Hardware-Tailored Diagonalization Circuits

Daniel Miller,[1,2,3,*] Laurin E. Fischer,[3] Kyano Levi,[1] Eric J. Kuehnke,[1]
Igor O. Sokolov,[3,4] Panagiotis Kl. Barkoutsos,[3,4] Jens Eisert,[1] and Ivano Tavernelli[3]

[1] *Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany*
[2] *Institute for Theoretical Nanoelectronics (PGI-2), Forschungszentrum Jülich, 52428 Jülich, Germany*
[3] *IBM Quantum, IBM Research Europe – Zurich, Säumerstrasse 4, 8803 Rüschlikon, Switzerland*
[4] *Current address: PASQAL, 7 rue Léonard de Vinci, 91300 Massy, France*

A central building block of many quantum algorithms is the diagonalization of Pauli operators. Although it is always possible to construct a quantum circuit that simultaneously diagonalizes a given set of commuting Pauli operators, only resource-efficient circuits can be executed reliably on near-term quantum computers. Generic diagonalization circuits, in contrast, often lead to an unaffordable Swap gate overhead on quantum devices with limited hardware connectivity. A common alternative is to exclude two-qubit gates altogether. However, this comes at the severe cost of restricting the class of diagonalizable sets of Pauli operators to tensor product bases (TPBs). In this article, we introduce a theoretical framework for constructing hardware-tailored (HT) diagonalization circuits. Our framework establishes a systematic and highly flexible procedure for tailoring diagonalization circuits with ultra-low gate counts. We highlight promising use cases of our framework and – as a proof-of-principle application – we devise an efficient algorithm for grouping the Pauli operators of a given Hamiltonian into jointly-HT-diagonalizable sets. For several classes of Hamiltonians, we observe that our approach requires fewer measurements than conventional TPB approaches. Finally, we experimentally demonstrate that HT circuits can improve the efficiency of estimating expectation values with cloud-based quantum computers.

INTRODUCTION

Since first-generation quantum computers were made publicly available eight years ago by IBM [1], the technological frontier is expanding at an ever increasing pace [2–12]. Nevertheless, decoherence and hardware errors still limit the applicability of these early-stage quantum devices, and practical quantum advantage yet remains to be demonstrated. To go beyond what is possible now, it is crucial to operate both classical and quantum computers in an orchestrated manner that exploits their respective strengths.

For example, in the *variational quantum eigensolver* (VQE) algorithm [13–15], a classical computer optimizes the parameters of a trial quantum state vector $|\psi\rangle$ to accurately approximate the minimal eigenvalue of an observable O, e.g., the Hamiltonian of a molecule. The tasks performed by the quantum processor are preparing $|\psi\rangle$ and gathering measurement data from which the expectation value $\langle O\rangle = \langle\psi| O |\psi\rangle$ can be estimated. In practice, the observable O cannot be measured directly as this would require a quantum circuit that diagonalizes it, i.e., a circuit that rotates the unknown eigenbasis of O to the computational basis. A common approach to circumvent this problem is to express O as a linear combination of n-qubit Pauli operators $P_i \in \{I, X, Y, Z\}^{\otimes n}$ with real coefficients $c_i \in \mathbb{R}$, as in

$$O = \sum_{i=1}^{M} c_i P_i. \tag{1}$$

Since any Pauli operator P_i can be diagonalized using single-qubit Clifford gates, it is straightforward to measure its expectation value $\langle P_i\rangle$. Once all $\langle P_i\rangle$ have been obtained, $\langle O\rangle$ is calculated from Eq. (1). Although the number M of Pauli operators for molecular Hamiltonians has a nominal scaling of up to $\mathcal{O}(n^4)$, measuring all Pauli expectation values individually would require a large number of quantum circuit executions ("shots") [15]. In addition to the mere evaluation of $\langle O\rangle$, usually also its gradient is estimated from the measured data [16, 17]. To keep resource requirements at an affordable level, one can make use of simultaneous measurements of commuting Pauli operators, a technique that is commonly applied. For example, with a diagonalization circuit that only contains single-qubit gates, one can measure a *tensor product basis* (TPB), i.e., a set of Pauli operators that are *qubit-wise commuting* (QWC) [2]. For typical problems it is possible to group an average number of three Pauli operators into a common TPB [18]. To further reduce the number of required diagonalization circuits from $\mathcal{O}(n^4)$ to $\mathcal{O}(n^3)$, grouping the Pauli operators into *general commuting* (GC) sets has been suggested [19–21]. Under ideal circumstances, such GC groupings would substantially decrease the number of shots required to estimate $\langle O\rangle$ to a desired accuracy [22]. Unfortunately, the corresponding diagonalization circuits consist of up to $\frac{n(n-1)}{2}$ two-qubit gates and, if the connectivity of the device is limited, a large number of additional Swap gates. As a first step to interpolate between these extremes, diagonalization circuits featuring a single layer of two-qubit gates have been introduced [23, 24]; however, it is an open challenge to fully exploit the trade-off between QWC and GC. As of today, the best method to experimentally estimate an expectation value $\langle O\rangle$ is unclear. Besides Pauli

* daniel.miller@fu-berlin.de

grouping [2, 18–26], there is active research in addressing this problem with classical shadows [27–32], unitary partitioning [33, 34], low-rank factorization [35], adaptive estimators [36–38], and decision diagrams [39].

In this article, we introduce a theoretical framework for constructing diagonalization circuits whose two-qubit gates are tailored to meet the connectivity restrictions imposed by most current quantum computing architectures, e.g., super- and semiconducting qubits [2–10]. Our flexible approach can be applied to any hardware connectivity. We demonstrate the viability of our techniques for a large class of paradigmatic Hamiltonians in the context of the Pauli grouping problem.

RESULTS

Framework for HT Diagonalization Circuits

The purpose of our theoretical framework is the construction of *hardware-tailored* (HT) Clifford circuits that diagonalize a given set of commuting n-qubit Pauli operators $P_1, \ldots, P_m$. After possibly replacing some of the P_j by $-P_j$, we can assume that the group they generate, which is denoted by $\langle P_1, \ldots, P_m \rangle$, does not contain $-I^{\otimes n}$. From now on, we will always assume that this is the case since it allows us to extend $\langle P_1, \ldots, P_m \rangle$ to the stabilizer group $\mathcal{S}$ of some stabilizer state vector $|\psi_{\mathcal{S}}\rangle$, where $|\psi_{\mathcal{S}}\rangle$ is defined as the common $+1$-eigenvector of all operators $S \in \mathcal{S}$, see the *Supplementary Material* (SM) Sec. III. Then, uncomputing the state vector $|\psi_{\mathcal{S}}\rangle$, i.e., applying some Clifford circuit $U_{\mathcal{S}}^\dagger$ for which $|\psi_{\mathcal{S}}\rangle = U_{\mathcal{S}} |0\rangle^{\otimes n}$, will simultaneously diagonalize $P_1, \ldots, P_m$ [40].

An important class of stabilizer states is that of graph states [41]. A graph with n vertices is defined in terms of its adjacency matrix $\Gamma = (\gamma_{i,j}) \in \mathbb{F}_2^{n \times n}$, where $\mathbb{F}_2$ is the binary field; a pair (i, j) of vertices is connected via an edge if and only if $\gamma_{i,j} = 1$. In this article, we do not distinguish between a graph and its adjacency matrix, and we follow the convention $\gamma_{i,j} = \gamma_{j,i}$ and $\gamma_{i,i} = 0$ for all $i, j \in \{1, \ldots, n\}$. Every graph Γ defines a graph state vector $|\Gamma\rangle = U_\Gamma |0\rangle^{\otimes n}$ whose preparation circuit

$$U_\Gamma = \left(\prod_{i<j} \mathrm{CZ}_{i,j}^{\gamma_{i,j}} \right) H^{\otimes n} \tag{2}$$

consists of a layer of Hadamard gates $H = \frac{1}{\sqrt{2}}(X + Z)$, followed by a two-qubit gate $\mathrm{CZ} = \mathrm{diag}(1, 1, 1, -1)$ for every pair of connected vertices. The stabilizer group of $|\Gamma\rangle$ is $\mathcal{S}_\Gamma = \{X^{\mathbf{k}} Z^{\Gamma \mathbf{k}} (-1)^{\sum_{i<j} k_i \gamma_{i,j} k_j} \mid \mathbf{k} \in \mathbb{F}_2^n\}$, where we define $X^{\mathbf{k}} = X^{k_1} \otimes \ldots \otimes X^{k_n}$ and similarly $Z^{\Gamma \mathbf{k}}$ for the matrix-vector product $\Gamma \mathbf{k}$. Hence, $U_\Gamma^\dagger$ would diagonalize our operators $P_1, \ldots, P_m$ if only they were of the form $\pm X^{\mathbf{k}} Z^{\Gamma \mathbf{k}}$.

Every stabilizer state is *local-Clifford* (LC) equivalent to a graph state [42]. Thus, there exist single-qubit Clifford gates $U_1, \ldots, U_n$ and a graph Γ such that

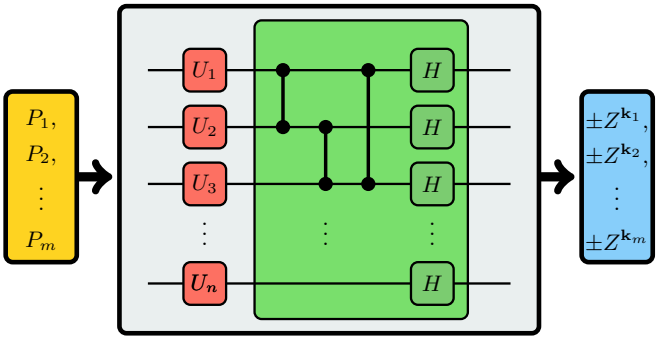

FIG. 1. Graph-based diagonalization circuit. A set of commuting Pauli operators $P_1, \ldots, P_m$ (yellow) is diagonalized in two steps: first, a layer of single-qubit Clifford gates $U = U_1 \otimes \ldots \otimes U_n$ (red) rotates them into a set of the form $\mathcal{S} = \{\pm X^{\mathbf{k}} Z^{\Gamma \mathbf{k}} \mid \mathbf{k} \in \mathbb{F}_2^n\}$, where $\Gamma \in \mathbb{F}_2^{n \times n}$ is an adjacency matrix. Afterward, $\mathcal{S}$ is rotated to the computational basis (blue) by uncomputing the graph state vector $|\Gamma\rangle$ (green). The existence of U and Γ is guaranteed because every stabilizer state is LC-equivalent to a graph state [42]. We call a graph-based diagonalization circuit *hardware-tailored* (HT) if Γ is a subgraph of the connectivity graph Γ_{con} of the considered quantum device.

$(U_1 \otimes \ldots \otimes U_n) |\psi_{\mathcal{S}}\rangle = |\Gamma\rangle$. We conclude that every set of commuting Pauli operators can be simultaneously diagonalized by some layer of single-qubit Clifford gates followed by some circuit of the form $U_\Gamma^\dagger$. We refer to this procedure as a graph-based diagonalization circuit, see Fig. 1.

The *connectivity graph* of a quantum computer is the graph Γ_{con} whose vertices and edges, respectively, are given by qubits and pairs of qubits for which a CZ gate can be physically implemented. For quantum devices with a limited connectivity, general graph-based diagonalization circuits require up to $\mathcal{O}(n^2)$ SWAP gates [43]. This overhead renders unconstrained graph-based diagonalization circuits infeasible for certain near-term applications, see SM Sec. II. However, if $\Gamma \subset \Gamma_{\mathrm{con}}$ is a subgraph of the connectivity graph, SWAP gates are avoided completely. We refer to graph-based diagonalization circuits that are designed to meet this condition as *hardware-tailored* (HT). By construction, HT circuits have a very low CZ depth that is upper bounded by the maximum degree of Γ_{con}, e.g., by 3 and 4 for heavy-hex and square-lattice connectivity, respectively [44].

