QPU-System Co-Design for Quantum HPC Accelerators*

Karen Wintersperger[1][0000−0002−2181−1860], Hila Safi[1,2], and Wolfgang Mauerer[2,1][0000−0002−9765−8313]

[1] Siemens AG, Corporate Technology,
Otto-Hahn-Ring 6, 81739 München, Germany
{karen.wintersperger,hila.safi}@siemens.com
[2] Technical University of Applied Sciences Regensburg,
Galgenbergstraße 32, 93053 Regensburg, Germany
wolfgang.mauerer@othr.de

Abstract. The use of quantum processing units (QPUs) promises speedups for solving computational problems, but the quantum devices currently available possess only a very limited number of qubits and suffer from considerable imperfections. One possibility to progress towards practical utility is to use a co-design approach: Problem formulation and algorithm, but also the physical QPU properties are tailored to the specific application. Since QPUs will likely be used as accelerators for classical computers, details of systemic integration into existing architectures are another lever to influence and improve the practical utility of QPUs.
In this work, we investigate the influence of different parameters on the runtime of quantum programs on tailored hybrid CPU-QPU-systems. We study the influence of communication times between CPU and QPU, how adapting QPU designs influences quantum and overall execution performance, and how these factors interact. Using a simple model that allows for estimating which design choices should be subjected to optimisation for a given task, we provide an intuition to the HPC community on potentials and limitations of co-design approaches. We also discuss physical limitations for implementing the proposed changes on real quantum hardware devices.

1 Introduction

Quantum computers available today are restricted in their performance by relatively small ($\approx$ 50–100) number of quantum bits (qubits), and suffer from various imperfections. Since they cannot yet implement error correction routinely, they classify as NISQ (Noisy Intermediate Scale Quantum) hardware [1], whose capabilities are subject to ongoing exploration. Nevertheless, there is a growing interest to deploy NISQ devices in high-performance computing (HPC) scenarios. [2]

* All authors acknowledge funding from the German Federal Ministry of Education and Research within the funding program quantum technologies—from basic research to market, contract number 13N16093.

Fitting complex problems to NISQ devices usually requires many simplifications. Also, the mapping of the problem to a quantum model and the algorithm can be optimised to reduce both, the number of necessary qubits, and the number of quantum operations [3]. Properties of quantum algorithms strongly depend on specific QPUs. Many factors, size (number of qubits), the geometric arrangement of and number of connections between the qubits, as well as the specific errors and execution times of quantum operations, influence quantum circuit execution. Prior to running a quantum circuit on a quantum device, the circuit structure must be adjusted to its requirements, which usually increases the number of quantum operations, and thus the circuit depth. Co-designing QPUs by adapting them to specific problem classes is therefore a promising approach.

QPUs will likely be used as accelerators for classical computers, and require integration with classical hardware. Moreover, hybrid algorithms that combine classical and quantum operations are a commonly occurring pattern. Thus, the interaction between quantum and classical devices needs to be taken into account to estimate the performance of a quantum program in practice.

In this work, we investigate optimisation potentials for the co-design of such CPU-QPU-systems, exemplified by a hybrid quantum algorithm used to solve the maximum cut (Max-Cut) problem. In detail, our contributions are as follows:

- We analyse the properties of compiled quantum circuits for various instances of the Max-Cut problem when modifying geometric properties of the QPU, namely the connectivity between qubits and the number of qubits.
- We estimate the runtime on real quantum hardware based on simulations, and investigate the overall runtime on QPU-systems including the communication between classical and quantum machines, and other classical calculations required to execute and evaluate quantum circuits.
- Based on these results, we give recommendations for the design of quantum hardware adapted to applications.
- We provide a self-contained replication package [4] for the simulations that is available at https://github.com/lfd/arcs2022.git.

The Max-Cut problem is an optimisation problem with applications in network design, clustering or statistical physics. Considering an undirected graph with a set of nodes V and a set of edges E, a cut is defined as a partition of the node set into two subsets. The Max-Cut problem seeks to find a partition such that the number of edges connecting the two subsets is maximised. Many applications for this primitive can be found in all areas of computing; for us, it suffices to state that Max-Cut serves as typical representative for hard problems that we will further motivate in Section 2.2.

There are different, yet equivalent models for quantum computation, such as gate-based [5, 6], measurement-based [7], adiabatic [8], or topological quantum computing. In this work, we focus on gate-based quantum computation. The Max-Cut problem is solved using the Quantum Approximate Optimisation Algorithm (QAOA) [9], which is a widely used variational hybrid quantum algorithm for solving combinatorial optimisation problems on NISQ hardware.

As a starting point for the investigation of quantum hardware designs, IBM-Q devices are used, which are based on superconducting qubit technology [10, 11]. Superconducting qubits are one of the most common and advanced quantum hardware platforms, used by many vendors such as IBM, Google and Rigetti. Since superconducting qubits are artificial quantum systems, they can in principle be designed at will. By now, several different types of superconducting qubits exist, and the technology is continuously developed. However, this also means that no two qubits are completely identical, and properties such as the gate fidelity, which is a measure for the quality of a quantum operation, differ for each qubit. The superconducting quantum devices of different vendors usually differ by the geometric arrangement of qubits, and the number and structure of connections between them. The IBM-Q topology that we use as basis for our considerations is described in detail in Section 2.4.

