Scaling of the quantum approximate optimization algorithm on superconducting qubit based hardware

Johannes Weidenfeller[1,2], Lucia C. Valor[1], Julien Gacon[1,3], Caroline Tornow[1,2], Luciano Bello[1], Stefan Woerner[1], and Daniel J. Egger[1]

[1]IBM Quantum, IBM Research Europe – Zurich
[2]ETH Zurich
[3]Institute of Physics, Ecole Polytechnique Fédérale de Lausanne (EPFL)

arXiv:2202.03459v2 [quant-ph] 1 Dec 2022

Quantum computers may provide good solutions to combinatorial optimization problems by leveraging the Quantum Approximate Optimization Algorithm (QAOA). The QAOA is often presented as an algorithm for noisy hardware. However, hardware constraints limit its applicability to problem instances that closely match the connectivity of the qubits. Furthermore, the QAOA must outpace classical solvers. Here, we investigate swap strategies to map dense problems into linear, grid and heavy-hex coupling maps. A line-based swap strategy works best for linear and two-dimensional grid coupling maps. Heavy-hex coupling maps require an adaptation of the line swap strategy. By contrast, three-dimensional grid coupling maps benefit from a different swap strategy. Using known entropic arguments we find that the required gate fidelity for dense problems lies deep below the fault-tolerant threshold. We also provide a methodology to reason about the execution-time of QAOA. Finally, we present a QAOA Qiskit Runtime program and execute the closed-loop optimization on cloud-based quantum computers with transpiler settings optimized for QAOA. This work highlights some obstacles to improve to make QAOA competitive, such as gate fidelity, gate speed, and the large number of shots needed. The Qiskit Runtime program gives us a tool to investigate such issues at scale on noisy superconducting qubit hardware.

1 Introduction

Gate-based quantum computers process information by applying unitary operations on information stored in qubits. Such computers may provide an advantage for complex computational tasks in chemistry [1–3], finance [4–6] and combinatorial optimization [7, 8]. We focus on the Quantum Approximate Optimization Algorithm (QAOA) [7–9] which maps combinatorial optimization problems, for instance, quadratic unconstrained binary optimization (QUBO) problems with n variables

$$\min_{x\in\{0,1\}^n} x^T\Sigma x,\ \Sigma\in\mathbb{R}^{n\times n}, \tag{1}$$

to the problem of finding the ground state of an Ising Hamiltonian, H_C [7]. Here, H_C is constructed by mapping each of the n decision variables to a qubit by the relation $x_i = (1-z_i)/2$ and replacing z_i by a Pauli spin operator Z_i to obtain H_C [10, 11]. Two qubits i,j thus only interact through Z_iZ_j if the corresponding quadratic term $\Sigma_{i,j}$ is not zero. The QAOA first creates an initial state which is the ground state of a mixer Hamiltonian H_M. A common choice of H_M and initial state is $-\sum_{i=0}^{n-1} X_i$ and $|+\rangle^{\otimes n}$ which is easy to prepare. Here, X_i are Pauli X operators. Next, a depth-p QAOA circuit creates the trial state $|\psi(\boldsymbol{\beta},\boldsymbol{\gamma})\rangle$ for vectors $\boldsymbol{\gamma},\boldsymbol{\beta}\in\mathbb{R}^p$ by applying $\exp(-i\beta_k H_M)\exp(-i\gamma_k H_C)$ at each layer $k=1,...,p$, implemented by $R_X(\beta)=\exp(-i\beta X/2)$ and $R_{ZZ}(\gamma)=\exp(-i\gamma ZZ/2)$ gates. A classical optimizer seeks the optimal values of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ to create a trial state which minimizes the energy of H_C. The potential for a quantum advantage of QAOA and its variants over highly-optimized classical solvers, such as CPLEX [12], must be explored empirically. Such benchmarks must be two dimensional where both the quality and the time to reach the proposed solution matter [13–16].

Business-relevant problems often require budget or capacity constraints, and thus, Σ tends to be dense [17] and the corresponding interaction graph non-planar [18]. Implementing such problems on superconducting qubit [19, 20] platforms is hindered by the limited qubit connectivity, expressed by the coupling map, and therefore requires SWAP gates [14]. By contrast, cold-atomic architectures based on the Rydberg blockade [21] and trapped ions [22] may overcome this issue [23] but in turn suffer from low repetition rates which limits the speed at which shots are gathered [24].

In this work we discuss all aspects relevant to scaling QAOA on superconducting qubits. First, in Sec. 2, we discuss strategies to map dense problems

Daniel J. Egger: deg@zurich.ibm.com

into linear, grid [14] and heavy-hex [25] coupling maps. Next, in Sec. 3, we estimate the quantum hardware requirements needed to solve such problems. Here, we estimate in Sec 3.1 gate fidelity requirements for problems of varying density and present a methodology to reason about the run-time of QAOA in Sec. 3.2. Indeed, QAOA can only provide an advantage by yielding better solutions than classical optimizers or comparable solutions in a shorter time. In Sec. 4 we present a QAOA Qiskit Runtime program to explore the QAOA scaling on noisy hardware. Finally, we discuss these results and conclude in Sec. 5.

2 Effects of limited device connectivity

The qubit connectivity in, e.g., superconducting qubit devices, is limited by engineering constraints and the need to avoid unwanted effects like cross-talk [26–28]. Typically, qubits are arranged in a planar graph, called the *coupling map* and two-qubit gates can only be applied to adjacent qubits. Therefore, additional SWAP gates are inserted into the circuits to make them hardware-compatible, a task known as qubit routing [29, 30]. The number of gates after transpilation to a hardware device thus depends on the problem, the coupling map, and the routing algorithm. We propose a set of hardware dependent routing algorithms that perform particularly well on dense circuits of commuting operators, such as the cost operator in a QAOA on a complete graph. We investigate the resulting circuit depth and gate count for linear, grid [14], and heavy-hex [25] coupling maps.

2.1 Hardware-optimized transpiler pass

Swap transpiler passes typically divide quantum circuits into layers of simultaneously executable gates on the coupling map [29]. They transition between the qubit mappings of different layers, i.e., a positioning of logical to physical qubits, by inserting SWAP gates consistent with the coupling map. Here, a logical qubit is a qubit in an algorithm and a physical qubit is a hardware qubit such as a transmon [31]. Mapping circuits to hardware is a hard optimization problem with combinatorial scaling, even on grid coupling maps [32]. A variety of heuristic algorithms have therefore been introduced [33] and different coupling maps studied [34, 35]. Application-specific transpilers leverage the mathematical properties of the application to reduce circuit depth. For example, the 2QAN transpiler exploits the flexibility of permuting Trotter operators in two-local Hamiltonians [36]. Swap synthesis is simpler when the considered gates commute, as in the cost layer of QAOA [37]. Here, various strategies have been developed such as applying gates according to some ranking or stitching layers of R_{ZZ} gates together [38]. Transpiler passes that do not consider commutativity yield sub-optimal gate counts for circuits with a high number of commuting gates.

We develop a transpiler pass that first identifies the subset of all device qubits to run on and then applies a corresponding *swap strategy*. The swap strategy exploits gate commutativity by reordering commuting gates and inserting layers of SWAP gates from a set of predefined *swap layers* $\mathcal{S} = \{S_0, ..., S_K\}$. Throughout this work a *layer* is a set of simultaneously executable gates on the coupling map and has thus depth one. Therefore, for a given coupling map, a swap strategy is a series of swap layers $S_{k_1}, S_{k_2}, ...$ of length L_S applied in a predefined order and chosen from $\mathcal{S}$, i.e. $k_i \in \{0, ..., K\}$. A swap strategy applies the following steps:

1. Split the circuit into sequential sets of commuting gates $\mathcal{T}_1, \mathcal{T}_2, ...$, and choose the first one, i.e. $i = 1$, as the current set.

2. Repeat the following steps (a) to (d) until all gates in the current set $\mathcal{T}_i$ are applied, see Fig. 1. Set $j = 1$, and

   (a) select all remaining gates $E_j \subseteq \mathcal{T}_i$ from the current set that are executable given the current qubit mapping and remove them from $\mathcal{T}_i$.

   (b) Partition the selected gates E_j into subsets of simultaneously executable gates $\mathcal{G}_1, \mathcal{G}_2, ...$ either by sorting them according to a provided edge coloring of the coupling map or by greedily building the sets.

   (c) Iterate through the subsets $\mathcal{G}_1, \mathcal{G}_2, ...$, e.g., in decreasing set size, to simultaneously apply all gates in each set.

   (d) Apply a single swap layer $\mathcal{S}_{f(i,j)}$ to alter the current qubit mapping. Here, $f(i, j)$ is the order in which we apply the swap layers. Increment j and move to step (a) if $\mathcal{T}_i$ is not empty.

3. Remove superfluous SWAP gates at the end of the circuit and continue to Step 2 with the next set of commuting gates $\mathcal{T}_{i+1}$, or terminate if all gates are applied.

We call a swap strategy optimal if it leads to full connectivity with the least possible number of swap layers. Our task is thus to find a good set of swap layers $\mathcal{S}$, the order in which to apply them $f(i, j)$, and the initial qubit mapping for a given problem and coupling map. For example, in QAOA, the first set $\mathcal{T}_1$ creates the inital state which is trivial to apply as it is made of single-qubit gates. Next, $\mathcal{T}_2$ corresponds to the cost operator which requires SWAP gates. We note that for QAOA the order in which the gates are applied in Step 2(c) is chosen to leverage gate cancellations between R_{ZZ} gates in E_j and SWAP gates in $S_{f(i,j)}$. In sub-sections 2.2 and 2.3 we present the

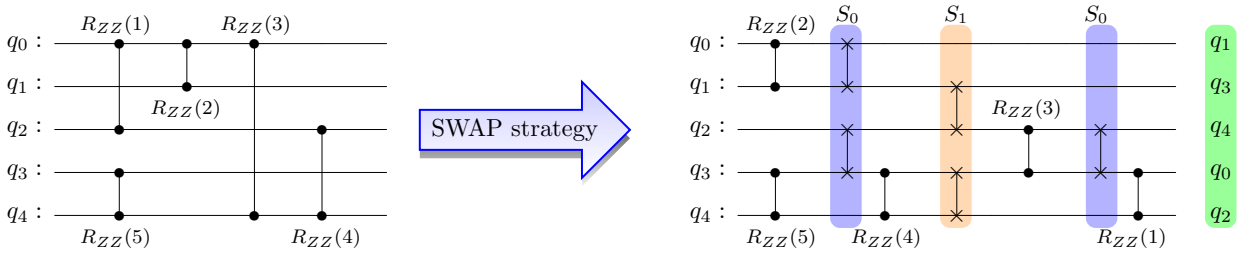


Figure 1: Transpilation of a five-qubit $\exp(-i\gamma H_C)$ circuit (left) to a line coupling map using $\mathcal{S} = \{S_0, S_1\}$. The swap layers alternate between the sets $S_0 = \{\mathrm{SWAP}_{0,1}, \mathrm{SWAP}_{2,3}\}$ and $S_1 = \{\mathrm{SWAP}_{1,2}, \mathrm{SWAP}_{3,4}\}$. In the transpiled circuit (right) a redundant $\mathrm{SWAP}_{0,1}$ gate is removed from the last layer. The resulting qubit mapping is highlighted in green.

swap strategies and the scaling of their gate count and depth with problem size and density, respectively.

2.2 Swap strategies

The number of swap layers L_S required to implement a quantum circuit U of commuting two-qubit gates under a fixed swap strategy $\mathcal{S}$ depends on its density D and structure. Here, D is the number of two-qubit terms normalized by its maximum possible number $n(n-1)/2$. We describe U with an interaction graph G_{int} where vertices correspond to qubits and edges to two-qubit gates. Let G_{L_S} be the graph of all possible qubit interactions implementable after L_S swap layers of $\mathcal{S}$. The circuit can be implemented with L_S swap layers of $\mathcal{S}$ if G_{int} can be embedded in G_{L_S}. The graphs G_{L_S} therefore describe the structure of a potential U that can be implemented after L_S swap layers. The density of G_{L_S} bounds the possible density of U from above. Vice versa, we obtain a lower bound $L_S(D)$ on the number of swap layers required to implement a given U with a density D using a particular swap strategy. Indeed, a G_{L_S} that achieves the same density as U does not necessarily have the required structure to implement U. We now discuss swap strategies that reach full connectivity, i.e. $D = 1$.

2.2.1 Linear coupling map

Transpiling arbitrary quantum circuits to a line coupling map has been studied in Ref. [39] and when all the gates commute it can also be done optimally [14, 37]. For a line coupling map with n qubits the swap strategy which alternates between two swap layers S_0 and S_1 which apply SWAP gates on all even and odd numbered edges [1], respectively, is provably optimal, see Appendix A. This strategy requires $L_S = n - 2$ swap layers and is illustrated in Figs. 1 and 2. For this strategy the minimum number of swap layers needed to reach a density D is $L_S(D) = (n-2)D$.

[1] We call an edge (i, j) of a line graph even if $i = 0 \mod 2$ and odd otherwise.

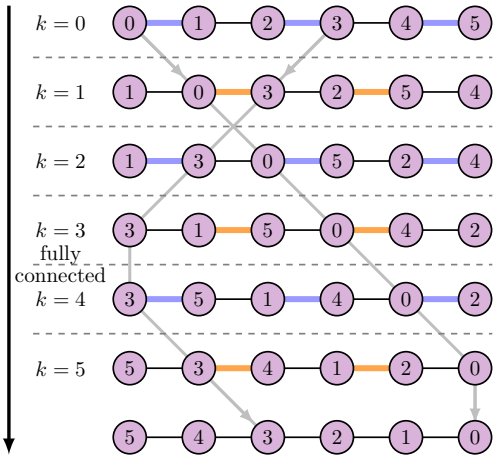


Figure 2: Optimal line swap strategy for $n = 6$. Swap layers are alternately applied to even (in blue) and odd (in orange) numbered edges. The numbers and gray lines show the positions of logical qubits as they move through the line. Full connectivity is reached after $k = n - 2 = 4$ layers and the line is fully reversed after $k = n = 6$ layers.

2.2.2 Grid coupling map

We adapt the line graph strategy to the two- and three-dimensional nearest-neighbour grid coupling maps to create strategies that reach full connectivity after $n/2 + \mathcal{O}(\sqrt{n})$ and $n/4 + \mathcal{O}(n^{2/3})$ swap layers, respectively. For the two dimensional case we consider square grids with x rows and columns, i.e. $n = x^2$. The swap strategy has four layers $S_0, ..., S_3$ and repeats two steps until full connectivity is reached:

1. Apply $x - 1$ steps of the line swap strategy to each row. Importantly, in the same swap layer, the SWAP gates in one of two neighboring rows are applied on even edges while in the other row they are applied on odd edges, see S_0 and S_1 in Fig. 3.

2. Swap rows by applying two steps of the line swap strategy to each column in parallel, see S_2 and S_3 in Fig. 3. Applying S_2 and S_3 directly after each other guarantees that each row has, up to edge effects, two new neighboring rows, one above and one below itself.

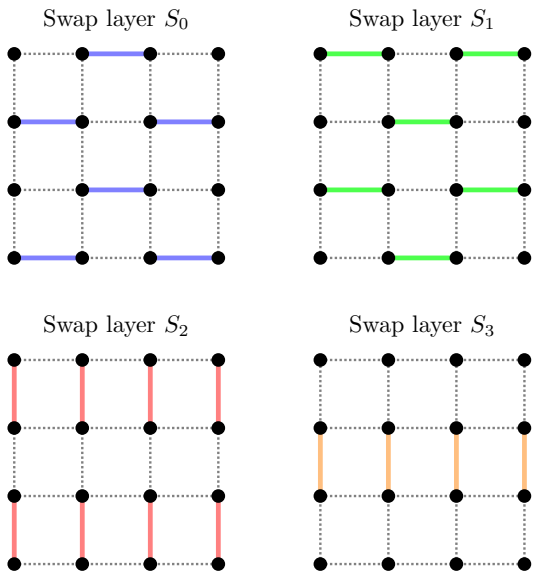


Figure 3: Swap strategy for a 4×4 grid. The black dots are physical qubits and dashed lines indicate the coupling map. The solid colored edges indicate the swap layer of the swap strategy $\mathcal{S} = \{S_0, S_1, S_2, S_3\}$. S_0 and S_1 are repeatedly applied to reach full connectivity in each row and between neighbouring rows (step 1). Next, swap layers S_2 and S_3 are each applied once such that each row becomes adjacent to two different rows (step 2).

