Vol.2021-QS-2 No.14
2021/3/29

Quantum Gate Pattern Recognition and Circuit Optimization for Scientific Applications

Wonho Jang[1] Koji Terashi[2,a)] Masahiko Saito[2] Christian W. Bauer[3] Benjamin Nachman[3]
Yutaro Iiyama[2] Tomoe Kishimoto[2] Ryunosuke Okubo[1] Ryu Sawada[2] Junichi Tanaka[2]

Abstract: There is no unique way to encode a quantum algorithm into a quantum circuit. With limited qubit counts, connectivities, and coherence times, circuit optimization is essential to make the best use of near-term quantum devices. We introduce two separate ideas for circuit optimization and combine them in a multi-tiered quantum circuit optimization protocol called AQCEL. The first ingredient is a technique to recognize repeated patterns of quantum gates, opening up the possibility of future hardware co-optimization. The second ingredient is an approach to reduce circuit complexity by identifying zero- or low-amplitude computational basis states and redundant gates. As a demonstration, AQCEL is deployed on an iterative and efficient quantum algorithm designed to model final state radiation in high energy physics. For this algorithm, our optimization scheme brings a significant reduction in the gate count without losing any accuracy compared to the original circuit. Additionally, we have investigated whether this can be demonstrated on a quantum computer using polynomial resources. Our technique is generic and can be useful for a wide variety of quantum algorithms.

Keywords: Quantum Computing, Quantum Circuit, Optimization, High Energy Physics

1. Introduction

Recent technology advances have resulted in a variety of universal quantum computers that are being used to implement quantum algorithms. However, these noisy-intermediate-scale quantum (NISQ) devices [1] may not have sufficient qubit counts or qubit connectivity and may not have the capability to stay coherent for entirety of the operations in a particular algorithm implementation. Despite these challenges, a variety of applications have emerged across science and industry. For example, there are many promising studies in experimental and theoretical high energy physics (HEP) for exploiting quantum computers. These studies include event classification, e.g., [2], [3], [4], reconstructions of charged particle trajectories, e.g., [5], [6] and physics objects [7], [8], unfolding measured distributions [9] as well as simulation of multi-particle emission processes [10], [11]. A common feature of all of these algorithms is that only simplified versions can be run on existing hardware due to the limitations mentioned above.

One of the general strategies for improving the performance of NISQ computers is circuit optimization, also known as circuit compilation. In particular, there is no unique way to encode a quantum algorithm into a set of gates, and certain realizations of an algorithm may be better suited for a given quantum device. One widely used tool is t|ket⟩ [12], which contains a variety of architecture-agnostic and architecture-specific routines. There are also a variety of other toolkits for circuit optimization, including hardware-specific packages for quantum circuits (see e.g., [13], [14], [15], [16]). Since t|ket⟩ is a generic framework which contains many algorithms that have already been benchmarked against other procedures, it will serve as our baseline.

We introduce two techniques that can be used to optimize circuits and that are complementary to existing methods. The first focuses on the identification of recurring sets of quantum gates in a circuit. Identifying such recurring sets of gates (RSG) can be very important, since any optimization of these RSGs has an enhanced effect on the overall circuit. Furthermore, identifying recurring gate sets can be useful for future hardware optimizations where the fidelity of certain common operations can be enhanced at the expense of other, less frequent operations. The second technique optimizes a generic circuit by eliminating unnecessary gates or unused qubits such that the circuit depth becomes as short as possible. One example where such an optimization can lead to simplifications is a case where a quantum circuit has been designed with complete generality in mind. In this case, for a certain initial state the circuit only reaches a select set of intermediate states such that some operations become trivial and can be eliminated. The elimination of unnecessary gate operations introduced here focuses on controlled operations such as a Toffoli or a CNOT gate. The heart of the elimination technique resides in the identification of zero- or low-amplitude computational basis states, that allows us to determine whether the entire gate or (part

[1] Department of Physics, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113–0033, Japan
[2] International Center for Elementary Particle Physics (ICEPP), The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113–0033, Japan
[3] Physics Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
[a)] koji.terashi@cern.ch

of) qubit controls can be removed. Both of these techniques are combined in an optimization protocol called AQCEL[*1].

To demonstrate the effectiveness of these techniques, we will use a quantum algorithm from HEP to perform a calculation in quantum field theory. The particular algorithm that we study models a *parton shower*, which is the collinear final state radiation from energetic charged (under any force) particles [11]. This algorithm is a useful benchmark because it provides an exponential speedup over the most efficient known classical algorithm and the circuit depth can be tuned for precision. While we show results for this specific circuit, the proposed protocol has a wide range of applicability for quantum computing applications across science and industry.

2. AQCEL optimization protocol

As already mentioned, the AQCEL protocol comprises two components: identifying recurring quantum gates (Sec. 2.1) and eliminating unnecessary gates and unused qubits (Sec. 2.2). This approach focuses on circuit optimization at the algorithmic level, instead of at the level of a specific implementation using the native gates for a particular quantum device. The individual optimization steps are described below.

2.1 Gate set pattern recognition

First, the AQCEL attempts to identify gate set patterns in an arbitrary quantum circuit and extract RSGs from the circuit.

2.1.1 Representation in directed acyclic graph

In a quantum circuit, individual qubits are manipulated by gate operations one by one, meaning that the quantum state represented at a certain point of the circuit should not be affected by gate operations applied afterward. Such a structure can be described by using a directed acyclic graph (DAG). The DAG allows us to easily check dependencies between qubits and to extract a subset of the circuit that functions for certain tasks.

First, we convert a quantum circuit to the form of DAG using the DAGCircuit class in Qiskit Terra API, where a node represents an operation by a quantum gate and an edge that connects the nodes represents a qubit. The gate set pattern recognition can be then performed by identifying two parts of quantum circuit functioning in an identical manner using DAG as a graph isomorphism problem. The algorithm of gate set pattern recognition consists of two steps: (1) finding RSG candidates with DAG representation using depth-first search with heuristic pruning, and (2) checking the DAG isomorphism by graph hashing with Weisfeiler Lehman graph hash [17], as implemented in the NetworkX library [18]. The details of the gate set pattern recognition including computational complexity are given in Appendix A.1, with the pseudocode of the algorithm.

2.1.2 Tiered extraction of recurring gate sets

The appearance pattern of RSGs in a circuit may depend on specific encoding of the quantum algorithm. To account for different patterns, we consider three different levels of matching criteria to define the gate recurrence:

Level 1 : Only matching in gate types,

[*1] AQCEL (pronounced "excel") stands for *Advancing Quantum Circuit by* ICEPP *and* LBNL.

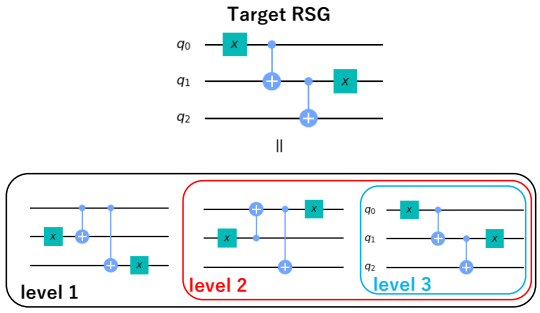


Fig. 1 Possible RSG patterns for a given target RSG corresponding to the three levels of matching criteria.

Level 2 : Matching in gate types and the *roles* of qubits that the gates act on,

Level 3 : Matching in gate types and both *roles* and *indices* of qubits that the gates act on.

The matching criterion in Level 1 is the least stringent: it just identifies the same sets of quantum gates appearing in the circuit, irrespective of which qubits they act on. The Level 2 is more strict and ensures that the qubits the RSGs act on have the same roles. In other words, the qubit connections between the gates inside a single RSG are maintained but the qubit indices might vary between the RSGs. The Level 3 applies the most stringent condition, where the qubits that the RSGs act on must have the same roles and qubit indices, that is, the RSGs must appear on an identical set of qubits in the circuit. The appearance patterns of the RSGs are illustrated in Fig. 1 for the three matching criteria.

The identified RSGs are ranked in terms of the product of the number of gates constituting the set and the number of occurrence of the set in the circuit. A fixed number of top-ranked RSGs are extracted from the circuit in this step.

