Quantum Approximate Multi-Objective Optimization

Ayse Kotil,[1, 2] Elijah Pelofske,[3] Stephanie Riedmüller,[2] Daniel J. Egger,[1]
Stephan Eidenbenz,[3] Thorsten Koch,[2, 4] and Stefan Woerner[1, *]

[1]*IBM Quantum, IBM Research Europe - Zurich*
[2]*Zuse Institute Berlin*
[3]*Los Alamos National Laboratory*
[4]*Technische Universität Berlin*
(Dated: April 1, 2025)

The goal of multi-objective optimization is to understand optimal trade-offs between competing objective functions by finding the Pareto front, i.e., the set of all Pareto optimal solutions, where no objective can be improved without degrading another one. Multi-objective optimization can be challenging classically, even if the corresponding single-objective optimization problems are efficiently solvable. Thus, multi-objective optimization represents a compelling problem class to analyze with quantum computers. In this work, we use low-depth Quantum Approximate Optimization Algorithm to approximate the optimal Pareto front of certain multi-objective weighted maximum cut problems. We demonstrate its performance on an IBM Quantum computer, as well as with Matrix Product State numerical simulation, and show its potential to outperform classical approaches.

I. INTRODUCTION

Quantum computing is a new computational paradigm that has the potential to disrupt certain disciplines, with (combinatorial) optimization frequently being mentioned as one of them [1]. However, in many cases, classical approaches can find good solutions quickly, leaving little room for further improvements. Thus, it is important to identify the right problem classes and instances that are truly difficult classically with a potential for improvement through quantum computers.

The goal of multi-objective optimization (MOO) is to find optimal trade-offs between multiple objective functions by identifying all Pareto optimal solutions, i.e., those solutions where no single objective value can be improved without degrading another one [2]. MOO can be difficult even if corresponding single-objective problems can be solved efficiently [2, 3], particularly with an increasing number of objective functions [4]. In this context, sampling-based approximate quantum optimization algorithms can be beneficial as they can quickly produce a large variety of good solutions to approximate the Pareto front.

We demonstrate how the quantum approximate optimization algorithm (QAOA) can be efficiently applied to multi-objective combinatorial optimization by leveraging transfer of QAOA parameters across problems of increasing sizes. This eliminates the need to train QAOA parameters on a quantum computer and removes a computational bottleneck for the considered problems. Using an IBM Quantum computer, we present promising results demonstrating that our algorithm has the potential to outperform classical approaches for multi-objective weighted maximum cut (MO-MAXCUT). Additionally, we show how today's quantum computers can help forecast the

performance of heuristics on future devices, serving as tools for algorithmic discovery.

The remainder of the paper is structured as follows. Sec. II introduces MOO. Sec. III discusses classical approaches to approximate the Pareto front. Then, Sec. IV defines the QAOA [5]–a core building block of our quantum algorithm–and the required theory. Afterwards, Sec. V discusses the related work on quantum algorithms for MOO and introduces our approach. Demonstrations of how the algorithm performs and how it compares to a classical benchmark are presented in Sec. VI. Sec. VII summarizes our work and discusses possible future research.

II. MULTI-OBJECTIVE OPTIMIZATION

The goal of MOO is to maximize a set of $m \in \mathbb{N}$ objective functions $(f_1, \ldots, f_m)$, with $f_i : X \to \mathbb{R}$, $i \in \{1, \ldots, m\}$, over a feasible domain $X \subset \mathbb{R}^n$, $n \in \mathbb{N}$ [2]:

$$\max_{x \in X} \quad (f_1(x), \ldots, f_m(x)). \tag{1}$$

Since it is usually not possible to optimize all f_i simultaneously, MOO refers to identifying the optimal trade-offs between possibly contradicting objectives.

Optimal trade-offs between objectives are defined via the concept of Pareto optimality. A solution $x \in X$ is said to dominate another solution $y \in X$ if $f_i(x) \geq f_i(y)$ for all $i \in \{1, \ldots, m\}$ and if there exists at least one $j \in \{1, \ldots, m\}$ with $f_j(x) > f_j(y)$. A solution x is called Pareto optimal if there does not exist any y that dominates x. In other words, x is Pareto optimal if we cannot increase any f_i without decreasing at least one f_j, $j \neq i$. The set of all Pareto optimal solutions is called the Pareto front or efficient frontier. Solving MOO exactly denotes computing the full Pareto front.

Suppose X is a convex set and all f_i are concave functions. Then, the Pareto front usually consists of infinitely

many points that lie on the surface of a convex set [2]. However, for non-convex problems, e.g., in the case of discrete optimization with $X \subset \mathbb{Z}^n$, this is not true anymore, and the Pareto front may consists of a finite discrete set of points [2]. We call Pareto optimal solutions supported if they lie on the convex hull of the Pareto front, and non-supported otherwise. Especially discrete MOO problems are often not efficiently solvable even when the corresponding single-objective optimization problems are in **P**, i.e., if they can be solved efficiently. This is the case, if the Pareto front grows exponentially with the input size or if the computation of non-supported solutions is demanding [2, 3]. Throughout this manuscript, we usually assume $X = \{0, 1\}^n$ and $f_i(x) = x^T Q_i x$ for matrices $Q_i \in \mathbb{R}^{n \times n}$, i.e., we consider the MOO variant of quadratic unconstrained binary optimization (QUBO), although many of the introduced concepts are more general.

Since the Pareto front and its approximations consist of a set of points, we cannot use the objective values directly to compare the quality of different given approximations. There exist many possible performance metrics for MOO with the hypervolume (HV) being the one most commonly used. A survey of performance metrics for MOO is given in [6]. The HV of a given set $\{(f_1(x), \ldots, f_m(x)) \mid x \in S\}$, where $S \subset X$ is a set of (unique) solutions, measures the volume spanned by the union of all hyperrectangles defined by a reference point $r \in \mathbb{R}^m$ and the points $(f_1(x), \ldots f_m(x))$, where r is assumed to be a component-wise lower bound for all (given) vectors of objective values. If S equals the Pareto front, the HV is maximized. The reference point r can be chosen, e.g., by minimizing all objective functions individually. Then r satisfies $r_i \leq f_i(x)$ for all $i \in \{1, \ldots, m\}$ and $x \in X$. Evaluating the HV exactly is in general #**P**-hard in the number of dimensions. However, efficient approximation schemes exist, e.g., leveraging Monte Carlo sampling [7–9]. Fig. 1 illustrates a non-convex Pareto front, its convex hull, (non-)dominated and (non-)supported solutions, as well as the optimal HV and the HV of a given set of solutions approximating the Pareto front.

III. CLASSICAL ALGORITHMS FOR MOO

There are different classical strategies to approximate the Pareto front. The two most common approaches are the *weighted sum method* (WSM) and the *ϵ-constraint method* (ϵ-CM) [2]. Both convert MOO into many single-objective optimization problems that are solved with conventional (combinatorial) optimization solvers.

The WSM constructs a single objective function

$$f_c(x) = \sum_{i=1}^m c_i f_i(x), \qquad (2)$$

where $c \in [0, 1]^m$ with $\sum_i c_i = 1$ denotes a vector of weights of the convex combination of objectives. Solving $\max_{x \in X} f_c(x)$ for an appropriate discretization of c vectors leads to solutions x_c^* with corresponding objective
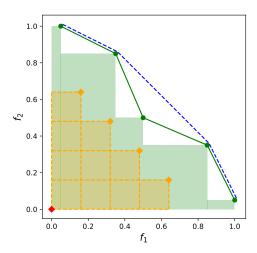

FIG. 1. Pareto front & HV for two objective functions: The red diamond represents the reference point r. The green points on the green solid line correspond to the optimal Pareto front, the green shaded area represents the optimal HV with respect to r, and the blue dashed line corresponds to the convex hull of the Pareto front. The green point in the middle of the Pareto front does not lie on the convex hull and illustrates a non-supported Pareto optimal solution, the four other green points lie on the convex hull and illustrate supported Pareto optimal solutions. The orange diamonds represent examples of dominated solutions approximating the Pareto front, with the orange area illustrating their corresponding HV.

vectors $(f_1(x_c^*), \ldots, f_m(x_c^*))$ and allows to approximate the Pareto front. Assuming we can solve these problems to global optimality, e.g., by using commercial solvers like CPLEX [10] or GUROBI [11], all resulting points lie on the Pareto front. However, the WSM can only find non-supported solutions, i.e., solutions whose corresponding vectors of objective values lie on the convex hull of the Pareto front [2]. As illustrated in Fig. 1 and shown in Appendix B, there can be a significant gap between the HV achieved by supported solutions and the optimal HV of all solutions, including non-supported ones. In case of QUBO objective functions, the WSM results in a new QUBO for each c and we denote the corresponding cost matrix by $Q_c = \sum_i c_i Q_i$.

The standard ϵ-CM constructs single objective optimization problems by maximizing only one $f_i(x)$ at a time subject to constraints $f_j(x) \geq \epsilon_j$, $j \neq i$, on the other objective functions:

$$\max_{x \in X} \quad f_i(x) \qquad (3)$$

$$\text{subject to:} \quad f_j(x) \geq \epsilon_j, \, \forall j \neq i. \qquad (4)$$

By solving such problems for all objective functions with appropriate discretizations of the ϵ_j, this method allows to approximate the whole Pareto front including non-supported solutions. Thus, if the discretization of the ϵ_j is sufficiently fine, the method can compute the exact Pareto front. While the ϵ-CM is more powerful than

the WSM, adding additional constraints can make the problem more difficult to solve. For instance, adding constraints maps QUBO to quadratic *constrained* binary optimization. Fig. 2 illustrates the ϵ-CM. In the presented form, the ϵ-CM may also generate dominated solutions. However, more advanced variants can overcome this inefficiency. For instance, one can combine the ideas of the WSM and the ϵ-CM in a hybrid method [2].
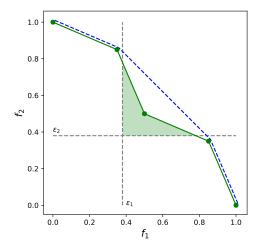

FIG. 2. The ϵ-CM: Given the lower bounds ϵ_i on the objective function values, the resulting feasible set (green area) only contains the non-supported Pareto optimal solution, which – in this simplified example – will be found by maximizing any of the two objective functions or convex combinations of them.

For both methods, the number of c, respectively, ϵ discretization points to find good approximations of the Pareto front, in general, scales exponentially with the number of objective functions considered. Thus, they are only practical for a very small number of objective functions. Therefore, we consider randomized variants of both that can also approximate the Pareto front for larger numbers of objective functions, as discussed in the following.

