



Reducing circuit depth with qubitwise diagonalization

Edison M. Murairi ^{1,*} and Michael J. Cervia ^{1,2,†}

¹*Department of Physics, The George Washington University, Washington, District of Columbia 20052, USA*

²*Department of Physics, University of Maryland, College Park, Maryland 20742, USA*



(Received 13 June 2023; accepted 14 November 2023; published 13 December 2023)

A variety of quantum algorithms employ Pauli operators as a convenient basis for studying the spectrum or evolution of Hamiltonians or measuring many-body observables. One strategy to reduce circuit depth in such algorithms involves simultaneous diagonalization of Pauli operators generating unitary evolution operators or observables of interest. We propose an algorithm yielding quantum circuits with depths $O(n \log r)$ diagonalizing n -qubit operators generated by r Pauli operators. Moreover, as our algorithm iteratively diagonalizes all operators on at least one qubit per step, it is well suited to maintain low circuit depth even on hardware with limited qubit connectivity. We observe that our algorithm performs favorably in producing quantum circuits diagonalizing randomly generated Hamiltonians as well as molecular Hamiltonians with short depths and low two-qubit gate counts.

DOI: [10.1103/PhysRevA.108.062414](https://doi.org/10.1103/PhysRevA.108.062414)

I. INTRODUCTION

Quantum simulations [1–7] are frequently cited as one of the most promising classes of quantum algorithms to see a practical quantum advantage [8], in an era where analog quantum simulators and universal quantum computers are becoming viable for many-body systems of size $O(50)$. In the case of qubit hardware, it is frequently convenient to decompose one’s Hamiltonian into Pauli operators, also known as “multiqubit Pauli matrices,” i.e., Pauli matrices on individual qubits strung together [6]. In fact, this convenience carries over to other algorithms implementing the action of a Hamiltonian on qubits, such as in a quantum Lanczos algorithm suggested by Ref. [9]. Moreover, for a yet wider collection of quantum algorithms, efficient measurement schemes must be devised to estimate the value of physical observables with respect to a given prepared state, even where direct simulation of time evolution is not required, such as in variational quantum eigensolvers [10] and quantum tomography [6,11–14]. Due to the qubitwise nature of measuring prepared quantum states, it is already natural to consider Pauli operators as a basis for measurements of observables in these many algorithms [6,10,11,13].

In practical implementations of such algorithms, one finds that the number of Pauli operators involved, i.e., in decomposing one’s Hamiltonian to be simulated or physical observables to be measured, may grow polynomially or even exponentially in system size. This growth can have troublesome consequences for the circuit depth involved in such computations [15]. In particular, one finds larger errors will be introduced into computations due to the decoherence of qubits associated

with two-qubit gates (with entangling power) in near-term hardware.

As such, one would like to augment these algorithms with strategies to minimize such costs. One such strategy involves simultaneous diagonalization of commuting Pauli operators [16], offering a reduction of the problem at hand to only the diagonal Pauli basis (i.e., strings of identity and Z Pauli matrices). Generally speaking, for measurement schemes on universal quantum computers, one must already perform a sort of diagonalization for each of the Pauli operators generating an observable of interest (since one typically measures in a Z basis as well), so the suggestion of simultaneous diagonalization [17] is a natural choice to reduce the number of measurements entailed. However, simultaneous diagonalization for measurements of Pauli operators may entail two-qubit gates that would otherwise be unnecessary in the state preparation or measurements of individual Pauli operators. Therefore, it is additionally important to minimize the number of such gates involved in simultaneous diagonalization to find the strategy advantageous. Conveniently, such a strategy can be used in tandem with other procedures to efficiently perform state tomography, such as randomized measurement protocols, e.g., in Ref. [18].

In what follows, we present a method to exactly and efficiently diagonalize a commuting set of Pauli operators. To provide context, we give a brief introduction and review to the mathematical language of stabilizer theory typically used to solve this problem in Sec. II. We present our algorithm in Sec. III and illustrate how it may be used to accommodate quantum hardware with limited connectivity. Furthermore, we provide estimates of worst-case circuit costs in Sec. IV. Additionally, we benchmark the resource costs of our algorithm in Sec. V by considering randomized sets of Pauli operators as well as molecular Hamiltonians, again taking limited qubit connectivity into account. Finally, we summarize our findings

*em712@gwu.edu

†cervia@umd.edu

and discuss future avenues for exploring circuit costs with our algorithm in Sec. VI.

II. REVIEW OF TABLEAU REPRESENTATION

Before we propose our algorithm, let us review language frequently used to pose the problem of efficient diagonalization of Pauli operators. Specifically, we briefly introduce here the tableau representation of Pauli operators, in particular, using the notation in Refs. [15,16]. We discuss the unitary operations on these Pauli operators in Sec. II A and pose the problem of diagonalization in Sec. II B.

Suppose we have N commuting Pauli operators acting on n qubits, e.g., generating an evolution operator, $U = \exp(i \sum_{i=1}^N c_i P_i)$, or physical observables that we would like to measure, $\{O = \sum_{i=1}^N c_i P_i\}$. The tableau corresponding to this collection of Pauli operators $\mathcal{P} = \{P_1, \dots, P_N\}$ is composed of three arrays: \mathcal{X} , \mathcal{Z} , and \mathcal{S} . The arrays \mathcal{X} and \mathcal{Z} each have dimensions $N \times n$, while the \mathcal{S} array is a column vector with N rows. For a given Pauli operator P_i with relative phase factor $c_i/|c_i| = \pm 1$, the entries of the three arrays are given by

$$\begin{aligned} \mathcal{X}[i, j] &= \begin{cases} 0 & \text{if the } j\text{th digit of } P_i \text{ is } Z \text{ or } I \\ 1 & \text{otherwise} \end{cases}, \\ \mathcal{Z}[i, j] &= \begin{cases} 0 & \text{if the } j\text{th digit of } P_i \text{ is } X \text{ or } I \\ 1 & \text{otherwise} \end{cases}, \\ \mathcal{S}[i] &= \begin{cases} 0 & \text{if } c_i > 0 \\ 1 & \text{otherwise} \end{cases}. \end{aligned} \quad (1)$$

For the remainder of the paper, we call the ‘‘tableau’’ of our Pauli operators the $N \times 2n$ block matrix $(\mathcal{X}|\mathcal{Z})$. Here, we neglect the relative phases tracked by the vector \mathcal{S} , as they are not necessary for developing a scheme to diagonalize the collection of commuting Pauli operators. Let us define the following notation for elements of these arrays: \mathcal{X}_i and \mathcal{Z}_i (with ‘‘lowered’’ indices) denote the i th column vectors of \mathcal{X} and \mathcal{Z} , respectively. In kind, ‘‘raised’’ indices \mathcal{X}^i and \mathcal{Z}^i will denote the i th row vectors. Moreover, a row of the tableau $u = (\mathcal{X}^j, \mathcal{Z}^j) \in \mathbb{F}_2^{2n}$ is said to ‘‘encode’’ a Pauli operator P if $P = \bigotimes_{i=1}^n X^{u_i} Z^{u_{n+i}}$ up to an overall phase.

A. Clifford gates and the symplectic group

Since a diagonal Pauli operator may contain only factors I and Z , the \mathcal{X} array of diagonal Pauli operators is zero. Consequently, diagonalization of Pauli operators may be viewed as conjugation with unitary operators to reduce the \mathcal{X} array to zero. Specifically, since we seek to transform Pauli operators into other Pauli operators, these unitary operators are Clifford gates, generated by the Hadamard (H), phase (S), and controlled-NOT (CNOT) gates [19].

