arXiv:2501.14447v2 [quant-ph] 30 Jan 2026

Quantum Circuit Optimization by Graph Coloring

Hochang Lee, Kyung Chul Jeong, and Panjin Kim

The Affiliated Institute of ETRI, Daejeon 34044, Korea

This work shows that minimizing the depth of a quantum circuit composed of commuting operations reduces to a vertex coloring problem on an appropriately constructed graph, where gates correspond to vertices and edges encode non-parallelizability. The reduction leads to algorithms for circuit optimization by adopting any vertex coloring solver as an optimization backend. The approach is validated by numerical experiments as well as applications to known quantum circuits, including finite field multiplication and QFT-based addition.

1 Introduction

Quantum circuits composed of finitely many unitary operations that mutually commute arise in several computational settings, including instantaneous quantum polynomial [1, 2], commuting local Hamiltonians [3, 4], and oracle constructions in quantum algorithms [5, 6]. Throughout this work, we refer to such circuits as commuting circuits. Because the underlying unitary operations mutually commute, such circuits are naturally amenable to parallelization, i.e., certain operations can be executed in parallel whenever the hardware supports it. Hardware-dependent constraints therefore determine which operations can be executed concurrently, and consequently lead to the notion of *depth* of a circuit.

In circuit synthesis, a function can generally be implemented by multiple constructions, and selecting one that minimizes resource usage is a central consideration in circuit design [7]. Under a fixed number of qubits, a design with smaller depth is typically preferred, as it reflects better utilization of the available parallelism. For this reason, among others, optimizing the depth of a circuit is an active area of research [7, 8, 9, 10, 11].

This study establishes a formal connection between depth optimization of commuting circuits and a well-known NP-hard problem in graph theory. The results are summarized as follows:

- Depth optimization of a commuting circuit is shown to be polynomial time reducible (in the number of operations) to solving the vertex coloring problem. The opposite direction is also proven to hold, meaning that the two problems are equivalent.
- The reduction implies *any algorithm* for the vertex coloring can be utilized to optimize a commuting circuit.
- Two concrete algorithms are numerically tested to validate the method.
- The applicability is demonstrated by optimizing circuits for finite field multiplications and additions by quantum Fourier transform (QFT). The first application exhibits the improvement upon Maslov et al.'s design [5] and the second one gives tradeoff options for Draper's quantum addition [6].

Panjin Kim: pansics@nsr.re.kr

The paper is organized as follows. Section 2 introduces the necessary preliminaries. Section 3 presents the main reduction and its proofs, followed by algorithms in Section 4. Section 5 discusses applications of the method, and Section 6 concludes the paper.

2 Preliminaries

This section introduces necessary definitions, known facts, and related works. Readers are assumed to be familiar with the basics of quantum information [12].

2.1 Parallel Execution of Commuting Operations

The purpose of this subsection is to bring out the motivation. Consider a Toffoli gate denoted by CX_{ijk} where i, j are indices of the control bits and k is the index of the target bit. Let $\boldsymbol{v}_1$ be an ordered set, $\boldsymbol{v}_1 = (CX_{134}, CX_{235})$ and consider a quantum circuit implementing gates in $\boldsymbol{v}_1$. Notice that the two gates share the same control bit (indexed by 3), and we assume a quantum machine cannot execute these gates simultaneously. Therefore in this case we count the depth of $\boldsymbol{v}_1$ as 2. (Definition of the depth is given below.)

Now let $\boldsymbol{v}_2 = (CX_{134}, CX_{235}, CX_{167}, CX_{268})$. The depth of $\boldsymbol{v}_2$ is 3 since CX_{235} and CX_{167} do not share any control or target bit, thus the two gates are simultaneously executable by the machine (assumed). There is a way to further decrease the depth of the circuit. Notice that the gates in $\boldsymbol{v}_2$ are pair-wise commuting; any permutation of the four gates leads to the same behavior. If one picks an order $\boldsymbol{v}_2' = (CX_{134}, CX_{268}, CX_{235}, CX_{167})$, and then the depth becomes 2.

2.2 Gate Ordering Problem

Definition 1 (Commuting operations)**.** *Two unitary operations written by matrices $v, w \in \mathbb{C}^{z\times z}, z \in \mathbb{Z}^+$ are commuting if and only if $vw = wv$.*

Definition 2 (Depth of a circuit)**.** *Given a hardware system, operations are parallelizable if and only if they can be simultaneously executed on the system. A layer is the maximal set of operations such that the operations are parallelizable. Depth of a circuit is the number of consecutive layers to run the circuit.*

Be reminded that depth is defined in terms of operations, not mere elementary gates. For example in a modular exponentiation circuit for modulus N (that appears in factoring or discrete logarithm algorithms), the modular multiplication takes place $O(\log N)$ times, thus if we count the multiplication-depth it would read $O(\log N)$. However, the multiplication can be further decomposed into $O(\log N)$ additions, in which each addition takes another $O(\log N)$ elementary operations, resulting in $O((\log N)^3)$ depth of elementary gates for the exponentiation. Notice in this example that modular multiplications are all pair-wise commuting whereas not all elementary gates are commuting during the exponentiation.

From now on, the term *gate* will often be used instead of operation. Let $\boldsymbol{v}$ be a sequence of gates where the gates are pair-wise commuting. The problem we want to solve is defined as follows:

Definition 3 (Gate ordering)**.** *For a given circuit $\boldsymbol{v}$ consisting of commuting gates, a gate ordering problem is to reorder the gates in $\boldsymbol{v}$ such that the depth is minimized.*

Here we do not consider cancelling or replacement of a subcircuit [13], for example, applying the same CNOT gate twice in a row is equivalent to doing nothing, but such optimization is not a point of interest.

2.3 Graph coloring

A graph is an ordered set $G = (V, E)$, where V is a set of vertices $v_1, \ldots, v_{|V|}$ and E is a set of edges denoted by e_{ij} connecting two vertices v_i and v_j. Only simple and undirected graphs are considered in this work [14].

A graph coloring of $G = (V, E)$ is a function $c : V \to C$, where C is a set of colors. Without loss of generality, we will use a convention that the colors are drawn incrementally from the natural numbers, $C = \{1, 2, 3, \ldots\} \subseteq \mathbb{N}$.

Definition 4 (Proper coloring)**.** *A proper coloring c of G is a coloring such that $c(v_i) \neq c(v_j)$ for all $e_{ij} \in E$.*

Notice that $e_{ij} \notin E$ does not mean $c(v_i) = c(v_j)$. Only the proper coloring is considered in this work, thus 'proper' will be dropped hereafter.

A colored graph of $G = (V, E)$ by a coloring c is denoted by $\widetilde{G} = (V, E, c)$. Define $\mathbf{Col}(G)$ as the set of colored graphs of G.

Definition 5 (Vertex coloring problem)**.** *Given a graph $G = (V, E)$, a vertex coloring problem is to find a colored graph $(V, E, c) \in \mathbf{Col}(G)$ such that $|\{c(v) : v \in V\}|$ is minimized.*

The term graph coloring frequently refers to vertex coloring throughout the paper. Chromatic number of a graph is the smallest number that is required to color the graph.

2.4 Related works

The optimization of a logic circuit has long been studied. The minimum circuit size problem asks if a Boolean function is computable by a circuit of limited size, and several papers of evidence suggest that the problem might be NP-hard [15, 16, 17, 18, 19]. In quantum computing, recent works show that optimizing a quantum circuit in some occasions is NP-hard [20, 21], and specifically in the paper by Wetering and Amy [21], wide range of optimization targets have been considered including T-count, T-depth, Toffoli-count, Toffoli-depth, CNOT-count and so on, proving that the quantum circuit optimization is generally hard. Along the line of study, one of the conclusions drawn from this work is that the depth optimization of a commuting quantum circuit is NP-hard, implying that this small class of quantum circuits is already hard to optimize.

Circuit optimization is one of the active research areas in the quantum computing community. On the one hand, it is to ask theoretically how small a circuit can be, but on the other hand, finding better optimization is closely related to overcoming or mitigating noise problems [22]. The aspect of optimization varies, for example T-depth optimization of Clifford+T circuits [23], pattern matching [24], local synthesis [25], or design automation [26]. This work pursues an optimization by the reordering of gates, in which the goal and the approach are different from previous works. The closest notion in the literature is the commutation analysis (built-in function of Qiskit [27]) whose purpose is to find cancellations or mergers, or to lower the overhead accounting for a hardware topology. We do not consider cancellations or so, but are interested in how much parallelization is achievable solely by reordering.

There exist works on quantum circuit optimization utilizing coloring in graph theory, by looking qubits as vertices. For example, edge coloring of a graph has been used to optimize the depth of Clifford circuits [28], or binary field inversion [29]. Such an approach is natural in the sense that once wires (qubits) are viewed as vertices and gates are considered as edges interconnecting the wires, the result of edge coloring tells which gates are parallelizable because two edges being connected to the same vertex implies they share the same qubit as the reference. Moreover, in cases where gates operating on more than two qubits are used, the approach can be generalized by considering hypergraphs where an edge can connect more than two vertices. For example in [30], the lower bound on the depth of the single optimization layer is given in terms of the chromatic index of the hypergraph [31].

