A Comparative Study on Solving Optimization Problems With Exponentially Fewer Qubits

DAVID WINDERL, NICOLA FRANCO (Member, IEEE), AND JEANETTE MIRIAM LORENZ
Fraunhofer Institute for Cognitive Systems IKS, 80686 Munich, Germany
Corresponding author: Nicola Franco. (e-mail: nicola.franco@iks.fraunhofer.de)
This work was part of the Munich Quantum Valley, which was supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus.

ABSTRACT Variational quantum optimization algorithms, such as the variational quantum eigensolver (VQE) or the quantum approximate optimization algorithm (QAOA), are among the most studied quantum algorithms. In our work, we evaluate and improve an algorithm based on the VQE, which uses exponentially fewer qubits compared to the QAOA. We highlight the numerical instabilities generated by encoding the problem into the variational ansatz and propose a classical optimization procedure to find the ground state of the ansatz in fewer iterations with a better or similar objective. In addition, we propose a method to embed the linear interpolation of the MaxCut problem on a quantum device. Furthermore, we compare classical optimizers for this variational ansatz on quadratic unconstrained binary optimization and graph partitioning problems.

INDEX TERMS Hybrid algorithms, optimization, quantum computing.

I. INTRODUCTION AND RELATED WORK
Since the emergence of variational quantum algorithms (VQAs) [5] and their subclasses, such as the variational quantum eigensolver (VQE) [21] and the quantum approximate optimization algorithm (QAOA) [10], the amount of research contributions toward the practical utility of quantum computers has increased every year. Applications of mathematical optimization are of particular interest, and the QAOA is widely adopted in this context. Although the QAOA is a heuristic method, researchers are constantly advancing through slight improvements and precise adjustments to achieve better performance on narrow optimization problems. Most authors focus on NP-hard optimization problems, such as the traveling salesman problem (TSP) [11] or MaxCut [7]. A more convenient approach is to reduce such problems to quadratic unconstrained binary optimization (QUBO) [12] or Ising [15] formulations.

As highlighted in [13], the inherent flaws of current noisy intermediate-scale quantum (NISQ) computers may limit the actual speedup of large-scale optimization problems with the QAOA to devices with several hundred qubits. Hence, recent contributions have offered decomposition methods to overcome this issue and decompose the original problem to a size compatible with NISQ devices. In this context, decomposition methods, such as in [14] and [28], offer solutions to overcome these limitations by splitting the original problem into smaller subproblems with sizes matching the requirements of NISQ hardware. In [14], after dividing a graph into two subgraphs that share common nodes, to obtain a possible candidate, the solutions of the respective subgraphs must overlap exactly. However, the complexity of these approaches increases with the number of common nodes, complicating the search for a viable candidate solution. Extending this, a new encoding strategy in [24] opens a new perspective as it requires exponentially less qubits to solve MaxCut with a VQE ansatz. Chatterjee et al. [6] provide a generalization of [24] for a large class of NP-hard problems.

In this work, we summarize the approach of [24], highlight its limitations, and propose improvements via a new encoding function. In addition, we discuss the solution landscape, propose an interpolation method that reshapes it, and contribute to the convergence of optimizers. We conduct experiments on structured and random QUBO problems in both the simulated and real quantum devices to demonstrate the viability of our improved approach.

In summary, our main contributions are as follows.
1) We propose a new version for the encoding function of [24], which solves numerical instabilities and leads to stable results while maintaining an exponential reduction of qubits.

2) We provide an alternating optimization (AltOpt) algorithm specifically tailored to the solution landscape described by Rančić [24].

3) We improve the solution landscape of the variational ansatz by linearly interpolating regions to avoid the vanishing gradient problem for specific regions of the ansatz solution landscape.

Despite these advancements, we acknowledge that optimizing VQAs remains an NP-hard challenge [4].

II. BACKGROUND

A. PRELIMINARIES

Considering a weighted graph $G = (V, E)$, we denote the set of vertices as $V = \{1, \ldots, n\}$ and the set of edges as $E = \{(i, j) \mid (i, j) \in V \times V\}$, with weights $w_{ij} \in \mathbb{R}$ for $(i, j) \in E$.

In the weighted MaxCut problem, a cut is defined as a partition of the original set $V$ into two subsets. Given a binary vector $x \in \{0, 1\}^n$, the cost function to be maximized is given by $C(x) = \sum_{i,j} w_{ij} k_i (1 - x_j)$. It represents the sum of weights of edges connecting points in the two different subsets, i.e., crossing.

MaxCut is a specific case of a more general class of optimization problems that fall under the QUBO umbrella. In the context of QUBO, one wants to minimize the following cost function: $f(x) = x^T Q x$, where $x \in \{0, 1\}^n$ is a binary vector and $Q \in \mathbb{R}^{n \times n}$ is a square matrix of real values. We can rewrite a QUBO problem to its Ising variant by setting $f(\sigma) = -\ln q(\vec{\sigma}, \alpha)$.

We want to point out to the reader that while the individual terms $e^{i \pi_1 R_f(\alpha_i, \alpha_j, m)}$ seem complex valued, they are map toward the states $\psi(\alpha_i) = e^{i \pi_1} = 1$ (in case of $R_f(\alpha_i, 0, m) = 1$) and $\psi(\alpha_j) = e^{i \pi_0} = 1$ (in case of $R_f(\alpha_i, 0, m) = 0$). We can, hence, see that this is approximately the same as computing $2R_f - 1$.