Now, let C be an arbitrary Clifford gate. Conjugating the Pauli operators of \mathcal{P} with C can equivalently be viewed as an action transforming each column \mathcal{X}_i and \mathcal{Z}_i . We denote this action by functions f_i and g_i , written

$$\mathcal{X}_i \mapsto \mathcal{X}'_i = f_i(\mathcal{X}_1 \dots \mathcal{X}_N, \mathcal{Z}_1 \dots \mathcal{Z}_N), \quad (2)$$

$$\mathcal{Z}_i \mapsto \mathcal{Z}'_i = g_i(\mathcal{X}_1 \dots \mathcal{X}_N, \mathcal{Z}_1 \dots \mathcal{Z}_N), \quad (3)$$

which we will show are necessarily linear:

$$f_i(\mathcal{X}, \mathcal{Z}) = \sum_{j=1}^n a_i^j \mathcal{X}_j + b_i^j \mathcal{Z}_j, \quad (4)$$

$$g_i(\mathcal{X}, \mathcal{Z}) = \sum_{j=1}^n \tilde{a}_i^j \mathcal{X}_j + \tilde{b}_i^j \mathcal{Z}_j, \quad (5)$$

where $a_i^j, b_i^j, \tilde{a}_i^j, \tilde{b}_i^j \in \{0, 1\}$, or, more compactly,

$$f(\mathcal{X}, \mathcal{Z}) = \mathcal{X}A + \mathcal{Z}B, \quad (6)$$

$$g(\mathcal{X}, \mathcal{Z}) = \mathcal{X}\tilde{A} + \mathcal{Z}\tilde{B}, \quad (7)$$

for $A, B, \tilde{A}, \tilde{B} \in \mathbb{F}_2^{n \times n}$. We give the matrices corresponding to elementary Clifford gate conjugations here, derived from rules presented in, e.g., Refs. [16,20]. First let \mathbf{e}_i be the unit column vector with i th entry equal to 1. Then, we may summarize.

(1) For a Hadamard gate conjugation on qubit i , $H(i)$, we have corresponding matrices $B = \tilde{A} = \mathbf{e}_i \mathbf{e}_i^\top$ and $A = \tilde{B} = \mathbb{1}_{n \times n} - B$.

(2) For a phase gate conjugation on qubit i , $S(i)$, we have corresponding matrices $A = \tilde{B} = \mathbb{1}_{n \times n}$, $B = \mathbb{0}_{n \times n}$, and $\tilde{A} = \mathbf{e}_i \mathbf{e}_i^\top$.

(3) For a conjugation with CNOT(i, j) gates controlled by qubit i and targeting qubit j , we have corresponding matrices $A = \tilde{B}^\top = \mathbb{1}_{n \times n} + \mathbf{e}_i \mathbf{e}_j^\top$ and $B = \tilde{A} = \mathbb{0}_{n \times n}$.

Since these gates generate the Clifford group, and all compositions of linear functions are also linear, we conclude that all Clifford elements are linear, i.e., are given by some $(A, B, \tilde{A}, \tilde{B})$. For example, conjugation with S on qubits 1 and 2 corresponds to $(\mathbb{1}, \mathbb{0}, \mathbf{e}_1 \mathbf{e}_1^\top + \mathbf{e}_2 \mathbf{e}_2^\top, \mathbb{1})$, and conjugation with H on qubit i followed by CNOT(i, j) corresponds to $(\mathbb{1} - \mathbf{e}_i \mathbf{e}_i^\top, \mathbf{e}_i \mathbf{e}_i^\top + \mathbf{e}_i \mathbf{e}_j^\top, \mathbf{e}_i \mathbf{e}_i^\top + \mathbf{e}_i \mathbf{e}_j^\top, \mathbb{1} - \mathbf{e}_i \mathbf{e}_i^\top)$.

Furthermore, we note that the overall matrices

$$C = \begin{pmatrix} A & \tilde{A} \\ B & \tilde{B} \end{pmatrix} \quad (8)$$

for operations $H(i)$, $S(i)$, and CNOT(i, j) each satisfy the identity

$$C \Lambda C^\top = \Lambda = \begin{pmatrix} \mathbb{0}_{n \times n} & \mathbb{1}_{n \times n} \\ \mathbb{1}_{n \times n} & \mathbb{0}_{n \times n} \end{pmatrix}. \quad (9)$$

Therefore, the same identity must immediately hold for all compositions of these gates. Thus, we explicitly see the connection of Clifford gates to the group of $2n \times 2n$ symplectic matrices on \mathbb{F}_2 , $\text{Sp}(2n, \mathbb{F}_2)$, as noted in Refs. [21–23].

B. Statement of the problem

Let n be the total number of qubits. Suppose we are given a set of N commuting Pauli operators $\mathcal{P} = \{P_1, \dots, P_N\}$. We would like to produce a unitary circuit with which we will conjugate the Pauli operators in this set in order to simultaneously diagonalize them. In the interest of simplifying applications to quantum hardware, we would like to reduce the overall circuit depth and the total number of two-qubit gates involved as well as enhance flexibility of diagonalization strategies to accommodate limited qubit connectivity.

It can be shown that such a set may be generated by at most $r \leq \min(n, N)$ Pauli operators. Let us call $T = \{t_1, \dots, t_r\}$

a generating set of \mathcal{P} if each element of \mathcal{P} is a product of elements of T . That is, for each $P \in \mathcal{P}$, $P = \prod_{i=1}^r t_i^{\alpha_i}$ for some $\alpha_i \in \{0, 1\}$ up to an overall phase. In addition, if no element of T can be expressed as a product of other elements of T , we say that the elements of T are independent.

Now, let T be an independent generating set of \mathcal{P} . T can be obtained by first constructing the tableau of \mathcal{P} and row reducing the \mathcal{X} and \mathcal{Z} arrays. Importantly, a basis that diagonalizes T will also diagonalize \mathcal{P} . Meanwhile, since T is an independent set, it is more constrained than \mathcal{P} is. Therefore, we may reduce our problem to finding a hardware-efficient diagonalization of T .

III. QUBITWISE DIAGONALIZATION ALGORITHM

In this section, we propose an algorithm for simultaneously diagonalizing Pauli operators generating a desired set. First, we outline the philosophy of the algorithm in Sec. III A. More technically, we outline precisely the recursive steps of the proposed algorithm in Sec. III B. In kind, we explain how one can apply these steps in order to accommodate limited qubit connectivity in Sec. III C.

A. Approach

Our approach to diagonalize T is to iteratively diagonalize all Pauli operators one qubit at a time. In terms of the tableau for T , $(\mathcal{X}|\mathcal{Z})$, for some qubit i we solve the equation $\mathcal{X}'_i = \sum_{j=1}^n a'_j \mathcal{X}_j + b'_j \mathcal{Z}_j = 0$ for factors a'_j and b'_j , which we will show prescribes a sequence of Clifford gates with which we conjugate the elements of T , rendering all operators diagonal on i . In matrix form,

$$\mathcal{X}'_i = \left(\mathcal{X} \mid \mathcal{Z} \right) \begin{pmatrix} a'_i \\ b'_i \end{pmatrix} = 0, \quad (10)$$

where \mathbf{a}_i denotes the column vector $(a'_1, \dots, a'_n)^\top$ and similarly for \mathbf{b}_i . We may interpret this null vector as the constraint that certain columns in \mathcal{X} or \mathcal{Z} must sum to a zero vector, i.e., $\sum_{j=1}^n \mathcal{X}_j a'_j + \mathcal{Z}_j b'_j = 0$.

Equation (10) has a nonzero solution if the null space of $(\mathcal{X}|\mathcal{Z})$ is nontrivial. Since T is an independent set of commuting Pauli operators, this tableau forms a matrix of rank $r \leq n$. Also, $(\mathcal{X}|\mathcal{Z})$ has at most n rows but $2n$ columns. Therefore, this matrix must have a null space of dimension $2n - r \geq n$, and so such a vector $(\mathbf{a}_i, \mathbf{b}_i)$ exists. We can then obtain a solution $(\mathbf{a}_i, \mathbf{b}_i)$ for example by applying Gauss-Jordan elimination on the matrix $(\mathcal{X}|\mathcal{Z})$, as we will discuss more explicitly in Sec. IV A.

