Algorithm selection and configuration for Noisy Intermediate Scale Quantum methods for industrial applications

Moussa, C.

Chapter 4

Unsupervised strategies for setting optimal parameters in Quantum Approximate Optimization Algorithm

QAOA stands out as a promising approach to tackling combinatorial problems. In Chapter 3, we assumed we found good parameters by optimization. However, finding these appropriate parameters is in general a difficult task [23]. In this chapter[1], we study a few unsupervised machine learning approaches for setting these parameters without optimization. These angle-finding strategies can be used to reduce calls to quantum circuits when leveraging QAOA as a subroutine. We showcase them within a recursive algorithm featuring QAOA as a subroutine called Recursive-QAOA. We applied it for MaxCut over 200 Erdős-Rényi graphs limiting the QAOA depth to 3 where the number of QAOA parameters used per iteration is limited to 3. We obtain similar performances to the case where we extensively optimize the angles, hence saving numerous circuit calls.

[1]Contents of this chapter are based on [138];Charles Moussa, Hao Wang, Thomas Bäck, and Vedran Dunjko. Unsupervised strategies for identifying optimal parameters in quantum approximate optimization algorithm. *EPJ Quantum Technology*, 9(1), 2022.

4.1 Introduction

QAOA exhibits a few properties that make it interesting for combinatorial optimization such as a perfect theoretical performance at infinite depth [57], a sampling advantage [58] and the concentration of parameters [26]. The latter suggests that optimal parameters found for one instance can be reused in another. Most importantly, this means we can reduce the classical optimization loop and number of calls to a quantum device (saving runtime of QAOA-featured algorithms).

Many works have studied or illustrated this concentration property [26, 207, 100, 111, 61, 167, 4, 187, 46]. However, in many algorithms which feature QAOA as a subroutine [113, 74, 139, 178, 29], many distributions of instances are generated and several areas of parameter concentrations may arise. Hence, balancing between finding good QAOA parameters and reducing circuit calls will be key to QAOA-featured algorithms.

In this chapter, we apply unsupervised learning for setting QAOA angles, namely clustering. Our main contributions are as follows:

- We consider different approaches for the problem of setting QAOA angles with clustering: using directly the angle values, instance features, and the output of a variational graph autoencoder as input to the clustering algorithm.

- We analyze our methods by comparing them on two types of problems: Max-Cut on Erdős-Rényi graphs and Quadratic Unconstrained Binary Problems on random dense matrices.

- We demonstrate that our techniques can be used to learn to set QAOA parameters with respectively a less than $1 - 2\%$ reduction (in relative value) in approximation ratio in cross-validation while reducing circuit calls.

- We show that leveraging instance encodings for angle-setting strategies yields better results than using angle values only.

- Finally, we demonstrate their usage in Recursive-QAOA (RQAOA) [29] up to depth 3 on the Erdős-Rényi graphs. We limit the number of QAOA circuit calls per iteration to 3 (in contrast to a de novo optimization which would require many more calls), and achieve a 0.94 median approximation ratio. With our approaches, we obtain similar performances to the case where we extensively optimize the angles, hence saving numerous circuit calls.

The structure of the chapter is as follows. Section 4.2 provides the necessary background and related works. Section 4.3 analyses the optimal angles found in both problems, pointing to concentration effects and the suitability of clustering. Section 4.4 shows different unsupervised learning strategies using different data encoding for clustering and the comparison between them. Section 4.5 sums up our experiments on RQAOA. We conclude with a discussion in Section 4.6.

4.2 Background

4.2.1 QUBO and QAOA

Quadratic Unconstrained Binary Optimization (QUBO) problems are specified by the formulation $\min_{x\in\{0,1\}^n} \sum_{i\leq j} x_i Q_{ij} x_j$ where n is the dimensionality of the problem and $Q \in \mathbb{R}^{n\times n}$. QUBO can express an exceptional variety of combinatorial optimization (CO) problems such as Quadratic Assignment, Constraint Satisfaction Problems, Graph Coloring, and Maximum Cut [106]. The QAOA algorithm [57] was designed with the goal to tackle CO problems as seen in chapter 2.4.

An interesting property of the algorithm is the concentration of the QAOA objective for fixed angles [26] due to typical instances having (nearly) the same value of the objective function. Additionally, the QAOA landscape is instance-independent when instances come from a «reasonable» distribution (with the number of certain types of subgraphs of fixed size themselves concentrate, which in turn implies the values concentrate). Hence, we can focus on finding good parameters on a subset of instances that could be re-applied to new ones, with a few extra calls to the quantum device in order to refine. As stated earlier, in the most general case, characterizing distributions which are «reasonable» may be involved, or even characterizing the distribution at hand may be hard. Previous work [26, 187, 46] referenced [4] reported concentrations over optimal parameters even when QAOA is applied on random instances. These distributions over optimal parameters are empirically shown to behave non-trivially with respect to n. [4] pointed out this problem as «folklore of concentrations» .

Hence, even though angles concentrate in many settings asymptotically, for finite-size problems, different areas of concentration may rise. Therefore, choosing good angle values is challenging, especially when considering the runtime of quantum algorithms. As such, some studies built on this property and resorted to using Machine Learning (ML) or characterizing instances by some properties for finding good QAOA parameters. We present a few of them in the next subsection.

4.2.2 Related Work

Many previous works have extensively employed the concentration property [207, 100, 111, 61, 167, 4, 187, 46]. Among them, a few employed ML or designed strategies for setting good QAOA parameters for different objectives. In [100], a simple kernel density model was trained on the best angles and instances solved by QAOA to exhibit better QAOA optimization than the Nelder-Mead optimizer. Parameter fixing strategies for QAOA are also studied in [111, 207] where the best-found angles at depth p are used as starting points for depth $p+1$ before using a classical optimizer.

[187] present a strategy to find good parameters for QAOA based on topological properties of the problem graph and tensor network techniques. [61] point out that the success of transferability of parameters between different problem instances can be explained and predicted based on the types of subgraphs composing a graph. Finally, meta-learning is used in [167] to learn good initial angles for QAOA. They focused on initialization-based meta-learners in which a single set of parameters is used for a distribution of problems as initial parameters of a gradient-based optimizer. The meta-learner is a simple neural network that takes as inputs some meta-features of the QAOA circuit to predict the angles to apply (depth and which angle to output the value). However, no instance-related features are involved in their work.

In our case, we focus on clustering with the goal of proposing many parameter values to try for new QAOA circuits. In contrast to all the approaches we discussed above, we do not use a classical optimization loop after setting them. Hence, our approaches allow balancing between circuit calls of small quantum computers and performances. Such settings for instance naturally occur in divide-and-conquer-type schemes to enable smaller quantum computers to improve optimization [113, 74, 139, 178], or in Recursive-QAOA [29] as we demonstrate later.