We now derive a technical condition for the existence of HT diagonalization circuits. Every n-qubit Clifford gate U defines a symplectic matrix

$$A = \begin{bmatrix} A^{xx} & A^{xz} \\ A^{zx} & A^{zz} \end{bmatrix} \in \mathrm{GL}(\mathbb{F}_2^{2n}) \tag{3}$$

with the property that

$$U X^{\mathbf{r}} Z^{\mathbf{s}} U^\dagger = \mathrm{i}^{\alpha(\mathbf{r},\mathbf{s})} X^{A^{xx}\mathbf{r} + A^{xz}\mathbf{s}} Z^{A^{zx}\mathbf{r} + A^{zz}\mathbf{s}} \tag{4}$$

holds for all vectors $\mathbf{r}, \mathbf{s} \in \mathbb{F}_2^n$, see Tab. I for the single-qubit case [40]. Hereby, a matrix A is called symplectic

TABLE I. Binary representation of the single-qubit Clifford group $\mathcal{C}_1$. Every $U \in \mathcal{C}_1$ is a product of H and $S = \mathrm{diag}(1, \mathrm{i})$. The six matrices $A \in \mathrm{GL}(\mathbb{F}_2^2)$ isomorphically correspond to the permutations of $\{X, Y, Z\}$.

U	I	H	S	HSH	HS	SH
$UXU^\dagger$	X	Z	$\mathrm{i}XZ$	X	$-\mathrm{i}XZ$	Z
$UZU^\dagger$	Z	X	Z	$-\mathrm{i}XZ$	X	$\mathrm{i}XZ$
$\alpha(0,1)$	0	0	0	3	0	1
$\alpha(1,0)$	0	0	1	0	3	0
$A = \begin{bmatrix} a^{xx} & a^{xz} \\ a^{zx} & a^{zz} \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$

if $A^{\mathrm{T}}\left[\begin{smallmatrix} 0 & \mathbb{1} \\ \mathbb{1} & 0 \end{smallmatrix}\right]A = \left[\begin{smallmatrix} 0 & \mathbb{1} \\ \mathbb{1} & 0 \end{smallmatrix}\right]$, with $\mathrm{GL}(\mathbb{F}_2^{2n})$ denoting the general linear group of $\mathbb{F}_2^{2n}$. For the time being, we can neglect the phases that are given by $\alpha(\mathbf{r}, \mathbf{s}) \in \mathbb{Z}/4\mathbb{Z} = \{0, 1, 2, 3\}$ (see Eq. (46) in SM Sec. XIII for how to recover them) and focus on the irreducible representation

$$\mathcal{C}_n \longrightarrow \mathrm{GL}(\mathbb{F}_2^{2n}), \qquad U \longmapsto A \tag{5}$$

of the n-qubit Clifford group $\mathcal{C}_n$. If $U = U_1 \otimes \ldots \otimes U_n$ is a single-qubit Clifford layer, the blocks in Eq. (3) are diagonal, i.e., $A^{xx} = \mathrm{diag}(a_1^{xx}, \ldots, a_n^{xx})$, and similarly for A^{xz}, A^{zx}, and A^{zz}. Hereby, a_1^{xx} is given by the xx-entry of the binary representation of U_1, etc. To construct a single-qubit Clifford layer that rotates $P_1, \ldots, P_m$ into the set $\{\pm X^{\mathbf{k}} Z^{\Gamma\mathbf{k}}\}$ for a graph Γ, we write $P_j = \mathrm{i}^{q_j} X^{\mathbf{r}_j} Z^{\mathbf{s}_j}$. By Eq. (4), the application of U, which is represented by A, transforms the operator P_j into $P_j' = X^{\mathbf{k}_j} Z^{A^{zx}\mathbf{r}_j + A^{zz}\mathbf{s}_j}$ (up to a global phase), where we have introduced the notation $\mathbf{k}_j = A^{xx}\mathbf{r}_j + A^{xz}\mathbf{s}_j$. Thus, $P_j' \in \{\pm X^{\mathbf{k}} Z^{\Gamma\mathbf{k}}\}$ is equivalent to $\Gamma\mathbf{k}_j = A^{zx}\mathbf{r}_j + A^{zz}\mathbf{s}_j$. These equivalent conditions can be phrased for all $j \in \{1, \ldots, m\}$ simultaneously as a binary matrix equation

$$\Gamma A^{xx} R + \Gamma A^{xz} S = A^{zx} R + A^{zz} S, \tag{6}$$

where $R = (\mathbf{r}_1 \ \cdots \ \mathbf{r}_m)$ and $S = (\mathbf{s}_1 \ \cdots \ \mathbf{s}_m)$ are the two matrices that store the exponent vectors of $P_1, \ldots, P_m$ as their columns.

Equation (6) is our first main result. First, by devising algorithms to solve it, we can tackle the challenge of constructing HT diagonalization circuits. This finally allows us to explore the trade-off between single-qubit Clifford layers and unrestricted Clifford circuits. On a related note, merely checking whether a guess (A, Γ) solves Eq. (6) is of course much more efficient than solving Eq. (6) from scratch. Therefore, our formulation of Eq. (6) supports the design of HT diagonalization circuits via educated guesses or dedicated case-based research for, in principle, arbitrarily large system sizes. We illustrate this idea in Tab. II for molecular Hamiltonians with up to $n = 120$ qubits.

For a quantum chip whose connectivity graph Γ_{con} has n vertices and e edges, there are 6^n and 2^e potential choices for A and Γ, respectively. Hence, a `brute-force solver` for Eq. (6) would loop through all $6^n 2^e$ choices,

TABLE II. Performance of an educated guess (A, Γ) as measured by the number m of jointly-HT-diagonalizable n-qubit Pauli operators $P_1, \ldots, P_m$ occurring in a hydrogen chain Hamiltonian $O = \sum_{i=1}^{M} c_i P_i$ for which (A, Γ) solves Eq. (6). The guess corresponds to the constant-depth circuit $H^{\otimes n}(\prod_{k=1}^{n/4} \mathrm{CZ}_{4k+1,4k+2}\mathrm{CZ}_{4k+2,4k+3})H^{\otimes n}$ which makes use of $n/2$ two-qubit gates and is tailored to a linear hardware connectivity. To diagonalize the other $M - m$ Pauli operators in O, one would need to find additional HT circuits, e.g., by making educated guesses based on careful inspection of the circuits in Tab. VII of SM Sec. XIII. The performance of such guesses can be easily assessed by checking for how many of the remaining Pauli operators Eq. (6) is fulfilled. The advantage of the here-presented HT circuit over tensor product bases is quantified by $N_{\mathrm{TPB}}^{\mathrm{circs}}$, which is the number of circuits needed to diagonalize the same operators $P_1, \ldots, P_m$ if two-qubit gates are forbidden. See methods for details about Hamiltonians.

n	20	40	60	80	100	120
M	7.2k	117k	595k	1.9M	4.6M	9.6M
m	191	781	1771	3161	4951	7141
$N_{\mathrm{TPB}}^{\mathrm{circs}}$	19	24	25	28	30	32

which quickly becomes infeasible due to the exponential size of the search space. Restricting to a polynomially-large random subset gives rise to an efficient, probabilistic, `restricted brute-force solver` for Eq. (6), however, we empirically find that this approach has a vanishingly low success probability. After closely investigating the mathematics behind Eq. (6), we can overcome this problem and formulate an efficient probabilistic solver that performs well in practice. Having available a huge search space will then turn into a powerful feature: the expressivity of Eq. (6) enables us to construct ultra-short diagonalization circuits.

Mathematical Results

We have reduced the task of constructing HT diagonalization circuits to the problem of solving Eq. (6) for a subgraph $\Gamma \subset \Gamma_{\mathrm{con}}$ and a symplectic, invertible matrix A whose blocks in Eq. (3) are diagonal. In our case, it is sufficient that A is invertible because every invertible matrix $A_i \in \mathbb{F}_2^{2\times 2}$, which represents U_i in $U = U_1 \otimes \ldots \otimes U_n$, is necessarily also symplectic, see Tab. I. Because of $\det(A) = \prod_{i=1}^{n} \det(A_i)$ and $\mathbb{F}_2 = \{0, 1\}$, the invertibility of A is equivalent to $\det(A_1) = \ldots = \det(A_n) = 1$. As we will show soon in Eq. (13), for a fixed graph Γ, it is possible to rewrite these determinants as quadratic forms

$$\det(A_i) = \boldsymbol{\lambda}^{\mathrm{T}} Q_i \boldsymbol{\lambda}. \tag{7}$$

The equation $\boldsymbol{\lambda}^{\mathrm{T}} Q_i \boldsymbol{\lambda} = 1$ defines what is known in algebraic geometry as a quadric hypersurface $\mathcal{L}_i \subset \mathbb{F}_2^d$, i.e., a generalization of a conic section, see Fig. 2. As a consequence, $\det(A_1) = \ldots = \det(A_n) = 1$ defines the intersection $\mathcal{L} = \bigcap_{i=1}^{n} \mathcal{L}_i$. Note that the points in $\mathcal{L}$ are

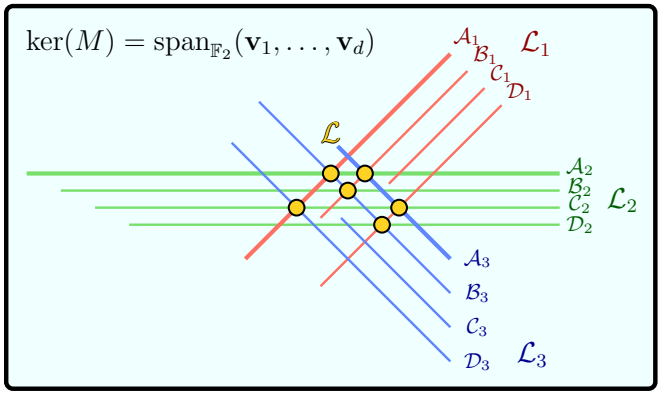

FIG. 2. Illustration of the solution space $\mathcal{L}$ (yellow dots) of Eq. (6) for a fixed graph Γ in a hypothetical scenario with $n = 3$ qubits. The ambient space is the d-dimensional null space of the binary matrix M defined in Eq. (9). In this example, $\mathcal{L}$ is the intersection of three quadric hypersurfaces $\mathcal{L}_1$ (red), $\mathcal{L}_2$ (green), and $\mathcal{L}_3$ (blue). Every hypersurface $\mathcal{L}_i$ is the union of four non-intersecting affine subspaces $\mathcal{A}_i$, $\mathcal{B}_i$, $\mathcal{C}_i$, and $\mathcal{D}_i$ (parallel lines) defined in Eqs. (15)–(18).

in one-to-one correspondence with single-qubit Clifford layers that transform $P_1, \ldots, P_m$ into stabilizers of $|\Gamma\rangle$. In particular, $\mathcal{L}$ is non-empty if and only if it is possible to diagonalize $P_1, \ldots, P_m$ with a circuit based on Γ as in Fig. 1.

In our setting, every quadric hypersurface $\mathcal{L}_i$ can be written as the union of at most four affine spaces. Therefore, their intersection $\mathcal{L} = \bigcap_{i=1}^n \mathcal{L}_i$ is the union of at most 4^n intersections of affine subspaces, each of which is itself an affine subspace. Using Gaussian elimination over $\mathbb{F}_2$, we can efficiently probe whether such an intersection is non-empty and, once a point $\lambda \in \mathcal{L}$ is found, we have successfully constructed a HT circuit.

