Graph optimization perspective for low-depth Trotter-Suzuki decomposition

Albert T. Schmitz,[1,2,*] Nicolas P. D. Sawaya,[3] Sonika Johri,[1,†] and A. Y. Matsuura[1]
[1]*Intel Labs, Intel Corporation, Hillsboro, Oregon 97124, USA*
[2]*Department of Physics and Center for Theory of Quantum Matter, University of Colorado, Boulder, Colorado 80309, USA*
[3]*Intel Labs, Intel Corporation, Santa Clara, California 95054, USA*

(Received 26 May 2023; accepted 13 March 2024; published 18 April 2024)

Hamiltonian simulation represents an important module in a large class of quantum algorithms and simulations such as quantum machine learning, quantum linear algebra methods, and modeling for physics, material science, and chemistry. One of the most prominent methods for realizing the time-evolution unitary is via the Trotter-Suzuki decomposition. However, there is a large class of possible decompositions for the infinitesimal time-evolution operator as the order in which the Hamiltonian terms are implemented is arbitrary. We introduce a perspective for generating a low-depth Trotter-Suzuki decomposition assuming the standard Clifford+RZ gate set by adapting ideas from quantum error correction. We map a given Trotter-Suzuki decomposition to a constrained path on a graph which we deem the Pauli frame graph (PFG). Each node of the PFG represents the set of possible Hamiltonian terms currently available to be applied, Clifford operations represent a move from one node to another, and so the graph distance represents the gate cost of implementing the decomposition. The problem of finding the optimal decomposition is then equivalent to solving a problem similar to the traveling salesman. Although this is an NP-hard problem, we demonstrate the simplest heuristic, greedy search, and compare the resulting two-qubit gate count and circuit depth to more standard methods for a large class of scientifically relevant Hamiltonians, both fermionic and bosonic, found in chemical, vibrational, and condensed-matter problems which naturally scale. We find, in nearly every case we study, the resulting depth and two-qubit gate counts are less than those provided by standard methods, by as much as an order of magnitude. We also find the method is efficient and amenable to parallelization, making it scalable for problems of real interest.

DOI: 10.1103/PhysRevA.109.042418

I. INTRODUCTION

In the near-term intermediate scale quantum (NISQ) era of quantum computation, a great deal of work has gone into the compilation and optimization of quantum circuits. Optimization is especially important as near-term quantum hardware is limited by depth and number of gates before noise and other constraints render the outcome unusable. Most current methods are focused on taking as user input a sequence of circuit elements which are then manipulated iteratively via a suite of optimization routines [1–9]. Although these methods reduce circuit complexity, they often only do so by finding local patterns and can greatly suffer from nonoptimal choices by the user. Thus, we look to find a method which exploits a higher-level representation to decrease the depth of the final quantum program and is not prohibitively expensive when applied to real problems, similar to methods found in Refs. [4,10–13].

In this paper, we consider a general Hermitian operator of any particle type as one such representation. An ubiquitous use for a quantum computer approximates the unitary generated by exponentiating such operators. Such a use can be the primary computation or a module within some larger algorithm. This includes problems found in condensed matter [14,15], high-energy physics [16], materials science [17,18], and chemistry [19–24]. The most common method for generating this approximation is the Trotter-Suzuki decomposition [25], though extensions and more advanced methods exist [26–32]. Moreover, Trotter-Suzuki decomposition is a basis for other quantum algorithms such as the variational quantum eigensolver (VQE) [33–35], and other advanced energy estimation algorithms [36–39]. One first writes the Hermitan operator in a basis consisting of all tensor products of single-qubit Pauli operators (henceforth referred to as Pauli operators) using several methods depending on the particle type or problem description (see Sec. VI A). The Trotter-Suzuki decomposition then applies multiple repeated *Trotter steps*, each of which is a sequence of unitary rotations about each Pauli operator term with an angle whose size is inversely proportional to the accuracy of the approximation. We refer to this type of circuit form as the *product-of-Pauli-rotations* form (PoPR).

Although the PoPR form is natural for qubits and is universal for computation, this expression of a unitary does not map immediately to most hardware implementations of quantum computation. In particular, we focus on implementations for which the native gate set consists of one two-qubit entangling Clifford gate such as the controlled-NOT (CX) or controlled-Z (CZ) gate and all single-qubit rotation gates, what we refer

[*]albert.schmitz@intel.com
[†]S.J. is now at Coherent Computing.

to as the Clifford+RZ gate set.[1] This gate set is known to be universal [40] and reflects the gate set of many popular platforms such as semiconductor quantum dot and superconductor transmon implementations. In both these cases, the two-qubit entangling Clifford gate is noisier and takes longer than any single-qubit rotation, and thus represents the primary limiting resource for these NISQ implementations. This consideration may change in the future for certain implementations of quantum error correction [11]. The typical translation of the PoPR form into a Clifford+RZ circuit is to implement each Pauli rotation via a so-called CX ladder or staircase [41], making them expensive individually, and then try to optimize the resulting circuit via Clifford gate optimizations at their intersection. Such optimizations are either local such as those which rely on local methods such as pattern matching or small-circuit synthesis [7,42,43] or relatively expensive if applied to every sequence of Clifford gates in a circuit [44–46]. However, all such methods are incapable of choosing an optimal ordering for the application of these rotations, implying that even the best Clifford optimizations can not express many available optimizations. This problem is particularly acute in the case of Trotter-Suzuki decomposition where within a single Trotter step, the order of rotations is arbitrary to within the allowed error of the method. Although there exist ordering methods for specific problem instances [47–49], one would like a general method. One such method can be found in Ref. [50] which uses block sorting of the Pauli operators representing the terms based on different considerations. In Ref. [51], we also find a general method based on synthesizing the circuit from the perspective of rotations, which is reminiscent of our methods. However, their methods use a strategy of breaking down larger Pauli operator rotations into the product of smaller, support 2 operators and reorder terms.

In this paper, we introduce a perspective and methodology familiar to quantum error correction, but applied to the problem of synthesizing a highly efficient Clifford+RZ circuit from the PoPR form. This perspective allows us to simultaneously choose an efficient ordering of Pauli rotations and efficient Clifford operations which connect these rotations. We do so by viewing any Clifford+RZ circuit as a walk through a hypothetical graph we deem the *Pauli frame graph* (PFG). The PFG contains nodes which correspond to a distinct Pauli frame (similar to the Pauli tableau of Refs. [44,45]) where edges are added between nodes if a Clifford gate in our gate set connects the two Pauli frames. Thus, we can view any circuit as a walk through this graph, effectively applying each Pauli rotation at some node along the path, where the gate cost of the circuit is now proportional to the length of the walk. Although this graph is never explicitly constructed, it allows us to view optimal circuit synthesis as a graph optimization problem where, despite being an NP-hard problem, we can apply known heuristics. As a demonstration of the power of this view, we implement the simplest heuristic, the greedy search. In spite of its apparent simplicity, we demonstrate a marked improvement over simple ladder methods for a set of condensed matter and chemistry simulations, both fermionic and bosonic, which have natural scaling properties: the Fermi-Hubbard model, the Bose-Hubbard model, a polyacetylene chain, and a vibronic model. The latter two models have significantly denser Hamiltonians (greater number of terms per qubit) than the former two. We find that in nearly every case our methods result in fewer two-qubit gates and a significantly lower depth. We also demonstrate that the method is computationally efficient in producing the final circuit. We also find that unlike in other studies [52], there is no appreciable difference in circuit cost between the different fermionic encodings using our methods.

The remainder of this paper is organized as follows: In Sec. II we motivate our ideas with a simple example. We follow this with a theoretical description of the PFG, the peripheral details, and how a walk on the PFG represents the synthesis of a circuit in Sec. III. Section IV discusses details of this perspective regarding the synthesis of one step in a Trotter-Suzuki decomposition and Sec. V outlines the ultragreedy heuristic algorithm using this perspective. Section VI describes the models we use to test the ultragreedy algorithm and the results before making some concluding remarks in Sec. VII.

II. EXAMPLE AND MOTIVATION FOR THE PFG METHOD

To motivate the following procedure, we start with a simple example. We assume a basic understanding of the first-order Trotter-Suzuki decomposition. A basic review can be found in Ref. [41].

Suppose we have a four-qubit system and we want to simulate the Hamiltonian

$$\begin{aligned} H = {} & \theta_{01}Z_0Z_1 + \theta_{12}Z_1Z_2 + \theta_{23}Z_2Z_3 + \theta_{03}Z_0Z_3 \\ & + \theta_{\text{all}}Z_0Z_1Z_2Z_3. \end{aligned} \tag{1}$$

The time-evolution operator generated by this Hamiltonian is

$$\begin{aligned} U(T) &= \exp(-iTH) \\ &= \exp(-iT\theta_{01}Z_0Z_1)\exp(-iT\theta_{12}Z_1Z_2) \\ &\quad \times \exp(-iT\theta_{23}Z_2Z_3)\exp(-iT\theta_{03}Z_0Z_3) \\ &\quad \times \exp(-iT\theta_{all}Z_0Z_1Z_2Z_3), \end{aligned} \tag{2}$$

where equality in this case is a consequence of all the constituent operators mutually commuting. Using the standard method of synthesis for this circuit, one generates $\exp(-iT\theta_{01}Z_0Z_1)$, for example, by computing or coherently writing the eigenvalue of Z_0Z_1 on qubit 1 via a control-X or CX gate, then applying a single-qubit Z rotation, RZ with angle $2T\theta_{01}$ and then uncomputing Z_0Z_1. One then moves on to the next term using the same strategy, possibly in parallel, and so on until all terms have been applied. However, we recognize that the uncompute is redundant as $Z_0Z_1Z_2Z_3 = Z_0Z_1 \times Z_2Z_3$. As we have computed half this operator after the first step, we

[1]As a practical matter, no implementation can truly realize all arbitrary single-qubit rotations natively. However, one only needs at minimum the Clifford H gate and the non-Clifford T gate, at which point any other rotation can be achieved with arbitrary precision [41]. Thus, it is merely a theoretical convenience to assume arbitrary rotations from the outset.

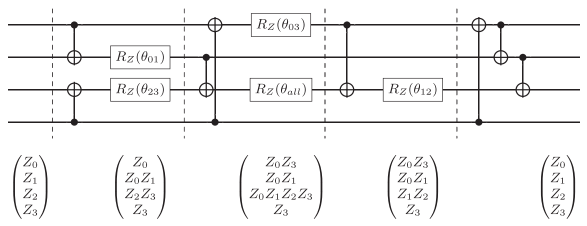

FIG. 1. Circuit diagram for our simple example. Below the circuit represents the frame which provides the effective Z-rotation axis for each qubit.

should keep Z_0Z_1 until the θ_{all} term is computed. Thus, we are looking to leverage linear dependence in the *Pauli space* (to be defined below) and use this dependence to avoid unnecessary uncompute sequences. To truly leverage this dependency, we need to be more systematic. The key observation is that each time we apply a CX, we are effectively applying a *symplectic automorphism* to the Pauli space. In this case, we only need to consider the space of all Z-operator products which is generated by or spanned by the basis, also referred to as a *frame* (as defined in Ref. [53]) (Z_0, Z_1, Z_2, Z_3). However, this frame is not unique in generating any Z-type operator. Just as a unitary operator is completely defined by its action on a basis, a symplectic automorphism is uniquely specified by how it transforms the frame. Consider the circuit in Fig. 1. In the first step of the circuit, we are taking $(Z_0, Z_1, Z_2, Z_3) \to (Z_0, Z_0Z_1, Z_2Z_3, Z_3)$. As this is a frame, any Z-type operator can be written as a product (or "sum" in the Pauli space) of these operators. For any given operator, the number of terms in its expansion in this frame (minus one) tells us how many CX gates it takes to compute a rotation for that operator. For example, after the first step in Fig. 1, we have three remaining terms to compute: $Z_1Z_2 = Z_0 \times Z_0Z_1 \times Z_2$, $Z_0Z_3 = Z_0 \times Z_2 \times Z_2Z_3$, and $Z_0Z_1Z_2Z_3 = Z_0Z_1 \times Z_2Z_3$. From this position, it is best to next compute the θ_{all} term by adding one additional CX gate. From there we can do the same analysis to find that the best move is to include one CX to calculate Z_0Z_3 and one CX to calculate Z_1Z_2. After the final rotation, we are in the frame $(Z_0Z_3, Z_0Z_1, Z_1Z_2, Z_3)$. As we want to return to the original frame, thus completing the Trotter step, we include three additional CX to return to the (Z_0, Z_1, Z_2, Z_3) frame. The final circuit uses eight CX gates and has a depth of 8 (moving one CX to the left in Fig. 1). Note that if we apply several Trotter steps, we do not need to return to the original frame, but rather start the new Trotter step in the ending Pauli frame of the last step. This leads to a method we refer to as *retracing* which we introduce in Sec. V A.

As one can now see, our circuit is generally traversing different frames, such that every operator of our Hamiltonian is in at least one frame which we discuss below as a path in the PFG. To complete a single Trotter step, we must then come back to the same frame, implying the path is a cycle. This example nicely demonstrates the general idea, but all our terms were mutually commuting which is not general. Instead, we must consider the full Pauli space and not just the Z-type Pauli space. This makes our Clifford automorphisms slightly more complicated, but the general idea is the same.

III. DESCRIPTION OF THE PFG AND ITS USES FOR CIRCUIT SYNTHESIS

In this section we discuss the general theory leading up to a full description of the PFG. This uses many ideas familiar to quantum error correction [54] and Clifford circuit simulation. The following draws heavily from and extends results for Refs. [44,45,55–57].