Instead of systematically discretizing the c vector in the WSM, we propose to sample it randomly and solving the resulting problems with an appropriate solver. We sample them uniformly at random from the feasible set, cf. Appendix A for more details. This sampling-based approach allows us to approximate the efficient frontier without having to enumerate an exponential number of grid points. As before, every sampled c leads to a supported solution x_c^* that lies on the Pareto front. Alternative variants of the WSM, e.g., using the Goemans-Williamson algorithm [12] to solve the resulting problems are possible, cf. Appendix B.

Similarly, we propose a randomized variant of the ϵ-CM. First, we minimize and maximize the individual objective functions to get lower and upper bounds $l_i \leq f_i(x) \leq u_i$, for all $x \in X$, where the lower bounds coincide with the reference point r that we use for the HV, cf. Sec. II. Then, we sample $(\epsilon_1, \ldots, \epsilon_m)$ uniformly at random from

$\bigotimes_{i=1}^m [l_i, u_i]$. Instead of optimizing a particular objective f_i, we take a sample c like in the randomized WSM and solve the problem

$$\max_{x \in X} \quad \sum_{i=1}^m c_i f_i(x) \tag{5}$$

$$\text{subject to:} \quad f_j(x) \geq \epsilon_j, \ \forall j. \tag{6}$$

This method is a bit more complex than the randomized WSM due to the additional constraints, but it has two important advantages: First, the randomized ϵ-CM is guaranteed to approximate the optimal Pareto front in the limit of infinitely many samples. Second, the ratio of feasible ϵ samples, i.e., samples where the resulting constrained optimization problem has a feasible solution, multiplied by the volume of the space of possible values $\prod_{i=1}^m (u_i - l_i)$ is a Monte Carlo estimator for the optimal HV. Both statements assume that the resulting mixed integer programs (MIPs) are solved to global optimality or that infeasibility can be proven if no feasible solution exists.

In recent years, numerous more involved algorithms have been proposed for specifically solving multi-objective integer problems for an arbitrary number of objectives. Many objective-space algorithms enumerate all solutions by decomposing the search space and applying a scalarization to the resulting subproblems [13–15] or by recursive reduction [16]. In contrast, there are also decision-space algorithms, for example, following a branch-and-bound approach [17].

Definitions of multi-objective integer programming vary across papers regarding the nature of the objective coefficients. While some algorithms are only proven to be correct for integer objective coefficients - such as DPA-a [15] and AIRA [16] - others, in theory, can also handle continuous objective coefficients, including Epsilon [13], DCM [14], and Multi-Objective Branch-and-Bound [17]. However, publicly available implementations of these algorithms are generally restricted to integer objective coefficients. Applying those algorithms to problems with continuous objective coefficients by scaling the objectives is possible, but not in all cases efficient due to a significant increase in the size of the objective space, which corresponds to the precision of the coefficients, leading to longer runtimes. As a representative for those multi-objective integer algorithms, we consider DPA-a and DCM. See Appendix B for further details.

In addition to the MIP-based algorithms mentioned above, there are also several genetic algorithms that have been proposed for MOO, see eg. [18]. Getting these algorithms to work is known to require a lot of fine tuning of all steps involved. A first test of available implementations [19] did not lead to competitive results and we leave it to future work to further study these approaches.

IV. QUANTUM APPROXIMATE OPTIMIZATION ALGORITHM

The QAOA [5, 20] considers a QUBO problem on n variables with cost matrix Q and maps it to the problem of finding the ground state of a corresponding diagonal Ising Hamiltonian H_C operating on n qubits, which, for example, can take the form

$$H_C = -\sum_{i<j} J_{ij}\sigma_i^z\sigma_j^z - \sum_i h_i\sigma_i^z, \tag{7}$$

where σ_i^z is the Pauli-Z operator acting on the i-th qubit, and h_i and J_{ij} are the problem coefficients. The encoding is achieved by representing the binary variables $x \in \{0,1\}^n$ by spin variables $z \in \{-1,1\}^n$, via the linear transformation $x_i = \frac{1-z_i}{2}$, and replacing the variables z_i by σ_i^z [21]. This mapping leads to $\langle x|H_C|x\rangle = -x^TQx$ (up to a constant energy offset), where $|x\rangle$ is the quantum state encoding the bitstring $x \in \{0,1\}^n$. Thus, finding the ground state $\min_{|x\rangle}\langle x|H_C|x\rangle$ corresponds to maximizing the objective function x^TQx. This Hamiltonian H_C that encodes the cost of the objective function is typically known as the *phase separator* Hamiltonian.

To solve the problem, QAOA first prepares the ground state of a *mixer* Hamiltonian, often conveniently chosen to be $H_X = -\sum_i \sigma_i^x$ with its ground state $|+\rangle = \frac{1}{\sqrt{2^n}}\sum_x |x\rangle$, where σ_i^x is the Pauli-X operator acting on the i-th qubit. Then, QAOA applies alternating simulation of the two non-commuting Hamiltonians H_X, H_C, resulting in the state

$$|\psi(\beta,\gamma)\rangle = \prod_{k=1}^p e^{-i\beta_k H_X}e^{-i\gamma_k H_C}|+\rangle, \tag{8}$$

with parameters $\beta,\gamma \in \mathbb{R}^p$, and number of repetitions $p \in \mathbb{N}$. As p approaches infinity, QAOA is guaranteed to converge to the ground state of the problem Hamiltonian [5], assuming the parameters β and γ are set appropriately. For finite p, carefully setting the parameters β and γ is crucial for the performance of the algorithm. The standard approach for selecting QAOA parameters uses a classical optimizer to minimize $\langle \psi(\beta,\gamma)|H_C|\psi(\beta,\gamma)\rangle$ with respect to (β,γ) in an iterative manner, where the objective function is evaluated using the quantum computer. Once good parameters have been found, sampling from $|\psi(\beta,\gamma)\rangle$ results in good solutions to the original QUBO. There exist several variants of QAOA, such as recursive QAOA [22] or warm-started QAOA [23, 24], which can help to further improve the performance. However, within this paper we focus on the original proposal.

The highly non-convex parameter landscape, sampling errors, and hardware noise make it challenging to find good parameters [25]. However, there exist multiple alternative strategies to find good parameters for QAOA and circumvent the trainability challenge. For example, as proposed in [26] the expectation values for training QAOA parameters can be evaluated classically. Then, the quantum computer is only used to generate the samples. This can be of potential interest since, for QUBO, H_C consists of only 1- and 2-local terms, which can be significantly cheaper to evaluate classically than sampling from the full quantum state. Alternatively, good angles can be derived for certain problem classes from an infinite problem size limit [27–31], by applying linear parameter schedules [32], or by using machine learning models [33, 34]. Another promising strategy is to reuse optimized parameters from similar problem instances [35–37]. This is often referred to as *parameter transfer* or *parameter concentration* and is discussed in more detail in the following.

Parameter transfer in QAOA is a method to reuse the parameters obtained for one problem instance for another one without the need for repeating the classical parameter optimization. Optimized QAOA parameters for one problem instance often result in high-quality solutions for another instance, if both problem instances exhibit similar properties. This has been proven analytically on low-depth MAXCUT problems on large 3-regular graphs [38] and observed empirically on QAOA with $p = 1$ for MAXCUT problems on random regular graphs with same parity [39]. Strategies to rescale the unweighted MAX-CUT parameters for weighted problems exist to enable parameter transfer [40, 41]. Several studies show parameter concentration of similar problem instances with respect to problem size on random Hamiltonians empirically and analytically [26, 37, 42–44]. In general QAOA parameter transfer is a good heuristic tool to efficiently find QAOA angles which perform reasonably well, and importantly can circumvent computationally costly variational learning. Importantly, parameter transfer does not yield optimal QAOA angles in general for all instances, but instead can work well on average for ensembles of similarly structured Hamiltonians. Parameter transfer is also possible from smaller to larger problem instances, and different training strategies can be combined, e.g., we can use some of the established schemes to initialize the parameters for a smaller problem, then train it using classical evaluation of the expectation values, and transfer the resulting parameters to larger problem instances with similar structure. This is the direction we will be applying in the following. A broad study on parameter transfer for different problem classes has been conducted in [45]. In the following, we show how to leverage QAOA and parameter transfer in the context of weighted MAXCUT MOO on specific quantum computer hardware graphs.

V. QAOA FOR MULTI-OBJECTIVE OPTIMIZATION

There exists only very little literature on quantum algorithms for MOO. Ref. [46] proposes to use QAOA as a solver in the WSM, similar to what we will propose here. Within the WSM, a weight vector c maps multiple QUBO cost matrices Q_i to a new combined QUBO cost matrix Q_c. Thus, one can run QAOA for every c vector independently. In [46], the authors propose to re-optimize the QAOA

parameters (β, γ) for every c, which is computationally expensive and leads to a bottleneck. Simlarly, Ref. [47] suggests to re-optimize the QAOA parameters. However, the authors suggest to warm-start every optimization with the parameters found in the previous iteration and motivate it by QAOA parameter transfer. In addition, they propose a Tchebycheff scalarization strategy to navigate the Pareto front. In contrast, [48] propose to directly use the HV as a single objective function in a variational quantum algorithm. Further, they propose an extended ansatz that includes QAOA cost operators for every objective function in the considered MOO problem. This can be potentially problematic, as one tries to generate a state that corresponds to a superposition of Pareto optimal solutions and thus can become more difficult to train in cases where the Pareto front consists of many points. Further, they extend the QAOA ansatz by including a cost operator for every objective function, which is not covered by any of the existing theory on the performance of QAOA and leads to deeper circuits. Lastly, [49] introduce an algorithm based on Grover's search [50] with a quadratic speed-up over brute force search leading to large quantum circuits that require error correction.

We propose to go one step further than Refs. [46, 47]and completely skip the optimization of the QAOA parameters for the considered problem instance. More precisely, in order to overcome the aforementioned computational bottleneck introduced by re-optimizing for every c vector, we leverage the transfer of parameters discussed in Sec. IV from a single (smaller) problem instance to the (larger) ones resulting for every sampled c vector. Thus, we only have to train the QAOA parameters once, which can be done offline, i.e., before the actual problem instance of interest is known. For this to work, we have to identify a representative single-objective problem and find good QAOA parameters for it. For the training, we use problems small enough to be simulated classically, i.e., there is no quantum computer needed for the parameter training. We then fix these QAOA parameters and only vary the cost Hamiltonian through sampling c vectors. The rationale is that QAOA with fixed parameters gives a diverse set of good solutions for all c vectors, although not necessarily the optimal solutions. However, a sub-optimal solution for a particular c vector might still be a Pareto optimal solution, and since it does not need to be optimal for the considered weighted sum, it might also result in non-supported Pareto optimal solutions.

