Quantum Science and Technology

PAPER

Greedy algorithm based circuit optimization for near-term quantum simulation

Yi Hu[1,2,3], Fanxu Meng[1,2,3], Xiaojun Wang[1,2,3,4], Tian Luan[5], Yulong Fu[5], Zaichen Zhang[1,2,3,4], Xianchao Zhang[1,6] and Xutao Yu[1,2,3,4,*]

[1] School of Information Science and Engineering, Southeast University, Nanjing 210096, People's Republic of China
[2] Frontiers Science Center for Mobile Information Communication and Security, Southeast University, Nanjing 210096, People's Republic of China
[3] Quantum Information Center of Southeast University, Nanjing 210096, People's Republic of China
[4] Purple Mountain Laboratories, Nanjing 211111, People's Republic of China
[5] Yangtze Delta Region Industrial Innovation Center of Quantum and Information Technology, Suzhou 215133, People's Republic of China
[6] Key Laboratory of Medical Electronics and Digital Health of Zhejiang Province, Jiaxing University, Jiaxing 314001, People's Republic of China
[*] Author to whom any correspondence should be addressed.

E-mail: yuxutao@seu.edu.cn

Keywords: noisy intermediate-scale quantum (NISQ) algorithm, Hamiltonian simulation, circuit optimization, circuit depth reduction, greedy algorithm, gate cancellation

OPEN ACCESS

RECEIVED
7 May 2022

ACCEPTED FOR PUBLICATION
16 June 2022

PUBLISHED
30 June 2022

Abstract

Simulating quantum systems is believed to be one of the most important applications of quantum computers. On noisy intermediate-scale quantum (NISQ) devices, the high-level circuit designed by quantum algorithms for Hamiltonian simulation needs to consider hardware limitations such as gate errors and circuit depth before it can be efficiently executed. In this work, we develop a hardware-agnostic circuit optimization algorithm to reduce the overall circuit cost for Hamiltonian simulation problems. Our method employ a novel sub-circuit synthesis in intermediate representation and propose a greedy ordering scheme for gate cancellation to minimize the gate count and circuit depth. To quantify the benefits of this approach, we benchmark proposed algorithm on different Hamiltonian models. Compared with state-of-the-art generic quantum compilers and specific quantum simulation compiler, the benchmarking results of our algorithm show an average reduction in circuit depth by 16.5× (up to 64.1×) and in gate count by 7.8× (up to 23.7×). This significant improvement helps enhance the performance of Hamiltonian simulation in the NISQ era.

1. Introduction

While building a fault-tolerant quantum computer is still infeasible, recent progress in quantum hardware reveals a significant outperformance of a quantum processor on certain computational tasks which would take thousands of years on a state-of-the-art classical supercomputer [1]. Quantum computers are reaching the stage where specific problems can be achieved within desirable accuracy [2]. We are entering into the noisy intermediate-scale quantum (NISQ) era, which is characterized by quantum computers with noisy qubits range from 50 to 100 and lacking full scale quantum error correction [3]. In order to make the best use of current quantum hardware, several restrictions must be taken into account: (1) limited numbers of qubits, (2) connective constraints of qubits, (3) two-qubit gate errors and limited circuit depth resulted from coherent and incoherent errors. So the most important question is how to find useful applications of NISQ-era hardware. For reaching this goal, quantum algorithms that can be executed on current quantum devices are developed to deal with practical problems [4–10].

Quantum simulation is possibly to be one of the first practical applications of quantum computing, it has broad applications in quantum many-body physics [11] and quantum chemistry [12–14]. Simulating the dynamics of quantum systems is extremely hard for classical computers as a result of the exponential

growth of the system size. Quantum computers promise to efficiently solve this problem, however, the gate cost of simulating quantum systems is still prohibitive [15]. The main challenge for quantum simulation is to construct an efficient circuit that closely approximates the time evolution of the Hamiltonian of a quantum system.

Different approaches have been proposed for efficient quantum simulation, including product formulas [11, 16–18], linear-combination-of-unitaries [19, 20], truncated Taylor series [21, 22], quantum walk [23], quantum signal processing [24], and multi-product formulas [25, 26]. Product formulas are utilized, as is most common, when the Hamiltonian can be decomposed as a sum of separate terms ($H = \sum_j H_j$), such that the time evolution of each H_j is easily implemented. Then the time evolution of the system can be described by $U = \mathrm{e}^{-\mathrm{i}t\sum_j H_j}$. A standard approach in product formulas is Trotter–Suzuki decomposition

$$\mathrm{e}^{-\mathrm{i}tH} \approx \lim_{k\to\infty}\left(\prod_{j=1}^{L}\mathrm{e}^{-\mathrm{i}(t/k)H_j}\right)^k, \tag{1}$$

this Trotter–Suzuki formula is referred to as first-order approximation. Each individual time evolution operator $\mathrm{e}^{-\mathrm{i}(t/k)H_j}$ is made up of efficiently implementable quantum gates, which can be run on a quantum computer. The approximation errors, which scale as $\mathcal{O}\left(t^2/k\right)$, arise from noncommutative terms in the Hamiltonian. If all the terms in the Hamiltonian commute, the exponent of sum of all terms is equal to products of the separate exponents (i.e., $\mathrm{e}^{-\mathrm{i}t\sum_j H_j} = \prod_j \mathrm{e}^{-\mathrm{i}tH_j}$). If some terms do not commute, which naturally exists in physical systems, $\left(\prod_j \mathrm{e}^{-\mathrm{i}(t/k)H_j}\right)^k$ asymptotically approximate $\mathrm{e}^{-\mathrm{i}tH}$ for large k. $\prod_j \mathrm{e}^{-\mathrm{i}(t/k)H_j}$ is called one *Trotter step*, and the circuit will repeat k times.

To make Trotter error arbitrarily small, one can increase the repetition number. However, this also increases the depth of the circuit which is limited on NISQ devices. For saving the simulation cost without losing accuracy, one can reduce k by using high-order approximation. For instance, the second-order approximation

$$U_2 = \left(\prod_{j=1}^{L}\exp\left(-\mathrm{i}tH_j/2k\right)\prod_{j=L}^{1}\exp\left(-\mathrm{i}tH_j/2k\right)\right)^k \tag{2}$$

can reduce k from $O\left((tL\Lambda)^2/\epsilon\right)$ (first-order) to $O\left((tL\Lambda)^{1+\frac{1}{2}}/\epsilon^{\frac{1}{2}}\right)$ [27, 28], where Λ is the magnitude of the strongest term and ϵ is the desired error threshold. An alternative way of decreasing cost is to find methods of reducing depth and gate count in one Trotter step, which is of most concern in this paper.