In fact, since the operators considered here are independent and commute, we have $\mathcal{X}^j \cdot \mathcal{Z}^k + \mathcal{Z}^j \cdot \mathcal{X}^k = 0$, implying that examples of such vectors in the null space include the r rows of $(\mathcal{Z}|\mathcal{X}) = (\mathcal{X}|\mathcal{Z})\Lambda$. More generally, we can see that a vector $u \in \mathbb{F}_2^{2n}$ is an element of the null space if and only if it encodes a Pauli operator that commutes with each Pauli operator encoded by $(\mathcal{Z}|\mathcal{X})$. [To see this fact, recall that a vector is in the null space when $(\mathcal{X}|\mathcal{Z})u = 0$, or equivalently $(\mathcal{Z}|\mathcal{X})\Lambda u = 0$. Writing out $u = (\mathbf{v}, \mathbf{w})$, where $\mathbf{v}, \mathbf{w} \in \mathbb{F}_2^n$, it follows that $\mathcal{Z}^k \cdot \mathbf{w} + \mathcal{X}^k \cdot \mathbf{v} = 0$, implying u commutes with the Pauli operator encoded by the row k of the tableau $(\mathcal{Z}|\mathcal{X})$, for each $k = 1, \dots, r$.] In terms of classical coding theory [6],

we may view the tableau as a parity check matrix for a linear code spanned by these null vectors as code words.

One can embed this null vector $(\mathbf{a}_i, \mathbf{b}_i)$ as the i th columns of matrices A and B encoding an overall unitary operation that renders all Pauli operators diagonal on qubit i . Importantly, there is still plenty of freedom in the choices of Clifford operations to specify \tilde{A} and \tilde{B} as well as the other columns of A and B . We exploit this freedom to prescribe the following instructions, which generate a particular choice of such matrices sending $\mathcal{X}_i \mapsto 0$. Additionally, these degrees of freedom as well as the choice of a suitable null vector can be further exploited to make choices that maintain low circuit depth given a particular quantum device where qubit connectivity is not all to all. For completeness, we will also show how a suitable null vector can be chosen efficiently.

In short: an element of the tableau's null space indicates columns of \mathcal{X} and \mathcal{Z} that are dependent and therefore may be added to diagonalize operators along some qubit i at each stage. As we detail below, these addition operations imply a concrete list of simple one- and two-qubit gates on a diagonalizing circuit.

B. The algorithm

Now, we are ready to prescribe the instructions to perform iteratively at each stage. Let $\alpha = 1, \dots, s \leq n$ be the stage where $n^{(\alpha)}$ digits are not yet diagonal for all Pauli operators. Moreover, $T^{(\alpha)}$ is the corresponding generating set of size $r^{(\alpha)} = |T^{(\alpha)}|$. Stage $\alpha = 1$ begins with the initial problem for our algorithm, i.e., $T^{(1)} = T$. At each stage α , we construct the corresponding tableau $(\mathcal{X}^{(\alpha)}|\mathcal{Z}^{(\alpha)})$. Then, we select a vector $(\mathbf{v}^{(\alpha)}, \mathbf{w}^{(\alpha)})$ in the null space of $(\mathcal{X}^{(\alpha)}|\mathcal{Z}^{(\alpha)})$. Choose a qubit i such that $v^{(\alpha)i} = 1$ or $w^{(\alpha)i} = 1$. Pauli operators will be diagonalized on this qubit in two steps by the procedure below.

(1) The first step consists of applying single-qubit Clifford gates to update each column $j = 1, \dots, n^{(\alpha)}$: $\mathcal{X}^{(\alpha)}_j \mapsto \mathcal{X}^{(\alpha)'}_j = v^{(\alpha)j} \mathcal{X}^{(\alpha)}_j + w^{(\alpha)j} \mathcal{Z}^{(\alpha)}_j$, according to the rules relating a and b to corresponding single-qubit Clifford gates outlined in Sec. II A.

(a) If $w^{(\alpha)j} = 0$, then we simply apply the identity gate, i.e., we need not do anything for this j .

(b) If $v^{(\alpha)j} = 0$ and $w^{(\alpha)j} = 1$, perform a conjugation with the gate $H(j)$.

(c) If $v^{(\alpha)j} = 1$ and $w^{(\alpha)j} = 1$, perform a conjugation with the gate $S(j)$ followed by a conjugation with the gate $H(j)$.

At the end of this step, we have

$$\mathcal{X}^{(\alpha)'}_j = v^{(\alpha)j} \mathcal{X}^{(\alpha)}_j + w^{(\alpha)j} \mathcal{Z}^{(\alpha)}_j \quad (11)$$

with $v^{(\alpha)j} = 1$ or $w^{(\alpha)j} = 1$. This step essentially selects all of the columns of $\mathcal{X}^{(\alpha)}$ and $\mathcal{Z}^{(\alpha)}$ that are involved in reducing the column $\mathcal{X}^{(\alpha)}_i$ to zero and stores them in $\mathcal{X}^{(\alpha)'}$.

(2) The final step is then to add all of these columns into $\mathcal{X}^{(\alpha)'}$. We carry out this step using conjugations with CNOT gates. In particular, for each $j = 1, \dots, n^{(\alpha)}$, if $v^{(\alpha)j} = 1$ or $w^{(\alpha)j} = 1$ and $j \neq i$, perform a conjugation with CNOT(j, i). At the end of this step, the columns $\mathcal{X}^{(\alpha)'}$ selected by having coefficients $v^{(\alpha)j} = 1$ or $w^{(\alpha)j} = 1$ are added to the column

$\mathcal{X}^{(\alpha)'}_i$. We therefore obtain

$$\mathcal{X}^{(\alpha)''}_i = \sum_{j=1}^n v^{(\alpha)j} \mathcal{X}_j^{(\alpha)} + w^{(\alpha)j} \mathcal{Z}_j^{(\alpha)} = 0,$$

which is guaranteed to equal zero as desired since $(\mathbf{v}^{(\alpha)}, \mathbf{w}^{(\alpha)})$ is in the null space of $(\mathcal{X}^{(\alpha)}|\mathcal{Z}^{(\alpha)})$; see Eq. (10) and the discussion below it.

Thus concludes stage α . After we have diagonalized the Pauli matrices acting on qubit i with unitary operations, the Pauli operators acting on the remaining qubits must also commute. Therefore, we can diagonalize the remaining qubits in recursive stages.

After stage α is complete, we have $n^{(\alpha+1)} \leq n^{(\alpha)} - 1$. Note $n^{(\alpha+1)}$ is not necessarily $n^{(\alpha)} - 1$, as other qubits beside i may have been diagonalized in the process. Proceeding to the next stage, to obtain $T^{(\alpha+1)}$, we exclude all the qubits that are acted upon only by I or Z . Consequently, note that $r^{(\alpha+1)} \leq r^{(\alpha)}$, as the number of distinct generators may decrease after reducing the number of qubits still in consideration. Then, we repeat the above process for $T^{(\alpha+1)}$. At a stage $\alpha = s$ where $n^{(s)} = 1$, $r^{(s)} = 1$ and step 1 of the procedure above is sufficient to diagonalize the qubit. Thus, the recursion indeed terminates. The complete algorithm is summarized below in pseudocode.

Note that the example of a simple construction for step 2 above provides a circuit with depth linear with the number

Algorithm 1. Qubitwise diagonalization.

Input: A set of commuting Pauli operators acting on n qubits.

Output: A circuit simultaneously diagonalizing the set.

```

1:  $\alpha \leftarrow 1$ 
2: Discard all the qubits on which all Paulis are already diagonal.
3:  $n^{(\alpha)} \leftarrow$  Number of remaining qubits
4: if  $n^{(\alpha)} = 0$  then
5:   Exit ▷ The diagonalization is complete.
6: end if
7: Find independent Pauli operators  $T^{(\alpha)}$  ▷ See Sec. II B.
8:  $\mathcal{M}^{(\alpha)} \leftarrow (\mathcal{X}^{(\alpha)}|\mathcal{Z}^{(\alpha)})$  ▷ Tableau encoding of  $T^{(\alpha)}$ 
9: Find the null space of  $\mathcal{M}^{(\alpha)}$ 
10: Select a vector  $(\mathbf{v}, \mathbf{w})$  in the null space of  $\mathcal{M}^{(\alpha)}$ 
11: Select  $i$  such that  $v^i = 1$  or  $w^i = 1$  ▷ qubit  $i$  on which all
    Paulis will be diagonalized
12: for  $j = 1$  to  $j = n^{(\alpha)}$  do
13:   if  $v^j = 0$  AND  $w^j = 1$  then
14:     Perform a conjugation with  $H(j)$ 
15:   end if
16:   if  $v^j = 1$  AND  $w^j = 1$  then
17:     Perform a conjugation with  $S(j)$ 
18:     Perform a conjugation with  $H(j)$ 
19:   end if
20: end for
21: for  $j = 1$  to  $j = n^{(\alpha)}$  do
22:   if  $(v^j = 1$  or  $w^j = 1)$  AND  $i \neq j$  then
23:     Perform a conjugation with  $\text{CNOT}(j, i)$ 
24:   end if
25: end for
26:  $\alpha \leftarrow \alpha + 1$ 
27: Go to Line 2.

