Clifford Circuit Optimization with Templates and Symbolic Pauli Gates

Sergey Bravyi[1], Ruslan Shaydulin[2], Shaohan Hu[3], and Dmitri Maslov[1]

[1]IBM Quantum, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598
[2]Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL 60439
[3]JPMorgan Chase & Co., New York, NY 10017

The Clifford group is a finite subgroup of the unitary group generated by the Hadamard, the CNOT, and the Phase gates. This group plays a prominent role in quantum error correction, randomized benchmarking protocols, and the study of entanglement. Here we consider the problem of finding a short quantum circuit implementing a given Clifford group element. Our methods aim to minimize the entangling gate count assuming all-to-all qubit connectivity. First, we consider circuit optimization based on template matching and design Clifford-specific templates that leverage the ability to factor out Pauli and SWAP gates. Second, we introduce a symbolic peephole optimization method. It works by projecting the full circuit onto a small subset of qubits and optimally recompiling the projected subcircuit via dynamic programming. CNOT gates coupling the chosen subset of qubits with the remaining qubits are expressed using symbolic Pauli gates. Software implementation of these methods finds circuits that are only 0.2% away from optimal for 6 qubits and reduces the two-qubit gate count in circuits with up to 64 qubits by 64.7% on average, compared to the Aaronson–Gottesman canonical form [3].

1 Introduction

One of the central challenges in quantum computation is the problem of generating a short schedule of physically implementable quantum gates realizing a given unitary operation, otherwise known as the quantum circuit synthesis/optimization problem. In this paper, we focus on a restricted class of quantum circuits belonging to the Clifford group, which is a subgroup of the group of all unitary transformations. Clifford group elements play a crucial role in quantum error correction [25], quantum state distillation [6, 20], randomized benchmarking [21, 23], study of entanglement [5, 25], and, more recently, shadow tomography [2, 16], to name some application areas. Clifford group elements are important and frequently encountered subsets of physical-level and fault-tolerant quantum circuits; sometimes, an entire quantum algorithm can be a Clifford circuit (e.g., Bernstein–Vazirani [25] and its generalizations [9]).

A special property of the Clifford group that plays the central role in many applications is being a unitary 2-design [11, 12]. It guarantees that a random uniformly distributed element of the Clifford group has exactly the same second order moments as the Haar random unitary operator. Thus random Clifford operators can serve as a substitute for Haar random unitaries in any application that depends only on the second order moments. However, in contrast to Haar random unitaries, any Clifford operators admit an efficient implementation by a quantum circuit. For example, randomized benchmarking [21, 23] provides a scalable fidelity metric for multi-qubit operations which is insensitive to the state preparation and measurement errors. Randomized benchmarking works by measuring the decay rate of a signal generated by a sequence of random Clifford operators of varying length. The 2-design property ensures that the effective noise model obtained after averaging over the Clifford group is the depolarizing channel with a single unknown noise parameter. As another example, classical shadows [16] provide a succinct classical description of a multi-qubit quantum state that can be efficiently measured in an experiment without performing the full state tomography. At the same time, a classical shadow determines many physically relevant properties of a state such as expected values of observables. A classical shadow of a quantum state ρ is obtained by repeatedly preparing a state $U\rho U^\dagger$ with a random Clifford operator U and measuring each qubit in the computational basis. The ability to realize a random element of the Clifford group by a short quantum circuit plays the central role in the above examples.

Clifford circuits also serve as a basis change transformation in quantum simulation algorithms. For example, simultaneous diagonalization of mutually commuting Pauli operators by a Clifford basis change can reduce the circuit depth for simulating quantum chemistry Hamiltonians [30]. Another example is tapering off qubits for quantum simulations by identifying Pauli-type symmetries of quantum chemistry Hamiltonians [8, 28]. Such symmetry operators can be mapped to single-qubit Pauli Z by applying a suitable Clifford circuit after which the respective qubits can be removed from the simulation.

Earlier studies of the synthesis of n-qubit Clifford circuits resulted in the construction of asymptotically optimal (i.e., optimal up to a constant factor) implementations in the number of gates used. Specifically, the canonical form introduced by Aaronson and Gottesman [3] accomplishes this using $\Theta\left(n^2/\log(n)\right)$ gates [26]. In contrast, in this paper we focus on the practical aspects of Clifford circuit optimization—our goal is to implement a given Clifford unitary by a circuit with the smallest possible number of entangling gates. We focus on the minimization of the CNOT gate count, drawing motivation from physical layer realizations where entangling gates come at a higher cost than the single-qubit gates, and ignore the connectivity constraints. While in the worst-case scenario ignoring connectivity may lead to an $O(n)$ blowup in the CNOT gate count or depth (consider the cost of implementation of the maximal-distance $\textsc{cnot}(x_1; x_n)$ gate in a linear chain with n qubits), known difference between the upper bound on the circuit depth between all-to-all and Linear Nearest Neighbor (LNN) architectures remains small. Indeed, for all-to-all architecture the best known upper bound on the two-qubit gate depth is $\frac{10}{3}n + O(\log(n))$ (obtained by combining Lemma 8 in [7] with Corollary III.2.2 in [15], and noting that CZ gate layer can be implemented in depth $(n{-}1)$ or n depending on whether n is even or odd), and the best-known lower bound is $\Omega\left(\frac{n}{\log(n)}\right)$ (obtained by a slight modification of the counting argument employed in [26]). In the LNN, upper bound is $9n$ [7], and lower bound is $2n{+}1$ [22]. The above suggests that executing a (random) Clifford circuit in restricted architectures (LNN may often be embedded in other architectures) comes with a relatively small overhead. We also note that our methods and algorithms can be straightforwardly modified to respect a restricted connectivity and target depth minimization rather than gate count minimization.

Current approaches to the synthesis of exactly optimal Clifford circuits are prohibitively expensive even for small parameters: the largest number of qubits for which optimal Clifford circuits are known is six [10]. Using these exhaustive tools leaves little hope of scaling optimal implementations beyond six qubits. Thus, efficient heuristics are desirable for practical applicability. Here we focus on the synthesis and optimization of Clifford circuits that cannot be obtained optimally, namely, circuits with $n > 6$ qubits.

Here we develop heuristic approaches for the synthesis and optimization of Clifford circuits. Our algorithms and their implementation bridge the gap between nonscalable methods for the synthesis of exactly optimal Clifford circuits and the suboptimal (albeit asymptotically optimal) synthesis methods. Our circuit synthesizer is based on the reduction of the tableau matrix representing Clifford unitary to the identity, while applying gates on both the input and output sides. Our optimization approach is based on the extension and modification of two circuit optimization techniques: template matching [24] and peephole optimization [27]. To generate an optimized circuit for a specific Clifford unitary, we first compile it using the tableau representation and then apply the optimization techniques to the compiled circuit. We note that the optimization techniques can be applied independently of the synthesizer considered in this paper.

The first optimization technique we develop is a Clifford-specific extension of the template matching method [24]. We discuss previous results on template matching in depth in Subsection 2.1. We introduce a three-stage approach that leverages the observation that in Clifford circuits Pauli gates can always be "pushed" to the end of the circuit without changing the non-Pauli Clifford gates (i.e., Hadamard, controlled-

NOT, and Phase gates) and that all SWAP gates can be factored out of any quantum circuit by qubit relabeling. We thus partition the circuit into "compute," "SWAP," and "Pauli" stages by "pushing" Pauli and SWAP gates to the end of the circuit. Next we optimize the "compute" stage using templates. Then we optimize the "SWAP" stage by exploiting the fact that a SWAP gate can be implemented at the effective cost of one entangling gate if it can be merged with a CNOT or a CZ gate.

