nature computational science

Article

https://doi.org/10.1038/s43588-025-00873-y

Quantum approximate multi-objective optimization

Received: 27 April 2025

Accepted: 20 August 2025

Published online: 24 October 2025

Check for updates

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

The goal of multi-objective optimization is to understand optimal trade-offs between competing objective functions by finding the Pareto front, that is, 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. Here we use a 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.

Quantum computing is a 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, that is, 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 a 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). Note that any quadratic unconstrained binary optimization (QUBO) problem can be mapped to MAXCUT by adding just one variable, as, for instance, outlined by Bhattacharyya et al.[5], making our demonstrations relevant to a wide range of applications. Additionally, we show how today's quantum computers can help forecast the performance of heuristics on future devices, serving as tools for algorithmic discovery.

Results

We demonstrate our proposal on instances of MO-MAXCUT. The goal of single-objective weighted MAXCUT 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:

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

To introduce m 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 $\mathcal{E}$, that

[1]IBM Quantum, IBM Research Europe—Zurich, Rueschlikon, Switzerland. [2]Zuse Institute Berlin, Berlin, Germany. [3]Los Alamos National Laboratory, Los Alamos, NM, USA. [4]Technische Universität Berlin, Berlin, Germany. e-mail: wor@zurich.ibm.com

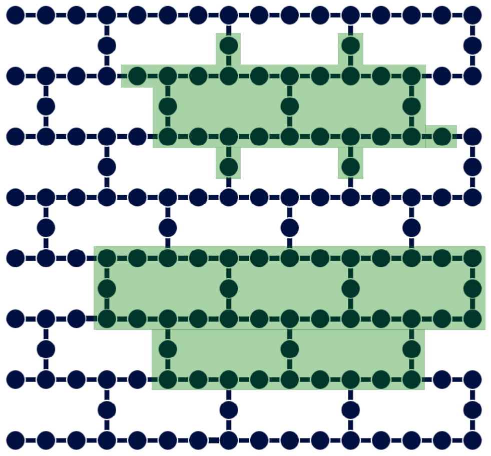


Fig. 1 | Graph topologies. The whole graph shows the coupling map of the 156-qubit ibm_fez device. The green-shaded areas show the 27-qubit graph of the training problem (top) and the 42-qubit graph of the considered MO-MAXCUT problem (bottom) embedded in the ibm_fez coupling map.

is, 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[6] of IBM Quantum devices[7] to facilitate execution on real quantum hardware (Fig. 1). Defining the problem instance using the native two-qubit gate connectivity graph of the hardware allows us to construct relatively low-depth QAOA circuits that can be executed with high fidelity[8–12]. 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. 'Quantum approximate optimization algorithm'). These gates are then compiled to the hardware native basis gate set $\{X, \sqrt{X}, \mathrm{RZ}, \mathrm{CZ}\}$, that is, an 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 that RZZ-gates are implemented using two CZ-gates. Since, on the considered device subgraph to which we compile the QAOA circuits, all RZZ-gates have a uniform duration of 84 ns, 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, that is, $w^i_{kl} \sim \mathcal{N}(0,1)$, for $(k,l) \in \mathcal{E}$, $i = 1, \ldots, m$. Note that considering continuous weights makes these problems particularly challenging for classical algorithms; cf. 'Classical algorithms for MOO' and Supplementary Information 'Classical results for MO-MAXCUT'.

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 s.d. $1/m$ accounts for the randomized convex combinations of the m objective functions used in our algorithm as discussed in 'QAOA for MOO'. For MO-MAXCUT, this corresponds to convex combinations of the edge weights w^i, leading to a lower s.d. compared with edge weights drawn from the standard normal distribution. The s.d. of $1/m$ is equivalent to taking the mean of m random weights drawn independently from a standard normal distribution. The corresponding (unweighted) 27-node and 42-node graphs are shown in Fig. 1. 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 to generate the results of the problem of interest.

To optimize the QAOA parameters for the 27-node single-objective MAXCUT problem, we use JuliQAOA[13], which is a Julia-based[14] QAOA simulator. It efficiently evaluates expectation values and corresponding gradients on the basis of the QAOA statevector and applies a basin-hopping optimization algorithm to navigate the many local minima in the parameter landscape. More details of the optimization pipeline are provided in Supplementary Information 'QAOA parameter optimization with JuliQAOA'. Then, we transfer the trained parameters to the larger instances as discussed in 'QAOA for MOO'. We empirically confirm that transfer of parameters works for the considered class of problems, as illustrated in Supplementary Information 'Transfer of QAOA parameters and compiled QAOA circuit structure'. This allows us to pre-optimize the QAOA angles for circuit complexity $p >= 1, \ldots, 6$, independently of the concrete 42-node MO-MAXCUT problem instance. In the following we discuss the results for the two considered problem instances.

MO-MAXCUT with three objective functions

To assess and compare the performance, we evaluate the progress of the achieved hypervolume HV_t over time t. We always report single-thread central processing unit and quantum processing unit times. First, we tested the classical algorithms discussed in 'Classical algorithms for MOO' on the 42-node MO-MAXCUT problem with $m = 3$ objective functions. DCM, DPA-a and the ϵ-constraint method (ϵ-CM) performed best and are discussed in more detail here. For the results of the other algorithms see Supplementary Information 'Classical results for MO-MAXCUT'.

For DCM and DPA-a, we scaled the continuous weights by a factor of 1,000 and rounded them to integers, since the available implementations of the algorithms cannot handle continuous weights natively. While we set a time limit of 10,000 s for both algorithms, DPA-a terminated after 3.6 min and DCM after 8 min. Thus, both find the global optimal HV for the truncated weights. The HV achieved by the resulting non-dominated points evaluated for the exact non-truncated weights is $\mathrm{HV}_{\max} = 43{,}471.704$. Thus, this is the optimal HV up to the error introduced by the truncation of weights. Both algorithms found 2,063 non-dominated solutions.

Note that DCM in particular 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 few as four non-dominated points before reaching the given time limit. This shows that continuous weights can pose a particular challenge for classical algorithms, since these problems do not admit a canonical grid structure in objective space. DPA-a was more robust to the scaling factor for the considered problem instance and achieved comparable performances in the two cases; cf. Supplementary Information 'Classical results for MO-MAXCUT'.