2 Quantum Max-Cut with QAOA

Before we discuss co-design optimisation possibilities, we need to set the stage for the considered problem, and illustrate the solution algorithm.

2.1 The Quantum Approximate Optimisation Algorithm

QAOA implements a quantum circuit consisting of $p \in \mathbb{N}$ layers of unitary operators (the elementary type of operation a quantum computer can effect on qubits) whose properties are specified by a set of $2p$ parameters $\vec{\beta}, \vec{\gamma} \in \mathbb{R}^p$. The algorithm can determine minima of objective functions specified in quadratic, unconstrained binary form; these are specified such that the minimum solution corresponds to a solution of a specific problem of interest. Using well-known techniques from computer science, all problems in **NP** can be reduced to *Quadratic Unconstrained Binary Optimisation* (QUBO) form [12]. Speedups of QAOA compared to classical approaches are not yet fully understood [13]; yet the existence of a classical algorithm that efficiently samples the output distribution of QAOA even for $p = 1$ is impossible, given reasonable complexity-theoretic assumptions [14]. This is seen as likely indicator for quantum advantage, but theoretical or experimental progress is required to find utility on practical problems.

Fig. 1 sketches the structure of the QAOA circuit for $p = 2$: After applying the operators to a well-defined initial state, the expectation value of H_P, which encodes the objective function, is measured in the final state. Using a classical optimiser, the parameters of the circuit are changed with the goal of minimising the expectation value of H_P. Each layer i consists of two different kinds of unitaries: First, $U(\beta_i) = e^{-i\beta H_B}$ is applied, implementing the evolution under a so-called mixer Hamiltonian H_B. The mixer Hamiltonian is commonly chosen as a superposition of X-rotations applied to each qubit, thus consisting of a series of rotation gates $R_X(\beta_i)$. The second part of each layer comprises $U(\gamma_i) = e^{-i\gamma H_P}$ consisting of single-qubit Z-rotations $R_Z(\gamma_i)$ and two-qubit

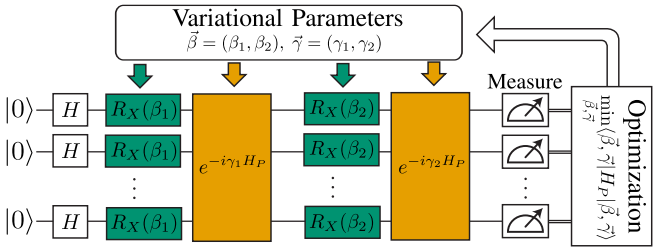


Fig. 1: Sketch of the QAOA-circuit for $p = 2$.

rotation gates $R_{ZZ}(\gamma_i)$. The repeated application of several QAOA layers corresponds to the discretised time evolution governed by the Hamiltonians H_P and H_B. It is known that the quality of the approximation increases for a larger number of layers [9]. The initial state of the QAOA algorithm is usually chosen as the ground state of H_B, in which each qubit is in an equal superposition of $|0\rangle$ and $|1\rangle$, prepared using a layer of Hadamard gates H.

To characterise the probability distribution of the final state depending on $\vec{\beta}, \vec{\gamma}$ after each iteration of the optimiser, the quantum circuit is executed several times (known as *sampling*), and the qubits are measured in the computational basis $\{|0\rangle, |1\rangle\}$. The mean of the expectation values of H_P for each measurement outcome is used as the objective function minimised by the classical optimiser. The optimal solution is then given by the state (or bit string) with the lowest energy expectation value taken from the probability distribution obtained for the final set of parameters.

2.2 Background on Max-Cut and QAOA

Given a graph $G = (V, E)$ consisting of a set of vertices V and a set of edges $E \subseteq V \times V$, the Max-Cut problem is a seminal graph-theoretic task that seeks two subsets $V_0, V_1 \subseteq V$ such that $V_0 \cup V_1 = V$ and $V_0 \cap V_1 = \varnothing$, and maximises the size $|C|$ of the cut set $C = \{(u, v) \in E : u \in V_0, v \in V_1\}$ (as a decision problem, the k-Cut variant seeks a cut with $|C| = k$). While the Max-Cut problem is very simple to formulate, it counts among the hardest optimisation problems to solve [15]. It is textbook knowledge that the decision variant lies in complexity class **NP**, while the optimisation variant is **APX**-hard, which essentially means that any polynomial-time approximation algorithm can at best find solutions whose approximation ratio (*i.e.*, the size of the found cut size divided by the optimal cut size) is bounded by a constant for general graphs.

Classical approximation algorithms and heuristics for Max-Cut, including formulations adapted to graphs with specific properties, have been studied in the literature, and polynomial-time approximation algorithms with non-trivial performance guarantees are known. In particular, the seminal algorithm of Goemans and Williamson [16] achieves an approximation ratio of 87.86% on generic graphs, while Khot *et al.* discuss the optimality of inapproximability results for

the problem. Ever since Farhi *et al.* [9] found that QAOA at circuit depth $p = 1$ (a) already achieves an approximation ratio of at least 69.25% (for the class of uniform, 3-regular graphs), yet (b) cannot be efficiently simulated by classical algorithms, assuming the validity of widely accepted complexity-theoretic hypotheses, there has been a steady interest in understanding properties and performance guarantees of QAOA on the Max-Cut problem.

