Evolving objective function for improved variational quantum optimization

Ioannis Kolotouros* and Petros Wallden†

University of Edinburgh, School of Informatics, EH8 9AB Edinburgh, United Kingdom

(Received 12 November 2021; accepted 3 May 2022; published 21 June 2022)

A promising approach to useful computational quantum advantage is to use variational quantum algorithms for optimization problems. Crucial for the performance of these algorithms is to ensure that the algorithm converges with high probability to a near-optimal solution in a small time. In Barkoutsos et al. [Quantum 4, 256 (2020)], an alternative class of objective functions, called conditional value at risk (CVaR), was introduced and it was shown that they perform better than standard objective functions. Here we extend that work by introducing an evolving objective function, which we call ascending-CVaR and that can be used for any optimization problem. We test our proposed objective function in an emulation environment, using as case studies three different optimization problems: MaxCut, number partitioning, and portfolio optimization. We examine multiple instances of different sizes and analyze the performance using the variational quantum eigen solver with hardware-efficient ansatz and the quantum approximate optimization algorithm. We show that ascending-CVaR in all cases performs better than standard objective functions or the constant CVaR of Barkoutsos et al. [Quantum 4, 256 (2020)] and that it can be used as a heuristic for avoiding suboptimal minima. Our proposal achieves higher overlap with the ideal state in all problems, whether we consider easy or hard instances—on average, it gives up to ten times greater overlap at portfolio optimization and number partitioning, while it gives an 80% improvement at MaxCut. In the hard instances we consider, for the number partitioning problem, standard objective functions fail to find the correct solution in almost all cases, CVaR finds the correct solution at 60% of the cases, while ascending-CVaR finds the correct solution in 95% of the cases.

DOI: 10.1103/PhysRevResearch.4.023225

I. INTRODUCTION

We have recently entered the era where quantum computers have scaled up from small proof-of-principle devices to devices that are beyond the classical simulation limit, opening the prospect for providing computational speedups. However, we are still very far from the point that large fault-tolerant quantum computers are developed. Our period has been termed the noisy intermediate scale quantum (NISQ) device era [1] and refers to the time that the existing devices vary from ≈50 qubits of Google’s quantum-advantage1 experiment [2] to devices with O(1000) qubits that are anticipated in a horizon of five to ten years.

There are two paths forward for quantum computing. The long-term path requires intensifying efforts (theoretical and experimental) to overcome existing barriers and truly scale up these devices to the large fault-tolerant regime. The near-term one is to determine if and how these NISQ devices can be used directly and offer advantages for problems of practical importance. A promising approach in the latter path is the use of hybrid quantum-classical algorithms. A leading class of candidate algorithms, both due to the possible importance of the applications and the promise it shows, is the class of variational quantum algorithms for optimization problems.

One can divide variational quantum algorithms (see more details in Sec. II) into three main steps. The first step is to map the targeted problem to the mathematical task that these algorithms are designed to solve, which is the search for the ground-state energy of a Hamiltonian.2 The second step is a method to estimate the energy of a quantum state, given a (polynomial in the size of the input) number of copies. Finally, the third step consists of a parameterized family of quantum states (ansatz) and a classical optimizer that, given the above tools, efficiently outputs an approximation of the ground-state energy. This is done by finding the choice of parameters that leads to the quantum state that has the smallest energy.

The success of the algorithms depend on all those steps and extensive research on improving each of them exists, indicatively [3], using warm-starting to improve QAOA on low depth [4], improved QAOA by introducing a nonlocal version which outperformed classical QAOA on 3-regular graphs, introducing different procedures on how to optimize the variational parameters, and [7] using reinforcement learning to assist the classical optimization.

What we focus on in this contribution is the third part, and specifically on how to use the measurement outcomes performed in estimating the energy of a quantum state to (i) accelerate the speed and (ii) improve the accuracy so the

1Also known as quantum computational supremacy.

2Mathematically, this is simply evaluating the smallest eigenvalue of a Hermitian matrix.
classical optimizer finds (an approximation of) the ground state and thus solves the problem optimally. Prior to our work, inspired by statistical physics, Ref. [8] used a Gibbs objective function to improve the performance. Minimizing the infidelity between the parameterized state and a target state [9,10] appears to be another promising approach.

For classical optimization problems, the solution (ground state) is one of the computational basis quantum states. Preparing a quantum state that has big overlap with that state is sufficient to give a good and quick approximation of the ground state. For example, if one can achieve a constant but possibly small overlap with the correct solution, it is guaranteed after sampling this state a constant number of times to obtain at least one sample of the true ground state. In Ref. [11], the authors used this idea, and instead of evaluating the proximity of a quantum state to the desired (ground state) by minimizing the (overall) energy, they aimed to minimize the energy of the lowest tail of a quantum state. This, intuitively, would succeed more quickly in finding a quantum state that has a non-negligible overlap with the solution (but not necessarily very high overlap). This state, however, suffices to solve the problem. This intuition was also confirmed with numerical simulations. In other words, the cost function used in the classical optimizer to find the optimal parameters was not the energy of the quantum state but the tail of the corresponding distribution.

Inspired by this idea but also by adiabatic quantum computing [12], we consider here an evolving cost function. In our proposal, the way that the cost of a quantum state is computed dynamically changes during the classical optimization process. We start with a cost function as in Ref. [11], focusing on a small tail, but during the optimization process we gradually increase the tail (fraction of the distribution we count) until we reach a point that all the distribution is included, i.e., we measure the full expectation value of the energy (as in standard cost functions).

Our contributions:

1. We introduce an evolving objective function that starts with the conditional value at risk (CVaR) defined in Ref. [11] and gradually in the optimization process becomes the full energy of the quantum state. Alternative forms of this ascending-CVaR objective functions are considered and a linear and a sigmoid functions (that appear to perform better) are selected.

2. We test our proposal with classical numerical simulations (using up to 20 qubits), both in the setting of VQE with hardware efficient ansatz and in QAOA. Our results suggest that our proposal leads to faster convergence with bigger overlap with the ideal solution than prior works, while crucially succeeding in obtaining the solution in (many) instances where other techniques fail altogether (see Sec. VIII for statistics and comparisons).

3. Our analysis is done for three different combinatorial optimization problems, namely MaxCut, number partitioning, and portfolio optimization. We consider many different instances and problem sizes where the conclusions persist in all cases. This has importance in its own right, since these problems are important by themselves, and our proposal gives an approach to improve the performance and bring us closer to achieving a useful quantum advantage. Interestingly, our method offered a greater advantage in hard instances of the problems, where the other methods frequently failed to find the solutions altogether.

II. STRUCTURE

In Sec. III, we give the essential background: We introduce the variational quantum algorithms and, specifically, the variational quantum eigensolver (VQE) and the quantum approximate optimization algorithm (QAOA). We then introduce the CVaR objective function of Ref. [11] and finally analyze the three different combinatorial optimization problems that we use as case studies. In Sec. IV, we introduce our method, called ascending-CVaR, and discuss the hyperparameters of our model. In Sec. V, we illustrate our method using a small instance as an example. In Sec. VI, we discuss our methodology. In Sec. VII, we present the results of our method compared to existing objective functions. We conclude in Sec. VIII with a general discussion of our method and future work.

III. PRELIMINARIES

We introduce two of the main variational quantum algorithms [13], the CVaR objective function [11] and the three types of combinatorial optimization problems that we use.

A. Variational quantum algorithms

Here we revise the methods with a focus on optimization problems. The general framework of a variational quantum algorithm is outlined in Fig. 1. The first step is to map the classical cost function $C(x)$ that describes the optimization

FIG. 1. General framework of a variational quantum algorithm. The optimization problem, described by a cost function $C(x)$, is mapped to an interacting qubit Hamiltonian $H_C$. A parameterized family of states (ansatz) $U(\theta)$ with random initial parameters is chosen and a quantum-classical feedback loop iteratively updates the parameters $\theta$. The optimization ends when the stopping condition is met and the optimal parameters $\theta^\ast$ are outputted.
problem, into an interacting qubit Hamiltonian $H_C$ whose ground state gives the solution we are seeking.