Step 1 creates full connectivity within each row and between adjacent rows. Note that while $x-2$ layers are sufficient to reach full connectivity in each row, see Sec. 2.2.1, we need $x-1$ swap layers to connect all qubits of adjacent rows. Step 2 swaps rows such that all rows are adjacent to one another at some point in the swap process. Each iteration of steps 1 and 2 requires $x+1$ swap layers. After repeating both steps $\lceil \frac{x}{2} \rceil$ times, full connectivity is reached. In total, the number of swap layers is

$$\left\lceil \frac{x}{2} \right\rceil (x+1) \leq \frac{n}{2} + \sqrt{n} + \frac{1}{2}.$$

This strategy generalizes to grids of higher dimension: for η-dimensional grid coupling maps a problem density D requires at least $L_S(D) = nD/2^{\eta-1} + \mathcal{O}(n^{1-1/\eta})$ swap layers. Details are in Appendix A.

2.2.3 Heavy-hex coupling map

IBM Quantum systems have a heavy-hex coupling map, i.e., a regular hexagonal lattice with additional nodes inserted on each edge [25]. A simple strategy applies the optimal line graph strategy to the longest continuous line embedded in the heavy-hex graph. This strategy cannot reach full connectivity since a single continuous line does not include all qubits. However, a swap strategy that applies the line strategy to the longest line in the heavy-hex graph and periodically swaps qubits positioned in the line with qubits that are not part of the line reaches full

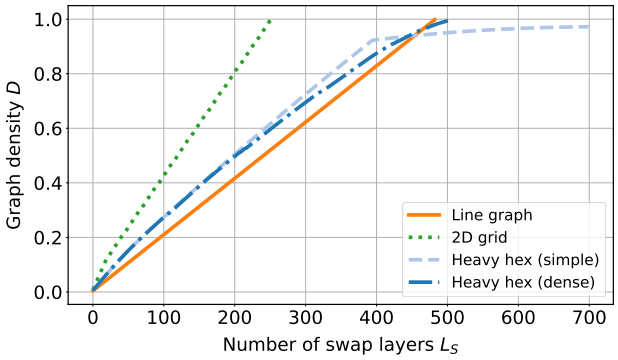


Figure 4: Maximal graph density as a function of the number of swap layers L_S for the line graph with 485 qubits (solid orange line), the 2D grid with $22 \times 22 = 484$ qubits (dotted green line), and the heavy-hex grid with 485 qubits (blue lines). Here, we chose 484 and 485 qubits so that the different coupling maps have a similar qubit count. The "heavy-hex (simple)" uses the line strategy on the longest line in the heavy-hex graph, and thus, does not reach full connectivity.

connectivity after at most

$$n + \sqrt{n} + 61 = n + \mathcal{O}(\sqrt{n})$$

swap layers. Details and a proof are given in Appendix A. A lower bound on the number of swap layers to implement a problem with density D is almost linear and we approximate it by nD, see Fig. 4 and Tab. 1.

2.3 Circuit depth and gate count

We investigate how the gate count and depth of QAOA circuits scale with the swap strategies described in Sec. 2.2. QAOA circuits transpiled with the swap strategies of Sec. 2.2 consist of alternating layers E_i and S_i, containing only R_{ZZ} and SWAP gates, respectively. The total number of CNOT layers L_{cx} and gates is therefore directly related to the required number of swap layers L_S. By definition, every swap layer S_i consists of SWAP gates simultaneously executable on the underlying coupling map. On cross-resonance based hardware a SWAP gate is executed by three CNOT gates. The number of CNOT layers in E_i depends on the coupling map, in particular its chromatic edge number, and the swap strategy. The total number of CNOT gates as a function of L_S is bounded from above by $(4\eta-1)nL_S/2$ for η-dimensional grids and $8nL_S/5$ for heavy-hex coupling maps. These formulas, derived in Appendix B, account for possible gate cancellations between subsequent layers E_i and S_i. We therefore need $\mathcal{O}(Dn^2)$ CNOT gates per application of $\exp(-i\gamma H_C)$ with prefactors, 1.5, 1.75, 1.375 and 1.6 for linear, 2D-grid, 3D-grid, and heavy-hex coupling maps, respectively, see Tab. 1. The two-dimensional grid strategy is thus worse than the line strategy for the same number of qubits while the

Coupling map	L_S	Number of CNOT layers L_{cx}	Total number of CNOT gates per $\exp(-i\gamma H_C)$ layer	Average number of CNOT gates per layer l_{cx}
Line	Dn	$3L_S = 3Dn$	$\frac{3}{2}nL_S = \frac{3}{2}Dn^2$	$n/2$
Grid	$\frac{1}{2}Dn$	$7L_S = \frac{7}{2}Dn$	$\frac{7}{2}nL_S = \frac{7}{4}Dn^2$	$n/2$
3D Grid	$\frac{1}{4}Dn$	$11L_S = \frac{11}{4}Dn$	$\frac{11}{2}nL_S = \frac{11}{8}Dn^2$	$11n/32$
Heavy-hex	Dn	$9L_S = 9Dn$	$\frac{8}{5}nL_S = \frac{8}{5}Dn^2$	$8n/45$

Table 1: Swap layer and gate count needed to run a depth-one QAOA circuit. Here, we only report the leading terms and omit the big-$\mathcal{O}$ for brevity. For $D < 1$ these numbers are lower bounds with $D = 1$ being the upper bound. L_S shows the lower bound of the number of swap layers as a function of decision variables n and graph density D. The derivation of the total number of CNOT gates for a single QAOA circuit is discussed in Appendix B.

three-dimensional grid is the best. This results from two competing scalings. On one hand, the number of swap layers required to reach full connectivity scales as $1/2^{\eta-1}$ with η, i.e., higher-dimensional grids require less swap layers. On the other hand, the number of CNOT gates per swap and R_{ZZ} layer combined scales as $4\eta - 1$, i.e., higher dimensional grids have less R_{ZZ} and CNOT cancellations. Therefore, the additional connectivity of the two-dimensional grid over the line is not useful for reaching full connectivity. On two-dimensional grids it is better to use a line swap strategy on the longest line in the grid instead of the gird swap strategy. By contrast, three-dimensional grids, such as cold atomic lattices [40], lead to a smaller gate count than a line and a two-dimensional grid.

We now benchmark the swap strategies of Sec. 2.2 to t|Ket⟩ [41], 2QAN [36], the QAOA Compiler [38], SabreSwap [42], and a commutation aware version of SabreSwap that we implemented and describe in Appendix C. We consider depth-one QAOA circuits of MaxCut [18, 43], formally introduced in Appendix D, for graphs chosen uniformly at random from the set of all graphs with n nodes and $\lceil D\frac{n(n-1)}{2} \rceil$ edges [2]. We compute the number of layers of simultaneous CNOT gates, and the CNOT gate count after each transpiler pass for heavy-hex coupling maps with a variable number of rows and columns, with $D \in \{0.25, 0.5, 0.75, 1\}$, details are in Appendix E. Commutative aware SabreSwap results in fewer CNOT gates but deeper circuits for problems with $D = 0.25$ and 0.5. 2QAN has the least number of CNOT gates but results in deeper circuits than the swap strategies. The swap strategies are clearly advantageous for dense graphs, see Fig. 5. This is expected since the swap strategies are tailored for dense problems and may perform unnecessary SWAP gates on qubits that do not need to be connected in incomplete graphs. For hardware subject to finite T_1 and T_2-times shallow circuits are advantageous as idling qubits accumulate errors. We find that the swap strategies result in the shallowest circuits as shown by the number of CNOT layers in Fig. 5. Additionally, the time taken to transpile with the swap strategies is significantly lower than the time needed by t|Ket⟩, 2QAN, the QAOA Compiler, and commutative aware SabreSwap and is comparable to SabreSwap, see Appendix E.

[2]The graphs are generated with the `gnm_random_graph` method from the networkx package.

3 Hardware requirements

We now discuss how gate fidelity and gate duration impact QAOA in Sec. 3.1 and Sec. 3.2, respectively.

3.1 Gate fidelity

The error-prone unitary gates limit performance. Entropic inequalities help bound the maximum circuit depth for QAOA [44, 45]. Following Proposition 2 of Ref. [45], the maximum depth of a QAOA circuit with a fraction f_1 of single-qubit gate layers and a fraction f_2 of two-qubit gate layers with depolarizing noise with probability p_1 and p_2, respectively, is bounded by

$$L_{\max} \approx \frac{\ln \epsilon^{-1}}{2(f_1 p_1 + f_2 p_2)}. \tag{2}$$

For circuits deeper than $L_{\max}$ there exists a polynomial time classical algorithm that finds a Gibbs state that we can classically sample from with the same energy up to an error $\epsilon||H_C||$ of the noisy QAOA state. Here, ϵ controls the precision with which we approximate the energy. Reference [45] argues that ϵ should range from 10^{-1} to 10^{-2} since most optimization algorithms require a number of shots with an inverse polynomial scaling in ϵ [46, 47]. This implies that going beyond an $\epsilon \sim 10^{-2}$ incurs a significant sampling cost. Since the CNOT gate is the dominant source of error and QAOA circuits for denser problems are dominated by two-qubit gate layers, we further simplify Eq. (2) to $L_{\max} \approx \ln(\epsilon^{-1})/2p_2$.

The CNOT fidelity $\mathcal{F}_{\text{cx}}$ quoted by IBM Quantum systems is the probability of a depolarizing channel since it is measured with randomized benchmarking [48–50]. Each CNOT gate layer in a QAOA circuit transpiled with the swap strategies presented in Sec. 2.2 will on average have l_{cx} CNOT gates, see

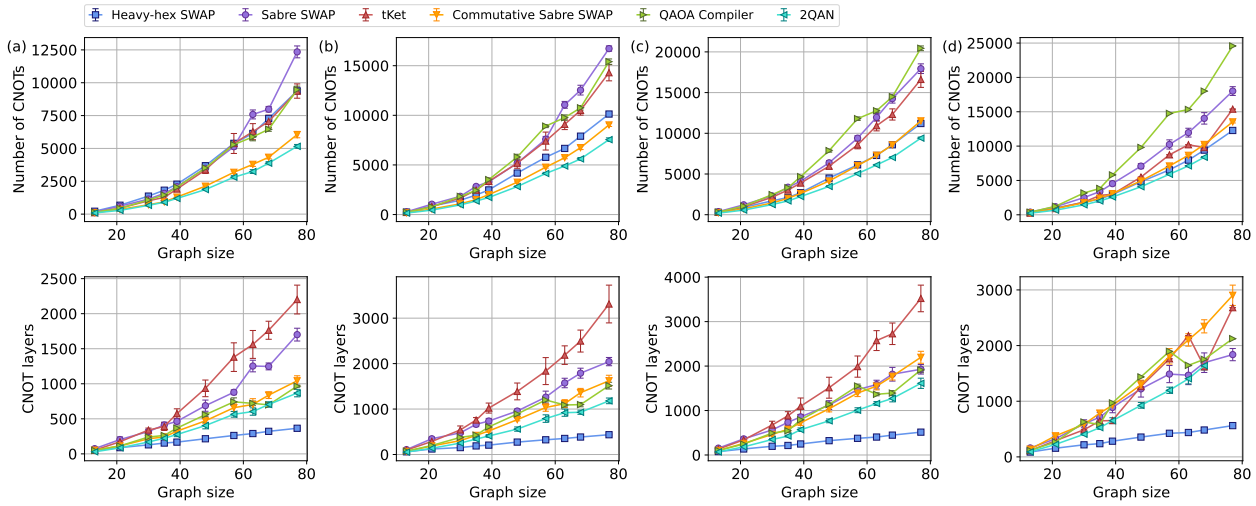


Figure 5: Swap strategies benchmarked on heavy-hex graphs defined by a number of rows and columns $(i, j) \in \{(1,1),(1,2),(1,3),(2,2),(1,4),(1,5),(1,6),(2,4),(3,3),(2,5)\}$. Subfigures (a), (b), (c), and (d) correspond to graph densities of 0.25, 0.5, 0.75 and 1.0, respectively. The top row shows the number of CNOTs and the bottom row shows the circuit depth measured as the number of CNOT layers.

Tab. 1. We make the simplifying assumption that the depolarizing probability of a layer of CNOT gates is $p_2 = 1 - \mathcal{F}_{\text{cx}}^{l_{\text{cx}}}$. This is an optimistic assumption since effects such as crosstalk may degrade the performance of gates applied in parallel [51, 52]. A QAOA with depth p using $L_{\text{cx}}(n, D)$ CNOT layers per application of $\exp(-i\gamma H_C)$ for a graph with n nodes and density D must satisfy

$$pL_{\text{cx}}(n, D) \leq \frac{\ln \epsilon^{-1}}{2\left(1 - \mathcal{F}_{\text{cx}}^{l_{\text{cx}}}\right)}, \tag{3}$$

otherwise there is a corresponding Gibbs state which can be sampled from classically in polynomial time [45]. Following Ref. [45] there is therefore little chance that running a quantum circuit that requires more CNOT layers than the bound in Eq. (3) will lead to a quantum advantage, i.e., we assume there is little hope for a quantum advantage if there exists a polynomial-time classical approximation.

We calculate the bound in Eq. (3) for a heavy-hex coupling map with 485 qubits as a function of D and $\mathcal{F}_{\text{cx}}$. The density-dependent upper bound on the gate error is one to three orders of magnitude lower than current hardware capabilities, see Fig. 6. The data indicate that non-hardware native optimization problems will require gate fidelities above error correction thresholds which typically range from 99% to 99.99% [25, 53, 54]. When such fidelities are reached it may still be advantageous to run QAOA on noisy hardware due to the large qubit overhead imposed by error correction and the potentially lower execution times.

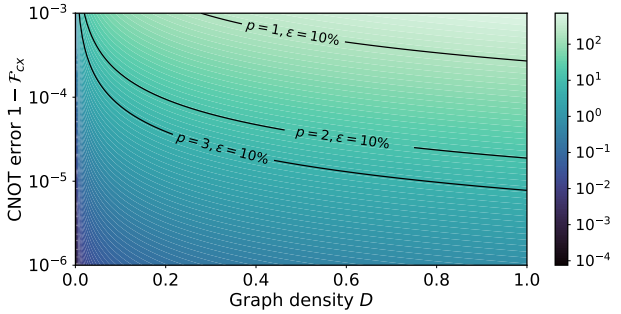


Figure 6: Gate depth criterion to implement QAOA on a heavy-hex graph. The color scale shows $18nD(1 - \mathcal{F}_{\text{cx}}^{l_{\text{cx}}})$ plotted on a logarithmic scale as a function of graph density D and $1 - \mathcal{F}_{\text{cx}}$ with $n = 485$ (as in Fig. 4) for the heavy-hex swap strategy which has an average of $l_{\text{cx}} = 8n/45$ CNOT gates per CNOT layer, see Tab. 1. The contour lines indicate $\ln(\epsilon^{-1})/p$ with $\epsilon = 10\%$ and different QAOA depths p.

3.2 Execution-time analysis

We now estimate the execution-time of QAOA as the product of the number of iterations of the classical solver N_{iter} times the time taken to gather the data at each iteration

$$\tau_{\text{QAOA}} = N_{\text{iter}} \left(N_{\text{shots}} \cdot \tau_{\text{shot}} + \tau_{\text{init}}\right). \tag{4}$$

Here, the number of measurements per iteration is N_{shots} and the duration of a single-shot $\tau_{\text{shot}} = \tau_{\text{circ}} + \tau_{\text{delay}}$. The time taken to run all the gates, measurement, and reset instruction is captured in τ_{circ} while τ_{delay} is a fix delay after each measurement used to improve the qubit reset [55, 56]. At each iteration the control hardware must be setup to gather the next shots and therefore incurs a time penalty τ_{init}. This decomposition is similar to the Circuit Layer Operations per Second (CLOPS) benchmark [55]. We esti-

	Current	Possible
$\sqrt{X}$ error	$7.2 \cdot 10^{-4}$	
CNOT error	$169.0 \cdot 10^{-4}$	
$\sqrt{X}$ duration	36 ns	4 ns [57]
CNOT duration	400 ns	30 ns [58]

Table 2: Performance metrics of quantum hardware. Current performance corresponds to typical values on IBM Quantum systems. Possible performance shows the gate duration that has been measured or could be implemented.

mate the execution-time τ_{QAOA} for different problem sizes n and on different coupling maps. We focus on problems with $D = 1$ since they upper bound the $D < 1$ instances. Substituting $D < 1$ in the following equations may underestimate the execution time depending on the graph structure.