The second technique we develop is symbolic peephole optimization. It is inspired by the peephole optimization method first introduced in the context of reversible computations [27]. At each step, the symbolic peephole algorithm considers subcircuits spanning a small set of qubits (2 and 3 in this paper) by introducing symbolic Pauli gates (SPGs) to replace the two-qubit gates that entangle qubits in the chosen set with a qubit outside of it. The resulting Clifford+SPG subcircuit is optimized via dynamic programming using a library of optimal circuits.

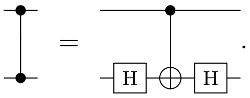
We numerically evaluate the proposed methods using two sets of benchmarks. The first benchmark is based on the database of optimal Clifford circuits constructed in [10]. We consider a selection of 1,003 randomly sampled 6-qubit Clifford unitaries, conditional on the optimal CNOT gate implementation cost being higher than 4 (otherwise, it is easy to implement such a unitary optimally). The set of tools developed in this work is able to recover an optimal (in terms of the CNOT count) implementation for 97.9% of the circuits, while producing circuits no more than one CNOT away from the optimal count in the worst case. Second, to evaluate the performance on "large" circuits, we consider a toy model of Hamiltonian evolution with a graph state Hamiltonian, defined as follows. For a given graph with n nodes, the Hamiltonian evolution performs the transformation $(\text{CZ} \cdot \text{H})^t$, where CZ gates apply to graph edges, H gates apply to graph nodes (individual qubits), and t is the evolution time. At integer times, the evolution by such a Hamiltonian is described by a Clifford unitary. Implementing it as a circuit $\text{CZ} \cdot \text{H}$ repeated t times turns out to be less efficient than implementing it by using the techniques reported here. The methods we developed are evaluated on a collection of 2,264 circuits and shown to reduce the average CNOT gate count by 64.7% compared with the methods proposed by Aaronson and Gottesman in [3]. We make the full benchmark and the raw results available online [1].

The rest of the paper is organized as follows. We begin by briefly revisiting relevant concepts and defining the notations (Section 2). We next discuss previous results that our work is based on (Subsection 2.1, Subsection 2.2). Following this discussion, we describe the proposed methods (Section 3), report numerical results, and evaluate the performance (Section 4). We conclude with a short summary (Section 5).

2 Background

We assume basic familiarity with quantum computing concepts, stabilizer formalism, and Clifford circuits. Below we briefly introduce relevant concepts and notations. For detailed discussion, the reader is referred to [25] and [3].

Clifford circuits (also known as stabilizer circuits) consist of Hadamard (H), Phase (S, also known as P gate), and controlled-NOT (CNOT) gates, as well as Pauli X, Y, and Z gates. We use I to denote the identity gate/matrix. We also utilize the controlled-Z (CZ) gate, which can be constructed as a circuit with Hadamard and CNOT gates as follows,



(1)

Clifford circuits acting on n qubits generate a finite group $\mathcal{C}_n$, known as the Clifford group. An important property of Clifford circuits is that Clifford gates H, S, and CNOT map tensor product of Pauli matrices into tensor products of Pauli matrices. This property can be employed to "push" Pauli gates through the Clifford gates H, S, and CNOT as follows:

$$\mathrm{HX} = \mathrm{ZH}, \quad \mathrm{HY} = -\mathrm{YH}, \quad \mathrm{HZ} = \mathrm{XH}, \quad \mathrm{SX} = \mathrm{YS}, \tag{2}$$
$$\mathrm{SY} = -\mathrm{XS}, \quad \mathrm{SZ} = \mathrm{ZS}, \tag{3}$$
$$\mathrm{CNOT}_{1,2}\mathrm{X}_1 = \mathrm{X}_1\mathrm{X}_2\mathrm{CNOT}_{1,2},\ \mathrm{CNOT}_{1,2}\mathrm{X}_2 = \mathrm{X}_2\mathrm{CNOT}_{1,2}, \tag{4}$$
$$\mathrm{CNOT}_{1,2}\mathrm{Z}_2 = \mathrm{Z}_1\mathrm{Z}_2\mathrm{CNOT}_{1,2},\ \mathrm{CNOT}_{1,2}\mathrm{Z}_1 = \mathrm{Z}_1\mathrm{CNOT}_{1,2}. \tag{5}$$

Our approach combines two building blocks: Clifford-specific extension of template matching and symbolic peephole optimization. Below we briefly review these techniques. While the developed methods reduce both single- and two-qubit gate count, in this paper we focus on the optimization of the number of two-qubit gates it takes to implement a Clifford group element. The reason for our focus is that the leading quantum information processing technologies, trapped ions [13] and superconducting circuits [17], both feature two-qubit gates that take longer time and have higher error rates compared with those of single-qubit gates.

2.1 Template Matching

A size m template [24] is a sequence of m gates that implements the identity function:

$$T = G_0 G_1 \dots G_{m-1} = I.$$

The templates can be used to optimize a target circuit as follows. First, a subcircuit $G_i G_{i+1 \pmod m} \dots G_{i+p-1 \pmod m}$ of the template is matched with a subcircuit in the given circuit. If the gates in the target circuit can be moved together, this sequence of gates can be replaced with the inverse of the other $m{-}p$ gates in the template. The larger the length p of the matched sequence is, the more beneficial it is to perform the replacement, and for any $p > \frac{m}{2}$ the gate count is reduced. The exact criteria for the application of the template depends on the choice of the objective optimization criteria (e.g., depth, total gate count, 2-qubit gate count). More formally, for parameter p, $\frac{m}{2} \leq p \leq m$, the template T can be applied in two directions as follows,

$$\begin{aligned} \textbf{Forward:}\ & G_i G_{i+1 \pmod m} \dots G_{i+p-1 \pmod m} \rightarrow G^\dagger_{i-1 \pmod m} G^\dagger_{i-2 \pmod m} \dots G^\dagger_{i+p \pmod m}, \\ \textbf{Backward:}\ & G^\dagger_i G^\dagger_{i-1 \pmod m} \dots G^\dagger_{i-p+1 \pmod m} \rightarrow G^\dagger_{i+1 \pmod m} G^\dagger_{i+2 \pmod m} \dots G^\dagger_{i-p \pmod m}. \end{aligned} \tag{6}$$

Any template T of size m should be independent of smaller templates; that is, an application of a smaller template should not decrease the number of gates in T or make it equal to another template. Circuit optimization using template matching is an iterative procedure where at each step we start at an index gate and attempt to match a given template by considering gates left to the index gate in the target circuit. If the matched gates can be moved together and the substitution is beneficial, the template is applied as defined above. This step is repeated by incrementing the position of the index gate by one when no match is found until the last gate is reached.

Circuit optimization with templates was originally proposed in [24]. This work has been extended with the introduction of graph-based matching techniques [18]. While the methods in these references are applicable to Clifford circuits since they are defined for universal quantum circuits, neither of them leverages the particular structure Clifford circuits have for optimization. After completion of the present work we became aware that template-based optimization techniques have been recently applied to Clifford circuits in [29].

2.2 Peephole Optimization of Quantum Circuits

Peephole optimization [27] is an iterative local optimization technique that optimizes a circuit by considering subcircuits spanning small subsets of qubits A and attempting to replace them with an optimized version drawn from a database (or synthesized on the spot in some other versions). At each step, for a given gate all subcircuits on a fixed small number of qubits (e.g., $|A|{=}4$ in [19]) including that gate are considered. For each subcircuit, its cost and the optimal cost (retrieved from the database of precomputed optimal circuits)

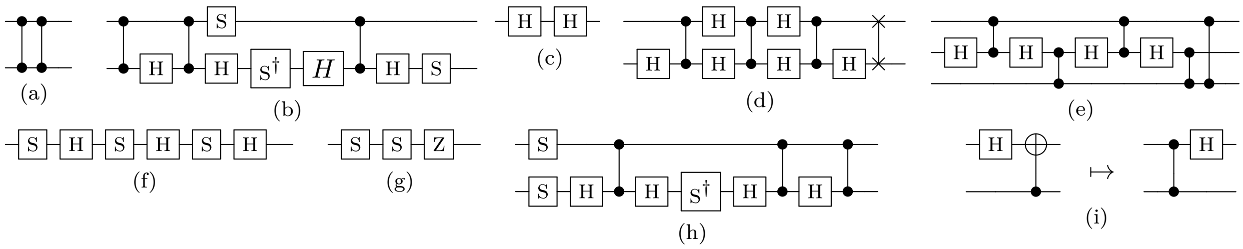


Figure 1: Templates (a)–(h) are used for template matching. The rewriting rule (i) is used for Hadamard gate pushing.

of the unitary it implements are compared. If a substitution is beneficial, the given subcircuit is replaced with its optimal implementation. The step is repeated for all gates until a convergence criterion is satisfied. Peephole optimization of reversible circuits was introduced in [27] and identified to be complementary to template matching. Since its introduction in the context of reversible computations, this approach has been applied to Clifford circuits [19].

