

Fermihedral: On the Optimal Compilation for Fermion-to-Qubit Encoding

Yuhao Liu
liuyuhao@seas.upenn.edu
University of Pennsylvania
United States

Shize Che
shizeche@seas.upenn.edu
University of Pennsylvania
United States

Junyu Zhou
junyuzh@sas.upenn.edu
University of Pennsylvania
United States

Yunong Shi
shiyunon@amazon.com
AWS Quantum Technologies
United States

Gushu Li
gushuli@seas.upenn.edu
University of Pennsylvania
United States

Abstract

This paper introduces Fermihedral, a compiler framework focusing on discovering the optimal Fermion-to-qubit encoding for targeted Fermionic Hamiltonians. Fermion-to-qubit encoding is a crucial step in harnessing quantum computing for efficient simulation of Fermionic quantum systems. Utilizing Pauli algebra, Fermihedral redefines complex constraints and objectives of Fermion-to-qubit encoding into a Boolean Satisfiability problem which can then be solved with high-performance solvers. To accommodate larger-scale scenarios, this paper proposed two new strategies that yield approximate optimal solutions mitigating the overhead from the exponentially large number of clauses. Evaluation across diverse Fermionic systems highlights the superiority of Fermihedral, showcasing substantial reductions in implementation costs, gate counts, and circuit depth in the compiled circuits. Real-system experiments on IonQ's device affirm its effectiveness, notably enhancing simulation accuracy.

CCS Concepts: • Computer systems organization → Quantum computing; • Software and its engineering → Formal methods; • Hardware → Emerging languages and compilers.

Keywords: Quantum Computing, Fermion-to-Qubit Encoding, Formal Methods, Boolean Satisfiability

ACM Reference Format:

Yuhao Liu, Shize Che, Junyu Zhou, Yunong Shi, and Gushu Li. 2024. Fermihedral: On the Optimal Compilation for Fermion-to-Qubit

Encoding. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3 (ASPLOS '24), April 27-May 1, 2024, La Jolla, CA, USA*. ACM, New York, NY, USA, 16 pages. <https://doi.org/10.1145/3620666.3651371>

1 Introduction

Simulating Fermionic systems is one crucial application domain of quantum computing. Fermionic systems are composed of Fermions (also known as Fermionic modes), one basic particle type in nature. Notable examples of Fermions include electrons, protons, and neutrons. Many physics models of practical interest are Fermionic systems, such as the molecule electron structure in quantum chemistry [19], the Fermi-Hubbard model [14] in condensed matter physics and material science, the SYK model [31] in quantum field theory. As a fundamentally quantum system, Fermionic systems are hard to simulate on classical computers at a large scale due to their exponential and super-exponential complexity. For example, in 2020, over one million node-hours were allocated to chemistry/material science simulation on the Summit supercomputer [28], and most of these simulations involve Fermionic systems.

Quantum computers are naturally suited to solve such quantum simulation problems. However, encoding a Fermionic system onto a quantum computer requires non-trivial efforts. The reason is that most quantum computers are composed of qubits, which satisfy a different statistical property compared with the Fermions in the Fermionic systems. This difference leads to the fact that Fermionic and qubit systems are usually described in two distinct languages. As shown at the top of Figure 1, the Hamiltonian H_f of a Fermionic system is formulated with an array of creation and annihilation operators $\{a_i^\dagger\}$, $\{a_i\}$ on each Fermionic mode. In the qubit system, however, the Hamiltonian H (in the middle of Figure 1) is formulated with Pauli string operators (e.g., $XYZI$, $ZZZZ$), which will later be compiled into executable quantum circuits.