As mentioned in Sec. IV, transfer of parameters can apply between similar instances of the same size, but also from smaller to larger instances. Here, we propose to train parameters classically on sufficiently small but representative instances and then re-use them for larger instances on a quantum computer. Since the parameters are considered independent of the concrete problem instance and can be determined offline, the training does not contribute to the runtime of an algorithm for a particular problem instance and we only consider the actual sampling from the QAOA circuits.

This approach of using QAOA to sample sub-optimal solutions to the underlying diagonal Hamiltonian cost function with the goal of sampling the Pareto optimal front is notable for several reasons: First, this means that we need both high-diversity solution sampling of configurations that share similar cost values–i.e. fair sampling [51, 52]–and high-variance sampling which specifically means we do not use QAOA circuits converged to the ground-state (i.e., approximation ratio of 1, which again would rule out non-supported solutions). Second, which follows from the first reason, we utilize low depth QAOA circuits, which makes the computation especially suitable for noisy quantum processors. Last, there are several no-go theorems on QAOA, usually providing lower bounds on the number of QAOA repetitions p with respect to the problem size to achieve on average a certain solution quality [22, 53]. However, for the considered algorithm, none of the known no-go theorems apply, since a solution that might be bad for a particular c vector may still be (non-supported) Pareto optimal.

In the following, we demonstrate the potential of our proposal on a concrete problem and show that QAOA with transfer of parameters represents a promising approach for MOO.

VI. EXPERIMENTAL RESULTS

We demonstrate our proposal on instances of MO-MAXCUT. The goal of single-objective weighted MAX-CUT is to partition the set of nodes $\mathcal{V}$ of a given weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$, with edge weights w_{kl} for edges $(k, l) \in \mathcal{E}$, into two sets, maximizing the sum of the edge weights of the edges connecting the two sets. Weighted MAXCUT can be formulated as a QUBO:

$$\max_{x \in \{0,1\}^n} \sum_{(k,l) \in \mathcal{E}} w_{kl}(x_k + x_l - 2x_k x_l). \tag{9}$$

To introduce multiple objective functions, we assume multiple weighted graphs $\mathcal{G}_i = (\mathcal{V}, \mathcal{E}, w^i)$, $i = 1, \ldots, m$, defined on the same set of nodes $\mathcal{V}$, and here, for simplicity, also on the same set of edges $\mathcal{E}$, i.e., they only differ in their edge weights w^i. Then, each graph defines a MAXCUT objective function over the same decision variables.

We consider subgraphs of the heavy-hex topology [54] of IBM Quantum devices [55] to facilitate execution on real quantum hardware, see Fig. 3. Defining the problem instance using the native two-qubit gate connectivity graph of the hardware allows to construct relatively short-depth QAOA circuits that can be executed with high fidelity [43, 44, 56–58]. More precisely, we consider 42-node graphs to define the MO-MAXCUT problem with $m = 3$ and $m = 4$ objective functions. To implement the corresponding circuits on hardware, we need a parametrized RZZ-gate for each edge in the graph for the implementation of the phase separator cf. Sec. IV. These gates are then compiled to the hardware native basis gate set $\{X,$

$\sqrt{X}$, RZ, CZ}, i.e., a RZZ-gate consists of two CZ-gates plus single qubit gates. The minimum edge-coloring of the sparse heavy-hex lattice requires three colors, thus, we can apply all RZZ-gates in three non-overlapping layers. This determines the RZZ-depth as three for each QAOA round and the corresponding CZ-depth as six, assuming RZZ-gates are implemented using two CZ-gates. Since, on the considered device sub-graph that we compile the QAOA circuits to, all RZZ-gates have a uniform duration of 84ns, the choice of edge coloring does not affect the total compiled circuit duration and therefore a random 3-edge-coloring is used. The mixing Hamiltonian is implemented using single qubit RX-gates per layer compiled to the basis gates.

For each graph, the edge weights are independently sampled from a standard normal distribution, i.e., $w_{kl}^i \sim \mathcal{N}(0,1)$, for $(k,l) \in \mathcal{E}$, $i = 1, \dots, m$. Note that considering continuous weights make these problems particularly challenging for classical algorithms, cf. Sec. III and Appendix B.

Further, we consider a 27-node graph for the classical QAOA parameter optimization with weights sampled independently from $\mathcal{N}(0, 1/m)$ for the different number of objective functions. The standard deviation $1/m$ accounts for the randomized convex combinations of the m objective functions used in our algorithm. For MO-MAXCUT, this corresponds to convex combinations of the edge weights w^i, leading to a lower standard deviation compared to edge weights drawn form the standard normal distribution. The standard deviation of $1/m$ is equivalent to taking the average of m random weights drawn independently from a standard normal distribution. The corresponding (unweighted) 27-node and 42-node graphs are shown in Fig. 3. The structure of the 27-node graph was inspired by the topology of the 27-qubit IBM Quantum Falcon devices. However, in the present context we simulate the corresponding circuit classically for initial parameter training and only execute the 42-qubit circuits on a quantum computer for generating the results of the problem of interest.

To optimize the QAOA parameters for the 27-node single-objective MAXCUT problem, we use `JuliQAOA` [59], which is a Julia-based [60] QAOA simulator. It efficiently evaluates expectation values and corresponding gradients based on the QAOA-statevector and applies a basin-hopping optimization algorithm to navigate the many local minima in the parameter landscape. More details on the optimization pipeline are provided in Appendix C. We empirically confirm that transfer of parameters works for the considered class of problems, as illustrated in Appendix D. This allows us to pre-optimize the QAOA angles for $p = 1, \dots, 6$, independently of the concrete 42-node MO-MAXCUT problem instance. In the following we discuss the results for the two considered problem instances.
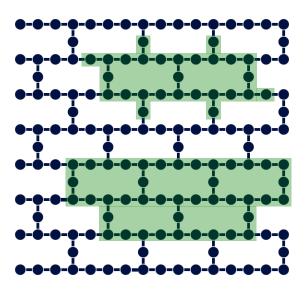

FIG. 3. Graph topologies: The whole graph shows the coupling map of the 156-qubit *ibm_fez* device. The green-shaded areas show the 27-qubit graph of the training problem (top) and the 42-qubit graph of the considered MO-MAXCUT problem (bottom) embedded in the *ibm_fez* coupling map.

A. MO-MAXCUT with 3 objective functions

To asses and compare the performance, we evaluate the progress of the achieved hypervolume HV_t over time t. We always report single-thread CPU and QPU times. First, we tested the classical algorithms discussed in Sec. III on the 42-node MO-MAXCUT problem with $m = 3$ objective functions. DCM, DPA-a, and the ϵ-CM performed best and are discussed in more detail here. For the results of the other algorithms see Appendix B.

For DCM and DPA-a, we scaled the continuous weights by a factor of 1000 and rounded them to integers, since the available implementations of the algorithms cannot handle continuous weights natively. While we set a time limit of 10,000s for both algorithms, DPA-a terminated already after 3.6min and DCM after 8min. Thus, both find the global optimal hypervolume for the truncated weights. The HV achieved by the resulting non-dominated points evaluated for the exact non-truncated weights is $HV_{max} = 43,471.704$. Thus, this is the optimal HV up to the error introduced by the truncation of weights. Both algorithms found 2063 non-dominated solutions.

Note that particularly DCM was very sensitive to the scaling factor. When the weights were scaled by a factor of 10,000 before rounding, the algorithm only found as little as four non-dominated points before reaching the given time limit. This shows that continuous weights can pose a particular challenge for classical algorithms. DPA-a was more robust to the scaling factor for the considered problem instance and achieved a comparable performance in both cases, cf. Sec. B.

The HV-progress over time for all algorithms discussed in this section is shown in Fig. 4. In all figures showing

the HV progress, we plot $(\mathrm{HV_{max}} - \mathrm{HV}_t + 1)$ in a log-log plot with a reversed y-axis to increase the contrast in the visualization while preserving the qualitative presentation of the results. The number of non-dominated points for all tested algorithms is shown in Fig. 5.

Next, we executed ϵ-CM, again setting a time limit of 10,000s. In our tests, every resulting MIP has been solved to optimality using GUROBI on average in 0.0218s using a single-thread on *AMD EPYC 7773X* CPUs. Thus, we sampled almost 460,000 random ϵ and c vectors and solved the corresponding MIPs. We based our implementation on the standard integer programming formulation of MAX-CUT. Since the randomized ϵ-CM adds constraints to the different objectives this is more suitable for MIP solvers than the QUBO formulation, which would result in non-convex quadratic constraints. See Appendix E for more details. The maximum HV achieved by the ϵ-CM was 43,471.222, i.e., slightly below $\mathrm{HV_{max}}$, where the Pareto front was approximated by 2054 non-dominated points.

As discussed in Sec. III, the ratio of feasible ϵ samples allows us to derive a Monte Carlo estimate of the optimal HV. From the close to 460,000 samples, 84.917% were feasible. The lower and upper bounds for the 3 objective functions are given by $l = (-12.137, -19.642, -18.331)$ and $u = (21.390, 19.130, 21.068)$, respectively. The volume of the resulting hyper rectangle is $V_{l,u} = 51,212.547$. Thus, the Monte Carlo estimate of the globally optimal HV is given by 43,488.364 with 95%-confidence interval equal to [43,435.378, 43,541.351], which contains and closely agrees with $\mathrm{HV_{max}}$.

As a first test of the QAOA-based algorithm, we use Qiskit Aer [61, 62] to run Matrix Product State (MPS) simulations of the QAOA circuits with bond dimensions $\chi = 1, 5, 10, 20, 30, 50$ and $p = 1, \ldots, 6$. For each setting we draw 5000 random c vectors as described in Appendix A. Given the precomputed QAOA parameters from the 27-qubit problem instance, every sampled c vector fully defines a corresponding QAOA circuit and we take 5000 shots from each of them. In total, this results in 25 million shots.