A. Pauli space, Pauli frames, and symplectic automorphisms

Let N be the number of qubits in our system such that the Hilbert space is $\mathcal{H} \simeq \mathbb{C}_2^{\otimes N}$. For each qubit, we have a natural operator basis in the form of the single-qubit Pauli operators

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{3a}$$

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \tag{3b}$$

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{3c}$$

along with the identity. Each Pauli operator is both Hermitian and unitary, implying it squares to the identity and has ± 1 eigenvalues. This operator basis can be extended to the full Hilbert space by taking arbitrary tensor products of these single-qubit operators. In general, we refer to such tensor products as Pauli operators and denote them only by their single-qubit support, i.e., $Z_0 = Z \otimes I \otimes I \ldots$. Note that all such Pauli operators either commute or anticommute.

Let $\mathcal{P}'$ be the *Pauli group*, or set of all tensor products of single-qubit Pauli operators acting on our system which is closed under multiplication. As we often do not care about the overall phases, we further define the *Pauli space* as $\mathcal{P} = \mathcal{P}'/U(1)$, i.e., we have modded out any phase in such a way that, for example, $X_iZ_j \simeq -iX_iZ_j$. Modding out the phase allows us to treat $\mathcal{P}$ as a vector space over the field of two elements $\mathbb{F}_2 = \{0, 1\}$ with dimension $2N$ as given by the fact that the space is generated by all single-qubit Pauli operators and $X_iZ_i \propto Y_i$. To see this forms a vector space, we consider the N-qubit identity operator to be the zero element of $\mathcal{P}$, the addition operation (which is modulo 2) is given by the product of operators, i.e., for $p, q \in \mathcal{P}$, $pq \to p + q$, and scalar multiplication is the power of the operator, i.e., for $a \in \{0, 1\}$, $p^a \to ap$. As $p^2 \simeq I$, $\mathbb{F}_2$ is the appropriate field. However, modding out the phase means we have lost the commutation relations between Pauli operators. To recover this, we introduce the nondegenerate symplectic form $\lambda : \mathcal{P} \times \mathcal{P} \to \mathbb{F}_2$ such that

$$(p, q) \mapsto \lambda(p, q) = \begin{cases} 0, & p \text{ and } q \text{ commute} \\ 1, & \text{otherwise.} \end{cases} \tag{4}$$

One should think of the combination $(\mathcal{P}, \lambda)$ much like an inner product space where the inner product form is replaced with a symplectic form [$\lambda(p, p) = 0$ for all $p \in \mathcal{P}$], thus making this a symplectic vector space. Also much like an inner product space, one generally wants to use λ to compute the expansion of any vector in some appropriate basis, thus making such a basis a frame [53]. For a symplectic vector space, the appropriate frame $B \subset \mathcal{P}$ which we organize as [45]

$$B \simeq \begin{pmatrix} s_0 & \tilde{s}_0 \\ s_1 & \tilde{s}_1 \\ s_2 & \tilde{s}_2 \\ \vdots & \vdots \end{pmatrix} \tag{5}$$

must satisfy

$$\lambda(s_i, s_j) = \lambda(\tilde{s}_i, \tilde{s}_j) = 0, \tag{6a}$$

$$\lambda(s_i, \tilde{s}_j) = \delta_{ij}. \tag{6b}$$

Any frame satisfying these relations is referred to as a *Pauli frame*. Those familiar with quantum error correction and Clifford circuit simulation will recognize the Pauli frame as equivalent to the Pauli tableau as defined in [45]. In the reference, the left side, untilded, operators are referred to as *stabilizers* and the right side, tilded, operators are referred to as *destabilizers*. We label the collection of all such frames as $\mathcal{B}$. In particular, one recognizes the single-qubit basis

$$B_0 \simeq \begin{pmatrix} Z_0 & X_0 \\ Z_1 & X_1 \\ Z_2 & X_2 \\ \vdots & \vdots \end{pmatrix} \tag{7}$$

has this form and is referred to as the *origin frame*. Using a given Pauli frame B, one can expand any $p \in \mathcal{P}$ in this basis according to

$$p = \sum_i [\lambda(s_i, p)\tilde{s}_i + \lambda(\tilde{s}_i, p)s_i]. \tag{8}$$

As an example of Eq. (8), we find that the expansion in the origin frame of $[X_0Y_1Z_2] \in \mathcal{P}$ is trivially

$$[X_0Y_1Z_2] = \lambda(Z_0, X_0Y_1Z_2)[X_0] \tag{9}$$
$$+\lambda(Z_1, X_0Y_1Z_2)[X_1] \tag{10}$$
$$+\lambda(X_1, X_0Y_1Z_2)[Z_1] \tag{11}$$
$$+\lambda(X_2, X_0Y_1Z_2)[Z_2]. \tag{12}$$

However, in the frame generated by CX_{01} [see Eq. (32)], we have that

$$[X_0Y_1Z_2] = \lambda(Z_0, X_0Y_1Z_2)[X_0X_1] \tag{13}$$
$$+\lambda(X_0X_1, X_0Y_1Z_2)[Z_0] \tag{14}$$
$$+\lambda(X_1, X_0Y_1Z_2)[Z_0Z_1] \tag{15}$$
$$+\lambda(X_2, X_0Y_1Z_2)[Z_2]. \tag{16}$$

Linear maps on the space $(\mathcal{P}, \lambda)$ also have a notion similar to unitarity in inner product spaces.[2] We refer to a linear map $\gamma : \mathcal{P} \to \mathcal{P}$ as a symplectic automorphism if and only if it preserves λ. That is, for all $p, q \in \mathcal{P}$

$$\lambda(\gamma(p), \gamma(q)) = \lambda(p, q). \tag{17}$$

[2]Recall that unitarity is defined in the context of an inner product space $(\mathcal{V}, \langle\cdot,\cdot\rangle)$ where a unitary operator is defined as any linear operator U such that for all $u, v \in \mathcal{V}$, $\langle U(u), U(v)\rangle = \langle u, v\rangle$. This then implies the usual notion of unitarity that $U^\dagger = U^{-1}$, where $\dagger$ signifies the adjoint with respects to $\langle\cdot,\cdot\rangle$.

For finite N, a symplectic automorphism γ can be generated by a member of the *unitary Clifford group* $V_\gamma \in C_N \subset U(2^N)$ which is defined as the normalizer of the Pauli group on the qubit system. The related symplectic automorphism is then given by conjugating all Pauli operators and modding out the phase, i.e., $\gamma \simeq \mathcal{P} \to (V_\gamma^\dagger \mathcal{P}' V_\gamma)/U(1)$. However, the correspondence is not one to one as right multiplying a Clifford unitary by any Pauli operator q and conjugating any other Pauli operator p by it only alters the sign of the result, i.e.,

$$(V_\gamma q)^\dagger p (V_\gamma q) = (-1)^{\lambda(p,\gamma^{-1}(q))} V_\gamma^\dagger p V_\gamma, \tag{18}$$

and likewise for left multiplication. Such a sign is then modded out in $\mathcal{P}$. However, if we keep track of the overall sign of the Pauli operators under this mapping, the resulting map is one to one with the corresponding Clifford unitary up to an overall phase [45]. This implies the symplectic group on the Pauli space $\text{Sp}(2N, \mathbb{F}_2)$ is equivalent to $C_N/\mathcal{P}'$ which we refer to as the *Clifford factor group*. Finally, the group naturally induces a group action on Pauli frames $(\text{Sp}(2N, \mathbb{F}_2), \mathcal{B}_N) \to \mathcal{B}_N$ via

$$\left(\gamma, \begin{pmatrix} s_0 & \tilde{s}_0 \\ s_1 & \tilde{s}_1 \\ s_2 & \tilde{s}_2 \\ \vdots & \vdots \end{pmatrix}\right) \mapsto \begin{pmatrix} \gamma(s_0) & \gamma(\tilde{s}_0) \\ \gamma(s_1) & \gamma(\tilde{s}_1) \\ \gamma(s_2) & \gamma(\tilde{s}_2) \\ \vdots & \vdots \end{pmatrix}. \tag{19}$$

We show in Appendix A that this group action is both free and transitive, which importantly implies that for any two frames B_1, B_2, there is a unique symplectic automorphism and thus unique member of the Clifford factor group which connects them.

To extend Pauli frames so as to encapsulate all Clifford unitaries, we only need to include the sign of each Pauli operator contained in the frame. By convention, a member of the Pauli space is mapped to the corresponding member of the Pauli group (i.e., we assign a phase) which corresponds to the conventional Hermitian version in terms of the X, Y, and Z operator definitions of Eqs. (3) for each tensor factor of the Hilbert space. Using this convention, we can unambiguously define a sign for all Hermitian Pauli operators. We then view the signed frame $\underline{B}$ as defining the Clifford unitary $V_{\underline{B}}$ via the relation

$$\underline{B} = V_{\underline{B}}^\dagger B_0 V_{\underline{B}}, \tag{20}$$

where conjugation by the unitary is entrywise for each member of the signed frame. By our convention, if we map the symplectic automorphism whose group action takes $B_0 \to B$ to the Clifford operator which takes the members of B_0 to B, where B is nearly $\underline{B}$ but with all positive signs, then $V_B^\dagger V_{\underline{B}}$ is equal to a Pauli operator. This residual Pauli operator p is such that it flips the signs of B to that of $\underline{B}$ under conjugation. As B is a basis for the Pauli space, the residual Pauli operator is given uniquely up to a phase by

$$p = \sum_i [\text{signbit}(\underline{s}_i)\tilde{s}_i + \text{signbit}(\underline{\tilde{s}}_i)s_i]. \tag{21}$$

Thus, the additional signs resolve the left Pauli operator for the Clifford $V_{\underline{B}}$ over that of V_B implying we have completely specified the full Clifford operation up to an overall phase.[3]

B. Clifford gate action on Pauli frames

As we assume the Clifford + RZ gate set, our Clifford gate set contains CX, H, and $S = \mathrm{RZ}(\frac{\pi}{2})$, where we note that $S^2 \propto Z$ which is then used to generate all of the $\mathcal{P}'$ factors of the Clifford group.[4] This implies $S^\dagger = S^3 \simeq S$ in $\mathrm{Sp}(2N, \mathbb{F}_2)$. Thus, we can also use CX, H, and S as the generators for all of $\mathrm{Sp}(2N, \mathbb{F}_2)$. We also assume all-to-all connectivity for now and leave methods for incorporating limited connectivity to future work.

In the case of sequential gate action, each Clifford gate represents two distinct group actions on the frame which we term the *backward* and *forward* action. That is, if a Clifford gate g is represented by the Clifford operator V_g, then the backward and forward group action on the Pauli frame is defined, respectively, as

$$\begin{aligned} B &\mapsto V_B^\dagger V_g^\dagger B_0 V_g V_B, \\ B &\mapsto V_g B V_g^\dagger, \end{aligned} \tag{22}$$

where we consider the frame-to-Clifford mapping as given in Eq. (20). For the same gate, the backward and forward actions result in generally distinct Pauli frames. The forward action applies the conjugation rule (the symplectic automorphism along with any sign transformation for the signed frame) directly to each member of the frame irrespective of any other. A backward transformation, on the other hand, first conjugates each member of the origin frame by the adjoint of the gate, and then transforms each of those by the unitary representing the frame. The overall result is a transformation rule which replaces one frame entry with some linear combination of the former entries. From the perspective of symplectic automorphisms, this distinction is artificial as a backward transformation is equivalent to some forward transformation via the relation $V_g^{\text{backward}} = (V_B V_g V_B^\dagger)^{\text{forward}}$. However, the forward transformation is agnostic to the current frame, where as the backward transformation is dependent on the current frame. As we show in a moment, we use the backward transformations for circuit synthesis.[5] Alternatively, quantum error correction is typically concerned with the forward action, as the left elements of the Pauli frame or tableau then represent the stabilizers for the state generated by applying the Clifford circuit to the all-zero computational state.

The backward action of the symplectic automorphism for each gate in our gate set is given by the following rules: For any $B = \{(s_i, \tilde{s}_i)\}_{0\leqslant i<N} \in \mathcal{B}_N$,

$$\begin{aligned} \mathrm{CX}_{ij} : s_j &\mapsto s_j + s_i; \\ \tilde{s}_i &\mapsto \tilde{s}_i + \tilde{s}_j, \end{aligned} \tag{23a}$$

$$H_i : s_i \leftrightarrow \tilde{s}_i, \tag{23b}$$

$$S_i : \tilde{s}_i \mapsto s_i + \tilde{s}_i, \tag{23c}$$

where members of B not mapped by these rules are unchanged. We shall also identify another important symplectic automorphism, the SWAP, whereby

$$\begin{aligned} \mathrm{SWAP}_{ij} : s_i &\leftrightarrow s_j; \\ \tilde{s}_i &\leftrightarrow \tilde{s}_j \\ &= \mathrm{CX}_{ij} \circ \mathrm{CX}_{ji} \circ \mathrm{CX}_{ij} \\ &= \mathrm{CX}_{ji} \circ \mathrm{CX}_{ij} \circ \mathrm{CX}_{ji}. \end{aligned} \tag{24}$$

The SWAP generates the symmetric subgroup $\mathcal{S}_N \subset \mathrm{Sp}(2N, \mathbb{F}_2)$ which corresponds to qubit swapping.

Rules for determining the resulting sign for these gates can be found in Appendix B.