A particular transformation called the Fermion-to-qubit encoding is naturally introduced to mitigate the gap between the two disparate languages and encode a Fermionic system

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ASPLOS '24, April 27-May 1, 2024, La Jolla, CA, USA

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0386-7/24/04...\$15.00

<https://doi.org/10.1145/3620666.3651371>

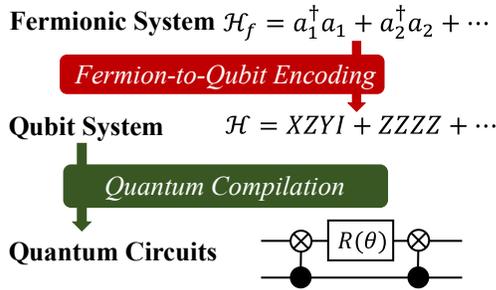


Figure 1. Simulating Fermionic systems with qubit systems

onto a quantum computer. This encoding aims to find a set of Pauli strings representing the creation and annihilation operators. These Pauli strings must satisfy a group of constraints to ensure that the unique statistical property of Fermions is preserved in the qubit system. This encoding is not unique, and different encodings will result in very different execution overhead (e.g., gate counts, circuit depth, etc.) on different Fermionic systems. Overall, it is desirable to have Fermion-to-qubit encodings that can minimize the cost when implementing the quantum circuit to simulate the corresponding Fermionic system.

Finding the optimal Fermion-to-qubit encoding for a targeted Hamiltonian is a highly complicated multi-variable constrained optimization problem. The constraints on a valid encoding, including the anticommutativity constraints, the algebraic independence constraints, the vacuum state preserving property, and the Hamiltonian implementation cost, are represented in linear algebra and natural number theory. To the best of our knowledge, how to unify and formalize these constraints together is unknown. Existing Fermion-to-qubit encodings are mainly theoretically constructed in a Hamiltonian-independent manner [17]. Although some encodings [4, 15, 22] have achieved asymptotically optimal encoding, it is still far from the optimal actual cost because the Hamiltonian of Fermionic systems from various domains can be very different.

In this paper, we overcome this challenge and propose Fermihedral, a compiler framework to find the actual **optimal** Fermion-to-qubit encoding for a targeted Fermionic Hamiltonian. The overview of Fermihedral is shown in Figure 2. **First**, by leveraging the Pauli algebra, we can simplify and convert all the constraints represented in linear algebra and natural number theory into Boolean variables and expressions with carefully designed encoding. Then, the optimal Fermion-to-qubit encoding compilation can be formalized into a Boolean Satisfiability (SAT) problem and solved with existing high-performance SAT solvers. **Second**, we identify the immediate bottleneck in our SAT formulation, the exponentially large number of clauses. We find two causes for this problem and propose corresponding techniques, *ignoring algebraic independence* and *simulated annealing on*

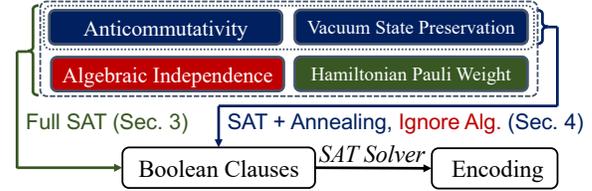


Figure 2. Overview of Fermihedral framework

Hamiltonian-independent optimal encoding. These two techniques can accommodate larger-scale cases by providing approximate optimal solutions with negligible failing probability.

We perform a comprehensive evaluation of Fermihedral on various Fermionic systems. The results show that our SAT-generated optimal Fermion-to-qubit encoding can outperform existing asymptotical optimal encodings [4] and widely adopted encoding [17] with 10% ~ 60% lower Hamiltonian implementation cost, 15% ~ 35% lower gate count and 15% ~ 60% lower circuit depth in the final compiled circuits, as well as more precise Fermionic system simulation results on noisy classical simulators. In particular, we perform real-system experiments showing that our optimal Fermion-to-qubit encoding can significantly increase the simulation accuracy on IonQ’s ion trap device.