Graph coloring has been used to optimally group Pauli strings to minimize the number of measurements in variational quantum eigensolver [32, 33, 34, 35, 36]. Given a set of Pauli strings, one tries to partition them into subsets such that all strings within each subset commute, while minimizing the number of subsets. The strings in each subset are simultaneously diagonalizable, allowing simultaneous measurement. This problem is reduced to solving graph coloring where the edges are drawn if two strings (vertices) are not commuting. The optimal partitioning of a set of Pauli strings is also considered in reducing the depth of a circuit for the time evolution of Sachdev-Ye-Kitaev model [37]. The authors successfully reduce the depth of the circuit by using the graph coloring algorithm.

We extend the idea by considering a circuit *consisting only of commuting operations*. The optimization approach developed in this paper uses an algorithm for graph coloring as the core routine. Any graph coloring algorithm can be used, for example including ones from previous works [38, 39, 40, 41], and it will soon be shown that the output of the graph coloring is directly related with the optimized circuit.

3 Reduction from gate ordering to graph coloring

The gate ordering problem is reduced to the graph coloring problem in this section. The opposite direction is also given in Appendix A, showing that the two problems are equivalent. Since the graph coloring problem is in NP-hard [14], it implies optimizing a commuting circuit is also hard.

3.1 Transformations

Two transformations are introduced before we give the reduction. One is from a circuit to a graph, and the other is from a colored graph to a circuit.

3.1.1 Circuit to graph

Let $\boldsymbol{v}$ be an ordered set of commuting gates, and let $\mathcal{V}$ be a multiset of gates in $\boldsymbol{v}$ (where identical elements may appear multiple times). One can think of $\boldsymbol{v}$ as a circuit and $\mathcal{V}$ as a collection of gates in the circuit. Given $\boldsymbol{v}$, construct a graph $G_{\mathcal{V}}$ of which *vertices are gates* by the following algorithm:

Algorithm 1 ConstructGraph

Input: $\boldsymbol{v} = (v_1, v_2, \ldots)$
Output: $G_{\mathcal{V}} = (V, E)$

$V \leftarrow \{v_1, v_2, \ldots\}$, $E \leftarrow \{\ \}$
for $i = 1$ **to** $|V|$ **do**
  for $j = 1$ **to** $i - 1$ **do**
    if v_i, v_j are not simultaneously executable **then**
      $E \leftarrow E \cup \{e_{ij}\}$
return (V, E)

It is a fairly natural construction. Gates that cannot be executed simultaneously correspond to *connected* vertices that cannot be identically colored. Therefore one first draws vertices corresponding to gates in $\mathcal{V}$, and then draws edges between vertices depending on the parallelizability of the gates. About the time complexity, it checks the parallelizability of two gates $O(|\mathcal{V}|^2)$ times in total.

Notice that any permutation of $\boldsymbol{v}$ results in the same graph by Algorithm 1. This property is used in the reduction later.

3.1.2 Colored graph to circuit

Given a colored graph in which *vertices are gates*, one can compose a circuit corresponding to the colored graph.

Algorithm 2 ToCircuit

Input: $\widetilde{G} = (V, E, c)$ ▷ $c : V \to C$, C being a color set
Output: a sequence of gates $\boldsymbol{v}$

Find a permutation $\rho : \{1, \ldots, |V|\} \to \{1, \ldots, |V|\}$ such that $c(v_{\rho(i)}) \leq c(v_{\rho(i+1)})$
return $(v_{\rho(1)}, v_{\rho(2)}, \ldots, v_{\rho(|V|)})$

Let c_i be $c(v_i)$ for $i \in \{1, \ldots, |V|\}$. What ρ does is sorting c_i such that smaller c_i comes first. We leave the sorting specification for $c_i = c_j$ to users, as whatever comes first does not affect the bound on the circuit depth (see below). The output is self-explanatory; all gates colored with 1 are drawn first, followed by the gates colored with 2, and so on. Therefore the depth is upper bounded by the number of colors used in $\widetilde{G}$. The time complexity of the algorithm reads $O(|V| \log |V|)$.

The resulting circuit satisfies the following:

Remark 6. *The depth of the resulting circuit of Algorithm 2 is either (a) smaller than or (b) equal to the number of colors of the input graph.*

Case (b) is straightforward, thus we leave a brief comment on the smaller case (a). Given a colored graph, unconnected vertices correspond to parallelizable gates, but the colors on these vertices can be different. Therefore, despite the different colors on the vertices, the corresponding gates from the output may happen to be in the same layer. An example is given in Appendix B.

3.2 Reduction

Suppose we are given a commuting circuit $\boldsymbol{v}$ and asked to reorder the gates such that its depth is minimized. We construct a graph $G_{\mathcal{V}}$ corresponding to $\boldsymbol{v}$ by Algorithm 1. The

graph coloring problem is then solved by some solver, outputting a colored graph with the minimum number of colors used. The colored graph is then transformed back into a circuit by Algorithm 2. The remaining task is to prove that the resulting circuit is the one with the minimum depth, completing the reduction from gate ordering problem to graph coloring problem. More notions are introduced below for the task, but they are solely required for the proof.

Recall that we are given a circuit $\boldsymbol{v}$, and $\mathcal{V}$ is the multiset of gates in $\boldsymbol{v}$. Define $A_{\mathcal{V}}$ as the set of all permutations of gates in $\mathcal{V}$. Note that $\boldsymbol{v} \in A_{\mathcal{V}}$. Since we are optimizing a circuit by reordering the gates in $\boldsymbol{v}$, the optimal circuit $\boldsymbol{v}_{\mathrm{opt}}$ (which is not necessarily unique) also belongs to $A_{\mathcal{V}}$; $\boldsymbol{v}_{\mathrm{opt}} \in A_{\mathcal{V}}$.

Recall that $G_{\mathcal{V}}$ is the graph drawn by Algorithm 1 for given $\boldsymbol{v}$. Let $B_{\mathcal{V}} = \mathbf{Col}(G_{\mathcal{V}})$, which is the set of all colored graphs of $G_{\mathcal{V}}$.

Claim 7. *Let $A_{\mathcal{V}}$ and $B_{\mathcal{V}}$ be as defined above. Define $m_{\mathrm{depth}} : A_{\mathcal{V}} \to \mathbb{Z}^{\geq 0}$ as a function computing the depth of a circuit $\boldsymbol{a} \in A_{\mathcal{V}}$. Define $m_{\mathrm{color}} : B_{\mathcal{V}} \to \mathbb{Z}^{\geq 0}$ as a function counting the number of colors used in a colored graph $b \in B_{\mathcal{V}}$. We claim that there exist functions $f : B_{\mathcal{V}} \to A_{\mathcal{V}}$ and $g : A_{\mathcal{V}} \to B_{\mathcal{V}}$ such that,*

- $\forall b \in B_{\mathcal{V}}$, $m_{\mathrm{color}}(b) \geq m_{\mathrm{depth}}(f(b))$,
- $\forall \boldsymbol{a} \in A_{\mathcal{V}}$, $m_{\mathrm{depth}}(\boldsymbol{a}) = m_{\mathrm{color}}(g(\boldsymbol{a}))$.

It will soon be shown that f is Algorithm 2, and g will be introduced later in this subsection. Assuming the claim is correct, we have a straightforward proposition.

Proposition 8. *If $m_{\mathrm{color}}(b) = \min_{b' \in B_{\mathcal{V}}} \left(m_{\mathrm{color}}(b') \right)$ holds, and so does $m_{\mathrm{depth}}(f(b)) = \min_{\boldsymbol{a}' \in A_{\mathcal{V}}} \left(m_{\mathrm{depth}}(\boldsymbol{a}') \right)$.*

Proof. We use the proof by contradiction. Let $b \in B_{\mathcal{V}}$ be the colored graph such that

$$m_{\mathrm{color}}(b) = \min_{b' \in B_{\mathcal{V}}} \left(m_{\mathrm{color}}(b') \right) \ . \tag{1}$$

Now suppose that $m_{\mathrm{depth}}(f(b)) > \min_{\boldsymbol{a}' \in A_{\mathcal{V}}} \left(m_{\mathrm{depth}}(\boldsymbol{a}') \right)$, then there exists $\boldsymbol{a} \in A_{\mathcal{V}}$ such that $m_{\mathrm{depth}}(\boldsymbol{a}) < m_{\mathrm{depth}}(f(b))$, $\boldsymbol{a} \neq f(b)$.

Due to the properties of f and g, we have $m_{\mathrm{depth}}(\boldsymbol{a}) = m_{\mathrm{color}}(g(\boldsymbol{a}))$ and $m_{\mathrm{depth}}(f(b)) \leq m_{\mathrm{color}}(b)$. In other words, there exists $g(\boldsymbol{a}) \in B_{\mathcal{V}}$ satisfying

$$m_{\mathrm{color}}(g(\boldsymbol{a})) = m_{\mathrm{depth}}(\boldsymbol{a}) < m_{\mathrm{depth}}(f(b)) \leq m_{\mathrm{color}}(b) \ ,$$

which contradicts to Eq. (1). □

Proposition 8 tells us that if a colored graph b is a graph achieving the chromatic number, then the circuit $f(b)$ is guaranteed to be a circuit with the minimum depth in $A_{\mathcal{V}}$.