What is gained? At first glance, not much: we still have 2^e choices for $\Gamma \subset \Gamma_{\mathrm{con}}$ and up to 4^n potential intersections in which a solution might lie. Hence, also in this reformulation, a brute force approach will be inefficient in the worst case. It turns out, however, that the efficient but probabilistic version, where only a polynomial number of subgraphs $\Gamma \subset \Gamma_{\mathrm{con}}$ and intersections is probed, has a sufficiently high empirical success probability to facilitate the construction of HT circuits for various practical problems as we will demonstrate later in Fig. 5.

For pedagogical reasons, we have already revealed the geometry behind Eq. (6). Let us now delve into the corresponding algebra and show that the picture in Fig. 2 is correct. To solve Eq. (6) in practice, we have to bring $\det(A) = 1$ into a form a classical computer can deal with. First, we exploit that the blocks of A in Eq. (3) are diagonal. Thus, we can replace A by the vector

$$\mathbf{a} = \begin{bmatrix} \mathbf{a}^{xx} \\ \mathbf{a}^{xz} \\ \mathbf{a}^{zx} \\ \mathbf{a}^{zz} \end{bmatrix} \in \mathbb{F}_2^{4n} \tag{8}$$

where the vector $\mathbf{a}^{xx} = (a_1^{xx}, \ldots, a_n^{xx}) \in \mathbb{F}_2^n$ defines the xx-block via $A^{xx} = \mathrm{diag}(\mathbf{a}^{xx})$, and likewise for $\mathbf{a}^{xz}$, $\mathbf{a}^{zx}$, and $\mathbf{a}^{zz}$. In this notation, Eq. (6) reads $M\mathbf{a} = 0$ for the $(mn \times 4n)$-matrix

$$M = \begin{bmatrix} \Gamma\mathrm{diag}(\mathbf{r}_1) & \Gamma\mathrm{diag}(\mathbf{s}_1) & \mathrm{diag}(\mathbf{r}_1) & \mathrm{diag}(\mathbf{s}_1) \\ \vdots & \vdots & \vdots & \vdots \\ \Gamma\mathrm{diag}(\mathbf{r}_m) & \Gamma\mathrm{diag}(\mathbf{s}_m) & \mathrm{diag}(\mathbf{r}_m) & \mathrm{diag}(\mathbf{s}_m) \end{bmatrix}. \tag{9}$$

Via Gaussian elimination, we can efficiently compute a basis $\mathbf{v}_1, \ldots, \mathbf{v}_d \in \mathbb{F}_2^{4n}$ of the null space of M. Thus, every vector $\mathbf{a} \in \mathbb{F}_2^{4n}$ with $M\mathbf{a} = 0$ is of the form

$$\mathbf{a} = \sum_{j=1}^{d} \lambda_j \mathbf{v}_j \tag{10}$$

for some vector $\boldsymbol{\lambda} \in \mathbb{F}_2^d$. In order to correspond to a physical solution, the vector $\mathbf{a}$ also needs to fulfill

$$\det(A_i) = a_i^{xx} a_i^{zz} + a_i^{zx} a_i^{xz} = 1 \tag{11}$$

for every $i \in \{1, \ldots, n\}$ as this will allow us to invert the irreducible representation $\mathcal{C}_n \to \mathrm{GL}(\mathbb{F}_2^{2n}), U \mapsto A$. Based on Ansatz (10), we find

$$a_i^{xx} a_i^{zz} = \sum_{j,j'=1}^{d} \lambda_j v_{i,j}^{xx} v_{i,j'}^{zz} \lambda_{j'} = \boldsymbol{\lambda}^{\mathrm{T}} (\mathbf{x}_i \mathbf{z}_i^{\mathrm{T}}) \boldsymbol{\lambda} \tag{12}$$

and similarly $a_i^{xz} a_i^{zx} = \boldsymbol{\lambda}^{\mathrm{T}} (\mathbf{w}_i \mathbf{y}_i^{\mathrm{T}}) \boldsymbol{\lambda}$, where we have introduced $\mathbf{x}_i = (v_{i,1}^{xx}, \ldots, v_{i,d}^{xx})$, $\mathbf{z}_i = (v_{i,1}^{zz}, \ldots, v_{i,d}^{zz})$, $\mathbf{w}_i = (v_{i,1}^{xz}, \ldots, v_{i,d}^{xz})$, and $\mathbf{y}_i = (v_{i,1}^{zx}, \ldots, v_{i,d}^{zx})$. Further inserting these expressions into Eq. (11), we obtain $\det(A_i) = \boldsymbol{\lambda}^{\mathrm{T}} (\mathbf{x}_i \mathbf{z}_i^{\mathrm{T}} + \mathbf{w}_i \mathbf{y}_i^{\mathrm{T}}) \boldsymbol{\lambda}$ and, by defining the matrix

$$Q_i = \mathbf{x}_i \mathbf{z}_i^{\mathrm{T}} + \mathbf{w}_i \mathbf{y}_i^{\mathrm{T}} \in \mathbb{F}_2^{d \times d}, \tag{13}$$

we finally arrive at Eq. (7). To proceed, we point out that $\det(A_i) = \boldsymbol{\lambda}^{\mathrm{T}} Q_i \boldsymbol{\lambda} = 1$ is equivalent to

$$\begin{bmatrix} \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{x}_i \\ \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{z}_i \\ \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{w}_i \\ \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{y}_i \end{bmatrix} \in \left\{ \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \right\}. \tag{14}$$

For example, if $\boldsymbol{\lambda}^{\mathrm{T}} \mathbf{x}_i = 0$, the value of $\boldsymbol{\lambda}^{\mathrm{T}} \mathbf{z}_i$ is irrelevant and we need $\boldsymbol{\lambda}^{\mathrm{T}} \mathbf{w}_i = \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{y}_i = 1$ for $\det(A_i) = 1$. These are the first two cases in Eq. (14). Defining affine hyperplanes $\mathcal{X}_i^{(c)} = \{\boldsymbol{\lambda} \in \mathbb{F}_2^d \mid \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{x}_i = c\}$ for $c \in \mathbb{F}_2$, and similarly $\mathcal{Z}_i^{(c)}$, $\mathcal{W}_i^{(c)}$, and $\mathcal{Y}_i^{(c)}$, we can rephrase these two cases as $\boldsymbol{\lambda}$ being contained in the affine subspace

$$\mathcal{A}_i = \mathcal{X}_i^{(0)} \cap \mathcal{W}_i^{(1)} \cap \mathcal{Y}_i^{(1)}. \tag{15}$$

For the middle two cases in Eq. (14), the roles of (x, z) and (w, y) are interchanged: these cases are equivalent to $\boldsymbol{\lambda}$ being contained in the affine subspace

$$\mathcal{B}_i = \mathcal{W}_i^{(0)} \cap \mathcal{X}_i^{(1)} \cap \mathcal{Z}_i^{(1)}. \tag{16}$$

Similarly, the remaining two cases in Eq. (14) lead to

$$\mathcal{C}_i = \mathcal{X}_i^{(1)} \cap \mathcal{Z}_i^{(0)} \cap \mathcal{W}_i^{(1)} \cap \mathcal{Y}_i^{(1)} \tag{17}$$

$$\text{and } \mathcal{D}_i = \mathcal{X}_i^{(1)} \cap \mathcal{Z}_i^{(1)} \cap \mathcal{W}_i^{(1)} \cap \mathcal{Y}_i^{(0)}, \tag{18}$$

respectively. Finally, by introducing

$$\mathcal{L}_i = \mathcal{A}_i \cup \mathcal{B}_i \cup \mathcal{C}_i \cup \mathcal{D}_i, \tag{19}$$

we obtain that $\det(A_i) = 1$ is equivalent to $\boldsymbol{\lambda} \in \mathcal{L}_i$. We defer further technical details about how to best implement the efficient probabilistic solver based on Gaussian elimination as well as additional mathematical insights for simplifying the problem to the methods section.

Reformulation as an Optimization Problem

It is possible to restate Eq. (6) as the feasibility problem of an *integer quadratically constrained program* (IQP), a special case of a mixed integer quadratically constrained program (MIQCP) which can be tackled with powerful numerical solvers such as Gurobi [45]. The problem instance is encoded into the binary matrices $R, S \in \mathbb{F}_2^{n\times m}$, which host the exponent vectors of the Pauli operators $P_1, \ldots, P_m$ to be diagonalized, and into the connectivity graph Γ_{con} with e edges. A straightforward albeit computationally expensive procedure is the following, which we refer to as `naive numerical solver`:

- Introduce $4n$ free binary (Boolean) variables $\mathbf{a} = (a_1^{xx}, \ldots, a_n^{zz})$ as vectorizations of matrices $A^{xx}, \ldots, A^{zz}$ as stated in Eq. (8).
- Introduce e binary (Boolean) variables $\gamma_{i,j}$, one for each edge in the connectivity graph Γ_{con}. The final values of $\gamma_{i,j}$ determine whether or not the corresponding hardware-supported CZ gates are used.
- Introduce a matrix $N = (\nu_{j,k}) \in \mathbb{Z}^{n\times m}$ of $n \times m$ integer slack variables and define nm quadratic constraints, one for each entry in the matrix equation $\Gamma A^{xx} R + \Gamma A^{xz} S = A^{zx} R + A^{zz} S + 2N$. Note that the term $2N$ exploits Eq. (6) being defined over $\mathbb{F}_2^{n\times m}$, i.e., equality only needs to hold modulo 2.
- Define quadratic constraints, $a_i^{xx} a_i^{zz} + a_i^{zx} a_i^{xz} = 1$ for each $i \in \{1, \ldots, n\}$, to ensure invertibility as a determinant constraint. Here, there is no need to introduce additional integer slack variables that account for the fact that $\det(A_i) = 1$ is supposed to hold modulo 2 because 1 is the only odd value that the expression $a_i^{xx} a_i^{zz} + a_i^{zx} a_i^{xz}$ can possibly take.
- Specify a target function to be minimized, e.g., the number of two-qubit gates in the final circuit, $\text{cost}(\mathbf{a}, \Gamma) = \sum_{i<j} \gamma_{i,j}$.

In general, IQP is computationally hard in worst case complexity: it is in NP, while the decision version is NP-complete [46]. We can reduce the complexity of the naive IQP above by leveraging our mathematical insights from the previous subsection. Treating only one fixed subgraph $\Gamma \subset \Gamma_{\text{con}}$ at a time, we can compute $Q_1, \ldots, Q_n$ via Eq. (13). Here, however, we regard the entries of Q_i as real numbers rather than equivalence classes of integers modulo 2. As a direct alternative to intersecting affine subspaces, we can attempt to find a solution $\boldsymbol{\lambda} \in \mathcal{L}$ by executing the following IQP which we call `informed numerical solver`:

- Introduce d binary variables $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_d)$ as well as n integer slack variables $\mu_1, \ldots, \mu_n$.
- Define n quadratic constraints, $\boldsymbol{\lambda}^{\mathrm{T}} Q_i \boldsymbol{\lambda} = 1 + 2\mu_i$.
- If necessary, specify a trivial target function.

Note that, by considering only one subgraph Γ at a time, the quadratic constraints in Eq. (6) become linear constraints over Boolean variables. This allows us to efficiently identify the ambient space in Fig. 2, however, the constraints ensuring invertibility of A still define quadratic constraints, which can be cast into the form of an IQP. The structured problems encountered here feature only n quadratic constraints and can be solved in practice for larger system sizes. Empirically, we will demonstrate later in Fig. 5 that highly-optimized, commercially available MIQCP solvers constitute a viable alternative to our provably-efficient algebraic solver.