```

of CNOT gates at each stage, assuming hardware on which commuting CNOT gates that share qubits *cannot* be performed in parallel. In Sec. IV B, we discuss another construction with circuit depth that grows only logarithmically with the number of CNOT gates at each stage. Via this latter construction we achieve the $O(n \log r)$ overall circuit depth advertised.

C. Accommodating qubit connectivity

We now turn our attention to discussing how this algorithm may be tailored to hardware with limited connectivity, in the sense that we would like it to involve relatively few SWAP gates. Suppose that the graph $G = (V, E)$ represents the hardware connectivity. That is, a qubit q_i is represented by a vertex in V , and there is an edge $(q_i, q_j) \in E$ if qubits q_i and q_j are directly connected on the hardware (i.e., permitting a CNOT operation with either qubit as target or control and without SWAP operations).¹ Qualitatively, the strategy is to exploit the degrees of freedom still allowed by the algorithm described in the previous section, particularly in the choices of the null vector and the order of additions in tableau columns prescribed by this null vector. We elaborate on how these choices can be related and how they can be made with limited qubit connectivity in mind.

The first task is to choose a null space vector (\mathbf{v}, \mathbf{w}) that is compatible with the qubit connectivity of a given quantum device. However, we stress that the choice of this vector also entails a list of qubits, for pairs of which $(q_i$ and $q_j)$ we add corresponding column vectors of the tableau, via CNOT gates (and potentially SWAP gates as well). We would like to choose qubit pairs whose paths involve relatively few SWAP gates to be realized. That is, for all i such that $v^i = 1$ or $w^i = 1$ and j such that $v^j = 1$ or $w^j = 1$, we desire a relatively short path between vertices q_i and q_j in the connectivity graph G . As such, we can choose a null vector for which the qubits q_i and q_j are relatively close to one another in G . Furthermore, we give below an explanation of how, given an arbitrary null space vector, we can choose pairs of qubits on which we will perform CNOT gates, in order to further accommodate limited connectivity.

Now given an arbitrarily chosen null vector, we can describe the problem of choosing how one adds columns as per step 2 for this null vector. (We may assume that step 1 of the algorithm is already completed, as it involves only one-qubit gates.) For step 2, let us denote the qubits where (\mathbf{v}, \mathbf{w}) is nonzero as $Q = \{q_1, q_2, q_3, \dots\}$ and the columns \mathcal{X}_q with $q \in Q$ to be added using CNOT gate conjugations. In general, since the pairs $(q_i, q_j) \subseteq V$ may be disconnected on the hardware in consideration, we would need to perform appropriate conjugations with SWAP gates resulting in $q_i \mapsto q'_i$ and $q_j \mapsto q'_j$ such that q'_i and q'_j are connected on the hardware (i.e., share an edge in E). This transformation can be accomplished along any path in G connecting q_i and q_j , and the

¹One may also consider hardware with yet further limited connectivity such that a gate $\text{CNOT}(i, j)$ may be performed but not $\text{CNOT}(j, i)$. The remainder of this discussion below may be generalized to this case, however potentially requiring yet additional resources to accommodate such limitations.

number of SWAP gates depends on the length of the path. As such, we seek to minimize the length of a path that traverses all of the qubits $q \in Q$. Our minimization problem is similar to the well-known traveling salesperson problem (TSP) [24]. However, in contrast with the TSP, which requires every node be visited exactly once, our problem requires only the nodes in Q be visited once. That is, a node not in Q may or may not be visited at all. This minimization problem naturally maps to the so-called shortest-path problem with specified nodes (SPPSN)² [25], where the specified nodes are the nodes in Q . This problem has been extensively studied, and several solutions are provided in, e.g., Refs. [25–28]. Indeed, solving the SPPSN gives us a sequence of qubits $\{q_{\sigma 1}, q_{\sigma 2}, q_{\sigma 3}, \dots\}$ along which the number of SWAP gates is minimized when performing step 2. Finally, we note that the choice of null vector essentially dictates which SPPSN we must solve.

As explained above, modifications in step 2 of our algorithm can allow for multiple avenues to reduce the SWAP gates required. Not only can we draw upon efficient solutions to the related generalization of a TSP [i.e., choose how to add columns of \mathcal{X}'], but also we are afforded a choice of which such problem we would like to solve [i.e., choose which columns of \mathcal{X}' to be added according to (\mathbf{v}, \mathbf{w})]. In Sec. VC, we demonstrate how this strategy can be adapted to hardware with linear connectivity.

IV. ANALYSIS OF THE ALGORITHM

Here, we estimate the size of quantum circuits produced by the algorithm presented in the previous section. In particular, we focus on the overall circuit depth as well as the number of CNOT gates entailed. First, we explain in Sec. IVA how these qualities of the circuit are related to the null vectors introduced in Sec. IIIA. With this understanding in hand, we then can estimate the circuit depth (Sec. IVB) and total two-qubit gate count (Sec. IV C) required to implement the diagonalization circuits prescribed by our algorithm.

A. Preliminaries

Before we proceed, it is helpful to recall the perspective that our algorithm essentially searches for \mathcal{X} and \mathcal{Z} columns that are dependent and adds them to diagonalize along some qubit i at each stage. This interpretation is useful in estimating the cost.

Definition 1. The symplectic weight of a vector $(\mathbf{v}, \mathbf{w}) \in \mathbb{F}_2^{2n}$ is defined as

$$\omega(\mathbf{v}, \mathbf{w}) = |\{i \mid (v^i, w^i) \neq (0, 0)\}|. \quad (12)$$

In other words, the symplectic weight counts the number of digits not equal to I in the corresponding Pauli operator (i.e., the operator’s Hamming weight). We will also find $\omega(\mathbf{v}, \mathbf{w})$ of null space vectors helpful in estimating the number of CNOT gate conjugations.

Lemma 1. Given a vector (\mathbf{v}, \mathbf{w}) in the null space of $(\mathcal{X}^{(\alpha)} | \mathcal{Z}^{(\alpha)})$, our algorithm prescribes $\omega(\mathbf{v}, \mathbf{w}) - 1$ CNOT gate conjugations to diagonalize one qubit.

Suppose qubit i is to be diagonalized using the vector (\mathbf{v}, \mathbf{w}) through steps 1 and 2. By Definition 1, step 1 will update $\omega(\mathbf{v}, \mathbf{w})$ columns in \mathcal{X} . Then, step 2 will use $\omega(\mathbf{v}, \mathbf{w}) - 1$ CNOT conjugations, because there are $\omega(\mathbf{v}, \mathbf{w}) - 1$ columns in \mathcal{X} to be added to column \mathcal{X}_i .

Lemma 2. The number of CNOT gate conjugations required to diagonalize a given qubit at stage α is at most $r^{(\alpha)}$ if $n^{(\alpha)} > r^{(\alpha)}$, and at most $r^{(\alpha)} - 1$ if $n^{(\alpha)} = r^{(\alpha)}$.