The performance of the standard peephole optimization is limited by the need to store the entire database of optimal circuits in memory and to perform $O\left(\binom{n-2}{|A|-2}g^3\right)$ lookups, where g is the number of gates in the circuit [27]. Furthermore, since the size of the n-qubit Clifford group (inclusive of the Pauli group) equals $2^{2n+n^2}\prod_{j=1}^{n}(2^{2j}-1)$ and grows very quickly with n, it is unlikely that all optimal circuits can be found and stored in a suitable database for more than 6 qubits [10].

3 Algorithms

We introduce two algorithms for Clifford circuit optimization and apply them to the problem of compiling optimized Clifford circuits. The first algorithm is a Clifford-specific extension of the template matching technique, which we describe in Subsection 3.2. The second algorithm is symbolic peephole optimization, detailed in Subsection 3.3.

These optimizations can be applied in at least the following two ways. First, if the input is a Clifford unitary, we begin by synthesizing a circuit using a "greedy" compiler (described in Subsection 3.1) and then reduce the gate count by our proposed circuit optimization techniques. Second, if the input is already a Clifford circuit, we can either resynthesize it or apply the circuit optimizations directly. The gate count in the final circuit can be further decreased at the cost of increasing the runtime by a constant factor if the circuit is resynthesized k times using a randomized version of the "greedy" compiler, the k circuits are optimized individually, and the best of the k results is picked. Note that the k repetitions can be done in parallel.

3.1 "Greedy" Compiler

Suppose $U \in \mathcal{C}_n$ is a Clifford unitary to be compiled and $L \in \mathcal{C}_n$ is an operator that reproduces the action of U on a single pair of Pauli operators, X_j and Z_j. In other words, $UPU^{-1} = LPL^{-1}$ for $P \in \{\mathrm{X}_j, \mathrm{Z}_j\}$. The requisite operator L, as well as a Clifford circuit with $O(n)$ CNOTs implementing L, can be easily constructed for any given qubit j by using the standard stabilizer formalism [3]. Then the operator $L^{-1}U$ acts trivially on the jth qubit and can be considered as an element of the Clifford group $\mathcal{C}_{n-1}$. The greedy compiler applies this operation recursively such that each step reduces the number of qubits by one. A qubit j, picked at each recursion step, is chosen such that the operator L has the minimum CNOT count. In the randomized version of the algorithm, qubit j is picked randomly. The compiler runs in time $O(n^3)$ and outputs a circuit with the CNOT count at most $3n^2/4 + O(n)$. We also developed and employ a bidirectional version of the greedy compiler that follows the same strategy as above except that each recursion step applies a transformation $U \leftarrow L^{-1}UR^{-1}$, where $L, R \in \mathcal{C}_n$ are chosen such that after the transformation U acts trivially on the jth qubit and the combined CNOT count of L and R is minimized. In Section 4, we use the bidirectional version of the greedy compiler as it leads to lower CNOT costs of optimized circuits. We include a detailed description of the greedy compilers in Appendix A.

3.2 Template Matching for Clifford Circuits

We extend template matching, described in Subsection 2.1, by introducing a three-stage approach that takes advantage of the observation that Clifford gates map tensor products of Pauli matrices into tensor products of Pauli matrices. Below we describe the features used in the proposed three-stage approach. In Subsection 3.4, we combine this approach with symbolic peephole optimization.

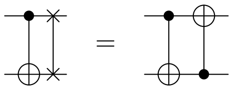
First, we partition the circuit into three stages, "compute," "SWAP," and "Pauli", by pushing SWAP and Pauli gates to the end of the circuit. Paulis are "pushed" according to the rules in Eqs. (2, 3, 4, 5). This step results in the construction of the "compute" stage consisting of H, S, CNOT, and CZ gates only.

Second, we apply the template matching to the "compute" stage. We further simplify template matching by converting all two-qubit gates into CZ gates (at the cost of introducing two Hadamard gates when the CNOT is considered) before performing template optimization. Templates are applied as described in Subsection 2.1. The list of templates is given in Fig. 1a-1h.

We reduce the single-qubit gate count and increase the opportunities for template application by introducing Hadamard and Phase gate pushing. Specifically, assuming that a circuit was optimized with templates, the idea is then to "push" Hadamard and Phase gates to one side of the two-qubit gates as far as possible. "Pushing" a gate through a two-qubit gate is implemented as the application of a template where a fixed subsequence must be matched. For example, the rule in Fig. 1i can be used to push a Hadamard to the right of the CNOT gate.

Note that once the circuit is optimized in terms of the two-qubit gate count, template matching can be applied to reduce the single-qubit gate count by restricting the set of templates and how they are applied. This can be accomplished by applying templates spanning a single qubit and considering certain applications of templates with an even number of two-qubit gates.

Third, we consider SWAP gate optimization as a separate problem. SWAP optimization is performed by observing that a SWAP gate can be implemented at the effective cost of one two-qubit gate if it is aligned with a two-qubit gate (CNOT or CZ) as, for example, in the following.



In order to reduce the number of SWAPs, the SWAP stage is resynthesized with the goal of aligning as many SWAPs as possible with the two-qubit gates in the "compute" stage.

3.3 Symbolic Peephole Optimization

As outlined in Subsection 2.2, various methods were proposed to create a database of optimal few-qubit Clifford circuits; some employ such databases to perform peephole optimization of larger Clifford circuits. However, these methods are limited to few-qubit subcircuits that must be completely decoupled from the remaining qubits. To address this limitation, we introduce a modified approach to Clifford circuit optimization, [illegible] peephole optimization.

Consider a circuit $U \in \mathcal{C}_n$ and a small subset of qubits $A \subseteq [n]$. Our goal is to meaningfully define and optimize the restriction of U onto A. Let $B = [n] \setminus A$ be the complement of A. We say that a CNOT gate is [illegible] if it couples A and B. Assume without loss of generality that each entangling CNOT has its target qubit in the set A (otherwise, switch the control and the target by adding extra Hadamards). Partition

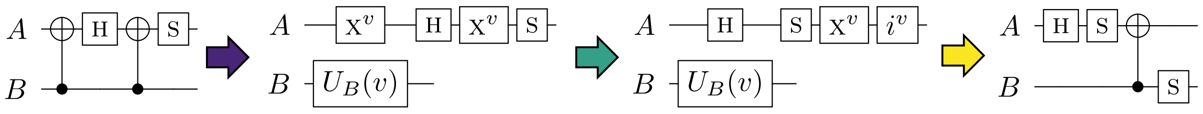


Figure 2: Example of symbolic peephole optimization. *Purple arrow:* each entangling CNOT gate is replaced with a symbolic Pauli gate (SPG) X^v, where $v \in \{0,1\}$. Now the subcircuit acting on A is isolated from the remaining qubits. *Green arrow:* The subcircuit acting on A is optimized to reduce the number of SPGs. Here we used the commutation rules $\mathrm{HX}^v = \mathrm{Z}^v\mathrm{H}$, $\mathrm{X}^v\mathrm{Z}^v = (-i\mathrm{Y})^v$, and $\mathrm{SY}^v = -\mathrm{X}^v\mathrm{S}$. *Yellow arrow:* The subcircuits acting on A and B are merged by replacing the SPG X^v with the CNOT. The phase factor i^v is replaced with the phase gate S acting on B.