The second step is to choose an ansatz family of unitary operators $U(\theta)$. This family is both efficiently expressible and trainable, parameterized by a $\mu$-parameter vector $\theta = (\theta_1, \ldots, \theta_\mu)$, where $\mu = O(\text{poly}(n))$ and $n$ is the system size. In general, the parameters are initialized at random.\footnote{There are cases that a clever initialization could lead to faster convergence [14,15].} The third step is to evaluate some objective function, usually taken to be the expectation value of the problem’s Hamiltonian on the state considered $\langle \psi(\theta)|H_C|\psi(\theta) \rangle$. This is done by preparing the state [applying the unitary $U(\theta)\text{ on the initial state} and then measuring the output in the computational basis and repeating this procedure for a given number of times (typically called shots). This number determines the accuracy the objective function is evaluated. The fourth step is to update the parameters and repeat step three, iteratively using some classical optimizer until a stopping condition is satisfied. We then say that the parameters are optimal, i.e.,

$$\theta^* = \arg \min_{\theta} O(\theta, H_C). \quad (1)$$

The state produced by these parameters, $|\psi(\theta^*)\rangle = U(\theta^*)|0\rangle^\otimes n$, can be used to give an estimate of the ground-state energy of the Hamiltonian $H_C$ and thus an approximate solution to the desired optimization problem. The objective function used during this process, as stated above, typically coincides with the expectation value of the problem’s Hamiltonian. However, we note here that other choices may also be possible, especially if we realize that the true target of the optimization algorithm is to sample, at least once, the optimal solution. This can efficiently be produced if the output state has a sufficiently large overlap with the optimal solution $|\langle \psi(\theta^*)|\psi_{\text{opt}} \rangle|^2$.

B. Conditional value at risk

Barkoutsos et al. [11] used an alternative objective function. They demonstrated that their proposal performed better than minimizing the expectation value. The key observation is that for optimization problems, the optimal solution is a computational basis state. For computational basis states, one can compute their energy (efficiently). For a general quantum state $|\psi(\theta)\rangle$, one can prepare and measure it (multiple times) in the computational basis, and the expectation value of the energy is simply the average of the individual computational basis state energies. To find the overlap of this state with the optimal solution (ground state), one can simply observe the frequency of the computational basis state with the smallest energy. Naturally, if that overlap is too small (or even zero), it is possible that none of the measurements outcomes will give the solution. On the other hand, it is also clear that the overlap of this state with computational basis vectors with high energy are irrelevant for finding the ground state. The idea of Ref. [11] was to use this observation and instead of using all the measurement outcomes and computing the expectation value, they used as an objective function the lower tail of the distribution of energies obtained, i.e., ignored all but a small fraction (with smallest energy) of their measurement outcomes.

They then demonstrated that their technique succeeded in more quickly getting a quantum state that has a sufficiently large overlap with the ground state. This, in turn, is sufficient to actually find this ground state since, as a final step, once the optimal $\theta^*$ is found, one can keep the computational vector that has the smallest energy only. Specifically, let $H_k$ be the energy corresponding to a computational basis vector and let us order them in such a way that larger $k$ corresponds to larger energy. For each state, one repeats the measurement $K$ times, so there are (up to) $K$ distinct values $H_k$. In Ref. [11], a parameter $\alpha$ was introduced. Let $\alpha (0, 1]$ be the fraction (part of the tail) that we want to keep. This fraction, typically, needs to be non-negligible (we can assume, for simplicity, that is constant). Then the objective function that was used, was the average of the smallest $\alpha K$ samples, i.e.,

$$\text{CVaR}_\alpha = \frac{1}{\alpha K} \sum_{k=0}^{\lfloor \alpha K \rfloor} H_k. \quad (2)$$

To achieve the same accuracy when evaluating this objective function, as the accuracy achieved when computing the expectation value using $K$ shots, it is clear that the number of runs of the preparation circuit need to be increased to $K/\alpha$.

As proven by Ref. [11], the angles $\theta^*$ that minimize CVaR do not (in general) correspond to minima of the expectation function, as the accuracy achieved when computing the expectation value using $K$ samples. This fact motivates us to introduce a lower $\alpha$-tail optimization so as to achieve an overlap with the optimal state of at least $\alpha$, i.e., find optimal $\theta^*$ that satisfy

$$|\langle \psi(\theta^*)|\psi_{\text{opt}} \rangle|^2 \geq \alpha. \quad (3)$$

C. Combinatorial optimization problems

We test our proposed method in various instances of three different combinatorial optimization problems. These are all important problems in their own right, so improving the performance of variational quantum algorithms for these problems is of independent interest. Moreover, testing our proposed objective function on different types of combinatorial optimization problems demonstrates that improvements observed are generic and motivates further use for different applications. Given that our proposal’s starting point is the work of Ref. [11], we included the problems that they tested in their proposal to allow for a more direct comparison.

The easiest way to use variational quantum algorithms for an optimization problem is to first map the problem to a quadratic unconstrained binary optimization (QUBO) problem. This is what we will do for all our examples. QUBO problems seek to solve (find the $x$ that minimizes the expression)

$$\min_x (b^T x + x^T A x). \quad (4)$$

where $b \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times n}$. These cost functions can easily be mapped to an Ising Hamiltonian [16] by first transforming
the binary variables \( x_i \in \{0, 1\} \) according to
\[
x_i = \frac{1 - z_i}{2},
\]
where \( z_i \in \{-1, +1\} \) are spin variables, and then turning the cost function to a Hamiltonian by promoting these variables to Pauli \( \sigma^z \) operators, one for each qubit \( i \). The QUBO problem then transforms to
\[
\min_{z} c^T z + z^T Q z.
\]
where the new \( c \in \mathbb{R}^n \) and \( Q \in \mathbb{R}^{n \times n} \) are easily computable.

Then, by replacing the spin variable \( z_i \) with the Pauli \( \sigma^y \) operator with corresponding eigenvalues \( \{-1, +1\} \), the problem translates into finding the ground state, i.e., the spin configuration, of an \( n \)-qubit system interacting with the Hamiltonian:
\[
H = \sum_{i=1}^{n} c_i \sigma^z_i + \sum_{i=1}^{n} q_{ij} \sigma^z_i \sigma^z_j.
\]

1. MaxCut problem

The first problem is MaxCut. It is one of the most studied combinatorial problems in the context of variational quantum algorithms due to the simplicity and guaranteed performance—at least for some instances [17,18].

Let \( G(V, E) \) be a nondirected \( n \)-vertex graph, where \( V \) is the set of vertices, \( E \) is the set of edges, and \( w_{ij} \) are the weights of the edges. A cut is defined as a bipartition of the set \( V \) into two disjoint subsets \( P, Q \), i.e., \( P \cup Q = V \) and \( P \cap Q = \emptyset \). Equivalently, we label every vertex with either 0 or 1, where it is understood that the vertex belongs to set \( P \) if it takes the value 0 and to set \( Q \) if it takes the value 1. The aim is to maximize the following cost function:
\[
C(x) = \sum_{i,j=1}^{n} w_{ij} x_i (1 - x_j).
\]

This intuitively corresponds to finding a partition of the vertices into two disjoint sets that cut the maximum number of edges. By applying the transformation, Eq. (5), the cost function transforms into
\[
C(z) = \sum_{(i,j) \in E} \frac{w_{ij}}{2} (1 - z_i z_j).
\]

Maximizing the cost function above corresponds into finding the ground state of the Hamiltonian:
\[
H_C = - \sum_{(i,j) \in E} \frac{w_{ij}}{2} (1 - \sigma^z_i \sigma^z_j).
\]

MaxCut is known to be NP-hard. The best classical approximation algorithm is that of Goemans and Williamson, which uses semidefinite programming to achieve an approximation ratio, Eq. (A4), \( r^* \approx 0.87856 \) for all graphs. Note that being NP-hard implies that we do not expect to have an efficient quantum algorithm (polytime) to solve the problem for its hardest instances,\(^5\) but we could definitely get improvements using quantum algorithms (either by smaller speedups or by heuristics that could solve more instances than classical heuristics).