Fuchs *et al.* [17] present an efficient encoding of (weighted) Max-Cut for QAOA, while Wurz and Lykov [18] discuss open conjectures regarding the quantum performance of the problem. On the negative side from the QPU point of view, Marwaha [19] showed that classical algorithms outperform QAOA with $p = 2$ with a certain type of graphs. On the positive side for quantum algorithms, Wurtz and Love [20] give performance guarantees for QAOA-Max-Cut for $p > 1$ in terms of an approximation ratio of 75.59%. Recent approaches solve Max-Cut with coherent networks [21], and also go beyond standard quantum hardware. A comprehensive evaluation of quantum annealing performance on different QPUs for the Max-Cut problem is provided by Willsch *et al.* [22].

More general investigations of QAOA are plentiful; Barthi *et al.* [13] summarise many of them. For instance, Xue *et al.* [23] consider the effect of noise on QAOA performance, while Yu *et al.* [24] provide an automatic depth optimisation technique. At present, a widely accepted and empirically observed, yet not fully understood hypothesis claims a concentration of the optimisation parameters $\vec{\beta}, \vec{\gamma}$ on relatively low-dimensional sub-manifolds of the possible search space. Akshay *et al.* [25] report positive results in this directions, and Zhou *et al.* [26] give a concrete construction to benefit from this concentration to improve the algorithmic performance of the classical component of the hybrid algorithm (their FOURIER construction is numerically conjectured to produce quasi-optima in time $\mathcal{O}(\text{poly}(p))$ instead of $2^{\mathcal{O}(p)}$ for the standard QAOA algorithm). Additionally, several extensions to the QAOA algorithm have been proposed, for instance for quantum alternating operator ansatz by Hadfield *et al.* [27], or the mixer-phaser ansätze by LaRose *et al.* [28].

However, to the best of our knowledge, we are not aware of any discussion on how to improve performance and feasibility of Max-Cut on quantum computers (QCs) using co-design, and how to holistically evaluate potential performance benefits including the overhead by unavoidable classical computing components beyond the optimisation algorithms employed in the hybrid approach. We discuss these issues in this paper, and believe they hold potential for more general insights on how to achieve first practical speedups for quantum algorithms given realistic systemic constraints and boundary conditions.

2.3 Modelling Max-Cut as QUBO

Following the seminal collection of transformations provided by Lucas [12], the Max-Cut problem can be cast as a QUBO using binary variables x_i with $x_i = 1$ if node i lies within the first subset, and $x_i = 0$ if it lies in the second subset. If an edge connecting the nodes i and j is part of the cut, thus connecting the two subsets, exactly one of x_i and x_j is equal to zero and the other one is equal

to one. In this case, $H_{i,j} = (x_i + x_j - 2x_i x_j)$ equals one and in the cases where $x_i = x_j$ it equals zero. Finding the maximum cut corresponds to maximising the sum of $H_{i,j}$ over all edges of the underlying graph, or, equivalently, minimising the sum over $-H_{i,j}$. In the following, the latter approach will be used. Thus, the optimal solution of the Max-Cut problem can be encoded as the ground state of the Hamiltonian $H_P = \sum_{i,j \in E}(2x_i x_j - x_i - x_j)$, which is passed to the generic QAOA algorithm as objective function to determine a minimum solution.

2.4 Setup

The problem graphs $G = (V, E)$ subjected to Max-Cut are characterised by the number of nodes, $N = |V|$, and the graph density defined as $d = |E|/|E_{\max}| \in [0, 1]$, where $|E|$ is the number of edges of G, and $|E_{\max}| = n(n-1)/2$ is the number of edges of a clique comprising $|V|$ nodes, which upper bounds the possible number of edges in G. Each node is represented by one qubit.

To run a quantum circuit on a QPU, it has to be compiled to meet the requirements of the hardware. The process of compilation consists of several steps and accounts for the limited connectivity between the qubits as well as for the native gate set, which describes the set of gates that can be executed on the specific hardware. Both properties depend on the chosen technology. For instance, quantum devices such as ion traps exhibit all-to-all connectivity, whereas others do not. If the circuit contains gates which are not part of the native gate set, they are decomposed accordingly. Missing connections between qubits are countered by adding so-called SWAP-gates, which themselves often need to be decomposed again into C–X gates, for instance when using the IBM-Q gate set considered here. All of these steps increase the depth of the circuit. Within this work, creation, compilation and simulation of quantum circuits was implemented using Qiskit. For compilation (which is called transpilation in Qiskit), we use a predefined routine of Qiskit. It consists of the following steps:

1. Virtual circuit optimisation, for instance, parallelisation of gates
2. Decomposition of gates containing three qubits or more into two-qubit gates
3. Placement of the virtual qubits on the physical qubits
4. Routing on coupling map, introduction of necessary SWAP gates
5. Translation to native gate set
6. Optimisation of the resulting physical circuit

Circuit optimisation can be executed at different levels. For all investigations in this work, the optimisation level was set to the maximal value of 3, which describes heavy optimisation, including also re-synthesis of two-qubit blocks. The placement of the SWAP gates is performed using a stochastic method, called stochastic SWAP, which leads to different compilation results for the same initial circuit. Therefore, we repeat the compilation process 20 times for each parameter set and consider the mean of the circuit depth over these values.