entangling CNOTs into groups such that all CNOTs in the same group have the same control bit. Let k be the number of groups. Expanding each entangling CNOT as $|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X$, one can write

$$U = \sum_{v \in \{0,1\}^k} U_A(v) \otimes U_B(v),$$

where $U_A(v)$ is a Clifford circuit obtained from U by retaining all gates acting on A and replacing each entangling CNOT from the ith group with the Pauli gate X^{v_i} acting on the target qubit of the respective CNOT. Likewise, $U_B(v)$ is a (nonunitary) circuit obtained from U by retaining all gates acting on B and replacing each entangling CNOT from the ith group with the projector $|v_i\rangle\langle v_i|$ acting on the control qubit of the respective CNOT. We refer to the single-qubit gates X^{v_i}, Y^{v_i}, and Z^{v_i} as [illegible]. These are similar to controlled Pauli gates except that the control qubit is replaced by a symbolic variable $v_i \in \{0,1\}$.

A symbolic Clifford circuit $U_A(v)$ can be optimized as a regular Clifford circuit on $|A|$ qubits with the following caveats. First, $U_A(v)$ must be expressed by using the Clifford+SPG gate set. The cost of $U_A(v)$ should be defined as the number of CNOTs plus the number of SPGs. Second, the optimization must respect the temporal order of SPGs. In other words, if $i<j$, then all SPGs controlled by v_i must be applied before SPGs controlled by v_j. Third, the optimization must preserve the overall phase of $U_A(v)$ modulo phase factors $(-1)^{v_j}$ or i^{v_j}. The phase factors can be generated by single-qubit gates Z or S applied to control qubits of the entangling CNOTs. These conditions guarantee that the optimized circuit $U_A(v)$ can be lifted to a full circuit $U' \in \mathcal{C}_n$ that is functionally equivalent to U. A toy optimization example is shown in Fig. 2.

We now describe the optimization of $U_A(v)$ in more detail. Let $\mathcal{P}_A$ and $\mathcal{C}_A$ be the groups of Pauli and Clifford operators acting on A, respectively. The circuit $U_A(v)$ can be compactly specified by a k-tuple of Pauli operators $P_1, P_2, \ldots, P_k \in \mathcal{P}_A$ and a Clifford operator $R \in \mathcal{C}_A$ such that $U_A(v) = P_k^{v_k} \cdots P_2^{v_2} P_1^{v_1} R$ for all $v \in \{0,1\}^k$. Indeed, any SPG can be commuted to the left since Clifford gates map Pauli operators to Pauli operators. The most general Clifford+SPG circuit that implements $U_A(v)$ can be parameterized as

$$U_A(v) = (U_k^{-1} Q_k^{v_k} U_k) \cdots (U_2^{-1} Q_2^{v_2} U_2)(U_1^{-1} Q_1^{v_1} U_1) R \tag{7}$$

for some Clifford operators $U_j \in \mathcal{C}_A$ and Pauli operators $Q_j = U_j P_j U_j^{-1} \in \mathcal{P}_A$. The cost of the circuit in Eq. 7 includes the CNOT count of subcircuits $U_j U_{j-1}^{-1}$ and the SPG count of controlled Pauli operators $Q_j^{v_j}$. Note that $Q_j^{v_j}$ is a product of $|Q_j|$ single-qubit SPGs, where $|Q_j|$ is the Hamming weight of Q_j. Denoting $U_0{:=}R^{-1}$, one can express the cost of the circuit in Eq. (7) as

$$f = \$(U_k) + \sum_{i=1}^{k} \$(U_i U_{i-1}^{-1}) + \sum_{i=1}^{k} |U_i P_i U_i^{-1}|. \tag{8}$$

Here $\$(V)$ is the CNOT cost of a Clifford operator $V \in \mathcal{C}_A$. Our goal is to minimize the cost function f over all k-tuples $U_1, U_2, \ldots, U_k \in \mathcal{C}_A$. We claim that the global minimum of f can be computed in time $O(k)$, as long as $|A| = O(1)$. The key observation is that the function f is a sum of terms that depend on at most two consecutive variables U_i and U_{i-1}. Such functions can be minimized efficiently using the dynamic programming method; see, for example, [4]. Indeed, define intermediate cost functions $f_1, f_2, \ldots, f_k : \mathcal{C}_A \to \mathbb{Z}_+$ such that f_j is obtained from f by removing the term $\$(U_k)$, retaining the first j terms in the sums over i, and taking the minimum over $U_1, U_2, \ldots, U_{j-1}$. More formally,

$$f_1(U_1) = \$(U_1 R) + |U_1 P_1 U_1^{-1}| \tag{9}$$

and

$$f_j(U_j) = \min_{U_1, U_2, \ldots, U_{j-1} \in \mathcal{C}_A} \quad \sum_{i=1}^{j} \$(U_i U_{i-1}^{-1}) + \sum_{i=1}^{j} |U_i P_i U_i^{-1}|$$

for $j = 2, 3, \ldots, k$. Using the induction in j, one can easily check that

$$f_j(U_j) = |U_j P_j U_j^{-1}| + \min_{U_{j-1} \in \mathcal{C}_A} \quad f_{j-1}(U_{j-1}) + \$(U_j U_{j-1}^{-1}) \tag{10}$$

for $j = 2, 3, \ldots, k$. Below we assume that a lookup table specifying the CNOT cost $\$(V)$ for all $V \in \mathcal{C}_A$ is available. Then one can compute a lookup table of f_1 by iterating over all $U_1 \in \mathcal{C}_A$ and evaluating the right-hand side of Eq. (9). Proceeding inductively, one can compute a lookup table of f_j with $j = 2, 3, \ldots, k$ by iterating over all $U_j \in \mathcal{C}_A$ and evaluating the right-hand side of Eq. (10). Each step takes time roughly $|\mathcal{C}_A|^2 = O(1)$ since we assumed that $|A| = O(1)$. Finally, use the identity

$$\min_{U_1, U_2, \ldots, U_k \in \mathcal{C}_A} f(U_1, U_2, \ldots, U_k) = \min_{U_k \in \mathcal{C}_A} \$(U_k) + f_k(U_k) \tag{11}$$

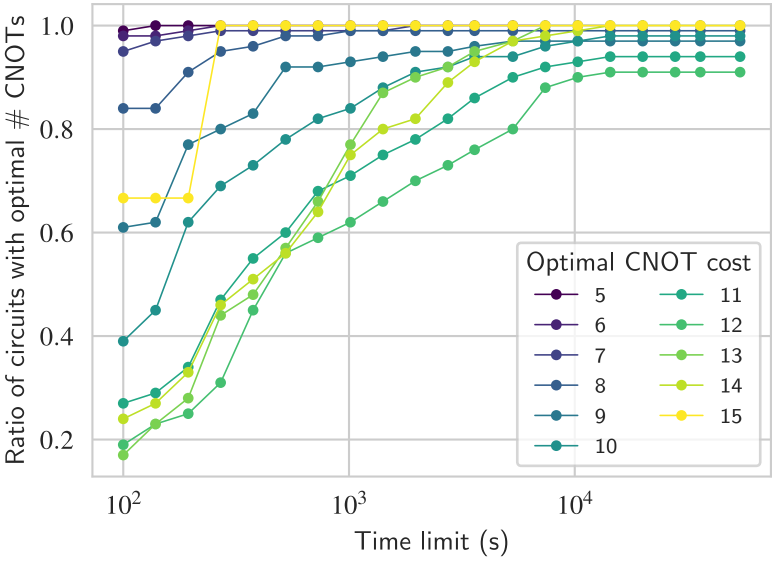
to compute the global minimum of f. Thus, the full computation takes time $O(k)$.