3.2.1 Duration of a single-shot

On cross-resonance based hardware [59, 60] the duration of a single-shot is determined by the duration of the CNOT gate τ_{cx}, the QAOA depth p, the problem density D, and the coupling map as discussed in Sec. 2.2. Since QAOA at sub-logarithmic depth is not expected to outperform classical solvers [61–63], we assume that p scales at least logarithmically with the number of variables n. We therefore chose $p = \log_2(n)$ for our runtime analysis. With the number of CNOT layers $L_{\text{cx}}(n, D)$ we estimate that a single-shot lasts at least

$$\tau_{\text{shot}} \geq \log_2(n) L_{\text{cx}}(n, D) \tau_{\text{cx}} + \tau_{\text{delay}}. \tag{5}$$

Since $L_{\text{cx}}(n, D)$ scales as $\Omega(Dn)$ the duration of a single QAOA shot scales at least as fast as $\tau_{\text{shot}} = \Omega(Dn \log_2 n)$. With a 400 ns CNOT gate and a heavy-hex coupling map, i.e. $L_{\text{cx}} = 9nD$, the duration of a shot is significant, see Fig. 7 which only includes the CNOT gate time. Here, measurement and reset instructions, which can last up to a few microseconds [56, 64, 65] and typically only appear once in a quantum circuit, are neglected. With current cross-resonance gate durations of $200 - 400$ ns, delays can also be neglected since $\tau_{\text{delay}} \approx 250$ μs for current hardware [55] which is two orders of magnitude faster than the circuit duration. The CNOT duration is thus currently the main driver of execution-time on noisy quantum hardware. For example, the circuit of a complete interaction graph with 485 variables has a single-shot duration of 14.9 ms on a heavy-hex coupling map. Optimal control schemes show that it is in principle possible to reduce the duration of the single- and two-qubit gates by an order of magnitude [57, 58], see Tab. 2. This reduces the QAOA run-time by an order of magnitude and will make the fixed delay τ_{delay} after each shot more relevant.

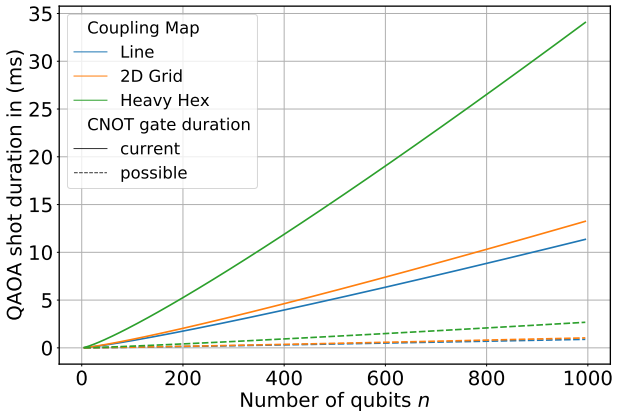


Figure 7: Scaling of the duration of a single QAOA circuit τ_{shot} as a function of problem size n for complete graphs with $p = \log_2(n)$. The solid lines show current CNOT gate durations while the dashed lines correspond to $\tau_{\text{cx}} = 30$ ns.

3.2.2 Number of shots required

The classical optimizer has to minimize the objective function $E(\boldsymbol{\theta}) = \langle \psi(\boldsymbol{\theta}) | H_C | \psi(\boldsymbol{\theta}) \rangle$, where $\boldsymbol{\theta} = (\boldsymbol{\beta}, \boldsymbol{\gamma})$, which it can only stochastically access [66]. A simulation of a variational algorithm must therefore consider this effect. For example, QAOAs simulated in Qiskit [67] always perform better with the state-vector simulator, which does not have sampling noise, than the shot-based QASM simulator, see Fig. 8. A large number of shots thus helps QAOA converge [15, 23] but increases its execution-time. Zeroth-order methods optimize by directly estimating $E(\boldsymbol{\theta})$ [47]. They prepare and measure each trial state N_{shots} times. Measurement k randomly projects $|\psi(\boldsymbol{\theta})\rangle$ onto a basis-state with an energy E_k thereby estimating $E(\boldsymbol{\theta})$ by $\overline{E} = N_{\text{shots}}^{-1} \sum_k E_k$. Reference [47] shows that for 1-local Hamiltonians with n qubits the total number of shots over the course of the optimization, i.e., $N_{\text{iter}} N_{\text{shots}}$, needed to reach a precision ϵ within the vicinity of the optima is lower bounded by $\Omega(n^3/\epsilon^2)$. By contrast, first-order methods optimize by taking measurements that correspond to the gradient $\partial_{\boldsymbol{\theta}} E(\boldsymbol{\theta})$ and are sometimes referred to as analytical gradient measurements [68, 69]. Furthermore, for 1-local Hamiltonians Ref. [47] shows that the total number of shots required by first-order methods to reach an ϵ precision scale as $\Theta(n^2/\epsilon)$. In this setting, first-order methods therefore converge faster but still require a large number of shots. Recently, optimizers that scale the number of shots based on the magnitude of the gradient have been developed to reduce the shot cost [70]. The $\Theta(n^2/\epsilon)$ scaling is a significant shot cost. However, since optimal QAOA parameters concentrate for similar problem instances [71] this cost may be amortized when solving many problem instances originating from the same reasonable distribution.

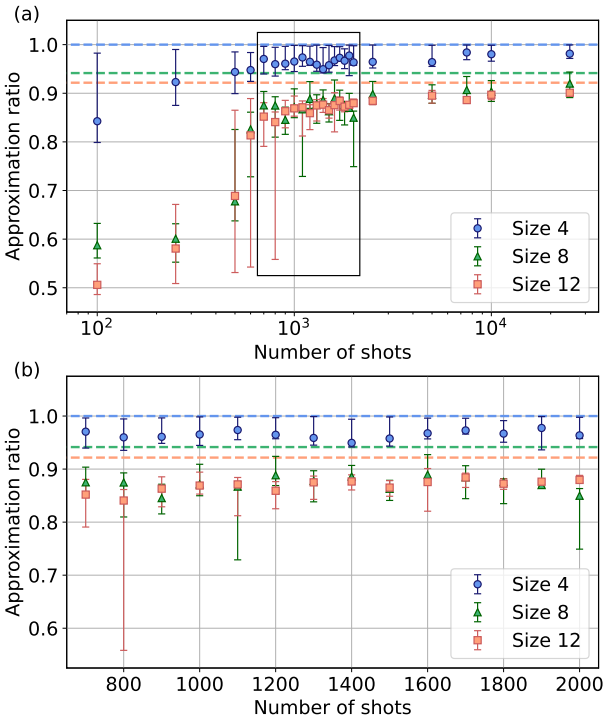


Figure 8: Depth 12 QAOA for Sherrington-Kirkpatrick graphs with 4, 8 and 12 nodes optimized with COBYLA. (a) Energy normalized to the ground-state energy, i.e., approximation ratio, after a depth 12 QAOA optimization. The points show the median of ten different random instances as a function of the number of measured shots. The error bars show the 25% and 75% quantiles. The dashed lines show a state vector simulation which is equivalent to $N_{\text{shots}} \to \infty$ up to machine precision. (b) Data points within framed region in (a).

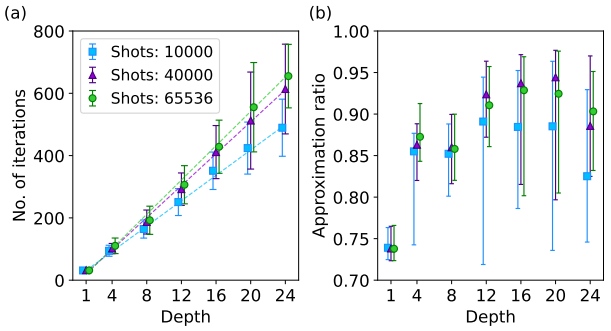


Figure 9: (a) Number of iterations completed by COBYLA. Each data point is the average number of iterations for 30 graphs with the standard deviation indicated as error bars. A small p shift is added for visualization purposes. The number of iterations is approximately constant with n (data not shown) and grows linearly with p, as shown by the linear fits (dashed lines) with slopes of 20.4, 25.6, and 27.5 for 10^4, $4 \cdot 10^4$, and 2^{16} shots, respectively. (b) The approximation ratio increases with QAOA depth. A larger number of shots results in better approximation ratios and more COBYLA iterations.

3.2.3 Number of iterations

We further investigate the number of iterations of the classical optimizer empirically with COBYLA by running QAOA simulations on Qiskit's QASM simulator. We set the number of maximum iterations to a high value (100,000) so that COBYLA terminates before reaching this threshold. The number of completed iterations is recorded for Sherrington-Kirkpatrick graphs with size n from 5 to 10, with edge weights randomly chosen from $\{-1, 1\}$ resulting in a total of 30 graphs, i.e., five per size. We initialize $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ for QAOA depths $p \in \{1, 4, 8, 12, 16, 20, 24\}$ with the Trotterized Quantum Annealing (TQA) protocol, i.e. a discretized annealing schedule, with a time-step of 0.75 [72] as it performs better than random guesses. We observe that the number of iterations grows linearly with p for all simulated graphs, see Fig. 9. At fixed p we did not see a change due to the graph size. The mean cut value shows a noticeable improvement when increasing the shots from 10^4 to $4 \cdot 10^4$ and the number of iterations increases with the number of shots since the optimizer is able to resolve finer details in the optimization landscape. Since we use $p = \Omega(\log_2 n)$ we therefore approximate $N_{\text{iter}} \approx 25 \log_2(n)$. This estimate is obtained with noiseless simulations. In practice experimental noise will make it harder to converge to a good solution [73] and advances in optimizers for variational quantum algorithms may help speed-up convergence [74]. Furthermore, if $N_{\text{iter}} N_{\text{shots}}$ scales as $\Theta(n^2/\epsilon)$, then a logarithmic scaling of N_{iter} with n implies that many shots are needed at each iteration further confirming that hardware initialization times can be neglected, i.e., $N_{\text{shots}} \tau_{\text{shot}} \gg \tau_{\text{init}}$.

3.2.4 Total QAOA execution time

We now combine the results from the previous three sections to estimate the execution time of QAOA as

$$\tau_{QAOA} \approx N_{\text{iter}} N_{\text{shots}} \cdot \log_2(n) L_{\text{cx}}(n, D) \tau_{\text{cx}}.$$

Here, the linear dependence on p of the single-shot duration gives a factor $\log_2(n)$. The required number of shots is the largest source of uncertainty in the estimation of the execution time. We require at least $N_{\text{shots}}(n) \geq 10^3$ for small problems in noiseless conditions. If $N_{\text{iter}} N_{\text{shots}}$ scales as $\Theta(n^2/\epsilon)$, as suggested by 1-local Hamiltonians [47], the impact on the execution time will be significant even for problems with a few hundred of variables, see Fig. 10. With 10^4 shots we estimate that the execution time of a complete graph with 485 nodes on a heavy-hex processor is 9.7 hours, see Fig. 10. With the same number of shots a sparse graph with $D = 0.1$ would require at least one hour to execute. Furthermore, these estimates show that decreasing the CNOT gate duration is crucial.

We have not taken into account the cost of error mitigation strategies. For example, the expectation

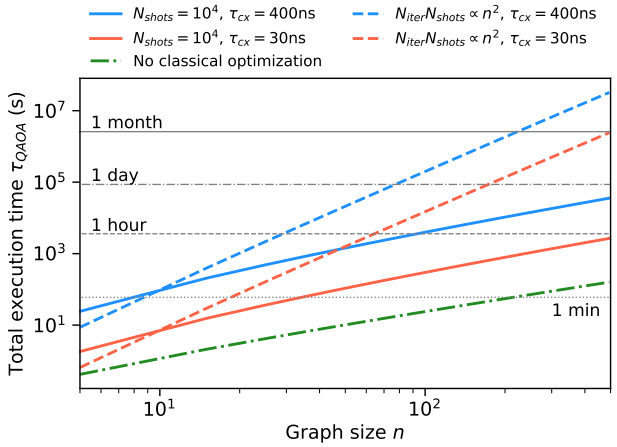


Figure 10: Total execution-time for complete graphs computed as $N_{\text{iter}}N_{\text{shots}}\tau_{\text{shot}}$ on a heavy-hex coupling map, i.e. $\tau_{\text{shot}} = L_{CX}(n)\log_2(n)\tau_{\text{cx}}$ with $L_{CX}(n) = 9n$. The quadratic scaling (dashed lines) corresponds to $N_{\text{iter}}N_{\text{shots}} = 25{\cdot}10^4\log_2(10)(n/10)^2$ chosen such that a size $n = 10$ graph requires a total of $25\log_2(10)\cdot 10^4$ shots following Fig. 9. Solid lines show a fixed number of shots per iteration, i.e., $N_{\text{iter}}N_{\text{shots}} = 25\cdot 10^4\log_2(n)$. The dashed-dotted line shows the total execution time without classical optimization, i.e. $N_{\text{iter}} = 1$, with a fixed number of shots $N_{\text{shots}} = 10^4$ and CNOT gates lasting 400 ns. This corresponds to a case where the classical optimization has been amortized over a large number of problem instances.

value can be extrapolated to the zero-noise level by measuring the energy at different noise levels [2, 75]. This multiplies $N_{\text{shots}}\cdot\tau_{\text{shot}} + \tau_{\text{init}}$ by the number of noise-levels measured. Less noisy energy evaluations may also reduce the number of iterations needed.

Care must be taken when comparing quantum-based optimizers to classical solvers [76] but with current quantum technology the estimated execution appears to be significant [77]. However, a possible advantage of QAOA over classical optimizers could lie in quickly generating good yet sub-optimal solutions by foregoing the classical optimization algorithm and initializing $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ from a known good schedule, such as an annealing schedule [72]. Furthermore, as optimal QAOA parameters tend to concentrate, optimizing one problem may be sufficient for a family of similar problems [71, 78–81]. We account for this possibility by removing the factor $N_{\text{iter}} = 25\log_2(n)$ in the execution time and use a fixed number of shots $N_{\text{shots}} = 10^4$. Under these assumptions candidate solutions for a graph with 500 nodes can be generated in under three minutes, see the dashed-dotted line in Fig. 10.

4 Qiskit Runtime hardware results

The current execution model on cloud-based quantum computers sends a set of circuits as a job through the entire stack and queue. Circuit transpilation and result analysis is done on the client side. This is particularly inefficient for variational algorithms. The Qiskit Runtime allows users to run an entire program in a containerized service close to the backend to avoid latencies between the user and the backend at each iteration. This enables a significantly faster execution of variational algorithms like QAOA [3].

We first demonstrate the QAOA Runtime program with a seven-variable weighted maximum cut optimization problem with a graph $\mathcal{G}_{10}$ with 10 unique Pauli Z_iZ_j terms and depths $p \in \{2, 3, 4\}$. Each edge (i, j) has a weight $\omega_{i,j}$ of -1 or 1 with a 50% probability, see Appendix F. $\mathcal{G}_{10}$ was constructed such that it can be implemented with one swap layer on the seven-qubit *ibm_nairobi* system. Since the energy E is related to the cut value C by $E = -2C + \sum_{(i,j)}\omega_{ij}$ we minimize the energy to maximize the cut. For $\mathcal{G}_{10}$ we have $E = -2C$. We run QAOA with SPSA due to the noisy environment [82, 83] and measure 2^{14} shots at each energy evaluation. The 0.005 learning rate and 0.01 perturbation of SPSA were chosen by calibrating it on a depth-one landscape. The initial $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ values come from TQA initialization [72]. First, we run $\mathcal{G}_{10}$ three times with $p = 2$ with the default transpiler setting of Qiskit which does not use swap strategies. Here, we do not observe any convergence, see the red data in Fig. 11. We attribute this to the 0.83% average CNOT gate error of *ibm_nairobi* and the deep CNOT gate count of the circuits. Indeed, the circuits have 89, 127, and 173 CNOT gates for depth $p = 2, 3$, and 4, respectively.