C. Circuit synthesis using the PFG

Using the Clifford + RZ gate set to generate the Clifford factor group, we can now define the *Pauli frame graph* $\mathrm{PFG}_N = (\mathcal{B}_N, \mathcal{E}_N)$ on N qubits as the graph such that the Pauli frames of $\mathcal{B}_N$ represent the vertices and the edges $\mathcal{E}_N \subseteq \mathcal{B}_N \times \mathcal{B}_N$ are such that $(B_1, B_2) \in \mathcal{E}_N$ if and only if $B_2 = \mathrm{CX}_{ij}(B_1)$, $B_2 = H_i(B_1)$ or $B_2 = S_i(B_1)$ for some indices i, j, where we are implicitly using the backward gate action. As every generator is its own inverse, this is an undirected graph. Any directed path on this graph represents an overall symplectic automorphism whose action transforms its starting point to its end point. Because the symplectic group acts freely and transitively on $\mathcal{B}_N$, any two paths between two frames represent an equivalent action on $\mathcal{P}$. As we can always map a path to some Clifford circuit, *two paths connected at their end points can be mapped to two Clifford circuits which are equivalent modulo a Pauli operator.* Furthermore, the graph distance between two points is the total unweighted gate cost of the Clifford circuit. Weights can then be added to the edges of PFG_N to account for unequal gate costs as discussed below.

For our purposes, we are not looking to perform a Clifford unitary as this is not universal and can be efficiently simulated on classical computers [45]. Rather we aim to use the Clifford unitaries to achieve a sequence of rotations of the form

$$U = \prod_\alpha \exp\left(-i\frac{\theta_\alpha}{2}p_\alpha\right), \tag{25}$$

[3]A reasonable question one might have is why we do not include an extra bit in the Pauli space to represent the sign. Although this is possible in principle, this causes two problems. By definition, this would make the binary two-form λ degenerate in this extended space, and thus could not be used to form any frame, let alone a Pauli frame. The second reason is that two-qubit Clifford gates such as CX are no longer linear in this extended space. Moreover, this nonlinearity is not a practical problem for tracking the sign. See Appendix B for details.

[4]The choice of CX over CZ is not consequential as we will see in Sec. IV.

[5]There is some intuition for why this is. Consider the analogy of rotating three-dimensional (3D) vectors. Rotation as an operation on some 3D vector can be viewed as either the direct or "forward" rotation of the vector while fixing coordinates, or the inverse or "backwards" rotation of the coordinates while fixing the vector. As we shall see, the methods we describe below are analogous to the former. We think of the Pauli operators as being fixed, while Clifford operations transform the "coordinates" of the space around it.

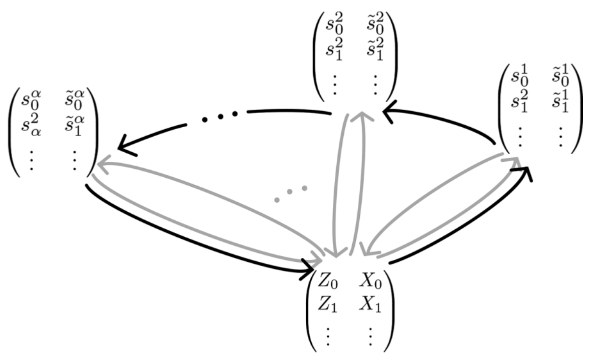
$\begin{pmatrix} s_0^\alpha & \tilde{s}_0^\alpha \\ s_\alpha^2 & \tilde{s}_1^\alpha \\ \vdots & \vdots \end{pmatrix}$
$\begin{pmatrix} s_0^2 & \tilde{s}_0^2 \\ s_1^2 & \tilde{s}_1^2 \\ \vdots & \vdots \end{pmatrix}$
$\begin{pmatrix} s_0^1 & \tilde{s}_0^1 \\ s_1^2 & \tilde{s}_1^1 \\ \vdots & \vdots \end{pmatrix}$
$\begin{pmatrix} Z_0 & X_0 \\ Z_1 & X_1 \\ \vdots & \vdots \end{pmatrix}$

FIG. 2. Simple depiction of a closed path in the PFG which might be used to realize a unitary in the PoPR form. Gray arrows represent the path generated by standard methods of compute and uncompute, whereas the black arrows represent a more direct path connecting different Pauli frames.

where $p_\alpha \in \mathcal{P}'$ is Hermitian and $\theta_\alpha \in (-\pi, \pi]$. The presentation of a unitary in this way is referred to as the *product-of-Pauli-rotations* (PoPR) form. The PoPR form is universal for computation. To perform any rotation of the form $\exp(-i\frac{\theta_\alpha}{2}p_\alpha)$, one must be in a Pauli frame such that $p_\alpha \in B_\alpha$, whereby a single-qubit operator can be used to apply the rotation. In particular,

$$\exp\left(-i\frac{\theta_\alpha}{2}p_\alpha\right) = V_\alpha^\dagger \exp\left(-i\frac{\theta_\alpha}{2}Z_{j_\alpha}\right)V_\alpha \iff Z_{j_\alpha} = V_\alpha p_\alpha V_\alpha^\dagger. \tag{26}$$

The Clifford V_α represents a Clifford automorphism $\gamma_\alpha(B_0) = B_\alpha$, and a path in PFG$_N$ such that $B_0 \to B_\alpha$ and $V_\alpha^\dagger$ represents the opposite path from $B_\alpha \to B_0$. Between the application of two adjacent angles θ_α and $\theta_{\alpha+1}$ we have the product $V_{\alpha+1}V_\alpha^\dagger$. This operator is also represented by a path in PFG$_N$ such that $B_\alpha \to B_0 \to B_{\alpha+1}$. But as any path performs almost the same Clifford unitary, we can dramatically reduce the gate cost (graph distance) by finding a direct path from $B_\alpha \to B_{\alpha+1}$, then mapping it to a Clifford unitary $V'_{\alpha\to\alpha+1}$ which must be equivalent to $V_{\alpha+1}V_\alpha^\dagger$ up to a Pauli operator p_{res} as depicted in Fig. 2. Furthermore, since $p_{\alpha+1}$ and p_{res} either commute or anticommute, we can always push p_{res} past our rotation at the cost of a sign for our rotation angle, i.e.,

$$\exp(-i\theta_{\alpha+1}p_{\alpha+1})p_{\text{res}} = p_{\text{res}}\exp(-i\sigma\theta_{\alpha+1}p_{\alpha+1}), \tag{27}$$

where $\sigma = (-1)^{\lambda(p_{\alpha+1},p_{\text{res}})}$. p_{res} can then be mapped to some other Pauli on the other side of $V'_{\alpha+1\to\alpha+2}$ and so on, in which case we can push all these Pauli operators to the end of our circuit. Thus, our total unitary is given by

$$U = p_{\text{res}}V'_{M\to 0}\left(\prod_{\alpha=1}^{M} V'_{\alpha\to\alpha+1}\exp\left(\pm i\theta_\alpha Z_{j_\alpha}\right)\right)V'_{0\to 1}, \tag{28}$$

where $p_{\text{res}} = V'_{M\to 0}(\prod_{\alpha=1}^{M} V'_{\alpha\to\alpha+1})V'_{0\to 1}$ is the *residual Pauli operator* and can be extracted if we considered the signed Pauli frame [10]. We can view the entire process as a path through PFG$_N$ which contains all B_α in order and, as it ends at B_0, forms a closed cycle.

The real power of this viewpoint is not just as a means of replacing $V_{\alpha+1}V_\alpha^\dagger$ with some $V'_{\alpha\to\alpha+1}$ as a reasonably good Clifford circuit optimization protocol can do this. Instead, we use the above to justify a flipped view on circuit synthesis. Instead of assuming the order as we have in Eq. (25), we allow the order to be permuted based on the use case. Then circuit synthesis performs a walk through PFG$_N$ where at every step, we have all members of the Pauli frame available to be applied, based on Eq. (20). Thus, the real power of this perspective is the ability to choose a more optimal ordering of the rotations, based upon the availability of rotations to be effectively applied in a given frame while simultaneously finding efficient Clifford circuits to connect them.

Although this process can be used for general purpose circuit optimization as discussed in Ref. [58], we focus on the use case where we wish to approximate a unitary $U = \exp(-iTH)$ with Hamiltonian H for a time T using the Trotter-Suzuki decomposition. We can always expand our infinitesimal Hamiltonian in the Pauli operator basis (see Sec. VI A) as

$$\delta H = \sum_{\alpha=0}^{M}\theta_\alpha p_\alpha, \tag{29}$$

where $\theta_\alpha \ll 1$. So long as the angles are small enough, error due to the order of rotations in Eq. (25) is within our error tolerance and so we are allowed to choose the order to minimize our cycle in PFG$_N$. For Trotter-Suzuki decomposition, the problem of PFG$_N$ circuit synthesis is stated as follows: *Find a cycle $\{B_\beta\}_{0\leqslant\beta<N}$ in PFG$_N$ such that for every α there exists a β such that $p_\alpha \in B_\beta$ in any order.* We refer to any cycle $\{B_\beta\}_{0\leqslant\beta<N}$ which satisfies this condition as a *Trotter cycle*. We then wish to minimize the length of the Trotter cycle in PFG$_N$.

IV. GENERAL DISCUSSION OF THE SHORTEST TROTTER CYCLE PROBLEM

When looking to solve the shortest Trotter cycle problem, we recognize the similarity to the traveling salesman problem (TSP), if we can efficiently compute the distance between Pauli frames. However, this comparison is not exact because the set of frames we have to visit in the cycle is not unique. To apply a rotation for $p \in \mathcal{P}$, any frame containing p is sufficient. So using a similar analogy as that used to describe the TSP, we imagine a new problem which we call the *traveling shopper problem* (TShP): Suppose we have a shopper with a list of items to buy. In general, no store sells all the items and many stores sell the same item. The shopper must travel to several stores, buy every item on the list, and return home. Furthermore, at any given place, the shopper can cheaply calculate the shortest distance to *a* store which sells a given item. The problem is then to find the sequence of stores with the shortest travel distance such that we buy all items and return home.

The class of problems described by TShP clearly contains TSP as we get the latter from the former by adding the promise that each item on the list is sold by a unique store. This implies that solving TShP exactly is at least NP-hard. Still, we can adapt similar heuristics as those used for TSP.

However, before we can discuss a heuristic solution to this problem, we need to be able to calculate the distance as given in the problem statement. That is, from any frame, we need to be able to efficiently calculate the distance from the current frame to any frame which contains a given Pauli operator. To do this, we are going to effectively add weights to the edges of PFG_N. We assume that for NISQ devices, single-qubit Clifford operations are essentially free and so edges associated to H and P gates have zero weight. Furthermore, we shall treat SWAP gates as effectively having zero weight since we do not care on which qubit we apply the RZ gate for a given rotation, i.e., we can effectively virtualize these SWAP operations. Thus, we define the nonentangling automorphisms as $\not{E}_N = \langle \mathrm{SWAP}_{ij}, H_i, S_i \rangle_{0\leqslant i,j<N}$ and a set of equally entangled frames (EEF) about B as those connected by a member of $\not{E}_N$, i.e.,

$$[B] = \{B' \in \mathcal{B}_N | B' = \gamma(B), \text{ for some } \gamma \in \not{E}_N\}. \tag{30}$$

As $\not{E}_N$ is a subgroup of $\mathrm{Sp}(2N, \mathbb{F}_2)$, the set of all EEFs represents a partitioning of PFG_N into subgraphs, and each of these subgraphs are connected by some number of CX. We can then define the coarse-grained graph $[\mathrm{PFG}_N]$ as the quotient graphs with respect to the EEF partitioning, i.e., two coarse-grained vertices $[B_1]$ and $[B_2]$ are connected in $[\mathrm{PFG}_N]$ if and only if there exists members $B_1' \in [B_1]$ and $B_2' \in [B_2]$ connected by a CX.

So by finding a Trotter cycle in $[\mathrm{PFG}_N]$ instead of PFG_N, we have effectively imposed our weight function on gates in the gate set. This also justifies our use of the *relative support*, $\mathrm{Supp} : \mathcal{P} \times \mathcal{B} \to \mathbb{N}$,

$$(d, B) \mapsto \mathrm{Supp}(p, B) = \sum_i \lambda(s_i, p) \vee \lambda(\tilde{s}_i, p), \tag{31}$$

where $B = \{(s_i, \tilde{s}_i)\}_{0\leqslant i<N}$ and $\vee$ is the logical OR. This is a generalization to the qubit support of a given Pauli operator, where here we are giving the support of the Pauli p relative to the frame B. We argue that relative support is exactly the “distance”[6] in $[\mathrm{PFG}_N]$ we are interested in as described in TShP. In particular, it has the following properties:

Proposition 1. The relative support satisfies the following properties for all $p, q \in \mathcal{P}$ and $B \in \mathcal{B}$:

(1) $\mathrm{Supp}(p, B) \geqslant 0$ and $\mathrm{Supp}(p, B) = 0$ if and only if $p = I$.

(2) $\mathrm{Supp}(p + q, B) \leqslant \mathrm{Supp}(p, B) + \mathrm{Supp}(q, B)$.

(3) $\mathrm{Supp}(p, \gamma(B)) = \mathrm{Supp}(p, B)$ for all $\gamma \in \not{E}_N$.

(4) $\mathrm{Supp}(p, B) - 1$ is the minimum distance in $[\mathrm{PFG}_n]$ to reach another frame B' which contains p.

We prove these properties in Appendix C. In general, calculating $\mathrm{Supp}(p, B)$ scales as $O(N^2)$ as calculate λ is $O(N)$. However, as we discuss below, we do not need to calculate the relative support directly for each step of the search, but rather we can update the binary expansion of a Pauli operator in the current frame, in which case calculating the relative support is $O(N)$. We shall use this in the ultragreedy heuristic discussed below.

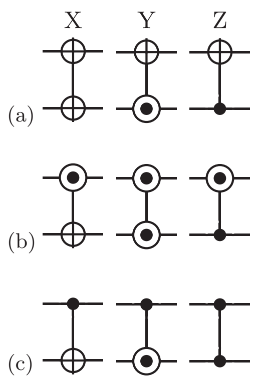

FIG. 3. Table of the nine 2qE gates.