The vectors $v \in \{-1, 1\}^{d_l}$ are created for each $\alpha_i$, where $\sum_{i=1}^{k} d_i + \rho = n$. Here, $\rho$ describes the amount of ones we are using for padding the diagonal. Each $d_i$ is then equivalently partitioned over the remaining elements to stretch dimensions. This leads to the diagonal gate $U(\alpha)$, defined as follows:

$$U(\alpha) = \text{diag}(v(\alpha_1), \ldots, v(\alpha_k), 1, \ldots, 1)$$

$$= \text{diag}(e^{i \pi_1 R_f(\alpha_1, 0, m)}, \ldots, e^{i \pi_1 R_f(\alpha_k, 1, m)}, \ldots, e^{i \pi_1 R_f(\alpha_1, d_i, m)}, \ldots, e^{i \pi_1 R_f(\alpha_k, d_i, m)}, 1, \ldots, 1).$$

The size of $U(\alpha)$ is $2^{\lceil \log_2(n) \rceil}$. Shende et al. [25] have provided a way to realize such a diagonal gate $U(\alpha)$ on a quantum device. Next, since the graph’s Laplacian is a Hermitian matrix, we can effectively encode it into a sum of tensor products of Pauli matrices [19], [22]. In this context,
Rančić [24] argues that the summands of such a decomposed hermitian are polynomially bounded by $O(|V|^2)$. Hence, we can measure the ansatz on an individual basis, provided by the tensor product of Pauli matrices, and compute the expected value as the weighted sum of the individual measurements using $O(|V|^2)$ many queries. Therefore, we can construct the variational ansatz as follows:

$$N_{\text{CUTS}} = \frac{1}{4} V^T L(G) V$$

$$= 2^{n-2} (0) | HU(\alpha) L(G) U(\alpha) H | 0 \rangle$$

$$= 2^{n-2} \sum_i (0) | HU(\alpha) P_i U(\alpha) H | 0 \rangle$$  \hspace{1cm} (3)

where $H$ describes the Hadamard transform and $L(G)$ is represented as a sum of tensor products of Pauli matrices and denoted as $L(G) = \sum_i P_i$ in such form. Finally, the overall required gates to realize this ansatz remain to be stated. We will argue in the computational overhead of CNOTs since, typically, those gates are the most expensive to implement on an NISQ device. As already pointed out, there are $O(|V|^2)$ many queries required to the quantum device, where at each query, we have to implement a Hadamard transform (no CNOTs are required) and a diagonal gate, which requires $O((23/48) 4^n - (\sqrt{2}) 2^n + \sqrt{3})$ many CNOTs [26]. Since $n = \lceil \log_2(|V|) \rceil$, the overall amount of CNOTs is asymptotically bounded by $O(|V|^2)$, which requires $O(|V|^2)$ many CNOTs overall to be executed on a quantum device.

In [24], it is important to note that the reduction in dimensionality is achieved by encoding the matrix multiplication into the amplitudes of the quantum device. $R_f$ in (1) can be used to map the exponential solution space of MaxCut to a continuous solution space of an arbitrary dimension $k$. It is worth to mention that this function is highly numerically unstable for larger graphs. To make this clear, let us assume a graph with $n = 6$; then, set $m = 6$ (since according to [24], $m \geq n$) and compute $R_f$ for $\alpha_1 = \pi/2$ and $q = 0$. In Fig. 1(b) one can see that partially evaluating $\exp(2^{6-0} \cdot \sin(2^{6} \pi + \alpha_0(0, 6))) \approx 6.2 \times 10^{27}$ already results in a large number. Hence, computing this function for large graphs with not enough dimensions for $\alpha$ is not feasible.

III. METHODS

As already pointed out, our main contributions are threefold. First, we discuss the improvements one can make to the function $R_f$ to encode the solution landscape. Second, we derive an AltOpt procedure tailored to the posed problem setting. Finally, we progress toward improving the solution landscape by linearly interpolating the problem on a quantum device.

A. IMPROVEMENTS ON $R_f$

As previously discussed, $R_f$ creates alternating plateaus on either zero or one, which are then used to enumerate all the possible binary combinations of length $2^n$. First, it is important to note that the version of [24] is both continuous and differentiable. Nevertheless, in case one wants to perform gradient-based optimization of this problem, it is possible to do this by computing the partial derivative. Furthermore, there seems to be no use of continuity in terms of optimizing the solution landscape. Therefore, continuity and differentiability, as described in [24], are not necessary for optimizing the solution landscape in general. Finally, given the optimized parameters, one can find that reconstructing the vectors $v$ is straightforward by reapplying the encoding.

However, during the empirical assessment of (4), we faced technical obstacles that $ParameterExpressions$ in Qiskit [23] do not support cased functions. \footnote{1 We used Qiskit 0.36.1 in our experiments.} Therefore, we propose an alternative definition, which is numerically stable and utilizes a modified version of the sawtooth wave

$$s(\alpha_i, q) = 1 - 2 \arctan(\cot(2^{q-1} \alpha_i)). \hspace{1cm} (5)$$

Hence, we can mimic $R_f$ with $[s(\alpha_i, q)]$, where $[\cdot]$ represents the floor function. Since the floor function is nonlinear, we can approximate it as

$$R_f(\alpha_i, q) = s(\alpha_i, q) - \frac{1}{2} s(2^{1-q} \pi s(\alpha_i, q), q). \hspace{1cm} (6)$$

In Fig. 2, we show both $s(\alpha_i, q)$ and $R_f(\alpha_i, q)$ for a given $\alpha_1$. In the end, we observe two advantages of the proposed function: 1) the values of $s(\alpha_i, q)$ are contained within the interval $[0, 2]$, which makes $R_f(\alpha_i, q)$ numerically stable and 2) the reduction in the number of hyperparameters due to the removal of $m$. 