We first consider the case $n^{(\alpha)} > r^{(\alpha)}$. At stage α , the rank of $(\mathcal{X}^{(\alpha)} | \mathcal{Z}^{(\alpha)})$ is $r^{(\alpha)}$, implying that any collection of $r^{(\alpha)} + 1$ columns is a dependent set. Therefore, there exists a vector (\mathbf{v}, \mathbf{w}) in the null space of $(\mathcal{X}^{(\alpha)} | \mathcal{Z}^{(\alpha)})$ with weight $\omega(\mathbf{v}, \mathbf{w}) \leq r^{(\alpha)} + 1$. Consequently, using Lemma 1, we demand at most $r^{(\alpha)}$ CNOT gate conjugations at this stage.

For the case $n^{(\alpha)} = r^{(\alpha)}$, we know $\omega(\mathbf{v}, \mathbf{w}) \leq r^{(\alpha)}$ for all (\mathbf{v}, \mathbf{w}) in the null space, because there are only $r^{(\alpha)}$ qubits. Therefore, this case entails at most $r^{(\alpha)} - 1$ CNOT gate conjugations.

Finally, we stress that, in either case, such a null space vector can be produced efficiently. For example, we can find such a null space vector by finding a set of $\leq r^{(\alpha)} + 1$ columns adding to zero. To demonstrate this bound, we first transform the matrix $(\mathcal{X}^{(\alpha)} | \mathcal{Z}^{(\alpha)})$ to its reduced row-echelon form in polynomial complexity [29]. Then, up to a relabeling of qubits, our tableau can be written in the so-called standard form of a stabilizer code (see, e.g., Ref. [6]):

$$(\mathbb{1}_{r^{(\alpha)} \times r^{(\alpha)}} \quad \mathcal{A}_{r^{(\alpha)} \times (n^{(\alpha)} - r^{(\alpha)})} \quad | \quad \mathcal{B}_{r^{(\alpha)} \times n^{(\alpha)}}) \quad (13)$$

where \mathcal{A} and \mathcal{B} are matrices over \mathbb{F}_2 with the indicated dimensions. In this form, any column of \mathcal{A} or \mathcal{B} is clearly a linear combination of the columns of $\mathbb{1}$; the m th column c_m ($r^{(\alpha)} < m \leq 2n$) of the transformed tableau above can be written as $c_m = \sum_{i=1}^{r^{(\alpha)}} c_m^i \mathbf{e}_i$. In other words, any column of \mathcal{A} or \mathcal{B} may be added with a subset of the columns of $\mathbb{1}$ to yield a zero vector, and so a vector encoding this sum would be a suitable null vector to start our algorithm of Sec. IIIB. Explicitly, the collection of these columns to be summed corresponds to a null vector $u \in \mathbb{F}_2^{2n^{(\alpha)}}$ with $u^i = 1$ and $u^m = 1$ only at i and m for which $c_m^i = 1$. In fact, one may choose the column c of \mathcal{A} or \mathcal{B} with the lowest (Hamming) weight to reduce the weight of the corresponding null vector $\omega(u)$ as well as the gate cost that it entails (as per Lemma 1).

B. Circuit depth

Now, we will analyze the overall circuit depth of the algorithm. Notably, on hardware where commuting CNOT gates can be performed in parallel, as discussed in, e.g., Ref. [30], we in fact already have constant depth at each stage via the instructions provided in Sec. IIIB, yielding overall depth $O(n)$. Remarkably, accommodating more generic hardware, we find that the circuit has depth $O(n \log r)$ in contrast with the number of gates, which we will show is $O(nr)$.

At each stage α , we note that step 1 produces a circuit of depth at most two, because only single-qubit gates may be performed on each qubit. Meanwhile, as we shall argue below, the CNOT gates conjugations in step 2 can be performed

²In other words, the problem is to find the shortest path passing through a subset of vertices in a graph.

Algorithm 2. Step 2 yielding logarithmic depth.

Input: A null space vector (\mathbf{v}, \mathbf{w}) .
Output: A modified step (2) of the algorithm.

- 1: $Q \leftarrow \{q = 1, \dots, n^{(\alpha)} \mid v^q = 1 \text{ or } w^q = 1\}$
- 2: **while** $|Q| > 1$ **do**
- 3: **for** $j = 1$ to $j = \lfloor |Q|/2 \rfloor$ **do**
- 4: Perform a conjugation with CNOT(q_{2j}, q_{2j-1})
- 5: **end for**
- 6: $Q \leftarrow Q \setminus \{q_{2j} \mid j = 1, \dots, \lfloor |Q|/2 \rfloor\}$
- 7: **end while**

with depth $O(\log r^{(\alpha)})$. Moreover, since there are at most n recursive stages in the algorithm, the overall circuit has depth $O(n \log r)$.

Recall that step 1 modifies the columns involved in the diagonalization according to Eq. (11), and step 2 adds these \mathcal{X}' columns. Without loss of generality, let us denote these columns by $\mathcal{X}'_1, \mathcal{X}'_2, \dots, \mathcal{X}'_\omega$ where ω is the weight of the null space vector used for diagonalization. In step 2, we can add these columns in pairs sequentially (as described earlier in Sec. III B), or alternatively we may form disjoint pairs to be added in parallel to the same end. That is, we can perform CNOT($2j, 2j-1$) resulting in $\mathcal{X}'_{2j-1} \leftarrow \mathcal{X}'_{2j-1} + \mathcal{X}'_{2j}$ for $j \in \{1, 2, \dots, \lfloor \omega/2 \rfloor\}$. Note that all of these CNOT gates amount to depth 1, since they each act on totally different qubits. Now, we add these resulting columns $\mathcal{X}'_1, \mathcal{X}'_3, \dots$ in the same fashion. This process repeats until we are left with only one \mathcal{X} column, and each round of this pairwise addition process is a circuit with depth 1. We summarize this procedure in pseudocode in Algorithm 2, replacing lines 21–25 of the instructions given in Algorithm 1.

Since each round of pairwise additions reduces the number of \mathcal{X} columns to be added by about one half, there are $O(\log_2 \omega)$ such rounds, each with depth 1. Using Lemma 2, we can choose $\omega \leq r^{(\alpha)} + 1$, and so each stage α involves a circuit with depth $O(\log r^{(\alpha)})$.

C. CNOT gate count

Having analyzed the circuit depth, we now turn our attention to discussing the total number of CNOT gates.

Theorem 1. The number of CNOT gate conjugations required by the algorithm (Sec. III B) is at most $n r - \frac{r(r+1)}{2}$.

To establish this bound, we will use Lemma 2 to count the number of CNOT gate conjugations at each stage. The first stage, $\alpha = 1$, requires at most $r^{(1)} = r$ CNOT gate conjugations. For the subsequent stage, $\alpha = 2$, if $n^{(2)} \geq r$, the number of independent generators at $r^{(2)}$ may remain r . Consequently, for all stages α such that $n^{(\alpha)} > r$, the inequality $r^{(\alpha)} \leq r$ holds. There are at most $(n-r)$ such stages, amounting to at most $(n-r)r$ CNOT gate conjugations. Once we reach a stage s such that $n^{(s)} = r$, the stage $\alpha = s+j$ has $r^{(s+j)} \leq n^{(s+j)} \leq r-j$, because each stage diagonalizes on at least one qubit. The number of CNOT gate conjugations in these stages is therefore at most $\sum_{j=0}^{r-1} (r-j-1) = r(r-1)/2$. Putting both CNOT gate conjugation counts together, we obtain $n r - r(r+1)/2$.

This bound is also derived for a different algorithm in Ref. [17]. Notably, in the limiting case $r = n$, this bound reduces to the known result $n(n-1)/2$ (see, e.g., Refs. [16,31]).

It is important to note that our algorithm guarantees this bound without further classical optimization. That is, one may select any $r^{(\alpha)} + 1$ columns to form a dependent set at each stage α , and this upper bound will hold. Now, we demonstrate that we can achieve a lower estimate of CNOT cost by minimizing the size of the dependent set of columns we select in each stage, entailing a higher classical complexity cost.

Lemma 3. A qubit can be diagonalized at stage α with at most $\lfloor \frac{r^{(\alpha)}}{2} \rfloor$ CNOT gate conjugations.