1. Two-qubit entangling gates

Even though $\not{E}_N$ is a subgroup of $\mathrm{Sp}(2N, \mathbb{F}_2)$ and ultimately represents a subgroup of the Clifford group, it is not a normal subgroup and, as a consequence, CX_{ij} acting on members $B, B' \in [B]$ of the same EEF may be taken to distinct EEFs. To enumerate the number of distinct two-qubit entangling (TQE) gates, we start by considering only two qubits. Any TQE gate can then be decomposed as some member of $\not{E}_2$ followed by a CX gate and then some other member of $\not{E}_2$. However, any members of $\not{E}_2$ after the CX does not change the EEF and we can mod such operations out in order to count distinct gates which connect different EEFs. Because of this, we can always push a SWAP to the right of the CX and thus we only need to consider pairs of single-qubit operators acting before the CX. Each qubit has six distinct single-qubit Clifford gates (modding by single-qubit Pauli operators), I, H, S, HS, SH, and $K \equiv HSH$. Note K is the phase gate for the $\pm$ basis. However, as S commutes with the control and K commutes with the target, we can always push such operators to the right and mod them out, thus cutting the number of relevant single-qubit operators in half for each qubit. This leaves $3^2 = 9$ distinct TQE gates for each qubit pair which can be organized into an array as shown in Fig. 3. Generalizing this to multiple qubits, this implies the coordination number for each vertex of $[\mathrm{PFG}_N]$ is $\frac{9}{2}N(N-1)$, i.e., nine TQE gates for every qubit pair. Using CX as the example, we use $-\oplus-$ to represent the X-type Pauli operator and $-\bullet- = -\boxed{H}\oplus\boxed{H}-$ to represent the Z-type Pauli operator. Similarly, we introduce the notation of $-\odot- = -\boxed{S^\dagger}\oplus\boxed{S}-$ to represent the Y-type Pauli operator. These definitions then extend to the TQE gates. As for the names of the TQE gates, we use the convention of labeling the rows by A, B, C and the columns by X, Y, Z as shown in Fig. 3. So, for example, CX_{01}, CY_{01}, and CZ_{01} are the usual control-X, control-Y, and control-Z operations while $\mathrm{AZ}_{0,1}$ and BZ_{01} are the first two gates in the opposite direction. By convention, we also always order the qubit indices from lowest to highest. We shall refer to these row and column labels as the type similar to that of the Pauli type, i.e., CX is of Z type on the first qubit and

[6]It should be noted that Supp is not a distance function in the usual sense for one major reason: the two entries of its domain are not in the same space. Still, Proposition 1 provides all the analogous properties to an actual distance function.

X type on the second qubit; BY is of Y type on both qubits. To demonstrate that the nine operators in Fig. 3 are distinct edges in $[\mathrm{PFG}_n]$, consider acting on the two-qubit origin frame with CX:

$$\begin{pmatrix} Z_0 & X_0 \\ Z_1 & X_1 \end{pmatrix} \xrightarrow{\text{CX}} \begin{pmatrix} Z_0 & X_0X_1 \\ Z_0Z_1 & X_1 \end{pmatrix}. \tag{32}$$

We then consider acting on the origin frame with CZ instead:

$$\begin{pmatrix} Z_0 & X_0 \\ Z_1 & X_1 \end{pmatrix} \xrightarrow{\text{CZ}} \begin{pmatrix} Z_0 & X_0Z_1 \\ Z_1 & Z_0X_1 \end{pmatrix}. \tag{33}$$

Now if the two resulting frames were in the same EEF, then any member of $\text{CZ}(B_0)$, say Z_1 should be such that $\text{Supp}(Z_1, \text{CX}(B_0)) = 1$, but in fact $\text{Supp}(Z_1, \text{CX}(B_0)) = 2$, so these two frames can not be in the same EEF. A similar method can be used to show the other seven TQE operators are also distinct from each other as well as CX and CZ.

V. OVERVIEW OF THE ULTRAGREEDY HEURISTIC FOR FINDING A HIGH-EFFICIENCY TROTTER CYCLE

We now look to use our knowledge of $[\mathrm{PFG}_n]$ to develop a simple heuristic for finding a Trotter cycle. The simplest heuristic one can apply to such a problem is that of a greedy strategy where at each step, we target one of the nearest Pauli operators in our list of rotations to apply. However, due to the possibility of multiple Pauli operators with the same minimum relative support and nonuniqueness of a frame which can achieve any such rotations, we need to consider how we make decisions in our greedy approach to moving through $[\mathrm{PFG}_N]$. In keeping with the simplest possible heuristic, we also use a greedy method to choose between these possible paths, thus we refer to this strategy as the *ultragreedy* heuristic which is outlined as follows.

For a given step, of the remaining terms, we find those Pauli operators with the minimum relative support greater than 1. We can calculate this cheaply as it is given by the support of our stored binary vectors. For each of these minimum-distant Paulis, we take every pair of qubits on which it is supported relative to the frame. The support on two qubits consists of four classical bits and there must be nonzero support on each of the two qubits. This leaves only nine possible configurations of the four bits, and for each of these, we argue in Appendix B that only four of the nine TQE gates can reduce the support by one. This gives us a list of possible gates to apply which will always reduce the support of at least one of these minimum-distance Pauli operators by one. Of these, we need some method of deciding which to apply. We consider two possibly competing considerations. The first is how the gate affects the support of the other remaining terms. So, of the remaining terms (including the minimum distant terms) we can calculate the average change in support for the remaining terms after the gate is applied and use this as part of a cost metric. The second consideration is choosing a gate which minimizes the circuit depth. This requires that we have a rough schedule for the gates. Assume all gates take the same amount of time, we keep track of the time of the last gate action for each qubit using ASAP scheduling. For a given gate under consideration, we schedule that gate and find the difference between the scheduled time and the latest scheduled gate (leading temporal edge of the circuit so far). We refer to this difference as the "pace" of the scheduled gate. So the cost metric for our possible gate g is

$$\text{cost}_c(g) = \langle \Delta\text{Supp}_g \rangle - c|\text{pace}(g)|, \tag{34}$$

where $\langle \Delta\text{Supp}_g \rangle$ is the average change in support of the remaining Paulis after applying g and $c > 0$ is a free parameter of the method we refer to as the *parallelization credit*. We then choose the gate which minimizes this cost metric. Once a gate is chosen, it is added to the output circuit and all binary vectors are updated. Additionally, the rotations are added to the output circuit at the top of this procedure whenever a Pauli operator with unit support is found. In such a case, we then add an RZ to the circuit for a local qubit support of (0,1), RX for (1,0), and RY for (1,1), at which time the term is removed from the list of terms to be applied. The algorithm proceeds until all rotations have been applied. The search necessarily halts as every TQE gate we choose reduces the support of at least one term by one, thus always bring us closer to terminating. A more detailed outline of the ultragreedy search heuristic is given in Appendix D.

A. Completing a Trotter cycle via retracing

The ultragreedy search only considers synthesizing a single Trotter step which must complete the Trotter cycle by returning to the origin frame. However, when applying several Trotter steps, this is not necessary. Once our greedy search method finds a complete *Trotter path*, i.e., a path through PFG such that every term appears in at least one frame but does not return to the origin, we are now free to start the next Trotter step in the ending frame. One could restart the search, but it seems reasonable that an efficient path for the next Trotter step is to simply apply the same circuit of the previous step in reverse order, a method we refer to as *retracing* as we are simply retracing our steps in the PFG.[7] The advantage of retrace is twofold: First, we avoid the costly origin return path, which can often be a serious downside to greedy search. Second, we automatically obtain the second-order Trotter-Suzuki decomposition, increasing the accuracy of the circuit in approximating the desired unitary. If retracing is undesired, one can also synthesize a return path using Clifford synthesis [44,45], where Ref. [10] uses methods very similar to ones used here for this purpose.

B. Time scaling for PFG ultragreedy search

To assess the performance of the ultragreedy search heuristic, we consider the expected time scaling with respect to number of qubits N and number of Pauli terms in the Hamiltonian, No.(H). In the worst case, the algorithm takes at most N iterations through the main loop to apply at least one rotation. However, in practice we expect the number of iterations to be closer to a constant, in which case the total number of iterations of this main loop should scale as the number of terms. Within that loop, we have two parts: finding the minimum

[7]Note that retracing does not undo the work done by the former step because we are not reversing the signs of the rotation angles.

distance terms while simultaneously looking for rotations to apply, and finding the minimum cost TQE gate. In general, finding the minimum cost gate is more costly. Worst case, we need to consider every possible TQE gate of which there are $O(N^2)$. We then need to evaluate the change in support for every remaining term which would require $O[N\ \mathrm{No.}(H)]$, but since each gate can only change the support locally, we can use this to reduce this time cost to $O[\mathrm{No.}(H)]$. Thus, we expect the time scaling to be closer to $O[N^2\ \mathrm{No.}(H)^2]$ with a worst case scaling of $O[N^3\ \mathrm{No.}(H)^2)$. This process is also amenable to parallelization to reduce this time. Details of parallelizing the algorithm can be found in Appendix E.

VI. RESULTS FOR THE PFG ULTRAGREEDY SEARCH ALGORITHM FOR FINDING A TROTTER PATH

In this section, we demonstrate the use of the PFG ultragreedy algorithm on four physical models of real-world interest. These models were chosen for their natural scaling behavior.

A. Physical examples and their Hamiltonian mapping to qubits

In order to study technologically useful or scientifically interesting problems in physics, chemistry, and materials science, it is often necessary to simulate a quantum Hamiltonian. Here we describe the fermionic and bosonic Hamiltonians for which we synthesize a circuit to produce an effective single Trotter step. For each type of particle, we consider a model with a low scaling exponent for the number of Pauli operator terms in the Hamiltonian, what we refer to as a “low-density” model, as well a model with a larger scaling exponent, what we refer to as a “high-density” model. The Fermi-Hubbard and Bose-Hubbard models are our low-density examples. Our high-density bosonic example is the vibronic problem from chemistry. Finally, our high-density fermionic instance is for the molecular electronic structure problem of polyacetylene (with varying chain length). Fermionic Hamiltonians were produced with the help of OPENFERMION [59]. For the bosonic degrees of freedom, we considered two different encodings to qubits: standard binary and the Gray code [60]. We considered the two cases $d = 4, 8$ for the level truncation for the bosonic model.

1. *Fermi-Hubbard model*

The Fermi-Hubbard model [15] is defined as

$$H_{\mathrm{FH}} = -t\sum_{ij\sigma}(a^\dagger_{i\sigma}a_{j\sigma} + a^\dagger_{j\sigma}a_{i\sigma}) + U\sum_{i} n_{i\uparrow}n_{j\uparrow}, \tag{35}$$

where $a^\dagger_{i\sigma}$ and $a_{i\sigma}$ are, respectively, fermionic creation and annihilation operators for site i and spin $\sigma \in \{\uparrow, \downarrow\}$, t is the coupling term, and U is the repulsion term. We then used Jordan-Wigner [19] and Bravyi-Kitaev [61,62] methods for mapping fermionic degrees of freedom to qubit degrees of freedom, with even numbers of 2 through 100 sites (for a maximum of 200 qubits), with periodic boundary conditions.

The scaling of terms for this model is No. $(H_{\mathrm{FH}}) \sim O(N)$ where N is the number of qubits.

2. *Polyacetylene*

In second quantized form, the electronic structure Hamiltonian [63–66] can be written

$$H_{PAc} = \sum_{ij} h_{ij}a^\dagger_i a_p + \sum_{ijpq} h_{ijpq}a^\dagger_i a^\dagger_j a_p a_q, \tag{36}$$

where spin orbitals are labeled by $\{i, j, p, q\}$, h_{ij}, and h_{ijpq} are one- and two-electron integrals, and an arbitrary number of couplings may be present.

In order to gain insight into scaling of the ultragreedy search algorithm for molecular electronic structure, we prepared Hamiltonians for a simplified model of cis-polyacetylene. For 2 through 20 carbon atoms, we prepared molecules $C_{n_\mathrm{C}}H_{n_\mathrm{C}+2}$ for even numbers of carbon atoms n_C, and the $C_{n_\mathrm{C}}H^-_{n_\mathrm{C}+2}$ anion for odd n_C. Using anions for odd n_C ensures that none of the molecules have radical electrons, which would lead to a significant qualitative change on the molecular properties. Approximate geometries were found using the software AVOGARDO 1.2 [67] by optimizing with the universal force field (UFF) [68]. We used the minimal STO-3G basis set to perform Hartree-Fock calculations in PSI4 [69] and OPENFERMION2PSI4 [59]. The active space was chosen by filling the lowest $4n_\mathrm{C}$ spin orbitals in the canonical orbital basis, and removing the $4n_\mathrm{C}$ highest-energy spin orbitals, leaving an active space of $4(n_\mathrm{C} + 1)$ spin orbitals. All terms smaller than 10^{-6} Hartrees in the second quantized Hamiltonian were removed. The Jordan-Wigner and Bravyi-Kitaev transformations were performed. The purpose of this data set is to investigate scaling for the Trotterization algorithm on molecular-related circuits: for truly accurate calculations, different choices would be made regarding geometry optimization, basis size, active space selection, and truncation of small terms.

The scaling of terms for this model is $\mathrm{No.}(H_{PAc}) \sim O(N^4)$, where N is the number of qubits used to encode the spin orbitals.