To improve convergence we rerun the QAOA Runtime program with optimized settings. First, we use the swap strategies discussed previously. Second, after the parameters are bound we run a pulse-efficient transpiler pass [84] at each QAOA iteration to remove any unnecessary single-qubit gates and to minimize the cross-resonance gate usage. The gains from these transpiler passes are summarized in Tab. 3. Third, we employ CVaR optimization with an α of 0.5, i.e. we retain only the best 50% of the measured shots at each iteration [85]. Finally, we use readout error mitigation to reduce measurement errors [86, 87]. We observe a significant reduction in energy for $\mathcal{G}_{10}$ for depth $p = 2$ as function of the iteration number. Depths 3 and 4 would require 72 and 96 CNOT gates, respectively, without pulse-efficient transpilation and are most likely noise limited. Crucially, the jobs that did see convergence manage to increase the probability of sampling a good cut when compared to random sampling, compare the histograms in Fig. 11 with the dashed grey line. Interestingly, the hardware has a higher probability of sampling the maximum cut which may be due to noise such as T_1-induced errors. Crucially, each run of the variational algorithm

[3] The QAOA Runtime program is publicly available through the IBM Quantum Services https://quantum-computing.ibm.com/.

	CNOT count depth p				Schedule duration (μs) depth p			
$\mathcal{G}_{10}$	1	2	3	4	1	2	3	4
Unoptimized	39	89	127	173	6.6	14.6	21.2	28.6
Optimized	24	48	72	96	2.1	4.1	6.1	8.1

Table 3: CNOT gate count and schedule duration for $\mathcal{G}_{10}$. The unoptimized row corresponds to the default transpiler settings in Qiskit while the optimized row corresponds to the swap strategies followed by a pulse-efficient transpilation shown for $\gamma = \pi/4$.

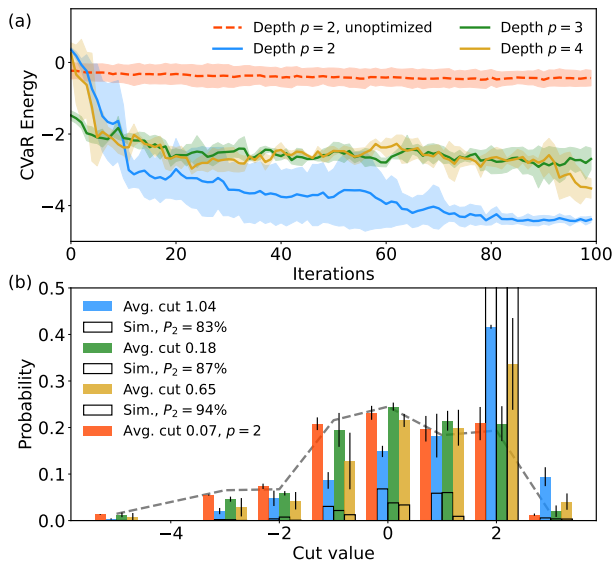


Figure 11: Execution of a seven-qubit QAOA on *ibm_nairobi*. (a) The average CVaR energy of SPSA's two evaluations per iteration averaged over three independent optimizations with TQA initial points. The shaded area is the standard deviation of the three runs. The dashed red line shows a depth-two QAOA with unoptimized transpiler settings. Solid lines show QAOA transpiled with swap strategies, pulse-efficient, and executed with CVaR with $\alpha = 0.5$. (b) Cut distribution of the best measured point averaged over the three optimizations. Vertical black lines show the standard deviation. Here, the dashed gray line indicates the distribution obtained by uniformly sampling integers from $\{0, \ldots, 2^n - 1\}$ and converting them to a bit string representing a cut. The black empty bars show the histogram of a noiseless QASM simulation with 2^{14} shots. The legend contains the simulated probability P_2 of a cut with value two. The execution time of each job is on average 1 hour and 5 minutes.

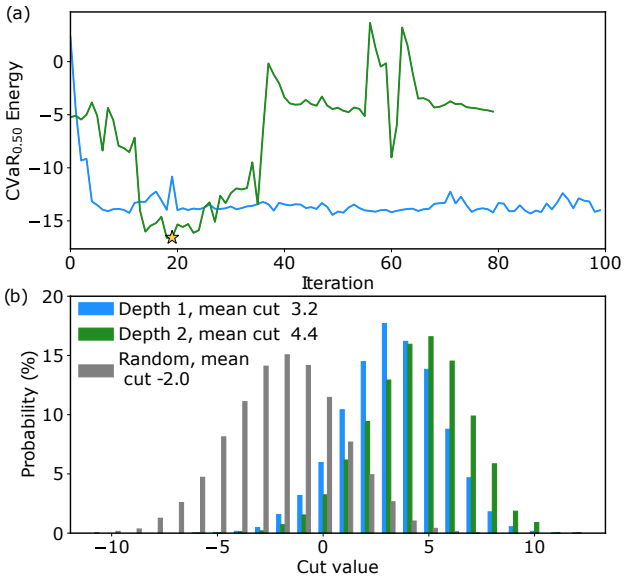


Figure 12: QAOA on a weighted maximum cut problem with a graph native to the hardware of *ibmq_mumbai* with a maximum cut of 12. At each energy evaluation 16384 shots are gathered. (a) Energy as function of the iteration number for depth-one and two QAOA initialized at $\gamma = 1$ and $\beta = 0.5$ and $\boldsymbol{\gamma} = (0.375, 0.75)$ and $\boldsymbol{\beta} = (0.375, 0)$, respectively. The star with a $\mathrm{CVaR}_{0.5}$ energy of -16.57 ± 0.16 marks the depth-two parameters $\boldsymbol{\gamma} = (1.1245, 0.343)$ and $\boldsymbol{\beta} = (1.317, 0.548)$ that resulted in the green distribution of cuts. (b) Probability distribution of the best measured cuts for depth-one (blue), depth-two (green), and random sampling (gray). The energy E is related to the cut value C by $E = -2C - 4$.

required only one hour on the cloud-based quantum computer. In addition, we evaluate the criterion in Eq. (3). A single layer of H_C has eleven and two layers of two and one CNOT gates, respectively. We therefore approximate this as 12 layers with two gates. The average gate fidelity on *ibm_nairobi* is 98.88% which results in a maximum gate bound of 52 and 103 layers for ϵ values of 10^{-1} and 10^{-2}, respectively. Equivalently, for the depth-four QAOA, which has 48 layers, there is a Gibbs state which can be sampled from classically in polynomial time which approximates the energy to within an ϵ of 0.115, see Sec. 3.1.

We also investigate QAOA on the 27 qubit device *ibmq_mumbai*. Here, we only use a hardware native graph with random edge weights chosen from $\{-1, 1\}$ since such graphs already require 56 CNOT gates per QAOA layer and *ibmq_mumbai* has an average CNOT gate error of $0.66 \pm 0.14\%$. A hardware native graph requires six CNOT layers with on average 28/3 CNOT gates per layer. Following Eq. (3) the maximum number of layers with $\epsilon = 10^{-1}$ is thus 13 when cross-talk effects from applying gates in parallel are neglected. Here, we set the learning rate and perturbation of SPSA to 0.005 and 0.01, respectively. Depth-one QAOA (which can be classically simulated efficiently) distinguishes noise from signal

Cut		QAOA		Single-shot probability		
size	Graph	$p=1$	$p=2$	Uniform	$p=1$	$p=2$
10	212	25	148	$1.6 \cdot 10^{-6}$	$1.5 \cdot 10^{-3}$	$9.0 \cdot 10^{-3}$
11	12	3	6	$8.9 \cdot 10^{-8}$	$1.8 \cdot 10^{-4}$	$3.7 \cdot 10^{-4}$
12	2	1	7	$1.5 \cdot 10^{-8}$	$6.1 \cdot 10^{-5}$	$4.3 \cdot 10^{-4}$

Table 4: Number of cuts with size ≥ 10. Graph: number of cuts of each size in the Graph. QAOA: number of measured cuts out of the 2^{14} shots shown in Fig. 12(b). The last three columns indicate the probability of a single-shot producing a cut of the given size with uniform sampling and the QAOA distributions of Fig. 12(b).

and we observe an improved average cut value compared to random sampling. Depth-two QAOA shows signs of convergence but is impacted by noise which causes large energy jumps. However, it still produces a lower energy state than depth-one QAOA. Details are given in Appendix F.

Evaluating the cut value of each of the 2^{27} possible solutions is still numerically feasible. There are 212, 12, and 2 solutions with a cut value of ten, eleven and twelve (the max-cut), respectively. Therefore, the probability of measuring a cut with a value of ten or more by random sampling is $1.6 \cdot 10^{-6}$. Out of the 2^{14} cuts sampled from the optimal depth-two state 7 cuts had a maximum value which corresponds to a probability of 0.043%, see Tab. 4. Here, a cut with a value of ten corresponds to an approximation ratio of 83% which is close to, but lower than, the Goemans-Williamson approximation ratio of $\sim 88\%$ [88]. According to the criterion in Eq. (3) with $\epsilon = 10^{-1}$ depth-two is just within reach.

5 Discussion and Conclusion

We have investigated swap strategies to implement dense circuits built from commuting two-qubit gates on linear, grid, and heavy-hex coupling maps. For QAOA higher connectivity is not always synonymous with lower gate count due to simplifications between R_{ZZ} and SWAP gates. Crucially, a line swap strategy is better on a two-dimensional grid than the grid strategy we put forward. However, our swap strategies on grid coupling maps with dimension three or higher reduce the circuit depth compared to a linear strategy. Furthermore, the heavy-hex coupling map is almost identical to a linear coupling map. We note that these strategies may not be optimal and that better strategies may exist, especially for low-density problems. Crucially, the ability to move logical variables through the physical coupling map means that digital quantum computers do not incur an embedding overhead in the number of qubits as do quantum annealers [89].

The fidelity estimates in Sec. 3.1 show that dense optimization problems will almost certainly require gates with an error rate far below fault-tolerance thresholds. Furthermore, evaluating the depth criterion in Eq. (3) with gate fidelities measured in isolation may yield a depth bound that is too optimistic. This also shows a need for application tailored hardware benchmarks [90, 91].

In Sec. 3.2 we provided a methodology to estimate the execution time of QAOA on noisy hardware which we found to be significant. We caution the reader that these numbers contain a large amount of uncertainty and can be impacted by noise [92]. This methodology can guide the development of the control hardware. For example, for the large number of shots ($> 10^3$) that QAOA needs the initialization time of the classical control hardware [55] is negligible if kept below a second per iteration since the duration of a single shot is likely larger than 1 ms for problems of practical interest, see Fig. 7. Nevertheless, the duration of the two-qubit gate and the number of shots needed to estimate the energy significantly impact the execution time. Pulse-efficient gate implementations may help reduce the execution time of QAOA circuits by leveraging the R_{ZX} gate instead of the CNOT gate [84, 93, 94]. The execution time estimation also depends on the variant of QAOA. For example, counteradiabatic driving reduces p by adding counteradiabatic gates and an extra parameter at each layer [95, 96] while Recursive-QAOA [62] will also produce different execution time estimates. However, the extra gates and different number of parameters to optimize may impact the execution time as well. By contrast, fault-tolerant architectures require a different execution time estimation methodology, as discussed in Ref. [97] which also found that faster error correcting codes are needed to make heuristics for combinatorial optimization competitive. This also suggests that algorithmic improvements to QAOA [95, 98] such as warm-start methods [99–101] will be required to get a quantum advantage in combinatorial optimization with heuristic algorithms.

We demonstrated a Qiskit Runtime program for QAOA on a cloud-based quantum computer. Using QAOA-tailored transpiler methods we significantly reduced the gate count and duration of the underlying schedules. This resulted in cut distributions biased towards high-value cuts when compared to random sampling. Here, we caution that Goemans-Williamson randomized rounding and related procedures are a more meaningful benchmark for large problems [88].

Ultimately, our results are limited by the fidelity of the cross-resonance gate and decoherence. CVaR aggregation made it possible to observe convergence at depth-two with 27 qubits. Deeper QAOA circuits may yet be possible, often at the expense of more shots, using advanced error mitigation methods [102] such as Pauli-Twirling [103], Probabilistic Error Cancelation [104], M3 readout error mitigation [105], and Zero-Noise Extrapolation [2, 75].

6 Acknowledgements

The authors acknowledge Sergey Bravyi, Giacomo Nannicini, and Libor Caha for useful discussions. This work was also supported by the Hartree National Centre for Digital Innovation program, funded by the Department for Business, Energy and Industrial Strategy, United Kingdom. We acknowledge the use of IBM Quantum services for this work. Code availability: The swap strategies have been implemented in Qiskit as transpiler passes for blocks of commuting two-qubit gates [4]. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. The current list of IBM trademarks is available at https: //www.ibm.com/legal/copytrade.

References

[1] Nikolaj Moll, Panagiotis Barkoutsos, Lev S. Bishop, Jerry M. Chow, Andrew Cross, Daniel J. Egger, Stefan Filipp, Andreas Fuhrer, Jay M. Gambetta, Marc Ganzhorn, and *et al.* Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Sci. Technol.*, 3(3):030503, 2018. DOI: 10.1088/2058-9565/aab822.

[2] Abhinav Kandala, Kristan Temme, Antonio D. Corcoles, Antonio Mezzacapo, Jerry M. Chow, and Jay M. Gambetta. Error mitigation extends the computational reach of a noisy quantum processor. *Nature*, 567:491–495, 2018. DOI: 10.1038/s41586-019-1040-7.

[3] Marc Ganzhorn, Daniel J. Egger, Panagiotis Kl. Barkoutsos, Pauline Ollitrault, Gian Salis, Nikolaj Moll, Andreas Fuhrer, Peter Mueller, Stefan Woerner, Ivano Tavernelli, and Stefan Filipp. Gate-efficient simulation of molecular eigenstates on a quantum computer. *Phys. Rev. Applied*, 11:044092, Apr 2019. DOI: 10.1103/PhysRevApplied.11.044092.

[4]The following pull requests contain the swap strategies as implemented in Qiskit: PR #7813, and PR #7979.

[4] Stefan Woerner and Daniel J. Egger. Quantum risk analysis. *npj Quantum Inf.*, 5:15, 2019. DOI: 10.1038/s41534-019-0130-6.

[5] Lee Braine, Daniel J. Egger, Jennifer Glick, and Stefan Woerner. Quantum algorithms for mixed binary optimization applied to transaction settlement. *IEEE Trans. on Quantum Eng.*, 2:1–8, 2021. DOI: 10.1109/TQE.2021.3063635.

[6] Daniel J. Egger, Claudio Gambella, Jakub Mareček, Scott McFaddin, Martin Mevissen, Rudy Raymond, Aandrea Simonetto, Sefan Woerner, and Elena Yndurain. Quantum computing for finance: State-of-the-art and future prospects. *IEEE Trans. on Quantum Eng.*, 1: 1–24, 2020. DOI: 10.1109/TQE.2020.3030314.

[7] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. 2014. DOI: 10.48550/ARXIV.1411.4028.

[8] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem. 2014. DOI: 10.48550/ARXIV.1412.6062.

[9] Zhi-Cheng Yang, Armin Rahmani, Alireza Shabani, Hartmut Neven, and Claudio Chamon. Optimizing variational quantum algorithms using pontryagin's minimum principle. *Phys. Rev. X*, 7:021027, May 2017. DOI: 10.1103/PhysRevX.7.021027.

[10] Andrew Lucas. Ising formulations of many NP problems. *Front. Phys.*, 2:5, 2014. ISSN 2296-424X. DOI: 10.3389/fphy.2014.00005.

[11] Bas Lodewijks. Mapping NP-hard and NP-complete optimisation problems to quadratic unconstrained binary optimisation problems. *arXiv:1911.08043*, 2019. DOI: 10.48550/ARXIV.1911.08043.

[12] IBM ILOG CPLEX Optimizer. URL `https://www.ibm.com/analytics/cplex-optimizer`.

[13] Andrey Kardashin, Anastasiia Pervishko, Jacob Biamonte, and Dmitry Yudin. Numerical hardware-efficient variational quantum simulation of a soliton solution. *Phys. Rev. A*, 104:L020402, Aug 2021. DOI: 10.1103/PhysRevA.104.L020402.

[14] Matthew P. Harrigan, Kevin J. Sung, Matthew Neeley, Kevin J. Satzinger, Frank Arute, Kunal Arya, Juan Atalaya, Joseph C. Bardin, Rami Barends, Sergio Boixo, and *et al.* Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. *Nat. Phys.*, 17(3):332–336, Mar 2021. ISSN 1745-2481. DOI: 10.1038/s41567-020-01105-y.

[15] Krishanu Sankar, Artur Scherer, Satoshi Kako, Sam Reifenstein, Navid Ghadermarzy, Willem B. Krayenhoff, Yoshitaka Inui, Edwin Ng, Tatsuhiro Onodera, Pooya Ronagh,

and Yoshihisa Yamamoto. Benchmark study of quantum algorithms for combinatorial optimization: Unitary versus dissipative. 2021. DOI: 10.48550/ARXIV.2105.03528.