The HV progress over time for all algorithms discussed in this section is shown in Fig. 2. 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 steep convergence of DPA-a to $\mathrm{HV}_{\max}$ near the end of the runtime results from the ordering of occurring subproblems along the first objective, which provably improves the overall performance but may delay finding impactful solutions[15]. The number of non-dominated points for all tested algorithms is shown in Fig. 3.

Next, we executed ϵ-CM, again setting a time limit of 10,000 s. In our tests, every resulting mixed integer program (MIP) has been solved to optimality using Gurobi in 0.0218 s on average using a single thread on AMD EPYC 7773X central processing units. 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 MAXCUT. Since the randomized ϵ-CM adds constraints

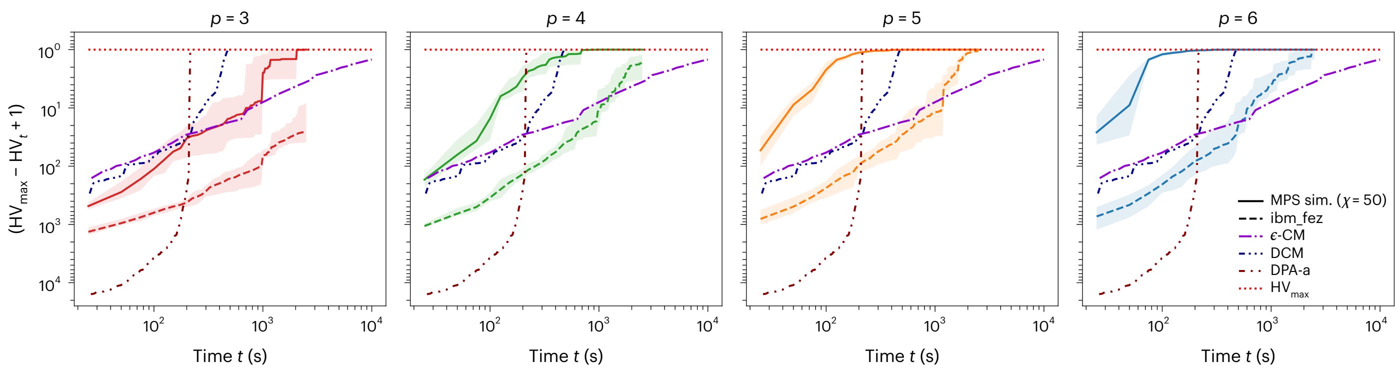


Fig. 2 | Comparison of quantum and classical results for MO-MAXCUT with *m* = 3 objective functions. Mean progress of HV_t over five repetitions of the MPS simulation ($\chi = 50$) and calculations on ibm_fez, as well as results of single runs of the classical algorithms DCM, DPA-a and ϵ-CM. The shaded areas denote the corresponding minimum and maximum performance over the five repetitions of the MPS simulation and results from ibm_fez. The best found solution $\mathrm{HV}_{\max}$ is indicated. The MPS simulations are assumed to achieve the same sampling rates as the considered quantum processing unit, that is, 10,000 shots per second.

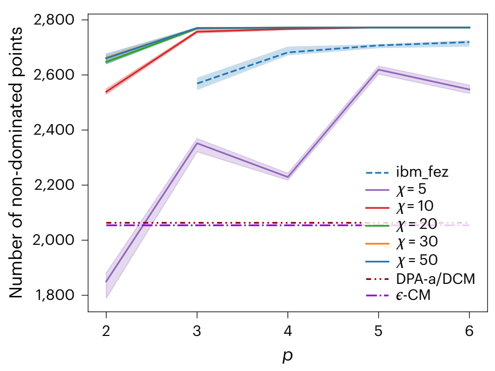


Fig. 3 | Number of non-dominated points. The solid lines show the mean number of unique non-dominated points approximating the Pareto front resulting from the five repetitions using MPS simulation, with the shaded areas denoting the corresponding minimum and maximum. The largest value reached was 2,772. 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 show the results from the five runs on ibm_fez, reaching a maximum of 2,726. 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 single runs of DPA-a and DCM (both achieve the same value) and the ϵ-CM.

to the different objectives this is more suitable for MIP solvers than the QUBO formulation, which would result in non-convex quadratic constraints. See Supplementary Information 'Mixed integer programming formulation for weighted-MAXCUT' for more details. The maximum HV achieved by the ϵ-CM was 43,471.222, that is, slightly below $\mathrm{HV}_{\max}$, where the Pareto front was approximated by 2,054 non-dominated points.

As discussed in 'Classical algorithms for MOO', 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 three 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 hyperrectangle 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[16,17] 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 5,000 random c vectors as described in Supplementary Information 'Sampling of c vectors'. Given the precomputed QAOA parameters from the 27-qubit problem instance, every sampled c vector fully defines a corresponding QAOA circuit and we take 5,000 shots from each of them. In total, this results in 25 million shots. These numbers were selected empirically and turned out to be sufficient for the considered experiments. After sampling a batch of shots corresponding to a c vector, we apply a non-dominated sorting, that is, we discard all dominated samples and only keep the non-dominated ones.

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. Supplementary Information 'Convergence & runtime analysis of MPS simulations'. Thus, 25 million shots are assumed to take in total 2,500 s. Then, we compute the corresponding HV_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 mean performance. The results are shown in Fig. 4. $\chi = 20$ seems sufficient to evaluate the considered circuits, as there are only small differences between the results for $\chi = 20$, 30 and 50. This is further supported by the analysis presented in Supplementary Information 'Convergence & runtime analysis of MPS simulations'. The resulting number of non-dominated points approximating the Pareto front is shown in Fig. 3. It achieves a maximum of 2,772. The difference from 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[7], again using 5,000 random c vectors and 5,000 shots per circuit. There are 12 native options to place the considered circuits on the topology of ibm_fez without inserting any SWAP gates, one of them shown in Fig. 1. 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, that is, subset of qubits, that leads to the highest fidelity. The estimated fidelities are reported in Table 1 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 substantially shorter (see the durations reported in Table 1), 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