We are interested in the performance when running the circuits on a real quantum computer. Thus, we assume a sampling rate of 10,000 shots per second, which is realistic as discussed later in the context of our hardware experiments, although the actual MPS simulations may take longer, cf. Appendix F. Thus, 25 million shots are assumed to take in total 2500s. Then, we compute the corresponding hypervolume HV_t at time t. With increasing χ and p, the simulations reach $\mathrm{HV_{max}}$. Each setting is simulated five times to analyze the robustness of the algorithm and we determine the resulting minimum, maximum, and average performance. The results are shown in Fig. 6. A bond dimension $\chi = 20$ seems sufficient to evaluate the considered circuits, as there are only small difference between the results for $\chi = 20, 30$ and 50. This is further supported by the analysis presented in Appendix F. The resulting number of non-dominated points approximating the Pareto front is shown in Fig. 5. It

p	#CZ	depth (CZ/total)	duration [s]	fidelity [%]
3	276	18 / 49	3.9×10^{-6}	13.63
4	368	24 / 65	4.7×10^{-6}	8.84
5	460	30 / 81	5.5×10^{-6}	5.42
6	552	36 / 97	6.3×10^{-6}	3.71

TABLE I. Summary of executed QAOA circuits: QAOA repetitions p, number of CZ gates, corresponding CZ depth, total gate depth, schedule duration, and estimated fidelity based on device calibration data. The duration of a CZ gate on *ibm_fez* uniformly is 84ns, which makes up for the majority of the circuit duration. The remainder is coming from single qubit gates and final measurements.

achieves a maximum of 2772. The difference to the number of points found by DCM and DPA-a can be explained by the truncation of weights for the classical algorithms. However, this also implies that the additional points have only a negligible contribution to the HV.

Finally, we run the quantum algorithm for $p = 3, 4, 5, 6$ on the *ibm_fez* device accessed via the IBM Quantum platform [55], again using 5000 random c vectors and 5000 shots per circuit. There are twelve native options to place the considered circuits on the topology of *ibm_fez* without inserting any SWAP gates, one of them shown in Fig. 3. Before running the experiments, we estimate the resulting fidelity of running the circuit on the quantum computer using the error rates for gates and measurements reported by the backend. Then we select the initial layout, i.e., subset of qubits, that leads to be highest fidelity. The estimated fidelities are reported in Tab. I together with number of gates, depth, and duration of executing the circuits. As in the MPS simulation, each experiment is repeated five times. The repetition delay between measuring one quantum circuit and starting the subsequent one is set to 10^{-4}s. Since, in the considered case, executing a circuit is significantly shorter, see the durations reported in Tab. I, the repetition delay dominates the overall runtime on the quantum computer. This results in the aforementioned sampling rate of 10,000 shots per second. Further reducing the delay may speed up the calculation but could reduce the quality by affecting the qubit initialization between shots. Except for the adjusted repetition delay, we use the default execution pipeline provided by the IBM Quantum *SamplerV2* primitive accessible via the IBM Qiskit Runtime [55]. This sampling rate is an idealistic assumption, since it excludes, e.g., circuit compilation. However, we believe this is reasonable since all required compilation could be done ahead of time for a parametrized template that is independent of the concrete problem instance.

Fig. 4 shows the results for DPA-a, DCM, ϵ-CM, the MPS simulations with $\chi = 50$, and the results from *ibm_fez*. Assuming a 10kHz sampling rate, the MPS simulations with $p \geq 5$ outperformed all other methods and achieved $\mathrm{HV_{max}}$ at about the same time as DPA-a. All runs with $p \geq 3$ reached $\mathrm{HV_{max}}$ within the 2500s for all five repetitions. The corresponding number of
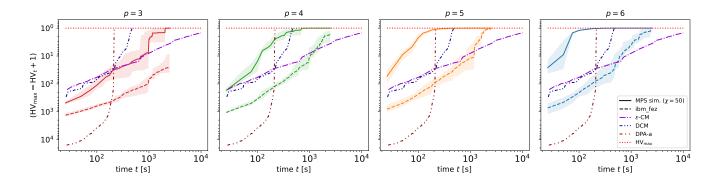
FIG. 4. Comparison of quantum and classical results for MO-MAXCUT with $m = 3$ objective functions: Average progress of HV_t of the MPS simulation ($\chi = 50$, solid line), *ibm_fez* (dashed line), and the classical algorithms DCM, DPA-a, and ϵ-CM (dotted-dashed lines). The shaded areas denote the minimum and maximum performance over the five repetitions of the MPS simulation and results from *ibm_fez*. The best found solution $\text{HV}_{\max}$ is indicated by the red dotted horizontal line.

FIG. 5. Number of non-dominated points: The solid lines show the average number of unique non-dominated points approximating the Pareto front resulting from the five repetitions using MPS simulation, with the shaded areas denoting the corresponding minimum and maximum. The largest value reached is 2772. For $\chi = 1$ or $p = 1$, the number of points was only between 170 and 255 and is not shown here. The dashed line and corresponding shaded area shows the results from the runs on *ibm_fez* reaching a maximum of 2726. For both, the MPS simulation and the results from *ibm_fez* 25 million shots were taken in total. The dotted-dashed lines show the result for DPA-a and DCM (both achieve the same value) and the ϵ-CM.

non-dominated points approximating the Pareto front are shown in Fig. 5. Notably, the samples generated by *ibm_fez* also improve with increasing p, i.e., with increasing circuit complexity, and for $p = 4, 5, 6$ overtake the ϵ-CM on average after 1230s, 1190s, and 535s, respectively. For $p = 4$, the average results from the *ibm_fez* after 2500s are comparable to the results of the ϵ-CM after 10,000s. For $p = 5, 6$, the average results even overtake the best results from the ϵ-CM and approach the best observed

value $\text{HV}_{\max}$. This is quite remarkable and underscores the potential of the proposed approach.

The presented results allow already to draw two key conclusions: First, they clearly show that performance improves with increasing bond dimension. While for $\chi = 1$, the circuit complexity p has no impact, the performance greatly improves as χ increases, i.e., as the MPS simulator approximates the circuits more accurately. Second, at larger bond dimensions, the performance consistently improves with increasing p, indicating that achieving good results requires executing sufficiently complex circuits with sufficiently high accuracy. This is a strong indication that with increasing scale, this algorithm requires execution on a real quantum computer, in contrast to an approximation using, e.g., an MPS simulator. These observations are consistent with the work of Ref. [63] that shows how genuine multi-partite entanglement is needed to obtain good results with quantum approximate optimization. This provides a first strong indication that MOO is a promising candidate for a potential quantum advantage in optimization. Last, for the problems considered here, with relatively small scale

The samples generated by the quantum computer are affected by noise, which explains the differences to the MPS simulations. The estimated fidelities in Tab. I, derived from the errors reported by the backend, provide an indication of the strength of the noise. As discussed in [64], the fidelity can be interpreted as the *probability of no error*. This implies that on average a fraction equal to the fidelity of the generated samples are drawn from the ideal noise-free distribution, while the remaining samples are potentially corrupted by noise. If, for simplicity, we assume that only samples corresponding to the noise-free distribution contribute to the HV and we discard the other ones, and if we scale the time it takes to generate the noisy samples from the quantum computer by the corresponding fidelity, the average HV_t corresponding to the noise-free MPS simulation should result in a lower bound of the HV_t corresponding to scaled hardware results. These scaled results, shown in Fig. 7, match the
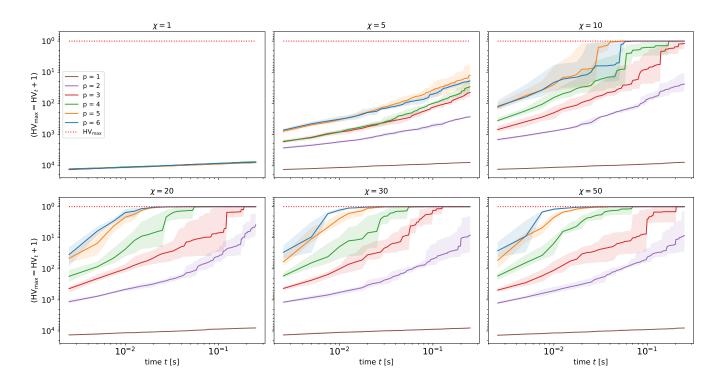
FIG. 6. MPS simulation results: Hypervolume HV_t for different bond dimensions χ and QAOA rounds (i.e., depth) p. Each line represents the average of five random repetitions of the same setting with shaded areas indicating the minimum and maximum values. The best found solution HV_{max} is indicated by the red dotted horizontal line.

MPS simulation results very well. This demonstrates how evaluating quantum heuristics on noisy devices can forecast their performance when being run on devices with lower error rates, including fault-tolerant devices, with a sampling overhead corresponding to the reciprocal of the fidelity. In the following section, we will use this insight in the opposite direction for the analysis of the instance with $m = 4$ objective functions.

B. MO-MAXCUT with $m = 4$ objective functions

Next, we analyze a MO-MAXCUT problem instance with $m = 4$ objective functions. The underlying unweighted graph is the same as in Sec. VIA and the weights for the four objective functions are again sampled from $\mathcal{N}(0, 1)$. As explained before, the weights for the 27-qubit training instance are sampled from $\mathcal{N}(0, 1/m)$ and the corresponding QAOA parameters are trained with JuliQAOA, cf. Sec. D for more details on the training and transfer of parameters.

As in Sec. VIA, we test DCM and DPA-a with weights scaled by a factor 1000 and then rounded to integers. Further, we run the ϵ-CM. For all three algorithms we set a time limit of 20,000s, which results in about 740,000 random c and ϵ vectors–i.e. MIPs–for the ϵ-CM. Since we have to add an additional constraint to the corresponding MIPs, the average time to solve one instance slightly increase to 0.0270s per MIP compared to 0.0218s
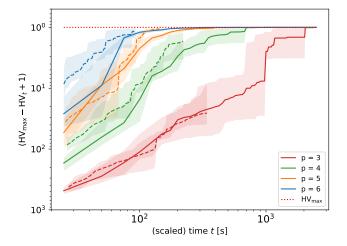

FIG. 7. Fidelity-scaled quantum results: MPS simulation and ibm_fez results from Fig. 4 where the time of the hardware results is scaled by the corresponding estimated fidelity reported in Tab. I.

for the case of three objective functions. Again, DPA-a performs best among the tested classical algorithms but takes significantly longer than for three objectives. It terminates just before reaching the time limit and finds the optimal Pareto front for the truncated weights, with $HV_{max} = 1,266,143.350$ when evaluated using the exact
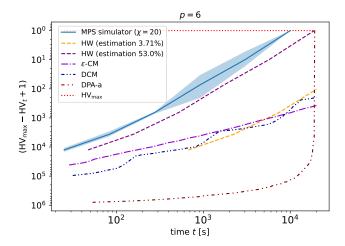
FIG. 8. Comparison of classical and estimated quantum results for MO-MAXCUT with $m = 4$ objective functions: Average progress of HV_t of the MPS simulation ($\chi = 20$, solid line), estimated hardware with 3.71% fidelity, like *ibm_fez* (dashed orange line), estimated hardware with 53% fidelity (dashed purple line), and the classical algorithms DCM, DPA-a, and ϵ-CM (dotted-dashed lines). The shaded area denotes the minimum and maximum performance over the five repetitions of the MPS simulation. The best found solution HV_{max} is indicated by the red dotted horizontal line.

weights.