[16] Hazel A. Chieza, Maxine T. Khumalo, Krupa Prag, and Matthew Woolway. On the computational performance of IBM quantum devices applied to combinatorial optimisation problems. In *2020 7th International Conference on Soft Computing Machine Intelligence (ISCMI)*, pages 260–264, 2020. DOI: 10.1109/ISCMI51676.2020.9311605.

[17] Harry Markowitz. Portfolio selection. *J. Finance*, 7(1):77–91, 1952. DOI: 10.2307/2975974.

[18] Francisco Barahona, Martin Grötschel, Michael Jünger, and Gerhard Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Oper. Res.*, 36(3):493–513, 2021/11/19/ 1988. DOI: 10.1287/opre.36.3.493.

[19] Michel Devoret and Robert J. Schoelkopf. Superconducting circuits for quantum information: An outlook. *Science*, 339:1169–1174, 2013. DOI: 10.1126/science.1231930.

[20] Philip Krantz, Morten Kjaergaard, Fei Yan, Terry P. Orlando, Simon Gustavsson, and William D. Oliver. A quantum engineer's guide to superconducting qubits. *Appl. Phys. Rev.*, 6 (2):021318, 2019. DOI: 10.1063/1.5089550.

[21] Larry Isenhower, Mark Saffman, and Klaus Mølmer. Multibit CkNOT quantum gates via Rydberg blockade. *Quantum Inf. Process.*, 10 (6):755, Sep 2011. DOI: 10.1007/s11128-011-0292-4.

[22] Guido Pagano, Aniruddha Bapat, Patrick Becker, Katherine S. Collins, Arinjoy De, Paul W. Hess, Harvey B. Kaplan, Antonis Kyprianidis, Wen Lin Tan, Christopher Baldwin, and *et al.* Quantum approximate optimization of the long-range Ising model with a trapped-ion quantum simulator. *Proc. Natl. Acad. Sci. U.S.A.*, 117(41): 25396–25401, 2020. ISSN 0027-8424. DOI: 10.1073/pnas.2006373117.

[23] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Phys. Rev. X*, 10:021067, Jun 2020. DOI: 10.1103/PhysRevX.10.021067.

[24] Henning Labuhn, Daniel Barredo, Sylvain Ravets, Sylvain de Léséleuc, Tommaso Macrì, Thierry Lahaye, and Antoine Browaeys. Tunable two-dimensional arrays of single rydberg atoms for realizing quantum ising models. *Nature*, 534(7609):667–670, Jun 2016. DOI: 10.1038/nature18274.

[25] 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. *Phys. Rev. X*, 10:011022, Jan 2020. DOI: 10.1103/PhysRevX.10.011022.

[26] Ron Schutjens, Fadi Abu Dagga, Daniel J. Egger, and Frank K. Wilhelm. Single-qubit gates in frequency-crowded transmon systems. *Phys. Rev. A*, 88:052330, Nov 2013. DOI: 10.1103/PhysRevA.88.052330.

[27] David C. McKay, Sarah Sheldon, John A. Smolin, Jerry M. Chow, and Jay M. Gambetta. Three-qubit randomized benchmarking. *Phys. Rev. Lett.*, 122:200502, May 2019. DOI: 10.1103/PhysRevLett.122.200502.

[28] Peng Zhao, Kehuan Linghu, Zhiyuan Li, Peng Xu, Ruixia Wang, Guangming Xue, Yirong Jin, and Haifeng Yu. Quantum crosstalk analysis for simultaneous gate operations on superconducting qubits. *PRX Quantum*, 3:020301, Apr 2022. DOI: 10.1103/PRXQuantum.3.020301.

[29] Alexander Cowtan, Silas Dilkes, Ross Duncan, Alexandre Krajenbrink, Will Simmons, and Seyon Sivarajah. On the Qubit Routing Problem. In Wim van Dam and Laura Mancinska, editors, *14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019)*, volume 135 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:32, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-112-2. DOI: 10.4230/LIPIcs.TQC.2019.5.

[30] Alwin Zulehner, Alexandru Paler, and Robert Wille. An efficient methodology for mapping quantum circuits to the IBM QX architectures. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 38(7):1226–1236, 2019. DOI: 10.1109/TCAD.2018.2846658.

[31] Jens Koch, Terri M. Yu, Jay Gambetta, Andrew A. Houck, David I. Schuster, Johannes Majer, Alexandre Blais, Michel H. Devoret, Steven M. Girvin, and Robert J. Schoelkopf. Charge-insensitive qubit design derived from the cooper pair box. *Phys. Rev. A*, 76:042319, Oct 2007. DOI: 10.1103/PhysRevA.76.042319.

[32] Aaron Lye, Robert Wille, and Rolf Drechsler. Determining the minimal number of swap gates for multi-dimensional nearest neighbor quantum circuits. In *The 20th Asia and South Pacific Design Automation Conference*, pages 178–183, 2015. DOI: 10.1109/ASPDAC.2015.7059001.

[33] Abhoy Kole, Kamalika Datta, and Indranil Sengupta. A heuristic for linear nearest neighbor realization of quantum circuits by swap gate insertion using N-gate lookahead. *IEEE Trans.*

Emerg. Sel. Topics Circuits Syst., 6(1):62–72, 2016. DOI: 10.1109/JETCAS.2016.2528720.

[34] Anirban Bhattacharjee, Chandan Bandyopadhyay, Robert Wille, Rolf Drechsler, and Hafizur Rahaman. A novel approach for nearest neighbor realization of 2D quantum circuits. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 305–310, 2018. DOI: 10.1109/ISVLSI.2018.00063.

[35] Azim Farghadan and Naser Mohammadzadeh. Mapping quantum circuits on 3D nearest-neighbor architectures. *Quantum Sci. Technol.*, 4(3):035001, apr 2019. DOI: 10.1088/2058-9565/ab177a.

[36] Lingling Lao and Dan E. Browne. 2QAN: A quantum compiler for 2-local qubit hamiltonian simulation algorithms. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, page 351–365, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450386104. DOI: 10.1145/3470496.3527394.

[37] Yuwei Jin, Lucent Fong, Yanhao Chen, Ari B. Hayes, Shuo Zhang, Chi Zhang, Fei Hua, Zheng, and Zhang. A structured method for compilation of QAOA circuits in quantum computing. 2021. DOI: 10.48550/ARXIV.2112.06143.

[38] Mahabubul Alam, Abdullah Ash-Saki, and Swaroop Ghosh. Circuit compilation methodologies for quantum approximate optimization algorithm. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 215–228, 2020. DOI: 10.1109/MICRO50266.2020.00029.

[39] Yuichi Hirata, Masaki Nakanishi, Shigeru Yamashita, and Yasuhiko Nakashima. An efficient method to convert arbitrary quantum circuits to ones on a linear nearest neighbor architecture. In *2009 Third International Conference on Quantum, Nano and Micro Technologies*, pages 26–33, 2009. DOI: 10.1109/ICQNM.2009.25.

[40] Loïc Henriet, Lucas Beguin, Adrien Signoles, Thierry Lahaye, Antoine Browaeys, Georges-Olivier Reymond, and Christophe Jurczak. Quantum computing with neutral atoms. *Quantum*, 4:327, September 2020. ISSN 2521-327X. DOI: 10.22331/q-2020-09-21-327.

[41] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. t|ket⟩: a retargetable compiler for NISQ devices. *Quantum Sci. Technol.*, 6 (1):014003, nov 2020. DOI: 10.1088/2058-9565/ab8e92.

[42] Gushu Li, Yufei Ding, and Yuan Xie. Tackling the qubit mapping problem for NISQ-era quantum devices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, pages 1001–1014, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362405. DOI: 10.1145/3297858.3304023.

[43] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001. DOI: 10.1145/502090.502098.

[44] Dorit Aharonov, Michael Ben-Or, Russell Impagliazzo, and Noam Nisan. Limitations of noisy reversible computation, 1996. URL `https://arxiv.org/abs/quant-ph/9611028`.

[45] Daniel S. França and Raul García-Patrón. Limitations of optimization algorithms on noisy quantum devices. *Nat. Phys.*, 17(11):1221–1227, Nov 2021. ISSN 1745-2481. DOI: 10.1038/s41567-021-01356-3.

[46] Daochen Wang, Oscar Higgott, and Stephen Brierley. Accelerated variational quantum eigensolver. *Phys. Rev. Lett.*, 122:140504, Apr 2019. DOI: 10.1103/PhysRevLett.122.140504.

[47] Aram W. Harrow and John C. Napp. Low-depth gradient measurements can improve convergence in variational hybrid quantum-classical algorithms. *Phys. Rev. Lett.*, 126: 140502, Apr 2021. DOI: 10.1103/PhysRevLett.126.140502.

[48] Easwar Magesan, Jay M. Gambetta, and Joseph Emerson. Scalable and robust randomized benchmarking of quantum processes. *Phys. Rev. Lett.*, 106:180504, May 2011. DOI: 10.1103/PhysRevLett.106.180504.

[49] Easwar Magesan, Jay M. Gambetta, and Joseph Emerson. Characterizing quantum gates via randomized benchmarking. *Phys. Rev. A*, 85:042311, Apr 2012. DOI: 10.1103/PhysRevA.85.042311.

[50] Antonio D. Córcoles, Jay M. Gambetta, Jerry M. Chow, John A. Smolin, Matthew Ware, Joel Strand, Britton L. T. Plourde, and Matthias Steffen. Process verification of two-qubit quantum gates by randomized benchmarking. *Phys. Rev. A*, 87:030301, Mar 2013. DOI: 10.1103/PhysRevA.87.030301.

[51] Alexander Erhard, Joel J. Wallman, Lukas Postler, Michael Meth, Roman Stricker, Esteban A. Martinez, Philipp Schindler, Thomas Monz, Joseph Emerson, and Rainer Blatt. Characterizing large-scale quantum computers via cycle benchmarking. *Nat. Commun.*, 10 (1):5347, Nov 2019. ISSN 2041-1723. DOI: 10.1038/s41467-019-13068-7.

[52] Prakash Murali, David C. Mckay, Margaret Martonosi, and Ali Javadi-Abhari. Software mitigation of crosstalk on noisy intermediate-scale quantum computers. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Pro-*

gramming Languages and Operating Systems, ASPLOS '20, pages 1001–1016, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371025. DOI: 10.1145/3373376.3378477.

[53] Panos Aliferis and Andrew W. Cross. Subsystem fault tolerance with the Bacon-Shor code. *Phys. Rev. Lett.*, 98:220502, May 2007. DOI: 10.1103/PhysRevLett.98.220502.

[54] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A*, 86:032324, Sep 2012. DOI: 10.1103/PhysRevA.86.032324.

[55] Andrew Wack, Hanhee Paik, Ali Javadi-Abhari, Petar Jurcevic, Ismael Faro, Jay M. Gambetta, and Blake R. Johnson. Quality, speed, and scale: three key attributes to measure the performance of near-term quantum computers. 2021. DOI: 10.48550/ARXIV.2110.14108.

[56] Daniel J. Egger, Max Werninghaus, Marc Ganzhorn, Gian Salis, Andreas Fuhrer, Peter Müller, and Stefan Filipp. Pulsed reset protocol for fixed-frequency superconducting qubits. *Phys. Rev. Applied*, 10:044030, Oct 2018. DOI: 10.1103/PhysRevApplied.10.044030.

[57] Max Werninghaus, Daniel J. Egger, Federico Roy, Shai Machnes, Frank K. Wilhelm, and Stefan Filipp. Leakage reduction in fast superconducting qubit gates via optimal control. *npj Quantum Inf.*, 7, 2021. DOI: 10.1038/s41534-020-00346-2.

[58] Susanna Kirchhoff, Torsten Keßler, Per J. Liebermann, Elie Assémat, Shai Machnes, Felix Motzoi, and Frank K. Wilhelm. Optimized cross-resonance gate for coupled transmon systems. *Phys. Rev. A*, 97:042348, Apr 2018. DOI: 10.1103/PhysRevA.97.042348.

[59] Chad Rigetti and Michel Devoret. Fully microwave-tunable universal gates in superconducting qubits with linear couplings and fixed transition frequencies. *Phys. Rev. B*, 81:134507, Apr 2010. DOI: 10.1103/PhysRevB.81.134507.

[60] Sarah Sheldon, Easwar Magesan, Jerry M. Chow, and Jay M. Gambetta. Procedure for systematically tuning up cross-talk in the cross-resonance gate. *Phys. Rev. A*, 93:060302, Jun 2016. DOI: 10.1103/PhysRevA.93.060302.

[61] Matthew B. Hastings. Classical and quantum bounded depth approximation algorithms. *Quantum Inf. Comput.*, 19, 2019. DOI: 10.26421/QIC19.13-14-3.

[62] Sergey Bravyi, Alexander Kliesch, Robert Koenig, and Eugene Tang. Obstacles to variational quantum optimization from symmetry protection. *Phys. Rev. Lett.*, 125:260505, Dec 2020. DOI: 10.1103/PhysRevLett.125.260505.

[63] Edward Farhi, David Gamarnik, and Sam Gutmann. The quantum approximate optimization algorithm needs to see the whole graph: A typical case. 2020. DOI: 10.48550/ARXIV.2005.08747.

[64] Kurtis Geerlings, Zaki Leghtas, Ioan M. Pop, Shyam Shankar, Luigi Frunzio, Robert J. Schoelkopf, Mazyar Mirrahimi, and Michel H. Devoret. Demonstrating a driven reset protocol for a superconducting qubit. *Phys. Rev. Lett.*, 110:120501, Mar 2013. DOI: 10.1103/PhysRevLett.110.120501.

[65] Antonio D. Córcoles, Maika Takita, Ken Inoue, Scott Lekuch, Zlatko K. Minev, Jerry M. Chow, and Jay M. Gambetta. Exploiting dynamic quantum circuits in a quantum algorithm with superconducting qubits. *Phys. Rev. Lett.*, 127:100501, Aug 2021. DOI: 10.1103/PhysRevLett.127.100501.

[66] Ryan Sweke, Frederik Wilde, Johannes Meyer, Maria Schuld, Paul K. Faehrmann, Barthélémy Meynard-Piganeau, and Jens Eisert. Stochastic gradient descent for hybrid quantum-classical optimization. *Quantum*, 4:314, aug 2020. ISSN 2521-327X. DOI: 10.22331/q-2020-08-31-314.

[67] Héctor Abraham and *et al.* Qiskit: An open-source framework for quantum computing. 2021. DOI: 10.5281/zenodo.2573505.

[68] Gian Giacomo Guerreschi and Mikhail Smelyanskiy. Practical optimization for hybrid quantum-classical algorithms. 2017. DOI: 10.48550/ARXIV.1701.01450.

[69] Jonathan Romero, Ryan Babbush, Jarrod R. McClean, Cornelius Hempel, Peter J. Love, and Alán Aspuru-Guzik. Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz. *Quantum Sci. Technol.*, 4(1):014008, oct 2018. DOI: 10.1088/2058-9565/aad3e4.

[70] Jonas M. Kübler, Andrew Arrasmith, Lukasz Cincio, and Patrick J. Coles. An adaptive optimizer for measurement-frugal variational algorithms. *Quantum*, 4:263, May 2020. ISSN 2521-327X. DOI: 10.22331/q-2020-05-11-263.

[71] Fernando G. S. L. Brandao, 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, 2018.

[72] Stefan H. Sack and Maksym Serbyn. Quantum annealing initialization of the quantum approximate optimization algorithm. *Quantum*, 5:491, July 2021. ISSN 2521-327X. DOI: 10.22331/q-2021-07-01-491.

[73] Gregory Quiroz, Paraj Titum, Phillip Lotshaw, Pavel Lougovski, Kevin Schultz, Eugene Dumitrescu, and Itay Hen. Quantifying the impact of precision errors on quantum approx-

imate optimization algorithms. 2021. DOI: 10.48550/ARXIV.2109.04482.

[74] Julien Gacon, Christa Zoufal, Giuseppe Carleo, and Stefan Woerner. Simultaneous Perturbation Stochastic Approximation of the Quantum Fisher Information. *Quantum*, 5:567, October 2021. ISSN 2521-327X. DOI: 10.22331/q-2021-10-20-567.

[75] Kristan Temme, Sergey Bravyi, and Jay M. Gambetta. Error mitigation for short-depth quantum circuits. *Phys. Rev. Lett.*, 119: 180509, Nov 2017. DOI: 10.1103/PhysRevLett.119.180509.

[76] Sanjeeb Dash and Jean-François Puget. On quadratic unconstrained binary optimization problems defined on chimera graphs. *OPTIMA*, 98, 2015. URL `http://www.mathopt.org/Optima-Issues/optima98.pdf`.