Now we prove the claim. It has already been shown that Algorithm 2 satisfies the property $\forall b \in B_{\mathcal{V}} : m_{\mathrm{color}}(b) \geq m_{\mathrm{depth}}(\mathsf{ToCircuit}(b))$ as in Remark 6. The function g in Claim 7 is given below. Let us handle a coloring function c as a table and fill its entry c_i one by one so that at the end of the algorithm we have $c(v_i) = c_i$ for all $i \in \{1, \ldots, |V|\}$.

Algorithm 3 ToColoredGraph

Input: $\boldsymbol{v} = (v_1, \ldots, v_n) \in A_{\mathcal{V}}$
Output: $\widetilde{G}_{\mathcal{V}} = (V, E, c) \in B_{\mathcal{V}}$

1: $(V, E) \leftarrow \mathsf{ConstructGraph}(\boldsymbol{v})$ ▷ $V = \{v_1, \ldots, v_n\}$
2: $i \leftarrow 1$, $c_1 \leftarrow 1$, $d \leftarrow 1$ ▷ d for depth
3: **while** $i < n$ **do**
4: $\quad j \leftarrow i + 1$
5: $\quad$ **while** $j \leq n$ **do**
6: $\quad\quad$ **if** $E \cap \{e_{kl} : k, l \in \{i, \ldots, j\}\} = \varnothing$ **then**
7: $\quad\quad\quad c_j \leftarrow c_i$
8: $\quad\quad\quad j \leftarrow j + 1$
9: $\quad\quad$ **else**
10: $\quad\quad\quad c_j \leftarrow c_i + 1$
11: $\quad\quad\quad d \leftarrow d + 1$
12: $\quad\quad\quad$ **break**
13: $\quad i \leftarrow j$
14: **return** (V, E, c) ▷ $c(v_i) = c_i$

Let $\boldsymbol{v}$ be an input to ToColoredGraph. Recall that the vertices are gates. Algorithm 3 sequentially identifies the parallelizable gates (vertices), and those vertices are identically colored in Line 7. Whenever a new color is introduced in Line 10, d is also increased by 1. Since d is increased only when v_j is non-parallelizable with any of $v_i, \ldots, v_{j-1}$ (deducible from Line 6), the final value of d at the end is equal to the depth of the circuit. Therefore, we see that $\forall \boldsymbol{a} \in A_{\mathcal{V}}$, $m_{\text{depth}}(\boldsymbol{a}) = m_{\text{color}}(\mathsf{ToColoredGraph}(\boldsymbol{a}))$. Time complexity of Algorithm 3 is $O(n^2)$.

We have shown that Claim 7 does hold, thereby completing the reduction from gate ordering to graph coloring. A reduction for the opposite direction, from graph coloring to gate ordering, is less important as the main purpose of this work is to suggest a strategy for circuit optimization. Therefore we leave the opposite reduction in Appendix A.

3.3 Circuit optimization strategy

The fact that gate ordering is reduced to graph coloring results in a way to optimize the depth of a commuting circuit. A simple strategy is to utilize any (approximate) algorithm for solving graph coloring.

1. Given a circuit, construct the corresponding graph by using Algorithm 1.
2. Solve graph coloring by using known solvers. The output is a colored graph.
3. Compose a circuit corresponding to the colored graph by using Algorithm 2.

It can be deduced from inspecting the reduction steps that an approximate solution of graph coloring leads to at least as good approximations to the gate ordering problem. Let $\boldsymbol{v}$ be a commuting circuit, $\boldsymbol{v}_{\text{opt}}$ be the optimal circuit of $\boldsymbol{v}$, b be an approximate solution for the graph coloring problem of $\mathsf{ConstructGraph}(\boldsymbol{v})$, and let $h' = m_{\text{color}}(b)$. Since b is not optimal, we have $\Delta := h' - h \geq 0$, where h is the chromatic number of the graph. Due to Remark 6, we see that $m_{\text{depth}}(\mathsf{ToCircuit}(b)) - m_{\text{depth}}(\boldsymbol{v}_{\text{opt}}) \leq \Delta$, meaning that any approximate solution b of the graph coloring leads to a circuit with smaller or equal gap to the minimum depth. Note however that the above argument holds only for approximate solutions. If the coloring is optimal, then the depth of the corresponding circuit is equal to the number of colors.

Since graph theory has been studied extensively over history, one may expect the strategy to be effective on some occasions. Computational complexity and the depth of the resulting circuit are determined by the algorithm for graph coloring.

4 Validation

In this section, the validity of the reduction strategy is put to the test by implementing Algorithm 4. Any coloring algorithm can be used, and we choose two well-known algorithms (solvers) and evaluate their performance.

Algorithm 4 OptimizeCircuit

Input: $\boldsymbol{v}$ ▷ input circuit
Output: $\boldsymbol{v}'$ ▷ output circuit

1: $G \leftarrow \mathsf{ConstructGraph}(\boldsymbol{v})$
2: $\widetilde{G} \leftarrow \mathsf{ColoringSolver}(G)$
3: $\boldsymbol{v}' \leftarrow \mathsf{ToCircuit}(\widetilde{G})$
4: **return** $\boldsymbol{v}'$

4.1 Coloring solvers

4.1.1 DSatur

DSatur is a saturation-degree-based greedy algorithm [39]. Saturation degree of a vertex is the number of different colors used by its neighbors. Roughly described, at each step of the algorithm, an uncolored vertex with the highest degree is selected and assigned a color such that the use of a new color is suppressed if possible.

The time complexity is easily estimated. Let the number of vertices of an input graph G be n. At each step, for a selected vertex, the algorithm inspects only its neighbors and then colors it. It repeats for all n vertices, thus the running time is bounded by $O(n^2)$.

About the number of colors of the output, let χ_G be the chromatic number of G. The algorithm is heuristic, and there is no general theoretical bound that characterizes how close DSatur's output is to χ_G. However, the empirical performance of DSatur has been studied extensively, and the benchmark results can be found in various literature [42, 43].

4.1.2 Backtracking

Backtracking is an algorithm for k-coloring, which asks if a given graph is colorable with at most k different colors [44]. Basically, Backtracking performs a depth-first search over color assignments. It colors vertices one by one while respecting constraints. When a vertex cannot be assigned a color anymore without violating the constraints, the algorithm backtracks to change earlier color choices, effectively pruning that branch of the search space. The worst-case time complexity is known to be $O(k^n)$ for k-coloring and it can be used to find the chromatic number by trying different k at most n times.

4.2 Lower bound on the depth of a commuting circuit

In Section 3, we have not specified conditions for operations to be simultaneously executable since the parallelizability depends on the underlying hardware. To analyze the algorithm however, the condition has to be fixed. We begin with the following assumption that two gates sharing no common qubit can be executed simultaneously:

Assumption 9. *Let v, w be unitary operations and Q_v, Q_w be sets of qubits v and w operate on, respectively. The operations v, w are simultaneously executable if and only if $Q_v \bigcap Q_w = \varnothing$.*

When a commuting circuit is generated randomly (each gate is chosen randomly), the corresponding graph by Algorithm 1 becomes a *random intersection graph* [45]. We skip the details on the model, but take the bound on the chromatic number. Let m be the number of qubits of a random commuting circuit and G be the corresponding graph, then

$$\chi(G) \geq \max_{1 \leq q \leq m} |L_q| \ , \tag{2}$$

where $\chi(G)$ is the chromatic number of G, and L_q is the set of gates that act on the q-th qubit. For example for a circuit $\boldsymbol{v}_2 = (CX_{134}, CX_{235}, CX_{167}, CX_{268})$, we have $L_1 = \{CX_{134}, CX_{167}\}$ and $L_2 = \{CX_{235}, CX_{268}\}$. Briefly explained, the largest set L_q naturally forms the maximum clique [14] whose size provides the non-tight lower bound on the chromatic number.

It is worth noting that for certain parameter regimes- the number of gates, the number of qubits, the sizes of gates- the bound is nearly tight and polynomial time algorithms can meet the lower bound [46, 47, 48]. However, delving into the optimal solvability of commuting circuits is beyond the scope of this work.

4.3 Experiments

We generate commuting circuits with $m \in \{10, 20, 30\}$ qubits and $n \in \{10, 20, ..., 200\}$ multi-controlled Z gates. When generating a gate, we randomly choose at least three to at most $\lfloor m/2 \rfloor$ qubits the gate will act on. The generated circuit is then fed to Algorithm 4 as an input. For each m and n, 100 circuits are generated and optimized by the algorithm. The results are shown in Figure 1.

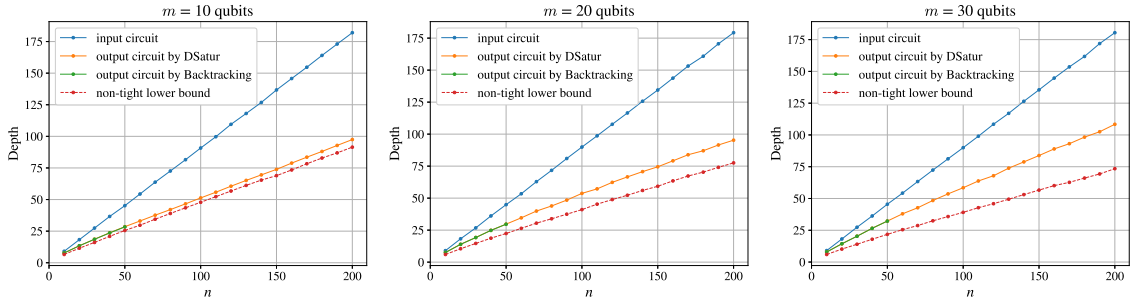

Figure 1: Optimization by coloring for randomly generated commuting circuits with $m \in \{10, 20, 30\}$ qubits and $n \in \{10, 20, ..., 200\}$ gates. Blue, orange, green, and red markers denote, respectively, the averaged depths of the input circuit, the output circuit produced by DSatur, the output circuit produced by Backtracking, and the non-tight lower bound by Eq. (2).