4.3 Revisiting the concentration property

In contrast to previous related works, we propose unsupervised approaches that also exploit these concentration effects. We take a data-driven approach where from examples of good angles, we will infer new good angles for new instances. Namely, we use clustering in order to obtain clusters that can be used to reduce calls to the quantum device to small numbers (in our case, less than 10) when applying QAOA on new instances, without further optimization.

We take a usual ML approach to this problem. First, from generated instances,

we apply *exploratory data analysis* [85] (EDA) that suggests clustering may be a good approach for recommending good angles to new instances. Namely, we look at the density of angle values and apply t-distributed stochastic neighbor embedding (t-SNE) [191] for visualizing concentration effects. t-SNE is a nonlinear dimensionality reduction technique for mapping high-dimensional data to a lower d-dimensional space (typically $d \in \{2, 3\}$). Briefly, this method constructs a probability distribution to measure the similarity between each pair of points, where closer pairs are assigned with a higher probability. Then, in the lower-dimensional space $\mathbb{R}^d$, we use a Student t-based distribution to quantify the similarity among the embeddings of the original data points. Finally, the optimal embeddings are chosen by minimizing the Kullback–Leibler divergence between the similarity distributions in the original and the lower-dimensional spaces. We follow by explaining how clustering is used in order to recommend angles for new instances. The approaches we outline differ in input to the clustering algorithm. We consider clustering from the angle values directly but also from instance encodings. Finally, we compare these approaches allowing us to provide recommendations for their usage.

4.3.1 Data generation

We generated two datasets that show different concentration behavior. The first one consists of 200 Erdős-Rényi graphs for MaxCut problems. The graphs have $10, 12, 14, 16$, and 18 nodes. We utilized the following probabilities of edge creation: $0.5, 0.6, 0.7, 0.8$. We have generated 10 graphs per number of nodes and probability. The second dataset consists of 100 instances of QUBO problems, specified by their weight matrix Q (20 per aforementioned number of nodes). Their coefficients are sampled uniformly in $[-1, 1]$. For the purpose of computing approximation ratios, we are interested in C_{opt} – the maximal value of the MaxCut (or QUBO) – over all possible bit configurations, and as a reference, this was computed using brute-force. Our experiments were achieved using a classical simulator.

We then obtained for each problem the best set of angles by running the BFGS optimizer [35] 1000 times for $p = 1, 2, 3$, and selecting the ones which achieve the best QAOA objective. BFGS with random restarts is deemed a very good optimizer for continuous differentiable functions [77]. These angles are saved as a database and apply unsupervised approaches to learn to set optimal angles for unseen instances. Our approach is clearly optimization method-specific but can be applied to other state-of-the-art optimizers. Different optimizers would give different data (as the optimizers

could fail to find the optimal QAOA parameters) but they can be combined and one would select the best set of angles found among all considered.

4.3.2 Exploratory Data Analysis

Having obtained the optimal angles, we apply EDA to observe concentration effects. We look at their corresponding performance ratios using the average cost yielded by QAOA for angles γ, β denoted with $E_{\gamma,\beta}(C)$. For MaxCut on unweighted Erdős-Rényi graphs, we compute approximation ratios as $\frac{E_{\gamma,\beta}(C)}{C_{opt}}$. This value is upper-bounded by 1, which is the optimal value. For QUBOs, we compute optimality gaps $\frac{C_{opt}-E_{\gamma,\beta}(C)}{C_{opt}}$ as the optima were all negative and the closer to 0, the better. We show boxplots in Fig 4.1 the ratios wrt depth. Increasing depth results in better ratios.

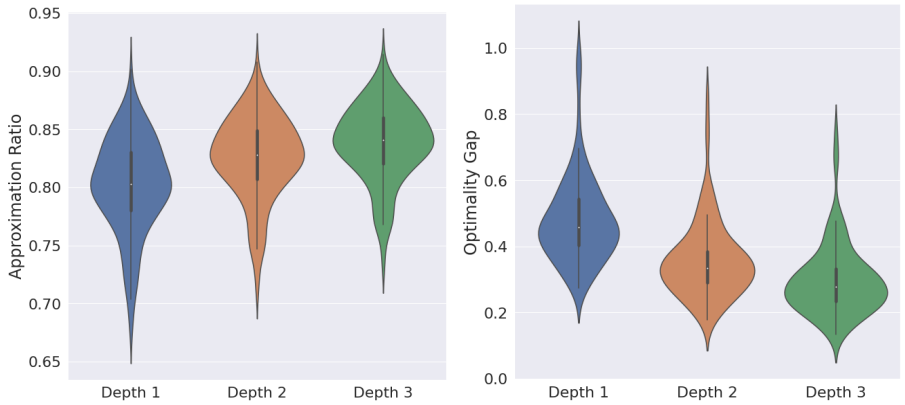


Figure 4.1: Violin plots of ratios on MaxCut and optimality gaps over QUBOs (bottom plot) for $p = 1, 2, 3$. The respective median by depth is $0.802954, 0.827901, 0.840478$ for MaxCuts and $0.457434, 0.335144, 0.278984$ for QUBOs, illustrating improved performances with increased depth.

Next, we looked at the distribution of γ_i, β_i values. Fig 4.2 shows that the concentration per each parameter is significant since their corresponding density functions are quite peaky. Also, we also observed multiple clusters of angles as the density functions are multimodal. Finally, we applied t-SNE with two components to visualize the angle values in $2D$ for $p = 2, 3$. This highlights potentially a number of clusters for each depth and problem. Note that it may be possible that we may not obtain global optima with these angles, or know if they are unique.

We notice that the probability of edge creation, represented by a different color, does not seem to influence the clusters. For dense QUBOs, we observe one important

cluster and a few instances that start to form another. Finally, in the case of the dense instances, we witness a more important spread in angle values at depth 1. This can be explained by differences between instances. Although the concentration effect is present, such an order of magnitude will impact the performances of parameter-setting strategies, and make an interesting playground to benchmark them.

Using clustering techniques can then reveal potential areas of QAOA angle values where good angles can be found to try on new instances. The angle values related to clusters can be used as recommendations for new instances. This becomes interesting as this enables lowering runtime and allow comparing based on function evaluations, or on the number of quantum circuit calls, in algorithms where QAOA would be used as a subroutine.