![FIGURE 2. $s(\alpha_i, q)$ and $R_f(\alpha_i, q)$ for $q \in \{0, 1, 2\}$.](image-url)
Example graph with its Laplacian matrix $L(G)$.

**B. ALTOPT PROCEDURE**

Here, we present our algorithm based on the AltOpt strategy [3] to find the ground state for the variational ansatz in (3). Essentially, this procedure uses the fact that the maximum number of variables in $\alpha$ can be determined as $|V| - 1$. Given that, one can decide per diagonal entry of $U(\alpha)$ to set it to $-1$ or $1$. In the space of our continuous variable $\alpha_i$, this corresponds to the values $\pi/2$ or $3\pi/2$. Thus, we propose the following procedure to minimize (3).

1) Initialize all the diagonal entries to $1$ (set their corresponding variable to $\pi/2$).
2) Travel the diagonal from the top left to bottom right, selecting the value with lower energy each time.
3) Restart from the top left until the objective no longer improves.

The proposed optimization method has several good properties: hyperparameter-free, faster convergence, and less dependence on the initial choice of the parameters. Nevertheless, it is specific for the variational ansatz of (3).

**C. IMPROVEMENTS ON THE SOLUTION LANDSCAPE**

The solution space described by Rančić [24] is continuous, so the classical optimizer can select any value $\alpha_i \in [0, 2\pi]$. Nevertheless, it shows barren plateaus in particular areas depending on the cut value of MaxCut. This circumstance can be viewed as the nearest neighbor interpolation of the discrete solution space defined by MaxCut.

In the following, we first define a more abstract way of this nearest neighbor interpolation. We then continue to extend this concept toward a linear interpolation of the problem. The main benefit we see in this extension is that the classical optimizer is guided in the states among the solutions to MaxCut. We consider the graph in Fig. 3 with the corresponding Laplacian matrix to showcase the results of interpolating the solution landscape.

1) NEAREST NEIGHBOR INTERPOLATION

As discussed in Section II-B, the key point of the algorithm in [24] is to convert the continuous vector $\alpha$ into a discrete vector of spins $v$, effectively enumerating every binary combination depending on the selected interval. We abstract this behavior by the function $\psi: \mathbb{R} \times \{0, 1\} \rightarrow \{-1, 1\}^{n-k}$, defined as

$$\psi(\alpha, b) = \begin{cases} 2\text{bin}([\alpha_i]) - 1, & b = 1 \\ 2\text{bin}([\alpha_i]) - 1, & b = 0. \end{cases} \quad (7)$$

Note that we decide not to use the fixed interval $\alpha_i \in [0, 2\pi]$ and move toward an interval $\alpha_i \in [0, 2^{n-k}]$. Nevertheless, since $R_f$ mimics the behavior of a binary encoding, one can see that both the methods are equivalent. Then, we can either floor or ceil $\alpha_i$ depending on the binary variable $b \in \{0, 1\}$. In the end, we compute the standard binary decomposition of the resulting integer and convert it toward $\{-1, 1\}$ base. For example, assume that we have an input $\alpha_i = 2.5$ with $b = 1$; hence, $\text{bin}([\alpha_i]) = \text{bin}(3) = (0 \ 1 \ 1)$. We would, hence, obtain $\psi(2.5, 1) = (-1 \ 1 \ 1)$. Thus, we can achieve nearest neighbor interpolation by setting

$$U'(\alpha) = \text{diag}(\psi(\alpha_1, 0), \ldots, \psi(\alpha_k, 0), 1, \ldots, 1) \quad (8)$$

where $b$ is fixed to $0$ for all the elements on the diagonal. This results in the solution landscape outlined by Rančić [24].

In Fig. 4, we show the landscape for the example of Fig. 3. Fig. 4(a) illustrates the landscape with nearest neighbor interpolation for a 1-D case, while Fig. 4(b) for a 2-D case.

2) LINEAR INTERPOLATION

To perform the linear interpolation of the problem, one needs to reformulate the expected value of the ansatz by including a binary vector $b \in \{0, 1\}^k$ into the evaluation of the expected value, with $b_1, \ldots, b_k$ as entries of $b$ and $k \leq n$. Therefore, we define the diagonal operator as follows:

$$U''(\alpha, b) = \text{diag}(\psi(\alpha_1, b_1), \psi(\alpha_2, b_2), \ldots, \psi(\alpha_k, b_k), 1, \ldots, 1). \quad (9)$$

We can then compute the expected value of the variational circuit as follows:

$$E(\alpha, b) = 2^{n-2}\langle 0 \vert H U''(\alpha, b) L(G) U'(\alpha, b) H \vert 0 \rangle$$

$$= \frac{1}{4} \sum_{i,j} \tilde{v}_i(\alpha, b) \cdot \tilde{v}_j(\alpha, b) L_{i,j} \quad (10)$$

$^2$In the context of [24], the interval is $[0, 2\pi]$.

$^3$In practice, $k \ll n$. 