The major contributions of this paper can be summarized in the following:

1. We propose Fermihedral, a compilation framework to find the actual *optimal* Fermion-to-qubit encoding for a targeted Fermionic system Hamiltonian.
2. By leveraging the Pauli algebra, we formulate all the required constraints and implementation costs into Boolean variables and expressions so that the encoding can be solved with an SAT solver.
3. We propose two techniques to remove unnecessary clauses in our SAT formulation. This allows us to find approximate optimal solutions with negligible failing probability for larger-size cases.
4. Experimental results show that Fermihedral can outperform current asymptotic optimal encodings on both Hamiltonian-dependent and independent Pauli weight, embodied by better simulation accuracy in noisy simulation and real-system study.

2 Background

This section briefly introduces the essential concepts and their properties to help understand this paper. We start with the Pauli strings, the key components in quantum simulation, followed by the introduction to the Fermionic quantum systems. We do not cover basic quantum computing concepts (e.g., qubit, gate, linear operator, circuit) and we recommend [27] for more details.

2.1 Pauli String

In quantum simulation, Hamiltonians are usually represented by their decomposition into the sum of Pauli strings. A N -length Pauli string for an N -qubit system is defined as the tensor product of Pauli operators: $P = \sigma_N \otimes \sigma_{N-1} \otimes \cdots \otimes \sigma_1$, where $\sigma_i \in \{I, X, Y, Z\}$. Each Pauli operator σ_i operates on the qubit i independently. The X , Y , and Z are three Pauli operators and I is the identity operator:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

2.1.1 Completeness. All the length N Pauli strings formulate an orthonormal basis for all the Hamiltonians of N qubits. Formally, for all N -qubit Hamiltonian \mathcal{H} , there is a unique linear decomposition over all length N Pauli strings:

$$\mathcal{H} = \sum_i w_i P_i, \text{ where } w_i \in \mathbb{R}, P_i \in \{I, X, Y, Z\}^{\otimes N}$$

2.1.2 Pauli String to Circuit. The goal of quantum simulation is to implement the operator $\exp(i\mathcal{H}t)$. It is usually hard to directly implement this many-qubit unitary operator in the circuit, and we need to compile the $\exp(i\mathcal{H}t)$ down to basic single- and two-qubit gates. In practice, this is usually realized by Trotterization [37]. Suppose $\mathcal{H} = \sum_j w_j P_j$ where $w_j \in \mathbb{R}$ and $\{P_j\}$ are Pauli strings. $\exp(i\mathcal{H}t)$ can be approximated by the following trotterization product formula:

$$e^{i\mathcal{H}t} = e^{it \sum_j w_j P_j} = \left(\prod_j e^{i w_j P_j \Delta t} \right)^{t/\Delta t} + O(t\Delta t)$$

Each term $\exp(i w_j P_j \Delta t) = \exp(i \lambda_j P_j)$ ($\lambda_j = w_j \Delta t$) is converted to basic quantum gates.

Figure 3 shows an example of how the Pauli string evolution operator $\exp(i\lambda XZYZ)$ converts to its quantum circuit. It includes the following steps:

- ①. A layer of single-qubit gates is applied to each qubit, corresponding to its Pauli operator. A H gate is applied if the corresponding operator is X (q_3 in the example), and Y is applied if the operator is Y (q_1 in the example).
- ②. A target qubit (q_2 in the example) is selected. CNOT is applied to connect each qubit other than the target qubit whose corresponding Pauli operator is non-identity with the target qubit.
- ③. A $R_z(2\lambda)$ rotation is applied to the target qubit (q_2 in the example).
- ④. Apply the CNOT gates in ② reversely.
- ⑤. Apply the inverse single-qubit gates in ①. In this example, Y^\dagger is applied to q_1 and H to q_3 .

In general, when a qubit's corresponding operator is I in the Pauli string P , then the circuit to implement the simulation of this Pauli string $e^{i\lambda P}$ will not result in any gates applied on that qubit. Only those qubits whose operators are Pauli operators will have gates involved. Roughly, the number of gates in the circuit implementation of $\exp(i\lambda_j P_j)$

is proportional to the number of non-identity Pauli operators in the Pauli string P_j .