2.2 Heuristic circuit optimization

After attempting to identify RSGs in the circuit, a heuristic optimization procedure takes place to make the circuit depth as short as possible by eliminating redundant gates or unused qubits. In this step, we consider two levels of optimization:

Level 1 : Optimize the entire circuit including RSGs,

Level 2 : Optimize the entire circuit, but for the RSGs only adjacent gate pairs are removed (see Sec. 2.2.4).

The Level 1 would provide a shorter, more efficient circuit. Compared to Level 1, the Level 2 likely results in a deeper circuit for most cases, while it provides more room for improvement in later compilation stages if the RSGs have specialized low-level implementations.

2.2.1 Basic idea of redundant controlled operations removal

A controlled operation such as a CNOT or a Toffoli gate performs different operations depending on the quantum state of the system at the point where the gate is applied. Let m be the number of control qubits of this operation. Consider expanding the state of the full system $|\psi\rangle$ into a superposition of computational basis states as

$$|\psi\rangle = \sum_{j,k} c_{j,k} \, |j\rangle_{\mathrm{ctl}} \otimes |k\rangle \,, \tag{1}$$

where $|\cdot\rangle_{\mathrm{ctl}}$ denotes the state of the control qubits, while the unsubscripted ket corresponds to the rest of the system. We write

the states as integers wtih $0 \le j \le 2^m - 1$ and $0 \le k \le 2^{n-m} - 1$. We assume that the controlled operation for the gate is applied when all control qubits are in the $|1\rangle$ state, corresponding to $|j\rangle_{\text{ctl}} = |11\ldots1\rangle = |2^m - 1\rangle_{\text{ctl}}$. This allows to classify the state of the system into three general classes with the amplitudes $c_{j,k}$:

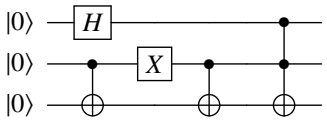
Triggering : $c_{j,k} \neq 0$ iff $j = 2^m - 1$. The controlled operation of the gate in question is applied for all computational bases in the superposition.

Non-triggering : $c_{2^m-1,k} = 0$ for all k. The controlled operation is never applied.

Undetermined : The state is neither triggering nor non-triggering.

A circuit containing triggering or non-triggering controlled gates can be simplified by removing all controls (triggering case) or by eliminating the gates entirely (non-triggering case). While an undetermined single-qubit controlled gate cannot be simplified under the current scheme, an undetermined multi-qubit controlled gate can be by removing the controls on some of the qubits, if the state of the system satisfies the condition described in Appendix A.2.

As an example of this concept, consider the following simple circuit:



If the second qubit is in the initial state $|0\rangle$, the first CNOT gate has no effect and can be removed from the circuit as the $|0\rangle$ is the non-triggering state of CNOT. The second qubit before the second CNOT gate is in the state $|1\rangle$, which is the triggering state. Therefore, the qubit control can be removed from the second CNOT gate. The first two qubits before the Toffoli gate are in the superposition of $|01\rangle$ and $|11\rangle$, which is an undetermined state for the Toffoli gate. Since the Toffoli gate has a triggering bitstring $\{11\}$, and the second qubit is always in the $|1\rangle$ state, this second qubit control can be removed from the Toffoli gate, replacing it with a CNOT gate controlled only on the first qubit.

The heuristic circuit optimization therefore requires, for each controlled gate, the identification of possible states the control qubits can take, and the removal of unnecessary parts of the controlled operations. These two steps are discussed in detail below.

It is well known that an arbitrary multi-qubit controlled-U gate with m control qubits can be decomposed into $O(m)$ Toffoli and controlled-U gates [19]. Therefore, in the remainder of this paper, we assume that all controlled gates are reduced to Toffoli gates denoted as $C^2[X]$, and singly-controlled unitary operation denoted as $C[U]$. This implies that the only triggering bitstrings we need to consider are either $\{1\}$ or $\{11\}$. For a n-qubit circuit composed of N multi-qubit controlled-U gates, each having at most n control qubits, this decomposition results in at most $\tilde{N} = nN$ controlled gates.

2.2.2 Identification of computational basis states

In general, a circuit consisting of n qubits creates a quantum state described by a superposition of all of the 2^n computational basis states. However, it is rather common that a specific circuit produces a quantum state where only a subset of the computational basis states has nonzero amplitudes. Moreover, the number of finite-amplitude basis states depends on the initial state. This is why the three classes of the states of the system arise.

The state classification at each controlled gate can be determined either through a classical simulation or by measuring the control qubits repeatedly. In the case of a classical simulation, one can either perform the full calculation of the amplitudes, or simply track all the computational basis states whose amplitudes may be nonzero at each point of the circuit without the calculation of the amplitudes. Aqcel adopts the latter method for the lower computational resource. When instead the quantum measurements are used, the circuit is truncated right before the controlled gate in question, and the control qubits are measured repeatedly at the truncation point. Finiteness of the relevant amplitudes can be inferred from the distribution of the obtained bitstrings, albeit within the statistical uncertainty of the measurements.

A few notes should be taken on the computational costs of the two methods. Consider an n-qubit circuit with N controlled gates. As discussed before, reducing this to either $C^2[X]$ or $C[U]$ results in $O(\tilde{N})$ single or double controlled gates. A classical simulation of the state vector before a given controlled gate has an exponential scaling in the number of qubits and requires $O(2^n)$ computations. On the other hand, measuring the $m = 1$ or 2 control qubits M times, which results in M bitstrings of length m, only requires $O(M)$ operations. Repeating this for all $\tilde{N}$ gates requires $O(\tilde{N}2^n)$ for the classical simulation and $O(\tilde{N}^2 M)$ when using quantum measurements.

More details on the estimates of the computational resource necessary for the identification of computational basis states, as well as other optimization steps, are described in Appendix A.3.

2.2.3 Elimination of redundant controlled operations

Once the nonzero-amplitude computational basis states are identified at each controlled gate, we remove the gate or its controls if possible. When using classical simulation, the entire circuit is analyzed first before the control elimination step. When quantum measurments are instead used, circuit execution, measurements and circuit optimization are performed at each controlled gate separately.

The control elimination step for each controlled gate proceeds as follows. For a $C[U]$ gate, compute the probability of observing $|1\rangle$ of the control qubit. If that probability is 1, eliminate the control and only keep the single unitary gate U. If the probability is 0, remove the controlled gate from the circuit. In all other cases, keep the controlled gate. For a $C^2[X]$ (Toffoli) gate, compute the probabilities of the four possible states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. If the probability of $|11\rangle$ is 1, remove the two controls and only keep the X gate. If the probability of $|11\rangle$ is 0, remove the entire Toffoli gate. If neither of those two conditions are true (the undetermined class), it is still possible to eliminate one of the two controls. This is true if the probability of the state $|01\rangle$ ($|10\rangle$) is zero, in which case one can eliminate the first (second) control.

Note that for noisy quantum circuits the measurements of the states will not be exact, and one expects contribution from errors in the probabilities of observing certain bitstrings. This means that one has to impose thresholds when deciding whether we call

the state triggering, non-triggering or undetermined. Once such a threshold has been decided, the number of measurements required has to be large enough for the statistical uncertainty to be smaller than this threshold. This will be discussed in more detail in Sec. 3 when we give explicit examples.

The computational cost of determining whether to eliminate controls or the entire controlled operation can easily be determined. Given the measured bitstrings, which can be determined with $O(\tilde{N}^2 M)$ operations, one can compute the probabilities for each possible bitstring, and therefore decide whether to simplify a controlled operation using $O(\tilde{N})$ operations.

Note that superfluous controlled operations can also be found and eliminated using the so-called ZX calculus [20], [21]. In fact, the ZX calculus is complete in the formal logic sense of the word, such that one can always prove that an unnecessary gate can be removed using the ZX calculus. However, in general this requires exponential resources, and therefore has no scaling advantage with respect to simply computing the state vectors. Of course, the ZX calculus is still incredibly powerful and underlies many of the optimization techniques of quantum transpilers, such as the t|ket⟩ compiler we compare to later.

2.2.4 Elimination of adjacent gate pairs

Note that if a unitary operator A and its Hermitian conjugate $A^\dagger$ act on the same set of qubits adjacently, resulting in an identity operation, the gates implementing these operators can be removed from the circuit. While this is an obvious simplification, the removal of gates through the optimization steps described above can result in a circuit with such cancelling gate pairs. For this reason, this step of gate reduction is applied before and after eliminating redundant controlled operations.