3. *Bose-Hubbard model*

The Bose-Hubbard model [70,71] is defined as

$$H_{\mathrm{BH}} = -t\sum_{ij}^{N} h_{ij}b^\dagger_i b_j + U\sum_{i}^{N} b^\dagger_i b_i(b^\dagger_i b_i - 1), \tag{37}$$

where t is a coupling term, U is the single-site interaction term, and $b^\dagger_i$ and b_i are, respectively, bosonic creation and annihilation operators for particle i. We used bosonic cutoffs $d = 4$ and 8, as well as the two bosonic encodings, standard binary (Std) and Gray code (Gray), and even site numbers N from 2 through 100.

The scaling of terms for this model is $\mathrm{No.}(H_{\mathrm{BH}}) \sim O(N)$ where N is the number of qubits.

4. *Vibronic Hamiltonian*

Vibronic (vibrational + electronic) transitions are ubiquitous in chemistry, occurring for example during absorption of light. Because distinct electronic potential energy surfaces (PESs) do not yield identical vibrational normal modes, there is a mixing of modes during the transition between surfaces, with each transition having a distinct intensity (known as a

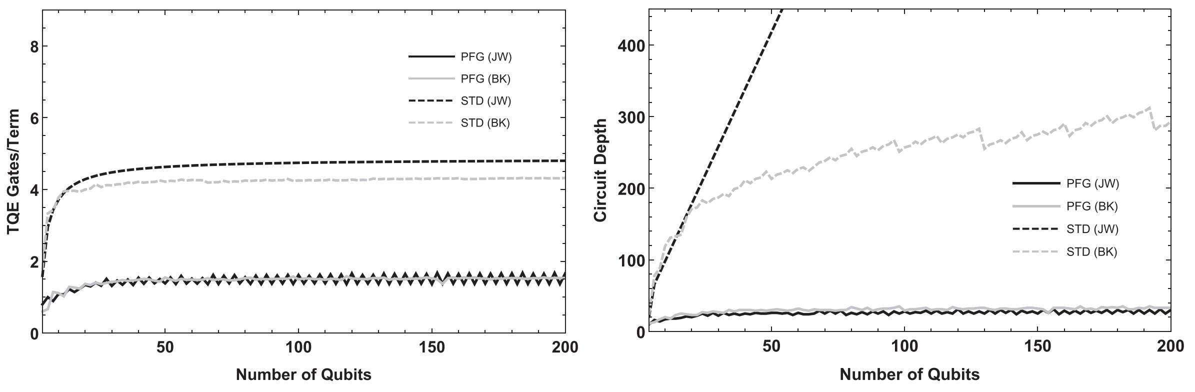

FIG. 4. Plots for the results of synthesizing an effective Trotter step for the Fermi-Hubbard model with the Jordan-Wigner (JW) and Bravyi-Kitaev (BK) fermionic encodings using the "standard" method (STD) and the method of this paper (PFG). The left plot is for number of TQE gates per term and the right plot is for circuit depth. In all cases, PFG outperforms STD.

Franck-Condon factor). The mixing of normal modes is determined by the unitary Duschinsky mixing matrix, denoted $\mathbf{S}$, and the frequencies ω_{sj} of mode j on surface s. Here we summarize the final Hamiltonian used in this work; more details for computations of this class can be found in [22–24,72]. The first step is to express the vibrational modes of surface B in the normal mode basis of surface A. This is done with the transformations

$$\vec{q}_B = [q_{B0}, q_{B1}, q_{B2}, \dots]^T = \mathbf{\Omega_B S \Omega_A^{-1}} \vec{q}_A + \vec{\delta}, \tag{38}$$

$$\vec{p}_B = [p_{B0}, p_{B1}, p_{B2}, \dots]^T = \mathbf{\Omega_B^{-1} S \Omega_A} \vec{p}_A, \tag{39}$$

where $\vec{q}_s$ and $\vec{p}_s$ are, respectively, position and momentum operators for surface s and $\mathbf{\Omega}_s = \text{diag}([\omega_{s1}, \dots, \omega_{sM}])^{\frac{1}{2}}$. Note that $\vec{q}_B$ and $\vec{p}_B$ are vectors *of operators*, not vectors of scalars.

The Hamiltonian is then a sum of independent Harmonic oscillators:

$$\hat{H}_{\text{FC}} = \frac{1}{2} \sum_j^M \omega_{Bj} \left(q_{Bj}^2 + p_{Bj}^2 \right), \tag{40}$$

where M is the number of vibrational modes. For implementation in a quantum device, $\vec{q}_B$ and $\vec{p}_B$ are expressed in terms of $\vec{q}_A$ and $\vec{p}_A$.

We encoded vibronic Hamiltonians of a fully dense $\mathbf{S}$ where all ω_{Ak} and ω_{Bk} are unique. This provides an upper bound to the resource count required for a vibronic problem of M modes; in real molecules, $\mathbf{S}$ often has sparsity or additional structure that can be exploited. Hence, each Hamiltonian is parametrized by the encoding type, the number of modes M, and the number of levels (d) allowed in the boson. We used two encodings (Gray and Std), $M = 6$ through 102, and bosonic cutoffs of $d = 4$ and 8.

The scaling of terms for this model is No.$(H_{\text{FC}}) \sim O(N^2)$ where N is the number of qubits used to encode the vibrational modes.

B. Results

The models described above were synthesized from the PoPR form to a circuit of TQE gates and single-qubit rotations using the ultragreedy PFG algorithm described above with a parallelization credit of $c = 0.1$, as well as more "standard" (STD; not to be confused with the Std encodings) methods given by decomposing each Pauli rotation using a CX staircase and appropriate single-qubit Clifford gates and applying a Clifford circuit optimization scheme to the resulting circuit.

For the STD method, in order to reduce circuit depth, the Pauli terms are first placed in sets for which all terms commute within each set. Next, each term is Trotterized using the well-known CNOT staircase construction. The third step is to map the resulting quantum circuit onto a graph, where each gate is a node and edges are present when two gates are adjacent on the same qubit. Finally, we loop through the vertices of this graph, canceling out adjacent gates where possible. A time limit of 600 s was set on this last step, meaning that there may be some remaining gate cancellations that are neglected.

Figures 4–7 each plot the resulting average number of TQE gates needed per Pauli Hamiltonian term and circuit depth for one effective Trotter step[8] as a function of qubit number. Due to the parallelization and overall efficiency of the ultragreedy PFG algorithm implementation, the PFG methods were able to synthesize much larger examples, in which case insets for some of these plots are a blown-up section of the graph where both methods produced results.

In general, one can see that the PFG method always results in fewer TQE gates per term and in almost every case a lower depth over standard methods, and often significantly so. In the case of the polyacetylene model, the reduction is as much as an order of magnitude in the largest examples where both methods returned a result. We do note that in the case of the bosonic models, it appears possible in the case of the Bose-Hubbard model and does in the vibronic model that standard methods overtake our PFG methods when considering depth. This may be an indication of the limitations of greedy search. Looking at the circuits for smaller instances, one can see that while in the beginning the circuits are highly dense, rotations generally begin to rarefy causing ever-longer TQE gate chains in order to reach some member of the dwindling pool of unapplied rotations. This is to be expected of a greedy method and demonstrates the need for more sophisticated PFG methods for finding a Trotter path. Although we do not see it here, it

[8]This does not include the return path as we assume retracing.

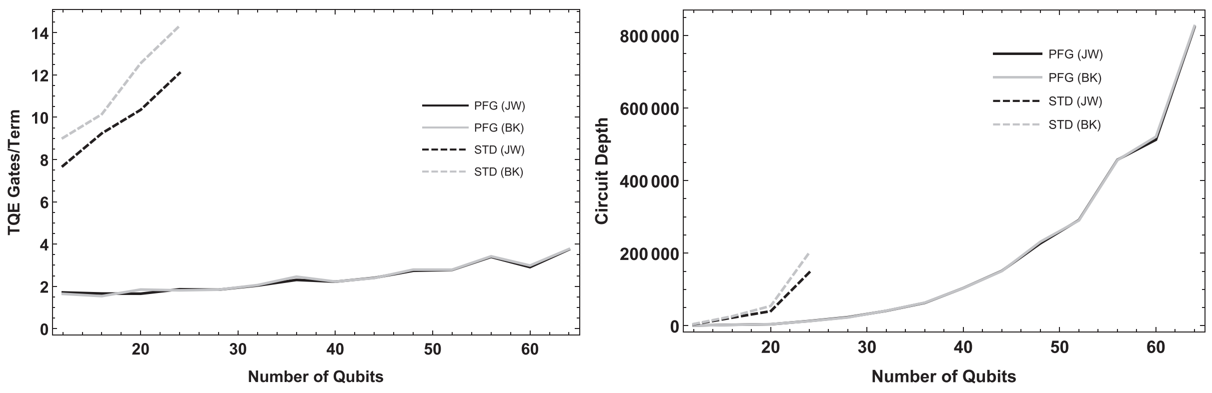

FIG. 5. Plots for the results of synthesizing an effective Trotter step for the polyacetylene model with the Jordan-Wigner (JW) and Bravyi-Kitaev (BK) fermionic encodings using the "standard" method (STD) and the method of this paper (PFG). The left plot is for number of TQE gates per term and the right plot is for circuit depth. In all cases, PFG outperforms STD.

can happen that our methods generate more TQE gates than standard methods, but result in a lower depth. Again this is due in part to the limitations inherent in greedy search, but also attributable to the addition of the parallelization credit in Eq. (34). Such considerations allow the method to add more TQE gates if doing so results in a lower depth. Multiple runs with different values for the parallelization credit were not performed for this study, though it was shown in Ref. [10] that although a different context, varying the parallelization credit can significantly reduce circuit metrics.

Figure 8 shows the CPU time to complete the PFG ultragreedy search as a function of (N) No.(H) on a log-log plot for each of the models described here. The PFG methods utilized as many 100 parallel processes for some of the larger problem instances presented in this paper, some of which were multithreaded processes. In every case we see an expected scaling exponent between 1 and 2. Table I gives the fitted scaling exponent. Here we see that the scaling exponent does appear to have some dependency on the density of the model. This could be due in part to the use of parallelization for larger models which comes with considerable communication overhead. See Appendix E for a description of this. This is also consistent with the fact that the $d = 4$ encodings have a generally smaller exponent than $d = 8$ for bosonic models,

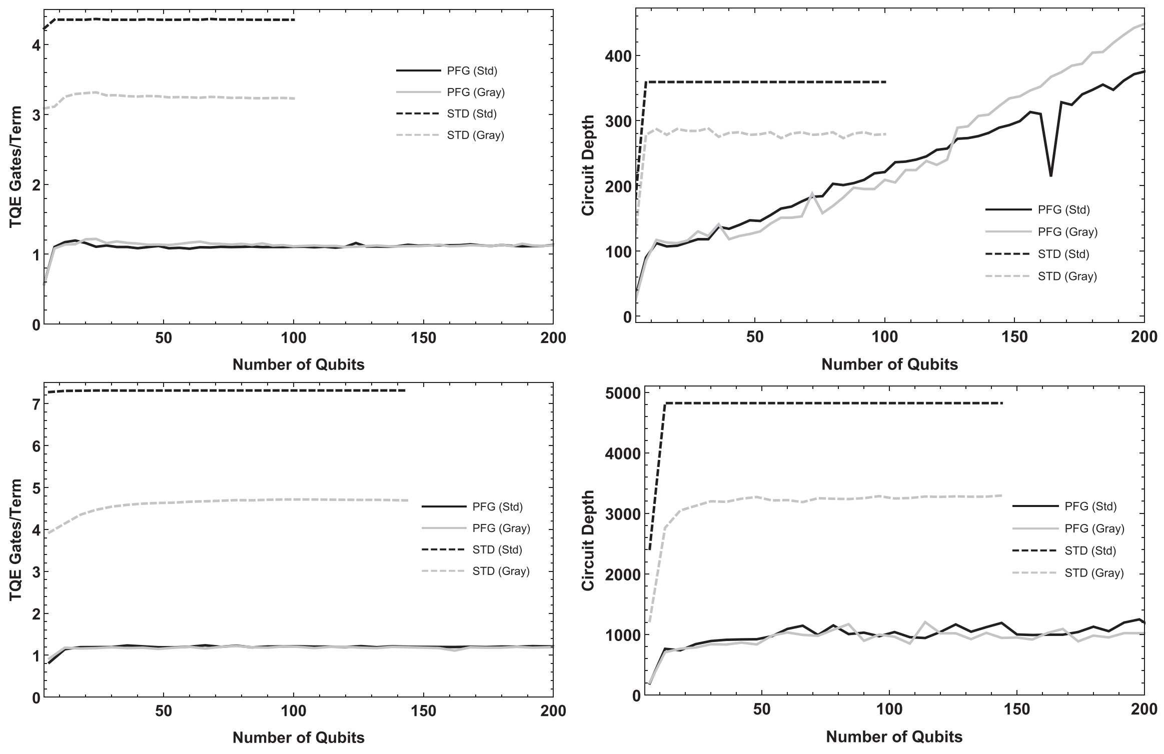

FIG. 6. Plots for the results of synthesizing an effective Trotter step for the Bose-Hubbard model with the Gray code (Gray) and standard (Std) bosonic encodings using the "standard" method (STD) and the method of this paper (PFG). The left two plots are for number of TQE gates per term and the right two plots are for circuit depth. The top two plots are for $d = 4$ and bottom two plots are for $d = 8$. In all cases, PFG outperforms STD.