Solver Overview

For an overview of all solvers, see Tab. III. In principle, each solver (except for `naive num.`) can be executed in an exhaustive mode, where all $s(n) = 2^e$ subgraphs of Γ_{con} instead of only $s(n) = \text{poly}(n)$ are probed. In the exhaustive mode, every solver has an exponential runtime which limits their range of applicability. For small system sizes (e.g., $n \lesssim 12$ in Fig. 5), we recommend using the `exhaustive, informed numerical solver` as it performs best in practice. For larger system sizes, we recommend either the `restricted, informed numerical solver`, which can be very fast, or the `restricted algebraic solver`, whose runtime can be easily controlled, see methods section "Runtime Analysis". Hereby, the hyperparameter $s(n)$ and, if applicable, also $c(n)$, should be selected appropriately, see Fig. 5, methods, and SM Sec. VII for further details and examples.

Experimental Demonstration

Next we demonstrate that our theoretical framework can enhance the efficiency of today's state-of-the-art

name	runtime	advantage
`BF`	exp.	yields conclusive answer
`restr. BF`	poly.	-
`exh. alg.`	exp.	as BF but often faster
`restr. alg.`	poly.	performs well in practice
`naive num.`	limited by IQP	CZ count minimization
`informed num.`	limited by IQP	faster than `naive num.`

TABLE III. Overview of our solvers for Equation (6). The `brute-force (BF) solver` loops over all $6^n 2^e$ choices of (A, Γ) and has an exponential (exp.) runtime in n. When restricted to a polynomial subset of choices for (A, Γ), the `restricted (restr.) BF solver` has a polynomial (poly.) runtime but a vanishingly low success probability. The `exhaustive (exh.) algebraic (alg.) solver` loops over all 2^e subgraphs of Γ_{con} and, in the worst case, through all 4^n intersections in Fig. 2; it can be fast as it terminates prematurely when either a solution $\boldsymbol{\lambda} \in \mathcal{L}$ is found or $\mathcal{L} = \varnothing$ is concluded. The `restricted algebraic solver` loops over a random selection of $s(n) = \text{poly}(n)$ subgraphs of Γ_{con} and employs a cutoff $c(n)$ as described in the methods section; it performs well in practice (see Fig. 5) and has poly. runtime by design. The `naive numerical (num.) solver` leverages an IQP solver with binary variables $\mathbf{a}, \Gamma$ and integer slack variables $\nu_{i,j}$ as described in the main text; it is not implemented here due to the large number of variables. The `informed num. solver` reduces the number of variables by leveraging our knowledge about the geometry of Eq. (6); this solver loops over a random selection of $s(n) = \text{poly}(n)$ subgraphs, for each of which it solves an IQP problem with binary variables λ_j and integer slack variables μ_i as described in the main text. For both numerical solvers, the runtime is limited by the leveraged IQP solver which can be fast in practice.

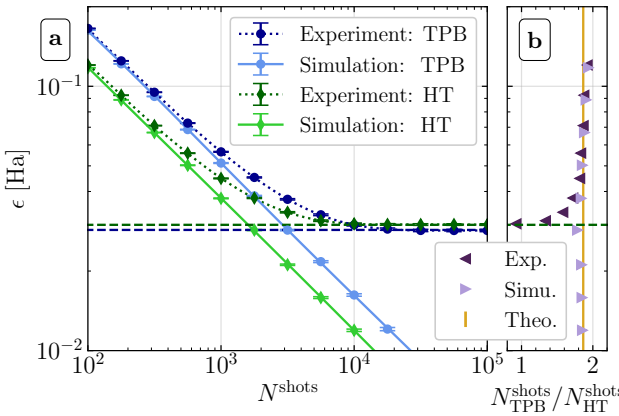

FIG. 3. Experimental shot reduction of HT readout circuits over TPBs. The experiment is conducted with eight superconducting qubits on *ibm_washington* and compared to an ideal Qiskit simulation [47]. **(a)** Error ϵ (see main text) as a function of the total number of optimally-allocated shots. Error bars show the error on the mean of ϵ averaged over $\lfloor 5 \times 10^7 / N^{\text{shots}} \rfloor$ independent repetitions of the experiment. **(b)** Ratio $N^{\text{shots}}_{\text{TPB}}/N^{\text{shots}}_{\text{HT}}$ of shots needed to estimate $\langle O \rangle$ up to a total error of ϵ. In the absence of implementation errors (Theo.), this ratio is independent of ϵ and given by $R_{\text{HT}}/R_{\text{TPB}} \approx 1.87$. See SM Sec. XIII for further details.

quantum computers in practice. A central task is to estimate the expectation value $\langle O \rangle = \text{Tr}[\rho O]$ for an observable O of interest and a state ρ prepared on the quantum computer. For a fixed shot budget (total number of available circuit executions), the goal is to estimate $\langle O \rangle$ as accurately as possible. As a concrete example, we consider an eight-qubit molecular Hamiltonian $O = \sum_{i=1}^{M} c_i P_i$ with $M = 184$ Pauli operators that represents a four-atomic linear hydrogen chain, see SM Sec. XIII for details. To enable a fair comparison of different estimation procedures, we restrict ourselves to the paradigm of non-overlapping Pauli groupings with reliably-executable readout circuits [25]. To the best of our knowledge, the previous state-of-the-art method within this paradigm is the collection of single-qubit readout circuits that arise from applying the *Sorted Insertion* algorithm [22] that groups $\{P_1, \ldots, P_{184}\}$ into, in this case, $N^{\text{circs}}_{\text{TPB}} = 35$ QWC subsets. As an alternative, we propose to use $N^{\text{circs}}_{\text{HT}} = 10$ HT readout circuits which allow us to measure the same $M = 184$ Pauli operators and which contain no more than four linearly-connected CZ gates. This reduction in the number of readout circuits is possible because being jointly-HT diagonalizable is a less stringent requirement than QWC. What matters in the end, however, is the error $\epsilon = |E_{\text{exp}} - E_{\text{ideal}}|$ by which the experimentally measured energy E_{exp} differs from the ideal one. Hereby, one should minimize ϵ by optimally distributing the available shots among the $N^{\text{circs}}_{\text{TPB}} = 35$ or $N^{\text{circs}}_{\text{HT}} = 10$ readout circuits [22], see SM Sec. IV. To enable a pronounced comparison between the two readout methods, we select a product target state vector $|\Psi\rangle = |\psi_1\rangle \otimes \ldots \otimes |\psi_8\rangle$ that can be prepared with a high fidelity. This state has a large energy $E_{\text{ideal}} = \langle \Psi | O | \Psi \rangle = -0.49264\,\text{Ha}$ compared to the ground state energy, $\min_{\Phi} \langle \Phi | O | \Phi \rangle = -2.26752\,\text{Ha}$, where all reported energies include Coulomb repulsion between the nuclei. We perform experiments for both readout methods and present the results in Fig. 3. The error ϵ is plotted in Fig. 3 **a** as a function of the shot budget N^{shots} for both TPB (blue circles) and HT (green diamonds). We compare the experimental results (dark) to classical, noise-free simulations (bright). In Fig. 3 **b** we display the shot reduction ratio $N^{\text{shots}}_{\text{TPB}}/N^{\text{shots}}_{\text{HT}}$ for achieving a target error ϵ that our HT circuits offer over conventional TPBs. For very low budgets of a few hundred shots in total, we see that the experimental data perfectly agree with the simulation. This is because sampling errors dominate here. While the simulated errors generally decrease as $1/\sqrt{N^{\text{shots}}}$, the experimental errors eventually saturate at a finite bias b stemming mainly from noise in the diagonalization circuits and the readout-error-mitigated [48] Z-measurements. Unsurprisingly, $b_{\text{HT}} = 0.0299(2)\,\text{Ha}$ is larger than $b_{\text{TPB}} \approx 0.0284(2)\,\text{Ha}$. For shot budgets below 10^4, however, we observe that ϵ is smaller for HT than for TPB. Hence, our HT readout

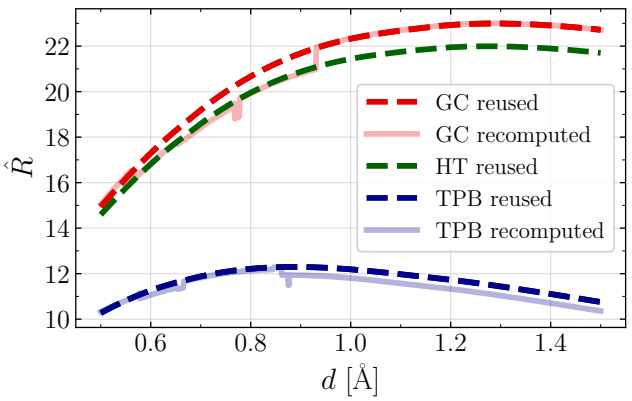

FIG. 4. Reusability opportunity. Estimated shot reduction $\hat{R}$ defined in Ref. [22] for measuring the energy $\langle O \rangle$ of a hydrogen chain Hamiltonian as a function of the interatomic distance d between the hydrogen atoms. Due to continuity, there is no need to regroup the Pauli operators in O when d is updated. This enables enormous savings in preprocessing cost.

circuits outperform TPBs in the low-shot regime, which is of practical importance, see SM Sec. I.

DISCUSSION

In the remainder of this article, we discuss the reusability potential of HT Pauli groupings, investigate the performance of our solvers, and highlight further potential use cases for our theoretical framework.

Every grouping of a fixed set $\{P_1, \ldots, P_M\}$ into jointly-diagonalizable subsets has vast reusability potential: for a fixed observable $O = \sum_{i=1}^{M} c_i P_i$, one can estimate $\mathrm{Tr}[\rho O]$ for countless experimental states ρ, e.g., different eigenstates of O as well as the myriad of VQE trial states that are encountered until such eigenstates are found. Similarly, in simulations of chemical reactions, $\mathrm{Tr}[\rho(t) O]$ is mapped out for a time series of quantum states $\rho(t)$.

Moreover, for any linear combination $O' = \sum_{i=1}^{M} c'_i P_i$ of the same Pauli operators, it is possible to estimate $\langle O' \rangle$ with the same readout circuits as for $\langle O \rangle$. Therefore, a single Pauli grouping can suffice to map out low-energy Born-Oppenheimer surfaces via, e.g., VQE. At the same time, we are guaranteed that the quality of the reused grouping does not deteriorate abruptly as the estimated shot reduction $\hat{R}$ depends continuously on the nuclear coordinates [22].

We illustrate this fact in Fig. 4 for the example of an eight-qubit Hamiltonian $O(d) = \sum_{i=1}^{184} c_i(d) P_i$ that represents a four-atomic linear hydrogen chain for which the interatomic spacing d is varied. For different Pauli groupings of $O(d)$, we plot $\hat{R}$, which should be regarded as a state-independent estimate of the shot reduction ratio $N_{\mathrm{GPM}}^{\mathrm{shots}} / N_{\mathrm{IPM}}^{\mathrm{shots}}$ (cf. Fig. 3 **b**), where $N_{\mathrm{GPM}}^{\mathrm{shots}}$ and $N_{\mathrm{IPM}}^{\mathrm{shots}}$, respectively, denote the number of shots required to measure $\langle O(d) \rangle$ to a fixed precision ϵ if grouped Pauli measurements (GPM) and individual Pauli measurements (IPM) are performed. For example, $O(d = 1.0\,\text{Å})$ coincides with the Hamiltonian from Fig. 3 and the estimated shot reduction ratio of HT over TPB, $\hat{R}_{\mathrm{HT}} / \hat{R}_{\mathrm{TPB}} \approx 1.76$, is close to the state-dependent value, $R_{\mathrm{HT}} / R_{\mathrm{TPB}} \approx 1.87$. As explained above, we can see in Fig. 4 that $\hat{R}$ depends continuously on d if a fixed Pauli grouping is reused (dark dashed lines). Also shown is the piecewise continuous dependence of $\hat{R}_{\mathrm{GC}}$ and $\hat{R}_{\mathrm{TPB}}$ if a new Pauli grouping is recomputed for every value of d (bright solid lines), where the jumps unveil the values of d at which the output of the Sorted Insertion algorithm changes. Since Sorted Insertion is a greedy algorithm, recomputing the Pauli grouping can even diminish the value of $\hat{R}$.