2.2.5 Elimination of unused qubits

After taking the above steps, the circuit is examined for the presence of qubits where no gate is applied, which can then be removed from the circuit. Such a situation occurs e.g., when a quantum circuit designed to work universally with different initial states is executed with a specific initial state. An example of such a circuit is the sequential algorithm we consider below.

3. Application to quantum algorithm

The circuit optimization protocol described in Sec. 2 has been deployed to a quantum algorithm designed for HEP [11]. The heuristic optimization (Sec. 2.2) is performed at Level 1 for the optimization on existing quantum hardware.

3.1 Quantum parton shower algorithm

Simulating quantum field theories is a flagship scientific application of quantum computing. It has been shown that a generic scattering process can be efficiently simulated on a quantum computer with polynomial resources [22]. However, such circuits require prohibitive resources in the context of near-term devices.

A complementary approach is to simulate one component of the scattering process. In particular, Ref. [11] proposed an algorithm to simulate the collinear radiation from particles that carry a nonzero fundamental charge. Such radiation approximately factorizes from the rest of the scattering amplitude and can therefore be treated independently. This factorization is the basis for parton shower Monte Carlo generators in HEP. The quantum parton shower (QPS) algorithm provides an exponential speedup over known algorithms when the charge is not the same for all particles that can radiate.

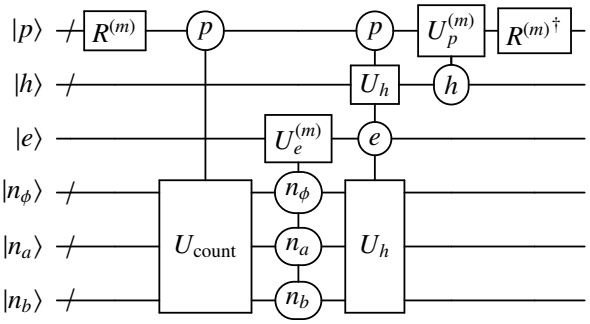


Fig. 2 The m^{th} step of the quantum circuit for the algorithm proposed in Ref. [11]. There are three physical registers: $|p\rangle$ containing the set of particles at this step; $|h\rangle$ for the branching history; and $|e\rangle$ which is a binary variable representing the presence or absence of an emission at this step. The three lower registers count the number of particles of type ϕ, a, and b and are uncomputed before the end of the circuit. The exact form of the rotation matrices $R^{(m)}$ and the unitary operations U_{count}, $U_e^{(m)}$, U_h, and $U_p^{(m)}$ can be found in Ref. [11].

The particular example demonstrated in Ref. [11] starts with n fermions that can be either type f_1 or f_2. These fermions can radiate a scalar particle ϕ, which itself can split into fermion-anti-fermion pairs (of the same or different type). The relevant parameters are the three couplings g_1, g_2, and g_{12} between f_1 and ϕ, f_2 and ϕ, and $f_1\bar{f}_2$ ($\bar{f}_1 f_2$) and ϕ, respectively. The shower evolution is discretized into N_{evol} steps and at each step, one of the particles could radiate / split or nothing happens. This produces a precise result when N_{evol} is large. Figure 2 shows the quantum circuit block for the m^{th} step of the quantum circuit. First, the fermions are rotated into a new basis f_a and f_b where the effective mixing g_{ab} between $f_a\bar{f}_b$ ($\bar{f}_a f_b$) and ϕ is zero. Then, the number of particles of each type are counted and stored in registers n_a, n_b, and n_ϕ. Next, a Sudakov factor is calculated to determine if an emission happens or not. This operation depends only on the total number of particles of each type. After the emission step, the particle and history registers are modified accordingly. Lastly, the fermions are rotated back into the f_1 and f_2 basis. Some of the steps in this algorithm are universal (independent of m) and some dependent on m due to the running of coupling constants with energy scale.

3.2 Experimental setup

The QPS simulation is implemented into a quantum circuit using IBM Qiskit version 0.22.0 [23] with Terra 0.15.2, Aer 0.6.1 and Ignis 0.4.0 APIs in Python 3.8 [24]. First, we attempt to optimize the circuits running on a classical computer with a single 2.4 GHz Intel core i5 processor.

In order to evaluate the Aqcel performance, the same QPS circuit optimized using t|ket⟩ [12] in pytket 0.6.1 before transpilation is used as a reference. The optimization using t|ket⟩ is done as follows. We consider the list of ten pre-defined passes[*2]. The passes

[*2] The following 10 pre-defined passes are considered for the t|ket⟩ optimization: *EulerAngleReduction(OpType.Rz,OpType.Rx)*, *RemoveRedundancies*, *GuidedPauliSimp*, *SquashHQS*, *FlattenRegisters*, *OptimisePhaseGadgets*, *KAKDecomposition*, *USquashIBM*, *CliffordSimp*, *FullPeepholeOptimise*. Two more passes, *RebaseIBM*, *Com-*

are tried one by one on the QPS circuit, and the one that reduces the number of gates the most is applied to the circuit. The same set of passes are tried again on the resulting circuit to identify and apply the pass that most effectively reduces the gate count. This iterative process is repeated until the gate count is no longer reduced by any of the passes. The selected sequence of passes are used for evaluating the t|ket⟩ performance in the remainder of the studies.

The QPS algorithm is executed on the 27-qubit "ibmq_sydney" [25], one of the IBM Quantum Falcon Processors, and the state-vector simulator in Qiskit Aer with and without optimizing the circuit. For the results obtained solely from the state-vector simulator, all the qubits are assumed to be connected to each other (referred to as the ideal topology). When executing the algorithm on the sydney, the gates in the circuit are transformed into machine-native single- and two-qubit gates, and the qubits are mapped to the hardware accounting for the actual qubit connectivity. For all the circuits tested with the sydney below, the noise-adaptive mapping is performed by taking into account the read-out and CNOT gate errors from the calibration data as well as the qubit connection constraints[*3]. Gate cancellations also take place at this stage using the commutativity of native gates and unitary synthesis, as documented in Qiskit Terra API. This qubit mapping and gate cancellation process are repeated eleven times, and the circuit obtained with the smallest number of gates is finally tested with the sydney.

3.3 Results

3.3.1 Circuit optimization for $N_{\rm evol} = 2$ branching steps using classical simulation

Circuit optimization performance of Aqcel is evaluated for the QPS simulation circuit with $N_{\rm evol} = 2$ branching steps assuming an ideal topology. The simulation does not consider any effects from hardware noise. The initial state is chosen to be $|f_1\rangle$, and the coupling constants are set to $g_1 = 2$ and $g_2 = g_{12} = 1$. Both $f \to f'\phi$ and $\phi \to f\bar{f}$ processes are considered[*4].

First, the RSG pattern recognition is performed against the circuit. When the Level 2 RSG pattern recognition is applied, two RSGs are identified with the requirements on the number of nodes in each RSG being between 5 and 7 and the number of repetitions being 4 or more. Next, the heuristic optimization (Sec. 2.2) is performed over the entire circuit at Level 1. This step consists of identifying nonzero-amplitude computational basis states, removing redundant controlled operations, removing adjacent cancelling gate pairs (performed twice), and removing unused qubits. Nonzero-amplitude computational basis states are identified through classical calculation.

After the algorithmic-level circuit optimization, the quantum gates in the circuit are decomposed into single-qubit gates (U_1, U_2, U_3) and CNOT gates. Figure 3 shows the numbers of the single-qubit and CNOT gates, the sum of the two, and the depth of the circuit before and after the optimization. The circuit depth is defined as the length of the longest path from the input to the measurement gates, with each gate counted as a unit. The figure compares the values from the original circuit and the circuits optimized with t|ket⟩ only, Aqcel only and the combination of the two. The Aqcel optimizer reduces the total number of gates by 54%, resulting in a 51% reduction of the circuit depth. In particular, the reduction of the number of CNOT gates is 47%. This compares to t|ket⟩, which reduces the total number of gates by 26%, CNOT by 1%, and the circuit depth by 10%. This means that, for the QPS algorithm, Aqcel is 38% more efficient than t|ket⟩ in reducing the gate counts, and 46% more specifically for CNOT, and makes the circuit 45% shorter. Combining the two optimizers, the gate count is reduced by 63% (50% for CNOT only) and the depth by 55% with respect to the original circuit. The combined optimizer is 51% more efficient than the t|ket⟩ alone for gate reduction (49% for CNOT only), producing a 50% shorter circuit.