The base coupling map is derived from the IBM-Q Washington backend with 127 qubits by adding two connections which are missing in the original hardware.

The qubits are arranged in the so-called heavy-hex lattice geometry illustrated in Fig. 6a. The native gate set of IBM-Q hardware is used: Rotation R_{Z}, phase shift S_{X}, Pauli (Not) X, and controlled X (C–X).

NISQ QPUs suffer from limited gate fidelities and noise. These effects could also be included when simulating and compiling quantum circuits. Within the compilation process, the placement of the virtual qubits could be adapted to the differences in noise characteristics between the qubits, which occur for superconducting quantum devices, minimising the overall effect of noise. However, the limited available space does not allow us to consider these aspects.

3 Hardware-System Co-Design

3.1 Optimisation Potentials

QPUs require integration with classical computers to solve problems [29], regardless if hybrid or "pure" quantum algorithms are used. This, invariably, induces temporal overheads that are usually ignored when studying the complexity-theoretic performance of quantum algorithms. However, especially for NISQ devices that are unlikely to produce exponential speedups, such details cannot be ignored to judge potential gains by quantum technology [30]. The time required to execute an algorithm on a QPU that interacts with a CPU comprises several contributions: (a) Circuit execution time t_{circ} on the QPU (also considering the number n_{samp} of samples required to obtain accurate statistics), (b) time t_{meas} for performing measurements of quantum states, (c) classical parameter optimisation time t_{opt} on the CPU, (d) the amount of optimisation iterations n_{iter}, and (e) time for passing input parameters and output results between QPU and CPU t_{comm}. A straightforward model to describe the execution time required for hybrid algorithms like QAOA is therefore given by

$$T = n_{\mathrm{iter}} \cdot [n_{\mathrm{samp}} \cdot (t_{\mathrm{circ}} + t_{\mathrm{meas}}) + t_{\mathrm{opt}} + t_{\mathrm{comm}}]. \tag{1}$$

3.2 Parameter Estimation on IBM-Q hardware

To evaluate optimisation potentials using the model in Eq. (1), we need to determine performance values on QPUs. Obviously, these parameters not only depend on size and possibly structure of the input instance, but also on the underlying execution platform. To illustrate the *relative* influence of the factors for a typically sized instance, it suffices to consider one set of parameters, for which we chose a graph with 20 nodes (amounting to 20 qubits) and a graph density of 0.5 solved using QAOA with a single layer.

In the following, we estimate the runtime of such a QAOA circuit using a custom routine that pre-compiles the logical circuit to the gate set provided by IBM-Q hardware, and then executes the result on a simulator. We deliberately do not resort to physical hardware to avoid any degradation by noise, as our goal is to find optimisation potentials for QPU-system co-design, not to evaluate limitations of current NISQ devices. The QAOA circuit is parameterised by $\vec{\beta}$

and $\vec{\gamma}$, which are optimised using the COBYLA routine provided by Qiskit. As an objective function, we compute the mean expectation value of H_P, as described in Sec. 2.3. (see the replication package for details). We sample the quantum circuit using $n_{\text{samp}} = 1,024$ shots in each iteration. We find $t_{\text{opt}} = 159\mu\text{s}$, $t_{\text{meas}} = 28.6\mu\text{s}$, and $n_{\text{iter}} = 25$.

A quantum circuit comprises different gates, and the time it takes each gate to operate is denoted as *gate time*. To obtain an estimate of the distribution of classical and quantum contributions to the total algorithmic runtime, we estimate the quantum circuit execution time t_{circ} from the (known) hardware gate times and the structure of the circuit.

Approximate values for the gate times as provided by the Qiskit *mock backend* FakeBrooklyn are used. In case of C–X gates, different values arise for different possible qubit pairs, and we consider the average value. The execution times for the single qubit gates are identical for all qubits. Table 1 lists the concrete gate times. The execution time of the circuit calculated from the gate times is $t_{\text{circ}} = 120 \pm 20\mu\text{s}$, averaged over 20 circuit transpilation runs.

Table 1: Gate times from backend FakeBrooklyn.

Gate	Exec$^{\underline{\text{n}}}$ time [ns]	Std. Dev.
R_Z	0	0
S_X	35.56	0
X	35.56	0
C–X	370	80

The CPU-QPU communication time t_{comm} depends on how QPUs are deployed, and may vary over several orders of magnitude. We consider three different scenarios in Table 2: (a) access to a QPU via cloud services, with a communication round-trip time of about 50ms; (b) a QPU with direct attachment to the local CPU (for instance via direct LAN connection), for which we expect $t_{\text{comm}} \approx 1\text{ms}$, and an integrated system comprising a QPU and CPU that communicate via an internal system bus with $t_{\text{comm}} \approx 25\mu\text{s}$.[4] Assume that, as in many scenarios of practical interest, we deal with a "typically" sized problem instance, for which we can assume that n_{iter}, n_{samp} and t_{opt} are approximately independent of the specific input. This leaves the communication time t_{comm} and the circuit time t_{circ} as candidates for optimisation. With n_{iter}, n_{samp} and t_{opt} constant, they linearly contribute to the growth of the total execution time T, and the corresponding slopes are 25 for t_{comm} and 25,600 for t_{circ}. The magnitude of the latter slope would seem to indicate that reducing communication times holds greater optimisation potential than decreasing circuit execution times. Alas, as Table 2 indicates, moving