In Figure 1, each data point represents an average over 100 independent trials. The blue marker indicates the depth of the input circuits, the orange (green) marker represents the depth of the output circuit obtained using the DSatur (Backtracking) solver, and the red marker denotes the non-tight lower bound given by Eq. (2). Since the running time of the exact solver is exponential in the number of gates, the green markers are present up to $n = 50$. For all (n, m) values where green markers are present, the orange markers lie unnoticeably slightly above or exactly on the green ones. How much more the two solvers deviate for larger circuit sizes is open for further experiment. Each set of circuit parameters

(n, m, and gate sizes) may correspond to a distinct hardness regime for graph coloring in random intersection graphs [48]. Based on the observations, we hypothesize that gate ordering is easier for small numbers of qubits, as the polynomial time heuristic algorithm produced circuits that were nearly optimal. For larger numbers of qubits however, we cannot immediately draw the conclusion that the problems are harder since the lower bound is non-tight.

In addition, we conducted an experiment on the coloring algorithms within the Qiskit framework. Direct comparison is limited because Qiskit transpiler does not work as a gate ordering solver. Instead, we let Qiskit to decompose a commuting circuit, with and without the coloring algorithms applied, into Clifford+T circuit and then let it optimize to reduce the number of T gates. Details are given in Appendix C.

5 Applications

It is clear that the proposed optimization strategy can be utilized to reduce the depth of instantaneous quantum polynomial or commuting local Hamiltonian circuits since these circuits consist only of commuting gates. In addition, commuting circuits are commonly found in quantum algorithms that require superposed queries. Let f be a function and consider its superposed query,

$$\sum_x |x\rangle|0\rangle \mapsto \sum_x |x\rangle|f(x)\rangle \ .$$

Implementation of such query often involves commuting gates, as the qubits in the first and the second registers are mainly used for controls and targets, respectively. Two such applications are examined in this section including finite field multiplications and integer additions based on quantum Fourier transform.

5.1 Finite field multiplication

A quantum circuit for multiplication in $\mathbb{F}_{p^n}$ for prime p and $n \in \mathbb{Z}^+$ is an important component in various quantum algorithms. Researchers have developed various designs for finite field multiplications [5, 29, 49, 50, 51, 52, 53], and here we focus on the one proposed by Maslov et al., which involves the minimum number of qubits [5]. The result for $p = 2$ is summarized as follows:

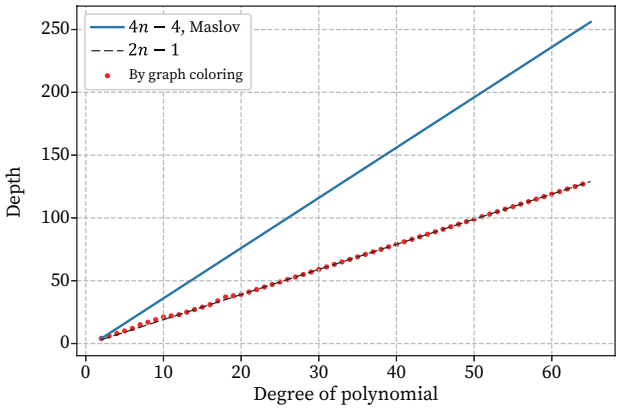

Figure 2: Depth optimization of multiplication in $\mathbb{F}_{2^n}$ by solving graph coloring. Red markers are obtained by using DSatur algorithm for graph coloring, black dashed line is the lower bound, and blue solid line is Maslov et al's original design.

Remark 10. *Using graph coloring, Toffoli-depth of the multiplication circuit is reduced to as small as* $2n-1$*, from* $4n-4$*.*

Maslov et al. originally proposed a circuit for binary field multiplication, but it can easily be extended to any $p > 2$. The binary field multiplication circuit takes as inputs three quantum registers holding $a \in \mathbb{F}_{2^n}$, $b \in \mathbb{F}_{2^n}$, and 0, respectively on n qubits each, and outputs $a, b, a \cdot b$. It requires n^2 Toffoli gates and $O(n^2)$ CNOT gates, with $4n-4$ Toffoli-depth. An important feature of the design is that no extra qubit is involved throughout the operation other than the initial $3n$ qubits.

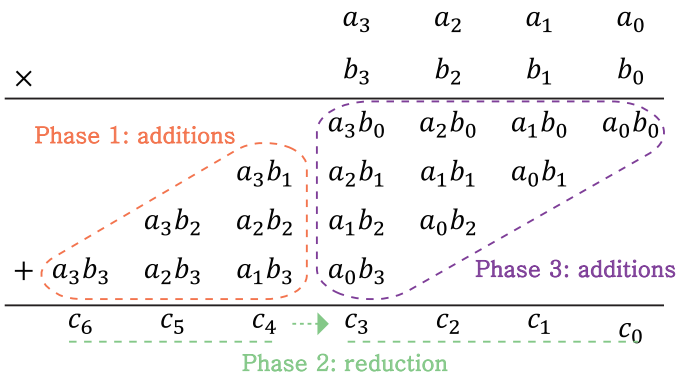

Figure 3: Illustration of the multiplication. The terms in each triangle are to be summed where the summing order does not affect the result.

The design approach is reviewed first, and then we point out where the graph coloring can be exploited. For our purpose, it suffices to review a simple example given by [5, Section 3]. Consider multiplying $a \cdot b$ in $\mathbb{F}_{2^4}$ with the irreducible reducing polynomial $P(x) = x^4 + x + 1$. Let $\boldsymbol{a}$ (and also $\boldsymbol{b}$) be a vector of coefficients of the polynomial; $\boldsymbol{a} = (a_3, a_2, a_1, a_0)$, $\boldsymbol{b} = (b_3, b_2, b_1, b_0)$. Define

$$\boldsymbol{d} = \begin{bmatrix} a_0b_0 \\ a_1b_0 + a_0b_1 \\ a_2b_0 + a_1b_1 + a_0b_2 \\ a_3b_0 + a_2b_1 + a_1b_2 + a_0b_3 \end{bmatrix}, \quad \boldsymbol{e} = \begin{bmatrix} a_3b_1 + a_2b_2 + a_1b_3 \\ a_3b_2 + a_2b_3 \\ a_3b_3 \end{bmatrix}, \quad Q = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix},$$

and then one can verify that $\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{d} + Q\boldsymbol{e}$ (see, [5] for details). Inspecting the equality, quantum multiplication can be divided into three phases. Phase 1 is computing $\boldsymbol{e}$ on the empty register, Phase 2 is multiplying (reducing) the result by Q, and Phase 3 is adding the result by $\boldsymbol{d}$. Writing down it as a unitary transformation,

$$|\boldsymbol{a}\rangle|\boldsymbol{b}\rangle|0\rangle \overset{\text{Phase 1}}{\longmapsto} |\boldsymbol{a}\rangle|\boldsymbol{b}\rangle|\boldsymbol{e}\rangle \overset{\text{Phase 2}}{\longmapsto} |\boldsymbol{a}\rangle|\boldsymbol{b}\rangle|Q\boldsymbol{e}\rangle \overset{\text{Phase 3}}{\longmapsto} |\boldsymbol{a}\rangle|\boldsymbol{b}\rangle|\boldsymbol{d} + Q\boldsymbol{e}\rangle \ .$$

Figure 3 illustrates the procedures. Focusing on the Toffol-depth, in $\mathbb{F}_{2^n}$, the authors estimated the depth to be $2n-3$, 0, $2n-1$ in Phase 1, Phase 2, and Phase 3, respectively, in total $4n-4$.

Now notice that all the additions occurring in Phase 1 are commuting. Similarly, all the additions in Phase 3 are also commuting. The graph coloring strategy is applied to each Phase 1 and Phase 3 in $\mathbb{F}_{2^n}$ for $n \leq 512$, adopting the solver DSatur which is a polynomial time heuristic algorithm for graph coloring [39]. Total Toffoli-depth is the sum of the coloring results in two phases. Figure 2 summarizes the result. (Plots for $n > 64$ are not shown in the figure, but all of them do not deviate at all from $2n-1$.) For $n \geq 6$, a

lower bound on the Toffoli-depth of $2n-1$ can be deduced from an elementary argument,[1] and DSatur has found circuits meeting the lower bound for most n tested.[2]

The above argument also holds in non-binary fields, $\mathbb{F}_{p^n}$ with prime $p > 2$. The difference is that the arithmetics of coefficients take place in the base field. Such arithmetics themselves may not consist entirely of commuting gates, but the addition operations are still commuting, allowing the optimization strategy to be applicable.