In addition to the classical algorithms, we run QAOA with $p = 6$, 20,000 random c vectors, and 5000 shots per resulting circuit, i.e., we generate in total 100 million samples–four times more than for three objectives. Note that the circuit structure does not change compared to Sec. VI A. Assuming again 10,000 shots per second this results in a total runtime of 10,000s. We run five repetitions of the MPS simulation with $\chi = 20$, which has been identified as sufficient in Sec. VI A. The resulting HV_t for the different algorithms are shown in Fig. 8.

In addition to the algorithms mentioned above, we estimate the performance of running on *ibm_fez* by scaling the time of the MPS simulation by the inverse of the estimated fidelity of 3.71% (cf. Tab. I), i.e., we multiply it by a factor of 26.954 and truncate after 20,000s. Following the previous discussion on the fidelity in Sec. VI A, this provides a lower bound on the performance to be expected when running the circuits on *ibm_fez*. It can be seen that after approximately 5000s the predicted HV overtakes and then stays above the HV achieved by DCM and the ϵ-CM. It is also above the HV achieved by DPA-a until the last few minutes of the runtime, when DPA-a accelerates towards HV_{max}. Further, we find that with a hypothetical fidelity of 53%, QAOA would achieve HV_{max} in the same time as DPA-a, while dominating all algorithms throughout the whole runtime.

We can see that the problem gets significantly more difficult for the classical algorithms when increasing the number of objective functions. The MPS simulation also

finds the maximum HV, however, it clearly dominates all other algorithms over the whole runtime and achieves the maximum HV the fastest.

VII. DISCUSSION AND CONCLUSION

We proposed and demonstrated an efficient quantum algorithm for Multi-Objective Optimization based on QAOA. In particular, we tested our approach on both MPS simulations and real quantum hardware and showed that parameter transfer allows to scale QAOA for the considered MO-MAXCUT problem and that the algorithm has the potential to outperform classical benchmarks. While the studied problem is still relatively simple, advanced classical algorithms already seem to struggle. As problem sizes, graph densities, and number of objective functions increase further, we expect the runtime of classical solvers to also significantly grow, implying a huge potential for the presented quantum algorithm as quantum computers continue to improve.

There are several opportunities to further improve the proposed quantum algorithm and extend its applicability. First, transfer of parameter and similar strategies to circumvent training need to be expanded to other problem classes, e.g., practically relevant problems in finance or logistics, where trade-offs between objective functions, such as risk and return, are omnipresent, or higher order binary optimization. Further directions to improve the algorithm include warm-starting, custom mixer designs, non-uniform sampling of weights or other scalarization strategies, as well as combining classical and quantum approaches. The whole approach leverages the assumption that QAOA returns a variety of good samples, which is closely related to the concept of fair sampling (e.g., how uniformly are different configurations that have the same objective value sampled). While we empirically confirm the generation of good and diverse samples, further theoretical understanding of QAOA and fair sampling is important, also for leveraging QAOA beyond MOO. In addition to enhancing the quantum algorithm, there may also be a potential to improve the classical algorithms or benchmark additional ones on the considered problem, to further support the potential for quantum advantage.

Our algorithm also offers a new approach for constrained optimization. Adding constraints as objective functions and approximating the Pareto front allows to select those solutions that satisfy the original constraints. This can be used to study equality as well as inequality constraints. Particularly the latter are usually difficult to include in QUBOs without adding additional variables as they cannot be easily modeled as penalty terms. But also for equality constraints it might be beneficial to approach them from the MOO perspective since this eliminates the need to tune the involved penalty factors.

Last, we have shown that today's quantum computers allow forecasting the performance of (heuristic) algorithms on future devices with lower errors or even fault-tolerance.

Similarly, we have shown the opposite direction, i.e., that simulations can be used to estimate the performance of algorithms on noisy quantum computers with different noise levels. This highlights that today's quantum computers are scientific tools to support algorithmic discovery and to augment theoretical research by empirical analyses.

Appendix A: Sampling of c vectors

A c vector, as introduced in Sec. III, forms a convex combination of real numbers, i.e., $c \in [0,1]^m$ with $\sum_i c_i = 1$. Only the non-negativity of vectors is strictly necessary. When sampling from an m-dimensional vector space $V \subset \mathbb{R}_{\geq 0}^m$ and projecting the samples onto the $(m-1)$-dimensional simplex by normalization, the vectors will not be distributed equally on the simplex. To obtain a balanced representation of different objectives, we follow the sampling strategy described in [65], which we hereby summarize. To obtain a single c vector, $m-1$ numbers are uniformly sampled from $[0,1]$ to form a set. Next, 0 and 1 are added to the set to extend it to a set of $m+1$ numbers. A second set is constructed by the differences of consecutive elements from the sorted set. This procedure guarantees that the second set consists of non-negative numbers summing up to 1. Using this procedure will ensure a uniformly distributed set of c vectors on the $(m-1)$-dimensional simplex.

Appendix B: Classical Results for MO-MAXCUT

In this section, we discuss several classical alternatives to the randomized ϵ-CM for solving MO-MAXCUT with $m = 3$ objective functions presented in Sec. VI A. It turns out that DPA-a, DCM, and the randomized ϵ-CM performed best in our tests, which is the reason we use them as a classical reference in the main text. The resulting HV for each algorithm is summarized in Tab. II, and the tested algorithms are discussed in the following. We report the runtimes to generate the samples, without the post-processing to determine non-dominated points or corresponding HV. All runtimes are determined using an AMD EPYC 7773X CPU.

As a trivial baseline, we draw samples uniformly at random from $\{0,1\}^n$ for $n = 42$. Since uniform sampling is cheap, we draw 25 million samples, like in the case of our quantum algorithm. We do not report a runtime, as the samples are generated almost immediately.

In addition, we test the randomized WSM. We use 400,000 random c vectors. It turns out that the WSM performs worse than, e.g., ϵ-CM, which we explain by the presence of non-supported Pareto optimal points that the weighted sum approach cannot find. This is supported by the fact that the WSM only returns 158 non-dominated points, i.e., over an order of magnitude less than the DPA-a and DCM. Every MIP in the WSM takes on average 0.0039s to solve, which, for 400,000 MIPs, results in a single core runtime of about 26 minutes.

Since every c vector defines a new single objective MAXCUT problem, we also test the randomized WSM with the Goemans-Williamson (GW) algorithm [12] as solver. For a given MAXCUT instance, the GW algorithm first maps it to a Semidefinite Program (SDP) that can be solved efficiently, and then applies randomized rounding to sample good solutions. The GW algorithm finds solutions that achieve on average at least 0.87856 of the MAXCUT value. Here, we apply the same strategy as in our quantum algorithm: We sample 5000 random c vectors and then draw 5000 random samples from the SDP solution, which leads to a total of 25 million samples. We use CVXPY [66] to solve the resulting SDPs, which takes 0.6341s per SDP on average. Thus, solving all SDPs on a single core takes about 53 minutes, where the runtime contribution of the randomized rounding is negligible.

The adapted version of the ϵ-CM proposed in [67], in the following referred to as "Adapted ϵ-CM", generates suitable constraints during the process, so that the number of iterations is no input parameter. We tested it with a time limit of 10,000s.

As a representative of multi-objective integer optimization algorithms, we consider the openly accessible implementations of DPA-a [15] and DCM [14]. Both DPA-a and DCM are exact algorithms that enumerate all non-dominated points by decomposing the objective space in search regions and solve the resulting subproblems by scalarization methods calling the solver CPLEX. Each found non-dominated point redefines the decomposition. For this kind of algorithm, there is a trade-off between the number and the complexity of the subproblems. Hence, their performance highly depends on the problem size and the number of objectives.

In DPA (Defining Point Algorithm), the search region

is a union of non-disjoint rectangular sets. Those rectangular subregions are described by upper bound vectors based on the already generated non-dominated points. In each iteration, the IP restricted to one rectangular search region is solved by a scalarization method. We here choose the augmented (DPA-a) ϵ-CM for solving each subproblem, i.e., ϵ-CM with objective function

$$f_i(x) + \rho \sum_{j=1}^{m} f_j(x)$$

for a small augmentation parameter $\rho > 0$. The algorithm guarantees that the number of local upper bound vectors is minimal and that each non-dominated point is generated exactly once.

In DCM (Disjunctive Constraint Method), the search region is decomposed using disjunctive constraints. Disjunctive constraints allow solving one IP call in several subregions by using additional binary variables. In a pure disjunctive constraint method, those constraints would decompose the search region in disjoint subregions. However, in DCM, the disjunctive constraints are only applied to the last generated non-dominated point. Thus, each subregion is defined by lower bounds, upper bounds, and a guiding point. The subproblems are solved by ϵ-CM with objective function

$$\sum_{i=1}^{m} f_i(x).$$

DCM guarantees each non-dominated point to be generated exactly once, and it limits the number of subregions and the number of sets of disjunctive constraints.

Since the implementations of DPA-a and DCM do not allow objective functions taking continuous values, we scale the objective functions by a factor of 1k and 10k, and round them to gain integer values. This corresponds to representing the weights with a precision of 10^{-3} and 10^{-4}. The precision strongly influences the performance. While a high precision can lead to very long runtimes, a low precision can cause a gap to the optimal HV.