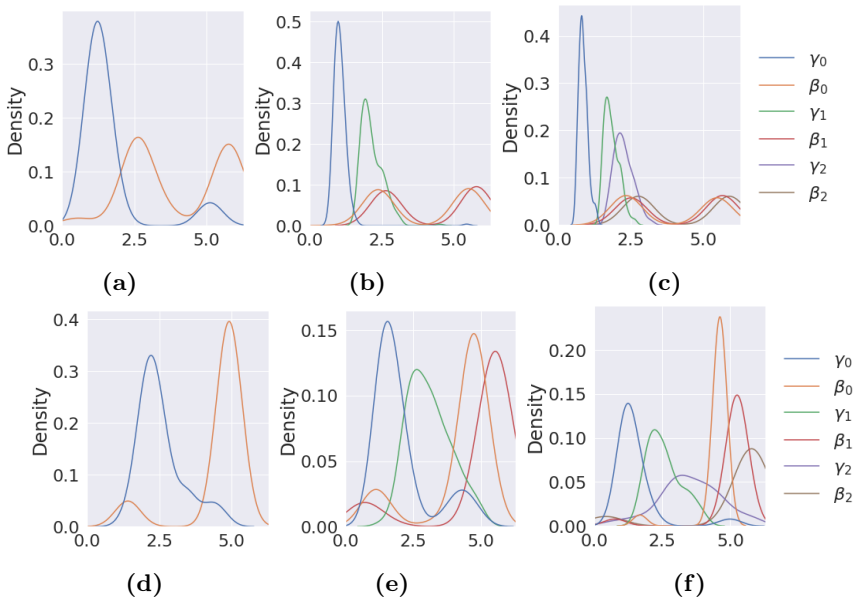


Figure 4.2: Distribution of angle values γ_i, β_i for each depth. Plots a), b) and c) concern MaxCut problems while the others refer to the dense QUBO matrices. We witness concentration effects of the angle values, suggesting the suitability of clustering as an angle-setting strategy.

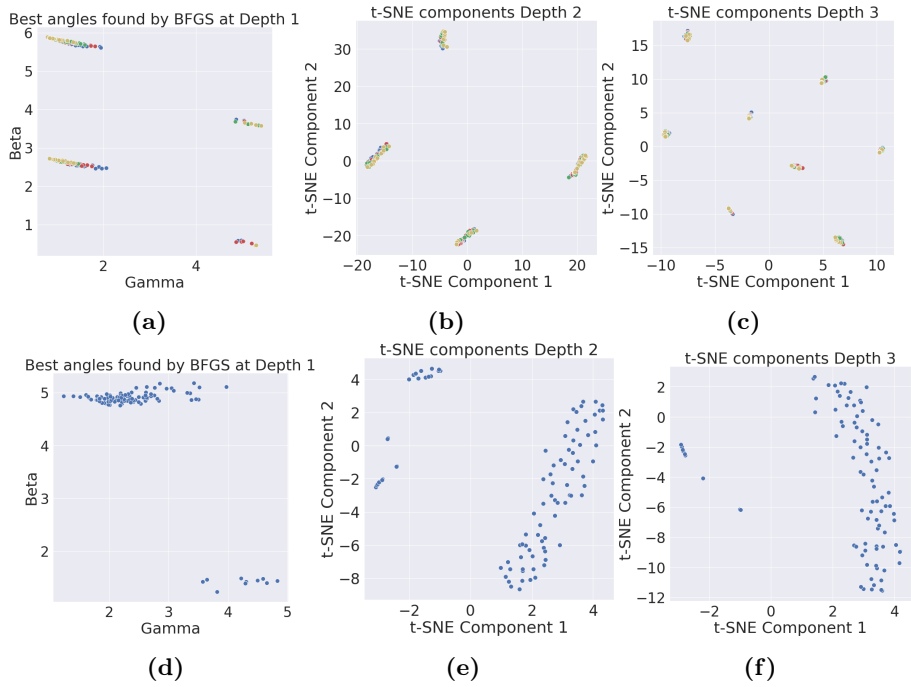


Figure 4.3: 2D angles visualization γ_i, β_i for each depth. Plots a), b) and c) concern MaxCut problems while the others refer to the dense QUBO matrices. For $p = 2, 3$, t-SNE is applied for projecting the angle values to 2D. Different areas of concentration are revealed again. We use different colors for differentiating the probability of edge creation of the Erdős-Rényi graphs, showing no correlation with clusters.

4.4 Clustering-based (unsupervised) learning for angles

As the EDA highlights a clustering effect, we propose different clustering approaches that use different data for angle recommendations. Namely, we describe first using the angle values directly for building clusters serving as angles to try. Then, we switch to using instance-related features. Finally, for the unweighted case, we use graph auto-encoders whose outputs can be used for clustering instead of computing graph features. In the following, we detail each clustering approach for flexible angle recommendation.

4.4.1 Identifying clusters of angles or problem instances

We first considered clustering using angle values. Given a database of optimal angles for Q problem instances $\{I_1, \cdots, I_Q\}, \{(\gamma^*, \beta^*)_1, \cdots, (\gamma^*, \beta^*)_Q\}$, this can be seen as computing or selecting a good set of angle values the database to apply on new instances. In this case, we do not use the problem instances during clustering. Given a user-specified number of angles to be tested K, this set of angle values is then applied to new QAOA circuits. To specify them, we can use a clustering algorithm on the database $\{(\gamma^*, \beta^*)_1, \cdots, (\gamma^*, \beta^*)_Q\}$. For instance, K-means [118] will output centroids to use directly as angle recommendations for QAOA on new instances. The K-means algorithm aims to partition a set of n data points x_i into K disjoint clusters C, characterized by the mean/centroid of the points within a cluster, denoted μ_j. The partition $P = \{P_1, P_2, \ldots, P_K\}$ $(\forall i \neq j \in [1..K], P_i \neq \emptyset, P_i \cap P_j = \emptyset, \cup_i P_i = \{x_i\}_{i=1}^n)$ is chosen by minimizing the within-cluster sum of squares, i.e., $\arg\min_P \sum_{i=1}^K \sum_{x \in P_i} ||x - \mu_i||^2$, where the centroid $\mu_i = |P_i|^{-1} \sum_{x \in P_i} x$. The algorithm iteratively updates the centroids by assigning each data point to its nearest centroid and computing the mean, until convergence.

To incorporate knowledge from instances when recommending angles, we change the data fed to the clustering algorithm. We distinguish computing instance features from learning an embedding, that is a user-defined $F-$dimensional representation or encoding of the instances as data. We denote an encoding of an instance I_t as $f(I_t)$. The angle recommendation framework using a clustering algorithm for such instance representation is presented in Alg. 1. First, clusters are learned from the encodings extracted from training data. Then, we find the instances in the database that are the closest in distance to the clusters, and their corresponding optimal angles [2]. The latter are then used for QAOA circuits on new instances, from which we keep the best QAOA output.