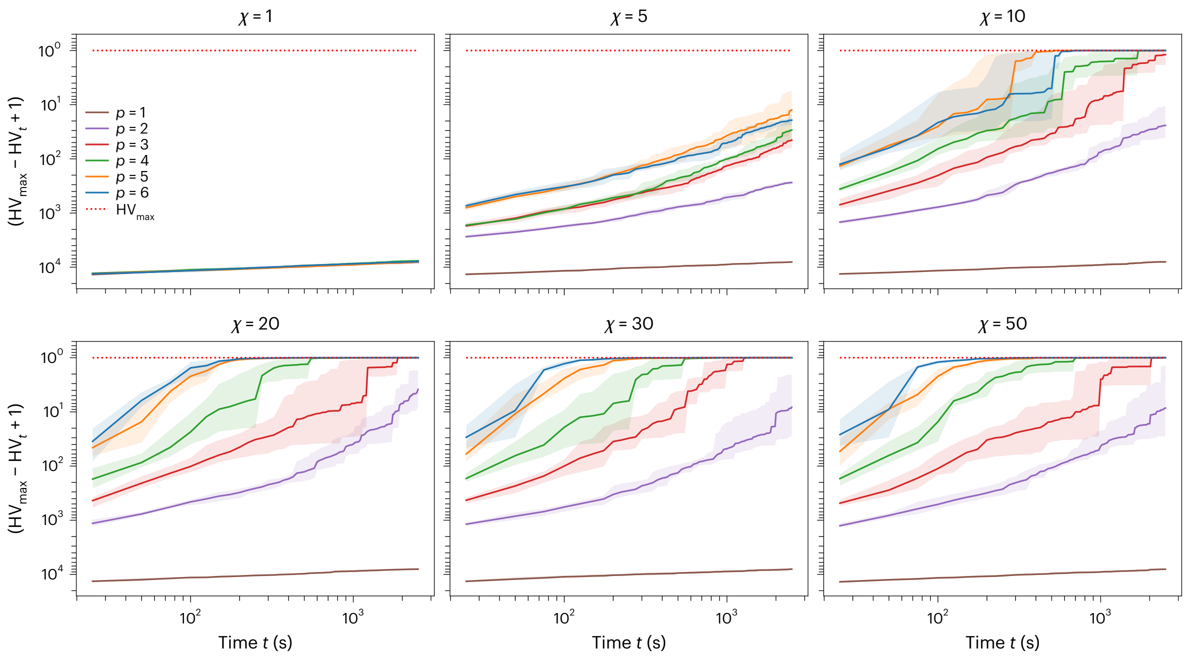


Fig. 4 | MPS simulation results. HV_t for different χ and QAOA rounds (that is, depth) p. Each line represents the mean of five random repetitions of the same setting with shaded areas indicating the minimum and maximum values. The best found solution $\mathrm{HV}_{\max}$ is indicated. The MPS simulations are assumed to achieve the same sampling rates as the considered QPU, that is, 10,000 shots per second.

Table 1 | 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

p	No. 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

The duration of a CZ gate on ibm_fez is uniformly 84 ns, which makes up the majority of the circuit duration. The remainder originates from single-qubit gates and final measurements.

primitive accessible via the IBM Qiskit Runtime[7]. This sampling rate is an idealistic assumption, since it excludes, for example, circuit compilation. However, we believe that 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.

Figure 2 shows the results for DPA-a, DCM, ϵ-CM and the MPS simulations with $\chi = 50$ and the results from ibm_fez. Assuming a 10-kHz 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 2,500 s for all five repetitions. The corresponding numbers of non-dominated points approximating the Pareto front are shown in Fig. 3. Notably, the samples generated by ibm_fez also improve with increasing p, and for $p = 4, 5$ and 6 overtake the ϵ-CM on average after 1,230 s, 1,190 s and 535 s, respectively. For $p = 4$, the mean results from ibm_fez after 2,500 s are comparable to the results of the ϵ-CM after 10,000 s. For $p = 5, 6$, the mean results even overtake the best results from the ϵ-CM and approach the best observed value $\mathrm{HV}_{\max}$. This is quite remarkable and underscores the potential of the proposed approach. A summary of the corresponding single-objective results can be found in Supplementary Information 'Single-objective QAOA results', highlighting that transfer of parameters from 27- to 42-qubit problems works.

The presented results allow us already to draw two key conclusions. First, they clearly show that performance improves with increasing bond dimension. While p has no impact for $\chi = 1$, the performance greatly improves as χ increases, that is, as the MPS simulator approximates the circuits more accurately. Second, for 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, for example, an MPS simulator. These observations are consistent with the work of Santra et al.[18] that shows how genuine multi-partite entanglement is needed to obtain good results with quantum approximate optimization. This provides a strong indication that MOO is a promising candidate for a potential quantum advantage in optimization.

The samples generated by the quantum computer are affected by noise, which explains the differences to the MPS simulations. The estimated fidelities in Table 1, derived from the errors reported by the backend, provide an indication of the strength of the noise. As discussed by Barron et al.[19], 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 is 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 mean 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 Extended Data Fig. 1, match the 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.

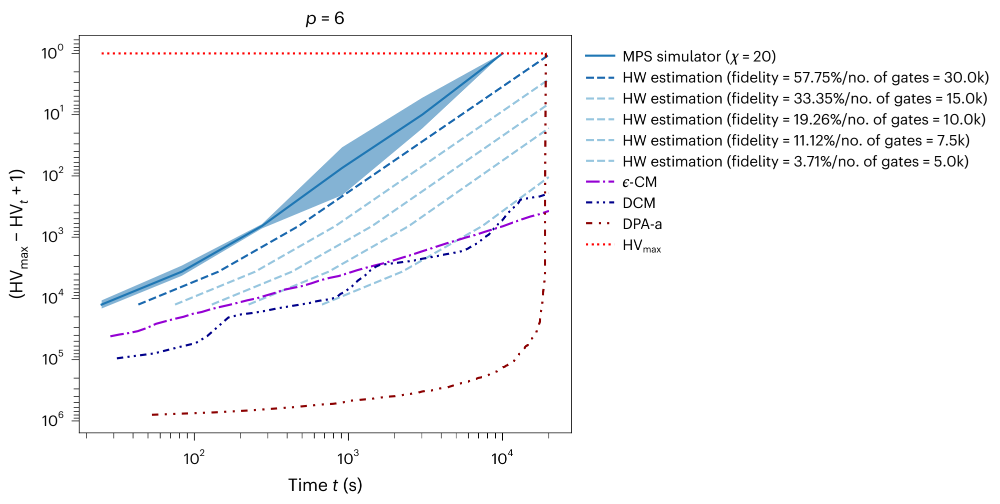


Fig. 5 | Comparison of classical and estimated quantum results for MO-MAXCUT with $m = 4$ objective functions. Mean progress of HV_t resulting from five MPS simulations ($\chi = 20$), estimated hardware (HW) for fidelities ranging from 3.71% to 57.75% and single runs of the classical algorithms DCM, DPA-a and ϵ-CM. 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.