Appendix C: QAOA Parameter Optimization with JuliQAOA

The QAOA angle vectors $\vec{\beta}$, $\vec{\gamma}$ are optimized using JuliQAOA [59] on an ensembles of five different 27-qubit problem instances each for the three- and four-objective case, see also Appendix D. JuliQAOA performs full statevector simulation, where the cost of all basis states for the optimization problem are precomputed and supplied to the simulation software (i.e., all 2^n variable configurations, with $n = 27$). The optimization routine is performed using a single basin-hopping iteration per p, and *angle extrapolation* is used to initialize the angle search at $p+1$ using the best angles found at the previous index p. Specifically, the best learned vector of angles

Algorithm	HV	#NDP	runtime [min]
DPA-a (10k)	43,471.704	2067	3.6
DPA-a (1k)	43,471.704	2063	3.6
DCM (1k)	43,471.704	2063	8
ϵ-CM	43,471.222	2054	167
GW algorithm	43,443.382	1567	53
WSM	43,088.010	158	26
Adapted ϵ-CM	39,217.589	117	time limit
Random sampling	35,870.921	220	–
DCM (10k)	30,445.587	4	time limit

TABLE II. Results of classical algorithms: For every tested classical algorithm we report the achieved HV, the corresponding number of unique non-dominated points (#NDP), and its runtime. The algorithms are sorted with respect to descending HV. We set a time limit of 10,000s and terminate the algorithms if they do not finish before. Since random sampling is very cheap to run, we do not provide a runtime.

found at p is then directly substituted into the $p+1$ vector, and the two new parameters for the additional cost and mixer layer are randomly initialized. This entire vector, including the depth-p angles, is then optimized. The Basin-hopping optimization [68] is used to search within the local vicinity of the extrapolated angles, see Ref. [59] for more details. The choice of basin hopping as the optimization routine, and using a single iteration per p, was found to be the best performing set of parameter choices for general QAOA learning in Ref. [59] (in particular, additional basin hopping iterations typically did not substantially improve the quality of the angles found), which is why we use these settings in this context. The single basin-hopping iteration refers to the basin-hopping implementation parameter where we only explore one local minima in the vicinity of the extrapolated angles. Similar ideas to QAOA angle extrapolation has been proposed in Ref. [69] under terms such as stationary points of QAOA energy landscapes and transition states - the underlying idea of using local minima angles found at lower depth allowing for good heuristic angles being found efficiently at higher rounds. The search at $p = 1$ is randomly initialized. JuliQAOA has been used for robust QAOA angle finding in several previous studies [44, 70, 71]. Importantly, although JuliQAOA combines several high quality QAOA angle finding techniques, the resulting angles are not guaranteed to be globally optimal and therefore we consider these angles *heuristic QAOA angles*.

The minimum requirement we impose on the QAOA angle training is that each step of p must reduce the expectation value, i.e., the energy of the cost Hamiltonian. Occasionally, for certain random initializations (of $p = 1$), this requirement is not met during training – if this occurs, then we simply terminate and re-attempt the simulation. To quantify QAOA performance we measure the (noiseless) expectation value of the optimized QAOA angles and then compute its approximation ratio, defined

by,

$$\text{Approximation Ratio} = \frac{C_{\text{Max}} - C_e}{C_{\text{Max}} - C_{\text{Min}}}. \qquad \text{(C1)}$$

This requires finding the minimum (C_{Min}) and maximum (C_{Max}) cost value out of all 2^n possible configurations. Here, C_e denotes the cost of the energy. Importantly, this is an approximation ratio definition for a minimization problem, where lower cost denotes higher approximation ratio. The reason for this is because the Hamiltonian cost function that QAOA is trained on is an energy minimization problem - but this is directly equivalent to the maximization problem of (weighted) MAXCUT. Note that this is an unconventional approximation ratio definition so as to accommodate negative, and positive, energies of the MAXCUT Hamiltonian. This definition also means that random sampling on average will result in an approximation ratio of ≈ 0.5.

Fig. 9 shows the learned QAOA approximation ratios for the five different random problem Hamiltonians with 3-objectives and five different random Hamiltonians with 4-objectives. For our experiments we selected the parameters resulting from the first random instance.
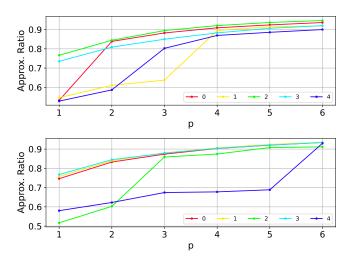

FIG. 9. Approximation ratio of the learned QAOA angles on the set of 5 random weighted MAXCUT instances with combined 3-objectives (top) and 4-objectives (bottom). The y-axis plots the noiseless approximation ratio and the x-axis is the QAOA depth p. Each line shows the approximation ratio for one of the five different distinct problem Hamiltonians, the (arbitrary) index for the problem Hamiltonian is shown by the legend(s). `JuliQAOA`'s angle learning is able to provide a consistent improvement of solution quality; at $p = 6$, QAOA achieves an approximation ratio better than 0.9.

Appendix D: Transfer of QAOA Parameters and Compiled QAOA Circuit Structure

In our framework, problem Hamiltonians are constructed via the weighted sum of the problem graphs.

The weighted sum does not change the structure of the problem Hamiltonians, i.e., the interacting qubit pairs remain the same. Only the coefficients of the individual Pauli terms differ with varying weight sets. Since the graph structure does not change and the weights are randomly selected, intuition suggests that we can transfer the parameters between different graph instances. We show this empirically as we do not have a formal proof. We examine MAXCUT instances on 27-node hardware-native graphs (i) with five different realizations with edge weights sampled from $\mathcal{N}(0, 1/3)$ for the case of three objective functions, and (ii) with five instances with weights sampled from $\mathcal{N}(0, 1/4)$ for the case of four objective functions.

Fig. 10 and 11 show parameter transfer of optimized QAOA angles from 27-qubit instances, as described in Sec. C, onto 27-qubit instances, for 3 and 4 objectives, respectively. Here we are showing the approximation ratio difference between the noiseless expectation value of original (donor) problem and the new problem (acceptor). As before, these QAOA simulations use the single-objective weighted MAXCUT problem Hamiltonian formed from linear combinations of either 3 or 4 other weighted MAXCUT Hamiltonians. This means that the energy of the classical Hamiltonian that the noiseless QAOA numerical simulation computes is measured on this linear combination of weighted MAXCUT Hamiltonians. Overall these small scale numerical experiments demonstrate that QAOA parameter re-use is reasonable to utilize across this set of problem instances – with the exception of one particular instance which did not transfer well (middle instance in the matrices of Fig. 10). In practice, we use the learned QAOA angles from the first random problem instance (donor 0) when executing the QAOA circuits on the noisy quantum hardware.

Appendix E: Mixed Integer Programming Formulation for Weighted-MAXCUT

In this section we briefly introduce the basic MIP formulation for Weighted-MAXCUT that we use in the ϵ-CM. Suppose a weighted graph $(\mathcal{V}, \mathcal{E}, w)$, then MAXCUT can be modeled as [72]

$$\max_{x,s} \sum_{(k,l)\in\mathcal{E}} w_{ij} e_{ij}$$

$$\begin{aligned}
\text{subject to: } & x_k \in \{0,1\}, & \forall k \in \mathcal{V}, \\
& e_{kl} \in \{0,1\}, & \forall (k,l) \in \mathcal{E}, \\
& e_{kl} \leq x_k + x_l, & \forall (k,l) \in \mathcal{E}, \\
& e_{kl} \leq 2 - x_k - x_l, & \forall (k,l) \in \mathcal{E}, \\
& e_{kl} \geq x_k - x_l, & \forall (k,l) \in \mathcal{E}, \\
& e_{kl} \geq x_l - x_k, & \forall (k,l) \in \mathcal{E}.
\end{aligned}$$

The node variables x_k indicate whether node $k \in \mathcal{V}$ is in the first or second class. The edge variables e_{kl} for
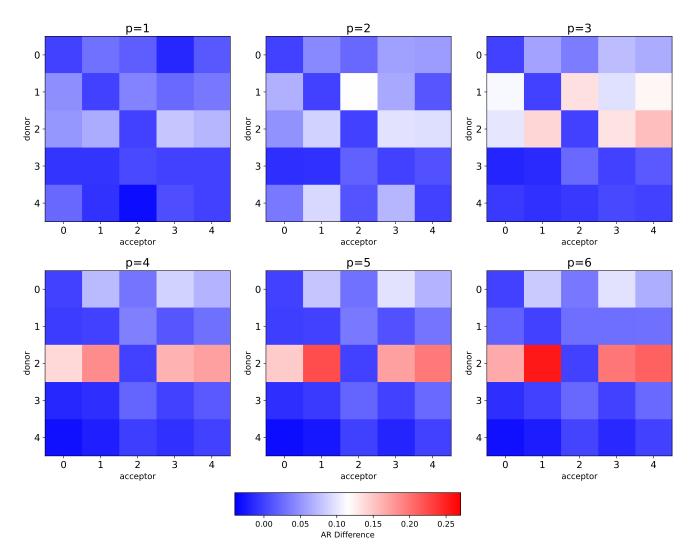
FIG. 10. QAOA parameter transfer matrices for $p = 1$ (top left) to $p = 6$ (bottom right). For a set of 5 fixed problem instances, 5 random coefficient choices each for hardware-native problem instances with 3 combined objective functions, `JuliQAOA` is used to optimize the angles for each problem instance. Each matrix plots the difference between the approximation ratio of QAOA using the angles trained for that problem instance and the approximation ratio of QAOA using those same angles applied to a different (unseen during training) problem instance; each unique instance is named on y-axis and the trained parameters for that instance are then applied to the other 4 problem instances (x-axis). Positive values mean that the original approximation ratio was better than the approximation ratio on the new (unseen) problem. Negative value means the parameters worked even better on the new (unseen) problem. Diagonal entries, by definition, have an approximation ratio difference of zero. The absolute possible range of this approximation ratio difference is $[-1, 1]$.

$(k, l) \in \mathcal{E}$ determine whether the corresponding edge is cut or not. The last two constraints are not needed in case of positive weights. However, if negative weights can occur, the lower bounds need to be added to enforce that $e_{kl} = 1$ is equivalent to $x_k \neq x_l$ for all $(k, l) \in \mathcal{E}$. Further, suppose we have multiple sets of weights w_{kl}^i, for $i = 1, \dots, m$. Then we can add additional constraints $\sum_{(k,l)} w_{kl}^i e_{kl} \geq \epsilon_i$ to enforce a lower bound on the individual MAXCUT objectives and adjust the weights in the objective to represent convex combinations of the w_{kl}^i.