4.4.2 Instance encodings

In this work, we show two main approaches to encoding the instances for clustering. First, we computed a set of features following [52, 139]. Such features were used in [52] to decide among classical heuristics to solve MaxCut and QUBO problems. Inspired by [52], the features were also used for choosing when to apply QAOA against a classical approximation algorithm [139]. For Erdős-Rényi graphs, we took the graph

[2] Since the clustering algorithm outputs encodings that do not contain QAOA angle information, we use the QAOA angles of the closest training instances to the clusters.

Algorithm 1: K-angle recommendation framework for QAOA. Comments are provided in curly brackets to explain the key concepts with the associated pseudocode.

Input: Clustering algorithm, number of clusters K,
Training Data: $\{I_1, \cdots, I_Q\}; \{(\gamma^*, \beta^*)_1, \cdots, (\gamma^*, \beta^*)_Q\}$,
Testing Data: $\{I_1^{'}, \cdots, I_R^{'}\}$,

Initialize $anglesToRecommend = [], encodings = []$
for $t = 1$ **to** Q **do**
  Compute $f(I_t)$ and append to *encodings*. {First, we compute the encodings from the training instances before applying the clustering algorithm.}
end for
Apply Clustering algorithm on *encodings*
for $c = 1$ **to** K **do**
  Get encoding of cluster f^c from the clustering algorithm
  Get closest point in *encodings* to f^c and extract index i_c {Here, we find the instances in the database that are the closest in distance to the clusters, and their corresponding optimal QAOA angles.}
  Append $(\gamma^*, \beta^*)_{i_c}$ to *anglesToRecommend*
end for
{Finally, the QAOA angles in *anglesToRecommend* are used for QAOA circuits on the test instances, from which we keep the best QAOA output.}
for $t = 1$ **to** R **do**
  $bestOutput_t = \inf$
  for $c = 1$ **to** K **do**
    Apply QAOA on $I_t^{'}$ with $anglesToRecommend[c]$ {We can then compute the QAOA expectation for the test instance $I_t^{'}$ using the QAOA angles $anglesToRecommend[c]$, denoted $E_{anglesToRecommend[c]}(C_{I_t^{'}})$.}
    if $E_{anglesToRecommend[c]}(C_{I_t^{'}}) < bestOutput_t$ **then**
      $bestOutput_t = E_{anglesToRecommend[c]}(C_{I_t^{'}})$
    end if
  end for
end for

density, the logarithm of the number of nodes and edges, the logarithm of the first and second-largest eigenvalues of the Laplacian matrix normalized by the average node degree and the logarithm of the ratio of the two largest eigenvalues. For QUBOs, we reduced them to the MaxCut formulation and used the logarithm of the number of nodes, and the weighted Laplacian matrix eigenvalues-based features.

We also show how to use graph embeddings using Graph Neural Networks (GNNs) [205], avoiding the need for the user to compute the features. We employ

the Variational Graph Auto-Encoders (VGAE) [104]. This technique only works on unweighted graphs by its design principle. Consequently, we only applied it to the MaxCut instances later in this work. A VGAE learns latent embeddings $\mathbf{Z} \in \mathbb{R}^{N \times F}$ where F is the dimension of the latent variables and N the number of nodes. Given the adjacency matrix A and nodes feature vector X, the model outputs the parameters of a Gaussian distribution μ, σ for the latent representation generation. We feed to the model the Erdős-Rényi graphs, and we add as node features the degree of the nodes. Once learning is completed, we compute the embeddings by a common average readout operation [205, 196]. The latter operation can be defined as averaging the node embeddings for a graph with vertex set $\mathcal{V}$ $\frac{1}{|\mathcal{V}|}\sum_{n\in\mathcal{V}} Z_n$. This allows having a fixed dimension F for the encoding to be used by a clustering algorithm.

Having defined different strategies for clustering, we apply them to the data we generated and compare their performances. In the following section, we present our results obtained by taking a Machine Learning approach, starting from a simple baseline and cross-validating each method.

4.4.3 Results

In this section, we apply the above-mentioned proposed strategies to the generated data where EDA revealed different areas of concentration. As the first baseline for the angle-setting strategy, we experiment with simple aggregation of angle values (median and average). Then we follow this up with K-means applied on angle values by varying the number of clusters from 3 to 10 as the underlying clustering algorithm. Finally, we change the K-means data to cluster based on instance encodings instead of angle values. We computed first a set of graph features that were used in a previous study [135]. Then we investigate graph autoencoders to learn the encodings of the Maxcut instances. We cross-validate each method using 5-fold cross-validation where we report the ratios $\frac{(C_{opt}-E_{\gamma,\beta}(C))}{(C_{opt}-E^{cluster}_{\gamma,\beta}(C))}$ on test instances. A value higher than 1 would mean that the average cost yielded by clustering has improved over the one found by optimization. We also consider the case where one trains on smaller instances to apply to the bigger ones.

From angle values

As a simple baseline, we compute the average and the median of the optimal angles from the database $\{(\gamma^*, \beta^*)_1, \cdots, (\gamma^*, \beta^*)_Q\}$. From depth-aggregated results, averaging the angle values yielded a median ratio of 0.524 for MaxCut and 0.672 for QUBOs,

while taking the median values increased it to respectively 0.950 and 0.941. This can be explained by the fact that the median value is statistically more robust than the mean when handling data sets with large variability.

As expected with K-means, increasing the number of clusters yielded better median ratios. With $K = 10$, the median ratios are 0.998 and 0.985 on each dataset, a less than $1 - 2\%$ reduction in performances w.r.t. the optimal angles. Fig. 4.4 shows the improvement with an increased number of clusters. We observe also that with increased depth, median ratio performances are reduced. We conjecture that, when the dimension of the parameter space increases, more clusters are naturally needed to ensure a sensible recommendation.

Also, such a deterioration of performance w.r.t. circuit depth is more substantial on the QUBO instances than on the MaxCut ones, which can be explained by the clustering patterns in the MaxCut scenario being more significant and regular (Fig. 4.3). In addition, this observation suggests that for future work, for dense QUBO instances where the cluster center is not representative of all points pertaining to it, it is more reasonable to take a supervised learning method, which takes the problem instance as input and predicts the optimal angle values.

We also observed that, for the MaxCut problem, the cluster centroid of K-means can be quite distant from the data points when the number of clusters is small and the circuit depth is high. Particularly, this phenomenon deteriorates the median ratio by ca. 30% for 3 and 4 clusters with $p = 3$. Hence, we decided to take the closest data point to the centroid in each cluster as the recommendation, which solves this issue. For QUBOs, using the cluster centroids directly yields better results.