Here, we propose several techniques to address this problem. First of all, we develop a novel sub-circuit synthesis based on [29] to decompose k-local Hamiltonian in Pauli intermediate representation (IR). Then follows the technique which exploits the advantages of reordering Trotter sequences. By rearranging the individual terms in Trotter formula, we search for a superior ordering scheme that minimizes the circuit depth and the gate count. The method for reordering is based on greedy algorithm and provides great benefits in gate cancellation which contributes to the reduction of circuit depth. Further, we parallelize the operations, leaving the removed gate count unchanged but additionally reducing the circuit depth. Moreover, by combining the strategy proposed in [30] for reducing CNOT gates with our ordering scheme, we further decrease the amount of CNOT gates. The techniques we developed are applicable to quantum simulation kernels (i.e., the circuit that implement the operator of equation (1)), which appear in a wide range of algorithms [11–14], and are independent of the underlying hardware.

This paper is organized as follows. In section 2 we introduce proposed sub-circuit synthesis in Pauli IR. In section 3 we explains how the error of Trotter–Suzuki approximation and the gate cancellation procedure depend on the Trotter ordering scheme, both of which influence the depth of overall circuit. Section 4 illustrates the proposed method for order search based on greedy algorithm. We introduce further optimizations of the circuit in section 5. Evaluation and benchmarking are exhibited in section 6. After discussion we conclude in section 7.

2. Sub-circuit synthesis in Pauli IR

For efficiently executing a quantum simulation circuit on NISQ devices, the Hamiltonian evolution operator in equation (1) need to be decomposed into executable single- and two-qubit operations, and compilers carry out this routine for every local Hamiltonians. The state-of-the-art generic compilers (e.g., Qiskit [31], t|ket⟩ [32]) will convert the multi-qubit operator into a sequence of their elementary gates such as single-qubit rotations and CNOT gates. Due to the lack of high-level IR of Pauli strings, they fail to

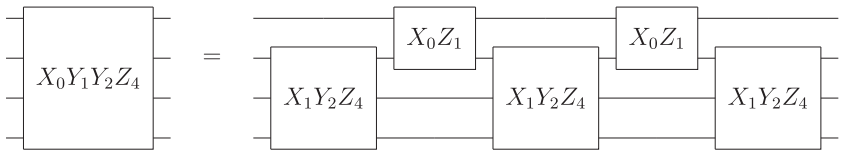


Figure 1. Intermediate circuit synthesis for $X_0Y_1Y_2Z_4$ operator, the $X_1Y_2Z_4$ operators are further decomposed in figure 2.

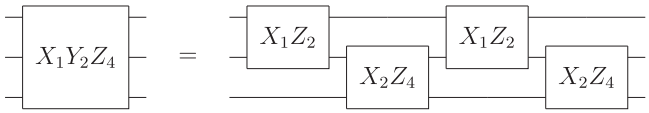


Figure 2. Intermediate circuit synthesis for $X_1Y_2Z_4$ operator.

leverage the optimization opportunities of synthesizing Pauli IR. Some compiler-level IR optimization frameworks such as Paulihedral [33] are designed to settle this problem, and quality improvement can be seen. Therefore, we capitalize on the intermediate optimization, proposing a novel sub-circuit synthesis in Pauli IR to better optimize quantum simulation kernels.

First, we introduce some analytic identities found by Clinton *et al* [29]. According to supplementary lemmas 7 and 8 in [29], for a Hamiltonian $H = \frac{1}{2i}[h_1, h_2]$ where h_1 and h_2 anti-commute and both square to identity, the evolution operator $U(t) = e^{itH}$ can be decomposed in two ways:

$$U(t) = e^{it_1h_1}\, e^{it_2h_2}\, e^{it_2h_1}\, e^{it_1h_2}, \tag{3}$$

or

$$U(t) = e^{it_1h_2}\, e^{-i\phi h_1}\, e^{it_2h_2}\, e^{i\phi h_1}\, e^{it_1h_2} \tag{4}$$

where t_1, t_2, ϕ are pulse times given by evolution time t. Note that equation (3) satisfies under conditions but equation (4) does not. Followed by the demonstration in [29], we can decompose an evolution operator e^{itH} recursively using equation (4) decomposition then end with equation (3) decomposition, we apply this principle in our synthesis techniques.

Before introducing our intermediate circuit synthesis, let us define some notations that are used in this paper. The Hamiltonians in quantum simulation kernel we need to synthesize are expressed in the form of tensor product of Pauli matrices $H = \sigma_0 \otimes \sigma_1 \otimes \cdots \otimes \sigma_{n-1}$, where $\sigma_i \in \{I, X, Y, Z\}$ describes the Pauli operator acting on the ith qubit, and n is number of qubits. For simplicity, in the rest of the paper we omit the identity operators and denote a time-evolution operator of the Hamiltonian by a Pauli string. For example, we denote the evolution operator $\exp(itX_0 \otimes Y_1 \otimes Y_2 \otimes I_3 \otimes Z_4)$ as $X_0Y_1Y_2Z_4$ which corresponding to a four-local Hamiltonian $X_0 \otimes Y_1 \otimes Y_2 \otimes Z_4$.

Following the decomposition schemes we described above, our intermediate circuit synthesis works as described below:

(a) For a k-local Hamiltonian H, we decompose e^{itH} using equation (4) while choosing h_1 as a two-local Hamiltonian and h_2 as a $(k-1)$-local Hamiltonian then implement the same decomposition procedure for $(k-1)$-local Hamiltonian iteratively until $k-1=3$. For the rest three-local Hamiltonians, we apply equation (3) decomposition picking both h_1 and h_2 as two-local Hamiltonians so that all the decomposed operators are two-local interactions.

(b) For each Hamiltonian need to be synthesized ($k \geqslant 3$), h_1, h_2 are computed by Pauli algebra.

We synthesize $X_0Y_1Y_2Z_4$ as an example. In the first iteration, we rewrite $X_0 \otimes Y_1 \otimes Y_2 \otimes Z_4$ as $X_0 \otimes \left(\frac{1}{2i}[Z_1, X_1]\right) \otimes Y_2 \otimes Z_4$ such that we can choose $h_1 = X_0 \otimes Z_1, h_2 = X_1 \otimes Y_2 \otimes Z_4$, then utilize the equation (4) decomposition.