Appendix F: Convergence & Runtime Analysis of MPS Simulations

In this section, we analyze the convergence of the MPS simulations with respect to the bond dimension. To this extent, we consider the 42-node 3-objective MO-MAXCUT problem instance and evaluate the average HV over time, i.e., $\overline{\mathrm{HV}} = 1/T \sum_{t=1}^{T} \mathrm{HV}_t$, for different bond dimensions χ and QAOA repetitions p, as well as the resulting maximum HV for each setting. The HV does not significantly change anymore for $\chi \geq 20$, see Fig. 12, i.e., we assume this is sufficient to represent the
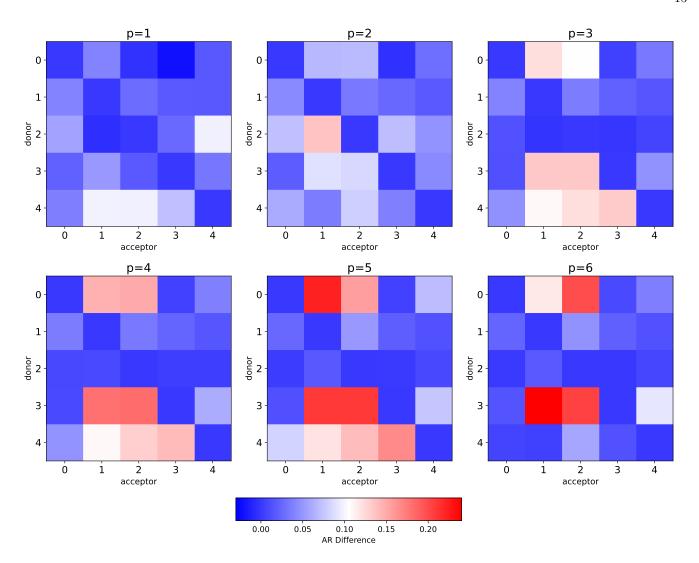
FIG. 11. QAOA parameter transfer matrices for $p = 1$ (top left) to $p = 6$ (bottom right) on five instances of 4-objective weighted MAXCUT heavy-hex instances. Same format as Figure 10, but now working on 4-objective instances as opposed to 3. Negative approximation ratio difference denotes that the trained parameters on the donor instance worked even better on the acceptor angles (this, as expected, happens rarely). Positive values mean the approximation ratio of the trained angles on the donor instance are better than the approximation ratio on the acceptor instance - the more positive the values are, the worse the parameter transfer worked.

considered 42-qubit quantum circuits.

In addition, we analyze the runtime of the MPS simulations to generate the 25 million samples from the 42-qubit QAOA circuits for different bond dimensions χ and different number of repetitions p, as discussed in Sec. VI. We measure the single-core runtime on an *Apple M1 Max* CPU with 32 GB. This can be easily translated to multi-core runtime assuming no memory bottlenecks. Tab. III shows the resulting runtimes. It can be seen that the

runtimes quickly exceed multiple hours for larger χ and p. If we consider $p \geq 3$ with $\chi = 20$, i.e., the settings that first start to achieve $\mathrm{HV}_{\max}$, the single-core runtimes are around three hours, i.e., more than 4-times slower than generating the samples using the real quantum computer. While the simulation can be easily parallelized, the results also very clearly show the exponential growth of the runtimes with increasing bond dimension, which implies that this will quickly become intractable for larger circuits.

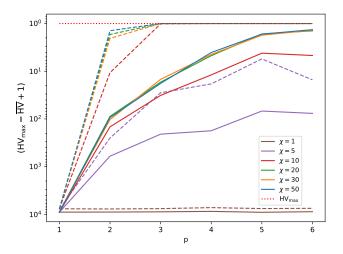[1] Amira Abbas, Andris Ambainis, Brandon Augustino, Andreas Bärtschi, Harry Buhrman, Carleton Coffrin, Giorgio Cortiana, Vedran Dunjko, Daniel J. Egger, Bruce G. Elmegreen, Nicola Franco, Filippo Fratini, Bryce Fuller, Julien Gacon, Constantin Gonciulea, Sander Gribling, Swati Gupta, Stuart Hadfield, Raoul Heese, Gerhard

FIG. 12. Convergence of MPS simulations: This plot shows the different HV time-averages $\overline{HV}$ as well as maximum achieved HV for different bond dimensions χ and QAOA repetitions p. As before, we show $HV_{max} - \overline{HV} + 1$ with a logarithmic and reversed y-axis. All simulations were run with 5000 random c vectors and 5000 shots for each resulting circuit.

				χ		
		1	5	10	20	30	50
	1	9	55	76	106	134	133
	2	9	60	84	168	309	679
p	3	9	59	86	178	341	819
	4	9	59	87	191	355	866
	5	10	59	87	186	364	910
	6	9	59	87	190	370	962

TABLE III. Runtime of MPS simulations: Runtime in minutes to generate 25 million samples for different bond dimensions χ and QAOA repetitions p using a single core on an *Apple M1 Max* CPU.

Kircher, Thomas Kleinert, Thorsten Koch, Georgios Korpas, Steve Lenk, Jakub Marecek, Vanio Markov, Guglielmo Mazzola, Stefano Mensa, Naeimeh Mohseni, Giacomo Nannicini, Corey O'Meara, Elena Peña Tapia, Sebastian Pokutta, Manuel Proissl, Patrick Rebentrost, Emre Sahin, Benjamin C. B. Symons, Sabine Tornow, Victor Valls, Stefan Woerner, Mira L. Wolf-Bauwens, Jon Yard, Sheir Yarkoni, Dirk Zechiel, Sergiy Zhuk, and Christa Zoufal, "Quantum optimization: Potential, challenges, and the path forward," (2023), arXiv:2312.02279 [quant-ph].

[2] Matthias Ehrgott, *Multicriteria Optimization* (Springer Berlin, Heidelberg, 2005).

[3] José Figueira, Carlos Fonseca, Pascal Halffmann, Kathrin Klamroth, Luis Paquete, Stefan Ruzika, Britta Schulze, Michael Stiglmayr, and David Willems, "Easy to say they are hard, but hard to see they are easy- towards a categorization of tractable multiobjective combinatorial optimization problems," Journal of Multi-Criteria Decision Analysis 24 (2016), 10.1002/mcda.1574.

[4] Richard Allmendinger, Andrzej Jaszkiewicz, Arnaud Liefooghe, and Christiane Tammer, "What if we increase the number of objectives? theoretical and empirical im-

plications for many-objective combinatorial optimization," Computers & Operations Research 145, 105857 (2022).

[5] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann, "A quantum approximate optimization algorithm," (2014), arXiv:1411.4028 [quant-ph].

[6] Nery Riquelme, Christian Von Lücken, and Benjamin Baran, "Performance metrics in multi-objective optimization," in *2015 Latin American Computing Conference (CLEI)* (2015) pp. 1–11.

[7] C.M. Fonseca, L. Paquete, and M. Lopez-Ibanez, "An improved dimension-sweep algorithm for the hypervolume indicator," in *2006 IEEE International Conference on Evolutionary Computation* (2006) pp. 1157–1163.

[8] Karl Bringmann and Tobias Friedrich, "Approximating the volume of unions and intersections of high-dimensional geometric objects," Computational Geometry 43, 601–610 (2010).

[9] Jingda Deng and Qingfu Zhang, "Combining simple and adaptive monte carlo methods for approximating hypervolume," IEEE Transactions on Evolutionary Computation 24, 896–907 (2020).

[10] IBM, *IBM ILOG CPLEX Optimization Studio* (2024), version 22.1.

[11] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," (2024).

[12] Michel X. Goemans and David P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," J. ACM 42, 1115–1145 (1995).

[13] Gokhan Kirlik and Serpil Sayın, "A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems," European Journal of Operational Research 232, 479–488 (2014).

[14] Natashia Boland, Hadi Charkhgard, and Martin Savelsbergh, "A new method for optimizing a linear function over the efficient set of a multiobjective integer program," European Journal of Operational Research 260, 904–919 (2017).

[15] Kerstin Dächert, Tino Fleuren, and Kathrin Klamroth, "A simple, efficient and versatile objective space algorithm for multiobjective integer programming," Mathematical Methods of Operations Research 100, 351–384 (2024).

[16] Melih Ozlen, Benjamin A. Burton, and Cameron A. G. MacRae, "Multi-objective integer programming: An improved recursive algorithm," Journal of Optimization Theory and Applications 160, 470–482 (2014).

[17] Julius Bauß, Sophie N. Parragh, and Michael Stiglmayr, "Adaptive improvements of multi-objective branch and bound," (2023), arXiv:2312.12192 [math.OC].

[18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," IEEE Transactions on Evolutionary Computation 6, 182–197 (2002).

[19] J. Blank and K. Deb, "pymoo: Multi-objective optimization in python," IEEE Access 8, 89497–89509 (2020).

[20] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann, "A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem," (2015), arXiv:1412.6062 [quant-ph].

[21] Andrew Lucas, "Ising formulations of many np problems," Frontiers in Physics 2 (2014), 10.3389/fphy.2014.00005.

[22] Sergey Bravyi, Alexander Kliesch, Robert Koenig, and Eugene Tang, "Obstacles to variational quantum optimization from symmetry protection," Physical Review

Letters **125** (2020), 10.1103/physrevlett.125.260505.

[23] Daniel J. Egger, Jakub Mareček, and Stefan Woerner, "Warm-starting quantum optimization," Quantum **5**, 479 (2021).

[24] Reuben Tate, Majid Farhadi, Creston Herold, Greg Mohler, and Swati Gupta, "Bridging classical and quantum with sdp initialized warm-starts for qaoa," (2022), arXiv:2010.14021 [quant-ph].

[25] Joel Rajakumar, John Golden, Andreas Bärtschi, and Stephan Eidenbenz, "Trainability barriers in low-depth QAOA landscapes," in *Proceedings of the 21st ACM International Conference on Computing Frontiers* (2024) pp. 199–206.

[26] Michael Streif and Martin Leib, "Training the quantum approximate optimization algorithm without access to a quantum processing unit," Quantum Science and Technology **5**, 034008 (2020).

[27] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Leo Zhou, "The Quantum Approximate Optimization Algorithm and the Sherrington-Kirkpatrick Model at Infinite Size," Quantum **6**, 759 (2022).

[28] Sami Boulebnane and Ashley Montanaro, "Predicting parameters for the Quantum Approximate Optimization Algorithm for MAX-CUT from the infinite-size limit," (2021), arXiv:2110.10685 [quant-ph].