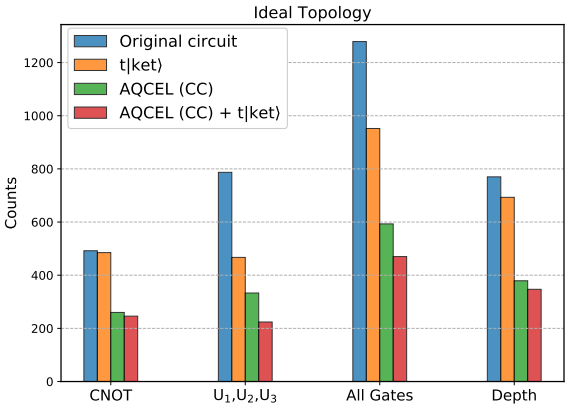


Fig. 3 Numbers of single-qubit ($U_{1,2,3}$) gates, CNOT gates and the sum of the two as well as the depth of the two-branching step QPS circuit decomposed into native gates before and after optimization. The computational basis states with nonzero amplitudes at controlled gates are identified using classical calculation in the heuristic optimization step of Aqcel.

For the Aqcel optimizer, the gate reduction occurs mostly at the stage where the redundant qubit controls are removed. Starting with 1279 gates (excluding barrier and measurement gates), the first adjacent gate-pair elimination, the redundant qubit control reduction, and the second gate-pair elimination steps remove 170, 510 (40% of the 1279 gates), and 6 gates, respectively. In terms of the computational cost, the wall time is by far dominated by the two adjacent gate-pair elimination steps combined, accounting for 91% of the total time, with a sub-dominant contribution of 7% from the redundant qubit control reduction.

Finally, the number of qubits is reduced from 24 to 21 with the Aqcel optimizer, while it is unchanged by t|ket⟩. One qubit is removed from each of the three registers n_a, n_b, and n_ϕ because those qubits are used only for $N_{\rm evol} \geq 3$ branching steps.

3.3.2 Circuit optimization for $N_{\rm evol} = 1$ branching step using classical simulation and quantum measurements

The quantum circuit for the two-branching step QPS simulation is still too deep to produce useful results on a real existing quantum computer, even after optimizing the circuit. Therefore, we consider the circuit with only one branching step using the

muteThroughMultis, are also used once before selecting the pass from the list.

[*3] This corresponds to the transpilation of level 3 pass manager, as implemented in Qiskit Terra.

[*4] Ref. [11] noted that when these are unphysically removed, the circuit can be simulated efficiently classically (see also Ref. [10]).

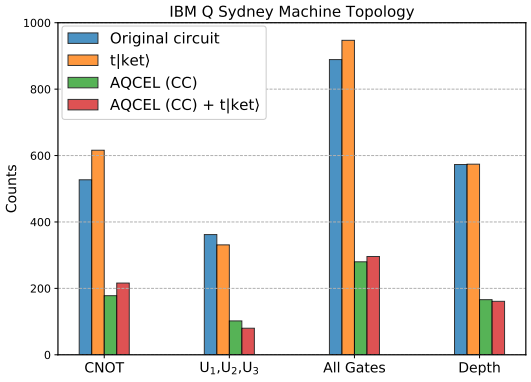


Fig. 4 Numbers of single-qubit ($U_{1,2,3}$) gates, CNOT gates and the sum of the two as well as the depth of the one-branching step QPS circuit transpiled considering the ibmq_sydney topology before and after the optimizations. The computational basis states with nonzero amplitudes at controlled gates are identified using classical calculation in the heuristic optimization step of Aqcel.

sydney and the state-vector simulator. The initial state, coupling constants, and considered processes are the same as those used for the $N_{\rm evol} = 2$ branching step simulation.

First, we examine the gate and qubit counts for the one-branching step QPS simulation assuming an ideal topology. Starting with 486 gates, the Aqcel optimizer removes 24, 260 (53% of 486 gates), and 2 gates in the three steps of the heuristic optimization, in the order given above. The adjacent gate-pair elimination step still dominates the wall time (96%). However, the redundant qubit control reduction now takes about 9 times less time than that for the two-branching step simulation, consistent with the exponential behavior of the computing cost of the step. The number of qubits is reduced from 15 to 13 with Aqcel. One of four ancilla qubits is removed because three ancillas are sufficient for decomposing all the multi-controlled gates in the $N_{\rm evol} = 1$ step. The register n_ϕ, composed of only one qubit, is also removed because it is used only for the case where the initial state is $|\phi\rangle$.

Next, the optimized circuits are transpiled considering the qubit connectivity of the ibmq_sydney. Figure 4 shows the same set of distributions as in Fig. 3, but for the one-branching step QPS simulation with the sydney-specific transpilation. The Aqcel optimizer reduces the number of native gates significantly in this case, too. The relative reduction is more drastic for the one branching step than the two branching steps, mainly because the former (shallow) circuit has relatively more zero-amplitude computational basis states than the latter (deep) circuit.

We then evaluate the performance of the optimizers using the sydney. A particular challenge when employing Aqcel with a real quantum computer is in the determination of the bitstring probabilities of the control qubits at each controlled gate using quantum measurements. Due to hardware noise, the list of observed bitstrings would contain contributions from errors on the preceding gates and the measurement itself. To mitigate the measurement errors, we obtain the correction by measuring the calibration matrix for the control qubits (with 8192 shots per measurement) using Qiskit Ignis API. The correction is then applied to the observed distribution with a least-squares fitting approach. The gate errors accumulate throughout the circuit execution and are difficult to correct. Instead, in Aqcel, we opt to ignore the observed bitstrings with occurrence below certain "cutoff" thresholds, under the assumption that the gate errors act as a perturbation that inserts spurious computational basis states with small amplitudes into the system. This can be improved in future with additional computational complexity by using gate error mitigation such as the zero noise extrapolation mentioned in Sec. 1.

The cutoff thresholds are defined as follows. We consider the errors in the $U_{1,2,3}$ and CNOT gates separately for all the hardware qubits. The reported error rates at the time of the experiment, measured during the preceding calibration run of the hardware, are used for the calculations. Let the $U_{1,2,3}$ and CNOT error rates be $\epsilon_U^{(i)}$ and $\epsilon_{CX}^{(i,j)}$, respectively, with i and j indicating qubits that the gates act on. We can approximately calculate the probability p_ϵ of measuring the states with at least one gate error occurring anywhere in the circuit by performing qubit-wise (index-dependent) multiplications of the error rates:

$$p_\epsilon = 1 - \left[\prod_i \left(1-\epsilon_U^{(i)}\right)^{n_U^{(i)}} \prod_{i\neq j}\left(1-\epsilon_{CX}^{(i,j)}\right)^{n_{CX}^{(i,j)}}\right] \sim N_{\rm CX}\epsilon_{\rm CX}\,, \quad (2)$$

where $n_U^{(i)}$ and $n_{CX}^{(i,j)}$ are the number of $U_{1,2,3}$ and CNOT gates acting on the corresponding qubits, respectively. In the last approximation, we have assumed that all CNOT errors are equal, much larger than single gate errors but still much smaller than one: $\epsilon_U^{(i)} \ll \epsilon_{CX}^{(i,j)} = \epsilon_{CX} \ll 1$. The first cutoff threshold is $s_\epsilon^{\rm high} := p_\epsilon$, corresponding to making an extreme assumption that any occurrence of a gate error during circuit execution results in a specific bitstring being observed at the measurement, and attempting to discard that bitstring. The second threshold, $s_\epsilon^{\rm low} := p_\epsilon/2^m$, where m is the number of the measured control qubits, is related to another extreme assumption that the gate errors result in a uniform distribution of all possible bitstrings. The third and final threshold is the average of the above two, $s_\epsilon^{\rm med} := (s_\epsilon^{\rm low} + s_\epsilon^{\rm high})/2$.

It should be noted that p_ϵ increases as the circuit execution proceeds, as it is obtained by multiplying the error rates of all the preceding gates in the circuit. As an alternative strategy, we also examine static thresholds s_ϵ^f that are kept constant throughout the circuit, with values between 5% and 40%. We also consider capping the dynamic thresholds $s_\epsilon^{\rm low}$, $s_\epsilon^{\rm med}$, and $s_\epsilon^{\rm high}$ at 25% (the reason behind the 25% will be given later).