MO-MAXCUT with four objective functions

Next, we analyze a MO-MAXCUT problem instance with $m = 4$ objective functions. The underlying unweighted graph is the same as in 'MO-MAXCUT with three objective functions' 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. Supplementary Information 'Transfer of QAOA parameters and compiled QAOA circuit structure' for more details on the training and transfer of parameters.

As in 'MO-MAXCUT with three objective functions', we test DCM and DPA-a with weights scaled by a factor of 1,000 and then rounded to integers. Further, we run the ϵ-CM. For all three algorithms we set a time limit of 20,000 s, which results in about 740,000 random c and ϵ vectors—that is, MIPs—for the ϵ-CM. Since we have to add an additional constraint to the corresponding MIPs, the mean time to solve one instance slightly increases to 0.0270 s per MIP compared with 0.0218 s for the case of three objective functions. Again, DPA-a performs best among the tested classical algorithms but takes substantially longer than for three objectives. It terminates just before reaching the time limit and finds the optimal Pareto front for the truncated weights, with $HV_{max} = 1{,}266{,}143.350$ for the exact weights.

In addition to the classical algorithms, we run QAOA with $p = 6$, 20,000 random c vectors and 5,000 shots per resulting circuit, that is, we generate in total 100 million samples—four times more than for three objectives. Note that the circuit structure does not change compared with 'MO-MAXCUT with three objective functions'. Assuming again 10,000 shots per second, this results in a total runtime of 10,000 s. We run five repetitions of the MPS simulation with $\chi = 20$, which has been identified as sufficient in 'MO-MAXCUT with three objective functions'. The resulting HV_t for the different algorithms are shown in Fig. 5.

In addition to the algorithms mentioned above, we estimate the performance on real quantum computers by scaling the MPS simulation time by the inverse of the estimated fidelities and truncating after 20,000 s. For ibm_fez, the fidelity has been estimated as 3.71% (cf. Table 1), corresponding to a scaling factor of 26.954. According to the IBM Quantum Roadmap[20], ibm_fez can execute circuits with up to 5,000 two-qubit gates using error mitigation (not applicable to the sampling needed here). Future devices are projected to support 7,500, 10,000 and 15,000 two-qubit gates in 2026, 2027 and 2028, respectively. First fault-tolerant devices are expected to emerge 2029. By comparing these capabilities with today's device, we extrapolate fidelities using the relation $0.0371^{5000/\text{num\_gates}}$. Using the same model, we find that a fidelity of 57.75% (corresponding to a device supporting 30,000 gates) enables the QAOA to match the final HV of DPA-a, while outperforming it throughout most of the runtime. Following the discussion on the fidelity in 'MO-MAXCUT with three objective functions', these provide lower bounds on the performance expected from ibm_fez and future devices. It can be seen that after approximately 5,000 s the predicted HV for ibm_fez (3.71% fidelity) 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}. Overall, even a modest fidelity improvement—as projected for near-term devices—makes the algorithm competitive with classical approaches.

We can see that the problem becomes substantially 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.

Discussion

We have proposed and demonstrated an efficient quantum algorithm for MOO based on a QAOA. In particular, we tested our approach on both MPS simulations and real quantum hardware and showed that parameter transfer allows us to scale the 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 numbers of objective functions increase further, we expect the runtime of classical solvers to also substantially 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—for example, 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. This is a very active domain of research and our approach will profit immediately from all related advances, allowing us to extend our results to broader problem classes. Further directions to improve the algorithm include warm-starting, custom mixer designs and non-uniform sampling of weights or other scalarization strategies, as well as combining classical and quantum approaches. The whole approach leverages the assumption that the QAOA returns a variety of good samples, which is closely related to the concept of fair sampling (for example, how uniformly different configurations that have the same objective value are sampled). While we empirically confirm the generation of good and diverse samples, further theoretical understanding of the QAOA and fair sampling is important, also for leveraging the QAOA beyond MOO. Related to this, the optimal trade-off between solution quality and solution diversity, for example, controlled by the QAOA depth, needs to be understood better. 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 an approach for constrained optimization. Adding constraints as objective functions and approximating the Pareto front allows us to select those solutions that satisfy the original constraints. This can be used to study equality as well as inequality constraints. The latter are usually particularly difficult to include in QUBOs without adding additional variables, as they cannot be easily modeled as penalty terms. However, it might be beneficial to approach also equality constraints from the MOO perspective, since this eliminates the need to tune the involved penalty factors.

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

Methods

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 $X \subset \mathbb{R}^n$, $n \in \mathbb{N}$ (ref. 2):

$$\max_{\mathbf{x}\in X} \quad (f_1(x), \ldots, f_m(x)). \tag{2}$$

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 $f_j(x) > f_j(y)$. x is called Pareto optimal if no y exists 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, for example, in the case of discrete optimization with $X \subset \mathbb{Z}^n$, this is no longer true, and the Pareto front may consist of a finite discrete set of points[2]. We call Pareto-optimal solutions supported if they lie on the boundary of the convex hull of the Pareto front, and non-supported otherwise. Discrete MOO problems in particular are often not efficiently solvable even when the corresponding single-objective optimization problems are in **P**, that is, 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 $Q_i \in \mathbb{R}^{n\times n}$, that is, we consider the MOO variant of 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 HV being the one most commonly used. A survey of performance metrics for MOO is given by Riquelme et al.[21]. The HV of a given set $\{(f_1(x), \ldots, f_m(x)) | x \in S\}$, 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 $\mathbf{r} \in \mathbb{R}^m$ 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 is equal to the Pareto front, the HV is maximized. Throughout this work we focus primarily on the HV since it captures both convergence and diversity in the objective space. The reference point r can be chosen, for example, 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—for example, leveraging Monte Carlo sampling[22–24]. Extended Data Fig. 2 illustrates a non-convex Pareto front, its convex hull and (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.