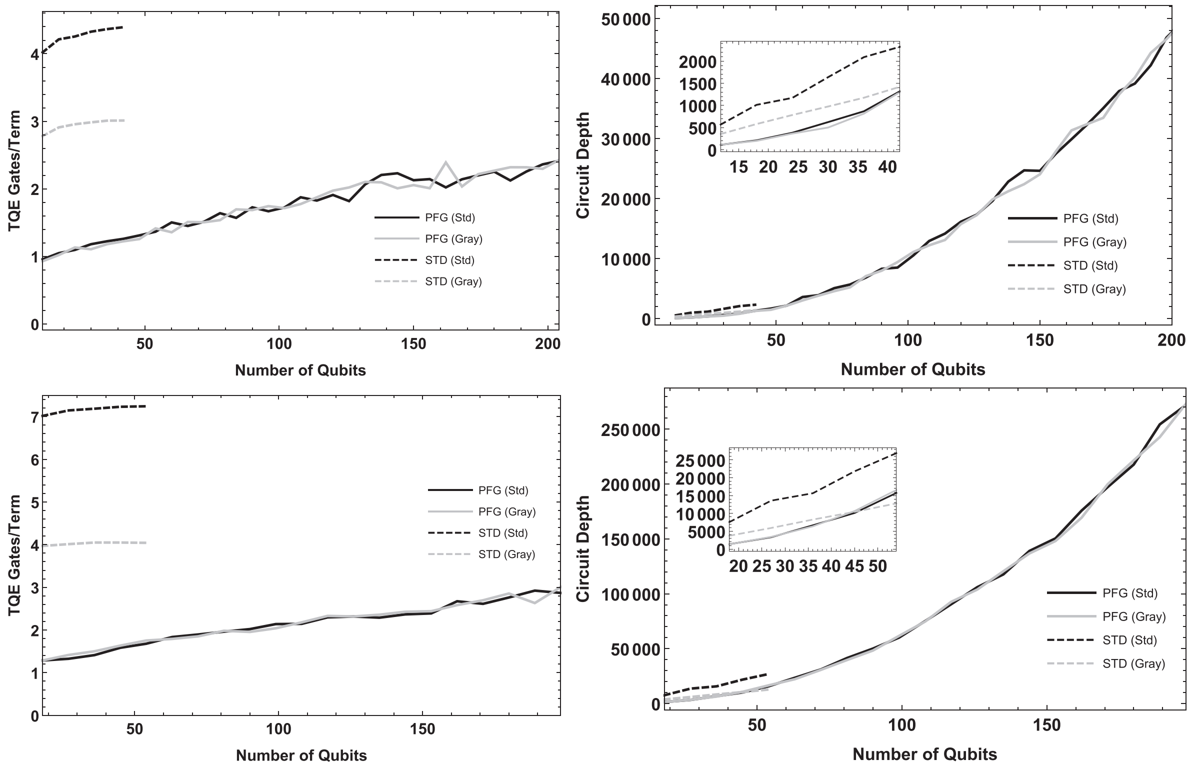

FIG. 7. Plots for the results of synthesizing an effective Trotter step for the vibronic model with the Gray code (Gray) and standard (Std) bosonic encodings using the "standard" method (STD) and the method of this paper (PFG). The insets show a zoomed-in section where both methods produce results. The left two plots are for number of TQE gates per term and the right two plots are for circuit depth. The top two plots are for $d = 4$ and bottom two plots are for $d = 8$. In nearly all cases, PFG outperforms STD.

which is true for both low- and high-density models. Overall, this study of the time scaling of the algorithm demonstrates the PFG ultragreedy search algorithm is both efficient and scalable, as also demonstrated by the large problem instances for which the method was able to produce a circuit.

To be fair to the standard methods, the code used was not optimized or parallelized to the same extent as the PFG ultragreedy search code. Furthermore, arbitrary timeout limits were set for practical reasons due to the large data set studied. Thus, we do not attempt to compare time scaling between the two methods.

VII. OUTLOOK AND CONCLUSIONS

In this paper, we have motivated and described the theory behind the Pauli frame graph and how it provides a perspective on synthesizing circuits from the product-of-Pauli rotations form. From there, we focused on the use case which synthesizes a Trotter step in the Trotter-Suzuki decomposition for the simulation of a Hamiltonian often used for applications in physics, material science, and chemistry as well as subroutines in many common quantum algorithms. Using this perspective, we also developed the ultragreedy search algorithm for synthesizing an efficient Trotter step, especially when using the retrace method. We then demonstrated the power of the algorithm by applying the method to four models, two of which were fermionic, two bosonic, two low density, and two high density. In nearly every case, the results of our algorithm produced circuits with significantly lower depth over the more conventional method and fewer two-qubit entangling gates. We also showed that this algorithm is both scalable and efficient.

Although we have found a reasonable circuit synthesis algorithm here, the real power of this work is in the paths it opens for future research. These methods have already been refined and adapted for general circuit synthesis [10] and measurement reduction for a mutually commuting set of Pauli operators [73] as a part of the Pauli-based circuit optimization, analysis, and synthesis toolchain (PCOAST). This includes future work in PCOAST to adapt all such use cases to include limited qubit connectivity. Moreover, the ultragreedy methodology is the simplest, least sophisticated method for using the PFG perspective of circuit synthesis. Thus, we look to develop more sophisticated synthesis methods in the future which balance both circuit optimization as well as overall unitary accuracy, such that we reduce the total circuit cost for the entire Trotterized simulation. Finally, we look to add this functionality to be used by the Intel® Quantum SDK [74] with extensions for algorithm abstraction such as Hamiltonian simulation.

ACKNOWLEDGMENTS

This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231 using NERSC Award No. DDR-ERCAP0018781.

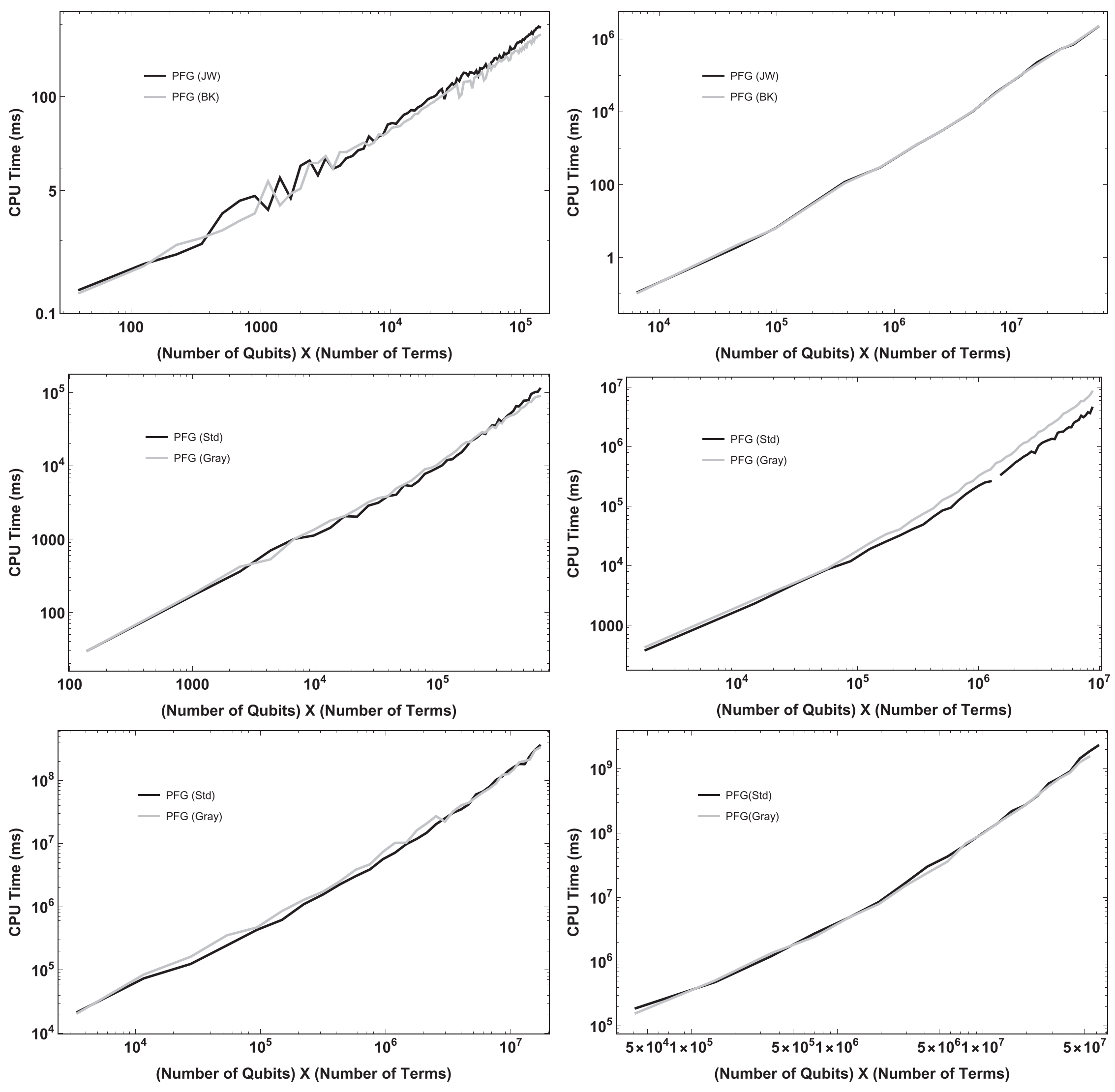

FIG. 8. Log-log plots of CPU time (wall-clock time times number of CPU processes used) as a function of number of qubits times number of Pauli Hamiltonian terms. Each plot is for the model with the Jordan-Wigner (JW) and Bravyi-Kitaev (BK) fermionic encodings or the Gray code (Gray) and standard (Std) bosonic encodings where applicable using the method of this paper (PFG). The linearity of these plots demonstrates the PFG method herein scales polynomially in the size of problem. The approximate slope of these lines can be found in Table I. Each plot pertain is for a given model such that: top left is for Fermi-Hubbard, top right is for polyacetylene, left middle is for Bose-Hubbard with $d = 4$, right middle is for Bose-Hubbard with $d = 8$, bottom left is for vibronic with $d = 4$ and bottom right is for vibronic with $d = 8$.

APPENDIX A: PROOF OF FREE AND TRANSITIVE GROUP PROPERTIES FOR THE SYMPLECTIC GROUP ACTING ON PAULI FRAMES

In this Appendix, we prove that the group action of the symplectic group on Pauli frames is free and transitive. See main text for definitions.

Proof. To prove transitivity, we must show that for frames $B_1 = \{(s_i, \tilde{s}_i)\}_{0\leqslant i<N}$, $B_2 = \{(t_i, \tilde{t}_i)\}_{0\leqslant i<N}$, there exists a symplectic automorphism $(\gamma : \mathcal{P} \to \mathcal{P}) \in \mathrm{Sp}(2N, \mathbb{F}_2)$ such that $\gamma(B_1) = B_2$. To this end, we consider the map γ such that

$$p \mapsto \sum_i [\lambda(p, s_i)\tilde{t}_i + \lambda(p, \tilde{s}_i)t_i]. \tag{A1}$$

Clearly this is linear. Also, for any $p, q \in \mathcal{P}$ we have that

$$\begin{aligned}\lambda(\gamma(p), \gamma(q)) &= \sum_i [\lambda(p, s_i)\lambda(\tilde{t}_i, \gamma(q)) \\ &\quad + \lambda(p, \tilde{s}_i)\lambda(t_i, \gamma(q))] \\ &= \sum_i [\lambda(p, s_i)\lambda(\tilde{s}_i, q) + \lambda(p, \tilde{s}_i)\lambda(s_i, q)] \\ &= \lambda(p, q).\end{aligned} \tag{A2}$$

Thus, $\gamma \in \mathrm{Sp}(2N, \mathbb{F}_2)$. Finally, if we apply γ to any member of B_1, we have $\gamma(s_i) = t_i$ and $\gamma(\tilde{s}_i) = \tilde{t}_i$ for all i. Therefore, $\gamma(B_1) = B_2$ and our group action is transitive.

TABLE I. Table of scaling exponent for the time scaling of the PFG ultragreedy search algorithm as a function of number of qubits times number of Pauli Hamiltonian terms, i.e., (CPU time to complete search) $\sim [N \text{ No. } (H)]^{\alpha}$.