[29] Joao Basso, Edward Farhi, Kunal Marwaha, Benjamin Villalonga, and Leo Zhou, "The Quantum Approximate Optimization Algorithm at High Depth for MaxCut on Large-Girth Regular Graphs and the Sherrington-Kirkpatrick Model," in *17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022)*, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 232, edited by François Le Gall and Tomoyuki Morimae (Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2022) pp. 7:1–7:21.

[30] Jonathan Wurtz and Peter Love, "Maxcut quantum approximate optimization algorithm performance guarantees for $p > 1$," Phys. Rev. A **103**, 042612 (2021).

[31] Jonathan Wurtz and Danylo Lykov, "The fixed angle conjecture for QAOA on regular MaxCut graphs," (2021), arXiv:2107.00677 [quant-ph].

[32] J. A. Montanez-Barrera and Kristel Michielsen, "Towards a universal qaoa protocol: Evidence of a scaling advantage in solving some combinatorial optimization problems," (2024), arXiv:2405.09169 [quant-ph].

[33] Sami Khairy, Ruslan Shaydulin, Lukasz Cincio, Yuri Alexeev, and Prasanna Balaprakash, "Learning to optimize variational quantum circuits to solve combinatorial problems," Proceedings of the AAAI Conference on Artificial Intelligence **34**, 2367–2375 (2020).

[34] Mahabubul Alam, Abdullah Ash-Saki, and Swaroop Ghosh, "Accelerating quantum approximate optimization algorithm using machine learning," in *Proceedings of the 23rd Conference on Design, Automation and Test in Europe*, DATE '20 (EDA Consortium, San Jose, CA, USA, 2020) p. 686–689.

[35] J. A. Montanez-Barrera, Dennis Willsch, and Kristel Michielsen, "Transfer learning of optimal qaoa parameters in combinatorial optimization," (2024), arXiv:2402.05549 [quant-ph].

[36] Alexey Galda, Xiaoyuan Liu, Danylo Lykov, Yuri Alexeev, and Ilya Safro, "Transferability of optimal qaoa parameters between random graphs," in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)* (2021) pp. 171–180.

[37] Brandon Augustino, Madelyn Cain, Edward Farhi, Swati Gupta, Sam Gutmann, Daniel Ranard, Eugene Tang, and Katherine Van Kirk, "Strategies for running the qaoa at hundreds of qubits," (2024), arXiv:2410.03015 [quant-ph].

[38] Fernando G. S. L. Brandão, Michael Broughton, Edward Farhi, Sam Gutmann, and Hartmut Neven, *For Fixed Control Parameters the Quantum Approximate Optimization Algorithm's Objective Function Value Concentrates for Typical Instances* (2019).

[39] Alexey Galda, Eesh Gupta, Jose Falla, Xiaoyuan Liu, Danylo Lykov, Yuri Alexeev, and Ilya Safro, "Similarity-based parameter transferability in the quantum approximate optimization algorithm," Frontiers in Quantum Science and Technology **2** (2023), 10.3389/frqst.2023.1200975.

[40] Shree Hari Sureshbabu, Dylan Herman, Ruslan Shaydulin, Joao Basso, Shouvanik Chakrabarti, Yue Sun, and Marco Pistoia, "Parameter setting in quantum approximate optimization of weighted problems," Quantum **8**, 1231 (2024).

[41] Ruslan Shaydulin, Phillip C. Lotshaw, Jeffrey Larson, James Ostrowski, and Travis S. Humble, "Parameter transfer for quantum approximate optimization of weighted maxcut," ACM Transactions on Quantum Computing **4** (2023), 10.1145/3584706.

[42] V. Akshay, D. Rabinovich, E. Campos, and J. Biamonte, "Parameter concentrations in quantum approximate optimization," Phys. Rev. A **104**, L010401 (2021).

[43] Elijah Pelofske, Andreas Bärtschi, and Stephan Eidenbenz, "Short-Depth QAOA circuits and Quantum Annealing on Higher-Order Ising Models," npj Quantum Information (2024), 10.1038/s41534-024-00825-w.

[44] Elijah Pelofske, Andreas Bärtschi, Lukasz Cincio, John Golden, and Stephan Eidenbenz, "Scaling whole-chip QAOA for higher-order Ising spin glass models on heavy-hex graphs," npj Quantum Information **10** (2024), 10.1038/s41534-024-00906-w.

[45] Vivek Katial, Kate Smith-Miles, Charles Hill, and Lloyd Hollenberg, "On the instance dependence of parameter initialization for the quantum approximate optimization algorithm: Insights via instance space analysis," INFORMS Journal on Computing **37**, 146–171 (2024).

[46] Shao-Hen Chiew, Kilian Poirier, Rajesh Mishra, Ulrike Bornheimer, Ewan Munro, Si Han Foon, Christopher Wanru Chen, Wei Sheng Lim, and Chee Wei Nga, "Multiobjective optimization and network routing with near-term quantum computers," IEEE Transactions on Quantum Engineering **5**, 1–19 (2024).

[47] Zakaria Abdelmoiz Dahi, Francisco Chicano, Gabriel Luque, Bilel Derbel, and Enrique Alba, "Scalable quantum approximate optimiser for pseudo-boolean multi-objective optimisation," in *Parallel Problem Solving from Nature – PPSN XVIII*, edited by Michael Affenzeller, Stephan M. Winkler, Anna V. Kononova, Heike Trautmann, Tea Tušar, Penousal Machado, and Thomas Bäck (Springer Nature Switzerland, Cham, 2024) pp. 268–284.

[48] Linus Ekstrom, Hao Wang, and Sebastian Schmitt, "Variational quantum multi-objective optimization," (2024), arXiv:2312.14151 [quant-ph].

[49] Hao Li, Daowen Qiu, and Le Luo, "Distributed exact multi-objective quantum search algorithm," (2024), arXiv:2409.04039 [quant-ph].

[50] Lov K. Grover, "A fast quantum mechanical algorithm

for database search," (1996), arXiv:quant-ph/9605043 [quant-ph].

[51] John Golden, Andreas Bärtschi, Daniel O'Malley, and Stephan Eidenbenz, "Fair Sampling Error Analysis on NISQ Devices," ACM Transactions on Quantum Computing **3**, 1–23 (2022).

[52] Elijah Pelofske, John Golden, Andreas Bartschi, Daniel O'Malley, and Stephan Eidenbenz, "Sampling on NISQ Devices: "Who's the Fairest One of All?"," in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, 2021) p. 207–217.

[53] Edward Farhi, David Gamarnik, and Sam Gutmann, "The quantum approximate optimization algorithm needs to see the whole graph: A typical case," (2020), arXiv:2004.09002 [quant-ph].

[54] Christopher Chamberland, Guanyu Zhu, Theodore J. Yoder, Jared B. Hertzberg, and Andrew W. Cross, "Topological and Subsystem Codes on Low-Degree Graphs with Flag Qubits," Physical Review X **10** (2020), 10.1103/physrevx.10.011022.

[55] IBM Quantum, "Ibm quantum platform," `https://quantum.ibm.com`, [Online; accessed 17-February-2025].

[56] Sebastián V. Romero, Anne-Maria Visuri, Alejandro Gomez Cadavid, Enrique Solano, and Narendra N. Hegade, "Bias-Field Digitized Counterdiabatic Quantum Algorithm for Higher-Order Binary Optimization," (2024), arXiv:2409.04477 [quant-ph].

[57] Youngseok Kim, Andrew Eddins, Sajant Anand, Ken Xuan Wei, Ewout Van Den Berg, Sami Rosenblatt, Hasan Nayfeh, Yantao Wu, Michael Zaletel, Kristan Temme, *et al.*, "Evidence for the utility of quantum computing before fault tolerance," Nature **618**, 500–505 (2023).

[58] Elijah Pelofske, Andreas Bärtschi, and Stephan Eidenbenz, "Quantum Annealing vs. QAOA: 127 Qubit Higher-Order Ising Problems on NISQ Computers," in *High Performance Computing* (Springer Nature Switzerland, 2023) p. 240–258.

[59] John Golden, Andreas Baertschi, Dan O'Malley, Elijah Pelofske, and Stephan Eidenbenz, "JuliQAOA: Fast, Flexible QAOA Simulation," in *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, SC-W 2023 (ACM, 2023).

[60] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah, "Julia: A fresh approach to numerical computing," SIAM review **59**, 65–98 (2017).

[61] "Qiskit aer," `https://github.com/Qiskit/qiskit-aer`, [Online; accessed 17-February-2025].

[62] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta, "Quantum computing with Qiskit," (2024), arXiv:2405.08810 [quant-ph].

[63] Gopal Chandra Santra, Sudipto Singha Roy, Daniel J. Egger, and Philipp Hauke, "Genuine multipartite entanglement in quantum optimization," Phys. Rev. A **111**, 022434 (2025).

[64] Samantha V. Barron, Daniel J. Egger, Elijah Pelofske, Andreas Bärtschi, Stephan Eidenbenz, Matthis Lehmkuehler, and Stefan Woerner, "Provable bounds for noise-free expectation values computed from noisy samples," Nature Computational Science **4**, 865–875 (2024).

[65] Donald B. Rubin, "The Bayesian Bootstrap," The Annals of Statistics **9**, 130 – 134 (1981).

[66] Steven Diamond and Stephen Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," Journal of Machine Learning Research (2016), to appear.

[67] Marco Laumanns, Lothar Thiele, and Eckart Zitzler, "An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method," European Journal of Operational Research **169**, 932–942 (2006).

[68] David J. Wales and Jonathan P. K. Doye, "Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms," The Journal of Physical Chemistry A **101**, 5111–5116 (1997).

[69] Stefan H. Sack, Raimel A. Medina, Richard Kueng, and Maksym Serbyn, "Recursive greedy initialization of the quantum approximate optimization algorithm with guaranteed improvement," Physical Review A **107** (2023), 10.1103/physreva.107.062404.

[70] Elijah Pelofske, Andreas Bärtschi, John Golden, and Stephan Eidenbenz, "High-Round QAOA for MAX k-SAT on Trapped Ion NISQ Devices," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, 2023) p. 506–517.

[71] John Golden, Andreas Bärtschi, Daniel O'Malley, and Stephan Eidenbenz, "The Quantum Alternating Operator Ansatz for Satisfiability Problems," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, 2023) p. 307–312.

[72] Jonas Charfreitag, Sven Mallach, and Petra Mutzel, "Integer programming for the maximum cut problem: A refined model and implications for branching," in *SIAM Conference on Applied and Computational Discrete Algorithms (ACDA23)* (2023) pp. 63–74, https://epubs.siam.org/doi/pdf/10.1137/1.9781611977714.6.