5.2 Depth-qubit tradeoffs for QFT-based addition

The graph coloring strategy can provide depth-qubit tradeoff options. As an example, we look for depth-qubit tradeoffs for QFT-based addition developed by Draper [6] and its variant [55]. Having the original design as one extreme end (qubit saving), the goal is to find circuits with various numbers of qubits and depths. The result is summarized in Figure 6.

Briefly reviewed, Draper's design for the addition works as follows. Consider adding two numbers $a, b \in \mathbb{Z}$, $|a\rangle|b\rangle \mapsto |a\rangle|b+a\rangle$. To this end, we first carry out the QFT on the second register, $\mathsf{QFT}|b\rangle = |\phi(b)\rangle$. Having qubits in $|a\rangle$ register as controls, conditional rotation operations are applied such that $|a\rangle|\phi(b)\rangle \mapsto |a\rangle|\phi(b+a)\rangle$. Finally, the inverse QFT transforms the second register back into the desired result, $\mathsf{QFT}^{-1}|\phi(b+a)\rangle = |b+a\rangle$. Figure 4 (a) describes an example.

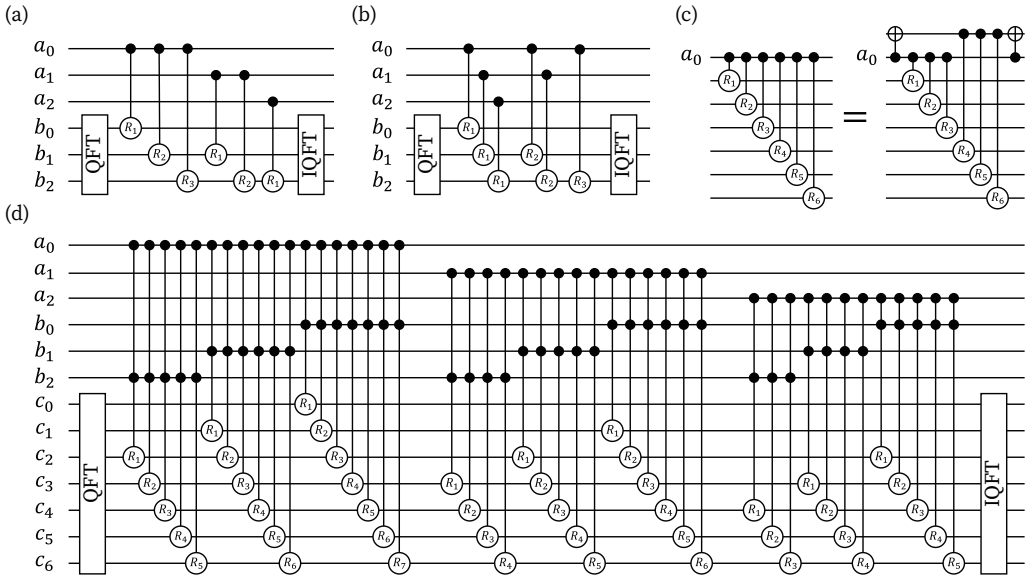

Figure 4: QFT-based addition $|a\rangle|b\rangle \mapsto |a\rangle|b+a\rangle$ is shown in (a) and its (trivially) optimized circuit is given in (b). Reducing the depth by extra work qubit is illustrated in (c). A circuit for $|a\rangle|b\rangle|c\rangle \mapsto |a\rangle|b\rangle|c+ab\rangle$ is given in (d).

Notice that excluding the two QFTs, the conditional rotation gates are all commuting. It is thus possible to optimize the circuit by solving the graph coloring, but the optimal depth (excluding two QFTs) is trivially inferred as $n = \max(\lceil \log_2 a \rceil, \lceil \log_2 b \rceil)$ by observa-

[1] It is the sum of the lengths of the bases of two triangles in Figure 3. For $n < 6$, the lower bound is larger than $2n-1$ since a right triangle of base length ≤ 4 (which appears for $n \leq 5$) cannot be fully parallelized.

[2] For some small n (< 20) the corresponding graphs are rather denser in edge density, and the adopted DSatur code [54] has failed to meet the bound.

tion. Instead of optimizing the circuit itself, one may consider further reducing the depth by introducing extra qubits. The basic idea is that if a specific qubit is referred to multiple times as controls or targets contributing a substantial amount to the overall depth, and then sharing its role (as control or target) with x more qubits reduce the relevant depth to around $1/(x+1)$. For example, introducing one initialized qubit in Figure 4 (c) reduces the depth by half (except the two CNOT gates at both ends) by *assisting* a_0. We call a circuit optimization by the above method parallelization by extra qubits.

An algorithm for the parallelization by extra qubits can be devised by using graph coloring. A heuristic algorithm is developed in which the central idea is noted below. (We do not elaborate on the algorithm as the algorithm itself is not the key finding.)

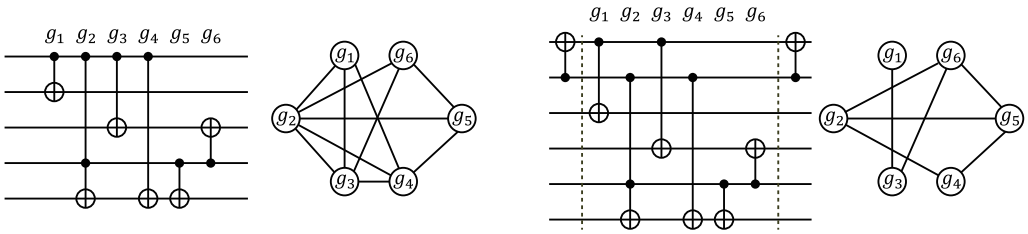

Figure 5: An example of the parallelization by extra qubits. Introducing one extra qubit gives rise to altering the graph such that it becomes sparser (in edge density).

Consider a commuting circuit and its corresponding graph $G = (V, E)$. Since gates are vertices, introducing extra qubits leaves V unchanged, but depending on which gates are lifted, E changes such that the number of connected edges decreases. Figure 5 shows how the graph can be altered by one extra qubit. The key idea of the algorithm is that as we cut out the edges from the graph, the chromatic number can only get smaller. For example, let $G(n,p)$ be a random graph, where n is the number of vertices and p is the probability that an edge exists between two vertices. Bollobás has shown that the chromatic number of a random graph $\chi(G(n,p))$ asymptotically reads [56],

$$\chi(G(n,p)) \sim \frac{n}{2\log_{1/(1-p)} n}(1+o(1)) \ .$$

In other words, the chromatic number of a random graph gets smaller as p approaches 0. Therefore, we use the extra qubits in a way that the altered graph becomes as sparse in edge density as possible.[3] By using the algorithm, we have parallelized the QFT-based additions $|a\rangle|b\rangle \mapsto |a\rangle|b+a\rangle$ and $|a\rangle|b\rangle|c\rangle \mapsto |a\rangle|b\rangle|c+ab\rangle$. For simplicity, we set $\lceil \log_2 a \rceil = \lceil \log_2 b \rceil = n$ and $c = 0$. The results are summarized in Figure 6. (Up to roughly 10,000 qubits are considered, far exceeding what current hardware can support at the time of writing. One should take it as a theoretical calculation.)

[3]Beware that not every commuting circuit corresponds to a random graph or a random intersection graph.

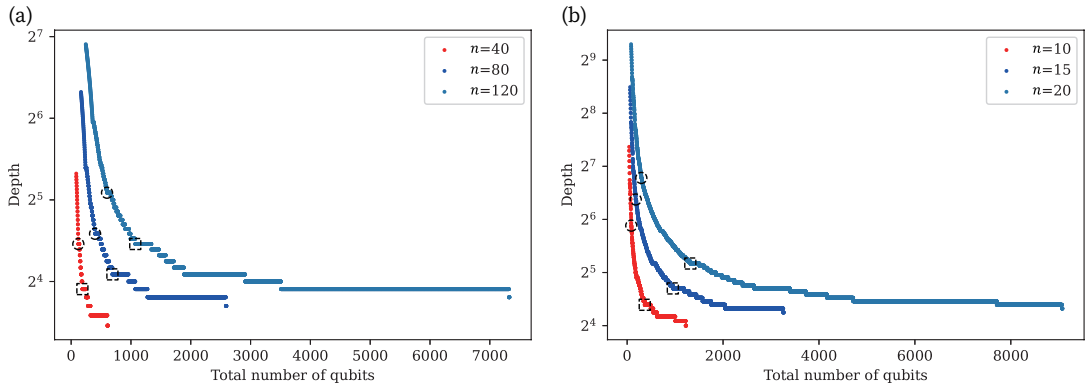

Figure 6: Depth-qubit tradeoffs for QFT-based addition circuits (a) $|a\rangle|b\rangle \mapsto |a\rangle|b+a\rangle$ and (b) $|a\rangle|b\rangle|c\rangle \mapsto |a\rangle|b\rangle|c+ab\rangle$, where a and b are n-bit numbers and c is set to zero. The circled and squared markers are the most efficient circuit designs with respect to the cost metrics in Eq. (3), respectively (see, Table 1).

Table 1: Cost ratio of C_{space} and C_{time} to the most cost-efficient circuit obtained by the algorithm, where C_{space} and C_{time} are the most space- and time-saving circuits for QFT-based additions. Two different metrics S and T in Eq. (3) are examined. The greater the value is than 1, the more inefficient it is. In Figure 6, in each tradeoffs curve, the circuits for $\min_C S(C)$ and $\min_C T(C)$ are highlighted by dashed circles and squares, respectively.