Finally, note that Pauli groupings can be reused for numerous other applications in the context of parameter-shift rules [16, 17, 49, 50]. It is straightforward to adapt Sorted Insertion to HT Pauli groupings, see SM Sec. V. Instead of checking if a set of Pauli operators (qubit-wise) commutes, one has to construct a HT readout circuit using one of the solvers from Tab. III. In Fig. 5, we investigate the performance of our various solvers by applying a modified Sorted Insertion algorithm to three classes of paradigmatic Hamiltonians. In all cases, we observe in Fig. 5 **a-c** that the estimated shot reduction ratio, $\hat{R}_{\mathrm{HT}} / \hat{R}_{\mathrm{TPB}}$, takes values in between 1.3 and 3.5. Therefore, our HT readout circuits consistently outperform conventional TPBs.

While Sorted Insertion has a runtime of $\mathcal{O}(M^2 n)$, the runtime of our adaptation is given by $\mathcal{O}(M^2 f(n))$, where M still denotes the number of Pauli operators in the observable O, and $f(n)$ is the complexity of the selected solver, e.g., $f(n) = \mathcal{O}(2^e 6^n n^3)$ for the `brute-force algebraic solver` and $f(n) = \mathcal{O}(\mathrm{poly}(n))$ for the `restricted algebraic solver`, see methods section "Runtime Analysis". For molecular hydrogen chain Hamiltonians for example, two polynomial fits (green dashed lines) in Fig. 5 **d** reveal total runtimes $t^{(1)} \propto n^{9.8}$ and $t^{(2)} \propto n^{9.4}$. Since the number of Pauli operators is here given by $M \approx 0.17 \times n^{3.68}$, we can conclude that the `restricted algebraic solver` has an empirical runtime $f(n)$ lying somewhere in between $\mathcal{O}(n^{2.0})$ and $\mathcal{O}(n^{2.4})$, which is in agreement with the expected runtime of $f(n) = \mathcal{O}(n^3)$, see methods.

For the example of random Hamiltonians, we compare in Fig. 5 **b** the `algebraic solver` with the `informed numerical solver` from Tab. III, both in combination with an exhaustive (exh.) search over all 2^e subgraphs $\Gamma \subset \Gamma_{\mathrm{con}}$. We find that both solvers produce HT Pauli groupings of the same quality, which we believe to be near optimal for the assumed linear connectivity constraint. By construction, both exhaustive solvers are inefficient, however, we observe in Fig. 5 **e** that the `numerical solver` is much faster in practice. This is unsurprising as this option leverages a highly-optimized MIQCP solver.

Since the `exhaustive numerical solver` is our best option for constructing near-optimal HT Pauli groupings,

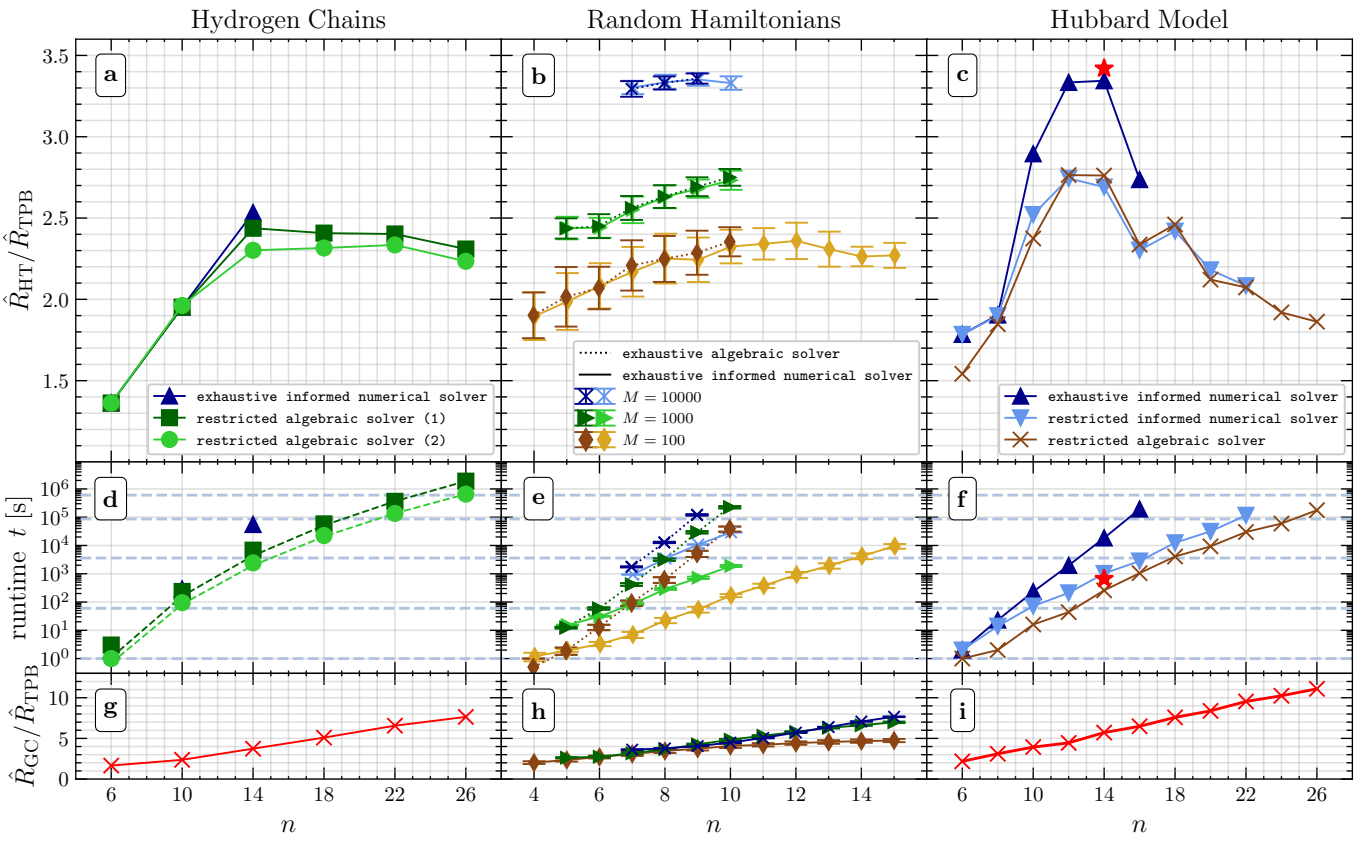

FIG. 5. Performance of a greedy Pauli grouping algorithm with various solvers from Tab. III. **(a-c)** Estimated shot reduction $\hat{R}_{\mathrm{HT}}/\hat{R}_{\mathrm{TPB}}$ for three classes of n-qubit Hamiltonians O. The HT readout circuits assume a linear hardware connectivity. Different curves correspond to different solvers from Tab. III and, in the case of random Hamiltonians (**b,e,h**), to different numbers M of Pauli operators in O. For the single data point with the red star symbol (★) at $n = 14$ in panels **c** and **f**, we use the `restr. alg. solver` with a fine-tuned choice of hyperparameters. **(d-f)** Runtime of our HT Pauli grouping algorithm. **(g-i)** Estimated shot reduction $\hat{R}_{\mathrm{GC}}/\hat{R}_{\mathrm{TPB}}$ that is in principle achievable with unrestricted Clifford circuits if two-qubit errors are neglected, which is an unrealistic assumption for near-term quantum devices. Therefore, HT readout circuits present the best viable option in this comparison. See methods for further details.

we can now turn to the question about how much the quality (as measured by $\hat{R}_{\mathrm{HT}}$) of the HT Pauli grouping decreases when we replace the `brute-force solver` by our provably-efficient `restricted algebraic solver`. For the example of the Hubbard model we observe in Fig. 5 **c** that both the `restricted numerical solver` and the `restricted algebraic solver` produce HT Pauli groupings with the same qualitative behavior of $\hat{R}$ as the (near-optimal) `exhaustive numerical solver`. For all solvers, $\hat{R}$ first increases with the number of qubits n, before it drops again. We attribute this saturation effect to the linear hardware connectivity constraint because $\hat{R}_{\mathrm{GC}}/\hat{R}_{\mathrm{TPB}}$ continues to grow as shown in Fig. 5 **i** (also see Fig. 1 in SM Sec. II). Whether a similar growth is retained with 2-dimensional HT readout circuits deserves further investigation. For $n = 14$, the `exhaustive numerical solver` constructs a grouping with $\hat{R}_{\mathrm{HT}}/\hat{R}_{\mathrm{TPB}} \approx 3.34$ after 5 hours. By carefully balancing the hyperparameters of the `restricted algebraic solver` (see methods), we find an even better grouping with $\hat{R}_{\mathrm{HT}}/\hat{R}_{\mathrm{TPB}} \approx 3.42$ in only 11 minutes (red star).

The discussion above shows that the classical bottleneck of our HT Pauli grouper can be overcome. To corroborate this claim, we propose and test a rudimentary HT Pauli grouper (Algorithm 2 in SM Sec. VI) for which the enabled runtime savings outweigh the classical preprocessing cost for a 52-qubit example. This demonstrates that the advantage of HT readout circuits can be scaled up to meaningful system sizes. Finally, let us point out that our approach also works for more sophisticated molecular basis sets, see SM Sec. XIV.

In this article, we have introduced a theoretical framework for the construction of diagonalization circuits that can be tailored to any given hardware connectivity. Our starting point was the observation of the fact that every set of commuting Pauli operators can be cast into the stabilizer group of a stabilizer state which is local-Clifford equivalent to a graph state. The Pauli operators can thus be diagonalized with a quantum circuit that

completely avoids SWAP gates whenever this graph state matches the connectivity of the quantum computer. We derived an algebraic criterion for the existence of such *hardware-tailored* (HT) diagonalization circuits and introduced solvers for their construction. An important empirical observation is that in many cases it is not necessary to apply unconstrained diagonalization circuits because also HT circuits can be constructed. For super- and semiconducting chips, for which the connectivity graph has a bounded degree, HT circuits have a constant depth. In comparison to state preparation circuits such as UCCSD, which have a gate count of $\mathcal{O}(n^4)$ and a gate depth of $\mathcal{O}(n^3)$, the circuit complexity of HT readout circuits is therefore negligible [15]. The construction of HT circuits can be computationally demanding but is worthwhile given their reusability potential. Finally, we have demonstrated the advantage of our approach over previous ones both in theory and in experiment.

There are multiple ways to further improve the efficiency of our solvers for the construction of HT diagonalization circuits, see methods section "Ideas for Improving our Solvers". Furthermore, it would be worthwhile to combine the paradigm of HT readout circuits with complementary state-of-the-art methods for lowering shot requirements such as iterative measurement allocation and iterative coefficient splitting [26].

Lowering shot requirements, e.g., in variational algorithms as discussed above, is just one of many potential applications for HT diagonalization circuits. Other, equally important use cases arise in the contexts of classical shadows [29], Hamiltonian time simulation [51–53], and quantum error correction [54]. For instance, an encoding circuit for a stabilizer quantum error-correcting code is the same as a time-reversed diagonalization circuit for its stabilizer group. For an outline how Hamiltonian exponentiation could potentially benefit from our HT diagonalization circuits, see SM Sec. IX.