Overall, increasing the number of angles attempted will improve the quality of the QAOA output. Clearly, the results with less than 4 clusters present examples where the ratio is low, worsening the median performances. For instance, with 3 clusters on QUBOs, the median ratio is 0.915. In the context where the budget of quantum circuit calls is very limited, this could be problematic and call for more robust approaches. To this end, we consider using instance features for clustering.

From instance encodings

To witness whether using instance features can improve the quality of clustering, we divided the ratios obtained with instance features by the ones using angle values. We show these results in Fig. 4.5 and Fig. 4.6 where we can clearly see better ratios with less than 4 clusters, and similar results on average otherwise.

As for learned encodings or embeddings with auto-encoders, the GNN model con-

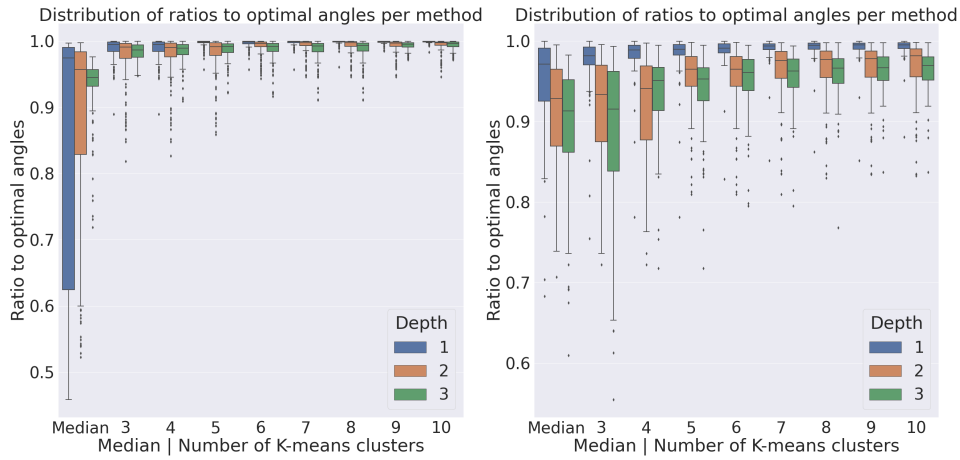


Figure 4.4: Boxplot visualization of ratios to optimal angles' expectation value per clustering method and depth on MaxCut (a) and dense QUBOs (b), when using the angle values. We show also the boxplots when computing the median angle values, yielding a median ratio of 0.524307 for MaxCut and 0.671572 for QUBOs. The median ratios are respectively 0.990579, 0.995610 and 0.998293 for 3, 5 and 10 clusters. For QUBOs, we get 0.956099, 0.970842, and 0.984787 taking the same number of clusters. With reference to the optimal angles' expectation value, this corresponds on average to a less than $1-2\%$ reduction in performances when using 10 clusters. For MaxCut, we had to use the closest data point in the dataset to the cluster, as it results in better performances. For instance, with 3 clusters at $p = 3$, the median ratio was 0.618275.

figuration we use is the same two-layer graph convolutional layer as [104]. Namely, the first one has 32 output-dimension using the ReLU activation function. This is followed by two 16-dimensional output layers for the generation of the latent variables. We train using Adam with a learning rate of 0.01 for 100 epochs and batch size set to the dataset size. Our implementation uses the Deep Graph Library (DGL) [196]. The embeddings obtained by averaging are of dimension $F = 16$. This allows having a fixed dimension for the encoding as input of the same K-means strategy described above. We observe in Fig. 4.7 that the results are similar to the ones obtained using instance features. Yet, in some instances, we see better results. Hence, many clustering results can be combined to improve the performances in ratios canceling each other's weaknesses at the cost of trying more angles to find the best ones. As future work, we could also decide which heuristic to use depending on a given test instance by using a ML model.

Finally, our approaches can save numerous circuit calls compared to de novo op-

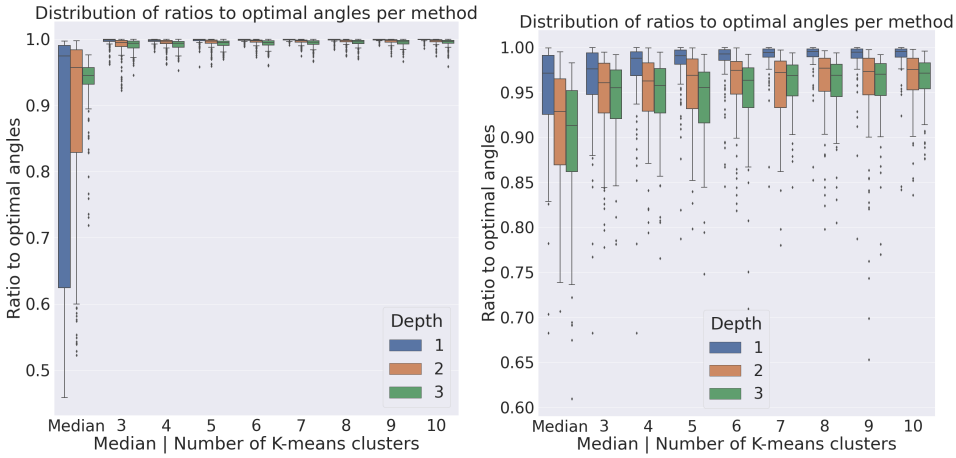


Figure 4.5: Boxplot plot visualization of ratios to optimal angles per clustering method and depth on MaxCut (a) and dense QUBOs (b), when using instance features. For Erdős-Rényi graphs, K-means yielded ratios $0.996214, 0.996368$ with $3, 4$ clusters and 0.998429 with 10. On dense QUBOs, we obtained respective median ratios of $0.963129, 0.971778$ and 0.982964. With reference to the optimal angles' expectation value, this corresponds on average to a less than $1 - 2\%$ reduction in performances when using 10 clusters.

timization. The median numbers of circuit calls for the BFGS runs giving the best QAOA angles were $56, 150, 320$ for each depth respectively on MaxCut and $44, 132, 252$ for QUBO, while in the cluster approach, the number of calls is always the cluster size, which is considerably smaller than the cost of BFGS. Instance size does not seem to affect the number of circuit calls by BFGS. In our approaches, we limited circuit calls to 10 and we do not need multiple restarts.