Although it was proven that constant-depth QAOA does not outperform GW for certain classes of problems [4], there are instances where the approximation ratio of the former is larger than the latter [19]. Note here that QAOA beats random guessing even at \( p = 1 \) [17], while machine-learning techniques have been used to classify which graph type it is better to use QAOA instead of GW [20]. In general, however, the performance of QAOA in intermediate depths is still highly unexplored.

2. Number partitioning

The second problem is number partitioning and is stated as follows. Given a set of \( N \) positive integers \( S = \{n_1, n_2, \ldots, n_N\} \), the target is to find a bipartition of the set \( S \) into two disjoint subsets \( P, Q \), where \( P \cup Q = S \) and \( P \cap Q = \emptyset \) so the difference between the sum of the elements on the set \( P \) and the set \( Q \) is minimized. We thus want to minimize the cost function:
\[
C(x) = \left( \sum_{i=1}^{N} (2x_i - 1)n_i \right)^2.
\]

The binary string \( x = x_1 x_2 \ldots x_n \) corresponds to one configuration where a number \( n_i \) is placed in the \( P \) set (\( x_i = 0 \)) or in the \( Q \) set (\( x_i = 1 \)). The cost function can easily be mapped to the Ising Hamiltonian:
\[
H_C = \left( \sum_{i=1}^{N} \sigma^z_i n_i \right)^2.
\]

By expanding the cost function Eq. (12), the cost function can be written as
\[
H_C = \sum_{i \neq j} (n_i n_j) \sigma^z_i \sigma^z_j + \sum_{i=1}^{N} n_i^2.
\]

If we neglect the constant term, we can see that the number partitioning problem can be easily mapped to the Sherrington-Kirkpatrick model, which is an energy minimization problem with an all-to-all random couplings which was recently analyzed in Ref. [21].

Although the problem is known to be NP-hard, it is also known as the easiest hard problem. That is, because there exists a hard-easy phase transition [22] where instances belonging in the easy-phase can be efficiently tackled using heuristics [23]. Interestingly, it appears that one may be able to tackle some of the instances in the hard phase, using variational quantum algorithms.

3. Portfolio optimization

The third problem is portfolio optimization [24,25] and is stated as follows. Given a set of \( n \) assets \( \{0, \ldots, n\} \),

\(^4\)Note the overall minus sign that turns the maximization of the cost function to finding the minimum energy for the Hamiltonian.

\(^5\)NP is strongly believed to not be included in Bounder-Error Quantum Polynomial time (BQP).
corresponding expected returns $\mu_i$ and covariances $\Sigma_{ij}$, a risk factor $q > 0$ and a budget $B \in \{1, \ldots, n\}$, the considered portfolio optimization problem tries to find a subset of assets $P \subset \{1, \ldots, n\}$ with $|P| = B$ such that the resulting $q$-weighted-mean-variance, i.e., $\sum_{i \in P} \mu_i - q \sum_{i,j \in P} \Sigma_{ij}$, is maximized. In other words, we want to maximize the cost function,

$$C(x) = \sum_{i=1}^{n} \mu_i x_i - q \sum_{i,j=1}^{n} \Sigma_{ij} x_i x_j,$$

along with the constraint

$$\sum_{i=1}^{n} x_i = B.$$

The portfolio vector $x \in \{0, 1\}^n$, consisting of $n$ binary decision variables, indicates whether an asset is picked ($x_i = 1$) or not ($x_i = 0$). The constraint in Eq. (15) is translated as an extra penalty term in the Hamiltonian $(\sum_{i=1}^{n} x_i - B)^2$.

The problem is known to be NP-complete \[26\]. We apply the transformation, Eq. (5), so the cost function transforms into

$$C(z) = -q \sum_{i,j=1}^{n} \Sigma_{ij} z_i z_j + \sum_{i=1}^{n} \left( \frac{\mu_i}{2} - \frac{n \Sigma_{ij}}{4} \right) z_i z_j,$$

$$+ \sum_{i=1}^{n} \left( \frac{\mu_i}{2} - \frac{n \Sigma_{ij}}{4} \right),$$

which, along with the extra penalty term, corresponds to minimizing the Hamiltonian:

$$H_C = \sum_{i,j=1}^{n} \frac{q \Sigma_{ij}}{4} z_i \sigma_i^+ \sigma_j^+ - \sum_{i=1}^{n} \left( \frac{\mu_i}{2} - \frac{n \Sigma_{ij}}{4} \right) z_i z_j,$$

$$- \sum_{i=1}^{n} \left( \frac{\mu_i}{2} - \frac{n \Sigma_{ij}}{4} \right) + \left( \sum_{i=1}^{n} \sigma_i^+ + n - B \right)^2.$$

Portfolio optimization as given in Eq. (14) was recently tackled using variational quantum algorithms \[27\] using warm-starting QAOA \[3\] and on D-wave systems using quantum annealing \[28\]. Prior to our work, Ref. \[29\] developed a quantum-walk-based optimization algorithm and Ref. \[30\] considered a more general setting of portfolio optimization called dynamic portfolio optimization, where one has to allocate weights to a number of assets in a period of time to maximize the overall return.

IV. ASCENDING-CVAR

The CVaR cost function of Ref. \[11\] was shown to perform better, in general, than the standard expected value. There are three observations, however, that motivates our proposal. First, as noted in Ref. \[11\], the choice of $\alpha$ is somehow random, and importantly, for different problems and even for different instances of the same class of problems, the optimal choice of $\alpha$ varies in a nonobvious (e.g., monotonic) way. The performance of the algorithm’s speed, but also if it finds the solution at all, depends on that choice. The second point is that optimizing with a fixed small $\alpha$ has further disadvantages: (i) it finds parameters $\theta$ that result to a state that does not have the greatest overlap with the solution and (ii) the true running time of the algorithm to achieve same accuracy is larger; in other words, for each iteration one requires $1/\alpha$ times more measurements to achieve the same accuracy in estimating the cost function (since only the lower $\alpha$ fraction of the measurements are used). Finally, the third observation is that the CVaR$_{\alpha}$ objective functions with different $\alpha$ have a different energy landscape. For any fixed choice of $\alpha$ the optimizer could get stuck at a local minimum. Interestingly, if one varies the $\alpha$ during the optimization, while we still ensure that if the algorithm finds the true ground state it remains there, we also avoid getting stuck at local minima since those are different for different choices of $\alpha$. Therefore, if the optimizer reaches a point that has a local minimum for one value of $\alpha$, when $\alpha$ changes this point (may) no longer be a local minimum and thus could continue moving toward the true global minimum (ground state).

Let’s say that an optimization problem has an optimal solution which is a computational basis state and we denote it as $|\psi_{\text{opt}}\rangle$. Let’s also assume that a parameterized family of gates, $U(\theta)$, acts on the $|0\rangle^{\otimes n}$ state and produces the state

$$|\psi(\theta)\rangle = a_{\text{opt}}(\theta)|\psi_{\text{opt}}\rangle + (1 - a_{\text{opt}}(\theta))|\psi_{\text{other}}\rangle,$$

where $|\psi_{\text{other}}\rangle$ is the superposition of all suboptimal computational basis states. Let’s also assume that this parameterized family of states can achieve a maximum overlap $\kappa$ with the optimal solution.\(^4\) We can write the state $|\psi\rangle$, corresponding to the state with the highest overlap, without loss of generality as

$$|\psi\rangle = \sqrt{\kappa} |\psi_{\text{opt}}\rangle + (1 - \sqrt{\kappa}) |\psi_{\text{other}}\rangle.$$

**Proposition 1.** For the family of states in Eq. (18) and for all $\alpha \leq \kappa$:

$$\min_{\theta} \text{CVaR}_{\alpha}(\theta) = \min_{\theta} \langle \phi | H_C | \phi \rangle,$$

i.e., all CVaR$_{\alpha}$ with $\alpha \leq \kappa$ share the same minimum objective function value which is the smallest eigenvalue of the Hamiltonian.