In the second iteration, the procedure is repeated for $h_2 = X_1 \otimes Y_2 \otimes Z_4$, and we select $h_1' = X_1 \otimes Z_2, h_2' = X_2 \otimes Z_4$. As a result, a k-local operator is decomposed into two-local operators. The circuit that explains this decomposition flow is depicted in figures 1 and 2.

After the proposed intermediate circuit synthesis, quantum simulation kernel is converted into two-local interaction sequences. The advantage of this synthesis is that it provides an opportunity for optimizing circuit by flexibly adjust the order of these two-local operators. The techniques will be described in the following sections.

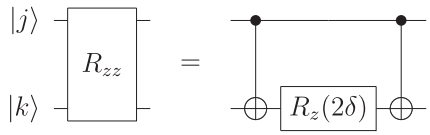


Figure 3. Circuit of a two-local $Z \otimes Z$ interaction operator.

3. Impact of Trotter terms ordering

The choice of ordering scheme has formerly been demonstrated to hold a great impact on the Trotterization [12–14, 17, 18]. Specific tasks determine the utilization of the Trotter ordering scheme, which affects the performance of Trotterization in terms of the circuit length. In the context of quantum chemistry, *a magnitude ordering* is physically meaningful, due to the fact that terms with higher magnitude are more likely to correspond to stronger interactions. Utilizing this ordering scheme can significantly reduce the error in one Trotter step but is not optimal in gate count [12, 17, 18]. Alternatively, *a lexicographic ordering* is applied for saving as much gate cost as possible through cancellation, as it maximizes the similarity of adjacent terms [12–14, 17, 18].

Choosing incompatible Trotter ordering schemes will differ in the length of the circuit. Minimizing the Trotter error may potentially acquire an order which is not optimal in terms of the circuit depth, where a lexicographic ordering holds a great advantage. Since we aim at optimizing the circuit of an arbitrary k-local Hamiltonian simulation, where the minimization of Trotter error which is useful to specific molecular Hamiltonians could possibly be outweighed by the effect of gate cancellation, we devote to addressing the latter one.

As the number of all possible orderings grows factorially with the amount of individual Trotter terms, it is difficult to find an optimal ordering scheme by performing an exhaustive search. However, for gate cancellation procedures, it is possible to find a superior order that minimizes the gate count as well as the circuit length, when the Hamiltonian is specified. Several approaches have been developed to this end [12–14, 17, 18], but they are simply based on the lexicographic ordering. Combining the greedy algorithm with the basic lexicographic ordering, we establish a method which significantly outperforms the previous ones.

After the sub-circuit synthesis we have discussed above, quantum simulation kernel is divided into two-local interaction sequences. So we first introduce the decomposition of the evolution operators for two-local Hamiltonians, which helps illustrate our techniques. A two-qubit $Z \otimes Z$ interaction operator can be decomposed as

$$\mathrm{e}^{-\mathrm{i}\delta Z_j \otimes Z_k} = \mathrm{CNOT}_{jk} \left(I_j \otimes R_z(2\delta)_k\right) \left(\mathrm{CNOT}_{jk}\right), \tag{5}$$

where j, k indicate the qubits on which the operator act. To make it more explicit, we give the gate level implementation as shown in figure 3.

This is one of the commonly used decompositions, which is also our choice, and we term the whole circuit in figure 3 as a two-local $Z \otimes Z$ interaction gate.

Here, we state some preliminary knowledge. Considering a Hermitian operator M with its spectral decomposition $M = S\Lambda S^\dagger = \sum_j \lambda_j |m_j\rangle \langle m_j|$, it holds that exponentiation of the operator M is equivalent to the sum of exponentiation of its eigenvalues,

$$\mathrm{e}^{\mathrm{i}\theta M} = S\,\mathrm{e}^{\mathrm{i}\theta\Lambda} S^\dagger = \sum_i \mathrm{e}^{\mathrm{i}\theta\lambda_j} |m_j\rangle \langle m_j| . \tag{6}$$

We focus on the diagonalization matrices $D = S^\dagger$ (i.e., $DMD^\dagger = \Lambda$). Pauli Z operator is already diagonal, which means $D_z = I$. The other two Pauli operators X and Y can be diagonalized to $\Lambda = Z$ with diagonalization matrices D_x and D_y separately. And it follows that

$$\mathrm{e}^{\mathrm{i}\theta X} = \mathrm{e}^{\mathrm{i}\theta D_x^\dagger Z D_x} = D_x^\dagger\, \mathrm{e}^{\mathrm{i}\theta Z} D_x = D_x^\dagger R_z(\theta) D_x \tag{7}$$

$$\mathrm{e}^{\mathrm{i}\theta Y} = \mathrm{e}^{\mathrm{i}\theta D_y^\dagger Z D_y} = D_y^\dagger\, \mathrm{e}^{\mathrm{i}\theta Z} D_y = D_y^\dagger R_z(\theta) D_y. \tag{8}$$

A general n-Pauli operator can be exponentiated in the same way by conducting tensor product of the diagonalization matrices corresponding to each of the terms, i.e.

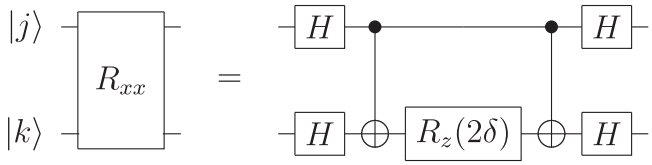


Figure 4. Circuit of a two-local $X \otimes X$ interaction operator.

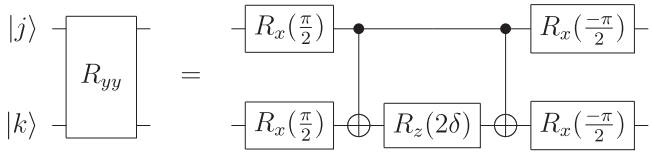


Figure 5. Circuit of a two-local $Y \otimes Y$ interaction operator.

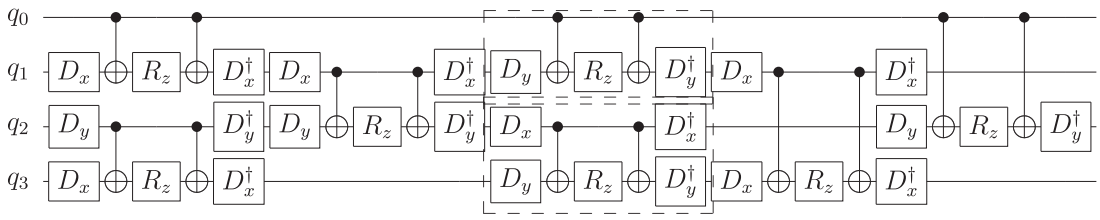


Figure 6. Original circuit.