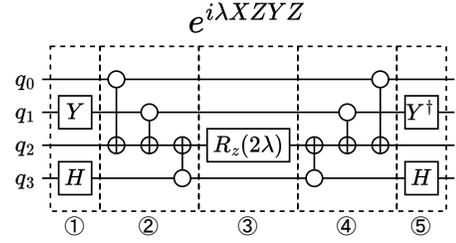


Figure 3. From Pauli string evolution operator $e^{i\lambda P}$ to corresponding circuit

2.1.3 Pauli Weight. The Pauli weight of a Pauli string is defined by the number of non-identity Pauli operators in this string. For example, string $IIXX$ has a Pauli weight of 2. As discussed above, the Pauli weight is roughly proportional to the number of gates in the circuit implementation of $e^{i\lambda P}$ [18, 39]. Therefore, a Hamiltonian \mathcal{H} whose Pauli strings have the minimal sum of Pauli weight will have the minimal gate count when implementing $\exp(i\mathcal{H}t)$ before any follow-up compilation/optimization. Although today's quantum compiler involves many complex transformation and optimization passes, providing a good input circuit with minimal gate count for the downstream compilation/optimizations can, in general, benefit the final compiled circuit.

2.1.4 Arithmetic. Multiplying Pauli strings follows the rule of multiplication over tensor product: $P^1 P^2 = (\sigma_N^1 \sigma_N^2) \otimes (\sigma_{N-1}^1 \sigma_{N-1}^2) \otimes \cdots \otimes (\sigma_1^1 \sigma_1^2)$, which is the tensor product of multiplying their corresponding Pauli operators.

With multiplication, we could define the anticommutator of two Pauli strings as $\{P_1, P_2\} = P_1 P_2 + P_2 P_1$. Two Pauli strings *anticommute* if their anticommutator is 0.

2.2 Fermionic System

The Fermionic quantum system refers to physical systems composed of Fermions. Typical Fermions include protons, electrons, and neutrinos. In the digital quantum simulation of Fermionic systems, the quantum state of Fermions is usually characterized by the occupation of Fermionic modes. Since Fermions satisfy the Pauli exclusion principle, each Fermionic mode can either be unoccupied (denoted by the $|0\rangle_{\mathcal{F}}$) or occupied by at most one Fermion (denoted by the $|1\rangle_{\mathcal{F}}$). Thus, each Fermionic mode has a 2-D state space $\text{span}\{|0\rangle_{\mathcal{F}}, |1\rangle_{\mathcal{F}}\}$. This is similar to a qubit but fundamentally different in statistical properties. For example, exchanging the indices of two Fermionic modes will negate the state vector, but exchanging the indices of two qubits will not. Thus, we cannot directly map a Fermionic mode into a qubit when simulating a Fermionic system on a quantum computer, and a special Fermion-to-qubit encoding is required.

2.2.1 Description of Fermionic Systems. To understand the Fermion-to-qubit encoding, we first introduce the description of Fermionic systems. The basic operators to describe a Fermionic system with N Fermionic modes are the N creation and N annihilation operators: $\{a_i^\dagger\}, \{a_i\}$ where $i = 1 \dots N$. These operators act on states that are described by vectors in the Fock space $\mathcal{F}(\mathbb{C}^N)$ (the state space of N Fermionic modes), a 2^N -D Hilbert space spanned by a set of orthonormal basis (Fock basis):

$$|x_1, x_2, \dots, x_N\rangle_{\mathcal{F}}, \text{ where } x_i = 1 \text{ or } 0$$