In exchange for a lower gate cost, one may accept a large classical complexity by searching the entire null space for the vector with the lowest symplectic weight by Lemma 1. Below, we will use the properties of the null space of $(\mathcal{X}|\mathcal{Z})$ at each stage to establish a stricter lower bound on the number of CNOT gates.

The rows of $(\mathcal{Z}^{(\alpha)}|\mathcal{X}^{(\alpha)})$ span a $r^{(\alpha)}$ -dimensional subspace of the null space, due to the fact that the Pauli operators represented by each row of $(\mathcal{X}^{(\alpha)}|\mathcal{Z}^{(\alpha)})$ commute with one another. Moreover, the Pauli operators in $(\mathcal{Z}^{(\alpha)}|\mathcal{X}^{(\alpha)})$ may be viewed as a $[[n^{(\alpha)}, k, d]]$ stabilizer code. The code maps $k = n^{(\alpha)} - r^{(\alpha)}$ logical qubits to $n^{(\alpha)}$ physical qubits with the capability of correcting at most $(d-1)/2$ errors. Moreover, we call S the stabilizer generated by the Pauli operators represented by the rows of $(\mathcal{Z}^{(\alpha)}|\mathcal{X}^{(\alpha)})$. For a more general review, see, e.g., Refs. [6,32].

Let $C(S)$ be the centralizer of S , the Pauli operators that commute with each element of S . For stabilizer codes, it is known that the normalizer is identical to the centralizer $N(S) = C(S)$ [32]. In Sec. III A, we argued that a vector u commutes with every element of $(\mathcal{Z}^{(\alpha)}|\mathcal{X}^{(\alpha)})$, i.e., is an element of $N(S)$, if and only if it is in the null space of $(\mathcal{X}^{(\alpha)}|\mathcal{Z}^{(\alpha)})$. Therefore, the task of finding a null space vector u of weight ω is equivalent to searching $N(S)$ for a Pauli operator of the desired weight. On the other hand, the quantum Singleton bound [21,33,34] directly implies that there exists a Pauli operator in $N(S)$ with weight $\omega = d \leq \lfloor r^{(\alpha)}/2 \rfloor + 1$. Finally, invoking Lemma 1 concludes the proof.

Theorem 2. The number of CNOT gate conjugations required by the algorithm (Sec. III B) is at most $n \lfloor \frac{r}{2} \rfloor - \lfloor \frac{r}{2} \rfloor^2$.

To achieve this bound, at each stage α , we select the vector with weight $d \leq \lfloor r^{(\alpha)}/2 \rfloor + 1$ promised by Lemma 3. Then, we follow an argument similar to that of Theorem 1 to find the total cost. That is, while $n^{(\alpha)} \geq r$, we use at most $\lfloor r/2 \rfloor$ CNOT gate conjugations. Since there are at most $n-r+1$ of such stages, they amount to at most $(n-r+1)\lfloor r/2 \rfloor$ CNOT conjugations. After the stage s where $n^{(s)} = r$, we use at most $\lfloor (r-i)/2 \rfloor$ for each subsequent stage $s+i$. These stages amount to at most $\sum_{i=1}^{r-1} \lfloor (r-i)/2 \rfloor = \lfloor r/2 \rfloor \lfloor (r-1)/2 \rfloor$ CNOT conjugations. Putting both contributions together, we obtain at most $(n-r+1)\lfloor r/2 \rfloor + \lfloor r/2 \rfloor \lfloor (r-1)/2 \rfloor$ CNOT conjugations, which gives us the bound claimed.

In the above argument, we have shown an equivalence between finding the distance of a quantum stabilizer code and reducing the CNOT count at one stage of our qubitwise diagonalization to $\sim r^{(\alpha)}$. Finding the distance of a code is known to be NP-hard [35], and so we may expect to practically find the operator(s) of the normalizer $N(S)$ with the

lowest weight only for a relatively small number of qubits ($n \lesssim 30$ on ordinary computers).

Equivalently, this problem may be restated as minimizing

$$\min_{\{b_1 + \dots + b_{2n-r} \leq z\}} \omega \left(\sum_{i=1}^{2n-r} b_i N^i \right), \quad (14)$$

where N^i are vectors encoding the independent generators of $N(S)$, each $b_i = 0$ or 1 , and the sum between $b_i N^i$ vectors is Boolean (i.e., XOR) and the sum over b_i is decimal. Here, we must take $z = 2n - r$ to search the entirety of $N(S)$ for the lowest-weight solution. Nevertheless, a search through linear combinations of fixed sizes $\leq z$ is still polynomial in n and still produces a vector of weight at least as low as that obtained from choosing among N^i .

On the other hand, it is crucial to note that there is a “greediness” to optimizing the (symplectic) weight of our null vector at each stage of our algorithm, as opposed to optimizing the choice for a circuit depth overall. In the sense, completely optimizing at this stage might not always produce the overall lowest circuit depth after diagonalizing over *all* qubits of our operators. In the following section, we will discuss further comparisons between circuit depths obtained with null vectors chosen via Gaussian elimination sans optimization (see Lemma 2 and its discussion below) and those chosen via complete optimization at each stage.

V. EXAMPLES

In this section, we evaluate the performance of our algorithm for a variety of example applications. In particular, we use a proposed method of randomized benchmarking [16] to assess the circuit cost resulting from our algorithm under various circumstances in Sec. V A. Secondly, in Sec. V B, we assess the algorithm’s performance in diagonalizing molecular Hamiltonians. Finally, we provide an example molecular Hamiltonian for which our algorithm accommodates limited qubit connectivity relatively well in Sec. V C.

A. Random sets of commuting Pauli operators

In this subsection, we apply our diagonalization algorithm to sets of commuting Pauli operators generated randomly. We use the algorithm constructed in Ref. [16] to generate a sample of 100 distinct sets, each containing n Pauli operators acting on n qubits. In our implementation, we use both the classically efficient algorithm outlined below Lemma 2 (“no optimization”) to select a null space vector of weight $\leq r$ and an exhaustive search through the null space to select the vector of lowest weight (“complete optimization”). For each value of n , we report the average number of CNOT conjugations needed over the 100 sets. The error bars in our results are given by the square root of the sample variance. We also superimpose our results to the lowest results obtained in Ref. [16] in Fig. 1.

In particular, we can see that there is a marked reduction in the quantum circuit cost when searching for a low-weight null vector to prescribe each stage of our algorithm, at least for totally randomized sets of Pauli operators. In other words, here we see a tradeoff, between classical complexity of

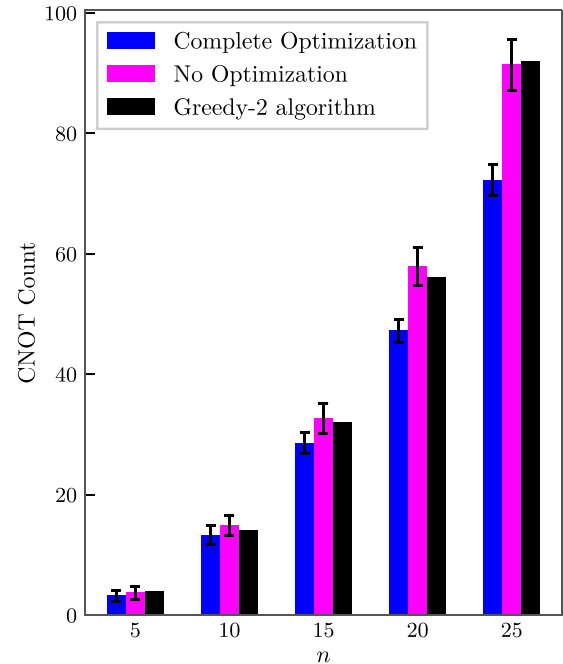


FIG. 1. The number of CNOT conjugations involved in diagonalizing n commuting Pauli operators acting on n qubits via our qubitwise diagonalization strategy, averaged over 100 sets generated randomly. Each uncertainty bar is the square root of the sample variance of the 100 sets. We compare gate counts found with and without minimizing the circuit depth, iteratively at each stage of our algorithm (see discussion following Theorem 2). In addition, the magenta (right) bars are in each case the smallest gate counts obtained in Ref. [16], via the “greedy-2” algorithm.

producing a low-weight null vector and resulting the quantum circuit complexity. Particularly, for $r \ll n$, this classical complexity may become highly prohibitive since the null space is of dimension $2n - r$, in which case the no optimization case conveniently produces a vector of (low) weight at most r anyhow. For the rest of this section, to assess the costs on a quantum computer of our algorithm while keeping classical complexity low, we limit our algorithm to no optimization (i.e., selecting null vectors via the procedure outlined under Lemma 2).