[77] Alain Billionnet and Sourour Elloumi. Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. *Math. Program.*, 109(1):55–68, Jan 2007. ISSN 1436-4646. DOI: 10.1007/s10107-005-0637-9.

[78] Vishwanathan Akshay, Daniil Rabinovich, Ernesto Campos, and Jacob Biamonte. Parameter concentrations in quantum approximate optimization. *Phys. Rev. A*, 104:L010401, Jul 2021. DOI: 10.1103/PhysRevA.104.L010401.

[79] Alexey Galda, Xiaoyuan Liu, Danylo Lykov, Yuri Alexeev, and Ilya Safro. Transferability of optimal qaoa parameters between random graphs. pages 171–180, 2021. DOI: 10.1109/QCE52317.2021.00034.

[80] Michael Streif and Martin Leib. Training the quantum approximate optimization algorithm without access to a quantum processing unit. *Quantum Sci. Technol.*, 5(3):034008, may 2020. DOI: 10.1088/2058-9565/ab8c2b.

[81] Ruslan Shaydulin, Phillip C. Lotshaw, Jeffrey Larson, James Ostrowski, and Travis S. Humble. Parameter transfer for quantum approximate optimization of weighted MaxCut. 2022. DOI: 10.48550/ARXIV.2201.11785.

[82] James C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Trans. Autom. Control*, 37(3):332–341, 1992. DOI: 10.1109/9.119632.

[83] James C. Spall. Accelerated second-order stochastic optimization using only function measurements. In *Proceedings of the 36th IEEE Conference on Decision and Control*, volume 2, pages 1417–1424 vol.2, 1997. DOI: 10.1109/CDC.1997.657661.

[84] Nathan Earnest, Caroline Tornow, and Daniel J. Egger. Pulse-efficient circuit transpilation for quantum applications on cross-resonance-based hardware. *Phys. Rev. Research*, 3:043088, Oct 2021. DOI: 10.1103/PhysRevResearch.3.043088.

[85] Panagiotis Kl. Barkoutsos, Giacomo Nannicini, Anton Robert, Ivano Tavernelli, and Stefan Woerner. Improving variational quantum optimization using CVaR. *Quantum*, 4:256, Apr 2020. DOI: 10.22331/q-2020-04-20-256.

[86] Sergey Bravyi, Sarah Sheldon, Abhinav Kandala, David C. McKay, and Jay M. Gambetta. Mitigating measurement errors in multiqubit experiments. *Phys. Rev. A*, 103:042605, Apr 2021. DOI: 10.1103/PhysRevA.103.042605.

[87] George S. Barron and Christopher J. Wood. Measurement error mitigation for variational quantum algorithms, 2020.

[88] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, nov 1995. DOI: 10.1145/227683.227684.

[89] Mario S. Könz, Wolfgang Lechner, Helmut G. Katzgraber, and Matthias Troyer. Embedding overhead scaling of optimization problems in quantum annealing. *PRX Quantum*, 2:040322, Nov 2021. DOI: 10.1103/PRXQuantum.2.040322.

[90] Akel Hashim, Rich Rines, Victory Omole, Ravi K. Naik, John Mark Kreikebaum, David I. Santiago, Frederic T. Chong, Irfan Siddiqi, and Pranav Gokhale. Optimized swap networks with equivalent circuit averaging for qaoa. *Phys. Rev. Research*, 4:033028, Jul 2022. DOI: 10.1103/PhysRevResearch.4.033028.

[91] Gopal Chandra Santra, Fred Jendrzejewski, Philipp Hauke, and Daniel J. Egger. Squeezing and quantum approximate optimization. 2022. DOI: 10.48550/ARXIV.2205.10383.

[92] Phillip C. Lotshaw, Thien Nguyen, Anthony Santana, Alexander McCaskey, Rebekah Herrman, James Ostrowski, George Siopsis, and Travis S. Humble. Scaling quantum approximate optimization on near-term hardware. *Scientific Reports*, 12(1):12388, Jul 2022. DOI: 10.1038/s41598-022-14767-w.

[93] Thomas Alexander, Naoki Kanazawa, Daniel J. Egger, Lauren Capelluto, Christopher J. Wood, Ali Javadi-Abhari, and David C. McKay. Qiskit pulse: programming quantum computers through the cloud with pulses. *Quantum Sci. Technol.*, 5(4):044006, Aug 2020. DOI: 10.1088/2058-9565/aba404.

[94] John P. T. Stenger, Nicholas T. Bronn, Daniel J. Egger, and David Pekker. Simulating the dynamics of braiding of majorana zero modes using an ibm quantum computer. *Phys. Rev. Research*, 3:033171, Aug 2021. DOI: 10.1103/PhysRevResearch.3.033171.

[95] P. Chandarana, N. N. Hegade, K. Paul, F. Albarrán-Arriagada, E. Solano, A. del Campo, and Xi Chen. Digitized-counterdiabatic quantum approximate optimization algorithm. *Phys. Rev. Research*, 4:013141, Feb 2022. DOI: 10.1103/PhysRevResearch.4.013141.

[96] Jonathan Wurtz and Peter J. Love. Counterdiabaticity and the quantum approximate optimization algorithm. *Quantum*, 6:635, jan 2022. DOI: 10.22331/q-2022-01-27-635.

[97] Yuval R. Sanders, Dominic W. Berry, Pedro C.S. Costa, Louis W. Tessler, Nathan Wiebe, Craig Gidney, Hartmut Neven, and Ryan Babbush. Compilation of fault-tolerant quantum heuristics for combinatorial optimization. *PRX Quantum*, 1:020312, Nov 2020. DOI: 10.1103/PRXQuantum.1.020312.

[98] David Amaro, Carlo Modica, Matthias Rosenkranz, Mattia Fiorentini, Marcello Benedetti, and Michael Lubasch. Filtering variational quantum algorithms for combinatorial optimization. *Quantum Sci. Technol.*, 7(1):015021, jan 2022. DOI: 10.1088/2058-9565/ac3e54.

[99] Daniel J. Egger, Jakub Mareček, and Stefan Woerner. Warm-starting quantum optimization. *Quantum*, 5:479, Jun 2021. DOI: 10.22331/q-2021-06-17-479.

[100] Reuben Tate, Majid Farhadi, Creston Herold, Greg Mohler, and Swati Gupta. Bridging classical and quantum with SDP initialized warm-starts for QAOA. *ACM Transactions on Quantum Computing*, jun 2022. DOI: 10.1145/3549554.

[101] Reuben Tate, Bryan Gard, Greg Mohler, and Swati Gupta. Classically-inspired mixers for QAOA beat Goemans-Williamson's Max-Cut at low circuit depths. 2021. DOI: 10.48550/ARXIV.2112.11354.

[102] Almudena Carrera Vazquez, Daniel J. Egger, David Ochsner, and Stefan Woerner. Well-conditioned multi-product formulas for hardware-friendly hamiltonian simulation. 2022. DOI: 10.48550/ARXIV.2207.11268.

[103] Youngseok Kim, Christopher J. Wood, Theodore J. Yoder, Seth T. Merkel, Jay M. Gambetta, Kristan Temme, and Abhinav Kandala. Scalable error mitigation for noisy quantum circuits produces competitive expectation values. 2021. DOI: 10.48550/ARXIV.2108.09197.

[104] Ewout van den Berg, Zlatko K. Minev, Abhinav Kandala, and Kristan Temme. Probabilistic error cancellation with sparse pauli-lindblad models on noisy quantum processors. 2022. DOI: 10.48550/ARXIV.2201.09866.

[105] Paul D. Nation, Hwajung Kang, Neereja Sundaresan, and Jay M. Gambetta. Scalable mitigation of measurement errors on quantum computers. *PRX Quantum*, 2:040326, Nov 2021. DOI: 10.1103/PRXQuantum.2.040326.

[106] Raban Iten, Romain Moyard, Tony Metger, David Sutter, and Stefan Woerner. Exact and practical pattern matching for quantum circuit optimization. *ACM Transactions on Quantum Computing*, 3(1), jan 2022. ISSN 2643-6809. DOI: 10.1145/3498325.

[107] Sung-Pil Hong. Inapproximability of the max-cut problem with negative weights. *Management Science and Financial Engineering*, 14(1): 87–90, 2008. URL https://koreascience.kr/article/JAKO200822179194611.page.

A SWAP Strategy Details

Here, we discuss details of the swap strategies summarized in Sec 2.2 of the main text. $G_0 = (V, E_0)$ is the graph of a coupling map and $N(e)$ the set of edges adjacent to edge e. Then, any swap layer S_i executable on the coupling map is defined by a subset of edges $S_i \subseteq E_0$ that satisfy $e \notin N(e')\ \forall\, e, e' \in S_i$. A swap strategy on G_0 is then a series of swap layers $\{S_i\}$ where $i \in \mathbb{N}$. We now discuss swap strategies for the line, grid and heavy-hex coupling maps.

A.1 Line

We consider the line graph of size n with vertices numbered from 0 to $n-1$ and the swap strategy shown in Fig. 2 of the main text.

Lemma 1. *For the line graph of size n the swap strategy that alternates between two swap layers, one on all odd numbered edges, and one on all even numbered edges reaches full connectivity in $n-2$ layers and is optimal.*

Proof. Let q_i denote the i-th qubit in the line. Following Lemma 1, each qubit moves continuously in one direction until reaching a line end where it reverses direction. Further, starting with the swap layer applied to the even numbered edges, the odd and even numbered qubits begin by moving left and right, respectively. The position of qubit q_i starting at node i after applying $k \leq n$ swap layers is

$$p_k(q_i) = \begin{cases} \min(i+k, 2n-i-1-k) & \text{if } i \text{ even} \\ \max(i-k, -i+k-1) & \text{if } i \text{ odd} \end{cases}$$

for $k \in \{0, \ldots, n\}$. Setting $k = n$ in the equation above, it holds that $p_n(q_i) = n-1-i$ for all $i \in 0, \ldots, n-1$. Then, after n steps, the line is fully reversed implying that any two qubits were positioned next to each other at some point during the process. This shows that the strategy reaches full connectivity after n steps.

It turns out, that we already reach full connectivity after $n-2$ steps. To prove this, note that after $n-2$ steps the position of q_i is

$$p_{n-2}(q_i) = \begin{cases} n-2 & \text{if } i = 0 \\ n-i+1 & \text{if } i \text{ even }, i > 0 \\ n-i-3 & \text{if } i \text{ odd }, i < n-2 \\ i-n+2 & \text{if } i \text{ odd }, i \geq n-2 \end{cases} \tag{6}$$

First, consider qubit q_0, i.e. case $i = 0$ in Eq. (6), which starts in the leftmost position. Its final position is the second rightmost node in the line. Therefore, during the process q_0 passes all nodes but the rightmost one. It follows that q_0 must have been positioned next to every other qubit at some point of the swap process. The qubit initially at $i \geq n-2$ with odd i, i.e. the fourth case in Eq. (6), arrives in position 0 or 1 after $n-2$ SWAP layers, i.e. $p_{n-2}(q_i) \in \{0, 1\}$. It was therefore positioned next to every other qubit at some point of the SWAP strategy. Now consider the second and third cases in Eq. (6). When $i_1, i_2 \in 1, \ldots, n-1$, such that qubit q_{i_1} is to the left of qubit q_{i_2}, i.e. $i_1 < i_2$, and neither i_1 nor i_2 are larger than or equal to $n-2$ and odd, then, after $n-2$ steps of the swap strategy $p_{n-2}(q_{i_1}) - p_{n-2}(q_{i_2})$ is equal to

$$\begin{cases} i_2 - i_1 & \text{if } i_1 = i_2 \mod 2 \\ i_2 - i_1 + 4 & \text{if } i_1 \text{ even and } i_2 \text{ odd} \\ i_2 - i_1 - 4 & \text{if } i_2 \text{ even and } i_1 \text{ odd} \end{cases} \tag{7}$$

In the first two cases $p_{n-2}(q_{i_1}) - p_{n-2}(q_{i_2}) > 0$, i.e. q_{i_1} is now to the right of q_{i_2} in the line, since by assumption $i_1 < i_2$. Hence, the corresponding qubits have switched their order within the line and must have been adjacent to each other at some point during the strategy. If i_2 is even and i_1 is odd, it suffices to consider the case where $i_1 < i_2 - 1$, since otherwise the corresponding qubits are initially adjacent. Then $i_2 - i_1 \geq 3$ and Eq. (7) implies $p_{n-2}(q_{i_1}) - p_{n-2}(q_{i_2}) \geq -1$. Hence, the corresponding qubits either end up adjacent to one another or have switched their order after $n-2$ steps. This proves that we reach full connectivity after $n-2$ swap layers.

To prove optimality, consider qubit q_0 starting in the leftmost position of the line. At any point of the process, all qubits left of q_0 have been adjacent to q_0 at some previous point. In particular, this holds for the adjacent qubit to the left of q_0. By this argument and as every node in the line graph has at most degree two, q_0 can only become connected to at most one additional qubit after every additional swap layer, namely by the edge to its right. Since q_0 is initially only connected to q_1 and needs to be connected to $n-2$ additional qubits to reach full connectivity, it then follows that no swap strategy with less that $n-2$ layers can lead to full connectivity in the line graph. □

A.2 Grid

The swap strategy for the two-dimensional grid is an extension of the line swap strategy.

Lemma 2. *For the two-dimensional grid of size n, there exists a swap strategy that reaches full connectivity in $n/2 + \mathcal{O}(\sqrt{n})$ layers. For the three dimensional grid of size n a strategy with depth $n/4 + \mathcal{O}(n^{2/3})$ exists.*

Proof. First, consider two equally long adjacent horizontal lines of qubits, where each qubit in the lower line is connected to the corresponding qubit in the upper line. We apply the line strategy in Lemma 1 to both lines, where we begin by applying SWAP gates to even numbered edges in one line and odd numbered edges in the other, see Fig 13. Since both lines are reversed after n steps, any two nodes in the upper and lower line were adjacent at some point. Crucially, this is only possible because SWAP gates for odd edges on one line are executed simultaneously with SWAP gates on even edges of the other line and vice versa. Thus, full connectivity is reached after at most n steps. Additionally, since both lines fully reverse no new connections are obtained in the last step and full connectivity is already reached after $n-1$ steps.

Second, consider the square grid with $n = x^2$ nodes divided into x rows and columns. The grid swap strategy, shown in Fig. 3 of the main text, repeats two steps.

1. Apply $x-1$ steps of the line swap strategy separately to each row. Importantly, on two adjacent rows the SWAP gates must never simultaneously be on edges with the same parity.

2. Swap the rows by applying exactly two steps of the line swap strategy to each column in parallel.

The double line example shows that after executing the first step a qubit in a row is connected to all the other qubits in its row and the neighboring rows. The second step of the strategy is executed with two swap layers. It swaps rows such that every row is now positioned next to two different rows. Thus, step 1 connects qubits of adjacent rows and step 2 shuffles the order of rows such that all rows are adjacent at some point. Since we perform two vertical swap layers in each iteration we reach full connectivity after repeating both steps $(x-2)/2$ times and step 1 one additional time at the end. We thus need

$$\frac{x-2}{2}(x+1) + (x-1) = \frac{n}{2} + \mathcal{O}(\sqrt{n})$$

swap layers, which proves the two-dimensional case. The proof for the three-dimensional case is similar. □

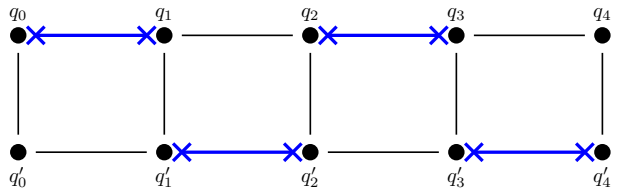


Figure 13: Swap strategy in a graph with two connected lines each with five vertices. The blue SWAP gates show the first swap layer. The qubit order in each line fully reverses after five steps and full connectivity is reached after four steps.

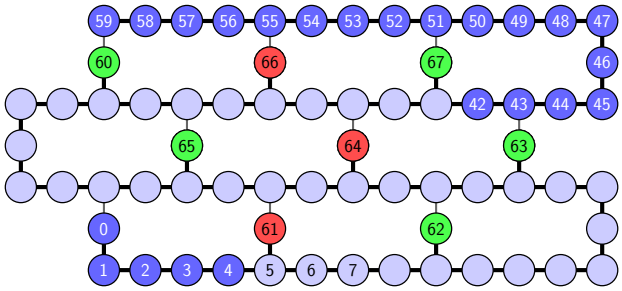


Figure 14: A three by three heavy-hex coupling map. The longest line ranges from qubits 0 to 60. By contrast, the blue qubits are part of the longest-line of the *unfolded* heavy-hex graph with $l = 60$. The dark blue nodes indicate the tails with five and $t = 18$ qubits. The red and green nodes indicate the A and B qubits, respectively. The grey edges are the edges that are removed to unfold the heavy-hex graph.