4.4.4 Aggregating results

Following the presentation of the different clustering approaches, we compare their performances to determine which approach works best. We propose to take the Empirical Cumulative Distribution Functions (ECDF) of the ratios as the performance measure to compare those different approaches. Given a sample $\{r_i\}_{i=1}^{R}$ of the ratios and a value of interest $t \in [0, 1]$, ECDF is the fraction of the sample points less or equal to t: $F(t) = \frac{1}{R}\sum_i \mathbb{1}_{[0,r_i]}(t)$, where $\mathbb{1}$ denotes the indicator function, which returns one only if $t \in [0, r_i]$ and zero otherwise. They enable us to aggregate the results of the different numbers of clusters and depths. A better method will have more proportion

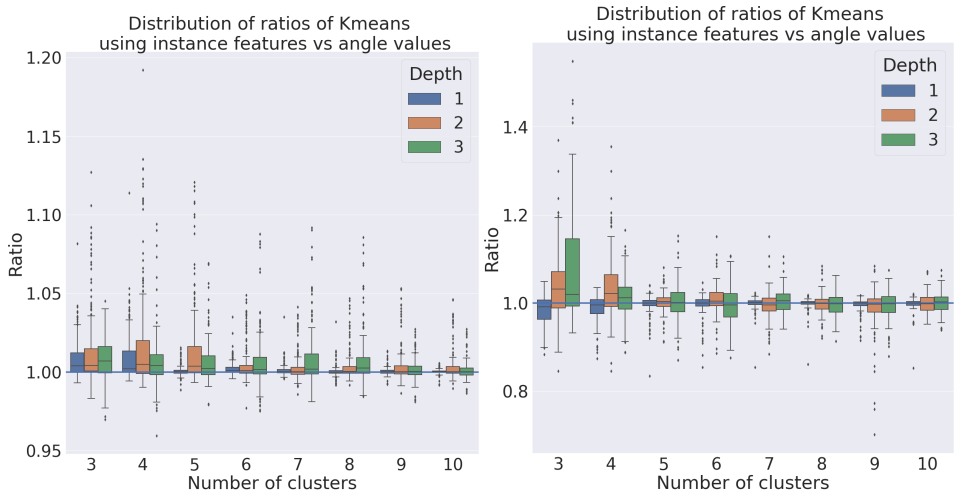


Figure 4.6: Boxplot of ratios comparing K-means with instance features against angle values on MaxCut (a) and QUBOs (b). A value higher than 1 (highlighted by a horizontal line) means using instance features results in better QAOA objective. We see an overall improvement with $3, 4$ clusters mainly at $p = 3$.

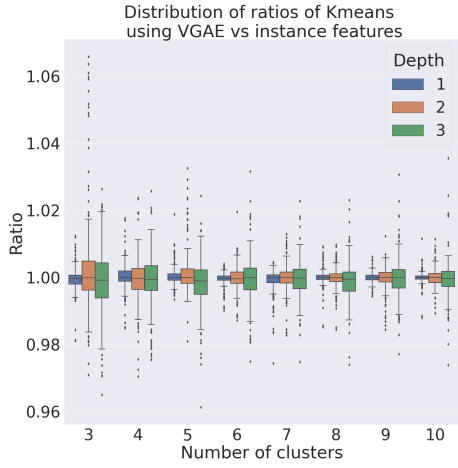
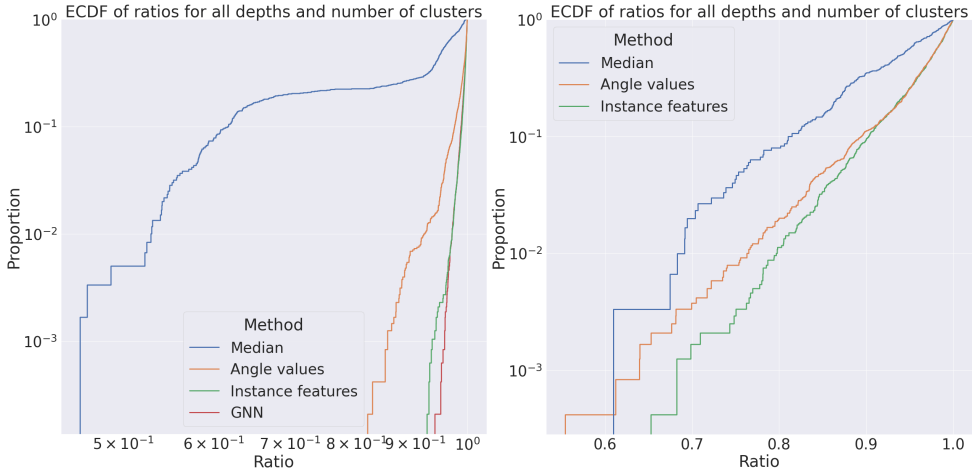


Figure 4.7: Boxplot visualization of ratios on MaxCut obtained using Variational Graph Auto-Encoders compared to using instance features. A value higher than 1 means using VGAE results in a better QAOA objective. Overall, performances are similar as the ratios are close to 1 on average.

of higher ratios, resulting in an ECDF curve located more to the right. From Fig. 4.8,

we observe that *using instance encodings is more successful in yielding better angles than using the angle values*. This is also witnessed in Fig. 4.9 with increased depth and a low number of clusters. Also, VGAE seems to be slightly better than instance features on the MaxCut problems. However, these methods can complement each other, especially as we do not need to increase the dataset size. Hence, combining them at the cost of circuit calls becomes an option for running QAOA, as we showcase with RQAOA in the next section.



(a) **(b)**

Figure 4.8: Empirical cumulative distribution functions of ratios to optimal angles' for all depths and number of clusters. A lower curve for an approach means better results when using it aggregating depths and number of clusters. We see for MaxCut (a) and QUBO problems (b) that instance features achieve better results, VGAE being competitive with instance features. When using 3 clusters, using VGAE on MaxCut instance and instance features for QUBOs leads to better ratios.

4.4.5 Case when test instances are bigger than training instances

One important consideration of these methods is to analyze scaling. This is relevant in settings where one is interested in solving larger instances given small ones. In our case, we apply these approaches in the case $K = 3$ by a $60-40\%$ train-test split. From Fig. 4.10 and 4.11, we find similar conclusions with respectively VGAE on MaxCut and instance features on the QUBO problems yielding better results. Note that we did not use the logarithm of the number of nodes and edges as features when using

