015004, 2022.

[29] D. P. Kingma and J. Ba, "ADAM: A Method for Stochastic Optimization," in *Proceedings of International Conference on Learning Representations, ICLR 2015*, 2015.

[30] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, pp. 409–435, 1952.

[31] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[32] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[33] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numerica*, vol. 4, pp. 1–51, 1995.

[34] M. J. Powell, "Direct search algorithms for optimization calculations," *Acta Numerica*, vol. 7, pp. 287–336, 1998.

[35] J. C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 332–341, 1992.

[36] J. Ceberio, A. Mendiburu, and J. A. Lozano, "A roadmap for solving optimization problems with estimation of distribution algorithms," *Natural Computing*, pp. 1–15, 2022.

[37] G. Aleksandrowicz, T. Alexander, P. Barkoutsos *et al.*, "Qiskit: An open-source framework for quantum computing," 2021.

[38] F. G. Brandao, M. Broughton, E. Farhi, S. Gutmann, and H. Neven, "For fixed control parameters the quantum approximate optimization algorithm's objective function value concentrates for typical instances," *arXiv preprint arXiv:1812.04170*, 2018.

[39] V. Akshay, D. Rabinovich, E. Campos, and J. Biamonte, "Parameter concentrations in quantum approximate optimization," *Physical Review A*, vol. 104, no. 1, p. L010401, 2021.

[40] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[41] M. A. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Washington DC: American Association of Physics Teachers, 2002.