A.3 Heavy-hex

A heavy-hex coupling map has a mixture of degree-two and degree-three nodes. Its qubits are placed on the edges and vertices of hexagons [25]. Each hexagon therefore has 12 qubits. Here, we focus on $i \times j$ heavy-hex graphs which have i rows and j columns of hexagons, as exemplified by the 3×3 heavy-hex graph in Fig. 14. The total number of qubits is $n = 5ij+4(i+j)-1$ and the length of the longest line in the graph is $l_{\max} = 4(ij+i+j)+1$. The length of this line is related to the total number of qubits by $l_{\max} = 4[n(i,j)+i+j+1]/5+1$ which, to leading order, scales as $4n/5$. Furthermore, $l_{\max}$ is bound from above by

$$\frac{4}{5}n + \frac{4}{5}\sqrt{n} + 1 \tag{8}$$

when the grid of hexagons is approximately square, i.e. $i \sim j$. These preliminaries allow us to formulate the following Lemma for the heavy-hex swap strategy.

Lemma 3. *For the approximately square heavy-hex grid of size n, there exists a swap strategy that reaches full connectivity in less than $n+\sqrt{n}+61$ swap layers.*

Proof. We prove Lemma 3 by unfolding the heavy-hex graph along a line of length l where $l \mod 4 = 0$, see Fig. 14. To unfold, we delete one edge connected to the nodes not in the longest line. The result is a line graph with an optional additional node connected to every fourth node, see Fig. 15. The graph additionally has a tail of 5 vertices on one side and a tail of t vertices on the other side, such that $t = 2 \mod 4$, see Fig. 14. Here, t depends on the width of the heavy-hex graph. We will prove that Lemma 3 holds for any graph of this kind.

The proof applies a line swap strategy on the unfolded heavy-hex graph modified such that at any time some qubits remain in the positions of the additional nodes without moving along the line. We divide the process into five iterations. In each iteration a qubit either moves along $1/4$ of the line or waits in one of the adjacent nodes. If every qubit only enters a waiting state once during the complete process, all qubits will have completed l steps of the simple line strategy after five iterations, leading to full connectivity. The difficulty then lies in ensuring that every qubit is swapped into a waiting position at most once. We divide the additional nodes into two sets A and B spaced apart by eight nodes in the line, see Fig. 15. We number the edges in the line by their position and define four swap layers to reach full connectivity.

- S_1: All odd-numbered edges in the line.
- S_2: All even-numbered edges in the line.
- S_3: All edges connected to vertices in group A.
- S_4: All edges connected to vertices in group B.

We claim that the swap strategy in which we

1. alternate between S_1 and S_2 $k-7$ times starting with S_1, and
2. apply S_4 once, and
3. alternate between S_1 and S_2 7 times starting with S_2, and
4. apply S_3 once, and
5. repeat Steps 1-4 five times,

reaches full connectivity on the unfolded heavy-hex if

$$k = \frac{l}{4} - \left(\frac{l}{4} \mod 8\right) + 10. \tag{9}$$

Here, k is chosen such that $k \mod 8 = 2$ and $k > l/4$ so that every qubit travels the full line with four iterations of steps 1. and 3. Steps 2. and 4. swap qubits in and out of nodes A and B, respectively. Steps 1. to 4. require $k+2$ swap layers so that after five iterations the total number of swap layers is $5k+10$ which is thus bounded from above by $n+\sqrt{n}+61$ as seen by injecting Eq. (8) in Eq. (9) and conservatively assuming $l/4 \mod 8 = 0$.

As argued above, it suffices to show that any particular qubit will remain in a position corresponding to groups A or B for at most one iteration. We assign

each node along the line to one of eight evenly-spaced groups V_i with $i \in \{0, \ldots, 7\}$. The set of all vertices is thus partitioned into ten sets, A, B and V_i, see Fig. 15. We now examine how the qubits switch groups during steps 1. to 4. Note that during each iteration k steps of the line strategy are executed and k was chosen, such that $k \bmod 8 = 2$.

We first ignore groups A and B. Since k is even and we begin by applying SWAP gates to odd numbered edges in the line, i.e. S_1, qubits moving towards the left will switch from group V_i to V_{i-2} and qubits moving towards the right will move from V_i to V_{i+2}. If a qubit reaches the end of the line during an iteration it will either

- switch direction from left to right and switch groups from V_i to V_{7-i}, or
- switch direction from right to left and switch groups from V_i to V_{7-i+4}.

This behavior results from $k \bmod 8 = 2$. The movement pattern thus undergone by individual qubits is depicted in Fig. 16.

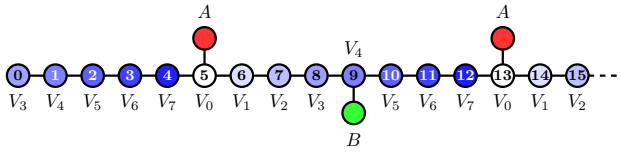


Figure 15: The unrolled heavy-hex graph with numbered nodes. The nodes are divided into ten subsets A (in red) B (in green) and V_i for $i \in \{0, \ldots, 7\}$ (from white to blue).

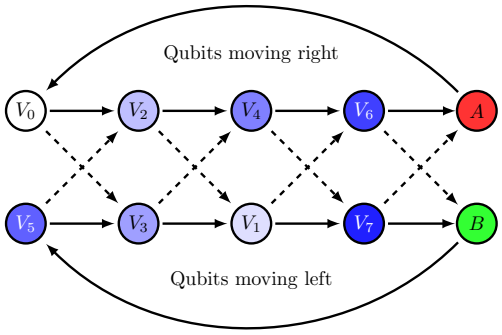


Figure 16: Qubit movement pattern during the heavy-hex swap strategy. Each node shows one of the ten categories of nodes defined in Fig. 15. The arrows show how the qubits change group after an iteration of steps 1. through 4. Solid and dashed lines show qubits that were and were not reflected at line ends during the iteration, respectively. Note that a dashed arrow from V_6 to V_5 is not shown.

We now consider groups A and B. During one iteration qubits positioned in groups A and B will switch to groups V_0 and V_5, respectively. From the outlined strategy, qubits only arrive in group A if they ended the previous iteration in a group V_i with

$$\begin{aligned} i + 2 = 0 \bmod 8 \quad &\implies i = 6 \\ \text{or} \quad 7 - i = 0 \bmod 8 \quad &\implies i = 7. \end{aligned}$$

Since the swap layer swapping with group B is executed after $k - 7$ layers of S_1 in alternation with S_2 and nodes of group B are connected with group V_4, a qubit can only arrive in group B if it was initially positioned in a group V_i with

$$\begin{aligned} i - 2 + 7 = 4 \bmod 8 \quad &\implies i = 7 \\ \text{or} \quad 7 - i + 4 + 7 = 4 \bmod 8 \quad &\implies i = 6. \end{aligned}$$

In either case, a qubit can only arrive in groups A or B if it was previously positioned in group V_6 or V_7. All possible movements of qubits among groups from one iteration to the next are thus captured by Fig. 16 and it is clear that within five iterations each qubit will only visit A or B at most once. This concludes the proof. □

B Circuit depth and CNOT gate count

We now investigate how the number of CNOT layers and gates in QAOA circuits, transpiled using the pass described in Sec. 2.1, scale with problem size. Here, $G_0 = (V, E_0)$ is either a heavy-hex or an η-dimensional grid coupling map and $\{S_i\}$ with $i \in \mathbb{N}$ is a swap strategy compatible with G_0. The transpiled circuit has alternating layers of R_{ZZ} and SWAP gates applied on edge sets E_i and S_i, respectively, see Fig. 17. We determine the number of CNOT gates and layers by counting gates and layers in E_i and S_i taking into account gate cancellations across layers.

In every layer E_i we apply R_{ZZ} gates on the edges in E_i. The number of edges in E_i thus determines the number of R_{ZZ} gates. Here, E_i is the set of edges in the hardware coupling map (i.e. $E_i \subseteq E_0$) which give new qubit connections after applying the swap layer S_{i-1}. Edges which may give new qubit connections after S_{i-i} are in the neighbourhood of swapped edges, i.e.

$$E_i \subseteq \bigcup_{e \in S_{i-1}} N(e), \tag{10}$$

as exemplified in Fig. 18. Since the SWAP gates in a swap layer S_i are executed in parallel S_i never contains two neighboring edges, i.e. $e \notin N(e') \; \forall \; e, e' \in S_i$. Equation (10) therefore implies $E_i \subseteq E_0 \setminus S_{i-1}$.

Carefully positioned CNOT gates across E_i and S_i may cancel. We therefore position all R_{ZZ} gates of E_i that are applied across edges contained in S_i at the end of E_i. This allows us to simplify two CNOT gates and implement SWAP$\cdot \mathrm{R_{ZZ}}(\theta)$ with three CNOTs and a $R_Z(\theta)$ gate, see Fig. 19. The number of CNOT layers $L_{cx}(E_i, S_i)$ required to implement E_i and S_i is

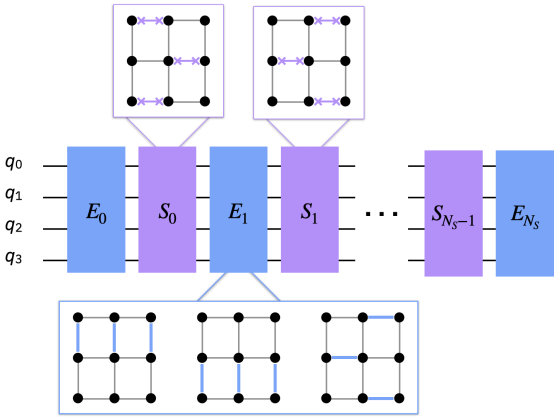


Figure 17: A QAOA cost layer transpiled with the hardware-optimized transpiler pass from Sec. 2.1. The circuit is split into blocks E_i and layers S_i of R_{ZZ} and SWAP gates, respectively, exemplified on a two-dimensional grid. Here, the last R_{ZZ} layer of E_i can be combined with S_i by the circuit identity in Fig. 19.

thus bounded by the edge chromatic number $\chi'(G_i)$ of the graph $G_i = (V, E_i \setminus S_i)$, i.e.

$$L_{cx}(E_i, S_i) \leq 2\chi'(G_i) + 3. \tag{11}$$

Here, the R_{ZZ} gates in $E_i \setminus S_i$ are divided into $\chi'(G_i)$ sets of gates to execute in parallel each with two CNOT gates. The extra three CNOT layers come from SWAP gates that may have absorbed R_{ZZ} gates. Similarly, the number of CNOT gates in E_i and S_i is

$$k_i = 2|E_i \setminus S_i| + 3|S_i| \tag{12}$$

since a swap layer will always require three CNOT layers even after absorbing R_{ZZ} gates from E_i. Since $E_i \subseteq E_0 \setminus S_{i-1}$ we may write $|E_i \setminus S_i| \leq |(E_0 \setminus S_{i-1}) \setminus S_i| = |E_0| - |S_{i-1}| - |S_i|$ where the last equality requires that S_{i-1} and S_i are disjoint. Under the same assumptions we also have $\chi'(G_i) \leq \chi'(G) - 2$. Therefore, Eq. (11) and (12) yield

Lemma 4. *If $S_i \cap S_{i-1} = \varnothing$ the number of CNOT gates k_i required to implement E_i and S_i satisfies*

$$k_i \leq 2|E_0| - 2|S_{i-1}| + |S_i|.$$

If there exists an edge coloring of the coupling map E_0 with Δ_i colors, such that one color corresponds to the set of edges contained in S_{i-1} and all edges of S_i are colored in the same color, then the number of CNOT layers $L_{cx}(E_i, S_i)$ required to implement E_i and S_i satisfies

$$L_{cx}(E_i, S_i) \leq 2\Delta_i - 1.$$

B.0.1 Grid coupling maps

We now apply Lemma 4 to the swap strategies on the line, two- and three-dimensional grids with L_S swap

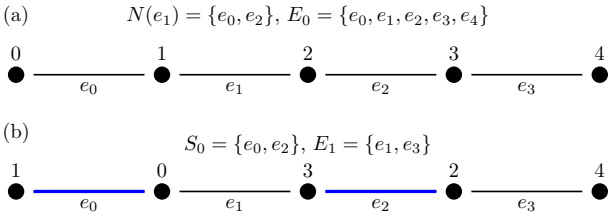


Figure 18: Explanation of the notation on a line graph. (a) A line graph with five nodes and, as example, $N(e_1)$. (b) Notation after the first SWAP layer. The set of edges that give new qubit connections is $E_1 = \{e_1, e_3\}$.

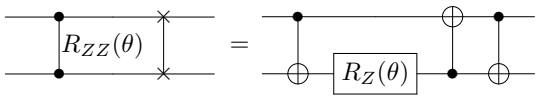


Figure 19: An R_{ZZ} and a SWAP gate combine into a gate sequence with three CNOTs and one R_Z gate.

layers. These strategies satisfy $S_i \cap S_{i-1} = \varnothing \, \forall \, i$ and have $S_{-1} = S_{L_S} = \varnothing$, see Appendix A. The number of CNOT layers in the complete cost layer is then bounded by

$$\begin{aligned} k = \sum_{i=0}^{L_S} k_i &\leq \sum_{i=0}^{L_S} 2|E_0| - 2|S_{i-1}| + |S_i| \\ &= 2|E_0|(L_S + 1) - \sum_{i=0}^{L_S} |S_i| \end{aligned} \tag{13}$$

In the line graph, every SWAP layer contains half of the edges (on average in case the number of edges is odd). More generally, for the η-dimensional grid strategy the number of swap gates is, on average,

$$|S_i| = \frac{1}{2\eta}|E_0| \text{ where } |E_0| = \eta n + \mathcal{O}(n^{\frac{\eta-1}{\eta}}). \tag{14}$$

Combining Eq. (13) and (14) yields the bound

$$k \leq \frac{4\eta - 1}{2} n(L_S + 1) + \mathcal{O}(L_S n^{\frac{\eta-1}{\eta}}) \tag{15}$$

on the total number of CNOT gates k required to implement $\exp(-i\gamma H_C)$. These bounds are summarized in Tab. 1 of the main text as $(4\eta - 1)/2nL_S$. Furthermore, since the swap layers in the grid strategy for the η-dimensional grid graph exactly correspond to an edge coloring with 2η colors, the second requirement from Lemma 4, i.e. the existence of the edge coloring, also holds with

$$\Delta_i = \begin{cases} 2\eta & 0 < i < L_S \\ 2\eta + 1 & i = 0. \end{cases}$$

Together with the $2(2\eta - 1)$ CNOT layers required to implement the final layer E_{L_S}, this bounds the total number of CNOT layers L_{cx} of a cost layer with L_S

swap layers following

$$L_{\text{cx}} \leq 2(2\eta - 1) + \sum_{i=0}^{L_S - 1} 2\Delta_i - 1$$
$$= (4\eta - 1)(L_S + 1) + 1.$$

Plugging in $\eta = 1, 2, 3$ gives the numbers in Tab. 1 of the main text.