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Here, $\sigma(\alpha, b)$ is the prefactor that determines the weight of the vertex value of the hypercube specified by the binary vector $b$. We can then formulate the interpolation for a vector of parameters $\alpha$ as follows:

$$E(\alpha) = \sum_{b \in \{0, 1\}^k} E(\alpha, b \cdot \sigma(\alpha, b))$$

$$= \sum_{p=0}^{2^k-1} E(\alpha, \text{bin}(p)) \cdot \sigma(\alpha, \text{bin}(p))$$

$$= \frac{1}{4} \sum_{p=0}^{2^k} \sum_{i,j} v_i(\alpha, \text{bin}(p)) \cdot v_j(\alpha, \text{bin}(p))$$

$$\cdot L_{i,j} \cdot \sigma(\alpha, \text{bin}(p)).$$

(12)

Note that if one interpolates this problem classically, it would require an exponential computational overhead regarding the number of variables $k$. Nevertheless, we can foster a speedup by embedding this interpolation into the amplitudes of our quantum device. To do this, one can define $p^{(i)} = \text{bin}(\frac{i}{2^k})$ and its bitwise negation $\overline{p}^{(i)}$, which leads to the following definition of $\hat{v}(\alpha) \in \mathbb{C}^{2^{2k} \cdot n}$:

$$\hat{v}(\alpha) = \sqrt{\sigma(\alpha, p^{(i)}) \cdot \hat{v}_i(\alpha, p^{(i)}) - i \sqrt{\sigma(\alpha, \overline{p}^{(i)}) \cdot \hat{v}_i(\alpha, \overline{p}^{(i)})}}.$$  

(13)

In an analogy toward [24], we can define a variational ansatz as

$$E'(\alpha) = 2^{n-2} \langle 0 | H U''''(\alpha) L(\mathcal{G})^{(k)} U'''(\alpha) H | 0 \rangle$$

$$= \frac{1}{4} \sum_{i,j} \hat{v}_i(\alpha) \cdot \hat{v}_j(\alpha) L_{i,j}^{(k)}$$

(14)

where we define $L(\mathcal{G})^{(k)} = (I^{(k-1)} \otimes L(\mathcal{G}))$ and the diagonal gate $U'''(\alpha) = \text{diag}(\hat{v}(\alpha))$. Finally, it is necessary to show that $\text{Real}(E'(\alpha)) = E(\alpha)$.

Therefore, note that

$$\text{Real}(\hat{v}_i(\alpha) \hat{v}_j(\alpha)) = \sqrt{\sigma(\alpha, p^{(i)}) \sigma(\alpha, p^{(j)})}$$

$$\cdot \hat{v}_i(\alpha, p^{(i)}) \cdot \hat{v}_j(\alpha, p^{(j)})$$

$$+ \sqrt{\sigma(\alpha, \overline{p}^{(i)}) \sigma(\alpha, \overline{p}^{(j)})}$$

$$\cdot \hat{v}_i(\alpha, \overline{p}^{(i)}) \cdot \hat{v}_j(\alpha, \overline{p}^{(j)}).$$

(15)

Next, note that for all $i, j \in [1 \cdot n, n(l+1)]$ for $0 \leq l \leq k$, $p^{(i)} = p^{(j)} = l$, with defining $b_i := \text{bin}(l)$. In addition, we set $\overline{p}^{(i)} = \overline{p}^{(j)} = \overline{l}$, for reasons of conciseness, we define the elementwise negation of $l$ as $\overline{l}$. This holds since we define $p^{(i)} = \text{bin}(\frac{l}{2^k})$.

We can, therefore, find for all $i, j \in [1 \cdot n, n(l+1)]$

$$\sqrt{\sigma(\alpha, p^{(i)}) \sigma(\alpha, p^{(j)})} = \sigma(\alpha, b)$$

(16)

* Real describes the operator that selects the real part of a complex number

FIGURE 4. Solution landscape with nearest neighbor interpolation. (a) One dimension. (b) Two dimensions.

FIGURE 5. Labeling of the edges of a 3-D hypercube.

where we define the vector $\hat{v}(\alpha, b)$ and the diagonal of the diagonal gate $U''''(\alpha)$ as follows:

$$\hat{v}(\alpha, b) = (\psi(\alpha_1, b_1), \ldots, \psi(\alpha_k, b_k), 1, \ldots, 1).$$

Intuitively, the resulting expected value describes a corner of the hypercube for each binary vector $b$. For instance, we can label those corners according to Fig. 5. If the classical optimizer now picks a point inside this hypercube, we can compute the value by linear interpolation among all the corners. Therefore, we define

$$\sigma(\alpha, b) = \prod_{i=1}^{k} \begin{cases} 1 - (\alpha_i - \lfloor \alpha_i \rfloor), & b_i = 0 \\ \alpha_i - \lfloor \alpha_i \rfloor, & b_i = 1. \end{cases}$$

(11)
\[
\sqrt{(\sigma(\alpha, \hat{p}^{(i)}) \sigma(\alpha, \hat{p}^{(j)})) = \sigma(\alpha, \hat{l}_b).}
\]  

(17)

For any combination where either \(i\) or \(j\) is outside of the interval \([l \cdot n, n(l + 1)]\), we see that \(L(G)_{i,j} = 0\). Hence, we rewrite the ansatz in (14), as follows:

\[
\text{Real}(E'(\alpha)) = \frac{1}{4} \sum_{i,j}^{2^{k-1}} \text{Real}\left(\hat{v}_i(\alpha)\hat{v}_j(\alpha)(I^{(k-1)} \otimes L(G))_{i,j}\right)
= \frac{1}{4} \sum_{i,j}^{2^{k-1}} \text{Real}(\hat{v}_i(\alpha)\hat{v}_j(\alpha))(I^{(k-1)} \otimes L(G))_{i,j}
= \frac{1}{4} \sum_{i,j}^{2^{k-1}} \text{Real}(\hat{v}_i(\alpha)\hat{v}_j(\alpha))(I^{(k-1)} \otimes L(G))_{i,j}
= \frac{1}{4} \sum_{i,j}^{2^{k-1}} \sigma(\alpha, \hat{l}_i)\sigma(\alpha, \hat{l}_j)\hat{v}_i(\alpha, \hat{l}_i)\hat{v}_j(\alpha, \hat{l}_j)L(G)_{i,j}
= \frac{1}{4} \sum_{i,j}^{2^{k-1}} \sigma(\alpha, \hat{l}_i)\sigma(\alpha, \hat{l}_j)\hat{v}_i(\alpha, \hat{l}_i)\hat{v}_j(\alpha, \hat{l}_j)L(G)_{i,j}
= E(\alpha).
\]  