Model	α(JW/Std)	α(BK/Gray)
Fermi-Hubbard	1.128	1.180
Polyacetylene	1.672	1.814
Bose-Hubbard $d = 4$	1.324	1.122
BoseHubbard $d = 8$	1.437	1.523
Vibronic $d = 4$	1.583	1.524
Vibronic $d = 8$	1.846	1.698

To show that this group action is free, we must show that if for some $\gamma, \delta \in \mathrm{Sp}(2N, \mathbb{F}_2)$ there exists a B such that $\gamma(B) = \delta(B)$, then $\gamma = \delta$. So suppose we do have such a Pauli frame for some γ and δ. As $\gamma(B) = \delta(B)$ is a basis for $\mathcal{P}$, by linearity for every $p \in \mathcal{P}$,

$$\begin{aligned}\gamma(p) &= \sum_i [\lambda(p, s_i)\gamma(\tilde{s}_i) + \lambda(p, \tilde{s}_i)\gamma(s_i)]\\ &= \sum_i [\lambda(p, s_i)\delta(\tilde{s}_i) + \lambda(p, \tilde{s}_i)\delta(s_i))]\\ &= \delta(p).\end{aligned} \tag{A3}$$

Therefore, $\gamma = \delta$ and our group action is free. ■

APPENDIX B: DERIVING THE ACTION AND SIGNS FOR CLIFFORD GATES

In this Appendix, we discuss the general methods for determining the transformation and sign of Pauli operators and Pauli frame elements under the action of one- and two-qubit Clifford gates. We do this by first defining two general Pauli-based Clifford gates. For Hermitian $p \in \mathcal{P}'$ we define the *root-of* gate

$$\sqrt{p} = \exp\left(-i\frac{\pi}{4}p\right). \tag{B1}$$

This generalizes the phase gate for any given Pauli operator, i.e., $\mathrm{S} = \sqrt{Z}$. It should be clear that the conjugation rule for any other $q \in \mathcal{P}'$ is

$$\sqrt{p}q\sqrt{p}^{\dagger} = \begin{cases} p, & \lambda(p, q) = 0\\ -ipq, & \lambda(p, q) = 1.\end{cases} \tag{B2}$$

For two Hermitian, mutually commuting, Pauli operators $p, q \in \mathcal{P}'$, we define the AND-Sign (ASIGN) gate

$$\begin{aligned}\mathrm{ASIGN}(p, q) &= \exp\left(-i\frac{\pi}{4}(1-p)(1-q)\right)\\ &\propto \sqrt{p}^{\dagger}\sqrt{pq}\sqrt{q}^{\dagger}.\end{aligned} \tag{B3}$$

This gate applies a minus sign to eigenstates for which p and q evaluate to -1. This generalizes the TQE gates, i.e., $\mathrm{CX}_{01} = \mathrm{ASIGN}(Z_0, X_1)$. Using Eq. (B2), we can see the conjugation rule for any $r \in \mathcal{P}'$ is

$$\begin{aligned}&\mathrm{ASIGN}(p, q)r\mathrm{ASIGN}(p, q)\\ &= \begin{cases} r, & \lambda(p, r) = \lambda(q, r) = 0\\ qr, & \lambda(p, r) = 1, \lambda(q, r) = 0\\ pr, & \lambda(p, r) = 0, \lambda(q, r) = 1\\ -pqr, & \lambda(p, r) = \lambda(q, r) = 1.\end{cases}\end{aligned} \tag{B4}$$

To derive the gate action on Pauli operators and Pauli frames, one recognizes that all two-qubit gate action, both forward and backward, can be reduced to these conjugation rules, if one implements sign and phase-aware Pauli multiplication.

In the case of forward action for two-qubit entangling (TQE) gates, transformation of the Pauli frame is just conjugating each Pauli operator entry. For TQE gate $g_{ij} = \mathrm{ASIGN}((\sigma_1)_i, (\sigma_2)_j)$ and $\sigma_{1(2)} \in \{X, Y, Z\}$, we compute $q = g_{ij}pg_{ij}$ as follows:

(1) Let $q \leftarrow p$.

(2) Compare the local support of p on qubit i to σ_1 and if they locally anticommute, $q \leftarrow (\sigma_2)_j q$.

(3) Repeat step 1 swapping the roles of the qubits.

(4) Include an additional sign if q was modified in both steps 2 and 3.

Note the order of the qubits in this process is irrelevant due to commutation of the relevant operators as can be seen in Eq. (B4). From this, one can derive the resulting gate action, including sign, for all $144 = 9 \times 4 \times 4$ possible local interactions. For completeness, we also note the single-qubit multiplication rule

$$XYZ = i, \tag{B5}$$

from which all single-qubit multiplication, with sign and phase, can be derived.

Note this discussion implies the assertion made in the main text that for any Pauli operator with support on qubits i, j, there exist four TQE gates acting on i, j, which reduce its support by one. To see this, suppose the Pauli has support X on i and Y on j. To remove support on i, we need the TQE gate to have type X on i and not have type Y on j. Similarly, to reduce support on j we need the TQE gate to have type Y on j and not have type X on i. Thus, the four gates would be $AX_{ij}, AZ_{ij}, BY_{ij}$, and CY_{ij}. This can be extended to any support.

For backward action by definition, we use Eq. (B4) to consider the action of the TQE gate on the origin frame and then swap out the single-qubit Paulis for the elements of the frame. But a TQE gate can only interact with single-qubit Pauli operators for the two qubits on which it acts. For TQE gate $g_{ij} = \mathrm{ASIGN}((\sigma_1)_i, (\sigma_2)_j)$ and $\sigma_{1(2)} \in \{X, Y, Z\}$ and $B = \{(s_i, \tilde{s}_i)\}_{0 \leqslant i < N}$, we compute the forward action $B' = g_{ij}(B)$ as follows:

(1) $B' \leftarrow \{(s_i, \tilde{s}_i)\}_{0 \leqslant i < N}$.

(2) Let

$$p = \begin{cases} \tilde{s}_j, & \text{if } \sigma_2 = X\\ -is_j\tilde{s}_j, & \text{if } \sigma_2 = Y\\ \tilde{s}_j, & \text{if } \sigma_2 = Z.\end{cases} \tag{B6}$$

(3) For elements in B',

$$\begin{cases} s_i \leftarrow s_i p, & \text{if } \sigma_1 = X \\ s_i \leftarrow s_i p, \tilde{s}_i \leftarrow \tilde{s}_i p, & \text{if } \sigma_1 = Y \\ \tilde{s}_i \leftarrow \tilde{s}_i p, & \text{if } \sigma_1 = Z. \end{cases} \tag{B7}$$

(4) Repeat steps 1 and 2 swapping the role of the qubits.

Note we use multiplication, not "binary addition," because this includes sign and thus order matters for the resulting sign. Also note the extra sign from Eq. (B4) does not enter here, so the sign always comes from the multiplication.

APPENDIX C: PROOF OF PROPOSITION 1

In this Appendix, we prove the claims of Proposition 1 of the main text.

Proof. We prove each claim in turn.

(1) $\mathrm{Supp}(p, B) \geqslant 0$ is trivially true. Furthermore, suppose $\mathrm{Supp}(p, B) = 0$ for some $p \in \mathcal{P}$ and $B \in \mathcal{B}$. This is true if and only if $\lambda(s_i, p) \vee \lambda(\tilde{s}_i, p) = 0$ for all i which is true if and only if $\lambda(s_i, p) = \lambda(\tilde{s}_i, p) = 0$ for all i. As B is a basis for $\mathcal{P}$, the former is true if and only if $p = I$.

(2) Consider the following difference:

$$\begin{aligned} &\mathrm{Supp}(p, B) + \mathrm{Supp}(q, B) - \mathrm{Supp}(p + q, B) \\ &\quad = \sum_i [\lambda(s_i, p) \vee \lambda(\tilde{s}_i, p) + \lambda(s_i, q) \vee \lambda(\tilde{s}_i, q) \\ &\qquad - \lambda(s_i, p + q) \vee \lambda(\tilde{s}_i, p + q)]. \end{aligned} \tag{C1}$$

So for each i term we have the general expression $x \vee \tilde{x} + y \vee \tilde{y} - (x \oplus \tilde{x}) \vee (y \oplus \tilde{y})$ using the binary linearity of λ. Such an expression is less than zero if and only if $x \vee \tilde{x} = y \vee \tilde{y} = 0$ and $(x \oplus \tilde{x}) \vee (y \oplus \tilde{y}) = 1$ but this is a contradiction as $x \vee \tilde{x} = y \vee \tilde{y} = 0$ implies that $x = \tilde{x} = y = \tilde{y} = 0$, in which case$(x \oplus \tilde{x}) \vee (y \oplus \tilde{y}) = 0$. Thus, every term on the right-hand side of Eq. (C1) is greater than zero, implying the left-hand side must be greater than zero. Therefore, condition (1) is true.

(3) To show this is true for all $\not{E}_N$, it is sufficient to show it is true for all its generators. First consider H_j for some j:

$$\begin{aligned} \mathrm{Supp}(p, \mathrm{H}_j(B)) &= \sum_{i \neq j} \lambda(s_i, p) \vee \lambda(\tilde{s}_i, p) \\ &\quad + \lambda(\tilde{s}_j, p) \vee \lambda(s_j, p) \\ &= \sum_i \lambda(s_i, p) \vee \lambda(\tilde{s}_i, p) \\ &= \mathrm{Supp}(p, B), \end{aligned} \tag{C2}$$

where we used the symmetry property of $\vee$. Also consider that $(x \oplus y) \vee y = x \vee y$. So

$$\begin{aligned} \mathrm{Supp}(p, \mathrm{S}_j(B)) &= \sum_{i \neq j} \lambda(s_i, p) \vee \lambda(\tilde{s}_i, p) \\ &\quad + \lambda(s_j + \tilde{s}_i, p) \vee \lambda(\tilde{s}_i, p) \\ &= \sum_i \lambda(s_i, p) \vee \lambda(\tilde{s}_i, p) \\ &= \mathrm{Supp}(p, B), \end{aligned} \tag{C3}$$

using the binary linearity of λ and the above Boolean equivalence. Finally, any qubit permutations such as SWAP trivially satisfies (3) as it only rearranges the terms defining Supp. Therefore, (3) is true.

(4) First, we show that some member of $B' \in [B]$ can be transformed so as to contain p using $\mathrm{Supp}(p, B) - 1$ CX, and then argue this is the minimum number. So let $d = \mathrm{Supp}(p, B)$. This implies there are d qubit indices, i such that $\lambda(s_i, p) \vee \lambda(\tilde{s}_i, p) = 1$. Let D be the set of these indices. For each such $i \in D$, if $\lambda(s_i, p) = \lambda(\tilde{s}_i, p) = 1$, apply $\mathrm{H}_i \circ \mathrm{S}_i(B)$ for which the ith part of the frame is then $(s_i + \tilde{s}_i, s_i)$. If $\lambda(s_i, p) = 1, \lambda(\tilde{s}_i, p) = 0$, apply $\mathrm{H}_i(B)$, for which the ith part of this frame is then $(\tilde{s}_i, s_i)$. All other cases, do nothing. In the resulting frame, which we refer to as B', $\lambda(s'_i, p) = 0, \lambda(\tilde{s}'_i, p) = 1$ for all i, and since $B \to B'$ uses only members of $\not{E}_N$, $B' \in [B]$. Because B' is a frame, we can use the expansion of p in this frame such that that $p = \sum_{i \in D} s'_i$. One can use $|D| - 1$ CNOTs to reduce this sum and reach a frame $B'' \ni p$. Furthermore, because this is the unique expansion of p in terms of this frame, this is the minimum number of CXs needed to reach such a frame. ■

APPENDIX D: DETAILED DESCRIPTION OF THE ULTRAGREEDY PFG SEARCH ALGORITHM

In this Appendix we provide a detailed description of the ultragreedy PFG search algorithm for a Hamiltonian path. For what follows, $\mathrm{bin}(p)$ converts p to its signed binary representation in the origin frame and N qubits. Recall c is a free parameter of the cost function $\mathrm{cost}_c(g) \leqslant 1$ as discussed in the main text. Details of the algorithms implementation are then outlined in Algorithm 1.

APPENDIX E: DETAILS OF PARALLELIZATION OF THE ULTRAGREEDY SEARCH ALGORITHM

In this Appendix, we discuss the methods used for parallelizing the ultragreedy search algorithm described in Appendix D and the main text. The code used in this study leveraged MPI to implement the parallelized program described here. It was found that the best method for parallelizing the algorithm was to assign each process (core or thread) a set of Hamiltonian terms for which that process is responsible. This does require a considerable amount of communication between the processes. The parallel program proceeded as follows:

[9]Note this method relies on the fact that no process will try to schedule a rotation on the same qubit as any another. This is guaranteed by two facts: (1) no Pauli terms are duplicated, and thus each process has a distinct list of Pauli operator terms. (2) Each pass through the main loop only applies one TQE gate. It should be clear that the application of a TQE gate only provides two new possible rotations not available in the previous frame. Furthermore, these are always on distinct qubits. Together, one can see that aside from the initial case (as was handled at the beginning by the rank 0 process), each pass through the main loop can not result in two rotation gates being scheduled on the same qubit.

ALGORITHM 1. PFG Ultragreedy Search.

```
Input: List of Hamiltonian terms and angles $H = \{(\theta_\alpha, p_\alpha)\}$
Return: List of circuit elements $C_{\text{return}}$
1: $C_{\text{return}} \leftarrow \{\}$, i.e., is an empty circuit.
2: ham $\leftarrow \{\text{bin}(p_\alpha)\}$.
3: While ham $\neq \emptyset$
4:     minsup $\leftarrow N$
5:     ham$_{\text{min}} = \{\}$
6:     cost $= 1.1$
7:     $g_{\text{min}} \leftarrow I$
8:     for $p \in$ ham
9:         if Supp$(p) = 1$
10:            Add rotation to $C_{\text{return}}$ according to local support
           for angle $\pm\theta_\alpha$ with sign given by sign of $p$.
11:            Remove $p$ from ham.
12:        else if Supp$(p) =$ minsup
13:            Add $p$ to ham$_{\text{min}}$
14:        else if $2 \leqslant$ Supp$(p) <$ minsup
15:            Clear ham$_{\text{min}}$ and add $p$
16:            minsup $\leftarrow$ Supp$(p)$
17:        end if
18:    end for
19:    for $p \in$ ham$_{\text{min}}$
20:        for all gates $g$ which reduces support of $p$
21:            if cost$_c(g) <$ cost
22:                $g_{\text{min}} \leftarrow g$.
23:                cost $\leftarrow$ cost$_c(g)$
24:            end if
25:        end for
26:    end for
27:    Add $g_{\text{min}}$ to $C_{\text{return}}$.
28:    Update all members of ham using the TQE transformation rules
       for $g_{\text{min}}$.
29: end while
```

(i) Any immediate support one terms are converted to a rotation and added to the output circuit by the rank-0 process (see footnote 9 for reason).

(ii) Each remaining term is assigned and communicated to its owning process.

(iii) Each process has a local copy of the leading time edge for each qubit, what we deem the *schedule*, which is initially communicated to each process.

(iv) The total global number of remaining terms is held locally and communicated to each process. This value being greater than zero is used as the continue condition for the main loop of the program.