B.0.2 Heavy-hex coupling maps

We now consider heavy-hex graphs with the same number of rows and columns and with n qubits. This graph has $|E_0| = 6n/5 + \mathcal{O}(\sqrt{n})$ edges and a longest line with length $l_{max} = 4n/5 + \mathcal{O}(\sqrt{n})$, as exemplified in Fig. 14. Thus, each SWAP layer S_i consists of at most $2n/5 + \mathcal{O}(\sqrt{n})$ SWAP gates. It also holds that

$$|E_i \setminus S_i| + |S_i| \leq \frac{3}{5}n + \mathcal{O}\left(\sqrt{n}\right)$$

for all $0 < i < L_S$. Using Eq. (12), we can therefore bound the number of CNOT gates k_i in the combined layers E_i and S_i for $0 < i < L_S$ by

$$k_i \leq \frac{8}{5}n + \mathcal{O}\left(\sqrt{n}\right).$$

E_0 and S_0 contain at most $14n/5 + \mathcal{O}(\sqrt{n})$ and E_{L_S} at most $12n/5 + \mathcal{O}(\sqrt{n})$ CNOT gates, yielding a bound for the total number of CNOT gates k

$$k \leq (4L_S + 9)\frac{2}{5}n + \mathcal{O}(\sqrt{n}).$$

Furthermore, every layer E_i can be executed with 2 individual R_{ZZ} layers. However, it is not possible to cancel out every gate in any of the resulting R_{ZZ} layers with the subsequent swap layer S_i. As a result, we get the following bounds for the total depth and number of CNOT layers d and d_{CNOT} in the heavy-hex case

$$d \leq 9L_S + 10$$
$$d_{CNOT} \leq 7L_S + 6$$

C Commutation aware SabreSwap

Quantum circuits can be represented as directed acyclic graphs (DAG) in which nodes are instructions and edges are qubits. The SabreSwap algorithm traverses the DAG. It first builds up a front layer with the gates in the DAG that are first executed on the qubits. Next, the algorithm inserts SWAPs attempting to minimize the distance between qubits that interact in the front layer. When two qubits become adjacent the gate from the front layer is inserted and the front layer is updated with the next gates from the DAG.

In the commutation aware version of SabreSwap, the front layer is adapted to contain all upcoming commuting gates obtained from the dependency DAG. The DAG dependency is built by taking commutation relations into account [106]. Here, two gate nodes are only connected by an edge if the corresponding gates do not commute.

D Maximum cut

In the weighted maximum cut problem we are tasked to partition the set of nodes V of a given graph $G = (V, E)$ in two such that the sum of the edge weights $\omega_{i,j}$ with $(i, j) \in E$ traversed by the cut is maximum [88]. This can be formulated as the QUBO problem

$$\max \frac{1}{2} \sum_{(i,j)\in E} \omega_{i,j}(1 - z_i z_j),$$
$$\text{such that} \quad z \in \{-1, 1\}^{|V|}.$$

The binary variable z_i indicates which side of the cut node i is. In Sec. 4 we consider weighted maximum cut instances in which the weights can be negative as well as positive [107].

E Swap strategy benchmark

This section provides additional details on the swap strategy benchmark presented in the main text. For SabreSwap, commutative aware SabreSwap, the QAOA Compiler, 2QAN, and t|Ket⟩ we generate the `qaoa_circuit` using R_{ZZ} gates based on the problem graph. For the swap strategies we create the `qaoa_circuit` as a single instruction built from the `PauliSumOp` in Qiskit which the transpiler must identify as being made of commuting two-qubit Pauli terms. In all cases the transpiler must map the instructions to hardware native CNOT gates. The

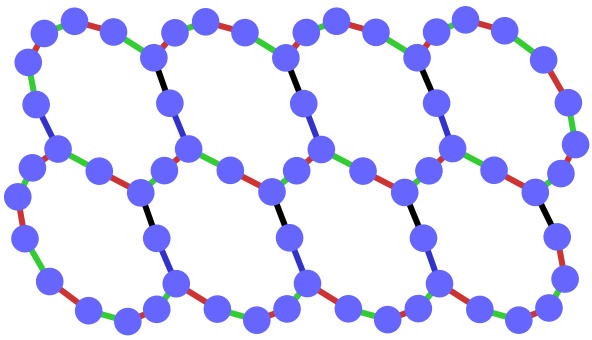


Figure 20: Heavy-hex graph with two rows and four columns. The color of the edges shows the edge coloring used to prioritize the R_{ZZ} gates which can be simultaneously applied. The order of the applied gates is red, green, blue, and black. Furthermore, red and green edges form the longest line.

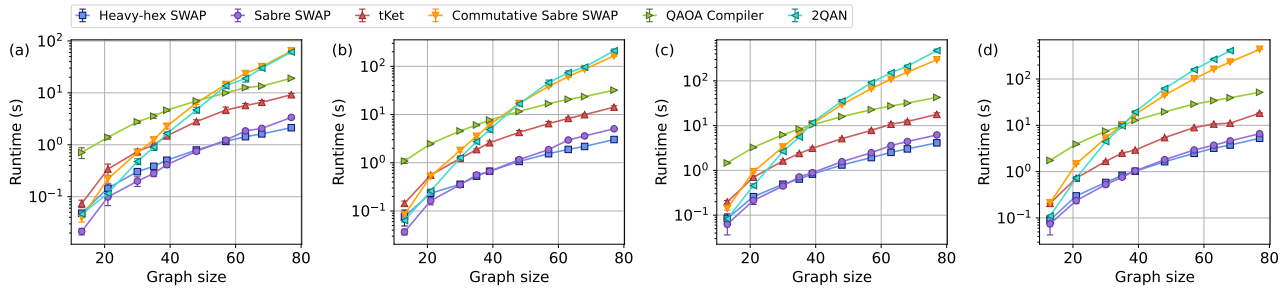


Figure 21: Logarithmic transpiler runtime for QAOA circuits of graph instances with different size and density after transpilation to a heavy-hex coupling map using SabreSwap, commutative aware SabreSwap, $t|\text{Ket}\rangle$, 2QAN, and the heavy-hex swap strategy. Each transpiler was allowed a maximum runtime of ten minutes. Each data point is an average over ten random graphs. The lines show the average and the error bars show the standard deviation. (a), (b), (c), and (d) correspond to graphs with densities of 0.25, 0.5, 0.75 and 1.0, respectively.

CNOT circuit depth shown in the main text is computed using the Qiskit function

```
circuit.depth(
    lambda x: isinstance(x.operation, CXGate)
)
```

We now list the key passes with which we benchmark the different transpilers. We obtain the execution time of each transpiler with `time` as exemplified for t|Ket⟩. The t|Ket⟩ transpiler steps are listed below.

```
tk_circ = qiskit_to_tk(qaoa_circuit)

cmap = Architecture(coupling_map.get_edges())

start = time.time()
placement = GraphPlacement(cmap)
CXMappingPass(cmap, placement).apply(tk_circ)
KAKDecomposition(allow_swaps=False).apply(tk_circ)
final = time.time()

final_circuit = tk_to_qiskit(tk_circ)
```

The QAOA Compiler, described in Ref. [38], performs an initial mapping of QAOA and routs the resulting circuit to the hardware using an external compiler. The external compiler is called on partial circuits and the results are then stitched together. We run the QAOA Compiler with Qiskit on optimization level 3 as external compiler and use the Variation-aware Incremental Compilation (VIC) setting described in Ref. [38]. We therefore create an instance of `CompileQAOAQiskit` and call its `run_incr_c` method with `variation_aware=True` and `initial_layout_method="qaim"`. The configuration used to initialize the `CompileQAOAQiskit` instance is

```
config = {
    "Backend": "qiskit",
    "Target_p": "1",
    "Packing_Limit": "10e10",
    "Route_Method": "sabre",
    "Trans_Seed": "0",
    "Opt_Level": "3",
}
```

To benchmark the swap strategies in Qiskit we use the pass manager

```
pm = PassManager([
    FindCommutingPauliEvolutions(),
    Commuting2qGateRouter(swap_strat, edge_color),
    Decompose(),
    Decompose(),
    CXCancellation(),
])
```

Here, the swap strategy contains the information on the coupling map and the double `Decompose` replaces the routed two-qubit Pauli evolution operators by CNOT gates. The `edge_color`, exemplified in Fig. 20, is used to prioritize which R_{ZZ} gates should be applied first at each step of the swap strategy. This increases the number of R_{ZZ} gates that are followed-by SWAP gates such that the `CXCancellation` pass can cancel consecutive CNOT gates as shown in Fig. 19. For SabreSwap we replace the `FindCommutingPauliEvolutions` and `Commuting2qGateRouter` with the single pass

```
SabreSwap(
    coupling_map=coupling_map,
    heuristic='basic'
)
```

The 2QAN [36] benchmark, shown below, uses the `QuRouter` imported from the `py2qan` package with a trivial initial mapping.

```
router = QuRouter(
    qasm=qaoa_circuit.qasm(),
    init_map={idx: idx for idx in range(n)},
    coupling_map=coupling_map.get_edges(),
)

circ_2qan, _ = router.run_qaoa(
    layers=1,
    gammas=[0.5],
    betas=[0.75],
    msmt=True
)
```

In addition to the circuit depth and CNOT gate count presented in Fig. 5 of the main text we also compute the time it takes to transpile the corresponding circuits. We find that the commutative aware version of SabreSwap and 2QAN have the runtime with the worst scaling while the swap strategies and SabreSwap are up to two orders of magnitude faster, see Fig. 21.

F Experiment details

The seven node graph used in the Qiskit Runtime example can be embedded with one swap layer $S_0 = \{(0,1),(3,5)\}$ on the seven qubit *ibm_nairobi* system. The weights of the graph were chosen at random from $\{-1,1\}$. The seven node graph given as sums of Pauli-Z operators is $\mathcal{G}_{10} = -Z_1Z_0 + Z_2Z_1 - Z_4Z_3 - Z_5Z_4 - Z_2Z_0 + Z_6Z_5 + Z_5Z_0 + Z_6Z_3 - Z_3Z_1 + Z_5Z_3$ and has a single maximum cut with a value of three, see Fig. 22. The properties of the hardware and the gates as reported by *ibm_nairobi* on the date the data were acquired are given in Tab. 5 and 6 .

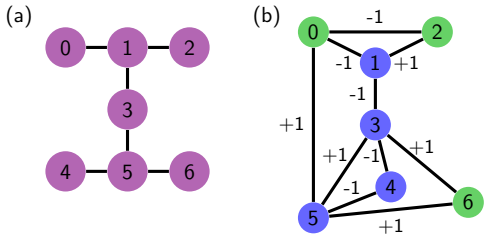


Figure 22: (a) Coupling map of *ibm_nairobi* and (b) the graph $\mathcal{G}_{10}$. The node coloring shows the maximum cut.

Qubit	T_1 (μs)	T_2 (μs)	$\sqrt{X}$ Error $\times 10^{-4}$	RO Error %
0	124	61	2.5	2.1
1	100	132	6.2	3.2
2	120	158	1.7	2.1
3	91	65	4.0	3.8
4	67	83	2.9	2.4
5	132	44	3.3	2.9
6	165	116	2.2	2.8

Table 5: Hardware properties of ibm_nairobi on 22.12.2021 as reported by the backend. RO stands for readout error.

Qubit pair	CNOT Error	CNOT length	Qubit pair	CNOT Error	CNOT length
(0, 1)	1.2	249	(1, 2)	0.5	391
(4, 5)	0.6	277	(5, 6)	0.8	306
(1, 3)	0.8	270	(3, 5)	1.2	235

Table 6: Two-qubit gate error and length in % and ns, respectively, reported by *ibm_nairobi* on 22.12.2021.

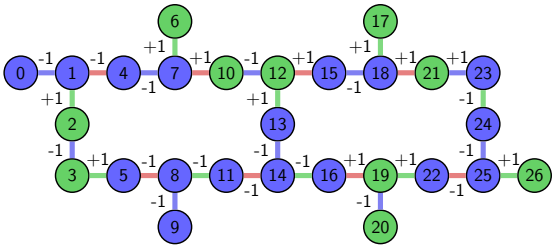


Figure 23: Coupling map of *ibmq_mumbai*. The edge coloring indicates which CNOT gates can be applied in parallel and the edge weights are those of the graph of the 27 node maximum cut problem. The node coloring shows the maximum cut, found with CPLEX, which has a value of 12.

The 27 node graph is native to the coupling map of *ibmq_mumbai*, see Fig. 23. The results presented in the main text use CVaR aggregation with $\alpha = 0.5$, i.e., the expectation value of H_C is computed from the best 50% of the shots. Without CVaR aggregation we observe a noisy convergence at depth-one and no convergence at depth-two, see Fig. 24. We allow SPSA 100 iterations at depth-one and 80 iterations at depth-two to avoid exceeding the maximum allowed time for a Qiskit Runtime program. When SPSA terminates Qiskit returns the last values of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ and samples the state produced by the corresponding circuit. For the 27 qubit problem instance with depth-two the optimizer does not converge properly. We therefore resample the circuit five times, each with 2^{14} shots, at the $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ values that yielded the minimum energy. The average energy -16.57 ± 0.16 of the five runs is shown as a star in Fig. 12(a) and the best counts distribution is shown in Fig. 12(b) of the main text. The seven qubit problem on *ibm_nairobi* calibrates a readout-error mitigation matrix with 2^7 circuits. We therefore do not use readout error mitigation on the 27 qubit problem and leave it up to future work to investigate the effect of scalable readout error mitigation such as M3 on QAOA [105]. The properties of the hardware and the gates as reported by *ibmq_mumbai* on the date the data were acquired are given in Tab. 7 and 8.

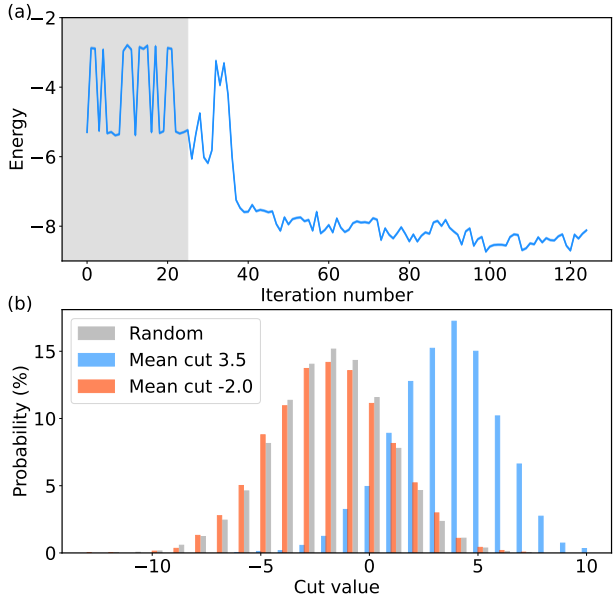


Figure 24: QAOA without CVaR on the graph native to the hardware of *ibmq_mumbai*. At each energy evaluation 16384 shots are gathered. (a) Energy as function of the iteration number for depth-one QAOA initialized at $\gamma = 1$ and $\beta = 0.5$. In the first 25 iterations (gray area), SPSA calibrates the learning rate and perturbation based on the measured energy. (b) Probability distribution of the best measured energy for depth-one (blue) compared to random sampling (gray) and depth-two (red).

Qubit	T_1 (μs)	T_2 (μs)	$\sqrt{X}$ Error $\times 10^{-4}$	RO Error %
0	99	23	2.7	4.0
1	168	171	1.9	2.1
2	100	101	1.9	2.8
3	119	192	2.4	1.9
4	115	39	2.4	1.6
5	121	137	2.9	1.8
6	116	46	2.5	1.5
7	170	71	2.4	2.5
8	155	80	2.5	4.7
9	69	98	6.0	2.4
10	131	222	2.0	1.2
11	96	88	1.8	3.0
12	215	313	1.7	1.8
13	165	40	1.6	3.7
14	169	337	3.5	2.9
15	65	84	2.3	2.1
16	193	129	1.5	2.3
17	135	178	2.2	1.5
18	194	217	1.2	1.4
19	211	272	1.7	1.3
20	99	149	1.9	1.8
21	65	81	3.5	2.9
22	172	87	2.4	2.0
23	109	30	4.8	11.9
24	148	62	1.7	1.9
25	180	71	1.6	1.5
26	140	241	2.0	1.8

Table 7: Hardware properties of *ibmq_mumbai* on 26.07.2022 as reported by the backend. RO stands for readout error.

Qubit pair	CNOT Error	CNOT length	Qubit pair	CNOT Error	CNOT length
(0, 1)	0.7	420	(1, 2)	1.5	704
(1, 4)	1.0	348	(2, 3)	0.7	391
(3, 5)	–	356	(4, 7)	0.9	590
(5, 8)	2.3	1316	(6, 7)	0.7	249
(7, 10)	0.7	398	(8, 9)	–	718
(8, 11)	4.0	590	(10, 12)	0.6	363
(11, 14)	1.9	370	(12, 13)	0.6	548
(12, 15)	0.6	370	(13, 14)	1.3	320
(14, 16)	1.4	526	(15, 18)	0.6	306
(16, 19)	1.9	512	(17, 18)	0.5	249
(18, 21)	0.7	476	(19, 22)	0.6	327
(19, 20)	0.5	370	(21, 23)	1.0	391
(22, 25)	0.6	448	(23, 24)	3.6	626
(24, 25)	0.5	434	(25, 26)	0.5	313

Table 8: Two-qubit gate error and length in % and ns, respectively, reported by *ibmq_mumbai* on 26.07.2022.