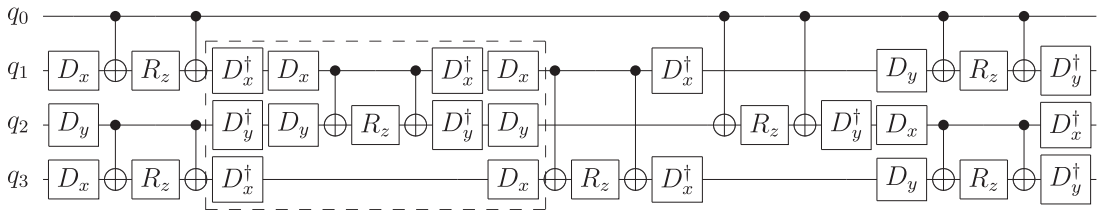


Figure 7. Circuit after reordering.

$$e^{i\theta P\otimes P\ldots} = e^{i\theta(D_p^\dagger\otimes D_p^\dagger\ldots)(Z\otimes Z\ldots)(D_p\otimes D_p\ldots)}$$
$$= (D_p^\dagger \otimes D_p^\dagger \ldots)e^{i\theta(Z\otimes Z\ldots)}(D_p \otimes D_p \ldots), \quad (9)$$

where $P \in \{X, Y, Z\}$. It implies that we can decompose any k-local interaction operators by this formula.

In the case of two-local interaction, for example, the Pauli $X \otimes X$ and $Y \otimes Y$ interaction operators can be decomposed to gate level as shown below in figures 4 and 5 (and likewise for other two-local interaction operators).

Here we choose diagonalization operators $D_x = H$ and $D_y = R_x(\pi/2)$ (where H is Hadamard transform operator and $R_x(\theta)$ is single-qubit rotation operator about the $\hat{x}$ axis, both of which are also corresponding quantum gates), but note it is not the only choice. The benefits of this form of decomposition are manifest, since the adjacent D_x and $D_x^\dagger$ (or D_y and $D_y^\dagger$) in the circuit will be canceled out immediately. It should be noted that all the diagonalization gates of a two-local interaction are possibly to be canceled out, which results in a significant reduction of the gate count. An example is shown in figure 7.

After rearranging the two-local gate sequences, the 10 diagonalization gates in the dashed box of figure 7 can all be canceled out, which will reduce the depth of circuit by 4. While with the ordering of original circuit in figure 6, the circuit depth can be reduced only by 2. As we can see in figure 7, the deposition of a two-local gate determines the cancellation on both the left side and the right side of it, and will affect the rest of the circuit. The problem of finding a superior order which maximizes the gate cancellation of the overall circuit is tricky. In the next section, we will illustrate our strategies to address this problem.

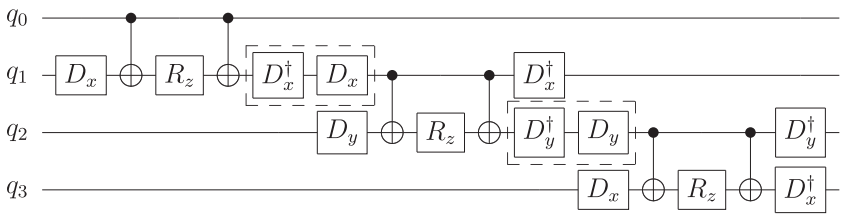


Figure 8. The layout of one layer before parallelization, where the order of Trotter terms now is $Z_0X_1 \to X_1Y_2 \to Y_2X_3$.

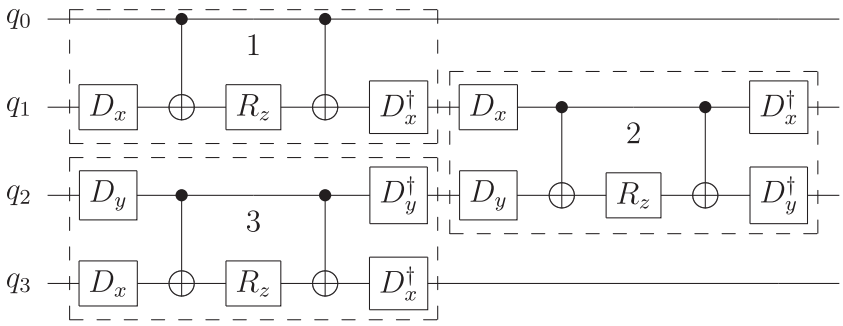


Figure 9. Parallelized circuit of one layer, where the order of Trotter terms in this layer is $Z_0X_1 \to Y_2X_3 \to X_1Y_2$.

4. Order search strategies

Since exhaustively searching all possible orders is impractical, alternatives need to be found. We cope with this problem by applying a greedy algorithm based heuristic search method. In our scenario, the reduction of gate count is the target of the optimization problem. Previous works have shown that *a lexicographic* ordering is beneficial for the purpose of gate cancellation. This ordering scheme intends to derive a maximum similarity of adjacent Pauli strings. But it is basically a numerical ordering either with respect to the fermionic operators or with respect to the individual Pauli operators [17, 18]. As we have shown above, the placement of a two-local gate influences the cancellation on both the left side and the right side of it. Merely sorting the Trotter sequences lexicographically may probably not give rise to the maximum amount of gate cancellation. Here we propose our ordering strategies and explain how it comes from greedy algorithm.

4.1. Trotter layers partition procedure

In the first stage of the algorithm, we partition the circuit into layers such that the reduction of gate count is locally optimal in each layer. That's where the greedy algorithm comes in. For reaching this goal, we develop several strategies.

First, we group two-local gates into different local Pauli-index pools in terms of the similarity of Pauli strings. Here we use a sequence X_0Y_2 to denote a two-local gate $e^{i\theta X_0Y_2}$ (qubit indices start at 0). A sequence X_0X_3 has the same Pauli string with X_0Y_2 on qubit 0, so they are both grouped into the $\{\}_{X_0}$ index pool. X_0Y_2 and Y_2Z_3 will both be grouped into the $\{\}_{Y_2}$ index pool consequently. Note that a two-local gate is labeled by two Pauli-index pools.

Next, we pick out the gates from Pauli-index pools one by one for each layer, following a strategy that adjacent gates in one layer are chosen from one Pauli-index pool where the second index of last gate and the first index of next gate in. Subsequently, all the two-local gates in a single layer are arrayed in a stepped arrangement from smallest qubit number to the largest, so that the operations in this layer can be easily parallelized for further optimization. To make it concrete, we demonstrate a layout of one layer before parallelization in figure 8, where we chose sequences $[Z_0X_1, X_1Y_2, Y_2X_3]$ for this layer.