To make the above algorithm more practical, we exploited symmetries of the cost function, Eq. (8). Namely, function f is invariant under multiplying U_j on the left by any element of the local subgroup $\mathcal{C}_A^0 \subseteq \mathcal{C}_A$ generated by the single-qubit gates H_a and S_a with $a \in A$. In other words, $f(U_1, U_2, \ldots, U_k)$ depends only on the right cosets of the local subgroup $\mathcal{C}_A^0 U_j$. Thus one can restrict the minimizations in Eqs. (10,11) to some fixed set of coset representatives $\mathcal{R} \subset \mathcal{C}_A$ such that each coset $\mathcal{C}_A^0 V$ has a unique representative $r(V) \in \mathcal{R}$. We chose $r(V)$ as the left-reduced form of V defined in [10, Lemma 2]. This lemma provides an algorithm for computing $r(V)$ with the runtime $O(|A|^2)$. Now each variable U_i takes only $|\mathcal{R}| = |\mathcal{C}_A|/|\mathcal{C}_A^0| = |\mathcal{C}_A|/24^{|A|}$ values. For example, $|\mathcal{R}|$=20 and $|\mathcal{R}|$=6720 for $|A|$=2 and $|A|$=3, respectively. Likewise, it suffices to compute the lookup table for the CNOT cost $\$(V)$ only for $V \in \mathcal{R}$. This computation was performed using the breadth-first search on the Clifford group $\mathcal{C}_A$.

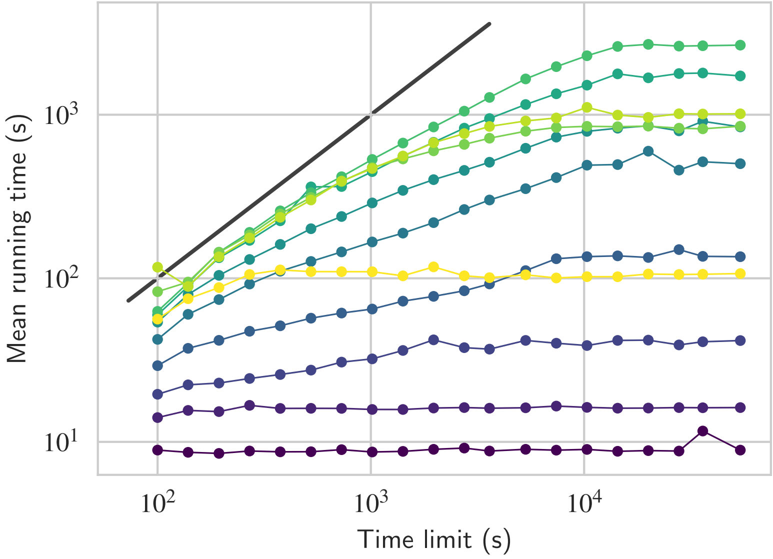
An important open question concerns the selection of the subsets A to be considered. From numerical experiments with $|A| \in \{2, 3\}$, our most successful strategy turned out to be the random subset selection. Specifically, we generate a list of all $\binom{n}{2}$ pairs and $\binom{n}{3}$ triples of qubits. We run passes of the symbolic peephole method first on pairs of qubits and next on triples of qubits until no further improvement can be obtained. At each pass of the symbolic peephole optimization, we randomly reshuffle both lists and run optimization on all the subsets in the reshuffled order. We continue passes until either the optimal CNOT count is reached (for circuits for which the optimal CNOT count is known) or there is no improvement between two consecutive passes.

3.4 Full Algorithm

We combine the components described above in the following way. We begin by synthesizing the circuit using the "greedy" compiler described in Subsection 3.1. Then the synthesized circuit is optimized as follows. First, the circuit is partitioned into three stages. Second, template matching and SWAP gate merging is performed until a pass yields no further optimization. Third, symbolic peephole optimization is performed, as described in Subsection 3.3. Lastly, a single pass of template matching is performed to reduce the single-qubit gate count.



(a) The ratio of circuits for which the implemented methods recover optimal CNOT count. The "nonsmoothness" of the line for the CNOT count of 15 is due to only 3 circuits being considered.



(b) Mean running time of the implemented methods. Note that the actual running times are significantly lower than the time limit (black line) for most instances.

Figure 3: Quality of the solution (a) and the mean running time (b) for 6-qubit circuits with known optimal CNOT gate count. To demonstrate the trade-off between running time and the quality of the solution, we consider 20 time limits between 100 seconds and 15 hours. We observe that for all problems there exists a time limit at which the ratio of the recovered optimal circuits and the mean running time stops increasing. This value depends on the hardness of the circuits; for the hardest circuits (optimal gate count of 12) the metrics are saturated at ≈4 hours. With this time limit we recover the optimal CNOT count for 97.9% of the circuits and observe the difference of 0.2% between the average optimal CNOT count and the average CNOT count recovered by our software. Small deviations from monotonic growth of running time and quality with the time limit are due to the experiments being performed on a heterogeneous computing cluster and the random nature of the algorithm implementation. The mean running time being above the time limit for time limit 100s is due to letting template matching complete even after the time limit is triggered.

Type	n_q	$t_{\max}$	C_{orig}	$C_{\text{A-G}}$	C_{greedy}	C_{opt}	r (%)	$C_{\text{no SWAP}}$	C_{tket}
Path graph	5	12	26.00	12.00	11.50	7.58	36.81	4.50	4.50
Path graph	15	32	231.00	69.72	73.06	37.22	46.62	22.50	22.62
Path graph	25	52	636.00	159.27	146.15	72.73	54.33	43.50	43.88
Path graph	35	72	1241.00	270.79	224.67	109.78	59.46	66.53	67.17
Path graph	45	92	2046.00	421.72	305.65	148.40	64.81	89.96	90.98
Path graph	55	112	3051.00	589.71	384.27	188.29	68.07	113.80	115.46
Cycle graph	5	10	27.50	19.60	12.90	7.80	60.20	7.10	7.30
Cycle graph	15	30	232.50	163.17	76.00	45.90	71.87	32.67	33.60
Cycle graph	25	50	637.50	424.60	155.56	88.12	79.25	60.72	61.94
Cycle graph	35	70	1242.50	796.24	236.14	133.07	83.29	89.10	90.66
Cycle graph	45	90	2047.50	1279.60	321.24	179.00	86.01	118.01	120.00
Cycle graph	55	110	3052.50	1872.48	407.36	227.57	87.85	147.04	149.19
Square lattice	4	4	10.00	6.00	5.00	3.50	41.67	3.50	3.50
Square lattice	9	8	54.00	35.25	33.12	16.50	53.19	15.38	16.50
Square lattice	16	12	156.00	95.42	93.17	40.08	57.99	40.42	45.33
Square lattice	25	24	500.00	254.12	206.50	97.38	61.68	89.75	102.38
Square lattice	36	36	1110.00	621.36	409.89	239.50	61.46	221.39	249.31
Square lattice	49	16	714.00	858.44	553.12	320.19	62.70	298.62	343.44
Square lattice	64	252	14168.00	2189.41	1237.07	887.38	59.47	824.60	904.80
Triangular lattice	3	6	10.50	2.83	3.33	2.83	0.00	2.83	2.83
Triangular lattice	6	10	49.50	22.60	18.60	9.70	57.08	9.60	10.20
Triangular lattice	10	36	333.00	72.58	60.36	29.78	58.97	26.42	28.78
Triangular lattice	15	90	1365.00	187.18	128.17	65.44	65.04	61.40	66.96
Triangular lattice	21	24	562.50	303.08	211.67	117.33	61.29	109.58	117.79
Triangular lattice	28	300	9481.50	726.63	418.46	272.20	62.54	256.57	272.00
Triangular lattice	36	60	2562.00	928.92	556.60	356.87	61.58	340.77	369.03
Triangular lattice	45	300	16254.00	1909.62	1023.72	801.87	58.01	736.90	767.44
Triangular lattice	55	72	4927.50	2195.36	1229.04	968.68	55.88	895.53	922.10
Hexagonal lattice	6	6	21.00	15.33	14.00	8.00	47.83	7.50	7.50
Hexagonal lattice	24	24	375.00	285.88	197.25	101.83	64.38	91.67	101.58
Hexagonal lattice	54	120	4356.00	1711.33	874.77	585.89	65.76	539.67	606.44
Heavy hexagon lattice	12	12	78.00	79.50	58.92	28.42	64.26	21.58	22.67
Heavy hexagon lattice	54	120	3630.00	1747.48	899.80	609.90	65.10	561.50	625.98