(18)

In order to obtain the last step, it is important to note that we encode half of the dimensions into the complex part of the variational ansatz by inverting the binary number. Therefore, we rewrite the two sums by changing the last value of the sum from \(2^{k-1}\) to \(2^k\). Finally, we can conclude that the real part of our variational ansatz results in a linear interpolation among the states described by the corners of a \(k\)-dimensional hypercube.

To visualize this ansatz, see Fig. 6 for a linear interpolation among the states of MaxCut in one dimension and two dimensions. We want to note that computing the gradient as in (13) leads to numerical instabilities since this would require evaluating an infinitesimally small value when computing the derivative of a square root of \(\sigma\), which is smaller than one. Nevertheless, this can be bypassed by recalling that according to (18), the ansatz is simply the linear interpolation of all the corners of a \(k\)-dimensional hypercube. In this sense, an interesting aspect would be constructing a variational ansatz to evaluate the partial derivative with respect to a specific point \(\alpha\) and an element \(\alpha_i\).

Finally, it remains to conduct a complexity analysis of the gate and qubit complexities studied by this novel approach. For both the qubit and gate complexities, note that the length of the vector of the diagonal gate \(v(\alpha)\) is \(2^{(k-1)n}\). Hence, overall \(O(\log_2(2^{(k-1)n})) = O(k\log_2(n))\) many qubits are required to implement the diagonal gate.

For any combination where either \(i\) or \(j\) is outside of the interval \([l \cdot n, n(l + 1)]\), we see that \(L(G)_{i,j} = 0\). Hence, we rewrite the ansatz in (14), as follows:

\[
\text{Real}(E'(\alpha)) = \frac{1}{4} \sum_{i,j}^{2^{k-1}} \text{Real}\left(\hat{v}_i(\alpha)\hat{v}_j(\alpha)(I^{(k-1)} \otimes L(G))_{i,j}\right)
= \frac{1}{4} \sum_{i,j}^{2^{k-1}} \text{Real}(\hat{v}_i(\alpha)\hat{v}_j(\alpha))(I^{(k-1)} \otimes L(G))_{i,j}
= \frac{1}{4} \sum_{i,j}^{2^{k-1}} \text{Real}(\hat{v}_i(\alpha)\hat{v}_j(\alpha))(I^{(k-1)} \otimes L(G))_{i,j}
= \frac{1}{4} \sum_{i,j}^{2^{k-1}} \sigma(\alpha, \hat{l}_i)\sigma(\alpha, \hat{l}_j)\hat{v}_i(\alpha, \hat{l}_i)\hat{v}_j(\alpha, \hat{l}_j)L(G)_{i,j}
= \frac{1}{4} \sum_{i,j}^{2^{k-1}} \sigma(\alpha, \hat{l}_i)\sigma(\alpha, \hat{l}_j)\hat{v}_i(\alpha, \hat{l}_i)\hat{v}_j(\alpha, \hat{l}_j)L(G)_{i,j}
= E(\alpha).
\]  

(18)

In order to obtain the last step, it is important to note that we encode half of the dimensions into the complex part of the variational ansatz by inverting the binary number. Therefore, we rewrite the two sums by changing the last value of the sum from \(2^{k-1}\) to \(2^k\). Finally, we can conclude that the real part of our variational ansatz results in a linear interpolation among the states described by the corners of a \(k\)-dimensional hypercube.

To visualize this ansatz, see Fig. 6 for a linear interpolation among the states of MaxCut in one dimension and two dimensions. We want to note that computing the gradient as in (13) leads to numerical instabilities since this would require evaluating an infinitesimally small value when computing the derivative of a square root of \(\sigma\), which is smaller than one. Nevertheless, this can be bypassed by recalling that according to (18), the ansatz is simply the linear interpolation of all the corners of a \(k\)-dimensional hypercube. In this sense, an interesting aspect would be constructing a variational ansatz to evaluate the partial derivative with respect to a specific point \(\alpha\) and an element \(\alpha_i\).

Finally, it remains to conduct a complexity analysis of the gate and qubit complexities studied by this novel approach. For both the qubit and gate complexities, note that the length of the vector of the diagonal gate \(v(\alpha)\) is \(2^{(k-1)n}\). Hence, overall \(O(\log_2(2^{(k-1)n})) = O(k\log_2(n))\) many qubits are required to implement the diagonal gate.

\[
I^{(k-1)} \otimes L(G) = I^{(k-1)} \otimes \left(\sum_i P_i\right) = \sum_i \left(I^{(k-1)} \otimes P_i\right).
\]  

IV. EXPERIMENTS

It remains to verify our proposed methods experimentally; therefore, we introduce three experiments. First, we evaluate the solution landscape provided by the \(R_f\) function proposed in [24] against our version \(R_f\). Next, we compare the performance of different classical optimizers, including our optimization strategy, AltOpt. Finally, we verify the performance.
increase using the linearly interpolated solution landscape previously derived.