Discarding all bitstrings with occurrence under certain thresholds obviously introduces errors of its own. For example, we observe that discarding bitstrings using unbounded $s_\epsilon^{\rm high}$ as the threshold for the one-branching step QPS simulation results in an elimination of most of the controlled gates in the later part of the circuit, rendering the circuit practically meaningless. Therefore, the actual threshold to be used with Aqcel should be chosen considering the tradeoff between the efficiency of the circuit optimization and the accuracy of the optimized circuit[*5].

Detailed outcomes of the gate counts (and fidelity measurements introduced below) with the dynamic and static thresholds are described in Appendix A.4. The summary of the measurements is described below, with the results from other optimization

[*5] In the actual implementation, the threshold corresponding to 5% of the number of shots is applied to all the three cases to suppress contributions from imperfect measurement error mitigation.

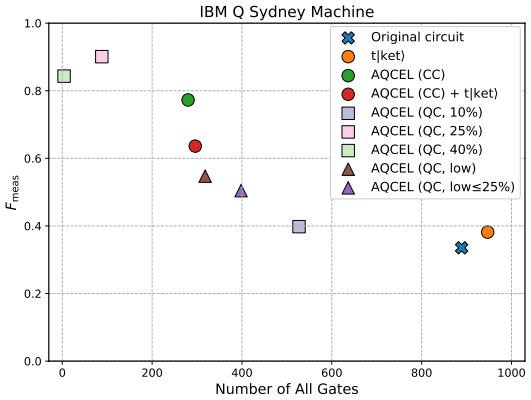


Fig. 5 Fidelity F_{meas} versus the number of native gates for the one-branching step QPS circuit transpiled considering the ibmq_sydney topology before and after optimization under different schemes. These transpiled circuits are executed on the sydney to obtain the F_{meas}.

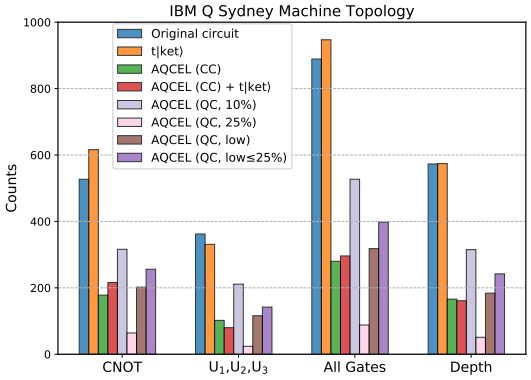


Fig. 6 Numbers of single-qubit ($U_{1,2,3}$) gates, CNOT gates and the sum of the two as well as the depth of the one-branching step QPS circuit transpiled considering the ibmq_sydney topology before and after optimization under different schemes.

schemes.

To evaluate the accuracy of the optimized circuit, we consider a classical fidelity of the final state of the circuit, which is defined in terms of the probability distribution of the computational basis states observed in the measurement at the end of the circuit. This quantity, denoted as F and referred to as just "fidelity" hereafter, is given by

$$F = \sum_k \sqrt{p_k^{\mathrm{orig}} p_k^{\mathrm{opt}}}, \tag{3}$$

where the index k runs over the computational basis states. The quantities p_k^{orig} and p_k^{opt} are the probabilities of observing k in the original and optimized circuits, respectively.

In fact, we compute two fidelity values for each optimization method. The first, denoted F_{sim}, aims to quantify the amount of modifications to the original circuit affected by the optimization at the algorithmic level. To calculate F_{sim}, both p^{orig} and p^{opt} are computed using the state-vector simulation. The unity of F_{sim} indicates that the optimized circuit is identical to the original circuit (up to a possible phase difference on each of the qubits), while a value different from unity gives a measure of how much the optimization has modified the circuit.

The second fidelity value, F_{meas}, is computed using measurements with actual quantum computer. The p^{opt} is estimated from

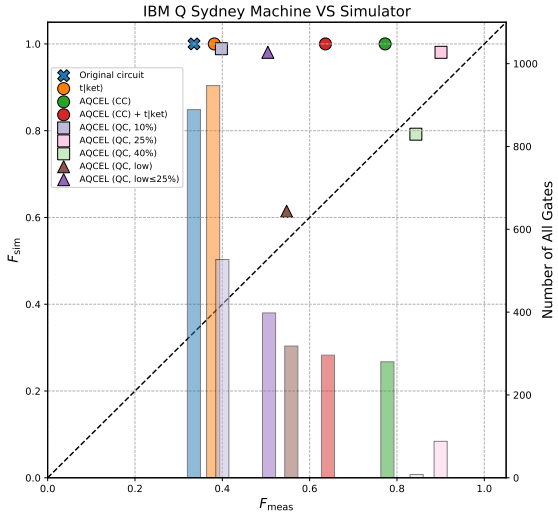


Fig. 7 Fidelities F_{meas} versus F_{sim} for the one-branching step QPS circuit transpiled considering the ibmq_sydney topology before and after optimization under different schemes. These transpiled circuits are executed on the sydney to obtain the F_{meas} and a state-vector simulator to obtain the F_{sim}. The vertical bars represent the total number of gates in the circuits (gauged by the right-hand vertical axis).

the rate of a bitstring occurring in a large number of repeated measurements. The p^{orig} is computed using simulation as for the F_{sim}. Even if the optimized circuit is identical to the original circuit, the presence of noise will make $F_{\mathrm{meas}} < 1$, with the discrepancy getting larger when more gates (in particular CNOT gates) are present in the circuit. Removing CNOT gates to obtain the optimized circuit will lower the overall effect of noise, raising the value of F_{meas}. However, in some cases the CNOT gate removal also affects low-amplitude computational basis states, meaning the optimized circuit can differ from the original circuit, that might suppress the F_{meas} value. Thus, F_{meas} is a measure that takes into account the tradeoff of making the circuit shorter and changing the circuit through optimization.

In Fig. 5, the fidelity F_{meas} versus the number of native gates ($U_{1,2,3}$, CNOT) before and after optimization is shown, where the Aqcel optimization is performed using the classical simulation, labelled as "(CC)" in the figure. One can see that shortening the circuit does increase the F_{meas} from the original circuit, as expected. The measurements are performed 81920 times for each circuit to obtain the F_{meas} values, and measurement error mitigation is not used in these or following F_{meas} measurements.

The results obtained from different approaches for finding nonzero-amplitude basis states and different choices of cutoff thresholds are summarized in Figs. 5 and 6. It is worth noting that most of the Aqcel-based optimization improve the F_{meas} value over the t|ket⟩-only optimization. Another interesting finding is that the determination of bitstring probabilities with quantum measurements brings a better performance than the identification of nonzero amplitudes with classical calculation, if the cutoff threshold is set properly (25% for this case). A qualitative explanation for this would be that the quantum measurements and the cutoff serve to remove qubit controls over low-amplitude basis states, where such states contribute little to the final result but the existence of the controlled gates produces those spurious states under the effect of hardware noise. An exact identifica-

tion of computational basis states with nonzero amplitudes using classical simulation does not allow removing such qubit controls, effectively degrading the F_{meas}.

Figure 7 shows the fidelities F_{sim} versus F_{meas} before and after optimization under different schemes. The figure shows that the F_{sim} is identical to unity for all circuits optimized using the classical simulation, validating that the optimization has not affected the computational accuracy with respect to the original circuit. Although it also shows that the F_{sim} is slightly lowered from unity for all circuits optimized using actual measurements on the ibmq_sydney, the F_{meas} improves from the original circuit because gate reductions suppress the effect of hardware noise.

4. Conclusion

We have proposed a new protocol, called Aqcel, for analyzing quantum circuits to identify recurring sets of gates and remove redundant controlled operations. The heart of the redundant controlled operations removal resides in the identification of zero- or low-amplitude computational basis states. In particular, this procedure can be performed through measurements using a quantum computer in polynomial time, instead of classical calculation that scales exponentially with the number of qubits. Although removing qubit controls triggered in low-amplitude states will produce a circuit that is functionally distinct from the original, it is observed that this may be a desirable feature in some cases under the existence of hardware noise. If a quantum circuit contains recurring sets of quantum gates, those gates will be considered as candidates for further optimization in terms of both gate synthesis and hardware implementation. In the proposed protocol, the underlying technique to identify recurring gate sets is demonstrated, leading to the possibility of hardware-aware optimization of such gates including microwave pulse controls.