It is clear from Proposition 1 that all CVaR$_{\alpha}(\theta)$ with $\alpha \leq \kappa$ share the same ground state, which is the true optimum of the optimization problem. Thus, all angles $\theta^*$ that correspond to a global minimum of CVaR$_{\alpha}$ will also correspond to a global minimum of CVaR$_{\kappa}$ if $\alpha_2 \leq \alpha_1 \leq \kappa$. For example, for an ansatz $U(\theta)$ that is able to attain 10% overlap with the optimal computational basis state, if one is able to find the global minimum of CVaR$_{\alpha_1}$, which means that 10% of the measurements correspond to the ground state, then it is clear that all $\alpha < 0.1$ will also be minimized by the same angles.

**Proposition 2.** Let an optimization problem with an optimal solution $|\psi_{\text{opt}}\rangle$ corresponding to a computational basis state. For any parameterized family of gates $U(\theta)$ that can achieve a maximum overlap $\kappa$ with the optimal solution, the angles $\theta^*$ that correspond to the global minimum of CVaR$_{\alpha}$ will also _______

\(^4\)In other words, the complex coefficient $a_{\text{opt}}(\theta)$ corresponding to the probability of sampling the optimal solution $\text{Prob}(\text{opt}) = |a_{\text{opt}}|^2$ has a maximum value: $\max_{\theta} |a_{\text{opt}}(\theta)|^2 = \kappa$. 

023225-5
correspond to a global minimum for $\text{CVaR}_\alpha$, if $\alpha_1 \leq \alpha_2 \leq \kappa$. The converse does not necessarily hold.

In other words, Proposition 2 states that if $\theta^* = \arg\min_\theta \text{CVaR}_\alpha(\theta)$ then also $\theta^\ast = \arg\min_\theta \text{CVaR}_\kappa(\theta)$ for all $\alpha_1 \leq \alpha_2 \leq \kappa$. This indicates why decreasing $\alpha$ may not seem like a good choice. If, for example, the optimizer is able to find the optimal angles that minimize $\text{CVaR}_\alpha$, with $\alpha_1 \leq \kappa$, then for all $\alpha_2 < \alpha_1$ they will still remain optimal angles and thus will not be able to achieve a higher overlap state.

**Proposition 3.** A local minimum for $\text{CVaR}_\alpha$ does not necessarily correspond to a local minimum for $\text{CVaR}_\kappa$ if $\alpha_1 \neq \alpha_2$.

Proposition 3 was proven using a counterexample in Ref. [11]. All these propositions are important for introducing a nonstationary optimization technique that avoids local minima. We know from Proposition 1 that all $\text{CVaR}_\alpha$ objective functions with $\alpha \in (0, \kappa]$ share the same minimum objective value which is the ground state energy of the Hamiltonian. We also know from Proposition 2 that many of the global minima for $\alpha_1$ may not be a global minimum for $\alpha_2$ if $\alpha_1 < \alpha_2$ and thus increasing $\alpha$ introduces extra information about the optimality of states. Finally, Proposition 3 indicates that different objective functions are associated with different energy landscapes as they do not agree on the local minima.

However, knowing the maximum overlap $\kappa$ in advance is not always possible. In the case of VQE with a hardware efficient ansatz, it can be shown that $\kappa = 1$ and so $\min_\alpha \text{CVaR}_\alpha(\theta) = \min_\alpha \langle \phi | H_C | \phi \rangle$ for every $\alpha \in (0, 1]$. On the other hand, for the QAOA ansatz, our experiments showed that $\kappa$ is usually small on low depth but increases with the number of layers.

The cost functions used in variational quantum algorithms, to our knowledge, are constant in time, meaning that the whole optimization is run with a fixed cost function. To solve the issue of selecting the best $\alpha$ and the other reasons listed above, we propose to use a dynamically evolving cost function that essentially passes through a fixed set of $\alpha$ values. In the case of VQE, it is initialized in a very small value and the optimization ends with $\alpha = 1$ that is the standard expectation value of the Hamiltonian. We call all these cost functions ascending-CVaR. This also has a great(er) number of free choices, since we can now freely choose the (ascending) function. However, all choices we tried for the ascending function performed (in general) better than fixed $\alpha$, which indicates that the evolving cost function is a promising approach. For the remainder of the paper, we focused on two functions that performed better:

The **linear ascending**, in which the parameter $\alpha_t$ is iteratively and discretely increased by the rule

$$
\alpha_{t+1} = \alpha_t + \lambda,
$$

$$
\text{CVaR}_\alpha = \frac{1}{|\alpha_t K|} \sum_{k=0}^{[\alpha_t \kappa]} H_k, \tag{21}
$$

where $\lambda \in [0.025, 0.045]$ is the ascending factor and $0 < \alpha_t \leq 1$.

The **sigmoid ascending** in which the parameter $\alpha_t$ is discretely increased according to the function

$$
\alpha_t = \frac{1}{1 + e^{3-\lambda t}}, \tag{22}
$$

where $\lambda \in [0.3, 0.4]$ is again the ascending factor and $0 < \alpha_t \leq 1$.

To reach this conclusion, we tested four different functions, a *sigmoid*, a *linear*, an *exponential*, and a *logarithmic* (see Fig. 2) on VQE-$\text{CVaR}_\alpha$ with various different ascending rates. All functions were tested on all three problems. The metrics used were the magnitude of the overlap with the optimal solution, the success rate (i.e., the number of times where it succeeds to achieve a non-negligible overlap) as well as the average time taken to achieve at least 10% overlap (for details, see Sec. VI).

The linear ascending, Eq. (21), and the sigmoid ascending, Eq. (22), functions have the most steady behavior as can be seen in Fig. 3, outperforming the other two types on the majority of instances. The sigmoid was slightly slower in terms of speed, which is why we mainly used the linear one. However, as we will discuss in the next section, it appears that it may
be better in some classes of problems especially on harder instances with \( \sim 50 \) qubits. It seems that in those cases, the optimizer is doing better spending the majority of its iterations on low \( \alpha \) values and thus the sigmoid performs better.

It is worth noting that increasing \( \alpha \) to \( \alpha = 1 \) where it becomes the expectation value is not necessary. The whole point of variational algorithms is to achieve a constant non-negligible overlap with the optimal or near-optimal solution. For that reason, one could only vary \( \alpha \) until it reaches a threshold \( \varepsilon \) truncating the optimization and reducing the number of iterations by a considerable amount.

We remark here that ascending-CVaR is fundamentally different from an adaptive strategy that selects the optimal value for the parameter \( \alpha \). Specifically, by looking at the results in Sec. VII we can see that our method is able to reach quantum states that result in a very high overlap with the optimal state (almost equal to unity) that no constant choice of \( \alpha \) would be able to attain. Even when the question is whether we find the solution with at least some small probability, ascending-CVaR succeeds in cases where all of the fixed \( \alpha \) failed.

To understand why our method works, one should think of what varying \( \alpha \) actually does. The optimizer, at each iteration, moves toward a local (or a global) minimum corresponding to the instantaneous value of \( \alpha \). By increasing \( \alpha \) at every step, the optimization is able to see a larger part of the energy distribution of the quantum state. This translates to the optimizer gaining additional information about the quantum state which modifies the objective function landscape.

This extra information alters the landscape and is thus able to erase false local minima, while using the previous step as a transformation, this change in the landscape will not affect any correct moves of the optimizer. In other words, if the optimization algorithm did converge in a suboptimal local minimum for a value of \( \alpha \), it may not still be in a local minimum by switching into a different value of \( \alpha \). Hence, the ascending-CVaR algorithm is guiding the trajectory of the optimizer in the highly parameterized space until it reaches a (nearly) optimal solution.