### A. IMPACT OF $R_f$ ON THE SOLUTION LANDSCAPE

When observing the solution landscape of [24], one can notice the occurrence of spikes among the plateaus. In this section, we want to outline our reasoning for those spikes and argue why our formulation of $R_f$ effectively eliminates them. The underlying graph used for this observation can be found in Fig. 7. Following [24], we can find the constraint of $m_r \geq \frac{2^n}{r} + 2$. Here, $r$ describes the position of the element on the diagonal. Hence, for $n = 4$ and $d_k = 1$, we evaluate the solution landscape three times: 1) $m = 30 > \frac{2^n}{r} + 2$; 2) $m = \frac{2^n}{r} + 2$; and 3) with $m = 5 < \frac{2^n}{r} + 2$. In addition, we once evaluate our version of the sawtooth wave. Observing Fig. 8, we can see that no spike is formed for a sufficiently large $m$. Nevertheless, for both the case on the constraint and the case below, the constraint will introduce spikes with undefined values on the solution landscape (observe $x = \pi$).

For our definition of $R_f'$, one can see that this is not the case. For a more analytical insight into this observation, we compute the $R_f$ values for $x_0 = \pi - 0.05, x_1 = \pi$, and $x_2 = \pi + 0.05$. We settled with $m = 5$ since this is the extreme case

\[
R_f(x_0, 0, 5) \approx 0 \quad R_f(x_0, 1, 5) \approx 0.99 \\
R_f(x_1, 0, 5) \approx 0.24 \quad R_f(x_1, 1, 5) \approx 0.50 \\
R_f(x_2, 0, 5) \approx 0.99 \quad R_f(x_2, 1, 5) \approx 0.00
\]

while when computing those values with the sawtooth wave, we obtain

\[
R_f'(x_0, 0) = 0 \quad R'_f(x_0, 1) = 0 \\
R_f'(x_1, 0) = 0 \quad R'_f(x_1, 1) = 1 \\
R_f'(x_2, 0) = 0 \quad R'_f(x_2, 1) = 1.
\]

Based on the example we have examined, it is evident that $R_f$ can assume values within the range of $[0, 1]$. This can result in the undesired spikes at the plateau transitions, illustrated in Fig. 8.

From this example, we can see that our version is a clear improvement in terms of numerical feasibility, the impact on the solution landscape, and the required hyperparameters.

### B. PERFORMANCE OF ALTOPT AGAINST OTHER OPTIMIZERS

We further compare the algorithm of [24] on three optimization problems: 1) randomly generated QUBOs; 2) TSP; and 3) MaxCut instances. To this end, we used a selection of classical optimizers to optimize the VQE ansatz: Nakanishi–Fujii–Todo (NFT) algorithm [16], basin hopping

---

With a maximum of 200 iterations and 500 function evaluations.
TABLE 1. Evaluation of MaxCut on d-Regular Graphs for Different Sizes

| Size | Cut value \( \uparrow \) | Number of Iterations \( \downarrow \) |
|------|-----------------|------------------|
|      | GA | AltOpt | BH | NPT | GA | AltOpt | BH | NPT |
| 16   | 32.2 | 32.3 | 30.8 | 31.5 | 219.4 | 45.2 | 193 | 293 |
| 32   | 125.1 | 127.4 | 127.8 | 127.0 | 340.7 | 120.7 | 201 | 394 |
| 64   | 505.0 | 512.7 | 500.5 | 511.4 | 404.4 | 355.8 | 201 | 402 |
| 128  | 2033.2 | 2050.2 | 2041.7 | 2040.7 | 404.9 | 1016.6 | 201 | 408 |

The bold values represent the best ones.

TABLE 2. Evaluation on Random QUBOs of Different Sizes

| Size | Energy \( \downarrow \) | Number of Iterations \( \downarrow \) |
|------|-----------------|------------------|
|      | Tabu | GA | AltOpt | BH | NPT | GA | AltOpt | BH | NPT |
| 15   | -8.6 | -8.4 | -8.2 | -5.6 | -4.5 | 228 | 43 | 195 | 314 |
| 31   | -25.4 | -22.8 | -24.5 | -11.7 | -5.9 | 345 | 119 | 201 | 394 |
| 63   | -72.9 | -9.3 | -73.1 | -14.4 | -43.7 | 399 | 298 | 201 | 402 |
| 127  | -203.9 | 1.1 | -199.4 | -25.6 | -46.0 | 401 | 806 | 201 | 408 |

The bold values represent the best ones.

(BH) [17], as well as the genetic algorithm (GA) with the settings described in [24], and our proposed new method AltOpt. Note that we consider the \( R_f \) version of [24] for the GA, while we consider our \( R'_f \) version for NFT, AltOpt, and BH. For a smaller subset of random QUBOs, we conduct the study using real hardware. Finally, we compare our energy to the Tabu Solver\(^7\) of Dwave [18], which is also a search heuristic, to solve QUBO or Ising problems of a larger size. The number of iterations is defined as a repetition loop between the classical and quantum solvers. We used this metric instead of measuring time in seconds because time additionally depends on the underlying hardware, the duration of simulation, or the latency toward the IBM-Cloud.

1) MAXCUT

Our first experiment was conducted on different instances of MaxCut on d-regular graphs. For the degree of the graph \( d \), we used nine uniformly distributed values between 0 and the size of the graph. We use 20 graphs per factor of \( d \), assign uniformly random edge weights in the interval \([0,5]\), and average the resulting cut value. The results are shown in Table 1. Interestingly, all the optimization algorithms tend to perform similarly. We account this and the difficulties to solve QUBO toward the fact that the reduction of QUBO matrices to MaxCut does not provide d-regular graphs.