	Figure 6 (a)			Figure 6 (b)		
	$n=40$	$n=80$	$n=120$	$n=10$	$n=15$	$n=20$
$S(C_{\text{space}})/\min_C S(C)$	1.21	1.33	1.41	1.33	1.48	1.57
$S(C_{\text{time}})/\min_C S(C)$	2.53	3.50	5.02	3.85	4.16	5.57
$T(C_{\text{space}})/\min_C T(C)$	3.03	5.14	6.66	6.90	12.27	18.84
$T(C_{\text{time}})/\min_C T(C)$	1.73	2.19	2.77	1.93	1.82	2.12

The tradeoff designs allow one to choose more economical options when the total cost of running a circuit is measured in terms of the depth (time cost) and the number of qubits (space cost). For example consider two cost functions S and T,

$$S(C) = \mathsf{depth}(C) \cdot \mathsf{qubit}(C), \quad T(C) = (\mathsf{depth}(C))^2 \cdot \mathsf{qubit}(C) \ , \tag{3}$$

where C is a circuit, $\mathsf{depth}(C)$ is the depth of C, and $\mathsf{qubit}(C)$ is the number of qubits required to run C. The function S puts equal weights on time and space, whereas T values time more than space as it requires quadratically more space to reduce time as in the quantum search algorithm [57]. Based on the two metrics S and T, the cost of two extreme designs (the most space- and time-saving ones) for QFT-based additions is compared with the most cost-efficient circuit obtained by the algorithm. Table 1 summarizes the results. For example, in QFT-addition with $n = 120$, $S(C_{\text{space}})/\min_C S(C) = 1.41$ means the most space-saving circuit is 1.41 times more expensive than the optimized one found by the algorithm when considering the cost metric S. The circuit with the lowest cost is highlighted in Figure 6.

6 Conclusion and future directions

In this work, optimizing the depth of a commuting circuit is reduced to solving graph coloring problem, leading to a strategy for circuit optimization. It has been shown that the strategy can be applied to improve the known circuits.

One limitation of the reduction strategy is that it only works on commuting circuits. Two workarounds can be further investigated. One direction is to transform a noncommuting circuit into one such that a portion of commuting parts is large. This is closely related with the circuit synthesis where the behavior of an input circuit should be maintained in the output [8, 26]. The other way is to divide a noncommuting circuit into many commuting parts and then applying the method where available. For example, LU decomposition is employed in optimizing linear circuits [58, 28], where we see a potential application of the coloring method.

We want to point out that the strategy developed in this work is capable of finding the optimal depth of a commuting circuit corresponding to a perfect graph [14], since there exists a polynomial time algorithm for graph coloring of a perfect graph [59]. The next question is then whether it is possible to find such a circuit that has practical uses while its optimal depth is not yet known. We have not formally proved yet, but considering that the perfect graphs do not have odd ring of length ≥ 5, a circuit consisting of commuting CNOT gates seems to correspond to a perfect graph.[4] However, we were not able to take advantage of such a fact for a useful application.

An anonymous reviewer raised a question whether the reduction idea can be generalized to account for hardware restrictions such as qubit connectivity and gate set. If hardware restrictions are to be considered, an originally commuting circuit will naturally become noncommuting one. We have tried to generalize the reduction idea so that it works on such restrictions, but came to the conclusion that it is highly nontrivial. There is a possibility that vertex coloring of a certain acyclic directed graph may properly reflect hardware restrictions, but we were not able to think of a clear and efficient way.

An anonymous reviewer raised another question if the vertex coloring approach is advantageous over the edge coloring of a hypergraph. In our estimation, it seems possible to find similar reduction steps by using the edge coloring in an appropriate hypergraph. To compare the better solvability of the edge coloring and the vertex coloring is an interesting but nontrivial task, since the target graphs as well as the known solvers are different. For example, the reduced hypergraph for edge coloring would have the same number of vertices and edges as the number of qubits and gates, respectively, whereas the reduced graph for vertex coloring has the same number of vertices as the number of gates but its edges are not directly relevant to the number of qubits. Resolving the above issues may be worth independent work.

Acknowledgement

We are grateful to Raghav Govind Jha for introducing the partitioning of Pauli strings and the relevant papers.

[4] On the other hand, CZ-only circuits generally do not correspond to perfect graphs.

References

[1] Dan Shepherd and Michael J. Bremner. "Temporally unstructured quantum computation". Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **465**, 1413–1439 (2009).

[2] Michael J. Bremner, Ashley Montanaro, and Dan J. Shepherd. "Achieving quantum supremacy with sparse and noisy commuting quantum computations". Quantum **1**, 8 (2017).

[3] Sergey Bravyi and Mikhail Vyalyi. "Commutative version of the local Hamiltonian problem and common eigenspace problem". Quantum Info. Comput. **5**, 187–215 (2005).

[4] A.Yu. Kitaev. "Fault-tolerant quantum computation by anyons". Annals of Physics **303**, 2–30 (2003).

[5] Dmitri Maslov, Jimson Mathew, Donny Cheung, and Dhiraj K. Pradhan. "An $O(m^2)$-depth quantum algorithm for the elliptic curve discrete logarithm problem over $GF(2^m)$". Quantum Info. Comput. **9**, 610–621 (2009).

[6] Thomas G. Draper. "Addition on a quantum computer" (2000). arXiv:quant-ph/0008033.

[7] Mehdi Saeedi and Igor L. Markov. "Synthesis and optimization of reversible circuits—a survey". ACM Comput. Surv.**45** (2013).

[8] Yunseong Nam, Neil J Ross, Yuan Su, Andrew M Childs, and Dmitri Maslov. "Automated optimization of large quantum circuits with continuous parameters". npj Quantum Information **4**, 23 (2018).

[9] J-H Bae, Paul M Alsing, Doyeol Ahn, and Warner A Miller. "Quantum circuit optimization using quantum Karnaugh map". Scientific reports **10**, 15651 (2020).

[10] Zhenyu Huang and Siwei Sun. "Synthesizing quantum circuits of AES with lower T-depth and less qubits". In Shweta Agrawal and Dongdai Lin, editors, Advances in Cryptology – ASIACRYPT 2022. Pages 614–644. Cham (2022). Springer Nature Switzerland.

[11] Matthew Amy, Dmitri Maslov, and Michele Mosca. "Polynomial-time T-depth optimization of Clifford+T circuits via matroid partitioning". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **33**, 1476–1489 (2014).

[12] Michael A. Nielsen and Isaac L. Chuang. "Quantum computation and quantum information: 10th anniversary edition". Cambridge University Press. (2010).

[13] Sergey Bravyi, Ruslan Shaydulin, Shaohan Hu, and Dmitri Maslov. "Clifford circuit optimization with templates and symbolic Pauli gates". Quantum **5**, 580 (2021).

[14] John Adrian Bondy and Uppaluri Siva Ramachandra Murty. "Graph theory". Springer Publishing Company, Incorporated. (2008).

[15] Valentine Kabanets and Jin-Yi Cai. "Circuit minimization problem". In Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing. Page 73–79. STOC '00New York, NY, USA (2000). Association for Computing Machinery.

[16] Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael Saks. "Minimizing disjunctive normal form formulas and AC^0 circuits given a truth table". SIAM Journal on Computing **38**, 63–84 (2008).

[17] S Hirahara, I Oliveira, and R Santhanam. "NP-hardness of minimum circuit size problem for OR-AND-MOD circuits". In 33rd Computational Complexity Conference (CCC 2018). Volume 102 of Leibniz International Proceedings in Informatics, pages 5:1–5:31. Schloss Dagstuhl (2018).

[18] Rahul Ilango, Bruno Loff, and Igor C. Oliveira. "NP-hardness of circuit minimization for multi-output functions". In Shubhangi Saraf, editor, 35th Computational Complexity Conference (CCC 2020). Volume 169 of Leibniz International Proceedings in Informatics (LIPIcs), pages 22:1–22:36. Dagstuhl, Germany (2020). Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[19] Shuichi Hirahara. "NP-hardness of learning programs and partial MCSP". In 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS). Pages 968–979. (2022).

[20] Kunihiro Wasa, Shin Nishio, Koki Suetsugu, Michael Hanks, Ashley Stephens, Yu Yokoi, and Kae Nemoto. "Hardness of braided quantum circuit optimization in the surface code". IEEE Transactions on Quantum Engineering **4**, 1–7 (2023).

[21] John van de Wetering and Matt Amy. "Optimising quantum circuits is generally hard" (2024). arXiv:2310.05958.

[22] John Preskill. "Quantum Computing in the NISQ era and beyond". Quantum **2**, 79 (2018).

[23] Matthew Amy, Dmitri Maslov, and Michele Mosca. "Polynomial-time T-depth optimization of Clifford+T circuits via matroid partitioning". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **33**, 1476–1489 (2014).

[24] D. Maslov, C. Young, D.M. Miller, and G.W. Dueck. "Quantum circuit simplification using templates". In Design, Automation and Test in Europe. Pages 1208–1213 Vol. 2. (2005).

[25] Mathias Soeken, Robert Wille, Gerhard W. Dueck, and Rolf Drechsler. "Window optimization of reversible and quantum circuits". In 13th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems. Pages 341–345. (2010).