Each x_i is the occupation number of the corresponding mode i , also given by the occupation number operator $a_i^\dagger a_i$. The $2N$ creation and annihilation operators act on the basis vectors as:

$$a_i^\dagger |\dots 0_i \dots\rangle_{\mathcal{F}} = |\dots 1_i \dots\rangle_{\mathcal{F}}, a_i |\dots 1_i \dots\rangle_{\mathcal{F}} = |\dots 0_i \dots\rangle_{\mathcal{F}}$$

There exists a vacuum state $|vac\rangle_{\mathcal{F}} = |0, \dots, 0\rangle_{\mathcal{F}}$ such that any annihilation operator applies on it results in 0:

$$\forall j, a_j |vac\rangle_{\mathcal{F}} = 0$$

The creation and annihilation operators in a Fermionic system must satisfy the Fermionic canonical anticommutativity:

$$\begin{aligned} \{a_i, a_j\} &= \{a_i^\dagger, a_j^\dagger\} = 0 \\ \{a_i^\dagger, a_j\} &= \mathbb{I}\delta_{ij} \end{aligned} \quad (1)$$

Here, $\delta_{ij} = 0$ when $i \neq j$ and $\delta_{ij} = \delta_{ii} = 1$ when $i = j$. \mathbb{I} is the identity operator.

The Hamiltonian of a Fermionic System is usually a Hermitian operator expressed by the addition of production of the Fermionic creation and annihilation operators. For example, a 2-Fermionic-mode Hamiltonian can be:

$$\mathcal{H}_{\mathcal{F}} = h_1 a_1^\dagger a_1 + h_2 a_2^\dagger a_2$$

where $h_1, h_2 \in \mathbb{R}$ are parameters.

2.2.2 Encoding Fermionic System in Qubit System. To encode a Fermionic system on a quantum computer composed of qubits, we need to find a set of operators in the qubits state space that also satisfy the Fermionic canonical anticommutativity mentioned above. This is usually achieved by finding Pauli strings for the so-called *Majorana operators*, which can later be converted to the Fermionic creation and annihilation operators.

Majorana operators: The N creation and N annihilation operators could be paired into $2N$ Majorana operators to simplify the problem:

$$\begin{aligned} M_{2j} &= a_j^\dagger + a_j & M_{2j-1} &= i(a_j^\dagger - a_j) \\ \Rightarrow \{M_i, M_j\} &= 2\mathbb{I}\delta_{ij} \end{aligned}$$

Majorana operators are usually set to be Pauli strings. A simple example is the 2 Fermionic-mode system ($N = 2$). With

Jordan-Wigner transformation [17], a widely used Fermion-to-qubit encoding, the 4 Majorana operators are the following 4 Pauli strings:

$$\begin{aligned} M_1 &\mapsto IY & M_2 &\mapsto IX \\ M_3 &\mapsto YZ & M_4 &\mapsto XZ \end{aligned} \quad (2)$$

Correspondingly:

$$\begin{aligned} a_1^\dagger &\mapsto 0.5 \cdot IX - 0.5i \cdot IY & a_1 &\mapsto 0.5 \cdot IX + 0.5i \cdot IY \\ a_2^\dagger &\mapsto 0.5 \cdot XZ - 0.5i \cdot YZ & a_2 &\mapsto 0.5 \cdot XZ + 0.5i \cdot YZ \end{aligned}$$

The anticommutativity could be tested easily, given $\{a_1^\dagger, a_1\}$ as an example:

$$\begin{aligned} \{a_1^\dagger, a_1\} &= \{0.5 \cdot IX - 0.5i \cdot IY, 0.5 \cdot IX + 0.5i \cdot IY\} \\ &= 0.25 \cdot (\{IX, IX\} - i\{IY, IX\} + i\{IX, IY\} + \{IY, IY\}) \\ &= 0.5 \cdot II - 0 + 0 + 0.5 \cdot II = \mathbb{I} \end{aligned}$$