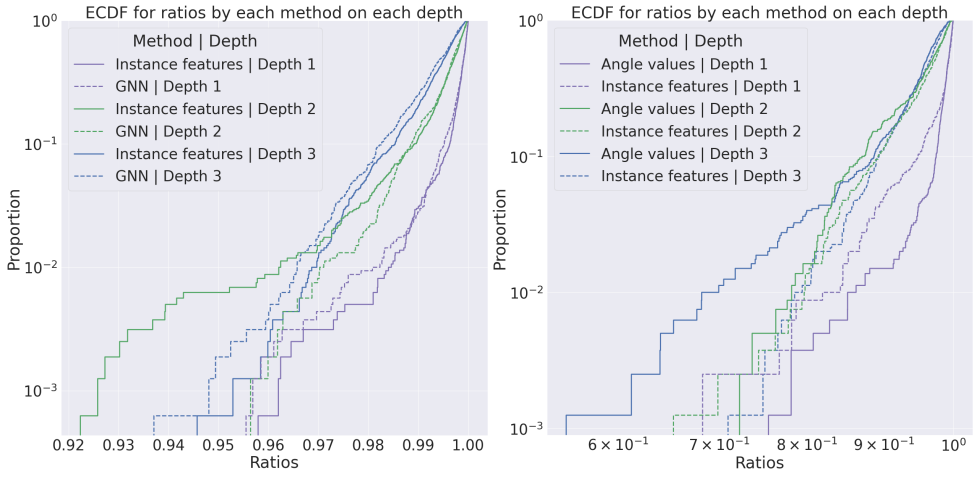


Figure 4.9: Empirical cumulative distribution functions of ratios to optimal angles per method and depth. The lower the curve, the better the method. In most cases, the curve corresponding to instance features was lower (except for QUBOs (b) at $p = 1$, and VGAE's curve was more competitive at $p = 2$ for MaxCut (a)). This was also the case when using 3 clusters.

instance features as the values between training and test are too different.

4.5 Demonstration with RQAOA

RQAOA [29] is a recursive algorithm where, given an Ising problem $\sum_{i,j} w_{ij} Z_i Z_j$, one starts by applying QAOA on the former. The quantum state output $|\gamma, \beta\rangle$ is then used to compute correlations $M_{ij} = \langle \gamma, \beta | Z_i Z_j | \gamma, \beta \rangle$. Then, variable elimination is carried out by selecting a pair of variables satisfying $(i_l, j_l) = \arg\max |M_{ij}|$, and substituting Z_{j_l} with $\text{sign}(M_{i_l,j_l}) Z_{i_l}$ in the Ising formulation. This reduces the number of variables by 1. We then get a new reduced problem and we reiterate the procedure for a user-defined number of iterations. The choice of iteration fixes the size of the final instance which is then solved using a brute-force (or some other classical) approach, and the substitutions are used onto it to obtain a final solution.

As RQAOA requires optimizing many QAOA instances that iteratively shrink in size, we demonstrate the application of our clustering approaches in this context. We do so for the MaxCut problems where we limit the number of iterations to half of the size of the Erdős-Rényi graphs. We do not consider the dense QUBOs as RQAOA

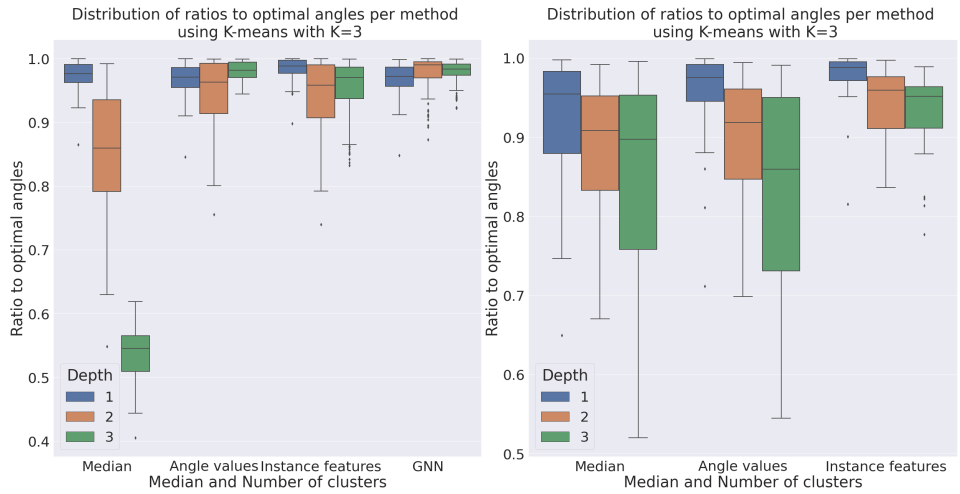
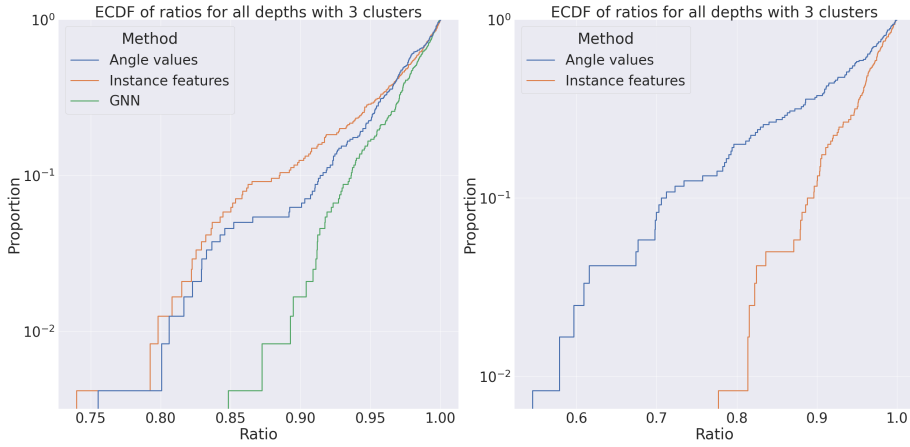


Figure 4.10: Boxplot of ratios comparing K-means $K = 3$ using instance features against angle values on MaxCut (a) and QUBOs (b). The ratios are obtained from 40% of the instances with the highest number of nodes. From depth-aggregated results, on MaxCut, using the median values gives a median ratio of 0.859316, 0.928519 with angle values, 0.976959 with instance features, and 0.981618 using VGAE. On QUBOs, we obtained respectively 0.926136 for the median of angle values, 0.936679 clustering with angle value and 0.963677 with instance features.

would reduce an original dense graph to non-dense intermediate subproblems not part of the database. As per the number of QAOA parameters attempted per iteration, we limit it to 3 and apply the three clustering approaches: angle-value, instance features, and VGAE-output based. We do so by using our previous database and training each method on all instances to get 3 QAOA parameter recommendations. The latter are then used for QAOA on the RQAOA-generated instances.