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-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

$$f_c(\mathbf{x}) = \sum_{i=1}^{m} c_i f_i(\mathbf{x}), \tag{3}$$

where $c \in [0, 1]^m$ with $\sum_i c_i = 1$ denoting a vector of weights of the convex combination of objectives. Solving $\max_{\mathbf{x}\in X} f_c(\mathbf{x})$ for an appropriate discretization of c vectors leads to solutions $\mathbf{x}_c^*$ with corresponding objective vectors $(f_1(\mathbf{x}_c^*), \ldots, f_m(\mathbf{x}_c^*))$ and allows us to approximate the Pareto front. Assuming that we can solve these problems to global optimality (for example, by using commercial solvers such as CPLEX[25] or Gurobi[26]), all resulting points lie on the Pareto front. However, the WSM can only find supported solutions, that is, solutions whose corresponding vectors of objective values lie on the convex hull of the Pareto front[2]. As illustrated in Extended Data Fig. 2 and shown in Supplementary Information 'Classical results for MO-MAXCUT', there can be a substantial gap between the HV achieved by supported solutions and the optimal HV of all solutions, including non-supported ones. In the 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:

$$\max_{\mathbf{x}\in X} \quad f_i(\mathbf{x}) \tag{4}$$

$$\text{subject to:} \quad f_j(\mathbf{x}) \geq \epsilon_j, \ \forall j \neq i. \tag{5}$$

By solving such problems for all objective functions with appropriate discretizations of the ϵ_j, this method allows us 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. 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].

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 solve the resulting problems with an appropriate solver. We sample them uniformly at random from the feasible set; cf. Supplementary Information 'Sampling of c vectors' 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 (for example, using the Goemans–Williamson algorithm[27] to solve the resulting problems) are possible, cf. Supplementary Information 'Classical results for MO-MAXCUT'.

Similarly, we propose a randomized variant of the ϵ-CM. First, we minimize and maximize the individual objective functions to obtain lower and upper bounds $l_i \le f_i(x) \le u_i$, for all $x \in X$, where the lower bounds coincide with the reference point r that we use for the HV, cf. 'Multi-objective optimization'. 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 as in the randomized WSM and solve the problem

$$\max_{\mathbf{x}\in X} \quad \sum_{i=1}^{m} c_i f_i(\mathbf{x}) \tag{6}$$

$$\text{subject to:} \quad f_j(\mathbf{x}) \ge \epsilon_j, \ \forall j. \tag{7}$$

This method is a little 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, that is, 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 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[15,28,29] or by recursive reduction[30]. In contrast, there are also decision-space algorithms, for example, following a branch-and-bound approach[31].

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[30], others, in theory, can also handle continuous objective coefficients, including Epsilon[28], DCM[29] and multi-objective branch-and-bound[31]. However, publicly available implementations of these algorithms are generally restricted to integer objective coefficients. Applying these algorithms to problems with continuous objective coefficients by scaling the objectives is possible, but not in all cases efficient due to a substantial increase in the size of the objective space, which corresponds to the precision of the coefficients, leading to longer runtimes. As representatives for these multi-objective integer algorithms, we consider DPA-a and DCM. See Supplementary Information 'Classical results for MO-MAXCUT' for further details.

In addition to the MIP-based algorithms mentioned above, there are also several genetic algorithms that have been proposed for MOO—for example, by Deb et al.[32]. Getting these algorithms to work is known to require a lot of fine tuning of all steps involved. A test of available implementations[33] did not lead to competitive results and we leave it to future work to further study these approaches.

Quantum approximate optimization algorithm

The QAOA[34,35] 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

$$H_C = -\sum_{i<j} J_{ij}\sigma_i^z\sigma_j^z - \sum_i h_i\sigma_i^z, \tag{8}$$

where σ_i^z is the Pauli-Z operator acting on the ith 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 (ref. 36). 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, the 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 ith qubit. Then, the QAOA applies alternating simulation of the two non-commuting Hamiltonians H_X, H_C, resulting in the state

$$|\psi(\beta,\gamma)\rangle = \prod_{k=1}^{p} e^{-i\beta_k H_X} e^{-i\gamma_k H_C}|+\rangle, \tag{9}$$

with parameters $\beta, \gamma \in \mathbb{R}^p$, and number of repetitions $p \in \mathbb{N}$. As p approaches infinity, the QAOA is guaranteed to converge to the ground state of the problem Hamiltonian[34], assuming that β and γ are set appropriately. For finite p, carefully setting 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 the QAOA, such as recursive QAOA[37] or warm-started QAOA[38,39], 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[40]. However, there exist multiple alternative strategies to find good parameters for the QAOA and circumvent the trainability challenge. For example, as proposed by Streif and Leib[41], 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 substantially 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[42–46], by applying linear parameter schedules[47] or by using machine learning models[48,49]. Another promising strategy is to reuse optimized parameters from similar problem instances[50–52]. This is often referred to as parameter transfer or parameter concentration, and is discussed in more detail in the following.

Parameter transfer in a QAOA is a method to reuse the parameters obtained for one problem instance for another one without the need to repeat the classical parameter optimization. Optimized QAOA parameters for one problem instance often result in high-quality solutions for another instance, if the two problem instances exhibit similar properties. This has been proven analytically on low-depth MAXCUT problems on large 3-regular graphs[53] and observed empirically on a QAOA with $p = 1$ for MAXCUT problems on random regular graphs with the same parity[54]. Strategies to rescale the unweighted MAXCUT parameters for weighted problems exist to enable parameter transfer[55,56]. Several studies show parameter concentration of similar problem instances with respect to problem size on random Hamiltonians empirically and analytically[10,11,41,52,57]. In general, QAOA parameter transfer is a good heuristic tool to efficiently find QAOA angles that 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: for example, 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 by Katial et al.[58]. In the following, we show how to leverage the QAOA and parameter transfer in the context of weighted MAXCUT MOO on specific quantum computer hardware graphs.