We have explored the Aqcel optimization scheme using the quantum parton shower simulation, a prototypical quantum algorithm for high-energy physics. For this algorithm, the proposed scheme shows a significant reduction in gate counts with respect to t|ket⟩, which is one of the industry-standard optimization tools, while retaining the accuracy of the probability distributions of the final state.

Acknowledgments We acknowledge the use of IBM Quantum Services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

CWB and BN are supported by the U.S. Department of Energy, Office of Science under contract DE-AC02-05CH11231. In particular, support comes from Quantum Information Science Enabled Discovery (QuantISED) for High Energy Physics (KA2401032).

We would like to thank Ross Duncan and Bert de Jong for useful discussions about the ZX Calculus.

References

[1] Preskill, J.: Quantum Computing in the NISQ era and beyond, *Quantum*, Vol. 2, p. 79 (online), DOI: 10.22331/q-2018-08-06-79 (2018).
[2] Mott, A., Job, J., Vlimant, J. R., Lidar, D. and Spiropulu, M.: Solving a Higgs optimization problem with quantum annealing for machine learning, *Nature*, Vol. 550, No. 7676, pp. 375–379 (online), DOI: 10.1038/nature24047 (2017).
[3] Terashi, K., Kaneda, M., Kishimoto, T., Saito, M., Sawada, R. and Tanaka, J.: Event Classification with Quantum Machine Learning in High-Energy Physics, *Comput. Softw. Big Sci.*, Vol. 5, No. 1, p. 2 (online), DOI: 10.1007/s41781-020-00047-7 (2020).
[4] Guan, W., Perdue, G., Pesah, A., Schuld, M., Terashi, K., Vallecorsa, S. and Vlimant, J.-R.: Quantum Machine Learning in High Energy Physics, *Machine Learning: Science and Technology*, (online), DOI: 10.1088/2632-2153/abc17d (2020).
[5] Zlokapa, A., Anand, A., Vlimant, J.-R., Duarte, J. M., Job, J., Lidar, D. and Spiropulu, M.: Charged particle tracking with quantum annealing-inspired optimization (2019).
[6] Bapst, F., Bhimji, W., Calafiura, P., Gray, H., Lavrijsen, W. and Linder, L.: A Pattern Recognition Algorithm for Quantum Annealers, *Comput. Softw. Big Sci.*, Vol. 4, p. 1 (online), DOI: 10.1007/s41781-019-0032-5 (2019).
[7] Wei, A. Y., Naik, P., Harrow, A. W. and Thaler, J.: Quantum algorithms for jet clustering, *Phys. Rev. D*, Vol. 101, No. 9, p. 094015 (online), DOI: 10.1103/PhysRevD.101.094015 (2020).
[8] Das, S., Wildridge, A. J., Vaidya, S. B. and Jung, A.: Track clustering with a quantum annealer for primary vertex reconstruction at hadron colliders (2019).
[9] Cormier, K., Di Sipio, R. and Wittek, P.: Unfolding measurement distributions via quantum annealing, *JHEP*, Vol. 11, p. 128 (online), DOI: 10.1007/JHEP11(2019)128 (2019).
[10] Provasoli, D., Nachman, B., Bauer, C. and de Jong, W. A.: A quantum algorithm to efficiently sample from interfering binary trees, *Quantum Science and Technology*, Vol. 5, No. 3, p. 035004 (online), DOI: 10.1088/2058-9565/ab8359 (2020).
[11] Bauer, C. W., Nachman, B., Provasoli, D. and De Jong, W. A.: A quantum algorithm for high energy physics simulations, *Phys. Rev. Lett.*, Vol. 126, No. 6, p. 062001 (online), DOI: 10.1103/PhysRevLett.126.062001 (2021).
[12] Sivarajah, S., Dilkes, S., Cowtan, A., Simmons, W., Edgington, A. and Duncan, R.: t—ket⟩: a retargetable compiler for NISQ devices, *Quantum Science and Technology*, Vol. 6, No. 1, p. 014003 (online), DOI: 10.1088/2058-9565/ab8e92 (2020).
[13] Häner, T., Steiger, D. S., Svore, K. and Troyer, M.: A software methodology for compiling quantum programs, *Quantum Science and Technology*, Vol. 3, No. 2, p. 020501 (online), DOI: 10.1088/2058-9565/aaa5cc (2018).
[14] Quantum AI team and collaborators: Cirq (2020).
[15] Smith, R. S., Peterson, E. C., Davis, E. J. and Skilbeck, M. G.: quilc: An Optimizing Quil Compiler (2020).
[16] Murali, P., Baker, J. M., Abhari, A. J., Chong, F. T. and Martonosi, M.: Noise-Adaptive Compiler Mappings for Noisy Intermediate-Scale Quantum Computers (2019).
[17] Shervashidze, N., Schweitzer, P., van Leeuwen, E. J., Mehlhorn, K. and Borgwardt, K. M.: Weisfeiler-Lehman Graph Kernels, *J. Mach. Learn. Res.*, Vol. 12, No. null, p. 25392561 (online), DOI: 10.5555/1953048.2078187 (2011).
[18] Hagberg, A. A., Schult, D. A. and Swart, P. J.: Exploring Network Structure, Dynamics, and Function using NetworkX, *Proceedings of the 7th Python in Science Conference* (Varoquaux, G., Vaught, T. and Millman, J., eds.), Pasadena, CA USA, pp. 11 – 15 (2008).
[19] Barenco, A., Bennett, C. H., Cleve, R., DiVincenzo, D. P., Margolus, N., Shor, P., Sleator, T., Smolin, J. A. and Weinfurter, H.: Elementary gates for quantum computation, *Phys. Rev. A*, Vol. 52, pp. 3457–3467 (online), DOI: 10.1103/PhysRevA.52.3457 (1995).
[20] Coecke, B. and Duncan, R.: Interacting quantum observables: categorical algebra and diagrammatics, *New Journal of Physics*, Vol. 13, No. 4, p. 043016 (online), DOI: 10.1088/1367-2630/13/4/043016 (2011).
[21] Duncan, R., Kissinger, A., Perdrix, S. and van de Wetering, J.: Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus, *Quantum*, Vol. 4, p. 279 (online), DOI: 10.22331/q-2020-06-04-279 (2020).
[22] Jordan, S. P., Lee, K. S. M. and Preskill, J.: Quantum Algorithms for Quantum Field Theories, *Science*, Vol. 336, pp. 1130–1133 (online), DOI: 10.1126/science.1217069 (2012).
[23] Aleksandrowicz, G. et al.: Qiskit: An Open-source Framework for Quantum Computing (2019).
[24] Van Rossum, G. and Drake, F. L.: *Python 3 Reference Manual*, CreateSpace, Scotts Valley, CA (2009).
[25] IBM Quantum: (2021).

Appendix

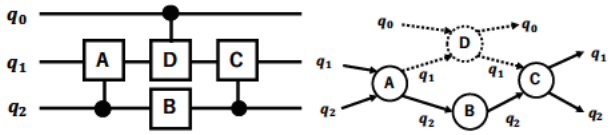


Fig. A·1 An example of quantum circuit (left) and its subgraph ($G' = \{A, B, C\}$) removed in our pattern recognition algorithm (right). A functionality of the corresponding circuit depends on the intermediate gate (D)).

A.1 Algorithms of graph pattern recognition

The pattern recognition algorithm of recurring set of quantum gates (RSG) is described in Algorithm 2. This algorithm is based on depth-first search with heuristic pruning.