Fig. 4.12 shows that with the three approaches, we obtain a median 0.94117 approximation ratio with RQAOA. The minimal ratio obtained is 0.8367 and the optima were found on 33 instances. When looking at each method independently, we observe that the angle-value clustering performances at $p = 3$ are lower than the others. This is due to the fact that we use the K-means clusters directly as it allowed us to find more instances with a ratio of 1. Graph features and VGAE seem similar in performance, with a small advantage at depth 2 for VGAE. Looking at the frequencies where the best ratio by instance was obtained, VGAE is more successful. Respectively, each method achieves the best-found ratios over 88, 118, and 165 instances. Finally, we also tried

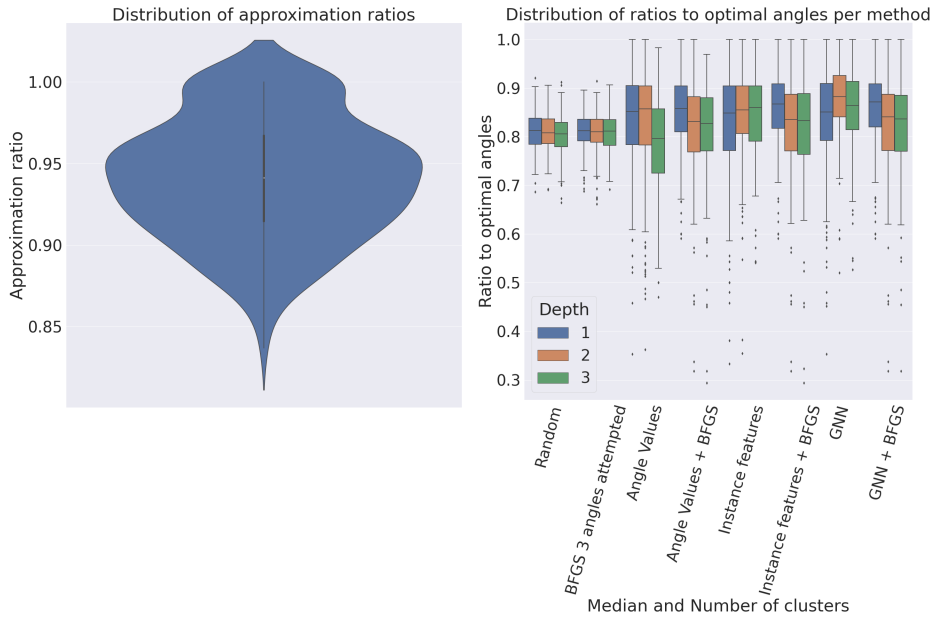


(a) **(b)**

Figure 4.11: Empirical cumulative distribution functions of ratios to optimal angles. The ratios are obtained from 40% of the instances with the highest number of nodes. Similar results to Fig. 4.8 and Fig. 4.9 are obtained.

using random angles, by sampling uniformly values in $[0, 2\pi]^p$ and optimizing further the angles from each approach with BFGS up to 100 iterations maximum. We clearly see better performances with clustering approaches compared to random angles. This is also the case when using BFGS (starting with random angles) limited to 3 circuit calls when optimizing, the same budget as our clustering-based approaches. Dividing the MaxCut ratios obtained with BFGS with the ones without further optimization yielded a median value of 1. Hence, the results were similar to the BFGS-optimized approaches, saving many circuit calls.

To conclude, our unsupervised approaches can be used to run quantum algorithms where QAOA is used as a subroutine. They are then considered as hyper-parameters that can be tweaked to achieve better performances for QAOA-featured algorithms, depending on a user-defined budget definition. In our RQAOA showcase, the maximal depth of QAOA, as well as the number of parameters to try at each iteration, was set to 3, and optimizing further did not improve. For MaxCut on Erdős-Rényi graphs, leveraging VGAE in RQAOA achieved the best ratios over 82.5% of the instances.



(a) **(b)**

Figure 4.12: The violin plot visualization represents ratios obtained on MaxCut using all three unsupervised approaches, using just 3 circuit calls per RQAOA iteration, without optimizing further with BFGS. A median ratio of 0.94117 was obtained. The boxplots represent the MaxCut ratios obtained using each approach. We added using random angles per iteration as a baseline as well as using BFGS (starting with random angles) with a budget of 3 angle values attempted during optimization, and we witness clustering approaches yielded better results. When dividing the ratios of the methods with the ones obtained by adding BFGS, we obtain a median ratio of 1, meaning we saved many circuit calls for similar results with clustering.

4.6 Discussion

In this chapter, we study different strategies for fixing the parameters of QAOA based on unsupervised learning. We focused on clustering given previous works highlighting the concentration property and exploratory data analysis of the best angles found for MaxCut on Erdős-Rényi graphs and dense QUBOs. We however use a methodology closer to machine learning by cross-validating compared to related work.

Furthermore, we demonstrated that these techniques can be leveraged to restrict the number of QAOA circuit calls to small numbers (less than 10) with a less than $1-2\%$ reduction in approximation ratio on average from the best angles found when cross-validating. We also showed how to compare different clustering strategies and

that leveraging instance encodings (by computing features or computing them with a model, in our case a VGAE) for angle-setting strategies yields better results than using angle values only. Although the VGAE embedding-based is quite competitive, we recommend using the simpler instance features in practice since the VGAE brings extra computation overhead. For generalization, in regard to the problem scale, both instance features- and VGAE-based clustering approaches manage to retain the performance for unseen problem instances larger than the training set. For dense QUBOs, increasing the clusters is less impactful compared to MaxCut, in which we conjecture that the clusters in QUBO are of large spread and less separable, hindering the performance of the clustering approach in higher dimensions. For both problems, it is necessary to increase the cluster number to retain a good performance when the circuit becomes deeper.

From an application perspective, we envision these techniques to be employed in algorithms where QAOA is run on a small part of the problem to solve such as divide-and-conquer [113, 74] and iterative algorithms [139, 178, 29]. Restricting to a few numbers of circuit calls will help decrease the runtime of quantum-featured or quantum-enhanced algorithms, making them closer to competing with classical heuristics. We showcased our approach in the context of Recursive QAOA as hyperparameters under a limited budget (QAOA depth and number of QAOA parameters per iteration limited to 3), where we were able to achieve a 0.94 median approximation ratio. With our approaches, we obtain quite comparable performance to the case where we extensively optimize the angles, hence saving numerous circuit calls.

For future work, other clustering techniques can be studied and extended to predict the angle values by instance in a semi-supervised approach, and for different problem instances. Plus, ML can be used to decide which heuristic to use depending on a given test instance. We also did not apply GNN to the dense QUBOs as graph autoencoders are mostly applied to unweighted graphs. Using VGAE that can reconstruct graph adjacency and node features is then another research direction. Since we use unsupervised methods, we expect the same methodology to be used on noisy hardware. Studying different approaches to resilience under different noisy settings would be also considered of main interest. Finally, these approaches can be studied within different QAOA-featured algorithms and under different settings (depth of QAOA, number of clusters, and instances properties to name a few).