METHODS

Choices of Hamiltonians

All molecular hydrogen chain Hamiltonians were computed in the STO-3G minimal basis using Qiskit nature [47], either in combination with pyquante [55] or PySCF [56]. More sophisticated basis sets are discussed in SM Sec. XIV. For Tab. II, Fig. 3, and Fig. 4, we use pyquante and the Bravyi-Kitaev fermion-to-qubit mapper to obtain n-qubit Hamiltonians representing a hydrogen chain with $\frac{n}{2}$ nuclei at an equidistant spacing of d, where we fix $d = 1.0\,\text{Å}$ in Tab. II and Fig. 3. The exact numbers of Pauli operators $M(n)$ of the n-qubit Hamiltonians in Tab. II are given by $M(20) = 7,150$, $M(40) = 116,594$, $M(60) = 594,954$, $M(80) = 1,886,542$, $M(100) = 4,611,050$, and $M(120) = 9,559,318$. For the hydrogen chain Hamiltonians in the left column of Fig. 5, on the other hand, we use PySCF and the parity-encoding fermion-to-qubit mapper, resulting in Hamiltonians with only $n = 2n_{\text{at}} - 2$ qubits, where n_{at} is the number of hydrogen atoms; also here the interatomic spacing is chosen as $d = 1.0\,\text{Å}$.

In Fig. 5 **b**, we average over twenty random Hamiltonians of the form $O = \sum_{i=1}^{M} c_i P_i$ for each choice of qubit number n and number of Pauli operators M. For every O, the Pauli operators $P_i \in \{I, X, Y, Z\}^{\otimes n} \backslash \{I^{\otimes n}\}$ and coefficients $c_i \in [-1, 1]$ are drawn uniformly at random. Error bars show one standard deviation. For an in-depth investigation of the results, see SM Sec. VIII.

Figure 5 **c** features the Hubbard model of L fermionic modes $\hat{c}^\dagger_{k_j,\sigma}$ with momentum $k_j = 2\pi j/L$ and spin σ on a 1-dimensional lattice with L sites and periodic boundary conditions. Its Hamiltonian is given by

$$O = \frac{U}{L} \sum_{i,j,l=1}^{L} \hat{c}^\dagger_{k_i - k_l,\uparrow} \hat{c}^\dagger_{k_j + k_l,\downarrow} \hat{c}_{k_j,\downarrow} \hat{c}_{k_i,\uparrow} + \sum_{j=1}^{L} \sum_{\sigma \in \{\uparrow,\downarrow\}} \epsilon_{k_j} \hat{c}^\dagger_{k_j,\sigma} \hat{c}_{k_j,\sigma} \tag{20}$$

where $U \geq 0$ is the Coulomb energy, $\epsilon_{k_j} = -2t\cos(k_j)$ is the dispersion relation for non-interacting ($U = 0$) fermions, and $t \geq 0$ is the hopping strength [57]. We use the block-spin Jordan-Wigner encoding to convert this Hamiltonian into a linear combination of Pauli operators. In Fig. 5, we assume $U = 1$ and $t = 1$ to obtain concrete values for $\hat{R}$.

Choices of Hyperparameters

Here we report the choices of hyperparameters of our solvers that were used to create Fig. 5. In all cases, we tailor the readout circuits to a linear connectivity by applying Algorithm 1 from SM Sec. V. The total runtime and the quality of the resulting Pauli grouping depends on the choice of the solver from Tab. III and its hyperparameters. Unless specified otherwise, all computations were carried out on an 18 core Intel Xeon CPU E5-2697 v4 @2.30GHz device [58].

The `exhaustive informed numerical solver` (dark blue triangles in Fig. 5 **d**,**f** and dark markers with dotted lines in Fig. 5 **e**) searches over all 2^{n-1} circuit templates $\Gamma \subset \Gamma_{\text{con}}$ and constructs readout circuits (one for each subgraph Γ) by solving the informed IQP problem (if possible). Then, the best circuit is chosen to assign a collection of Pauli operators to a joint measurement, see SM Sec. V for further details. The `exhaustive algebraic solver` (bright markers with solid lines in Fig. 5 **e**) similarly searches over all 2^{n-1} circuit templates but applies the `algebraic brute-force solver` instead of the `informed numerical solver`. Both exhaustive solvers have an exponential runtime because they take all subgraphs $\Gamma \subset \Gamma_{\text{con}}$ into account.

The `restricted numerical solver` has one hyperparameter: the number $s(n)$ of subgraphs $\Gamma \subset \Gamma_{\text{con}}$

that are probed. In our current implementation, the specific choice of subgraphs is taken uniformly at random. For the Hubbard model (Fig. 5 **f**), we work with $s(n) = \min\{1.5n^2 - 0.5n, 2^{n-1}\}$ random subgraphs.

The `restricted algebraic solver` has two hyperparameters: the number $s(n)$ of random subgraphs and the cutoff $c(n)$ defined before Eq. (27). For the Hubbard model (Fig. 5 **f**), we work with $s(n) = \min\{n^2, 2^{n-1}\}$ and $c(n) = \lfloor \log_2(n) \rfloor$, except for the fine-tuned parameter choice (red star) where the number of subgraphs is increased from $s(n) = n^2 = 196$ to $s = 1000$. For hydrogen chains (Fig. 5 **d**), on the other hand, we use two different choices of constant hyperparameters:

(1) $s(n) = 2000$ and $c(n) = 5$.

(2) $s(n) = 1000$ and $c(n) = 3$.

Besides HT Pauli groupings, we also compute GC Pauli groupings in Figs. 4 and 5 and groupings of O into TPBs in Figs. 3–5. These groupings were obtained by applying the Sorted Insertion algorithm [22] with the insertion conditions "commute" and "qubitwise commute", respectively.

For additional numerical investigations about how the choice of hyperparameters influences the performance of the HT Pauli grouping algorithm, see SM Secs. VII and XIV.

Technical Details on the Algebraic Solver

Here we continue our technical discussion about how to algebraically construct HT diagonalization circuits. For every qubit $i \in \{1, \dots, n\}$, there are three cases how the hypersurface $\mathcal{L}_i = \{\boldsymbol{\lambda} \in \mathbb{F}_2^d \mid \boldsymbol{\lambda}^{\mathrm{T}} Q_i \boldsymbol{\lambda} = 1\}$ could look like: since the image of the $\mathbb{F}_2$-linear map defined by the matrix Q_i is contained in the span of $\mathbf{x}_i$ and $\mathbf{w}_i$, the rank of Q_i can only take one of the three values: 0, 1, and 2. If $\mathrm{rank}_{\mathbb{F}_2}(Q_i) = 0$, i.e., $Q_i = 0$, then $\mathcal{L}_i = \varnothing$ is empty, which implies $\mathcal{L} = \bigcap_{i=1}^n \mathcal{L}_i = \varnothing$ and proves the non-existence of a HT diagonalization circuit for the fixed choice of $P_1, \dots, P_m$ and Γ. On the other hand, if $\mathrm{rank}_{\mathbb{F}_2}(Q_i) = 1$, e.g., $\mathbf{x}_i \mathbf{z}_i^{\mathrm{T}} = 0$ but $\mathbf{w}_i \mathbf{y}_i^{\mathrm{T}} \neq 0$, then we need $\boldsymbol{\lambda}^{\mathrm{T}} \mathbf{w}_i \mathbf{y}_i^{\mathrm{T}} \boldsymbol{\lambda} = 1$, which is equivalent to $\boldsymbol{\lambda}^{\mathrm{T}} \mathbf{w}_i = \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{y}_i = 1$. Thus, this case is degenerate and the hypersurface collapses to an affine subspace

$$\mathcal{L}_i = \mathcal{W}_i^{(1)} \cap \mathcal{Y}_i^{(1)}. \tag{21}$$

For a detailed explanation about how to compute intersections of affine subspaces numerically, see SM Sec. X. Finally, in the most general case of $\mathrm{rank}_{\mathbb{F}_2}(Q_i) = 2$, the hypersurface $\mathcal{L}_i$ defies a description simpler than the union of four affine subspaces as in Eq. (19). As we show next, the occurrence of rank-1 hypersurfaces can greatly simplify the problem.

Denote the number of rank-2 hypersurfaces by k. In our search for a point $\boldsymbol{\lambda} \in \mathcal{L}$, we first compute the intersection of all rank-1 hypersurfaces $\mathcal{L}_i$. After potentially relabelling some of the qubits, we can assume $Q_i = \mathbf{x}_i \mathbf{z}_i^{\mathrm{T}}$ and $Q_j = \mathbf{w}_j \mathbf{y}_j^{\mathrm{T}}$ for all $i \in \{1, \dots, l\}$ and $j \in \{l+1, \dots, n-k\}$. Then, the intersection of all rank-1 quadric hypersurfaces is given by

$$\begin{aligned} \bigcap_{i=1}^{n-k} \mathcal{L}_i &= \left\{ \boldsymbol{\lambda} \in \mathbb{F}_2^d \;\middle|\; \begin{array}{l} \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{x}_i = \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{z}_i = \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{y}_j = \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{w}_j = 1 \\ \text{for all } 1 \leq i \leq l < j \leq n-k \end{array} \right\} \\ &= \left\{ \boldsymbol{\lambda} \in \mathbb{F}_2^d \;\middle|\; C\boldsymbol{\lambda} = \mathbf{1} \right\}, \end{aligned} \tag{22}$$

where $\mathbf{1} = [1, \dots, 1]^{\mathrm{T}}$ and for the $(2(n-k) \times d)$-matrix $C = [\mathbf{x}_1, \mathbf{z}_1, \dots, \mathbf{x}_l, \mathbf{z}_l, \quad \mathbf{y}_{l+1}, \mathbf{w}_{l+1}, \dots, \mathbf{y}_{n-k}, \mathbf{w}_{n-k}]^{\mathrm{T}}$ whose rows are given by the row vectors $\mathbf{x}_1^{\mathrm{T}}$ etc. Using Gaussian elimination over $\mathbb{F}_2$, we can compute the reduced row-echelon form (RREF) of the extended matrix $[C, \mathbf{1}] \in \mathbb{F}_2^{2(n-k) \times (l+1)}$. If the last column of the RREF is a pivot column, we are in the solutionless case $\mathcal{L} = \varnothing$. Otherwise, this column is an offset vector for $\mathcal{L}_1 \cap \ldots \cap \mathcal{L}_{n-k}$, while every non-pivot column of the RREF of C yields a basis vector in the standard way of linear algebra, see SM Sec. X for technical details.