Finally, the reason our method is sensitive to the choice of the ascending factor \( \lambda \) is related to the speed that the optimizer is receiving this extra information (see Appendix C on how we numerically test the ascending factors for the different optimization problems).

The pseudocode for the ascending-CVaR algorithm is outlined in Algorithm 1.

V. WHY OUR METHOD WORKS: AN EXAMPLE

It would be illustrative to describe how local minima may vanish when the objective function is changed during the optimization. In Fig. 4, we plot the CVaR\(_{\alpha} \) objective function landscape for different values of \( \alpha \). We choose to draw the landscape for the QAOA algorithm with depth \( p = 1 \) because the two parameters \( \beta, \gamma \) make it suitable to visualise in a 2D plot. On the contrary, VQE with a hardware efficient ansatz even on depth \( p = 1 \) would require \( 2n \) parameters.

It can be easily seen that the positions of the local minima change but the position of the true global minimum remain the same while the condition \( \alpha \lesssim \kappa \) holds (see Proposition 1). However, to make it clearer for the reader, we choose to circle the position of a local minimum, located at \( \gamma = 0.15, \beta = 1.75 \). In this case, we can see how the local minima vanishes during the variation of the objective function. An optimizer that could stick during the optimization on a fixed value of \( \alpha \) could unstick with the change of \( \alpha \).

The problem corresponding to the figures is a small instance of the number partitioning problem with size \( n = 8 \). Even in a small size instance like this, the landscape is full of suboptimal local minima where the optimizer could falsely converge to. This case problem, however, does not constitute an example to prove the value of our method but only to visualise the changes in the energy landscape. The biggest improvements were observed in a high-dimensional expressive parameterized family of gates like the VQE with a hardware-efficient ansatz or larger depth QAOA which cannot be plotted in a two-dimensional contour.

VI. METHODS

One common metric used, especially in QAOA, is the approximation ratio as given in Eq. (A4). However, as we noted earlier, the true aim of variational quantum algorithms for combinatorial optimization is to quickly obtain a sufficiently high (but not necessarily close to unity) overlap with the optimal solution. The CVaR method, for example, is constructed in a way that the maximum overlap achieved is not unity but determined by the risk \( \alpha \). While our approach does achieve a high approximation ratio, to make a fair and more complete comparison with prior works and importantly with Ref. [11], we use different metrics. Specifically, to benchmark and test our proposed method, we used three different types of metrics. The first is the overlap with the optimal solution. If \( |\psi_{opt,i}\rangle \) is a \( d \)-degenerate ground state of the problem Hamiltonian, then the overlap is defined as

\[
\sum_{j=1}^{d} |\langle \psi(\theta) | \psi_{opt,i}\rangle|^2,
\]

(23)
i.e., the probability of obtaining the optimal solution, given the parameters \( \theta \). It follows that the parameterized state with the highest overlap with the optimal solution leads to
FIG. 4. Visualization of local and global minima for different CVaR $\alpha$ objective functions. In the top two figures, corresponding to $\alpha = 0.05$ on the left and $\alpha = 0.08$ on the right, you can see the local minima drawn in the red circle. However, in the bottom figures, corresponding to $\alpha = 0.11$ on the left and $\alpha = 0.14$ on the right, the local minima no longer exist.

The second metric we want to test is the time taken to reach a given fixed overlap. We set a threshold of 10% probability of obtaining the optimal solution and we tested which method achieves at least that probability faster. We note, however, that to test which method converges to a 10% overlap faster, we have to use $\alpha \geq 0.1$ because all $\alpha < 0.1$ are not guaranteed to converge in an overlap of 10% since the parameters $\theta$ that minimize $\alpha$ lead in an overlap smaller than 0.1.

To summarize the results and compare better the different approaches, for each cost function we divided the problem instances to those where the cost function is successful and to those where it fails. The meaning of what constitutes a successful run or a failed run cannot be unambiguously defined. For our paper, we consider that an optimizer is successful at a given instance of a problem if it achieves at least 10% overlap with the optimal solution.

We should note that even a much smaller overlap is sufficient to find the solution at least once, provided that the number of shots is sufficiently large.

In our experiment, for comparing with fixed $\alpha$ we used four different choices: $\alpha = 0.1, 0.2, 0.5, 1$. The $\alpha = 1$ choice corresponds to a non-CVaR objective function. Specifically, $\alpha = 1$ refers to the expectation value (it includes all the measurement outcomes) and it is the objective function that has been used in the overwhelming majority of the existing literature on variational quantum algorithms. In Ref. [11], they did an extensive comparison of CVaR with the expectation value (on the same combinatorial problems we make our analysis). For that reason, we choose to make the comparison of all different choices of $\alpha$ with our proposed $\alpha_t$ and plot our results in one section (see Sec. VII).

We also note that ascending factors $\lambda \in [0.025, 0.045]$ and $\lambda \in [0.3, 0.4]$ were found to be a good choice for the three different problems on instances with 15 to 20 qubits for the linear and sigmoid ascending, respectively (see Appendix C). However, we would like to stress that if the sizes of the instances increase or even if the problems change but the sizes remain the same, one would have to readjust the hyperparameter $\lambda$. Investigating theoretically the choice of both the ascending factor and the ascending function given the characteristics of the problem as well as possible connections of our method to adiabatic quantum computing goes beyond this leads to interesting behavior where the methods analyzed differ in their performance.
the scope of this paper but will be investigated in a subsequent work.

In the QAOA algorithm, we tested instances using depth $p = 1$ to $p = 6$ while on VQE we worked only on the depth $p = 1$, since this depth was sufficient to get very good accuracy. In near term devices for the QAOA algorithm, increasing the depth even more becomes impractical due to noise and decoherence. For this reason, we did not consider greater depth, despite the fact that theoretically this could lead to better performance. This means that the variational ansatz for QAOA has only 2 to 12 parameters, i.e., only a fraction of the total parameters present in hardware-efficient ansatz used for VQE in depth-1.

To account for the different sizes of problem instances and to make a fair comparison for the speed of convergence, we used the normalized optimizer iterations [31]. Note that this choice is made to be able to compare the performance of the algorithm among instances that involve different numbers of qubits, and see how the improvement offered by ascending-CVaR is independent of the instance size. Concretely, the normalized optimizer iterations are defined as the number of times the optimizer evaluates the objective function divided by the function’s number of parameters, i.e., the number of parameters of the ansatz. In the case of the VQE, the number of parameters are $n(1 + p)$ while on QAOA are $2p$. We note, however, that the real time of convergence could be used as seen in Appendix B, where we compare the performance with respect to the total number of circuit repetitions. However, as we show below, there are instances where the constant CVaR does not achieve even a small overlap with the optimal solution and in those cases the time taken becomes irrelevant.

We ran our experiments on IBM’s Qiskit Aer simulator, allowing noiseless multishot executions of our circuit. We set the number of executions of our circuit to $K = 1000$, which scaled up as $K/\alpha$ with the choice of $\alpha$. All instances were given a maximum of $(66 \times p)$ optimizer iterations which is more than enough iterations for an optimizer to converge to a minimum in the problems we implemented. They were initialized with a random choice of parameters, but the same for all different choices of $\alpha$. We used the same gradient-free optimizer, COBYLA [32], for all different problems and instances as it was shown to outperform other classical optimizers [33].

### VII. RESULTS

We will analyze the results for each of the three combinatorial optimization problems separately. For each of them, we will present the results for VQE with a hardware-efficient ansatz first and then the results for QAOA. We note that for all three combinatorial optimization problems and for all methods used (ascending-CVaR, constant CVaR, expectation value), VQE performs (much) better than QAOA, at least for the sufficiently shallow circuits that we consider. Our method improves the performance in both cases (VQE, QAOA) but since VQE gives much better results for these problems, in the comparison and discussion we will focus on VQE instances only.