These strategies we proposed are determined by the following two considerations. First, the diagonalization gates on the one side of a two-local gate in the intermediate qubits will be canceled out immediately (see dashed boxes in figure 8). Second, we can parallelize two operations in every three adjacent operations without changing the amount of removed gates (where operations are shown in figure 9 as numbered blocks), which maximizes the decrease of circuit depth.

In addition, we give priority to choosing the gates from Pauli-index pools where there are other gates remained in the corresponding pools can be chosen in the next layer, which means that the diagonalization

Algorithm 1. Order search algorithm.

Input: a list of sequences of two-local interaction gates $[(P_jP_k)^{(n)}]$
Output: a list of ordered sequences of two-local interaction gates $[(P_jP_k)^{(n)}]_{\text{ordered}}$

function OrderSearch ($[(P_jP_k)^{(n)}]$)
  for all $P_jP_k \in (P_jP_k)^{(n)}$ **do**
    IndexPools: $\{\}_{P_j}, \{\}_{P_k} \leftarrow P_jP_k$ (▷) Group two-local gates P_jP_k into Pauli-index pools $\{\}_{P_j}, \{\}_{P_k}$.
  end for
  $[(P_jP_k)^{(n)}]_{\text{ordered}} \leftarrow \emptyset$ (▷) Set $[(P_jP_k)^{(n)}]_{\text{ordered}}$ to empty-list
  nextlayer $\leftarrow$ true
  $\{\}_{\text{lastlayerpools}} \leftarrow \emptyset$ (▷) Store IndexPools in last layer
  CoIndex $\leftarrow \emptyset$ (▷) Store concatenation index among adjacent gates in one layer
  repeat (▷) Pick out two-local gates one by one for each layer to conduct gate cancellation procedure
    if nextlayer = true **then** (▷) Choose first two-local gate of a layer
      for P_jP_k in $(P_jP_k)^{(n)}$ **do**
        if P_j, P_k in $\{\}_{\text{lastlayerpools}}$ **then**
          $[(P_jP_k)^{(n)}]_{\text{ordered}} \leftarrow P_jP_k$ (▷) Append P_jP_k to $[(P_jP_k)^{(n)}]_{\text{ordered}}$
          $[(P_jP_k)^{(n)}] \leftarrow [(P_jP_k)^{(n)}] - P_jP_k$, $\{\}_{\text{lastlayerpools}} \leftarrow \{\}_{P_j}, \{\}_{P_k}$, CoIndex $\leftarrow P_k$, nextlayer $\leftarrow$ False, break (▷)Remove P_jP_k from $[(P_jP_k)^{(n)}]$, update last layer IndexPools with new $\{\}_{P_j}, \{\}_{P_k}$, find CoIndex and jump out of FOR loop
        else if $j = 0$ **then**
          $[(P_jP_k)^{(n)}]_{\text{ordered}} \leftarrow P_0P_k$
          $[(P_jP_k)^{(n)}] \leftarrow [(P_jP_k)^{(n)}] - P_0P_k$, $\{\}_{\text{lastlayerpools}} \leftarrow \{\}_{P_0}, \{\}_{P_k}$, CoIndex $\leftarrow P_k$, nextlayer $\leftarrow$ false, break
        end if
      end for
    else
      repeat (▷) Choose the rest gates of a layer
        for CoIndex in $\{\}_{\text{CoIndex}}$ and P_jP_k in $(P_jP_k)^{(n)}$ **do**
          if P_j = CoIndex and P_j, P_k in $\{\}_{\text{lastlayerpools}}$ **then**
            $[(P_jP_k)^{(n)}]_{\text{ordered}} \leftarrow P_jP_k$
            $[(P_jP_k)^{(n)}] \leftarrow [(P_jP_k)^{(n)}] - P_jP_k$, $\{\}_{\text{lastlayerpools}} \leftarrow \{\}_{P_j}, \{\}_{P_k}$, CoIndex $\leftarrow P_k$, break
          end if
        end for
      until nextlayer = True
    end if
    Reorder existent two-local sequences in $[(P_jP_k)^{(n)}]_{\text{ordered}}$ for parallelization
  until no P_jP_k in rest $[(P_jP_k)^{(n)}]$ contribute to gate cancellation
  append the rest two-local sequences to $[(P_jP_k)^{(n)}]_{\text{ordered}}$ list with parallelization
  return $[(P_jP_k)^{(n)}]_{\text{ordered}}$
end function

gates of the two-local gates we chose can be canceled out with those in the next layer. And further, we select the gates that act on as much qubits as possible in a layer (i.e., $[X_0X_1, X_1Y_2, Y_2Z_3, Z_3X_4]$ instead of $[X_0X_1, X_1Y_2, Y_2Y_4]$), which exploits the advantage of parallelization.

4.2. Gate allocation among Trotter layers

As for the third strategy, the gates in each layer are chosen from the Pauli-index pools in last layer (except for the first layer). Additionally, for the gates act on qubit 0, which are always on the top of a layer, we can simply select a lexicographic order among layers.

Since every diagonalization gate of a two-local gate can be canceled out at most once, regardless of which one is canceled out with, we move all the gate cancellation procedures ahead quickly as the reason of the third strategy. Explicitly, if two diagonalization gates on the left side of a two-local gate can both be canceled out with those on the right side of last layer, this two-local gate would be the first choice, and the case of only one diagonalization gate can be canceled out would be the second choice. As an example, the choice of gates as shown in the dashed boxes of figure 6 is outperformed by that on the corresponding position of figure 7.

Joining the adjacent gate cancellation procedures together, all the diagonalization gates that contribute to gate reduction are exhausted. That is, these locally optimal reductions, combined together, have resulted in a globally optimal solution in terms of the amount of removed gates. Furthermore, benefiting from the proposed ordering strategies, we can parallelize possible operations among adjacent layers without decreasing the amount of gate cancellation, which additionally reduces the circuit depth. In the last step of our ordering strategies, the rest of the two-local gates whose diagonalization gates cannot be canceled out are placed at the end of circuit with parallelization. The complete order search algorithm is depicted in algorithm 1.