Finally, we turn to the computationally most demanding part that deals with the rank-2 hypersurfaces $\mathcal{L}_{n-k+1}, \dots, \mathcal{L}_n$. Similar to the matrix C in Eq. (22), we introduce a $(4k \times d)$-matrix $C' = [\mathbf{x}_{n-k+1}, \mathbf{z}_{n-k+1}, \mathbf{w}_{n-k+1}, \mathbf{y}_{n-k+1}, \dots, \mathbf{x}_n, \mathbf{z}_n, \mathbf{w}_n, \mathbf{y}_n]^{\mathrm{T}}$, i.e., for every $j \in \{1, \dots, k\}$, the four rows of C' from row $4j-3$ to row $4j$ are given by $\mathbf{x}_{n-k+j}^{\mathrm{T}}$, $\mathbf{z}_{n-k+j}^{\mathrm{T}}$, $\mathbf{w}_{n-k+j}^{\mathrm{T}}$, and $\mathbf{y}_{n-k+j}^{\mathrm{T}}$. Then, a vector $\boldsymbol{\lambda} \in \mathbb{F}_2^d$ is contained in $\mathcal{L}_{n-k+1} \cap \ldots \cap \mathcal{L}_n$ if and only if $C'\boldsymbol{\lambda} = \mathbf{b}_{i_1} \oplus \ldots \oplus \mathbf{b}_{i_k}$ for some $i_1, \dots, i_k \in \{1, \dots, 6\}$, where the vectors $\mathbf{b}_1, \dots, \mathbf{b}_6$ in the direct sum

$$\mathbf{b}_{i_1} \oplus \ldots \oplus \mathbf{b}_{i_k} = \begin{bmatrix} \mathbf{b}_{i_1} \\ \vdots \\ \mathbf{b}_{i_k} \end{bmatrix} \tag{23}$$

are the six vectors from Eq. (14), i.e., $\mathbf{b}_1 = [0, 0, 1, 1]^{\mathrm{T}}$ etc. If it is our goal to unambiguously ascertain whether or not $\mathcal{L}$ is empty, we have to check an exponential number of cases. This is only feasible for small qubit numbers n or for graphs with small components, as we explain in SM Sec. XII. To save computing time, we treat all combinations of the vectors $\mathbf{b}_i$ at once by introducing (4×6^j)-matrices

$$B_j = [\underbrace{\mathbf{b}_1, \dots, \mathbf{b}_1}_{6^{j-1} \text{ times}}, \dots, \underbrace{\mathbf{b}_6, \dots, \mathbf{b}_6}_{6^{j-1} \text{ times}}] \tag{24}$$

for $j \in \{1, \dots, k\}$ and using them as blocks for enlarging C' to a matrix $B \in \mathbb{F}_2^{4k \times (d+6^k)}$. Hereby, the four rows from row $(4j-3)$ to row $4j$ of B are given by

$$\left[\begin{array}{l} \mathbf{x}_{n-k+j}^{\mathrm{T}} \\ \mathbf{z}_{n-k+j}^{\mathrm{T}} \\ \mathbf{w}_{n-k+j}^{\mathrm{T}} \\ \mathbf{y}_{n-k+j}^{\mathrm{T}}, \end{array} \quad \underbrace{B_j, \dots, B_j}_{6^{k-j} \text{ times}} \right] \in \mathbb{F}_2^{4 \times (d+6^k)}. \tag{25}$$

In other words, B is the matrix that arises from C' by appending all vectors of the form $\mathbf{b}_{i_1} \oplus \ldots \oplus \mathbf{b}_{i_k}$ as additional columns. Next, we use Gaussian elimination to bring B to a row-echelon form; here, time can be saved as it is not necessary to compute the RREF of B. This reveals the non-pivot columns of B. Every non-pivot column of the form $\mathbf{b}_{i_1} \oplus \ldots \oplus \mathbf{b}_{i_k}$ indicates the existence of at least one vector $\boldsymbol{\lambda} \in \mathcal{L}_{n-k-1} \cap \ldots \cap \mathcal{L}_n$. However, we are looking for a $\boldsymbol{\lambda}$ that also lies in the affine space $\mathcal{L}_1 \cap \ldots \cap \mathcal{L}_{n-k}$. To accomplish this, we start by computing a basis and an offset vector of the entire affine space

$$\left\{\boldsymbol{\lambda} \in \mathbb{F}_2^d \mid C'\boldsymbol{\lambda} = \mathbf{b}_{i_1} \oplus \ldots \oplus \mathbf{b}_{i_k}\right\}. \tag{26}$$

Then, we use the procedure explained in SM Sec. X to compute the intersection of the two affine subspaces in Eqs. (22) and (26). Since this results in a subset of $\mathcal{L}$, we can finish if we find a non-pivot column $\mathbf{b}_{i_1} \oplus \ldots \oplus \mathbf{b}_{i_k}$ in the right part of B for which this intersection is not empty. Otherwise, if this approach fails for all non-pivot columns, we can finally infer $\mathcal{L} = \varnothing$. In any case, we obtain a conclusive answer whether or not a layer of single-qubit Clifford gates exists such that the corresponding graph-based circuit (see main text, Fig. 1) diagonalizes the given set of commuting Pauli operators.

Restricting the Algebraic Solver

In the brute force approach of the previous subsection to algebraically solve Eq. (6) from the main text, we iterate through a number of affine subspaces that grows exponentially in the number $k \leq n$ of qubits i for which $\mathcal{L}_i$ is a rank-2 quadric hypersurface. For large problem sizes, this is infeasible and we can instead restrict the search to a smaller number $c(n) \leq k$ of rank-2 quadric hypersurfaces. For example, we can work with a constant cutoff

$$c(n) = \text{const.} \tag{27}$$

or with a logarithmically-growing cutoff

$$c(n) = \text{const.} \times \lfloor \log(n) \rfloor. \tag{28}$$

This restriction turns our `algebraic solver` (for attempting to find a solution $\boldsymbol{\lambda} \in \mathcal{L}$) into an efficient but probabilistic algorithm because only a polynomial number of intersections of affine subspaces will be probed, see methods section "Runtime Analysis" below. If no solution is found this way, we treat this case as if $\mathcal{L}$ was empty, i.e., we skip the current subgraph in our Pauli grouping algorithm from SM Sec. V.

We can incorporate the cutoff $c = c(n)$ by replacing the matrix $B \in \mathbb{F}_2^{4k \times (d+6^k)}$ in Eq. (25) by a smaller matrix $B' \in \mathbb{F}_2^{(4c+3(k-c)) \times (d+6^c)}$. The first $4c$ rows of B' are again given by Eq. (25), but with 6^{c-j} instead of 6^{k-j} blocks of the form B_j. For the remaining part, we set the three rows from row $(4c+3j-2)$ to row $(4c+3j)$ to

$$\begin{bmatrix} \mathbf{x}_{n-k+c+j}^{\mathrm{T}} & 0 & \cdots & 0 \\ \mathbf{w}_{n-k+c+j}^{\mathrm{T}} & 1 & \cdots & 1 \\ \mathbf{y}_{n-k+c+j}^{\mathrm{T}} & 1 & \cdots & 1 \end{bmatrix} \in \mathbb{F}_2^{3 \times (d+6^c)} \tag{29}$$

for all $j \in \{1, \ldots, k-c\}$, i.e., we dispose of the $\mathbf{z}$-row. In this way, we have effectively combined the two cases that correspond to $\mathbf{b}_1$ and $\mathbf{b}_2$ from Eq. (14). The remainder of our approach stays unchanged. By restricting from B to B', we will only be able to find solutions $\boldsymbol{\lambda} \in \mathcal{L}_{n-k+1} \cap \ldots \cap \mathcal{L}_n$ which are contained in the subspace

$$\left(\bigcap_{i=n-k+1}^{n-k+c} \underbrace{\mathcal{A}_i \cup \mathcal{B}_i \cup \mathcal{C}_i \cup \mathcal{D}_i}_{=\mathcal{L}_i} \right) \cap \bigcap_{i=n-k+c+1}^{n} \underbrace{\mathcal{A}_i}_{\subset \mathcal{L}_i}. \tag{30}$$

Let us illustrate the working principle of the cutoff c. In the hypothetical example of Fig. 2 in the main text, we have highlighted the affine subspaces

$$\mathcal{A}_i \subset \mathcal{L}_i = \mathcal{A}_i \cup \mathcal{B}_i \cup \mathcal{C}_i \cup \mathcal{D}_i \tag{31}$$

with an increased line width to distinguish them from $\mathcal{B}_i$, $\mathcal{C}_i$, and $\mathcal{D}_i$. In this example, the number of rank-2 hypersurfaces is $k = 3$. Hence, there are four possible choices for the cutoff $c \in \{0, \ldots, k\}$. For $c = k = 3$, we recover the original approach and are able to probe all six depicted intersection spaces (yellow dots). For $c = 2$, the restricted search space $\mathcal{L}_1 \cap \mathcal{L}_2 \cap \mathcal{A}_3$ only contains two non-empty intersection spaces, namely $\mathcal{B}_1 \cap \mathcal{A}_2 \cap \mathcal{A}_3$ and $\mathcal{D}_1 \cap \mathcal{C}_2 \cap \mathcal{A}_3$. For $c = 1$, only $\varnothing \neq \mathcal{B}_1 \cap \mathcal{A}_2 \cap \mathcal{A}_3 = \mathcal{L}_1 \cap \mathcal{A}_2 \cap \mathcal{A}_3$ remains. Note that neither $\mathcal{A}_1 \cap \mathcal{A}_2$ nor $\mathcal{A}_1 \cap \mathcal{A}_3$ nor $\mathcal{A}_2 \cap \mathcal{A}_3$ is empty, but $\mathcal{A}_1 \cap \mathcal{A}_2 \cap \mathcal{A}_3$ is. Therefore, we would not be able to find any solution for $c = 0$ in the example of Fig. 2 in the main text. For a non-hypothetical example, see SM Sec. XI.

Runtime Analysis

The `restricted algebraic solver` consists of an outer loop over $s(n) \leq 2^e$ subgraphs and an inner loop over $4^{c(n)}$ subspace intersections, where $s(n)$ and $c(n)$ are two hyperparameters that may explicitly depend on n. For each subgraph and each intersection, the `restricted algebraic solver` applies Gaussian elimiation to a binary matrix B' of size $(4c(n) + 3(k - c(n))) \times (d + 6^{c(n)})$, where $k \leq n$ and $d \leq 4n$. If we choose a constant cutoff $c(n) = \text{const.}$ as in Eq. (27), then the size of B' is proportional to $n \times n$. Hence, Gaussian elimination has a runtime of $\mathcal{O}(n^3)$, i.e., the `restricted algebraic solver` has a runtime of $\mathcal{O}(s(n) \times n^3)$. This is efficient if we choose $s(n) = \mathcal{O}(\text{poly}(n))$. In particular, for the hydrogen chain Hamiltonians in Fig. 5 where we work with constant values of $s(n)$, the solver should have a runtime of $f(n) = \mathcal{O}(n^3)$. This is consistent with our empirical estimate of a runtime that lies somewhere in between $\mathcal{O}(n^{2.0})$ and $\mathcal{O}(n^{2.4})$.

On the other hand, if we choose a logarithmically-growing cutoff $c(n) = c_0 \times \log_6(n)$, the inner loop of the `restricted algebraic solver` iterates over $6^{c(n)} = n^{c_0}$ many intersections. Furthermore, we apply Gaussian elimination of a matrix B' of size proportional to $n \times n^{c_0}$, which has a runtime of $\mathcal{O}(n^{2+c_0})$ assuming $c_0 \geq 1$. Hence, the overall runtime of the `restricted algebraic solver` follows as $f(n) = \mathcal{O}(s(n) \times n^{2c_0+2})$, which is efficient for $s(n) = \mathcal{O}(\text{poly}(n))$.

Ideas for Improving our Solvers

There is much room to further improve the runtime of both the solver for Eq. (6) and the Pauli grouping algorithm into which it is embedded:

1. Since our solvers are highly parallelizable, a more distributed software implementation would allow us to trade runtime for computational resources.

2. Currently, our Pauli grouper calls the solver without making use of previously computed solutions. Warm starting methods could improve upon this. For example, the RREF of M from Eq. (9) could perhaps be reused.

3. Methods for automatically tuning the hyperparameters of our `restricted algebraic solver` deserve further investigation.

4. Instead of selecting the polynomially-large subset of subgraphs at random, one could implement a more sophisticated approach such as simulated annealing.