QAOA for MOO

There exists limited literature on quantum algorithms for MOO. Chiew et al.[59] propose to use a 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 the QAOA for every c vector independently: Chiew et al. propose to re-optimize the QAOA parameters (β, γ) for every c, which is computationally expensive and leads to a bottleneck. Simlarly, Dahi et al.[60] suggest re-optimizing the QAOA parameters. However, those authors suggest warm-starting 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, Ekstrøm et al.[61] 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 we try 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 a QAOA and leads to deeper circuits. Finally, Li et al.[62] introduce an algorithm based on Grover's search[63] 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 the approaches proposed by Chiew et al.[59] and Dahi et al.[60] and completely skip the optimization of the QAOA parameters for the considered problem instance. More precisely, to overcome the aforementioned computational bottleneck introduced by re-optimizing for every c vector, we leverage the transfer of parameters discussed in 'Quantum approximate optimization algorithm' 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, that is, 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, that is, 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 the QAOA with fixed parameters gives a diverse set of good solutions for all c vectors, although not necessarily the optimal solutions. However, a suboptimal 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 'Quantum approximate optimization algorithm', 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 reuse 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 a QAOA to sample suboptimal solutions to the underlying diagonal Hamiltonian cost function with the goal of sampling the Pareto-optimal front is notable for several reasons. First, it means that we need both high-diversity solution sampling of configurations that share similar cost values—that is, fair sampling[64,65]—and high-variance sampling, which specifically means we do not use QAOA circuits converged to the ground state (that is, 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. Finally, there are several no-go theorems on the QAOA, usually providing lower bounds on p with respect to the problem size to achieve on average a certain solution quality[37,66]. 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.

Data availability

All the data used in this study are available at the Zenodo repository[67]. Source data are provided with this paper.

Code availability

Code to generate and execute all quantum circuits, to generate all data and to create all figures and tables presented in this manuscript is available at the Zenodo repository[67] and https://github.com/stefan-woerner/qamoo/.

References

1. Abbas, A. et al. Challenges and opportunities in quantum optimization. *Nat. Rev. Phys.* **6**, 718–735 (2024).
2. Ehrgott, M. *Multicriteria Optimization* 2nd edn (Springer, 2005).
3. Figueira, J. et al. Easy to say they are hard, but hard to see they are easy—towards a categorization of tractable multiobjective combinatorial optimization problems. *J. Multi-Criteria Decis. Anal.* **24**, 82–98 (2016).
4. Allmendinger, R., Jaszkiewicz, A., Liefooghe, A. & Tammer, C. What if we increase the number of objectives? Theoretical and empirical implications for many-objective combinatorial optimization. *Comput. Oper. Res.* **145**, 105857 (2022).
5. Bhattacharyya, B., Capriotti, M. & Tate, R. Solving general QUBOs with warm-start QAOA via a reduction to Max-Cut. Preprint at http://arxiv.org/abs/2504.06253 (2025).
6. Chamberland, C., Zhu, G., Yoder, T. J., Hertzberg, J. B. & Cross, A. W. Topological and subsystem codes on low-degree graphs with flag qubits. *Phys. Rev.* X **10**, 011022 (2020).