#### A. MaxCut

For the MaxCut problem, we worked on unweighted graphs with 15-19 vertices, drawn from different graph classes and sampled them using the NetworkX library [34].

#### 1. CVaR$_{\alpha}$-VQE

For regular graphs, CVaR$_{\alpha}$-VQE behaved equally well with constant-$\alpha$’s optimization as well as with the expectation value. All methods reached the chosen threshold of 10% overlap with the optimal state at almost equal times without any difficulty. For that reason, we focused on harder nonregular instances where our method outperformed the latter methods. In Table 1, we give a summary of the results for 100 random nonregular unweighted graph instances with 15 to 19 vertices. We can see that our method succeeds in more instances while the overlap achieved is also much higher.

There are many reasons why nonregular random graphs are harder than regular graphs. The first is that the ground state of a regular graph, due to its symmetry, is highly degenerate where the optimizer could easily reach without sticking in a suboptimal minimum. The second is that the Hamiltonian corresponding to a random graph has more distinct eigenvalues and, as shown numerically by Ref. [33], the number of distinct eigenvalues correlates inversely with the performance of hardware-efficient ansatz.

Indicatively, in Fig. 5, we plot the probability of sampling the optimal solution over the normalized number of iterations for two random graphs with 17 vertices. For the left figure, we can see how the optimizer for the ascending-CVaR optimization is able to to find the optimal solution in under ten normalized iterations, which by the end of the optimization is able to increase the probability up to 70%. Notably, the expectation value or constant CVaR completely fail. The right part of the figure gives another example where our approach performs better. This instance constitutes an example where smaller $\alpha$’s do not lead to better performance for constant CVaR.8 We can see in the figure that while $\alpha = 0.1$ failed, $\alpha = 0.2$ was able to achieve a high quality parameterized state. This is another indication why our approach is more flexible.

8In most cases, small $\alpha$ gives better performance, but one cannot know a priori which is the suitable $\alpha$ in the constant CVaR case.

---

**TABLE I.** Results table for the MaxCut problem (VQE) for 100 random non-regular unweighted graph instances with 15 to 19 vertices.

<table>
<thead>
<tr>
<th>MaxCut</th>
<th>Successful instances</th>
<th>Average overlap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\alpha$</td>
<td>0.1</td>
</tr>
<tr>
<td>Random graphs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MaxCut</td>
<td>$\alpha$</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>96</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>64.69</td>
<td>12.13</td>
</tr>
</tbody>
</table>
FIG. 5. MaxCut instances with 17 vertices for random nonregular unweighted graphs. Ascending-CVaR, drawn with a blue line (diamond marker), results in a fast and high overlap with the optimal solution in contrast to constant CVaR.

2. CVaR$^\alpha$-QAOA

Solving the Max-Cut problem using QAOA, with small depth circuits, does not seem a very promising approach in any of the methods considered (constant CVaR or Ascending-CVaR). In terms of speed, all methods converged equally fast but in states with small overlap with the solution (with relatively small differences within different approaches). Having said that, as explained below, our method still gives improved performance.

While CVaR$^\alpha$-VQE optimization results in high overlap states, CVaR$^\alpha$-QAOA produces flat states, a behavior also observed in Ref. [11]. These states have almost equal probability amplitudes to the majority of the computational basis states. For the MaxCut problem, as noted in Ref. [17], it seems that the states produced with QAOA with small $p$ result in states with energy close to the (random) initialization point. The spread of the energies does increase with $p$, possibly leading to a state close to the ground state, but in our analysis we focused on small $p \leq 6$. Intuitively, the main reason why QAOA cannot achieve the same probability amplitudes as VQE, in the same depth, is due to having a smaller number of parameters as well as the architecture of the ansatz [35].

Note that the parameter space is filled with suboptimal local minima. Constant CVaR objective functions with different confidence level $\alpha$’s lead to different energy landscapes. This means that a local minimum for a confidence level $\alpha_1$ does not, in general, correspond to a local minimum for a confidence level $\alpha_2$ if $\alpha_1 \neq \alpha_2$. This is probably the reason that we get improved performance. For example, in Fig. 6 we see how ascending-CVaR can avoid local minima. In this example, all constant CVaR achieve less than 3% overlap with the ground state, while the ascending-CVaR gives 7%.

B. Number partitioning

On number partitioning, we tested instances with 17 to 20 integers, on both VQE and QAOA.

1. CVaR$^\alpha$-VQE

On CVaR$^\alpha$-VQE, we tested 300 instances with 17 to 20 integers, sampled randomly from three sets: $N_1 = \{0, \ldots, 200\}$, $N_2 = \{0, \ldots, 500\}$, and $N_3 = \{0, \ldots, 750\}$. We highlight that the smaller the set the numbers are uniformly drawn from, the easier the optimizer succeeds in finding the optimal solution. A summary of the results is given in Table II.

<table>
<thead>
<tr>
<th>Successful instances</th>
<th>Average overlap</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>$\alpha_1$ 0.1</td>
</tr>
<tr>
<td>$N_1$</td>
<td>87</td>
</tr>
<tr>
<td>$N_2$</td>
<td>80</td>
</tr>
<tr>
<td>$N_3^*$</td>
<td>95</td>
</tr>
</tbody>
</table>
FIG. 7. Probability of sampling the optimal solution for number partitioning instances with 17–20 integers uniformly drawn from the sets \( N_1 = \{0, \ldots, 200\} \) (on the left) and \( N_2 = \{0, \ldots, 500\} \) (on the right). The blue line (diamond marker), indicating ascending-CVaR, outperforms constant CVaR in terms of speed and overlap with the optimal solution.

For the first two sets, we used a linear ascending function with an ascending factor \( \lambda = 0.03 \). Further optimization of the parameter may lead to either faster convergence or more successful instances. Either way, the ascending-CVaR method outperforms constant CVaR and the expectation value objective function on the aforementioned metrics (e.g., see typical performance in Fig. 7).

For the last set \( N_3 \), constant CVaR and the expectation value as objective functions struggled to achieve even a small overlap with the optimal solution. Indicatively, at 40% of the cases, none of the constant CVaR objective functions could be successful.\(^9\) We found that by choosing a sigmoid ascending function, the optimizer is able to attain a high-quality parameterized state and succeed in the majority of instances (95%). The trade-off is that using the sigmoid ascending function, in contrast to linear ascending, comes with some cost of more circuit shots to achieve the same accuracy. Note also that the linear ascending function, while performing worse than the sigmoid, was still more successful than the constant CVaR objective functions.

2. CVaR\(_\alpha\)-QAOA

While CVaR\(_\alpha\)-VQE optimization efficiently achieved a high overlap state already within the first layer for instances drawn from the two sets \( N_1 \) and \( N_2 \), CVaR\(_\alpha\)-QAOA failed to achieve a high overlap on small depths. To address this issue without having to increase the depth of the ansatz, we chose to work on instances drawn from the smaller set \( M = \{0, \ldots, 50\} \). For the number partitioning problem, the cost function’s parameter space is highly dependent on the set we draw the numbers from. The unitary transformation \( e^{i\gamma H_C} \) is composed of \( e^{i\gamma n_k n_l \sigma_k^z \sigma_l^z} \) terms where \( n_k, n_l \) correspond to the numbers on the \( k \) and \( l \) indexes, respectively. The parameter \( \gamma \) is then restricted to \( 0 \leq \gamma < 2\pi/(n_j n_m) \) with \( n_j \) and \( n_m \) corresponding to the two smallest numbers of the set.

Our method succeeds in finding quantum states with higher overlap, unreachable with constant CVaR optimization, possibly because it avoids the high amount of local minima. Indicatively, in Fig. 8 we see an example where ascending-CVaR achieves more than double overlap with the optimal solution than other methods, but is still below the threshold of 10% required to classify this as a successful run.