First, RSG candidates are built from seeding a quantum gate (node) by seeking possible combinations of RGSs that have descending connected quantum gates. A target node used as a seed, i.e., the beginning node, is selected with postorder traversal with a memorization technique to avoid a repeating calculation. The computational complexity of the algorithm is $O(N_{\text{nodes}}!)$*6. Due to a large number of combinations of recurring gates, the complexity is worse than the typical complexity of a classical computer, $O(n_{\text{qubits}}!)$ or $O(2^{n_{\text{qubits}}})$, because of $N_{\text{nodes}} = n_{\text{gates}} \geq n_{\text{qubits}}$ in most cases, and therefore it loses the benefit of quantum computer. To reduce the computational complexity, we prune the RSG candidates by requiring the length of the longest path, the minimum number and the maximum number of elements in RSG. The requirement of the minimum number of elements rejects a trivial RSG (e.g. $G = \{X\}$). The computational complexity reduces to $O(N_{\text{nodes}}^{N_{\text{thr}}})$*7 where N_{thr} is a threshold value for the pruning, and the classical computer can calculate this in polynomial time when N_{thr} is fixed. However, this algorithm sometimes causes ill-defined RSGs, as shown in Fig. A·1. The functionality of the quantum circuit from such an RSG depends on the intermediate gate that is not used in the RSG. These RSGs are rejected in this algorithm by requiring that there is no node, which is both a child and a parent nodes but not an element of the RSG ($\exists g_i, g_j \subseteq G', \{g_k | g_i \to g_k, g_k \to g_j\} \not\subseteq G'$).

After building the RSG candidates, they are grouped by graph isomorphism using the Weisfeiler Lehman graph hash. The use of graph hash does not ensure that two graphs are isomorphic, but the accuracy is sufficient for our use case. For the Level-1 matching criteria which consider only gate types, we assign the gate type as a node feature and assign nothing for an edge feature. For the Level-2 matching criteria which consider both gate types and qubit *roles*, we assign the gate type as a node feature and assign the target or control label as an edge feature. For the Level-3 matching criteria which consider gate types, qubit *roles* and *indices*, we assign the gate type as a node feature and assign the absolute qubit index as an edge feature.

Finally, the top-k RSGs are selected based on the frequency times the graph size.

*6 The i-th node has $N_{\text{nodes}} - i$ RSG candidates in the worst case. Therefore, the computational complexity of the combination of the number of child-node's RSG candidates is $O(N_{\text{nodes}}!)$.

*7 We take N_{thr} RSG candidates from N_{nodes} nodes. Therefore, the computational complexity is $\binom{N_{\text{nodes}}}{N_{\text{thr}}} \approx N_{\text{nodes}}^{N_{\text{thr}}}$.

Algorithm 2: Gate set pattern recognition with DAG

```
for all quantum gate (node) (g_i) in the circuit (G) do
  for all subset (G') beginning with the target node (g_i) do
    if the longest path is longer than the threshold then
      continue
    end if
    if number of elements in subset is out of thresholds then
      continue
    end if
    if ∃g_i, g_j ⊆ G', {g_k|g_i → g_k, g_k → g_j} ⊄ G' then
      continue
    end if
    G' is a RSG candidate.
  end for
end for
Make a set of RSGs (S(h) = {G'|hash(G') = h})
Select top-k sets of RSGs (S(h)) ordering by the frequency |S(h)| times RSG size (|G'|)
```

A.2 General conditions to eliminate qubit controls

Given a multi-qubit controlled gate $C^m[U]$ and a system in the "undetermined" state $|\psi\rangle$ following the classification in Sec. 2.2.1, we can derive the condition for removal of a part of the controls to be allowed in the following way.

Let x be the number of controls to be removed. Without loss of generality, the decomposition of $|\psi\rangle$ can be rewritten as

$$|\psi\rangle = \sum_{i,l,k} \tilde{c}_{i,l,k} |i\rangle_{\text{ctl}'} \otimes |l\rangle_{\text{free}} \otimes |k\rangle , \tag{A.1}$$

where $|\cdot\rangle_{\text{ctl}'}$ and $|\cdot\rangle_{\text{free}}$ are the states of the $m - x$ remaining control qubits and the x qubits from which the controls are removed. From Eq. (1),

$$|i\rangle_{\text{ctl}'} \otimes |l\rangle_{\text{free}} = |2^x i + l\rangle_{\text{ctl}} , \tag{A.2}$$

and therefore

$$\tilde{c}_{i,l,k} = c_{2^x i + l, k}. \tag{A.3}$$

Applying the original controlled gate to $|\psi\rangle$ yields

$$C^m[U]|\psi\rangle = \sum_{j=0}^{2^m-2} \sum_k c_{j,k} |j\rangle |k\rangle + \sum_k c_{2^m-1,k} |2^m - 1\rangle U |k\rangle , \tag{A.4}$$

where ket subscripts and the tensor product symbols are omitted for simplicity. In contrast, the new gate with fewer controls give

$$C^{m-x}[U]|\psi\rangle = \sum_{i=0}^{2^{m-x}-2} \sum_{l,k} \tilde{c}_{i,l,k} |i\rangle |l\rangle |k\rangle + \sum_{l,k} \tilde{c}_{2^{m-x}-1,l,k} |2^{m-x} - 1\rangle |l\rangle U |k\rangle . \tag{A.5}$$

For the removal of x qubit controls to be allowed, the right hand sides of Eqs. (A.4) and (A.5) must be identical. This requires

$$\sum_{l=0}^{2^x-2}\sum_k \tilde{c}_{2^{m-x}-1,l,k}\,|2^{m-x}-1\rangle\,|l\rangle\,U\,|k\rangle = \sum_{l=0}^{2^x-2}\sum_k c_{2^m-2^x+l,k}\,|2^m-2^x+l\rangle\,|k\rangle\,. \quad \text{(A.6)}$$

Denoting

$$U\,|k\rangle = \sum_{k'} u_{kk'}\,|k'\rangle \quad \text{(A.7)}$$

and recalling Eqs. (A.2), Eq. (A.6) implies (replacing $k' \leftrightarrow k$ on the left hand sides)

$$\sum_{l=0}^{2^x-2}\sum_{k,k'} \tilde{c}_{2^{m-x}-1,l,k'}\,u_{k'k}\,|2^{m-x}-1\rangle\,|l\rangle\,|k\rangle = \sum_{l=0}^{2^x-2}\sum_k c_{2^m-2^x+l,k}\,|2^{m-x}-1\rangle\,|l\rangle\,|k\rangle\,. \quad \text{(A.8)}$$

Then, with Eq. (A.3), we have

$$\sum_{k'} \tilde{c}_{2^{m-x}-1,l,k'}\,u_{k'k} = \tilde{c}_{2^{m-x}-1,l,k} \quad \forall l,k. \quad \text{(A.9)}$$

Equation (A.9) holds if the row vector $\{\tilde{c}_{2^{m-x}-1,l,k}\}_k$ is an eigenvector of the matrix u with eigenvalue 1 under right multiplication for all l, or if $\tilde{c}_{2^{m-x}-1,l,k} = 0$ for all l and k.

Since the cost of exactly computing the complex amplitudes of the quantum state is high, in Aqcel we only consider this second condition of $\tilde{c}_{2^{m-x}-1,l,k} = 0$. When using quantum measurements to estimate the bitstring probabilities at the control qubits, this requirement translates to observing no bitstring with 1 in the qubits that are considered for control removal.

A.3 Computational resources for the proposed optimization scheme

The computational cost needed to perform the proposed optimization scheme is evaluated here. We consider a quantum circuit that contains n qubits and N multi-qubit controlled gates, each acting on m control qubits and one target qubit.

The elimination of adjacent gate pairs proceeds, for each gate, by checking a pair-wise matching to the next gate until the end of the gate sequence. Since the gate could contain at most n qubits, the computational cost would be $O(nN)$.

The next step in the optimization scheme is the identification of computational basis states. If we use the classical calculation for simply tracking all the computational basis states whose amplitudes may be nonzero at each point of the circuit without the calculation of the amplitudes, it requires the computation of $O(N2^n)$ states, so grows exponentially with n. This method allows the lower computational resource than a state-vector simulation though it neglects rare eliminations of redundant controlled operations. If we measure the control qubits at each controlled gate M times using a quantum computer, the total number of gate operations and measurements is given by $M\{m + (1 + m) + (2 + m) + \cdots + (N - 1 + m)\} = \frac{1}{2}MN(N-1) + mMN$. Therefore, the computational cost grows polynomially in $O(MN^2 + mMN)$.