Using this encoding, the Fermionic Hamiltonian example in the last section can be converted to a qubits system Hamiltonian in the following:

$$\begin{aligned} \mathcal{H}_{\mathcal{F}} &= h_1 a_1^\dagger a_1 + h_2 a_2^\dagger a_2 \\ \mapsto \mathcal{H}_{qubit} &= \frac{h_1 + h_2}{2} \cdot II - \frac{h_1}{2} \cdot IZ - \frac{h_2}{2} \cdot ZI \end{aligned}$$

3 Fermion-to-Qubit Encoding via SAT

The Fermion-to-qubit encoding is not unique. Different encodings will result in qubit Hamiltonians with different Pauli weights and circuit implementation overhead. In this section, we introduce Fermihedral to find the optimal Fermion-to-qubit encoding with minimal Pauli weight. We summarize the constraints and optimization objectives of finding a Fermion-to-qubit encoding and then introduce how they can be efficiently formulated into an SAT problem.

3.1 Encoding Constraints and Objectives

As introduced in Section 2.2, the Fermionic creation and annihilation operators on N Fermionic modes can be turned into $2N$ Majorana operators via a simple linear transformation, and the Majorana operators' constraints are much more straightforward. As a result, finding a Fermion-to-qubit encoding usually involves finding the Majorana operators and then pairing them to generate the Fermionic operators.

Constraints: In summary, the $2N$ Majorana operators for an N -Fermion to N -qubit encoding are $2N$ Pauli strings $\{S\}$ satisfying the following four constraints [19, 30]:

- **Anticommutativity:** Any two of the $2N$ Majorana operators must anticommute.

$$\forall P_i, P_j \in \{S\}, \{P_i, P_j\} = P_i P_j + P_j P_i = 2\delta_{ij} \quad (3)$$

- **Linear independence:** All the $2N$ Majorana operators must be linear independent.

$$\sum_{i=1}^{2N} \alpha_i P_i = 0 \implies \alpha_i = 0, 1 \leq i \leq 2N \quad (4)$$

- **Algebraic independence:** All the $2N$ Majorana operators must be algebraically independent. For any two unequal subsets of $\{S\}$, the multiplication of all the Pauli strings in one subset cannot be proportional to the multiplication of all Pauli strings in the other subset.

$$\forall S_a, S_b \subseteq \{S\}, S_a \neq S_b \implies \prod_{P_a \in S_a} P_a \not\propto \prod_{P_b \in S_b} P_b \quad (5)$$

- **Vacuum state preserving:** The vacuum state $|vac\rangle_{\mathcal{F}}$ of Fock basis is represented by qubit state $|0\rangle^{\otimes N}$. This restricts the Majorana operators:

$$\forall 1 \leq j \leq N, \frac{M_{2j} + iM_{2j+1}}{2} |0\rangle^{\otimes N} = 0 \quad (6)$$

This constraint is optional and will not affect the correctness/optimalty of a Fermion-to-qubit encoding.

Since the Pauli strings naturally formulate an orthonormal basis (see Section 2.1.1), satisfying the anticommutativity constraint, which requires all the Pauli strings to be different, already implies that the linear independence constraint is satisfied. Consequently, the linear independence constraint is safely disregarded in the rest of this paper.

Objective: Overall, the optimization objective of a Fermion-to-qubit encoding is to minimize the overhead of simulating this Fermionic system on the quantum computer, that is, the Pauli weight discussed in Section 2.1.3. In this paper, we adopt two types of objectives:

- **Hamiltonian-independent:** We only consider the overhead of implementing the $2N$ Majorana operators. The sum of the Pauli weights of all the $2N$ Majorana operators is minimized. These $2N$ Majorana operators are then used to generate the actual Hamiltonian. This solution may not be optimal for a specific Hamiltonian but can generally demonstrate good performance.
- **Hamiltonian-dependent:** We encode the overhead of implementing the actual Hamiltonian of the target Fermionic system in the SAT optimization. This objective will give us the *optimal* Fermion-to-qubit encoding that can achieve minimal implementation overhead for a specific Hamiltonian.