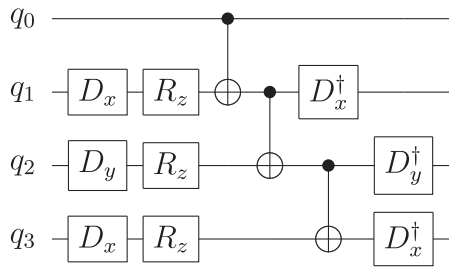


Figure 10. Optimized first layer of circuit after gate cancellation procedure and DFS CNOT reduction procedure, we used the same example as in figure 8 (i.e., $[Z_0X_1, X_1Y_2, Y_2X_3]$ for this layer).

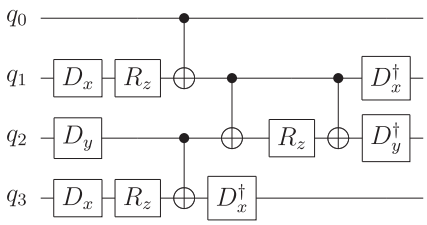


Figure 11. Optimized first layer of circuit after gate cancellation procedure and EC CNOT reduction procedure, used the same example with figure 10.

5. Further optimizations

As we mentioned previously, error rates of the two-qubit gates on NISQ devices are non-negligible, which makes it vital to minimize the cost of two-qubit gates. In this section we discuss the second stage of our algorithm, which further optimizes the circuit by integrating the approach proposed in [30] into our gate reduction framework, for decreasing the amount of CNOT gates. To begin with, a briefly review of the approach we will employ is necessary.

In [30], the authors developed two hardware independent approaches for reducing the count of CNOT gates in the circuit. The first one is based on edge coloring (EC) method where parallelization of circuit takes priority, while the second one is based on depth first search (DFS) which holds an advantage in the amount of gate reduction. Both of the two approaches derived from the theorem 1 that they found as described in paper [30]. On account of the decomposition form we use, our two-qubit interaction operators match with the form of operators which can be optimized under the condition of theorem 1. That is to say, these two approaches can be grafted onto our optimization framework after the ordering and gate cancellation routine. Here we elucidate more details of implementation.

According to theorem 1 in [30], an operator V_1 of the form

$$V_1 = \left(\mathrm{CNOT}_{jk}\right)\left(I_j \otimes R_z(2\gamma_l)_k\right)\left(\mathrm{CNOT}_{jk}\right) \tag{10}$$

can be replaced by

$$V_2 = \left(\mathrm{CNOT}_{jk}\right)\left(I_j \otimes R_z(2\gamma_l)_k\right) \tag{11}$$

under certain conditions, where the operator V_1 is exactly our two-qubit $Z \otimes Z$ interaction operator in equation (5) if we substitute γ_l with δ. (In our case, δ describes a small time-step of Trotterization.) Following the techniques they proposed, the first CNOT gate of corresponding operators in the first layer of our circuit can be eliminated. We demonstrate this with examples based on the DFS method and EC method respectively, as shown in figures 10 and 11.

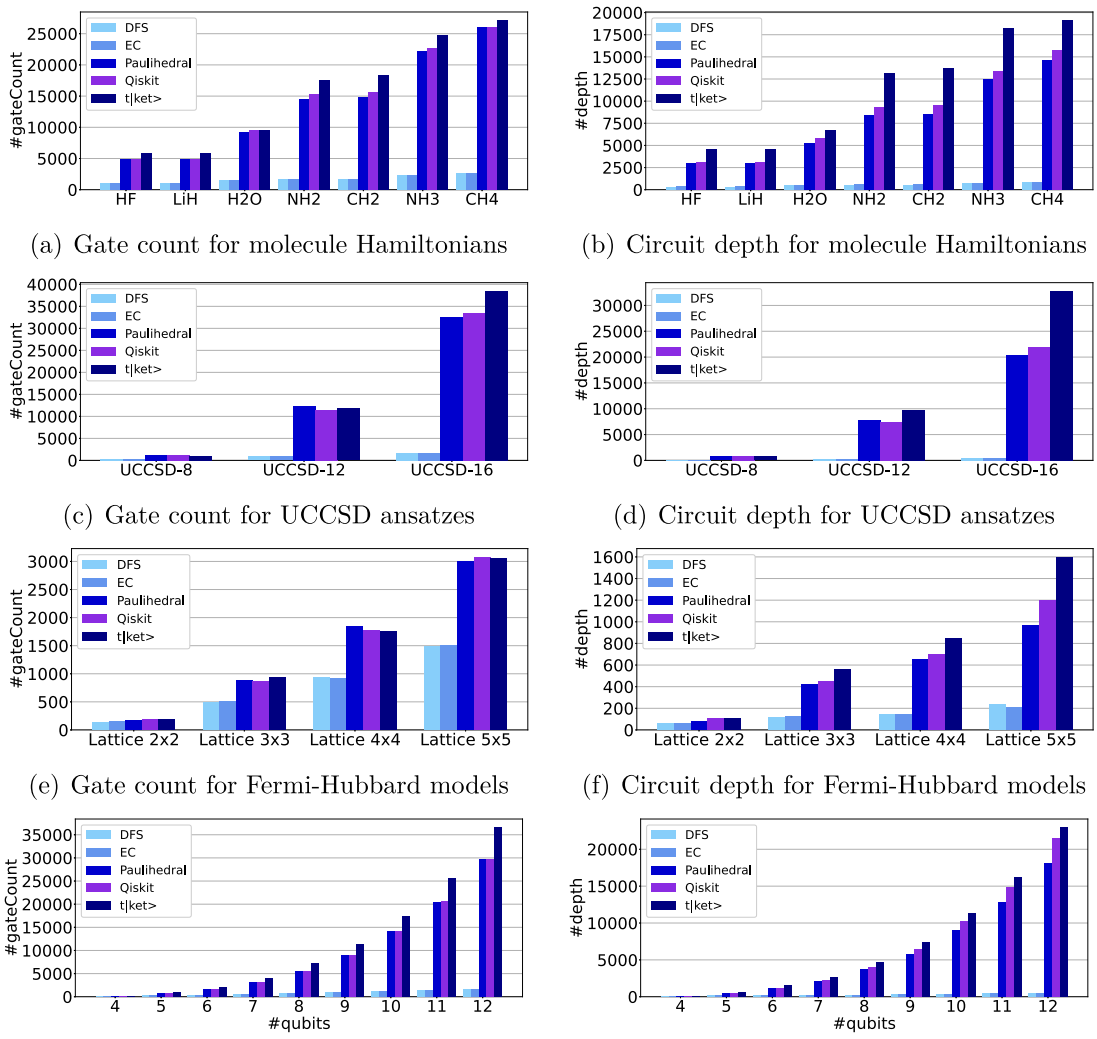
Utilizing the DFS based method for CNOT gates reduction, we can diminish the amount of CNOT gates up to $N-1$, where N is the amount of qubits. In the case of EC based method, the gate count can be reduced up to $N/2$, but with lower circuit depth when the amount of qubits is large. The overall circuit optimization algorithm is summarized in algorithm 2.