[26] Robert Wille, Lucas Berent, Tobias Forster, Jagatheesan Kunasaikaran, Kevin Mato, Tom Peham, Nils Quetschlich, Damian Rovara, Aaron Sander, Ludwig Schmid, Daniel Schönberger, Yannick Stade, and Lukas Burgholzer. "The MQT handbook : A summary of design automation tools and software for quantum computing". In 2024 IEEE International Conference on Quantum Software (QSW). Pages 1–8. (2024).

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

[28] Dmitri Maslov and Ben Zindorf. "Depth optimization of CZ, CNOT, and Clifford circuits". IEEE Transactions on Quantum Engineering **3**, 1–8 (2022).

[29] Brittanney Amento, Martin Rötteler, and Rainer Steinwandt. "Quantum binary field inversion: improved circuit depth via choice of basis representation". Quantum Info. Comput. **13**, 116–134 (2013).

[30] Rebekah Herrman, James Ostrowski, Travis S. Humble, and George Siopsis. "Lower bounds on circuit depth of the quantum approximate optimization algorithm". Quantum Information Processing **20**, 59 (2021).

[31] Claude Berge. "Hypergraphs: combinatorics of finite sets". Volume 45. Elsevier. (1984).

[32] Andrew Jena, Scott Genin, and Michele Mosca. "Pauli partitioning with respect to gate sets" (2019). arXiv:1907.07859.

[33] Vladyslav Verteletskyi, Tzu-Ching Yen, and Artur F. Izmaylov. "Measurement optimization in the variational quantum eigensolver using a minimum clique cover". The Journal of Chemical Physics **152**, 124114 (2020).

[34] Ikko Hamamura and Takashi Imamichi. "Efficient evaluation of quantum observables using entangled measurements". npj Quantum Information **6**, 56 (2020).
[35] Pranav Gokhale, Olivia Angiuli, Yongshan Ding, Kaiwen Gui, Teague Tomesh, Martin Suchara, Margaret Martonosi, and Frederic T. Chong. "$O(N^3)$ measurement cost for variational quantum eigensolver on molecular hamiltonians". IEEE Transactions on Quantum Engineering **1**, 1–24 (2020).
[36] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. "A variational eigenvalue solver on a photonic quantum processor". Nature Communications **5**, 4213 (2014).
[37] Muhammad Asaduzzaman, Raghav G. Jha, and Bharath Sambasivam. "Sachdev-Ye-Kitaev model on a noisy quantum computer". Phys. Rev. D **109**, 105002 (2024).
[38] D. J. A. Welsh and M. B. Powell. "An upper bound for the chromatic number of a graph and its application to timetabling problems". The Computer Journal **10**, 85–86 (1967).
[39] Daniel Brélaz. "New methods to color the vertices of a graph". Commun. ACM **22**, 251–256 (1979).
[40] Frank Thomson Leighton. "A graph coloring algorithm for large scheduling problems". Journal of research of the national bureau of standards **84**, 489–506 (1979).
[41] Hussein Al-Omari and Khair Eddin Sabri. "New graph coloring algorithms". American Journal of Mathematics and Statistics **2**, 739–741 (2006).
[42] Murat Aslan and Nurdan Akhan Baykan. "A performance comparison of graph coloring algorithms". International Journal of Intelligent Systems and Applications in Engineering **4**, 1–7 (2016).
[43] Isabelle Devarenne, Hakim Mabed, and Alexandre Caminada. "Intelligent neighborhood exploration in local search heuristics". In 2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06). Pages 144–150. (2006).
[44] J. Randall Brown. "Chromatic scheduling and the chromatic number problem". Management Science **19**, 456–463 (1972).
[45] Michał Karoński, Edward R Scheinerman, and Karen B Singer-Cohen. "On random intersection graphs: The subgraph problem". Combinatorics, Probability and Computing **8**, 131–159 (1999).
[46] Michael Behrisch, Anusch Taraz, and Michael Ueckerdt. "Coloring random intersection graphs and complex Networks". SIAM Journal on Discrete Mathematics **23**, 288–299 (2009).
[47] Valentas Kurauskas and Katarzyna Rybarczyk. "On the chromatic index of random uniform hypergraphs". SIAM Journal on Discrete Mathematics **29**, 541–558 (2015).
[48] Katarzyna Rybarczyk. "The chromatic number of random intersection graphs". Discussiones Mathematicae Graph Theory **37**, 465–476 (2017).
[49] Brittanney Amento, Martin Rötteler, and Rainer Steinwandt. "Efficient quantum circuits for binary elliptic curve arithmetic: reducing T-gate complexity". Quantum Info. Comput. **13**, 631–644 (2013).
[50] Martin Rötteler and Rainer Steinwandt. "A quantum circuit to find discrete logarithms on ordinary binary elliptic curves in depth $O(\log^2 n)$". Quantum Info. Comput. **14**, 888–900 (2014).
[51] Shane Kepley and Rainer Steinwandt. "Quantum circuits for $\mathbb{F}_{2^n}$-multiplication with subquadratic gate count". Quantum Information Processing **14**, 2373–2386 (2015).
[52] Iggy Van Hoof. "Space-efficient quantum multiplication of polynomials for binary finite fields with sub-quadratic Toffoli gate count". Quantum Info. Comput. **20**, 721–735 (2020).

[53] Qing-bin Luo, Guo-wu Yang, Xiao-yu Li, and Qiang Li. "Quantum reversible circuits for GF(2^8) multiplicative inverse". EPJ Quantum Technology **9**, 24 (2022).
[54] Mashiyat Zaman, Kotaro Tanahashi, and Shu Tanaka. "PyQUBO: python library for mapping combinatorial optimization problems to QUBO form". IEEE Transactions on Computers **71**, 838–850 (2022).
[55] Lidia Ruiz-Perez and Juan Carlos Garcia-Escartin. "Quantum arithmetic with the quantum Fourier transform". Quantum Information Processing **16**, 152 (2017).
[56] B. Bollobás. "The chromatic number of random graphs". Combinatorica **8**, 49–55 (1988).
[57] Christof Zalka. "Grover's quantum searching algorithm is optimal". Phys. Rev. A **60**, 2746–2751 (1999).
[58] Jiaqing Jiang, Xiaoming Sun, Shang-Hua Teng, Bujiao Wu, Kewen Wu, and Jialin Zhang. "Optimal space-depth trade-off of CNOT circuits in quantum logic synthesis". In Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms. Page 213–229. SODA '20USA (2020). Society for Industrial and Applied Mathematics.
[59] M. Grötschel, L. Lovász, and A. Schrijver. "Polynomial algorithms for perfect graphs". In C. Berge and V. Chvátal, editors, Topics on Perfect Graphs. Volume 88 of North-Holland Mathematics Studies, pages 325–356. North-Holland (1984).

A Reduction from graph coloring to gate ordering

The graph coloring problem can be reduced to the gate ordering. An algorithm that takes as input a graph and outputs a circuit is required for the reduction, but since parallelizability of gates is dependent on the underlying hardware, it is difficult to describe the algorithm in an abstract manner. Therefore, we fix the condition for two gates to be parallelizable by taking Assumption 9 from the main text, but a similar algorithm can be developed for different conditions in general.

Let $CZ_{\mathcal{C}}$ denote a multi-controlled phase shift gate (by an arbitrary angle θ), where $\mathcal{C}$ is the set of referenced qubits. For instance, $CZ_{\{q_1,q_2\}}$ gate endows the quantum state with phase $e^{i\theta}$ when $q_1 = q_2 = 1$.

Algorithm 5 ConstructCircuit

Input: $G = (V, E)$
Output: $\boldsymbol{v} = (v_1, \ldots, v_{|V|})$

$t \leftarrow 1$
for $i = 1$ **to** $|V|$ **do**
  $\mathcal{C}_i \leftarrow \{q_1, \ldots, q_t\}$, $\mathcal{T} \leftarrow \{\}$
  for $j = 1$ **to** $i - 1$ **do**
    if $e_{ij} \notin E$ **then**
      $\mathcal{C}_i \leftarrow \mathcal{C}_i \backslash \mathcal{C}_j$
  for $j = 1$ **to** $i - 1$ **do**
    if $e_{ij} \in E$ **and** $\mathcal{C}_i \cap \mathcal{C}_j = \{\}$ **then**
      for $q \in \mathcal{T}$ **do**
        $p \leftarrow$ True
        for $k = 1$ **to** $j - 1$ **do**
          if $e_{kj} \notin E$ **and** $q \in \mathcal{C}_k$ **then**
            $p \leftarrow$ False
            break
        if $p =$ True **then**
          $\mathcal{C}_i \leftarrow \mathcal{C}_i \cup \{q\}$, $\mathcal{C}_j \leftarrow \mathcal{C}_j \cup \{q\}$
          continue
      if $\mathcal{C}_j \cap \mathcal{T} = \{\}$ **then**
        $t \leftarrow t + 1$
        $\mathcal{C}_i \leftarrow \mathcal{C}_i \cup \{q_t\}$, $\mathcal{C}_j \leftarrow \mathcal{C}_j \cup \{q_t\}$
        $\mathcal{T} \leftarrow \mathcal{T} \cup \{t\}$
  if $\mathcal{C}_i = \{\}$ **then**
    $t \leftarrow t + 1$
    $\mathcal{C}_i \leftarrow \mathcal{C}_i \cup \{q_t\}$
for $i = 1$ **to** $|V|$ **do**
  $v_i \leftarrow CZ_{\mathcal{C}_i}$
return $\boldsymbol{v} = (v_1, \ldots, v_{|V|})$

Algorithm 5 constructs a circuit such that any two connected vertices (gates) share at least one qubit as a common reference. The resulting circuit consists of $|V|$ commuting gates and no more than $|V|^2/2$ qubits. (We have observed that the number of qubits tends to get smaller as the edge density approaches either zero or 1.) Figure 7 illustrates how the circuit is composed by Algorithm 5. Note that for given graph G, we have $G = \mathsf{ConstructGraph}(\mathsf{ConstructCircuit}(G))$, where ConstructGraph is Algorithm 1.