2) RANDOM QUBOS ON SIMULATED HARDWARE

Next, we generated random QUBO matrices of different sizes to test the average performance of each algorithm. For each size, we average over densities (how many nonzero elements) in \([0.1, 0.2, \ldots, 0.9]\) with 20 samples per size and density. All the elements of matrix \( Q \) are sampled from a normal distribution with mean 0 and variance 1. Then, we evaluate the symmetric matrix \( \frac{1}{2}(Q + Q^T) \).

The results are presented in Table 2. We find that for random QUBOs, AltOpt shows comparable results to the Tabu solver in terms of energy. The GA demonstrates similar energy to AltOpt for small instances; nevertheless, for instances of 63 and 127, the energy is much larger. Both BH and NFT have worse results on average.

3) TRAVELING SALESMAN PROBLEM

Another question we asked ourselves was how does the approach of [24] perform on problems that provide an inherent structure to the graph or QUBO problem. For this reason, we took the TSP, which comes along with a QUBO or Ising formulation [15]. For a fair comparison, we evaluate the results in terms of energy, number of iterations, and normalized feasibility.\(^8\) A solution is feasible if it produces a Hamiltonian path through the graph. Given the results in Table 3, we see a similar pattern of random QUBOs. Furthermore, it is worth noting that AltOpt returns a substantial percentage of feasible solutions. On the one hand, the GA, BH, and NFT struggle to explore the solution landscape and stop at unfeasible local minima. On the other hand, the maximum number of iterations set at 400 for the GA and NFT and 200 for BH limits their search capabilities. With this observation in mind, we direct the reader toward [4]. In their work, they show that finding the ground state of a VQA is NP-hard. Hence, we cannot generally expect the algorithm to solve MaxCut effectively. Nevertheless, we can see that AltOpt provides an approximate solution to the problem. We can see that the number of iterations follows a polynomial pattern for AltOpt; nevertheless, the approximation ratio decreases with the size of the problem. We can, hence, conclude that other heuristics, such as a branch and bound method for instance, could provide further improvements on the approximation factor. Nevertheless, we do not expect it to asymptotically outperform one-fourth the approximation factor given by Bittel and Kliesch [4].

4) RANDOM QUBOS ON REAL HARDWARE

In addition to simulated hardware, we conduct experiments on gate-based quantum computers. In this context, we access the seven qubits systems: lagos and jakarta, with Qiskit [23] from IBM Quantum,\(^9\) both based on the Falcon r5.11 architecture, and having a quantum volume of 32 and 16, respectively. In Table 4, we present results for a random QUBO problem of size 31 over three density levels. The performance of AltOpt, BH, and NFT is lower in all the three tests, with AltOpt and BH performing better in general. We attribute this behavior at the intrinsic noise generated by the system, where to improve the result quality, a larger number of shots will be required, as demonstrated in [24], where 8192 was used instead of 1024 of our evaluation.

\(^7\)With initial state \( \pi \), step size of 2\( \pi \), an interval to update the step size of 10, a maximum of 200 iterations, and a local minimizer COBYLA with a rhogeb of \#Variables/2\(^{30}\).

\(^8\)Dwave-tabu. [Online]. Available: https://github.com/dwavesystems/dwave-tabu

\(^9\)With normalized feasibility, we refer to the fraction number of feasible samples/number of samples.

\(^{10}\)IBM Quantum. [Online]. Available: https://quantum-computing.ibm.com/
TABLE 3. Evaluation on Randomly Generated TSP Instances for Different Sizes

| Size | Energy | Tabu | GA | BH | NPT | Feasibility | Number of iterations |
|------|--------|------|----|----|-----|-------------|---------------------|
|      |        | Tabu | AltOpt | BH | NPT | Tabu | AltOpt | BH | NPT | Tabu | AltOpt | BH | NPT |
| 3    | 349.9  | 349.9| 349.9| 410| 349.9| 1 | 0.65  | 0.45 | 199 | 18 | 240 | 344 |
| 5    | 771.2  | 721.5| 727.2| 280| 902.7| 1 | 0.75  | 0.95 | 339 | 74 | 201 | 406 |
| 7    | 1158.9 | 666.35| 1036.7| 1857.3| 406.8| 1 & 0 | 0 | 404 | 141 | 201 | 490 |
| 9    | 1507.4 | 1650.4| 1552.6| 2717.1| 10795.1| 1 | 0.95  | 0 | 405 | 241 | 201 | 408 |
| 11   | 1898.9 | 10503.5| 1627.8| 56191.7| 29490.3| 1 | 0 | 405 | 339 | 201 | 408 |

We take the average over 20 samples per size.

The bold values represent the best ones.

TABLE 4. Evaluation on Random QUBO Problems With Quantum Hardware

| Size | Density | Energy | Tabu | AltOpt | BH | NPT | Number of iterations |
|------|---------|--------|------|--------|----|-----|---------------------|
|      |         |        | Tabu | AltOpt | BH | NPT | Tabu | AltOpt | BH | NPT |
| 3    | 0.1     | -10.8  | -1.1 | -3.8  | 0.4 | 83  | 201 | 408 |
| 5    | 0.5     | -24.8  | -4.3 | -0.3  | 14.5 | 63  | 201 | 396 |
| 7    | 0.9     | -27.8  | -1.3 | -1.7  | 12.9 | 70  | 201 | 395 |

We run three instances per algorithm and average the energy and number of iterations. The bold values represent the best ones.