[4] The given communication times are rough estimates supposed to *illustrate* optimisation potentials and relative parameter influence. We obtained the numbers by measuring typical `ping` durations in cloud and local network scenarios, and estimate QPU-CPU communication time by the round-trip time of an inter-processor-interrupt in a RiscV-system, given the assumptions that a QPU-CPU SoC design will likely be based on modifiable classical architectures, and that communication times between QPU and CPU are similar to inter-core communication times.
Owing to the restricted space, we do not provide more fine-grained and realistic estimates of and models for these quantities, but remark that they would very likely not substantially change our findings and conclusions.

from a local QPU deployment to on-chip integration is much less beneficial than moving from a cloud deployment to local operation of QPUs.[5]

Table 2: Communication and total execution time T for QPU deployment scenarios.

Scenario	t_{comm}	T
Cloud	50ms	5.07s
Local Bus	1ms	3.84s
SoC[3]	25μs	3.82s

However, when we consider the "Local Bus" scenario and assume that the circuit execution time can be reduced from 120μs to 30μs (see Fig. 3 and later explanations for a rationale), the overall execution time reduces from $T = 3.84$s to $T = 1.28$s, which indicates substantial potential for optimisation using hardware adaption. Comparing the baseline scenario (cloud + standard topology) with the co-design results (local communication, adapted topology), we find a total execution time reduction of $1 - 1.28/5.07 \approx 75\%$, which may substantially impact practical scenarios, especially given that accelerators often solve the same primitive repeatedly in inner loops.

3.3 Physical Possibilities and Limitations

Different properties of QPUs can be considered to reduce circuit execution times. On the one hand, there is the geometric layout of the qubits and their connectivity. On the other hand, the native gate set, as well as the fidelities and execution times of the gates influence the performance of algorithms, as well as the effects of noise. Depending on the hardware platform, the gate times and fidelities can differ for each individual (pair of) qubit(s), as it is the case for superconducting qubits. In this work, we focus on the effects of the qubit connectivity and the number of qubits available on the device compared to the problem size.

Changing the qubit connectivity for superconducting devices necessitates to physically re-wire qubits. Moreover, issues such as cross-talk can occur if the connectivity increases too much. This is reflected in the heavy-hex lattice design of IBM-Q, which features reduced connectivity compared to previous layouts.

In contrast, trapped ion quantum computers feature all-to-all connectivity [31], but are currently limited to few qubits (≈ 20). QCs based on neutral atom technology do not feature all-to-all connectivity, but their connectivity is usually higher than nearest-neighbour, and can be further increased.

3.4 Variation of the Coupling Density

The connectivity between qubits is described by the coupling density c given by $c = N_C/N_{C,\max}$, where N_C denotes the number of connections between pairs of qubits (that is, the possible interactions), and $N_{C,\max}$ gives the maximal *possible* number of connections, which is obviously reached for a clique connectivity with

[5] Moving from locally connected components to on-chip integration might be beneficial for latency-critical embedded systems with quantum acceleration, but is likely not overly relevant for many HPC use-cases.

$N_{C,\max} = n(n-1)/2$ for n available qubits. Thus, $c = 1$ describes a quantum device with all-to-all connectivity. The base topology of the IBM-Q devices has a coupling density of $c \approx 0.0139$. For the simulations in this subsection, the size of the backend is kept constant at 127 qubits. The coupling density is increased by randomly adding connections between the qubits.[6] Each data point shown in this section is an average over 20 compilation runs, using again the standard transpilation process of Qiskit with optimisation level 3.

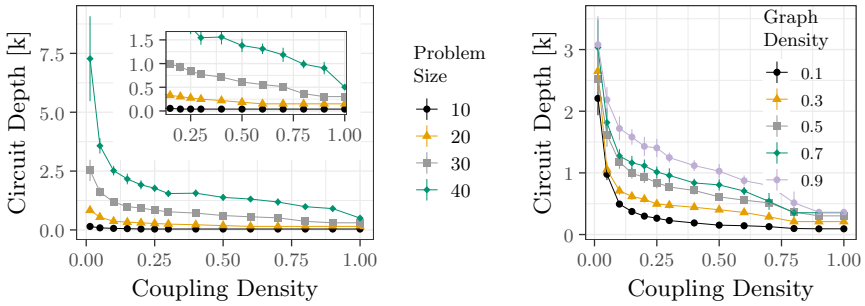


Fig. 2: Mean circuit depth vs. coupling density for $p = 1$ (127 qubit backend), varying problem size (lhs) for $d = 0.5$ and graph density (rhs) for $N = 60$.

We investigate the effect of increasing coupling density for different problem sizes, graph densities, and QAOA layers. In general, a higher coupling density reduces the number of SWAP gates needed to realise the desired two-qubit interactions, and decreases circuit depth. This is evident in the left panel of Fig. 2, where the resulting circuit depth is plotted vs. coupling density for different problem sizes and $d = 0.5$. The circuit depth saturates for higher coupling densities. The saturation density c_{sat} as well as the saturation value of the circuit depth increase with the problem size, which can also be seen in the inset of the plot. Solving larger problem instances requires more qubits and thus also more two-qubit gates, which leads to deeper circuits in general.