C. Portfolio optimization

On portfolio optimization, we tested instances with 16 to 20 assets, on both VQE and QAOA, with a budget drawn uniformly at random from the set \( B = \{0, \ldots, n\} \), where \( n \) is the number of assets and many different risk factors \( q \).

\(^9\)Recall that successful in our convention means to achieve overlap of at least 10% with the optimal solution.
TABLE III. Results table for the portfolio optimization problem (VQE) for 100 random portfolios with 16 to 20 assets.

<table>
<thead>
<tr>
<th>Portfolio optimization</th>
<th>Successful instances</th>
<th>$\alpha_t$</th>
<th>0.1</th>
<th>0.2</th>
<th>0.5</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random portfolios</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>16</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Portfolio optimization</th>
<th>Average overlap</th>
<th>$\alpha$</th>
<th>0.1</th>
<th>0.2</th>
<th>0.5</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random portfolios</td>
<td></td>
<td>63.25</td>
<td>13.35</td>
<td>24.74</td>
<td>9.42</td>
<td>0.64</td>
</tr>
</tbody>
</table>

1. CVaR$_{\alpha_t}$-VQE

We used linear ascending with an ascending factor $\lambda = 0.045$ and the confidence level was initialized on $\alpha_0 = 0.01$. The results are summarized in Table III. In Fig. 9, we see the typical performance of two different instances where we plotted the probability of obtaining the optimal solution over the normalized number of optimizer iterations for the CVaR$_{\alpha_t}$-VQE.

We highlight the fact that ascending-CVaR and constant CVaR with $\alpha = 0.1, 0.2$ succeed in achieving at least 10% overlap in all instances tested (see results in Table III), while the expectation value ($\alpha = 1$) failed in almost all cases. Moreover, it is worth noting that our method offers a significant improvement in comparison with all the other approaches in the speed that this overlap was achieved (in terms of normalized optimizer iterations and circuit repetitions) and in the overall magnitude of the overlap achieved (see also Table III).

2. CVaR$_{\alpha_t}$-QAOA

CVaR$_{\alpha_t}$-QAOA, similarly with Ref. [11], underperforms significantly in terms of overlap with the optimal state, compared to CVaR$_{\alpha_t}$-VQE. Specifically, keeping the depth as in previous parts, and without increasing the shots, each circuit is implemented, and all methods fail, achieving overlap with the optimal solution well below 1%. There are several reasons for this failure, including the reachability deficits [36], the large problem density [33], and barren plateaus [37]. This, however, goes beyond the focus of this paper, that is, to find a way to improve the performance of previously used objective functions. To illustrate the improvement, we could have used a (significantly) larger number of shots, where ascending-CVaR would start showing better performance. This would make the comparison with other problems unfair (where in all cases we used the same “normalized” number of shots), and it would still not present a practical way to solve the portfolio optimization problem (VQE is much better), so we omitted it.

VIII. CONCLUSIONS

We introduced an objective function, ascending-CVaR, to be used in variational quantum algorithms for any combinatorial optimization problem. The starting point is the (constant) CVaR objective function of Ref. [11], where they illustrated that for any choice of risk $\alpha$ the true ground state is a minimum, and that with (typically small values of) $\alpha$ one can improve the performance compared to the standard expectation-value objective function. Our idea was to use an evolving objective function that passes through all the different values of $\alpha$ to finish at the expectation value. This, intuitively, avoids getting stuck at local minima since the energy landscape for different $\alpha$’s differs, apart from the global minimum.

We numerically tested the proposal on three combinatorial optimization problems (MaxCut, number partitioning, and portfolio optimization), where, in agreement with prior works, we found that for these problems VQE seems more promising than QAOA with small depth. The improvement that ascending-CVaR provides to VQE and QAOA is similar but we focus on VQE here since this was the overall more promising approach to solve the corresponding optimization problems.

![Fig. 9. Portfolio optimization problem for 18- and 20-asset instances with linear ascending and $\lambda = 0.045$. The blue line (diamond marker) indicates the CVaR$_{\alpha_t}$-VQE optimization which already within the first 15 optimizer iterations has achieved over 40% overlap compared to constant $\alpha$’s, where they either fail ($\alpha = 0.5, 1$) or lead to slower and suboptimal convergence ($\alpha = 0.1, 0.2$).]
We observed that ascending-CVaR gave much greater on average overlap with the optimal solution (see Table IV). In portfolio optimization and number partitioning, we got ten times greater overlap than the expectation value (while we got at least double overlap than the best constant CVaR choice). In MaxCut, we got smaller improvement (80%) compared to the expectation value, but note that the constant CVaR actually gave much smaller overlap. Perhaps the most important feature is that in the number partitioning and MaxCut, ascending-CVaR succeeded in finding the solution in many instances that no other approach achieved more than the small chosen threshold of 10% overlap. This indicates that not only the approach improves the quality of the results, but is plausible that instances that are believed to be hard with the other methods will become easy and thus solvable.

Beyond the accuracy of the result, another factor to evaluate the performance of variational quantum algorithms is the speed, which can be counted with respect to the (average) number of iterations the optimization needs to run until the algorithm outputs a (candidate) solution. Since our proposal passes through several choices of \( \alpha \), one could expect that the trade-off for better overlap would be slower speed and thus more optimization iterations. Interestingly, not only do we not get any cost in speed, but in most cases we see an improvement, i.e., our method requires fewer iterations to reach the threshold of 10% overlap with the solution (see Table V). The only case that our method required slightly more iterations than \( \alpha = 0.1 \) was for the set \( N_3 \), where the overlap was seven times better than the next-best case, and 350 times greater overlap than the expectation value (see Table II).

Our paper not only offers a generic method to improve the performance of variational quantum algorithms for combinatorial optimization problems, it also suggests a direction of research where dynamic objective functions can be used to boost the performance in terms of quality and speed of near-term quantum algorithms. An immediate follow-up to the proposal suggested here is to generalize our approach. Concretely, our method introduces two extra degrees of freedom: the hyperparameter \( \lambda \) and the function according to which the parameter \( \alpha \) increases. It is worth exploring a more systematic rule on how to fix these degrees of freedom according to the problem considered and the features of the specific instance. Finally, considering other dynamic objective functions is another direction that is worth pursuing.

The code for the experiments is available at Github [38].

### ACKNOWLEDGMENTS

P.W. acknowledges support by the UK Hub in Quantum Computing and Simulation, part of the UK National Quantum Technologies Programme with funding from UKRI EPSRC Grant No. EP/T001062/1 and the Collaborative Computational Project—Quantum Computing (CCP-QC) with funding from UKRI EPSRC Grant No. EP/T026715/1.

### APPENDIX A: VARIATIONAL QUANTUM ALGORITHMS

In this Appendix, we introduce the details of the two main variational quantum algorithms used in this paper.

#### a. Variational quantum eigensolver

The variational quantum eigensolver, as proposed by Ref. [39], is a hybrid quantum-classical algorithm, originally designed to solve quantum chemistry problems, but it can be used to tackle optimization problems [33]. The main idea is to map the optimization problem into a cost function that is translated into an interacting qubit Hamiltonian \( H_C \), whose ground state corresponds to the solution of the optimization problem.