5. The geometry of the solution space $\mathcal{L}$ could be further explored. If we had access to a lexicographical Gröbner basis for the ideal generated by $\boldsymbol{\lambda}^{\mathrm{T}} Q_1 \boldsymbol{\lambda} + 1$, …, $\boldsymbol{\lambda}^{\mathrm{T}} Q_n \boldsymbol{\lambda} + 1 \in \mathbb{F}_2[\lambda_1, \ldots, \lambda_d]$ (which defines the Zariski-closed set $\mathcal{L} \subset \mathbb{F}_2^d$), we could construct a solution $\boldsymbol{\lambda} \in \mathcal{L}$ via elimination theory [59].

DATA AVAILABILITY

All relevant data are available from the authors upon reasonable request.

CODE AVAILABILITY

An open-source C++ implementation of our Pauli grouping algorithm based on our `informed numerical solver` is available under https://github.com/Mc-Zen/HT-Grouper.

ACKNOWLEDGEMENTS

This research is part of two projects that have received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreements No. 847471 and No. 955479. This research was supported by the NCCR SPIN, funded by the Swiss National Science Foundation (SNF). I.O.S. acknowledges the financial support from the SNF through the grant No. 200021-179312. We acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

In the final phase of this project, DM received financial support from the Munich Quantum Valley (K-8), the Bundesministerium für Bildung und Forschung (BMBF) via the projects HYBRID, REALISTIQ, and MUNIQC-Atoms as well as from the EU Quantum Technology Flagship via the project MILLENION. Research was sponsored by IARPA and the Army Research Office, under the Entangled Logical Qubits program, and was accomplished under Cooperative Agreement Number W911NF-23-2-0212. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of IARPA, the Army Research Office, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

The authors are thankful to Lennart Vincent Bittel, Libor Caha, Felix Huber, Seyed Sajjad Nezhadi, Max Rossmannek, Maria Spethmann, David Sutter, Stefan Woerner, James Robin Wootton, and Nikolai Wyderka for stimulating discussions. SDG

COMPETING INTERESTS

The authors declare no competing financial or non-financial interests.

AUTHOR CONTRIBUTIONS

DM developed the theory, implemented and tested the grouping algorithm, and wrote the manuscript. DM and PB performed the experiment. DM and LF analyzed the experimental data. DM, LF, IS, PB, and IT were involved in the design of the experiment and discussed the results. KL implemented the open-source version of the grouping algorithm. KL, EK, JE, and IS supported DM in carrying out the study in SM Sec. XIV. All authors contributed to the manuscript.

[1] IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. The current list of IBM trademarks is available at https://www.ibm.com/legal/copytrade.

[2] A. Kandala *et al.*, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, Nature **549**, 242 (2017).

[3] F. Arute *et al.*, Quantum supremacy using a programmable superconducting processor, Nature **574**, 505 (2019).

[4] P. Krantz *et al.*, A quantum engineer's guide to superconducting qubits, Appl. Phys. Rev. **6**, 021318 (2019).

[5] A. Blais, A. L. Grimsmo, S. M. Girvin, and A. Wallraff, Circuit quantum electrodynamics, Rev. Mod. Phys. **93**, 025005 (2021).

[6] S. Bravyi, O. Dial, J. M. Gambetta, D. Gil, and Z. Nazario, The future of quantum computing with superconducting qubits, J. Appl. Phys. **132**, 160902 (2022).

[7] G. Burkard, T. D. Ladd, A. Pan, J. M. Nichol, and J. R. Petta, Semiconductor spin qubits, Rev. Mod. Phys. **95**, 025003 (2023).

[8] S. Krinner *et al.*, Realizing repeated quantum error correction in a distance-three surface code, Nature **605**, 669 (2022).

[9] Y. Kim *et al.*, Scalable error mitigation for noisy quantum circuits produces competitive expectation values, Nat. Phys. **19**, 752 (2023).

[10] Y. Kim *et al.*, Evidence for the utility of quantum computing before fault tolerance, Nature **618**, 500 (2023).

[11] D. Hangleiter and J. Eisert, Computational advantage of quantum random sampling, Rev. Mod. Phys. **95**, 035001 (2023).

[12] D. Bluvstein *et al.*, Logical quantum processor based on reconfigurable atom arrays, Nature **626**, 58 (2024).

[13] A. Peruzzo *et al.*, A variational eigenvalue solver on a photonic quantum processor, Nature Comm. **5**, 4213 (2014).

[14] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, New J. Phys. **18**, 023023 (2016).

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

[16] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Evaluating analytic gradients on quantum hardware, Phys. Rev. A **99**, 032331 (2019).

[17] R. Sweke *et al.*, Stochastic gradient descent for hybrid quantum-classical optimization, Quantum **4**, 314 (2020).

[18] V. Verteletskyi, T.-C. Yen, and A. F. Izmaylov, Measurement optimization in the variational quantum eigensolver using a minimum clique cover, J. Chem. Phys. **152**, 124114 (2020).

[19] P. Gokhale *et al.*, $O(N^3)$ measurement cost for variational quantum eigensolver on molecular Hamiltonians, IEEE Trans. Quantum Eng. **1**, 1 (2020).

[20] T.-C. Yen, V. Verteletskyi, and A. F. Izmaylov, Measuring all compatible operators in one series of single-qubit measurements using unitary transformations, J. Chem. Theory Comput. **16**, 2400 (2020).

[21] A. Jena, S. N. Genin, and M. Mosca, Optimization of variational-quantum-eigensolver measurement by partitioning Pauli operators using multiqubit Clifford gates on noisy intermediate-scale quantum hardware, Phys. Rev. A **106**, 042443 (2022).

[22] O. Crawford *et al.*, Efficient quantum measurement of Pauli operators in the presence of finite sampling error, Quantum **5**, 385 (2021).

[23] I. Hamamura and T. Imamichi, Efficient evaluation of quantum observables using entangled measurements, npj Quant. Inf. **6**, 56 (2019).

[24] F. Escudero, D. Fernández-Fernández, G. Jaumà, G. F. Peñas, and L. Pereira, Hardware-efficient entangled measurements for variational quantum algorithms, Phys. Rev. Appl. **20**, 034044 (2023).

[25] B. Wu, J. Sun, Q. Huang, and X. Yuan, Overlapped grouping measurement: A unified framework for measuring quantum states, Quantum **7**, 896 (2023).

[26] T.-C. Yen, A. Ganeshram, and A. F. Izmaylov, Deterministic improvements of quantum measurements with grouping of compatible operators, non-local transformations, and covariance estimates, npj Quant. Inf. **9**, 14 (2023).

[27] M. Ohliger, V. Nesme, and J. Eisert, Efficient and feasible state tomography of quantum many-body systems, New J. Phys. **15**, 015024 (2013).

[28] S. Aaronson, Shadow tomography of quantum states, in *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018 (Association for Computing Machinery, New York, NY, USA, 2018) p. 325–338.

[29] H.-Y. Huang, R. Kueng, and J. Preskill, Predicting many properties of a quantum system from very few measurements, Nature Phys. **16**, 1050 (2020).

[30] C. Hadfield, S. Bravyi, R. Raymond, and A. Mezzacapo, Measurements of quantum Hamiltonians with locally-biased classical shadows, Comm. Math. Phys. **391**, 951 (2022).

[31] H.-Y. Huang, R. Kueng, and J. Preskill, Efficient estimation of Pauli observables by derandomization, Phys. Rev. Lett. **127**, 030503 (2021).

[32] C. Hadfield, Adaptive Pauli shadows for energy estimation, Preprint at: https://arxiv.org/abs/2105.12207 (2021).

[33] A. F. Izmaylov, T.-C. Yen, R. A. Lang, and V. Verteletskyi, Unitary partitioning approach to the measurement problem in the variational quantum eigensolver method, J. Chem. Theory Comput. **16**, 190 (2020).

[34] A. Zhao *et al.*, Measurement reduction in variational quantum algorithms, Phys. Rev. A **101**, 062322 (2020).

[35] W. J. Huggins *et al.*, Efficient and noise resilient measurements for quantum chemistry on near-term quantum computers, npj Quant. Inf. **7**, 23 (2021).

[36] G. García-Pérez *et al.*, Learning to measure: Adaptive informationally complete generalized measurements for quantum algorithms, PRX Quantum **2**, 040342 (2021).

[37] L. E. Fischer *et al.*, Ancilla-free implementation of generalized measurements for qubits embedded in a qudit space, Phys. Rev. Research **4**, 033027 (2022).

[38] A. Shlosberg *et al.*, Adaptive estimation of quantum observables, Quantum **7**, 906 (2023).

[39] S. Hillmich, C. Hadfield, R. Raymond, A. Mezzacapo, and R. Wille, Decision diagrams for quantum measurements with shallow circuits, IEEE Trans. Quantum Eng. **2**, 24 (2021).

[40] J. Dehaene and B. De Moor, Clifford group, stabilizer states, and linear and quadratic operations over GF(2), Phys. Rev. A **68**, 042318 (2003).

[41] M. Hein, J. Eisert, and H. J. Briegel, Multiparty entanglement in graph states, Phys. Rev. A **69**, 062311 (2004).

[42] M. Van den Nest, J. Dehaene, and B. De Moor, Graphical description of the action of local Clifford transformations on graph states, Phys. Rev. A **69**, 022316 (2004).

[43] D. Maslov, Linear depth stabilizer and quantum fourier transformation circuits with no auxiliary qubits in finite-neighbor quantum architectures, Phys. Rev. A **76**, 052310 (2007).

[44] C. Chamberland, G. Zhu, T. J. Yoder, J. B. Hertzberg, and A. W. Cross, Topological and subsystem codes on low-degree graphs with flag qubits, Phys. Rev. X **10**, 011022 (2020).

[45] Gurobi Optimization, LLC, Gurobi optimizer reference manual (2023), `https://www.gurobi.com`.

[46] A. D. Pia, S. S. Dey, and M. Molinaro, Mixed-integer quadratic programming is in NP, Mathematical Programming **162**, 225 (2017).

[47] Qiskit: An open-source framework for quantum computing (2021).

[48] S. Bravyi, S. Sheldon, A. Kandala, D. C. Mckay, and J. M. Gambetta, Mitigating measurement errors in multiqubit experiments, Phys. Rev. A **103**, 042605 (2021).

[49] J. J. Meyer, J. Borregaard, and J. Eisert, A variational toolbox for quantum multi-parameter estimation, npj Quant. Inf. **89**, 89 (2021).

[50] T. Hubregtsen, F. Wilde, S. Qasim, and J. Eisert, Single-component gradient rules for variational quantum algorithms, Quant. Sc. Tech. **7**, 035008 (2022).

[51] E. van den Berg and K. Temme, Circuit optimization of Hamiltonian simulation by simultaneous diagonalization of Pauli clusters, Quantum **4**, 322 (2020).

[52] P. K. Faehrmann, M. Steudtner, R. Kueng, M. Kieferova, and J. Eisert, Randomizing multi-product formulas for Hamiltonian simulation, Quantum **6**, 806 (2022).

[53] A. Anand and K. R. Brown, Leveraging commuting groups for an efficient variational Hamiltonian ansatz, Preprint at: https://arxiv.org/abs/2312.08502 (2023).

[54] D. A. Lidar and T. A. Brun, *Quantum error correction* (Cambridge University Press, 2013).

[55] R. Muller, Pyquante: Python quantum chemistry (accessed 25 February 2022).

[56] Q. Sun *et al.*, PySCF: the Python-based simulations of chemistry framework, Wiley Interd. Rev.: Comput. Mol. Sci. **8**, e1340 (2018).

[57] F. H. L. Essler, H. Frahm, F. Göhmann, A. Klümper, and V. E. Korepin, *The one-dimensional Hubbard model* (Cambridge University Press, 2005).

[58] Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

[59] D. Cox, J. Little, and D. O'Shea, *Ideals, varieties, and algorithms: An introduction to computational algebraic geometry and commutative algebra* (Springer Science & Business Media, 2013).