Now, we illustrate how our number of CNOT gates also depends on r , the number of independent generators of a set of commuting Paulis. We first fix n , the number of qubits, and again consider randomly generated sets of r independent commuting Pauli operators. To obtain a set with only $r \leq n$ operators, we randomly select $n - r$ operators to delete from the set. Then, we construct diagonalization quantum circuits and report the resulting number of CNOT gates. Figure 2 shows the results for $n = 10$. As expected per our discussion in Sec. IV C, we find that, for fixed n , gate count follows a quadratic trend in r as $f(r) = -ar^2 + br$ (up to a constant shift) for $a, b > 0$.

B. Molecular Hamiltonians

Having assessed the algorithm’s performance in diagonalizing commuting Hamiltonians generated randomly, we will

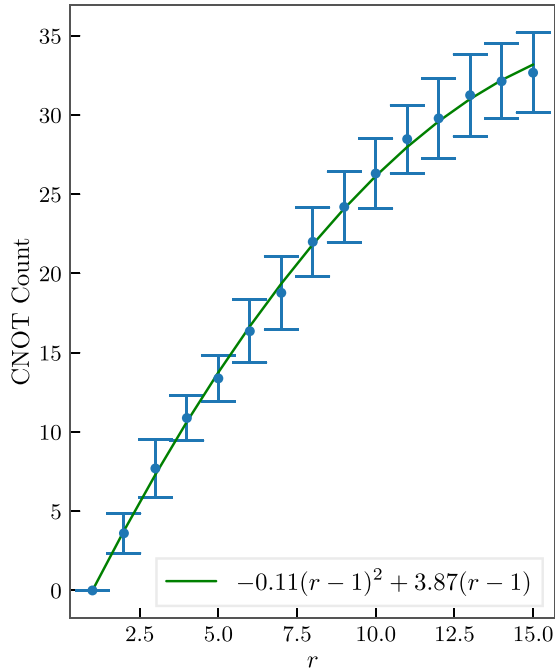


FIG. 2. The number of CNOT conjugations obtained in diagonalizing a set of r independent commuting Pauli operators acting on $n = 15$ qubits using the procedure outlined below Lemma 2 (no optimization). The data points are averages over 100 randomly generated sets. The uncertainty bars displayed are the square roots of the variances.

now consider its performance for a collection of molecular Hamiltonians as more practical examples. In particular, we consider one ionic compound HeH^+ as well as the molecules LiH , BeH_2 , NH_3 , and BH_3 .

The Hamiltonians as linear combinations of Pauli operators are obtained from PennyLane [36] Quantum Chemistry datasets. For each of these species, we use the STO-3G basis, and the optimal bond length given in the dataset. Given a Hamiltonian, before diagonalization we must divide the Pauli operators into sets of mutually commuting operators. As in Refs. [15,16,37], this problem can be reduced to a graph-coloring problem, and we use the independent set strategy in NETWORKX [38] to color the graph.

Table I shows the results of diagonalization of each of these Hamiltonians. In each case, we observe that both the average

number of CNOT gate conjugations and the circuit depth tend to be near once or twice the number of independent operators generating each set ($r \leq n$). However, it is worth noting that different choices of strategies to form commuting sets of Pauli operators (while using the same diagonalization algorithm) may allow for higher or lower average gate counts or depths per set.

Empirically we find that, for these more physically typical Hamiltonian models rather than randomized sets, in fact a lower circuit complexity is obtained from no optimization rather than complete optimization. From these cases, we may reasonably suspect that, in general, relatively local and symmetrical Hamiltonians modeling physical systems are more amenable to this simple classical procedure for producing circuit instructions than arbitrarily nonlocal and asymmetrical Hamiltonians of randomized benchmarking.

C. Linear qubit connectivity

Further, we demonstrate the strategy presented in Sec. III C assuming a hardware with linear connectivity. As a proof of principle, we use the H_4 chain Hamiltonian derived in Ref. [31]. This Hamiltonian was obtained by setting the interatomic distance $\Delta = 1.0 \text{ \AA}$ and using the STO-3G basis. The Hamiltonian is then mapped to eight-qubit Pauli operators using the Bravyi-Kitaev transformation [39]. To compare, we use the same division of the Hamiltonian into ten collections of commuting Pauli operators.

First, we consider the set of 20 Pauli operators shown in Fig. [15] of Ref. [31]. There, the authors derived a circuit with 13 H gates, 18 CNOT gates, 21 CZ gates, and 66 SWAP gates assuming a hardware with linear connectivity. Via our algorithm of Sec. III, we can produce a simpler circuit, with only seven H gates, five CNOT gates, and 11 SWAP gates. Figures 3(a) and 3(b) show the diagonalization circuit assuming fully connected hardware and linearly connected hardware, respectively.

The nine remaining sets of commuting Pauli operators are much simpler than the one discussed above. In the previous literature, the authors were able to derive simple circuits, notably with no SWAP gates and at most two two-qubit gates. Our algorithm produces similar results diagonalizing the Pauli operators in each set with no SWAP gates and at most two two-qubit gates per set.

TABLE I. Results of qubitwise diagonalization of molecular Hamiltonians. Throughout the diagonalization, the null space vector at each stage is chosen using the (classically efficient) algorithm described below Lemma 2. In each row, n and N are the number of qubits and the total number of Pauli operators in each Hamiltonian, respectively. κ is the number of sets of commuting Pauli operators, while r is the average number of independent generators per commuting set. The columns CNOT and Depth list the average numbers of CNOT gate conjugations and the average quantum circuit depths obtained, respectively, with averages taken over the different sets of commuting Pauli operators in the Hamiltonian. Finally CNOT SD and Depth SD are the standard deviations of the corresponding columns.

Molecule	n	N	κ	r	CNOT	CNOT SD	Depth	Depth SD
HeH^+	4	26	3	4.00	2.00	1.63	3.33	3.40
LiH	12	630	25	9.00	7.44	3.42	11.12	4.34
BeH_2	14	665	19	11.16	9.00	3.28	11.68	4.22
NH_3	16	2296	80	13.10	21.44	8.34	22.49	7.44
BH_3	16	2496	97	12.01	18.21	6.36	21.26	6.64

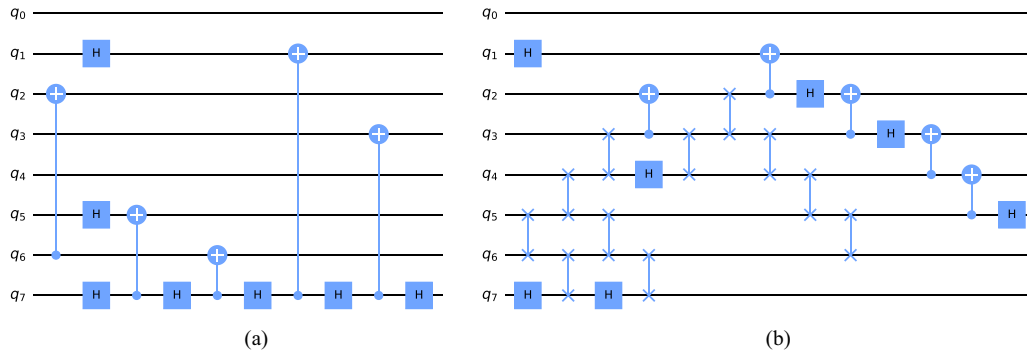


FIG. 3. Diagonalization circuits of a large commuting sub-Hamiltonian of a H_4 chain, derived in Fig. 15 of Ref. [31]. We assume a device (a) with full qubit connectivity and (b) with linear qubit connectivity. In the latter case, we apply the methods of Sec. III C, involving extra SWAP gates in order to implement more local two-qubit operations. We use the no optimization strategy to find a null space vector throughout the diagonalization.