Since the cost of total gate count and circuit depth are both crucial indicators of the performance of an optimization algorithm on NISQ devices, we evaluate both of the two optimization pathways with various circuit models. The synthetic performances of them will be shown in the next section.

Algorithm 2. Circuit optimization algorithm.

Input: original circuit of an arbitrary two-local Hamiltonian simulation
Output: optimized circuit of an arbitrary two-local Hamiltonian simulation

```
if DFS path then
    execute order search routine without parallelizing operations in the first layer
    conduct gate cancellation procedure
    use DFS based CNOT gates reduction subroutine
else if edge coloring path then
    execute the complete order search routine
    conduct gate cancellation procedure
    use edge coloring based CNOT gates reduction subroutine
end if
output the optimized circuit
```



(a) Gate count for molecule Hamiltonians (b) Circuit depth for molecule Hamiltonians

(c) Gate count for UCCSD ansatzes (d) Circuit depth for UCCSD ansatzes

(e) Gate count for Fermi-Hubbard models (f) Circuit depth for Fermi-Hubbard models

(g) Gate count for random Hamiltonians (h) Circuit depth for random Hamiltonians

Figure 12. Evaluation results of molecule Hamiltonians, UCCSD ansatzes, Fermi–Hubbard models, and random Hamiltonians. **DFS** and **EC** denote the depth first search and edge coloring optimization paths, which significantly outperform **Paulihedral**, **Qiskit**, and **t|ket⟩** compilers.

6. Evaluation

In this section, we exhibit results of some representative benchmarks for constructing baselines by which to compare the performance of our optimization algorithm with other state-of-the-art optimizers.

6.1. Experiment setup

Benchmarks: to cover as many varieties of applications as possible, we select different kinds of Hamiltonians as our benchmarks, including seven molecule Hamiltonians generated by PySCF [34]

Table 1. The complete benchmarking results of DFS, EC, Paulihedral, qiskit, and t|ket⟩.

	DFS		EC		Paulihedral		qiskit		tket	
	Total	Depth	Total	Depth	Total	Depth	Total	Depth	Total	Depth
HF	1070	338	1081	362	4884	2966	4810	3059	5888	4525
LiH	1070	338	1081	362	4884	2966	4810	3059	5888	4525
H_2O	1584	512	1593	511	9234	5253	9466	5844	9567	6712
NH_2	1704	560	1670	565	14 532	8372	15 237	9274	17 529	13 166
CH_2	1704	560	1670	565	14 871	8576	15 631	9540	18 320	13 774
NH_3	2365	743	2374	787	22 179	12 516	22 639	13 396	24 814	18 255
CH_4	2714	852	2706	853	25 982	14 634	26 058	15 755	27 124	19 113
UCCSD-8	287	145	296	151	1171	761	1259	874	917	761
UCCSD-12	896	269	895	291	12 320	7800	11 440	7503	11 811	9803
UCCSD-16	1622	511	1620	511	32 530	20 470	33 530	21 898	38 460	32 772
Fermi–Hubbard 2 × 2	139	59	151	58	166	82	188	107	182	109
Fermi–Hubbard 3 × 3	489	118	502	128	890	421	870	453	933	561
Fermi–Hubbard 4 × 4	941	142	924	146	1835	656	1764	699	1758	845
Fermi–Hubbard 5 × 5	1491	232	1505	204	2996	968	3075	1201	3056	1597
Random-4	119	78	118	79	168	110	140	89	174	129
Random-5	282	149	284	150	745	504	711	493	953	654
Random-6	420	182	426	186	1697	1167	1665	1185	2098	1527
Random-7	619	223	612	225	3152	2170	3123	2273	3979	2712
Random-8	826	264	842	271	5537	3668	5527	3972	7173	4630
Random-9	1023	329	988	312	8913	5756	8870	6443	11 363	7394
Random-10	1174	371	1138	363	14 139	8997	14 066	10 176	17 456	11 294
Random-11	1360	431	1351	437	20 445	12 884	20 642	14 915	25 646	16 272
Random-12	1589	524	1567	498	29 719	18 138	29 705	21 550	36 573	22 957

(HF, LiH, H_2O, NH_2, CH_2, NH_3, CH_4) and VQE UCCSD ansatzes [7] of three sizes. We also prepare the Fermi–Hubbard models of four square lattice sizes, which are well known in condensed matter physics to describe the interactions between particles in a lattice. And finally we choose random Hamiltonians with the number of qubits range from 4 to 12 for a more comprehensive assessment. For the randomly generated Hamiltonians, we set the amount of Pauli strings increase polynomially with respect to the size of the qubits.

Metrics: we use the total gate count and depth of circuit as metrics to evaluate the performance of different quantum circuit optimizers. Due to the diverse circuit synthesis, different gate sets for representing a quantum circuit may give rise to different gate count and circuit depth. To establish a fair comparison, we compute the total gate count by adding up all the single-qubit gates and CNOT gates after circuit optimization, as these gates can be executed on NISQ devices directly. And the same for circuit depth.

Comparisons: we compare our circuit optimization algorithm with two generic state-of-the-art quantum compilers Qiskit [31] and t|ket⟩ [32], and a specific quantum simulation compiler Paulihedral [33]. Qiskit compile quantum simulation kernels into Ising type gates in its elementary gate library then apply the level 3 optimization, t|ket⟩ compiler has particular techniques based on simultaneous diagonalization for quantum simulation subroutine then follows 'full-pass' optimization. Paulihedral performs the block-wise optimization for quantum simulation kernels, it employs a specific Pauli IR that can preserve high-level semantics then conduct its block scheduling passes for gate cancellation utilizing lexicographic ordering. We compare both DFS and EC optimization paths with the above compilers.

6.2. Comparison result

Figure 12 shows the results of evaluation, note that due to the huge gap between DFS (EC) optimization and other compilers, the results of DFS and EC are difficult to read, it also reveals the great advantages of our optimization algorithm. Comparing with Qiskit, DFS (EC) optimization can reduce total gate count by 7.5× (7.5×), circuit depth by 15.4× (15.2×) on average. The reduction in total gate count and circuit depth when comparing with t|ket⟩ are 8.5× (8.5×) and 19.8× (19.5×) on average, with respect to DFS (EC) optimization. The reason of low efficiency in t|ket⟩ is probably the high overhead induced by simultaneous diagonalization technique. As we can see, generic compilers perform poorly in quantum simulation kernel optimization tasks when compared with our algorithm.