Table 1: Optimization results for Hamiltonian evolution circuits. For each graph on n_q qubits, we generate and optimize $t_{\max}$ circuits corresponding to all integer numbers of steps between 1 and $t_{\max} = \min(t_p, 300)$. C_{orig} is the average CNOT gate count in the original circuits, $C_{\text{A-G}}$ is the average CNOT gate count of the circuits in Aaronson–Gottesman canonical form [3], C_{greedy} is the average CNOT gate count of the circuits produced by the bidirectional "greedy" compiler, C_{opt} is the average CNOT gate count of the optimized circuits, and $r = (C_{\text{A-G}} - C_{\text{opt}})/C_{\text{A-G}}$ is the improvement in the average CNOT gate count over the Aaronson–Gottesman canonical form. For all runs we set the time limit to 36 hours and stop both peephole optimization and template matching when the time limit is reached. We note that the "greedy" compiler by itself (without any further optimization) reduces the CNOT gate count by 48.6% compared to [3]. We additionally compare the performance of our methods with the `CliffordSimp` method of tket framework [29] applied to the output of our "greedy" compiler (column C_{tket}). As tket ignores the SWAP gates, we modified our implementation such that once the SWAPs are factored out in template matching phase, they are ignored. The CNOT gate counts are presented in column $C_{\text{no SWAP}}$. We observe that our optimizations result in CNOT counts that are 6.58% lower on average as compared to tket, with larger improvements (up to 17.8%) observed for harder (deeper) circuits.

4 Experimental Results

We ran two sets of computational experiments designed to test the performance of our synthesis and optimization algorithms, detailed in the next two subsections. In addition, we compared our results to [29] as well as to 8-qubit T gate free circuits from [14]. The comparison to [29] is detailed in Table 1 and the comparison to [14] reads 24.4529 (obtained using 10,000 random samples) to 50+ in [14, Figure 3].

4.1 Recovering Optimal CNOT Count for Clifford Unitaries on Six Qubits

First we compare the proposed heuristic methods with the optimal Clifford compiler for $n \leq 6$ qubits [10]. The latter uses breadth-first search on the Clifford group to construct a database specifying the optimal CNOT gate count of each Clifford operator. As shown in [10], the optimal CNOT gate count for 6-qubit Clifford operators takes values $0, 1, \ldots, 15$. We generate 1,003 uniformly sampled random Clifford unitaries with the CNOT gate counts between 5 and 15. We consider only unitaries with the CNOT gate count $\geq$5 because one needs at least 5 CNOTs to entangle all 6 qubits. For the CNOT gate counts from 5 to 14, we consider 100 circuits for each cost value. For the CNOT gate count of 15, there are only 3 Clifford circuits (modulo single-qubit Cliffords on the left and on the right and modulo qubit permutations) to consider [10].

For each Clifford unitary, we start by synthesizing it using the bidirectional "greedy" compiler. The optimization is run as described in Subsection 3.4. The circuit is then resynthesized by using the randomized version of the compiler, and the resynthesized circuit is optimized. This process is repeated until the time limit is reached, and the circuit with the lowest CNOT count is chosen as the output. Note that we also stop the peephole optimization when the time limit is reached, but we allow template matching to complete. The reason is that template matching is fast as compared to peephole optimization and allowing it to complete results in the actual running time above the time limit by only 0.66% of the instances considered.

The quality of the solution obtained by the implemented methods as a function of the time limit is shown in Fig. 3a. Our algorithm converges before exhausting the time limit on most instances. Fig. 3b shows actual observed mean running time as a function of the time limit. We note that the combination of the iterative nature of symbolic peephole optimization and the randomized resynthesis allows the user to trade off the quality of the optimization and the running time as desired.

4.2 Circuits for Hamiltonian Evolution

To evaluate the performance of the proposed methods on circuits with n>6 qubits, we consider a toy model of Hamiltonian time evolution. Suppose $G = (V, E)$ is a fixed graph with n vertices. We place a qubit at each vertex of G. Define a Hamiltonian evolution circuit with time t as

$$\left(\prod_{(i,j) \in E} \mathrm{CZ}_{i,j} \prod_{j=1}^{n} \mathrm{H}_j \right)^t .$$

The layers of Hadamard and CZ gates model time evolution under an external magnetic field and nearest neighbor two-qubit interactions, respectively. We consider several choices for the interaction graph. First, we take instances of the path and cycle graphs with the number of qubits $n \in \{5, 15, 25, 35, 45, 55\}$. Second, we include all three regular plane tessellations (by triangles, squares, and hexagons). We choose the numbers of vertices between 6 and 64 such that the convex hull spanned by the centers of masses of individual tiles in the gapless regular tiling is congruent to the basic tile. Third, we consider a heavy hexagon grid, obtained from hexagonal tessellation by adding a node in the middle of each edge. This set includes some of the frequently appearing qubit-to-qubit connectivities/architectures. We consider the number of layers $1 \leq t \leq t_{\max} = \min(t_p, 300)$, where t_p is the period such that the Hamiltonian evolution with the number of layers t_p produces the identity transformation. For each interaction graph G we compute the CNOT gate count of optimized circuits averaged over the number of layers $t = 1, 2, \ldots, t_{\max}$. The total number of circuits considered is 2,264. We set the time limit to 36 hours and stop both peephole optimization and template matching when the time limit is reached, only allowing the current pass of template matching to complete. Allowing the current pass of template matching to complete results in only a small ratio of problems (4.5%) to significantly (≥10%) exceed the time limit. The results are reported in Table 1. The maximum graph size in these experiments is n=64 because we represent n-qubit Pauli operators by a pair of 64-bit integers in our C++ implementation; this limitation can be easily removed by revising the data structure.

5 Conclusion

We reported a bidirectional synthesis approach and two circuit optimization techniques that extend known approaches to Clifford circuits by exploiting the unique properties of the Clifford group. We demonstrate the effectiveness of these methods by recovering optimal CNOT gate count for 98.9% 6-qubit circuits (over 1,003 samples) and by reducing the CNOT gate count by 64.7% on average for Hamiltonian evolution circuits with up to 64 qubits (2,264 circuits considered), compared to Aaronson–Gottesman canonical form [3]. We show evidence of the improvement in the gate count by a factor of 2 compared to other techniques, such as [14].

Appendix A

In this section, we describe greedy synthesizers that we developed to obtain initial circuits implementing the Clifford unitaries considered. There are two versions of the synthesizer: unidirectional and bidirectional.

Let us start with the unidirectional version. Suppose C is an n-qubit Clifford operator. The greedy compiler sequentially constructs Clifford circuits $L_1, L_2, \ldots, L_n$ such that $C = L_1 L_2 \cdots L_j C_j$, where C_j acts trivially on the first j qubits. The circuit L_j has the CNOT cost $\$(L_j) \le 3(n-j)/2 + O(1)$. By assumption, C_n acts trivially on all n qubits, that is, C_n is proportional to the identity. In other words, $C = \omega L_1 L_2 \cdots L_n$ for some irrelevant phase factor ω. This gives a circuit implementing C with the CNOT cost

$$\sum_{j=1}^{n} \$(L_j) \le \sum_{j=1}^{n} (3(n-j)/2 + O(1)) = 3n^2/4 + O(n).$$

The desired circuits L_j are constructed using the following lemma.