C. MAXCUT INTERPOLATION

In this section, we explore the following question: How does the performance of “gradient-based” optimizers, such as COBYLA or SPSA, change when we introduce interpolation? Therefore, we consider MaxCut instances on Erdős–Rényi Graphs [9] of sizes in \{8, 16, 32\} with densities in \{0.3, 0.5, 0.7\}. We generate 20 instances for each configuration of those parameters. In addition, we select the set of parameters \(k \in \{2, 4, 5, 7\}\). We then compare the linear and nearest neighbor interpolation of the problem for each number of parameters \(k\) by normalizing them with the brute-force solution obtained by the linear program in [8]. To solve it, we used the SCIP [2] solver within OR-tools [20]. We limited the max_iter parameter of SPSA toward 200; for COBYLA, we have not changed any hyperparameters.

We found that overall for both the optimizers, linear interpolating the problem shows an advantage over the nearest neighbor interpolation. Interestingly, by keeping the number of variables constant, we observe a decline in the optimizer’s performance, allowing the nearest neighbor interpolation to mostly catch up. This is attributed to the fact that plateaus and, thus, information become more densely encoded. As a result, interpolating the intermediate states becomes less critical. Nevertheless, it can be seen from Table 5 that linearly interpolating the problem leads to an advantage in our experiments.

V. CONCLUSION

Overall, in this work, we provide an improved version \(R'_f\) of the previously proposed function \(R_f\) [24]. We show that \(R'_f\) is numerically stable and robust against larger inputs. In addition, it naturally removes spikes at the plateau transitions, which occur in [24], particularly when hyperparameters are incorrectly set.

Our proposed optimization method AltOpt proves comparable to the Tabu optimizer in our experiments and outperforms state-of-the-art variational optimizers. Nevertheless, the comparability is not granted, since AltOpt most probably converges to a local optima, since only strictly decreasing energies are allowed. An interesting future work aspect in this context would be to adjust other metaheuristics, such as a Tabu search or simulated annealing, to provide an escape from the respective local minima by temporary allowing non-decreasing energy values.

Finally, it is worth mentioning that we found a method to embed a linear interpolation of the ansatz of [24] on a quantum device. Our tests on MaxCut problems with Erdős–Rényi Graphs show improved convergence for gradient-based optimizers. Future studies could explore the impact of noise on linear interpolation in current NISQ hardware and compare this approach to others like in [27].

ACKNOWLEDGMENT

AUTHORS’ CONTRIBUTION

David Winderl and Nicola Franco wrote the main manuscript. Jeanette Miriam Lorenz provides useful advice and comments.

AVAILABILITY OF DATA AND MATERIALS

The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

REFERENCES

[1] F. Barahona, M. Jünger, and G. Reinelt, “Experiments in quadratic 0–1 programming,” Math. Program., vol. 44, pp. 127–137, 1989, doi: 10.1007/BF01587084.
[2] K. Bestuzheva et al., “The SCIP optimization suite 8.0,” Dec. 2021. [Online]. Available: http://www.optimization-online.org/DB_HTML/2021/12/8728.html
[3] C.J. Bezek and R. J. Hathaway, “Some notes on alternating optimization,” in Advances in Soft Computing - AFSS 2002, New York, NY, USA: Springer, 2002, pp. 288–300, doi: 10.1007/3-540-45631-7_39.
[1] L. Bittel and M. Kliesch, “Training variational quantum algorithms is NP-hard,” Phys. Rev. Res., vol. 12, 2021, Art. no. 120502, doi: 10.1103/PhysRevLett.127.120502.

[2] M. Cerezo et al., “Variational quantum algorithms,” Nature Rev. Phys., vol. 30, no. 9, pp. 625–644, 2021, doi: 10.1038/s42254-021-00348-9.

[3] Y. Chatterjee, E. Bourreau, and M. J. Raničić, “Solving various NP-hard problems using exponentially fewer qubits on a quantum computer,” 2023, arXiv:2301.06978, doi: 10.48550/arXiv.2301.06978.

[4] C. W. Commander, “Maximum cut problem, MAX-CUT,” in Encyclopedia of Optimization. Boston, MA, USA: Springer, 2008, pp. 1991–1999, doi: 10.1007/978-0-387-74759-0_358.

[5] M. Cerezo et al., “Variational quantum algorithms,” 2021, vol. 30, no. 9, pp. 625–644, 2021, doi: 10.1038/s42254-021-00348-9.

[6] Y. Chatterjee, E. Bourreau, and M. J. Raničić, “Solving various NP-hard problems using exponentially fewer qubits on a quantum computer,” 2023, arXiv:2301.06978, doi: 10.48550/arXiv.2301.06978.

[7] W. Fernandez de la Vega and C. Kenyon-Mathieu, “Linear programming relaxations of maxcut,” in Proc. 18th Annu. ACM-SIAM Symp. Discrete Algorithms, 2007, pp. 53–61, doi: 10.5555/1283383.1283390.

[8] P. Erdos and A. Rényi, “On random graphs. I,” Publ. Math. Debrecen, vol. 6, pp. 290–297, 1959, doi: 10.5486/PMD.1959.6.3-4.12.

[9] J. Li, M. Alam, and S. Ghosh, “Large-scale quantum approximate optimization via divide-and-conquer,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 42, no. 6, pp. 1852–1860, Jun. 2023, doi: 10.1109/TCAD.2022.3212196.

[10] L. Bittel and M. Kliesch, “Training variational quantum algorithms is NP-hard,” Phys. Rev. Res., vol. 12, 2021, Art. no. 120502, doi: 10.1103/PhysRevLett.127.120502.

[11] M. Cerezo et al., “Variational quantum algorithms,” Nature Rev. Phys., vol. 30, no. 9, pp. 625–644, 2021, doi: 10.1038/s42254-021-00348-9.

[12] F. Glover, G. Kochenberger, R. Hennig, and Y. Du, “Quantum bridge an-