7. *IBM Quantum Platform* https://quantum.ibm.com (IBM Quantum, 2025).
8. Romero, S. V., Visuri, A.-M., Cadavid, A. G., Solano, E. & Hegade, N. N. Bias-field digitized counterdiabatic quantum algorithm for higher-order binary optimization. *Commun. Phys.* https://doi.org/10.1038/s42005-025-02270-3 (2025).
9. Kim, Y. et al. Evidence for the utility of quantum computing before fault tolerance. *Nature* **618**, 500–505 (2023).
10. Pelofske, E., Bärtschi, A. & Eidenbenz, S. Short-depth QAOA circuits and quantum annealing on higher-order Ising models. *npj Quantum Inf.* **10**, 30 (2024).
11. Pelofske, E., Bärtschi, A., Cincio, L., Golden, J. & Eidenbenz, S. Scaling whole-chip QAOA for higher-order Ising spin glass models on heavy-hex graphs. *npj Quantum Inf.* **10**, 109 (2024).
12. Pelofske, E., Bärtschi, A. & and Eidenbenz, S. Quantum annealing vs. QAOA: 127 qubit higher-order Ising problems on NISQ computers. In *High Performance Computing. ISC High Performance 2023* Lecture Notes in Computer Science Vol. 13948 (eds Bhatele, A. et al.) 240–258 (Springer, 2023).
13. Golden, J., Baertschi, A., O'Malley, D., Pelofske, E. & and Eidenbenz, S. JuliQAOA: fast, flexible QAOA simulation. In *Proc. SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis* 1454–1459 (ACM, 2023).
14. Bezanson, J., Edelman, A., Karpinski, S. & Shah, V. B. Julia: a fresh approach to numerical computing. *SIAM Rev.* **59**, 65–98 (2017).
15. Dächert, K., Fleuren, T. & Klamroth, K. A simple, efficient and versatile objective space algorithm for multiobjective integer programming. *Math. Methods Oper. Res.* **100**, 351–384 (2024).
16. IBM Quantum Qiskit Aer. *GitHub* https://github.com/Qiskit/qiskit-aer (2025).
17. Javadi-Abhari, A. et al. Quantum computing with Qiskit. Preprint at http://arxiv.org/abs/2405.08810 (2024).
18. Santra, G. C., Roy, S. S., Egger, D. J. & Hauke, P. Genuine multipartite entanglement in quantum optimization. *Phys. Rev. A* **111**, 022434 (2025).
19. Barron, S. V. et al. Provable bounds for noise-free expectation values computed from noisy samples. *Nat. Comput. Sci.* **4**, 865–875 (2024).
20. *IBM Quantum Roadmap* (IBM, accessed 19 April 2025); https://www.ibm.com/roadmaps/quantum.pdf
21. Riquelme, N., Von Lücken, C. & Baran, B. Performance metrics in multi-objective optimization. In *2015 Latin American Computing Conference (CLEI)* 1–11 (IEEE, 2015).
22. Fonseca, C. M., Paquete, L. & Lopez-Ibanez, M. An improved dimension-sweep algorithm for the hypervolume indicator. In *2006 IEEE International Conference on Evolutionary Computation* 1157–1163 (IEEE, 2006).
23. Bringmann, K. & Friedrich, T. Approximating the volume of unions and intersections of high-dimensional geometric objects. *Comput. Geom.* **43**, 601–610 (2010).
24. Deng, J. & Zhang, Q. Combining simple and adaptive Monte Carlo methods for approximating hypervolume. *IEEE Trans. Evol. Comput.* **24**, 896–907 (2020).
25. IBM ILOG CPLEX Optimization Studio v.22.1 (IBM, 2024).
26. *Gurobi Optimizer Reference Manual* (Gurobi Optimization, 2024).
27. Goemans, M. X. & Williamson, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM* **42**, 1115–1145 (1995).
28. Kirlik, G. & Sayın, S. A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. *Eur. J. Oper. Res.* **232**, 479–488 (2014).
29. Boland, N., Charkhgard, H. & Savelsbergh, M. A new method for optimizing a linear function over the efficient set of a multiobjective integer program. *Eur. J. Oper. Res.* **260**, 904–919 (2017).
30. Ozlen, M., Burton, B. A. & MacRae, C. A. G. Multi-objective integer programming: an improved recursive algorithm. *J. Optim. Theory Appl.* **160**, 470–482 (2014).
31. Bauß, J., Parragh, S. N. & Stiglmayr, M. Adaptive improvements of multi-objective branch and bound. Preprint at http://arxiv.org/abs/2312.12192 (2023).
32. Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**, 182–197 (2002).
33. Blank, J. & Deb, K. pymoo: multi-objective optimization in Python. *IEEE Access* **8**, 89497–89509 (2020).
34. Farhi, E., Goldstone, J. & Gutmann, S. A quantum approximate optimization algorithm. Preprint at http://arxiv.org/abs/1411.4028 (2014).
35. Farhi, E., Goldstone, J. & Gutmann, S. A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem. Preprint at http://arxiv.org/abs/1412.6062 (2015).
36. Lucas, A. Ising formulations of many np problems. *Front. Phys.* **2**, 5 (2014).
37. Bravyi, S., Kliesch, A., Koenig, R. & Tang, E. Obstacles to variational quantum optimization from symmetry protection. *Phys. Rev. Lett.* **125**, 260505 (2020).
38. Egger, D. J., Mareček, J. & Woerner, S. Warm-starting quantum optimization. *Quantum* **5**, 479 (2021).
39. Tate, R., Farhadi, M., Herold, C., Mohler, G. & Gupta, S. Bridging classical and quantum with SDP initialized warm-starts for QAOA. *ACM Trans. Quantum Comput.* **4**, 9 (2023).
40. Rajakumar, J., Golden, J., Bärtschi, A. & Eidenbenz, S. Trainability barriers in low-depth QAOA landscapes. In *Proc. 21st ACM International Conference on Computing Frontiers, CF '24* 199–206 (Association for Computing Machinery, 2024).
41. Streif, M. & Leib, M. Training the quantum approximate optimization algorithm without access to a quantum processing unit. *Quantum Sci. Technol.* **5**, 034008 (2020).
42. Farhi, E., Goldstone, J., Gutmann, S. & Zhou, L. The quantum approximate optimization algorithm and the Sherrington–Kirkpatrick model at infinite size. *Quantum* **6**, 759 (2022).
43. Boulebnane, S. & Montanaro, A. Predicting parameters for the quantum approximate optimization algorithm for MAX-CUT from the infinite-size limit. Preprint at http://arxiv.org/abs/2110.10685 (2021).
44. Basso, J., Farhi, E., Marwaha, K., Villalonga, B. & Zhou, L. 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 (eds Le Gall, F. & Morimae, T.) 7:1–7:21 (Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2022).
45. Wurtz, J. & Love, P. Maxcut quantum approximate optimization algorithm performance guarantees for $p > 1$. *Phys. Rev. A* **103**, 042612 (2021).
46. Wurtz, J. & Lykov, D. Fixed-angle conjectures for the quantum approximate optimization algorithm on regular maxcut graphs. *Phys. Rev. A* **104**, 052419 (2021).
47. Montañez-Barrera, J. A. & Michielsen, K. Toward a linear-ramp QAOA protocol: evidence of a scaling advantage in solving some combinatorial optimization problems. *npj Quantum Inf.* **11**, 131 (2025).
48. Khairy, S., Shaydulin, R., Cincio, L., Alexeev, Y. & Balaprakash, P. Learning to optimize variational quantum circuits to solve combinatorial problems. *Proc. AAAI Conference on Artificial Intelligence* **34**, 2367–2375 (2020).

49. Alam, M., Ash-Saki, A. & Ghosh, S. Accelerating quantum approximate optimization algorithm using machine learning. In *Proc. 23rd Conference on Design, Automation and Test in Europe, DATE '20* 686–689 (EDA Consortium, 2020).
50. Montañez-Barrera, J. A., Willsch, D. & Michielsen, K. Transfer learning of optimal QAOA parameters in combinatorial optimization. *Quantum Inf. Process.* **24**, 129 (2025).
51. Galda, A., Liu, X., Lykov, D., Alexeev, Y. & Safro, I. Transferability of optimal QAOA parameters between random graphs. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)* 171–180 (IEEE, 2021).
52. Augustino, B. et al. Strategies for running the QAOA at hundreds of qubits. Preprint at http://arxiv.org/abs/2410.03015 (2024).
53. Brandao, F. G. S. L., Broughton, M., Farhi, E., Gutmann, S. & Neven, H. For fixed control parameters the quantum approximate optimization algorithm's objective function value concentrates for typical instances. Preprint at http://arxiv.org/abs/1812.04170 (2018).
54. Galda, A. et al. Similarity-based parameter transferability in the quantum approximate optimization algorithm. *Front. Quantum Sci. Technol.* **2**, 1200975 (2023).
55. Sureshbabu, S. H. et al. Parameter setting in quantum approximate optimization of weighted problems. *Quantum* **8**, 1231 (2024).
56. Shaydulin, R., Lotshaw, P. C., Larson, J., Ostrowski, J. & Humble, T. S. Parameter transfer for quantum approximate optimization of weighted maxcut. *ACM Trans. Quantum Comput.* **4**, 19 (2023).
57. Akshay, V., Rabinovich, D., Campos, E. & Biamonte, J. Parameter concentrations in quantum approximate optimization. *Phys. Rev. A* **104**, L010401 (2021).
58. Katial, V., Smith-Miles, K., Hill, C. & Hollenberg, L. On the instance dependence of parameter initialization for the quantum approximate optimization algorithm: insights via instance space analysis. *INFORMS J. Comput.* **37**, 146–171 (2024).
59. Chiew, S.-H. et al. Multiobjective optimization and network routing with near-term quantum computers. *IEEE Trans. Quantum Eng.* **5**, 3101419 (2024).
60. Dahi, Z. A., Chicano, F., Luque, G., Derbel, B. & and Alba, E. Scalable quantum approximate optimiser for pseudo-boolean multi-objective optimisation. In *Parallel Problem Solving from Nature—PPSN XVIII: 18th International Conference, PPSN 2024* Lecture Notes in Computer Science Vol. 15151 (eds Affenzeller, M., et al.) 268–284 (Springer, 2024).
61. Ekstrøm, L., Wang, H. & Schmitt, S. Variational quantum multiobjective optimization. *Phys. Rev. Res.* **7**, 023141 (2025).
62. Li, H., Qiu, D. & Luo, L. Distributed exact multi-objective quantum search algorithm. Preprint at http://arxiv.org/abs/2409.04039 (2024).
63. Grover, L. K. A fast quantum mechanical algorithm for database search. Preprint at http://arxiv.org/abs/quant-ph/9605043 (1996).
64. Golden, J., Bärtschi, A., O'Malley, D. & Eidenbenz, S. Fair sampling error analysis on NISQ devices. *ACM Trans. Quantum Comput.* **3**, 8 (2022).
65. Pelofske, E., Golden, J., Bartschi, A., O'Malley, D. & Eidenbenz, S. Sampling on NISQ devices: "who's the fairest one of all?". In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)* 207–217 (IEEE, 2021).
66. Farhi, E., Gamarnik, D. & Gutmann, S. The quantum approximate optimization algorithm needs to see the whole graph: a typical case. Preprint at http://arxiv.org/abs/2004.09002 (2020).
67. Woerner, S. qamoo. *Zenodo* https://doi.org/10.5281/zenodo.16878920 (2025).