VI. CONCLUSIONS

This paper introduces an algorithm to simultaneously diagonalize Pauli operators, whose combinations are frequently used to compute observables as well as unitary operations such as in quantum simulation algorithms. Due to the iterative nature of the algorithm, in which we seek to diagonalize all operators over one qubit at a time, we find this algorithm is additionally convenient in adaptations to quantum hardware on which qubit connectivity is very limited. Moreover, we are able to see from analytic arguments that the scaling for depths of the resulting circuits prescribed by our algorithm is exceptionally low, scaling linearly in the total number of qubits and logarithmically in the number of independent generators of the operators. In kind, we can analytically count the total number of two-qubit gates to be competitive with other recent constructive diagonalization algorithms.

In this paper, we have further demonstrated that the circuits we produce maintain short circuit depth and low two-qubit gate count in example Hamiltonians such as random Hermitian operators and those describing molecular systems. It remains to be demonstrated whether the circuit depth obtained here can be substantially improved or is in fact optimal. Likewise, we have seen that there are choices in our algorithm that can be exploited to further accommodate limited connectivity,

though making the optimal such choice may be formulated as a generalized TSP. Therefore, further work can be done to optimize these choices based upon the particular hardware in consideration. Nevertheless, we expect that the framework for diagonalization suggested by our algorithm will enable future work to accommodate such hardware.

Additionally, let us point out that algorithms that form the clusters of commuting operators to be diagonalized must be designed in parallel with these diagonalization algorithms to produce the simplest circuits possible overall. As such, developments in forming commuting clusters of operators would similarly assist in the effort to efficiently diagonalize such clusters overall as well.

ACKNOWLEDGMENTS

We thank Ophelia Crawford, Ridwan Syed, Ewout van den Berg, and Daochen Wang for helpful correspondence. This material is based upon work supported in part by the U.S. Department of Energy, Office of Nuclear Physics under Grants No. DE-SC0021143 and No. DE-FG02-95ER40907. The modules QISKIT [40], NUMPY [41], GALOIS [42], and NETWORKX [38] have been significantly used throughout this project.

-
- [1] Y. Manin, *Computable and Uncomputable* (Sovetskoye Radio, Moscow, 1980) [in Russian].
 - [2] P. Benioff, *J. Stat. Phys.* **22**, 563 (1980).
 - [3] R. P. Feynman, *Int. J. Theor. Phys.* **21**, 467 (1982).
 - [4] R. P. Feynman, *Found. Phys.* **16**, 507 (1986).
 - [5] S. Lloyd, *Science* **273**, 1073 (1996).
 - [6] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, New York, 2010).
 - [7] J. D. Whitfield, J. Biamonte, and A. Aspuru-Guzik, *Mol. Phys.* **109**, 735 (2011).
 - [8] A. J. Daley, I. Bloch, C. Kokail, S. Flannigan, N. Pearson, M. Troyer, and P. Zoller, *Nature (London)* **607**, 667 (2022).
 - [9] W. Kirby, M. Motta, and A. Mezzacapo, *Quantum* **7**, 1018 (2023).
 - [10] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, *Nat. Commun.* **5**, 4213 (2014).
 - [11] U. Fano, *Rev. Mod. Phys.* **29**, 74 (1957).
 - [12] D. F. V. James, P. G. Kwiat, W. J. Munro, and A. G. White, *Phys. Rev. A* **64**, 052312 (2001).
 - [13] M. Paris and J. Řeháček, *Quantum State Estimation*, 1st ed. (Springer, New York, 2004).
 - [14] M. Cramer, M. B. Plenio, S. T. Flammia, R. Somma, D. Gross, S. D. Bartlett, O. Landon-Cardinal, D. Poulin, and Y.-K. Liu, *Nat. Commun.* **1**, 149 (2010).
 - [15] E. M. Murairi, M. J. Cervia, H. Kumar, P. F. Bedaque, and A. Alexandru, *Phys. Rev. D* **106**, 094504 (2022).
 - [16] E. van den Berg and K. Temme, *Quantum* **4**, 322 (2020).

- [17] O. Crawford, B. v. Straaten, D. Wang, T. Parks, E. Campbell, and S. Brierley, *Quantum* **5**, 385 (2021).
- [18] J. Bringewatt, J. Kunjummen, and N. Mueller, [arXiv:2303.15519](https://arxiv.org/abs/2303.15519) (2023).
- [19] D. Gottesman, in *Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics*, edited by S. P. Corney, R. Delbourgo, and P. D. Jarvis (International Press, Cambridge, MA, 1999), pp. 32–43..
- [20] S. Aaronson and D. Gottesman, *Phys. Rev. A* **70**, 052328 (2004).
- [21] E. M. Rains, [arXiv:quant-ph/9703048](https://arxiv.org/abs/quant-ph/9703048) (1997).
- [22] A. Klappenecker and P. K. Sarvepalli, *Proc. R. Soc. A* **463**, 2887 (2007).
- [23] R. Koenig and J. A. Smolin, *J. Math. Phys.* **55**, 122202 (2014).
- [24] R. M. Karp, *Reducibility among combinatorial problems*, in *Complexity of Computer Computations: Proceedings of a Symposium on the Complexity of Computer Computations, Held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and Sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*, edited by R. E. Miller, J. W. Thatcher, and J. D. Bohlinger (Springer, New York, 1972), pp. 85–103.
- [25] T. Volgenant and R. Jonker, *J. Oper. Res. Soc.* **38**, 1073 (1987).
- [26] G. Laporte, H. Mercure, and Y. Norbert, *Oper. Res.* **18**, 203 (1984).
- [27] S. E. Dreyfus, *Oper. Res.* **17**, 395 (1969).
- [28] T. Ibaraki, *SIAM Rev.* **15**, 309 (1973).
- [29] Çetin K. Koç and S. N. Arachchige, *J. Parallel Distrib. Comput.* **13**, 118 (1991).
- [30] P. Høyer and R. Špalek, *Theory of Computing* **1**, 81 (2005).
- [31] D. Miller, L. E. Fischer, I. O. Sokolov, P. K. Barkoutsos, and I. Tavernelli, [arXiv:2203.03646](https://arxiv.org/abs/2203.03646) (2022).
- [32] D. Gottesman, Ph.D. thesis, California Institute of Technology, 1997.
- [33] E. Knill and R. Laflamme, *Phys. Rev. A* **55**, 900 (1997).
- [34] N. J. Cerf and R. Cleve, *Phys. Rev. A* **56**, 1721 (1997).
- [35] A. Vardy, *IEEE Trans. Inf. Theory* **43**, 1757 (1997).
- [36] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. Sohaib Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, J. M. Arrazola, U. Azad, S. Banning, C. Blank, T. R. Bromley, B. A. Cordier, J. Ceroni, A. Delgado, O. Di Matteo, A. Dusko *et al.*, [arXiv:1811.04968](https://arxiv.org/abs/1811.04968) (2018).
- [37] T. Tomesh, K. Gui, P. Gokhale, Y. Shi, F. T. Chong, M. Martonosi, and M. Suchara, in *37th International Cosmic Ray Conference on Rebooting Computing (ICRC)* (IEEE, New York, 2021).
- [38] A. A. Hagberg, D. A. Schult, and P. J. Swart, in *Proceedings of the Seventh Python in Science Conference*, edited by G. Varoquaux, T. Vaught, and J. Millman (SciPy, 2008), pp. 11–15.
- [39] S. B. Bravyi and A. Y. Kitaev, *Ann. Phys. (NY)* **298**, 210 (2002).
- [40] Qiskit contributors, Qiskit: An open-source framework for quantum computing, 2023, doi:[10.5281/zenodo.2573505](https://doi.org/10.5281/zenodo.2573505).
- [41] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant *et al.*, *Nature (London)* **585**, 357 (2020).
- [42] M. Hostetter, Galois: A performant NumPy extension for Galois fields, 2020, <https://github.com/mhostetter/galois>.