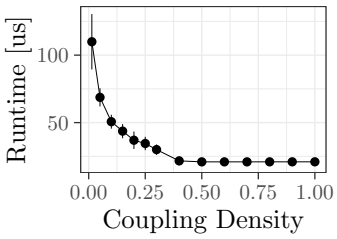


Fig. 3: Circuit runtime for $N = 20, d = 0.5, p = 1$.

Modifying the graph density for a fixed problem size does not change c_{sat}, as illustrated by the data in the right panel of Fig. 2 for $N = 60$, where the circuit depth remains constant for $c > 0.8$. The overall circuit depth and its saturation value increase with graph density. This can be traced back to an increased density of the QUBO resulting from an increasing amount of edges that necessitate more two-qubit gates. For larger number

[6] The placement of new connection favours augmenting regions with existing high connectivity density, following the assumption that adding extra connections is easier for regions that are already well connected. Given the lack of space, we refer readers to the replication package for the exact details.

of QAOA layers, the circuit depth increases linearly, whereas the saturation density c_{sat} remains unchanged. The decrease in circuit depth for higher coupling densities results in shorter runtimes as illustrated in Fig. 3. Since a relatively small problem instance with 20 qubits is considered, the runtime saturates for moderate coupling densities, similar as the circuit depth (see Fig. 2).

Both graphs in Fig. 2 show an important trend: Even a moderate increase in coupling density causes a substantial decrease in circuit depth - growing the coupling density from the standard topology to a 10% extended density reduces circuit depth for $N = 100$ from 7,000 to slightly over 2,000 (the effect is similar, yet becomes less pronounced for small input instances). We find this decrease to be a crucial improvement with regards to circuit execution times, but it also benefits NISQ systems, since shorter circuits pick up less effects of noise. Given limited space, we can unfortunately not study the impact of noise further in this paper, but retain this aspect for future work.

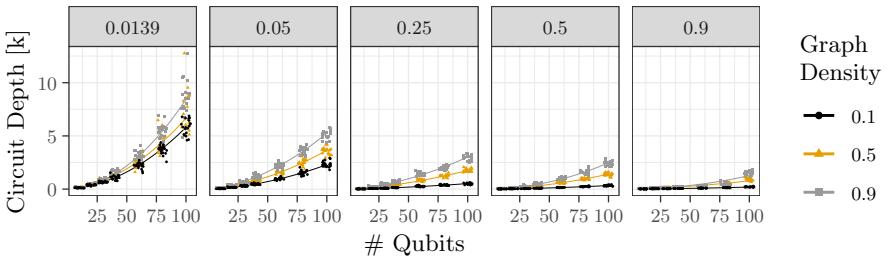


Fig. 4: Circuit depth growth behaviour over problem size for different coupling densities in the panels. Solid lines represent regression models for varying input graph dependencies.

To quantify the growth of the circuit depth with problem size, we construct a univariate regression model[7] of the form $f(x) = c_0 + c_1 x + c_2 x^2$. We have performed the usual regression diagnostics, and an ANOVA based model selection procedure unambiguously confirms that the choice of quadratic growth behaviour is preferable to linear and exponential alternative models. Since Fig. 4 visually demonstrates an excellent match between data and model, we do not explicitly spell out details of these diagnostics.

It is interesting to observe the behaviour of the regression coefficients with increasing connectivity map density in Fig. 5: The quadratic contribution is most pronounced for the unmodified topology, but quickly wanes with increasing connectivity map densities, and saturates in the connectivity density map range $[0.25, 0.75]$. Regardless of input problem structure (graph density), any quadratic contribution to growth vanishes for fully connected topologies. In general, more effort in implementing physical connections pays off with more favourable QPU scaling behaviour.

[7] Technically, we employ a robust quantile regression approach [32] because the stochastic circuit generation process produces pronounced outliers.

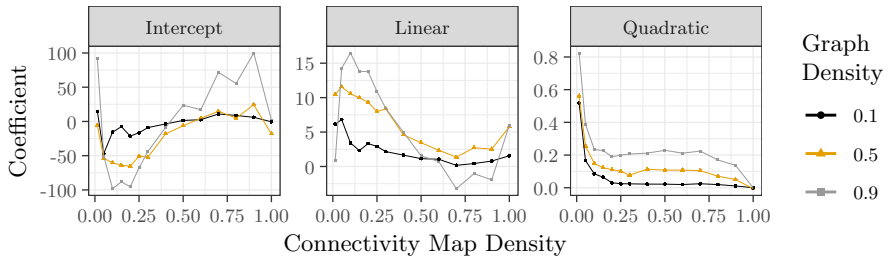


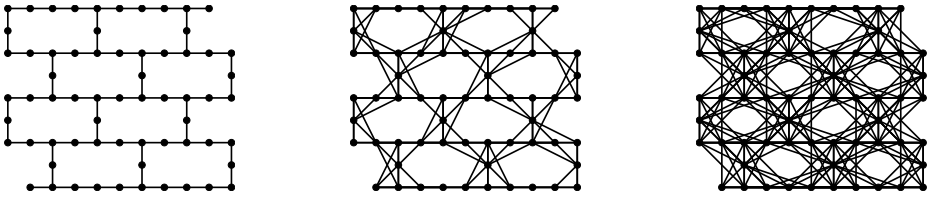
Fig. 5: Coefficients of the quadratic quantile regression model depending on graph and coupling density.