Acknowledgements
A.K. was employed at IBM Quantum and Zuse Institute Berlin at the time of the research and is now at planqc GmbH. The work of A.K. and S.R. for this article at Zuse Institute Berlin in the Research Campus MODAL was funded by the Federal Ministry of Research, Technology and Space (BMFTR) (funds 05M14ZAM, 05M20ZBM, 05M2025). E.P. and S.E. were supported by the US Department of Energy through the Los Alamos National Laboratory and the NNSA's Advanced Simulation and Computing Beyond Moore's Law Program at Los Alamos National Laboratory. Los Alamos National Laboratory is operated by Triad National Security, LLC, for the National Nuclear Security Administration of the US Department of Energy (contract 89233218CNA000001). LANL report number LA-UR-25-22069. D.J.E. acknowledges funding within the HPQC project by the Austrian Research Promotion Agency (FFG, project number 897481) supported by the European Union—NextGenerationEU. The funders had no role in study design, data collection and analysis, decision to publish or preparation of the manuscript.

Author contributions
All authors contributed to the discussions of the theory and results and to writing the manuscript. A.K., D.J.E. and S.W. developed the code and optimized circuits for execution on real hardware. A.K. and S.W. conducted the quantum experiments and corresponding classical simulations. E.P. and S.E. trained the QAOA parameters and analyzed parameter transfer. S.R. and T.K. planned the classical algorithm tests and S.R. implemented and ran them. T.K. and S.W. conceived the idea, and S.W. coordinated the project.

Competing interests
The authors declare no competing interests.

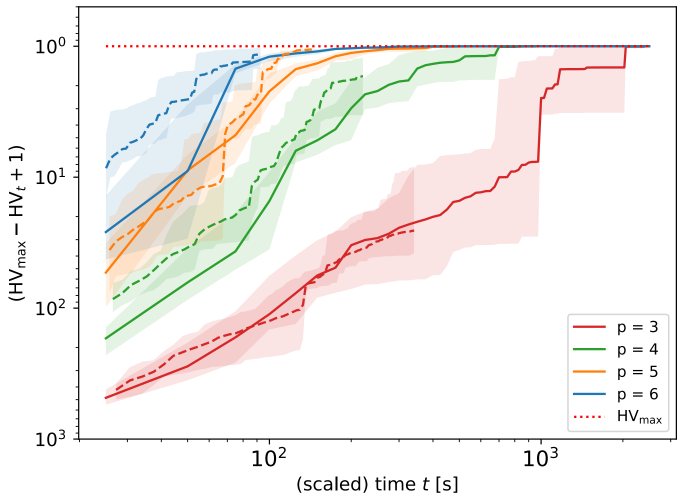
Additional information
Extended data is available for this paper at https://doi.org/10.1038/s43588-025-00873-y.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s43588-025-00873-y.

Correspondence and requests for materials should be addressed to Stefan Woerner.

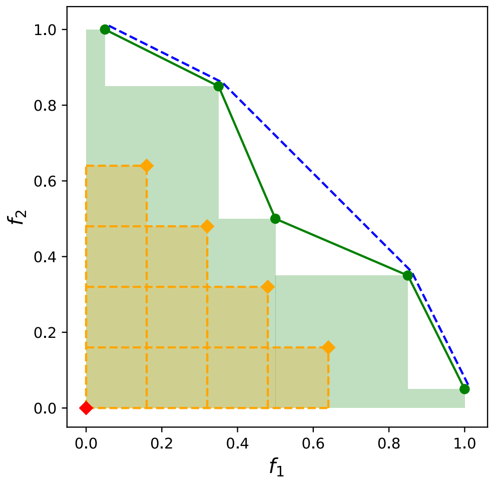
Peer review information *Nature Computational Science* thanks Mile Gu and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. Primary Handling Editor: Jie Pan, in collaboration with the *Nature Computational Science* team. Peer reviewer reports are available.

https://doi.org/10.1038/s43588-025-00873-y



Extended Data Fig. 1 | Fidelity-scaled quantum results. MPS simulation and *ibm_fez* results from Fig. 2 where the time of the hardware results is scaled by the corresponding estimated fidelity reported in Table 1. The (dashed) lines represent the average and the shaded areas the minimum and maximum over the five repetitions.

https://doi.org/10.1038/s43588-025-00873-y



Extended Data Fig. 2 | Pareto front & HV for two objective functions. 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.