Next we turn to the comparison with Paulihedral, a specific quantum simulation compiler. The results in figure 12 exhibits that DFS (EC) optimization can still reduce total gate count by 7.5× (7.5×), and circuit depth by 14.3× (14×) on average, which means that our techniques for synthesizing quantum simulation kernels combined with greedy ordering scheme for gate cancellation greatly outperforms the lexicographic ordering scheme. The complete benchmarking results are presented in table 1.

For the effect of DFS and EC optimization paths, it can be seen that in most cases their performances are similar, while in the situation of random generated Hamiltonians, EC optimization method can yields lower circuit depth when the size of circuits is large. It suggest that for a generic quantum simulation kernel with large interaction size, EC optimization method can be more practical.

In summary, our optimization algorithm outperforms Qiskit, t|ket⟩, and Paulihedral compilers in both total gate count and circuit depth, especially for the capability of depth reduction. For all benchmarks, DFS optimization algorithm can reduce the total gate count by 7.8× (up to 23.7×) and circuit depth by 16.5× (up to 64.1×) on average when comparing to other compilers. And EC optimization algorithm shows an average reduction in gate count by 7.8× (up to 23.7×) and in circuit depth by 16.3× (up to 64.1×) for all.

7. Conclusion

In this work, we developed a circuit optimization algorithm for Hamiltonian simulation problems. Distinct from the conventional circuit synthesis, we employ a novel sub-circuit synthesis in Pauli IR and combined with a greedy ordering scheme which explores the search space for an superior Trotter ordering scheme. By searching locally optimal Trotter orders and joining them together, we obtain an advantageous overall order which significantly decreases the gate count and circuit depth from gate cancellation. Moreover, we grafted the approach proposed in [29] onto our gate reduction framework for further optimization, which results in two optimization pathways, namely EC based optimization and DFS based optimization. Evaluation and benchmarking results showed that our algorithm significantly outperforms both state-of-the-art generic quantum compilers t|ket⟩ and Qiskit and specific quantum simulation compiler Paulihedral in terms of total gate count and circuit depth. For a large variety of quantum simulation kernels, DFS and EC optimization algorithm can substantially reduce the overall circuit cost. We believe our techniques will help enhance the performance of quantum simulation on NISQ devices and allow them to solve more practical problems.

Acknowledgments

This work was supported by the National Science Foundation of China (Nos. 61871111 and 61960206005) and the Fundamental Research Funds for the Central Universities (Nos. 2242022k30006 and 2242022k30001).

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

ORCID iDs

Yi Hu https://orcid.org/0000-0002-2270-8099
Xutao Yu https://orcid.org/0000-0001-8625-3241

References

[1] Arute F *et al* 2019 *Nature* **574** 505–10
[2] Arute F *et al* 2020 *Science* **369** 1084–9
[3] Preskill J 2018 *Quantum* **2** 79
[4] Bharti K *et al* 2021 arXiv:2101.08448
[5] Deutsch I H 2020 *PRX Quantum* **1** 020101
[6] Cerezo M *et al* 2020 arXiv:2012.09265
[7] Peruzzo A, McClean J, Shadbolt P, Yung M H, Zhou X Q, Love P J, Aspuru-Guzik A and O'brien J L 2014 *Nat. Commun.* **5** 4213
[8] McClean J R, Romero J, Babbush R and Aspuru-Guzik A 2016 *New J. Phys.* **18** 023023
[9] Bharti K 2020 arXiv:2009.11001
[10] Bharti K and Haug T 2021 *Phys. Rev.* A **104** L050401
[11] Raeisi S, Wiebe N and Sanders B C 2012 *New J. Phys.* **14** 103017
[12] Tranter A, Love P J, Mintert F and Coveney P V 2018 *J. Chem. Theory Comput.* **14** 5617–30
[13] Babbush R, McClean J, Wecker D, Aspuru-Guzik A and Wiebe N 2015 *Phys. Rev.* A **91** 022311
[14] Poulin D, Hastings M B, Wecker D, Wiebe N, Doherty A C and Troyer M 2014 arXiv:1406.4920
[15] Childs A M, Maslov D, Nam Y, Ross N J and Su Y 2018 *Proc. Natl Acad. Sci. USA* **115** 9456–61
[16] Zhang Z-J, Sun J, Yuan X and Yung M-H 2020 arXiv:2011.05283
[17] Tranter A, Love P J, Mintert F, Wiebe N and Coveney P V 2019 *Entropy* **21** 1218

[18] Hastings M B, Wecker D, Bauer B and Troyer M 2014 arXiv:1403.1539
[19] Childs A M and Wiebe N 2012 arXiv:1202.5822
[20] Low G H and Chuang I L 2019 *Quantum* **3** 163
[21] Lau J W Z, Haug T, Kwek L C and Bharti K 2021 arXiv:2103.05500
[22] Berry D W, Childs A M, Cleve R, Kothari R and Somma R D 2015 *Phys. Rev. Lett.* **114** 090502
[23] Berry D W and Childs A M 2009 arXiv:0910.4157
[24] Low G H and Chuang I L 2017 *Phys. Rev. Lett.* **118** 010501
[25] Low G H, Kliuchnikov V and Wiebe N 2019 arXiv:1907.11679
[26] Faehrmann P K, Steudtner M, Kueng R, Kieferova M and Eisert J 2021 arXiv:2101.07808
[27] Suzuki M 1991 *J. Math. Phys.* **32** 400–7
[28] Lao L and Browne D 2021 arXiv:2108.02099
[29] Clinton L, Bausch J and Cubitt T 2021 *Nat. Commun.* **12** 4989
[30] Majumdar R, Madan D, Bhoumik D, Vinayagamurthy D, Raghunathan S and Sur-Kolay S 2021 arXiv:2106.02812
[31] Anis M S *et al* (2021) Qiskit: An Open-Source Framework for Quantum Computing https://doi.org/10.5281/zenodo.2573505
[32] Sivarajah S, Dilkes S, Cowtan A, Simmons W, Edgington A and Duncan R 2020 *Quantum Sci. Technol.* **6** 014003
[33] Li G, Wu A, Shi Y, Javadi-Abhari A, Ding Y and Xie Y 2022 Paulihedral: a generalized block-wise compiler optimization framework for quantum simulation kernels *Proc. 27th ACM Int. Conf. Architectural Support for Programming Languages and Operating Systems* pp 554–69
[34] Sun Q *et al* 2018 *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **8** e1340