We next consider removing redundant qubit controls from a controlled gate with m control qubits. Using a quantum computer that measures the m control qubits M times, the measured number of computational basis states is M if $M < 2^m$, otherwise 2^m. For the classical calculation, the number of basis states is 2^m. Imagine that we choose an arbitrary combination among 2^m possible combinations of new qubit controls on the same controlled gate. If we want to know whether the chosen combination can act as the correct qubit control, we would need to check, for a given measurement done previously with quantum computer, if all the possible "unmeasured" computational basis states satisfy the chosen one or not. This requires $O(m2^m)$. Since this has to be checked for all the measurements, the cost would be $O(Mm2^m)$ if $M < 2^m$, otherwise $O(m4^m)$. Therefore, the overall computation for the determination of redundant qubit controls would become $O(Mm4^mN)$ or $O(m8^mN)$ for N multi-qubit controlled gates, each having 2^m computational basis states. The classical calculation would require $O(m8^mN)$ as well.

It is known than an arbitrary multi-qubit controlled-U gate with m control qubits can be decomposed into $O(m)$ Toffoli and two-qubit controlled-U gates [19]. Therefore, if a controlled gate in the circuit is decomposed in this way, then above computational cost for the redundant qubit controls would become $O(mN)$. With this decomposition, the total number of gate operations and measurement increases due to $O(m)$ times more controlled gates. However, the computational cost for the identification of computational basis states becomes only $\frac{1}{2}mMN(mN-1) + 2mMN$, so still behaves polynomially in $O(m^2MN^2)$ in case the quantum computer is used. For the classical calculation, the cost becomes $O(mN2^n)$.

The final step of the optimization scheme is the elimination of unused qubits. This is just performed by checking qubits that all the gates in the circuit act on, therefore the computational cost is $O(nN)$.

Given that a controlled gate has at most $n-1$ control qubits, the total computational cost for the entire optimization steps is $O(n^2MN^2)$ and $O(nN2^n)$ if the computational basis state measurement is performed using a quantum computer and classical calculation, respectively.

A.4 Detailed results of gate counts and classical fidelity measurements

Here, the results of the optimizations based on the dynamic and static thresholds for the quantum measurements are described.

Figure A·2 shows the gate counts obtained from Aqcel optimizations using the measurements on the sydney with the dynamic cutoff thresholds. The gate counts decrease as the threshold is raised from s_ϵ^{low} to s_ϵ^{high}, as expected. The same distributions obtained with the static thresholds (Fig. A·3) show that almost no gate survives under the threshold of 40%, likely implying a significant loss of accuracy of the computation result. The number of qubits is reduced from 15 to 13 with the threshold of s_ϵ^{low} or s_ϵ^{med}, and to 11 with s_ϵ^{high}. Under the static thresholds, the number of qubits is reduced from 15 to 13 for $5\% \le s_\epsilon^f \le 25\%$, but a significant reduction to 8 is seen for $s_\epsilon^f \ge 35\%$.

The F_{meas} versus gate counts is shown in Fig. A·4 for the dynamic thresholds of s_ϵ^{high}, s_ϵ^{med} and s_ϵ^{low}, as well as the capped variants where the threshold is capped at 25%. The F_{meas} generally improves with higher thresholds, but is worse with s_ϵ^{high}

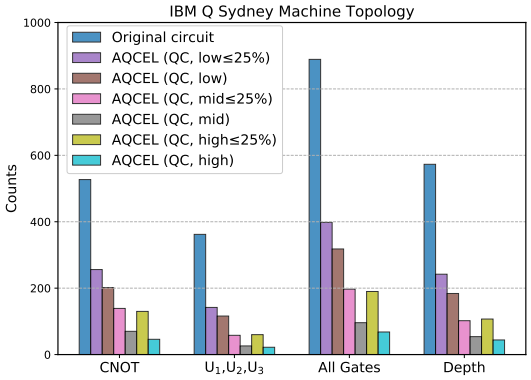


Fig. A·2 Numbers of single-qubit ($U_{1,2,3}$) gates, CNOT gates and the sum of the two as well as the depth of the one-branching step QPS circuit transpiled considering the ibmq_sydney topology before and after optimization. The probabilities of observing various bitstrings in the control qubits are measured using the sydney in the heuristic optimization step, and the three dynamic cutoff thresholds of $s_\epsilon^{\mathrm{low}}$, $s_\epsilon^{\mathrm{med}}$ and $s_\epsilon^{\mathrm{high}}$ are applied, as well as their capped variants.

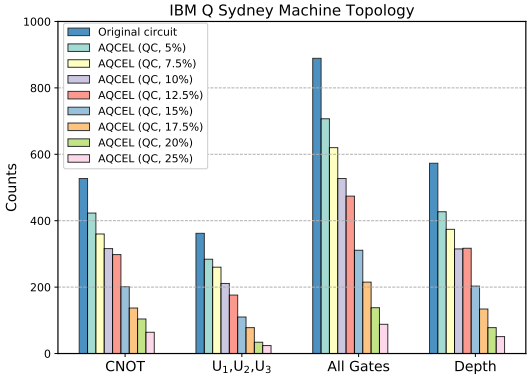


Fig. A·3 Numbers of single-qubit ($U_{1,2,3}$) gates, CNOT gates and the sum of the two as well as the depth of the one-branching step QPS circuit transpiled considering the ibmq_sydney topology before and after optimization. The probabilities of observing various bitstrings in the control qubits are measured using the sydney in the heuristic optimization step, and the static cutoff thresholds of s_ϵ^f are applied.

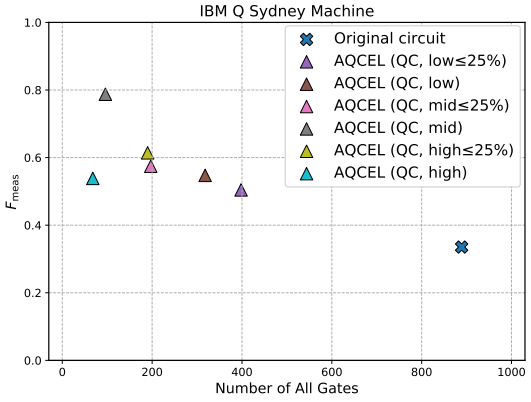


Fig. A·4 Fidelity F_{meas} versus the number of native gates for the one-branching step QPS circuit transpiled considering the ibmq_sydney topology before and after optimization. The probabilities of observing various bitstrings in the control qubits are measured using the sydney in the heuristic optimization step, and the three dynamic thresholds of $s_\epsilon^{\mathrm{low}}$, $s_\epsilon^{\mathrm{med}}$ and $s_\epsilon^{\mathrm{high}}$ are applied, as well as their capped variants. These transpiled circuits are executed on the sydney to obtain the F_{meas}.

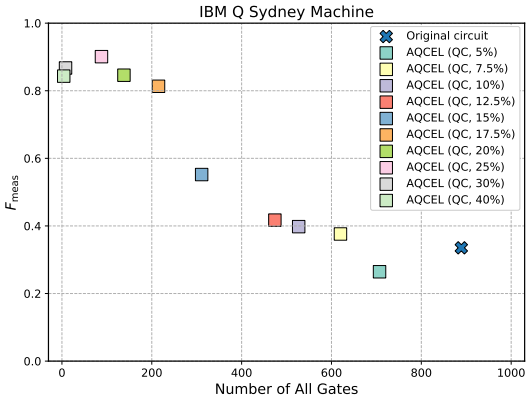


Fig. A·5 Fidelity F_{meas} versus the number of native gates for the one-branching step QPS circuit transpiled considering the ibmq_sydney topology before and after optimization. The probabilities of observing various bitstrings in the control qubits are measured using the sydney in the heuristic optimization step, and the static thresholds of s_ϵ^f are applied. These transpiled circuits are executed on the sydney to obtain the F_{meas}.

than with $s_\epsilon^{\mathrm{med}}$ because the accuracy is significantly degraded due to too aggressive threshold in the former case. The capped variants leave more gates in the circuit and have lower F_{meas} than the unbounded cases, except for when using $s_\epsilon^{\mathrm{high}}$, where capping of the threshold seems to mitigate the loss of accuracy from overly aggressive optimization.

Figure. A·5 shows the F_{meas} versus gate counts for the s_ϵ^f thresholds. We observe that with increasing s_ϵ^f value the F_{meas} first increases, which indicates more aggressive optimization, but that at some point the F_{meas} starts to worsen, signaling that the optimized circuit becomes too far from the original circuit. For the circuit considered here, the performance of the optimization seems best for $s_\epsilon^f \sim 25\%$.