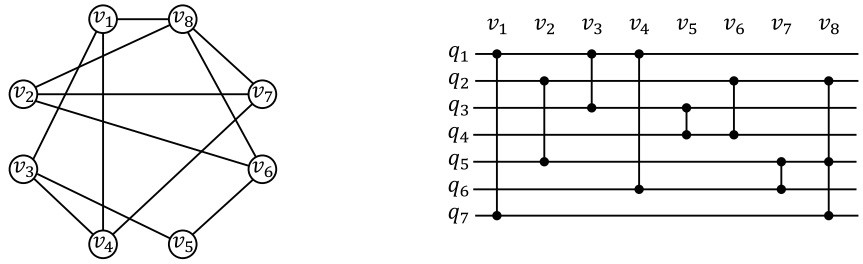

Figure 7: An input graph (left) and the output circuit (right) of Algorithm 5. Each gate in the circuit is a phase shift gate.

Given a graph G, we construct a commuting circuit $\boldsymbol{v}_G$ corresponding to G by Algorithm 5. Let $\mathcal{V}$ be the multiset of gates in $\boldsymbol{v}_G$, let $A_{\mathcal{V}}$ be the set of all permutations of gates in $\mathcal{V}$, and let $B_{\mathcal{V}}$ be the set of all colored graphs of G. Since $G = \mathsf{ConstructGraph}(\mathsf{ConstructCircuit}(G))$, Claim 7 still holds and we have two functions f and g as stated in the claim. Then we have the following proposition:

Proposition 11. *If* $m_{\mathrm{depth}}(\boldsymbol{a}) = \min_{\boldsymbol{a}' \in A_{\mathcal{V}}} m_{\mathrm{depth}}(\boldsymbol{a}')$ *holds, and then it is also true that* $m_{\mathrm{color}}(g(\boldsymbol{a})) = \min_{b' \in B_{\mathcal{V}}} (m_{\mathrm{color}}(b'))$.

Proof. We use the proof by contradiction. Let $\boldsymbol{a} \in A_{\mathcal{V}}$ be the circuit such that

$$m_{\mathrm{depth}}(\boldsymbol{a}) = \min_{\boldsymbol{a}' \in A_{\mathcal{V}}} (m_{\mathrm{depth}}(\boldsymbol{a}')) \ . \tag{4}$$

Now suppose that $m_{\mathrm{color}}(g(\boldsymbol{a})) > \min_{b' \in B_{\mathcal{V}}} (m_{\mathrm{color}}(b'))$, then there exists $b \in B_{\mathcal{V}}$ such that $m_{\mathrm{color}}(b) < m_{\mathrm{color}}(g(\boldsymbol{a}))$, $b \neq g(\boldsymbol{a})$.

Due to the properties of f and g, we have $m_{\mathrm{depth}}(f(b)) \leq m_{\mathrm{color}}(b)$ and $m_{\mathrm{color}}(g(\boldsymbol{a})) = m_{\mathrm{depth}}(\boldsymbol{a})$. In other words, there exists $g(\boldsymbol{a}) \in B_{\mathcal{V}}$ satisfying

$$m_{\mathrm{depth}}(f(b)) \leq m_{\mathrm{color}}(b) < m_{\mathrm{color}}(g(\boldsymbol{a})) = m_{\mathrm{depth}}(\boldsymbol{a}) \ ,$$

which contradicts to Eq. (4). □

B On remark 6

Case (a) in Remark 6 can only occurs when the coloring algorithm produces an approximate solution, not the true optimal one. Figure 8 (a) illustrates a commuting circuit and its corresponding graph. Having Assumption 9, the depth of the circuit is four. In Figure 8 (b), the graph is colored with the minimum number of colors, yielding the optimal solution. Consequently, the number of colors coincides with the depth of the resulting circuit. By contrast, Figure 8 (c) shows an approximate solution for the coloring. However, the depth of the resulting circuit is three since v_2 and v_4 happen to be simultaneously executable.

C Coloring within Qiskit framework

Since the proposed method and Qiskit transpiler's optimization are not directly comparable, we let the Qiskit compiler optimize commuting circuits, with and without the coloring method applied beforehand, into Clifford+T circuits. By measuring the differences in two cases (with and without coloring applied), we assess how effectively the coloring enhances the compiler.

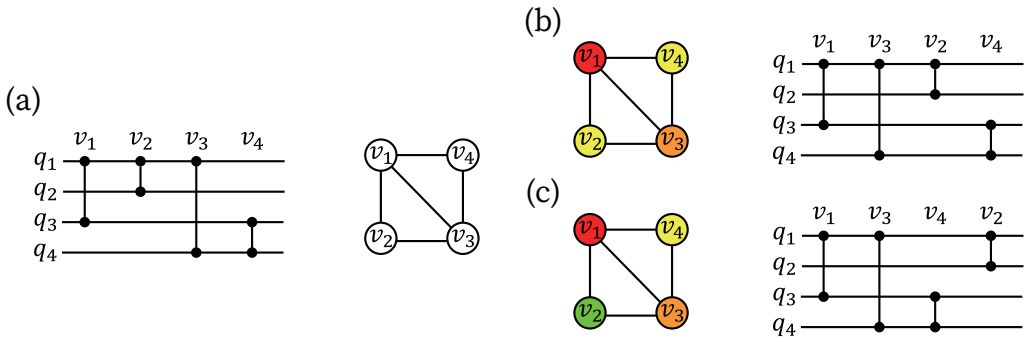

Figure 8: A commuting circuit and corresponding graph are shown in (a), and its colored graphs and resulting circuits are shown in (b) and (c).

We randomly generate commuting circuits with $m = 10$ qubits and $n \in \{10, 20, ..., 100\}$ CCZ gates. Unlike the experiments in the main text, we fixed the size of the gates because when decomposing into a Clifford+T circuit, imposing the same degree of parallelization on gates of different sizes (e.g., CCZ and CCCZ) is not well aligned with the goal of the experimental setup. The key Qiskit transpiler settings are as follows: `OptimizationMetric` is set to `COUNT_T` and `optimization_level` is set to 3. The results are shown in Figure 9.

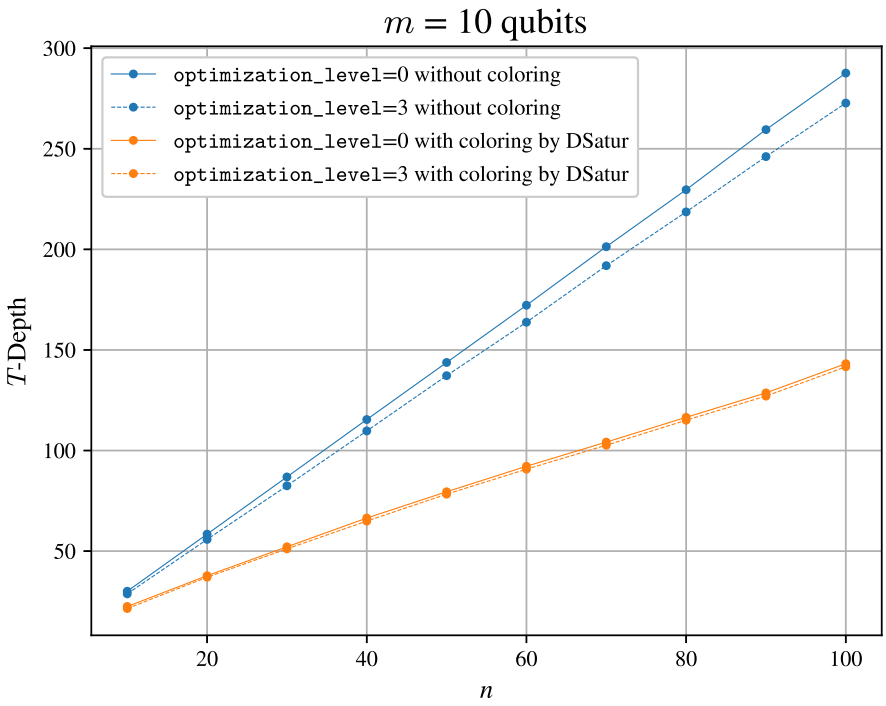

Figure 9: Optimization by Qiskit transpiler with and without the coloring method applied for randomly generated commuting circuits with $m = 10$ qubits and $n \in \{10, 20, ..., 100\}$ gates. Markers with solid and dashed lines are output by Qiskit with `optimization_level` $= 0$ and `optimization_level` $= 3$, respectively.