Lemma 1. *There exists an algorithm that takes as input anti-commuting n-qubit Pauli operators O and O' and outputs a Clifford circuit $L \in \mathcal{C}_n$ such that*

$$L^{-1} O L = \mathrm{X}_1 \quad \textit{and} \quad L^{-1} O' L = \mathrm{Z}_1. \tag{12}$$

The circuit L has the CNOT *cost $\$(O, O') := \$(L) \le 3n/2 + O(1)$. The algorithm has runtime $O(n)$.*

We refer to a Clifford operator L satisfying Eq. (12) as a disentangler for the pair (O, O').

To implement a Clifford operator C as a circuit, apply the lemma to Pauli operators $O = C\mathrm{X}_1C^{-1}$ and $O' = C\mathrm{Z}_1C^{-1}$. Let L be the corresponding disentangler and $C_1 = L^{-1}C$. From Eq. (12) we obtain $C_1\mathrm{X}_1 = \mathrm{X}_1C_1$ and $C_1\mathrm{Z}_1 = \mathrm{Z}_1C_1$. This means that C_1 commutes with all Pauli operators on the first qubit, which is possible only if C_1 acts trivially on the first qubit. We can now apply the lemma to C_1. Ignoring the trivial action of C_1 on the first qubit, one can regard C_1 as an element of the smaller Clifford group $\mathcal{C}_{n-1}$, thereby reducing the number of qubits that need to be considered by one. After n applications of the lemma we obtain the desired disentanglers $L_1, L_2, \ldots, L_n$.

Proof of Lemma 1. Let us say that Pauli operators O and O' are in the standard form if their action on any qubit j falls into one of the five cases shown below.

Case	A	B	C	D	E
O_j	X	X	X	I	I
O'_j	Z	X	I	Z	I

Recall that the single-qubit Clifford group $\mathcal{C}_1$ acts by permutations on the Pauli operators X, Y, and Z. Thus one can transform any Pauli pair (O, O') into the standard form by applying a layer of single-qubit Clifford operators. This gives rise to a partition of n qubits into five disjoint subsets A, B, C, D, and E. Note that A has odd size since otherwise O and O' would commute. Let $A(j)$ be the j-th qubit of A. Consider the following circuit.

Algorithm 1 Disentangling circuit

```
if $1 \notin A$ then
    $\mathrm{SWAP}_{1,A(1)}$
end if                                             ▷ Now $O_1 = \mathrm{X}$ and $O'_1 = \mathrm{Z}$
for $j \in C$ do                                   ▷ Clean $O$ from $C$
    $\mathrm{CNOT}_{1,j}$
end for
for $j \in D$ do                                   ▷ Clean $O'$ from $D$
    $\mathrm{CNOT}_{j,1}$
end for
$i \leftarrow$ first qubit of $B$
for $j \in B \setminus \{i\}$ do                   ▷ Clean $(O, O')$ from $B \setminus \{i\}$
    $\mathrm{CNOT}_{i,j}$
end for
$\mathrm{CNOT}_{i,1}\mathrm{H}_i\mathrm{CNOT}_{1,i}$   ▷ Clean $(O, O')$ from $i$
$k \leftarrow (|A| - 1)/2$                         ▷ Due to anticommutativity, $|A|$ is odd; group all but qubit 1 in pairs
for $j = 1$ to $k$ do                              ▷ Clean $(O, O')$ from $A \setminus \{1\}$
    $\mathrm{CNOT}_{A(2j+1),A(2j)}$
    $\mathrm{CNOT}_{A(2j),A(1)}$
    $\mathrm{CNOT}_{A(1),A(2j+1)}$                 ▷ Perform simultaneous mapping $\mathrm{XXX} \mapsto \mathrm{XII}$ and $\mathrm{ZZZ} \mapsto \mathrm{ZII}$
end for
```

Let L be the operator realized by the above circuit combined with the initial layer of single-qubit Cliffords. Direct inspection shows that L has the desired property, Eq. (12), up to sign factors. The latter can be fixed by applying Pauli X_1 or Y_1 or Z_1 as the first gate of L. Simple algebra shows that L has the CNOT gate count of at most $(3/2)|A| + |B| + |C| + |D| + O(1) \leq 3n/2 + O(1)$. □

Below we use the notation $\$(O, O')$ for the CNOT gate count of the disentangling circuit constructed in Algorithm 1.

Our implementation of the greedy compiler optimizes the order in which the qubits are disentangled. Namely, suppose that at some step j of the compiler a subset of qubits S_j has been disentangled such that $|S_j| = j$ and $C = L_1 L_2 \cdots L_j C_j$, where C_j acts trivially on S_j. Let

$$p = \arg\min_{p \notin S_j} \$(C_j \mathrm{X}_p C_j^{-1}, C_j \mathrm{Z}_p C_j^{-1})$$

be a qubit with the smallest disentangling cost. Let L be the circuit disentangling $C_j \mathrm{X}_p C_j^{-1}$ and $C_j \mathrm{Z}_p C_j^{-1}$. Set $S_{j+1} = S \cup \{p\}$ and $L_{j+1} = L \cdot \mathsf{SWAP}_{1,p}$. Then $C = L_1 L_2 \cdots L_{j+1} C_{j+1}$, where C_{j+1} acts trivially on S_{j+1}. Thus one can proceed inductively. The extra SWAP gates are isolated and incorporated back into the compiled circuit as described in Subsection 3.2.

The greedy synthesizer described above has the runtime $O(n^3)$. Indeed, consider the first step of the synthesis. Since the disentangling cost $\$(O, O')$ can be computed in time $O(n)$, picking a qubit with the smallest disentangling cost takes time $O(n^2)$. Computing the disentangling circuit L and the product $L^{-1}C$ takes time $O(n^2)$ since L contains $O(n)$ gates and the action of a single gate can be simulated in time $O(n)$ using the stabilizer formalism [3]. Thus the full runtime of the greedy synthesizer is $O(n^3)$.

The bidirectional greedy synthesizer sequentially constructs Clifford circuits $L_1, L_2, \ldots, L_n$ and $R_1, R_2, \ldots, R_n$ such that

$$C = (L_1 L_2 \cdots L_j) C_j (R_j \cdots R_2 R_1),$$

where C_j acts trivially on the first j qubits. This gives a circuit implementing C with the CNOT cost at most

$$\sum_{j=1}^{n} \$(L_j) + \$(R_j).$$

Consider the first step, j=1. Let us construct the circuits $L = L_1$ and $R = R_1$ (all subsequent steps are analogous). Our goal is to minimize the combined cost $\$(L) + \$(R)$ subject to the constraint that $C_1 = L^{-1}CR^{-1}$ acts trivially on the first qubit. Equivalently, C_1 should commute with the Pauli operators X_1 and Z_1. Define n-qubit Pauli operators

$$P := R^{-1}\mathrm{X}_1 R,\ P' := R^{-1}\mathrm{Z}_1 R,\ O := CPC^{-1},\ \text{and}\ O' := CP'C^{-1}.$$

By definition, R^{-1} is a disentangler for the pair (P, P'). Simple algebra shows that C_1 commutes with the Pauli operators X_1 and Z_1 if and only if L is a disentangler for the pair (O, O'). The above shows that minimizing the combined cost $\$(L) + \$(R)$ subject to the constraint that $C_1 = L^{-1}CR^{-1}$ acts trivially on the first qubit is equivalent to minimizing the function

$$f(P, P') = \$(P, P') + \$(CPC^{-1}, CP'C^{-1})$$

over all pairs of n-qubit anti-commuting Pauli operators P and P'. Note that $f(P, P')$ can be computed in time $O(n)$ for a given pair (P, P'). Once the optimal pair (P, P') is found, one chooses L and R^{-1} as disentanglers for the Pauli pairs (O, O') and (P, P'), respectively. Since the total number of n-qubit anti-commuting Pauli pairs grows exponentially with n, the global minimum of $f(P, P')$ cannot be computed exactly for large n. To make the problem tractable, we restricted the minimization to Pauli operators P, P' with weight at most two. The number of such pairs (P, P') is at most $O(n^3)$ since the anti-commutativity condition implies that the supports of P and P' must overlap on at least one qubit. Now the minimum of $f(P, P')$ can be computed in time $O(n^4)$, and thus the full runtime of the compiler is $O(n^5)$. Note that the unidirectional greedy compiler described earlier corresponds to R=I, that is, $P = \mathrm{X}_1$ and $P' = \mathrm{Z}_1$. Thus the bidirectional compiler subsumes the unidirectional one, even with the restricted minimization domain.