The encoded qubit Hamiltonian, \( H_C \), is decomposed into a linear combination of Pauli strings \( P_\alpha \), consisting of tensor

---

**Table IV. Overview of our method.**

<table>
<thead>
<tr>
<th>Problem type</th>
<th>Successful instances %</th>
<th>Overlap improvement %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \alpha ) 0.1 0.2</td>
<td>( \alpha ) 0.1 0.2</td>
</tr>
<tr>
<td>Portfolio optimization</td>
<td>100 100 100</td>
<td>9782 2125 4023</td>
</tr>
<tr>
<td>Number partitioning</td>
<td>87.3 70 39.6</td>
<td>10225 1858 1464</td>
</tr>
<tr>
<td>MaxCut</td>
<td>96 84 81</td>
<td>78.5 −66.52 −40.8</td>
</tr>
</tbody>
</table>

**Table V. Average normalized optimizer iterations to achieve at least 10% overlap with the optimal solution for the three different combinatorial problems.**

<table>
<thead>
<tr>
<th>Problem type</th>
<th>Instance class</th>
<th>Average normalized iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>( \alpha ) 0.1 0.2</td>
</tr>
<tr>
<td>Portfolio optimization</td>
<td>Random portfolios</td>
<td>9.64 11.13 16.29</td>
</tr>
<tr>
<td>Number partitioning</td>
<td>( N_1 )</td>
<td>12.1 19.76 25.09</td>
</tr>
<tr>
<td></td>
<td>( N_2 )</td>
<td>14.73 24.13 28.86</td>
</tr>
<tr>
<td></td>
<td>( N_3^* )</td>
<td>27.12 25.12 33.62</td>
</tr>
<tr>
<td>MaxCut</td>
<td>Random graphs</td>
<td>8.75 9.3 10.8</td>
</tr>
</tbody>
</table>
products of Pauli Matrices $\hat{\sigma}^x, \hat{\sigma}^y, \hat{\sigma}^z$,

$$H_C = \sum_{k=1}^{M} c_k P_k,$$

where $M = O(\text{poly}(n))$, $n$ is the system size, and $c_k$ is the complex coefficient of the $P_k$ Pauli string. However, for combinatorial optimization problems where the Hamiltonian is a diagonal matrix, $H_C$ is decomposed only on Pauli strings consisting of $\sigma^z$ operators.

VQE is initialized by creating a parameterized state $|\psi(\theta)\rangle$ whose parameters are iteratively updated by a classical optimizer so as to minimize an objective function, usually the expectation value of Eq. (A1). The parameterized state is created by choosing a variational form $U(\theta)$ which acts on the initial state $|0\rangle^{\otimes n}$ and produces $|\psi(\theta)\rangle$.

Our choice of variational form $U(\theta)$ is a hardware-efficient ansatz [11,41], where the qubits are initialized in the $|0\rangle$ state and $R_i(\theta_i)$ rotations are applied in each qubit along with controlled-Z operators (see Fig. 10). Each layer of the variational form consists of $\text{CZ}_{ij}$ operations with $i$ the control qubit and $j$ the target qubit, as long as the condition $i < j$ holds, and $R_i(\theta_i)$ rotations for every qubit. If $p$ is the number of layers, then the number of parameters is linear, $\mu = n(1 + p)$, in the number of qubits and the variational form spans every basis state already within the first layer.

The hardware efficient ansatz falls in the more general category of problem-agnostic ansatze, meaning that the structure of the ansatz carries no information about the problem itself and is mostly suited for optimization problems. Other problems use different ansatz families, like the unitary coupled cluster, which is widely used in chemistry to obtain the ground state of a molecule [42], or the variational Hamiltonian ansatz, which encodes the problem’s Hamiltonian [43].

b. Quantum approximate optimization algorithm

The QAOA [17] is a variational quantum algorithm mostly used in combinatorial optimization problems and, while in shallow depths it is analytically and numerically explored for some problems [18,44], its performance in intermediate depths is still unknown.

The QAOA algorithm applies an alternation of two unitary transformations, one encoding the cost function $H_C$, $U(H_C) = e^{-i\gamma H_C}$ and the other a mixer Hamiltonian $H_B = \sum \sigma^z_i$, $U(H_B) = e^{-i\beta H_B}$, where $\gamma$ and $\beta$ are variational angles specifying the time for which the unitary transformations are applied. The system is initialized at the ground state of $H_B$ and the alternating ansatz of $U(H_B)U(H_C)$ is applied $p$ times, with $p$ defining the number of layers of the algorithm (see Fig. 11), producing the state

$$|\beta, \gamma\rangle = e^{-i\beta \gamma \gamma} e^{-i\gamma \gamma} \ldots e^{-i\beta \gamma \gamma} e^{-i\gamma \gamma}|+\rangle,$$

where $|+\rangle$ is the uniform superposition state, $\gamma = (\gamma_1 \ldots \gamma_p)$ and $\beta = (\beta_1 \ldots \beta_p)$.

With sufficient repetitions of the algorithm, the expectation value is calculated as

$$F_p(\beta, \gamma) = \langle \beta, \gamma | H_C | \beta, \gamma \rangle$$

until the $2p$ optimal parameters $(\beta^*, \gamma^*)$ are found.

If $C_{\text{opt}}$ is the optimal cost function, then the target of the algorithm is to maximize the approximation ratio, defined as

$$r^* = \frac{F_p(\beta^*, \gamma^*)}{C_{\text{opt}}}.$$  

Finding the optimal parameters is far from trivial since the expectation value landscape is highly nonconvex, filled with local minima where a classical optimizer could easily get stuck.

The hardest part of QAOA, and, in general, of a variational quantum algorithm, is finding the optimal parameters that will lead in a high overlap with the optimal (or near-optimal) bit-string or low expectation value. Recently, Ref. [45] proved that training the optimization parameters is NP-hard and that the landscape of the objective function is filled with far-from-optimal local minima. One way to avoid getting stuck in a local minima is using multistart methods [46] or heuristic methods like using the global optimum of one layer, in QAOA, as a starting point for the next [44].

FIG. 10. Single-layer hardware efficient ansatz for three qubits.

FIG. 11. General framework of a $p$-layer QAOA consisting of $2p$ variational angles.

FIG. 12. Probability of sampling an optimal solution over the circuit repetitions for a number-partitioning instance.
FIG. 13. Performance of ascending-CVaR algorithm with linear ascending for different choices of the ascending factor $\lambda$. The yellow (square marker) and green (down-pointing marker) lines which refer to $\alpha = 0.08$ and $\alpha = 0.1$, respectively, are not able to reach a good approximation to the optimal solution. On the other hand, the orange (line marker) and light blue (circle marker) lines which correspond to $\alpha = 0.04$ and $\alpha = 0.025$ are both able to achieve an overlap larger than 50%. Finally, the dark blue line (diamond marker) is still able to reach a good approximation to the ground state but it lacks in terms of speed of convergence and magnitude of overlap achieved. The graph on the left refers to a number partitioning problem while the graph on the right to the portfolio optimization problem.

APPENDIX B: CIRCUIT REPETITIONS

In this Appendix, we demonstrate how our method outperforms, in terms of real circuit repetitions and quality of the output state, the previously used objective functions. We set our default circuit repetitions to $K = 1000$ which we then scale up, along the discretely increasing $\alpha$ using the expression $K/\alpha$, for each given time. While one may think that this would weaken our results, as illustrated in Fig. 12, it seems that in terms of circuit repetitions our method converges to the chosen threshold of 10% faster than the best of constant CVaR or the expectation value approaches.

APPENDIX C: NUMERICAL ANALYSIS OF ASCENDING FACTOR

In this Appendix, we illustrate how the performance of our algorithm depends on the choice of the ascending factor $\lambda$. We numerically tested a large number of instances of sizes from 16 to 20 qubits and observed that the algorithm performed optimally for ascending factors drawn from the set $[0.025, 0.045]$. For this reason, as an example, we choose to draw the behavior of our algorithm for two random instances (one for portfolio optimization and one for number partitioning) for different choices of the hyperparameter $\lambda$ (see Fig. 13).

The performance of our method is sensitive to the choice of $\lambda$. A small $\lambda$ still converges to an optimal solution but requires a large number of iterations, compared to $\lambda$ chosen from the set $[0.025, 0.045]$, which is able to attain 10% within a small number of iterations. On the other hand, choosing $\lambda$ to be large (hoping for a faster convergence) fails to achieve even a minor overlap with the optimal solution. A careful tuning of $\lambda$ is therefore necessary for the optimal performance of the algorithm given the size and class of the problem at hand.


[38] https://github.com/ioanakolev/ascending_cv.