3.5 Variation of the Backend Size

The properties of the compiled circuit also depend on the size of the quantum device, namely the number of available qubits relative to the problem size. If the backend has more qubits available than needed, there are more possibilities for routing the virtual to physical qubits. In general, a more efficient placement of the circuit on the hardware can be found, for instance, by assigning more of the virtual qubit pairs that share two-qubit gates to physical qubits which are directly connected.

Table 3: Backend sizes used for the data in Fig. 7, characterised by the number of unit cell rows N_{rows}, unit cell columns N_{cols} and number of qubits.

N_{rows}	N_{cols}	N_{qubits}	N_{rows}	N_{cols}	N_{qubits}
4	2	65	5	3	108
3	3	70	4	4	113
5	2	79	6	3	127
4	3	89	5	4	137
6	2	93	6	4	161



(a) $c \approx 0.0139$ (b) $c = 0.1$ (c) $c = 0.2$

Fig. 6: Sketch of the heavy-hex lattice coupling map for 65 qubits with base (a) and extended (b, c) coupling density.

To examine the influence of the backend size on the circuit depth, the circuit for a Max-Cut problem with $N = 60$, $d = 0.5$ and $p = 1$ is compiled on backends

of different sizes between 65 and 161 qubits. The qubits are arranged in the heavy-hex lattice geometry and the size is increased by successively adding unit cells below or on the right. The smallest backend corresponds to the IBM-Q Brooklyn device, consisting of four rows and two columns of unit cells, as depicted in Fig. 6 for the base coupling density as well as $c = 0.1$ and $c = 0.2$. Starting from this layout, ten different sizes are considered, as summarised in Tab. 3.

In Fig. 7, the resulting mean circuit depths are shown as a function of the number of qubits for various coupling densities. In general, the circuit depth decreases with the backend size, illustrating the effects of a more efficient placement of the circuit described above. These effects become less pronounced for higher coupling densities, since the backend then exhibits more qubit pairs that are directly connected. Consequently, for a backend with all-to-all connectivity ($c = 1$), the backend size has no influence on the depth of the compiled circuit.

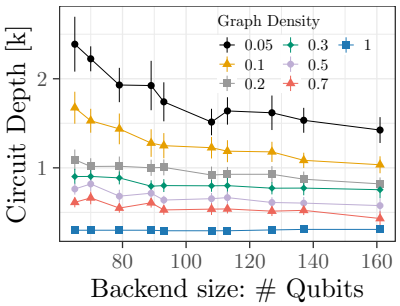


Fig. 7: Mean depth vs. backend size for $N = 60$, $d = 0.5$ and $p = 1$.

4 Conclusion & Outlook

In this paper, we have investigated integration and co-design possibilities for QPUs that are supposed to act as computational accelerators in high performance computing systems. Our results show that designing purpose-specific QPUs with adapted topologies holds promises in terms of computational capabilities. We have also shown that circuit depth is reduced by increasing the connectivity map density, but already saturates at values $c \ll 1$, with a slight dependency on the problem size. Thus, all-to-all connectivity is not needed in all cases, relaxing the requirements on the quantum hardware devices. We have also discussed that how integration of QPUs is performed can have effects on their capabilities, even if these are not as pronounced as for topology adaptations.

For now, we have focused on perfect QPUs that do not suffer from noise and imperfections. Future work will incorporate these deficiencies into our analysis, which is important to transfer our results to present-day NISQ systems, and will help progressing towards practical utility of early-stage quantum computers.

Acknowledgement: We thank Manuel Schönberger for providing his topology adaptation simulation code as starting point for our efforts.

References

1. J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug 2018. [Online]. Available: https://doi.org/10.48550/arXiv.1801.00862
2. Quantum Technology and Application Consortium—QUTAC, "Industry quantum computing applications," *EPJ Quantum Technology*, vol. 8, no. 1, p. 25, 2021.