Disclaimer

This paper was prepared for information purposes with contributions from the Future Lab for Applied Research and Engineering (FLARE) Group of JPMorgan Chase & Co. and its affiliates, and is not a product of the Research Department of JPMorgan Chase & Co. JPMorgan Chase & Co. makes no explicit or implied representation and warranty, and accepts no liability, for the completeness, accuracy or reliability of information, or the legal, compliance, tax or accounting effects of matters contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction.

Acknowledgments

This work was supported in part by the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research AIDE-QC and FAR-QC projects and by the Argonne LDRD program under contract number DE-AC02-06CH11357. Clemson University is acknowledged for generous allotment of compute time on the Palmetto cluster. We gratefully acknowledge the computing resources provided on Bebop, a high-performance computing cluster operated by the Laboratory Computing Resource Center at Argonne National Laboratory. SB is partially supported by the IBM Research Frontiers Institute.

References

[1] https://github.com/rsln-s/Clifford_Circuit_Optimization_with_Templates_and_Symbolic_Pauli_Gates.

[2] Scott Aaronson. Shadow tomography of quantum states. In [illegible], STOC 2018, page 325–338, New York, NY, USA, 2018. Association for Computing Machinery. DOI: 10.1145/3188745.3188802.

[3] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. [illegible], 70(5), November 2004. DOI: 10.1103/PhysRevA.70.052328.

[4] Dorit Aharonov, Itai Arad, and Sandy Irani. Efficient algorithm for approximating one-dimensional ground states. [illegible], 82(1):012315, 2010. DOI: 10.1103/PhysRevA.82.012315.

[5] Charles H. Bennett, David P. DiVincenzo, John A. Smolin, and William K. Wootters. Mixed-state entanglement and quantum error correction. [illegible], 54(5):3824, 1996. DOI: 10.1103/PhysRevA.54.3824.

[6] Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal Clifford gates and noisy ancillas. [illegible], 71(2):022316, 2005. DOI: 10.1103/PhysRevA.71.022316.

[7] Sergey Bravyi and Dmitri Maslov. Hadamard-free circuits expose the structure of the Clifford group. [illegible], 67(7):4546–4563, 2021. DOI: 10.1109/TIT.2021.3081415.

[8] Sergey Bravyi, Jay M. Gambetta, Antonio Mezzacapo, and Kristan Temme. Tapering off qubits to simulate fermionic Hamiltonians. [illegible]*701.08213*, 2017.

[9] Sergey Bravyi, David Gosset, and Robert König. Quantum advantage with shallow circuits. [illegible], 362(6412):308–311, 2018. DOI: 10.1126/science.aar3106.

[10] Sergey Bravyi, Joseph A. Latone, and Dmitri Maslov. 6-qubit optimal Clifford circuits. [illegible]*2012.06074*, 2020.

[11] Richard Cleve, Debbie Leung, Li Liu, and Chunhao Wang. Near-linear constructions of exact unitary 2-designs. [illegible], 16(9 & 10):0721–0756, 2016.

[12] Christoph Dankert, Richard Cleve, Joseph Emerson, and Etera Livine. Exact and approximate unitary 2-designs and their application to fidelity estimation. [illegible], 80(1):012304, 2009. DOI: 10.1103/PhysRevA.80.012304.

[13] Shantanu Debnath, Norbert M. Linke, Caroline Figgatt, Kevin A. Landsman, Kevin Wright, and Christopher Monroe. Demonstration of a small programmable quantum computer with atomic qubits. [illegible], 536(7614):63–66, 2016. DOI: 10.1038/nature18648.

[14] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. Graph-theoretic simplification of quantum circuits with the ZX-calculus. [illegible], 4:279, 2020. DOI: 10.22331/q-2020-06-04-279.

[15] Timothee Goubaultdebrugiere, Marc Baboulin, Benoit Valiron, Simon Martiel, and Cyril Allouche. Reducing the depth of linear reversible quantum circuits. [illegible], 2021. DOI: 10.1109/TQE.2021.3091648.

[16] Hsin-Yuan Huang, Richard Kueng, and John Preskill. Predicting many properties of a quantum system from very few measurements. [illegible], 16:1050—1057, 2020. DOI: 10.1038/s41567-020-0932-7.

[17] IBM. IBM Quantum Experience. last accessed 11/9/2020. URL https://quantum-computing.ibm.com/.

[18] Raban Iten, Romain Moyard, Tony Metger, David Sutter, and Stefan Woerner. Exact and practical pattern matching for quantum circuit optimization. [illegible]*909.05270*, 2019.

[19] Vadym Kliuchnikov and Dmitri Maslov. Optimization of Clifford circuits. [illegible], 88(5):052307, 2013. DOI: 10.1103/PhysRevA.88.052307.

[20] Emanuel Knill. Quantum computing with realistically noisy devices. [illegible], 434(7029):39–44, 2005. DOI: 10.1038/nature03350.

[21] Emanuel Knill, Dietrich Leibfried, Rolf Reichle, Joe Britton, R. Brad Blakestad, John D. Jost, Chris Langer, Roee Ozeri, Signe Seidelin, and David J. Wineland. Randomized benchmarking of quantum gates. [illegible], 77(1):012307, 2008. DOI: 10.1103/PhysRevA.77.012307.

[22] Samuel A. Kutin, David Petrie Moulton, and Lawren M. Smithline. Computation at a distance. [illegible]0701194, 2007.

[23] Easwar Magesan, Jay M. Gambetta, and Joseph Emerson. Scalable and robust randomized benchmarking of quantum processes. [illegible], 106(18):180504, 2011. DOI: 10.1103/PhysRevLett.106.180504.

[24] Dmitri Maslov, D. Michael Miller, Gerhard W. Dueck, and Camille Negrevergne. Quantum circuit simplification and level compaction. [illegible], 27(3):436–444, March 2008. DOI: 10.1109/TCAD.2007.911334.

[25] Michael A. Nielsen and Isaac Chuang. [illegible]. 2002. DOI: 10.1017/CBO9780511976667.

[26] Ketan N. Patel, Igor L. Markov, and John P. Hayes. Optimal synthesis of linear reversible circuits. [illegible], 8(3):282–294, March 2008. ISSN 1533-7146.

[27] Aditya K. Prasad, Vivek V. Shende, Igor L. Markov, John P. Hayes, and Ketan N. Patel. Data structures and algorithms for simplifying reversible circuits. [illegible], 2(4):277–293, October 2006. DOI: 10.1145/1216396.1216399.

[28] Kanav Setia, Richard Chen, Julia E. Rice, Antonio Mezzacapo, Marco Pistoia, and James D. Whitfield. Reducing qubit requirements for quantum simulations using molecular point group symmetries. [illegible], 16(10):6091–6097, 2020. DOI: 10.1021/acs.jctc.0c00113.

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

[30] Ewout van den Berg and Kristan Temme. Circuit optimization of Hamiltonian simulation by simultaneous diagonalization of Pauli clusters. [illegible], 4:322, 2020. DOI: 10.22331/q-2020-09-12-322.