(v) Each process enters the main loop for which the following happens:

(a) Each process searches its owned terms for those with support one. Such terms are converted to a rotation and added to a local circuit list with a timing stamp as obtained from the schedule.

(b) Once each process completes the above search, the schedule is synced between the processes by finding and sharing the global maximum value for each qubit.[9]

(c) Simultaneously with the steps above, each process searches for its minimum support terms. The resulting minimum cost is shared with the rank-0 process and a global minimum is communicated to every process. If the process has a local cost greater than the global cost, its local list of minimum support terms is cleared.

(d) Each process collects all TQE gates which would reduce its local list of minimum Pauli operator terms. These lists are collected by the master, duplicates are removed, and the final global list of TQE gates is communicated to each process.

(e) For each candidate TQE gate, the unnormalized average change in support is calculated locally for the terms owned by that process, and the global sum collected by the rank-0 process. The rank-0 process then normalizes this value by the global number of terms and completes the cost function calculation by including the parallelization credit.

(f) Once all candidate TQE gates are evaluated, the rank-0 process holds the minimum cost gate. This gate is communicated to the other processes. The master adds this gate to its circuit list with a time stamp from the schedule (maximum time between the two qubits acted upon).

(g) Each process updates all of its owned terms with the new TQE gate action, and its local schedule.

(h) The total global number of terms remaining is found and communicated to all processes.

(vi) Once all processes exit the main loop, each process communicates to the rank-0 process its list of circuit elements; this list is combined into one list and the list is sorted in time order.

(vii) The rank-0 process returns this final circuit list.

Another method for parallelization we investigated was to have every process work on the full list of terms, while splitting the candidate TQE gate cost evaluations evenly between the processes. Although this required far less communication between the processes, it was found that the speedup was limited by the search for the minimum support Pauli terms as this step could not be parallelized other than to do the above. So although the above method of splitting terms between the processes was considerably more complex, it resulted in a parallelized program which more closely matched the ideal speedup. Furthermore, other nonparallelization methods exist for reducing the cost of gate cost evaluation, but were not exploited in the study of the main text. We hope to explore additional speedups in future work.

[1] A. JavadiAbhari, S. Patil, D. Kudrow, J. Heckey, A. Lvov, F. T. Chong, and M. Martonosi, Parallel Comput. **45**, 2 (2015), computing Frontiers 2014: Best Papers.

[2] D. S. Steiger, T. Häner, and M. Troyer, Quantum **2**, 49 (2018).

[3] D. Koch, L. Wessing, and P. M. Alsing, arXiv:1903.04359.

[4] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, Quantum Sci. Technol. **6**, 014003 (2021).

[5] V. Kliuchnikov and D. Maslov, Phys. Rev. A **88**, 052307 (2013).

[6] N. Abdessaied, M. Soeken, and R. Drechsler, in *Reversible Computation: 6th International Conference, RC 2014* (Springer, Berlin, 2014), pp. 149–162.

[7] Y. Nam, N. J. Ross, Y. Su, A. M. Childs, and D. Maslov, npj Quantum Inf. **4**, 23 (2018).

[8] J. Pointing, O. Padon, Z. Jia, H. Ma, A. Hirth, J. Palsberg, and A. Aiken, arXiv:2111.11387.

[9] M. Xu, Z. Li, O. Padon, S. Lin, J. Pointing, A. Hirth, H. Ma, J. Palsberg, A. Aiken, U. A. Acar *et al.*, in *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation* (Association for Computing Machinery, New York, 2022), pp. 625–640.

[10] J. Paykin, A. Schmitz, M. Ibrahim, X. Wu and A. Matsuura, PCOAST: A Pauli-Based Quantum Circuit Optimization Framework, in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, (Bellevue, WA, USA, 2023), pp. 715–726.

[11] V. Vandaele, S. Martiel, S. Perdrix, and C. Vuillot, ACM J. **5**, 1 (2024).

[12] A. Cowtan, S. Dilkes, R. Duncan, W. Simmons, and S. Sivarajah, in *16th International Conference on Quantum Physics and Logic 2019* (Open Publishing Association, 2019), pp. 213–228.

[13] Z. Lu, P.-X. Shen, and D.-L. Deng, Phys. Rev. Appl. **16**, 044039 (2021).

[14] D. Wecker, M. B. Hastings, N. Wiebe, B. K. Clark, C. Nayak, and M. Troyer, Phys. Rev. A **92**, 062318 (2015).

[15] D. S. Abrams and S. Lloyd, Phys. Rev. Lett. **79**, 2586 (1997).

[16] B. Nachman, D. Provasoli, W. A. de Jong, and C. W. Bauer, Phys. Rev. Lett. **126**, 062001 (2021).

[17] B. Bauer, S. Bravyi, M. Motta, and G. K.-L. Chan, Chem. Rev. **120**, 12685 (2020).

[18] H. Ma, M. Govoni, and G. Galli, npj Comput. Mater. **6** (2020).

[19] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. D. Sawaya, S. Sim, L. Veis, and A. Aspuru-Guzik, Chem. Rev. **119**, 10856 (2019).

[20] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, Rev. Mod. Phys. **92**, 015003 (2020).

[21] V. E. Elfving, B. W. Broer, M. Webber, J. Gavartin, M. D. Halls, K. P. Lorton, and A. Bochevarov, arXiv:2009.12472.

[22] S. McArdle, A. Mayorov, X. Shan, S. Benjamin, and X. Yuan, Chem. Sci. **10**, 5725 (2019).

[23] N. P. D. Sawaya, F. Paesani, and D. P. Tabor, Phys. Rev. A **104**, 062419 (2021).

[24] P. J. Ollitrault, A. Baiardi, M. Reiher, and I. Tavernelli, Chem. Sci. **11**, 6842 (2020).

[25] M. Suzuki, Commun. Math. Phys. **51**, 183 (1976).

[26] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma, Phys. Rev. Lett. **114**, 090502 (2015).

[27] G. H. Low and I. L. Chuang, Phys. Rev. Lett. **118**, 010501 (2017).

[28] G. H. Low and I. L. Chuang, Quantum **3**, 163 (2019).

[29] A. M. Childs, A. Ostrander, and Y. Su, Quantum **3**, 182 (2019).

[30] E. Campbell, Phys. Rev. Lett. **123**, 070503 (2019).

[31] D. W. Berry, A. M. Childs, and R. Kothari, in *2015 IEEE 56th Annual Symposium on Foundations of Computer Science* (IEEE, Piscataway, NJ, 2015), pp. 792–809.

[32] A. M. Childs, Y. Su, M. C. Tran, N. Wiebe, and S. Zhu, Phys. Rev. X **11**, 011020 (2021).

[33] A. Peruzzo, J. Mcclean, P. Shadbolt, M. H. Yung, X. Zhou, P. Love, A. Aspuru-Guzik, and J. O'Brien, Nat. Commun. **5** (2013).

[34] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, Rev. Mod. Phys. **94**, 015004 (2022).

[35] D. Fedorov, B. Peng, N. Govind, and Y. Alexeev, Mater. Theory **6**, 2 (2022).

[36] C. L. Cortes and S. K. Gray, Phys. Rev. A **105**, 022417 (2022).

[37] L. Lin and Y. Tong, PRX Quantum **3**, 010318 (2022).

[38] M. Motta, C. Sun, A. Tan, M. O'Rourke, E. Ye, A. Minnich, F. Brandão, and G. Chan, Nat. Phys. **16**, 231 (2020).

[39] Y. Dong, L. Lin, and Y. Tong, PRX Quantum **3**, 040305 (2022).

[40] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, Phys. Rev. A **52**, 3457 (1995).

[41] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. (Cambridge University Press, New York, 2011).

[42] A. Fagan and R. Duncan, EPTCS **287**, 85 (2019).

[43] B. Nash, V. Gheorghiu, and M. Mosca, Quantum Sci. Technol. **5**, 025010 (2020).

[44] D. Maslov and M. Roetteler, IEEE Trans. Inf. Theory **64**, 4729 (2018).

[45] S. Aaronson and D. Gottesman, Phys. Rev. A **70**, 052328 (2004).

[46] A. Cowtan, S. Dilkes, R. Duncan, W. Simmons, and S. Sivarajah, *Proceedings 16th International Conference on Quantum Physics and Logic, QPL 2019, Chapman University, Orange, CA*, 213 (2019).

[47] M. Motta, E. Ye, J. R. McClean, Z. Li, A. J. Minnich, R. Babbush, and G. K.-L. Chan, npj Quantum Inf. **7**, 83 (2021).

[48] M. B. Hastings, D. Wecker, B. Bauer, and M. Troyer, Quantum Inf. Comput. **15**, 361 (2015).

[49] D. Poulin, M. Hastings, D. Wecker, N. Wiebe, A. Doherty, and M. Troyer, Quantum Inf. Comput. **15** (2014).

[50] G. Li, A. Wu, Y. Shi, A. Javadi-Abhari, Y. Ding, and Y. Xie, in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '22 (Association for Computing Machinery, New York, 2022), p. 554–569.

[51] Y. Hu, F. Meng, X. Wang, T. Luan, Y. Fu, Z. Zhang, X. Zhang, and X. Yu, Quantum Sci. Technol. **7**, 045001 (2022).

[52] A. Tranter, P. Love, F. Mintert, and P. Coveney, J. Chem. Theory Comput. **14**, 5617 (2018).

[53] B. G. Bodmann, M. Le, L. Reza, M. Tobin, and M. Tomforde, Involve **2**, 589 (2009).

[54] B. M. Terhal, Rev. Mod. Phys. **87**, 307 (2015).

[55] D. Gottesman, Stabilizer codes and quantum error correction, Ph.D. thesis, California Institute of Technology, 1997.

[56] A. T. Schmitz, Ann. Phys. **410**, 167927 (2019).

[57] J. Tolar, J. Phys.: Conf. Ser. **1071**, 012022 (2018).

[58] J. Paykin, A. T. Schmitz, M. Ibrahim, X.-C. Wu, and A. Y. Matsuura, PCOAST: A Pauli-based quantum circuit optimization framework (extended version), 2023, arXiv:2305.10966 [quant-ph].

[59] J. R. McClean, N. C. Rubin, K. J. Sung, I. D. Kivlichan, X. Bonet-Monroig, Y. Cao, C. Dai, E. S. Fried, C. Gidney, B. Gimby, P. Gokhale, T. Häner, T. Hardikar, V. Havlíček, O. Higgott, C. Huang, J. Izaac, Z. Jiang, X. Liu, S. McArdle *et al.*, and Quantum Sci. Technol. **5**, 034014 (2020).

[60] N. P. D. Sawaya, T. Menke, T. H. Kyaw, S. Johri, A. Aspuru-Guzik, and G. G. Guerreschi, npj Quantum Inf. **6**, 49 (2020).

[61] S. B. Bravyi and A. Y. Kitaev, Ann. Phys. **298**, 210 (2002).

[62] A. Tranter, S. Sofia, J. Seeley, M. Kaicher, J. McClean, R. Babbush, P. V. Coveney, F. Mintert, F. Wilhelm, and P. J. Love, Int. J. Quantum Chem. **115**, 1431 (2015).

[63] T. Helgaker, P. Jørgensen, and J. Olsen, *Molecular Electronic-Structure Theory* (Wiley, New York, 2000).

[64] J. D. Whitfield, J. Biamonte, and A. Aspuru-Guzik, Mol. Phys. **109**, 735 (2011).

[65] J. R. McClean, R. Babbush, P. J. Love, and A. Aspuru-Guzik, J. Phys. Chem. Lett. **5**, 4368 (2014).

[66] R. Babbush, J. McClean, D. Wecker, A. Aspuru-Guzik, and N. Wiebe, Phys. Rev. A **91**, 022311 (2015).

[67] M. D. Hanwell, D. E. Curtis, D. C. Lonie, T. Vandermeersch, E. Zurek, and G. R. Hutchison, J. Cheminformatics **4**, 17 (2012).

[68] A. K. Rappe, C. J. Casewit, K. S. Colwell, W. A. Goddard, and W. M. Skiff, J. Am. Chem. Soc. **114**, 10024 (1992).

[69] R. M. Parrish, L. A. Burns, D. G. A. Smith, A. C. Simmonett, A. E. DePrince, E. G. Hohenstein, U. Bozkaya, A. Y. Sokolov, R. Di Remigio, R. M. Richard, J. F. Gonthier, A. M. James, H. R. McAlexander, A. Kumar, M. Saitow, X. Wang, B. P. Pritchard, P. Verma, H. F. Schaefer, K. Patkowski *et al.*, J. Chem. Theory Comput. **13**, 3185 (2017).

[70] M. P. A. Fisher, P. B. Weichman, G. Grinstein, and D. S. Fisher, Phys. Rev. B **40**, 546 (1989).

[71] I. Bloch, J. Dalibard, and W. Zwerger, Rev. Mod. Phys. **80**, 885 (2008).

[72] N. P. D. Sawaya and J. Huh, J. Phys. Chem. Lett. **10**, 3586 (2019).

[73] A. T. Schmitz, M. Ibrahim, N. P. D. Sawaya, G. G. Guerreschi, J. Paykin, X.-C. Wu, and A. Y. Matsuura, Optimization at the Interface of Unitary and Non-unitary Quantum Operations in PCOAST, in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, Bellevue, WA, USA (IEEE, 2023), pp. 727–738.

[74] P. Khalate, X.-C. Wu, S. Premaratne, J. Hogaboam, A. Holmes, A. Schmitz, G. G. Guerreschi, X. Zou, and A. Y. Matsuura, An LLVM-based C++ compiler toolchain for variational hybrid quantum-classical algorithms and quantum accelerators, arXiv:2202.11142.