3. T. Krüger and W. Mauerer, "Quantum annealing-based software components: An experimental case study with SAT solving," p. 445–450, 2020. [Online]. Available: https://doi.org/10.1145/3387940.3391472
4. W. Mauerer and S. Scherzinger, "1-2-3 reproducibility for quantum software experiments," *Q-SANER@IEEE International Conference on Software Analysis, Evolution and Reengineering*, 2022.
5. D. E. Deutsch, "Quantum computational networks," *Proceedings of the Royal Society A*, vol. 425, p. 73–90, 1989. [Online]. Available: https://doi.org/10.1098/rspa.1989.0099
6. A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Phys. Rev. A*, vol. 52, p. 3457–3467, 1995. [Online]. Available: https://doi.org/10.1103/PhysRevA.52.3457
7. R. Raussendorf, D. E. Browne, and H. J. Briegel, "Measurement-based quantum computation on cluster states," *Nature Physics*, vol. 68, p. 022312, 2003. [Online]. Available: https://doi.org/10.1103/PhysRevA.68.022312
8. E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, "A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem," *Science*, vol. 292, pp. 472–475, 2001. [Online]. Available: https://doi.org/10.1126%2Fscience.1057726
9. E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv:1411.4028*, Nov 2014. [Online]. Available: https://doi.org/10.48550/arXiv.1411.4028
10. H.-L. Huang, D. Wu, D. Fan, and X. Zhu, "Superconducting quantum computing: A review," *arXiv:2006.10433*, 2020. [Online]. Available: https://doi.org/10.48550/arxiv.2006.10433
11. M. Kjaergaard, M. E. Schwartz, J. Braumüller, P. Krantz, J. I.-J. Wang, S. Gustavsson, and W. D. Oliver, "Superconducting qubits: Current state of play," *Annual Review of Condensed Matter Physics*, vol. 11, pp. 369–395, mar 2020. [Online]. Available: https://doi.org/10.1146%2Fannurev-conmatphys-031119-050605
12. A. Lucas, "Ising formulations of many NP problems," vol. 2, 2014. [Online]. Available: https://doi.org/10.48550/arXiv.1302.5843
13. K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, "Noisy intermediate-scale quantum algorithms," *Rev. Mod. Phys.*, vol. 94, p. 015004, Feb 2022. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.94.015004
14. E. Farhi and A. W. Harrow, "Quantum supremacy through the quantum approximate optimization algorithm," *arXiv:1602.07674*, 2016.
15. C. W. Commander, *Maximum cut problem, MAX-CUT Maximum Cut Problem, MAX-CUT*. Boston, MA: Springer US, 2009, pp. 1991–1999.
16. M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *J. ACM*, vol. 42, no. 6, 1995. [Online]. Available: doi.org/10.1145/227683.227684
17. F. G. Fuchs, H. Øie Kolden, N. H. Aase, and G. Sartor, "Efficient encoding of the weighted max k-cut on a quantum computer using QAOA," *SN Computer Science*, vol. 2, no. 2, 2021. [Online]. Available: https://doi.org/10.1007/s42979-020-00437-z
18. J. Wurtz and D. Lykov, "Fixed-angle conjectures for the quantum approximate optimization algorithm on regular maxcut graphs," *Phys. Rev. A*, vol. 104, p. 052419, Nov 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.104.052419

19. K. Marwaha, "Local classical MAX-CUT algorithm outperforms $p = 2$ QAOA on high-girth regular graphs," *Quantum*, vol. 5, p. 437, Apr 2021. [Online]. Available: https://doi.org/10.22331/q-2021-04-20-437
20. J. Wurtz and P. Love, "Maxcut quantum approximate optimization algorithm performance guarantees for $p > 1$," *Phys. Rev. A*, vol. 103, p. 042612, Apr 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.103.042612
21. S. Harrison, H.Sigurdsson, S. Alyatkin, J. Töpfer, and P. Lagoudakis, "Solving the max-3-cut problem with coherent networks," *Phys. Rev. Applied*, vol. 17, p. 024063, Feb 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevApplied.17.024063
22. M. Willsch, D. Willsch, F. Jin, H. D. Raedt, and K. Michielsen, "Benchmarking the quantum approximate optimization algorithm," *Quantum Information Processing*, vol. 19, no. 7, p. 197, Jun 2020. [Online]. Available: https://doi.org/10.1007/s11128-020-02692-8
23. C. Xue, Z.-Y. Chen, Y.-C. Wu, and G.-P. Guo, "Effects of quantum noise on quantum approximate optimization algorithm," *Chinese Physics Letters*, vol. 38, no. 3, mar 2021. [Online]. Available: doi.org/10.1088/0256-307x/38/3/030302
24. Y. Pan, Y. Tong, and Y. Yang, "Automatic depth optimization for a quantum approximate optimization algorithm," *Phys. Rev. A*, vol. 105, p. 032433, Mar 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.105.032433
25. V. Akshay, D. Rabinovich, E. Campos, and J. Biamonte, "Parameter concentrations in quantum approximate optimization," *Phys. Rev. A*, vol. 104, p. L010401, Jul 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.104.L010401
26. L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," *Phys. Rev. X*, vol. 10, p. 021067, Jun 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.10.021067
27. S. Hadfield, Z. Wang, B. O'Gorman, E. Rieffel, D. Venturelli, and R. Biswas, "From the quantum approximate optimization algorithm to a quantum alternating operator ansatz," *Algorithms*, vol. 12, no. 2, p. 34, 2019. [Online]. Available: https://doi.org/10.3390%2Fa12020034
28. R. LaRose, E. Rieffel, and D. Venturelli, "Mixer-phaser ansätze for quantum optimization with hard constraints," *Quantum Machine Intelligence*, vol. 4, no. 2, p. 17, Jun 2022. [Online]. Available: https://doi.org/10.1007/s42484-022-00069-x
29. M. Schönberger, M. Franz, S. Scherzinger, and W. Mauerer, "Peel | Pile? Cross-framework portability of quantum software," in *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*, 2022, pp. 164–169.
30. M. Franz, L. Wolf, M. Periyasamy, C. Ufrecht, D. D. Scherer, A. Plinge, C. Mutschler, and W. Mauerer, "Uncovering instabilities in variational-quantum deep q-networks," *Journal of The Franklin Institute*, 2022. [Online]. Available: https://doi.org/10.1016/j.jfranklin.2022.08.021
31. C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-ion quantum computing: Progress and challenges," *Applied Physics Reviews*, vol. 6, p. 021314, 2019. [Online]. Available: https://doi.org/10.1063%2F1.5088164
32. R. Koenker, *quantreg: Quantile Regression*, 2022, r package version 5.88. [Online]. Available: https://doi.org/10.1201/9781315120256