3.2 Encode Majorana Operators

Majorana operators are Pauli strings, while the SAT problem is formulated with Boolean variables. Our first step is to encode the Pauli operators with Boolean variables and then map the operation of the Pauli operators/strings to Boolean expressions.

Pauli Operator Encoding: A Pauli operator σ has four different possible values $\sigma \in \{X, Y, Z, I\}$ and can be encoded by a pair of two Boolean variables (two bits). We denote the encoding by E :

$$E : \sigma \mapsto \{(0, 0), (0, 1), (1, 0), (1, 1)\}$$

Table 1. Truth table of Pauli operator multiplication $\sigma_3 = \sigma_1\sigma_2$ using the Pauli operator encoding in Equation (7)

$\sigma_3 \backslash \sigma_2$	$I(0, 0)$	$X(0, 1)$	$Y(1, 0)$	$Z(1, 1)$
σ_1				
$I(0, 0)$	$I(0, 0)$	$X(0, 1)$	$Y(1, 0)$	$Z(1, 1)$
$X(0, 1)$	$X(0, 1)$	$I(0, 0)$	$iZ(1, 1)$	$-iY(1, 0)$
$Y(1, 0)$	$Y(1, 0)$	$-iZ(1, 1)$	$I(0, 0)$	$iX(0, 1)$
$Z(1, 1)$	$Z(1, 1)$	$iY(1, 0)$	$-iX(0, 1)$	$I(0, 0)$

A possible encoding strategy E is shown:

$$\begin{aligned} E(I) &= (0, 0) & E(X) &= (0, 1) \\ E(Y) &= (1, 0) & E(Z) &= (1, 1) \end{aligned} \quad (7)$$

Pauli Operator Multiplication Encoding: Given the specific encoding strategy, multiplication between Pauli operators can then be expressed by Boolean expressions. The multiplication of two Pauli operators $\sigma_3 = \sigma_1\sigma_2$ is summarized in Table 1. The additional coefficients produced by the multiplication can be ignored because they do not affect the algebraic independence and the anticommutativity checking. As a result, Table 1 can be considered the truth table when we derive the Boolean expression for the multiplication of the Pauli operator. The Boolean expression for $\sigma_3 = \sigma_1\sigma_2$ is:

$$\begin{aligned} E(\sigma_3).1 &= E(\sigma_1).1 \oplus E(\sigma_2).1 \\ E(\sigma_3).2 &= E(\sigma_1).2 \oplus E(\sigma_2).2 \end{aligned} \quad (8)$$

where $E(\sigma).1$ and $E(\sigma).2$ denote the first and the second bit encoding the Pauli operator σ .

Note that this result indicates that multiplication is symmetric regarding the permutation of X, Y, Z , and I . Thus, any shifting of the encoding strategy would always produce a similar outcome without possessing unique algebraic properties, meaning the encoding scheme we selected in Equation (7) does not lose generality.

Pauli String Encoding: We then extend the Boolean variable encoding to a Pauli string from an individual Pauli operator. A Pauli string of length N , $P = [\sigma_1, \dots, \sigma_N]$, has two equivalent forms of representation used in this paper:

- The operator form:

$$E_{op}(P)_i = E(\sigma_i)$$

- The bit sequence form:

$$E_{bit}(P)_i = \begin{cases} E(\sigma_{(i+1)/2}).1 & i \text{ is odd} \\ E(\sigma_{i/2}).2 & \text{otherwise} \end{cases}$$

Pauli String Multiplication Encoding: The Boolean expression for the multiplication of two Pauli strings can be the combination of the Pauli operator multiplication at each location, following the definition in Section 2.1.4. Suppose $P^1 = [\sigma_1^1, \dots, \sigma_N^1]$ and $P^2 = [\sigma_1^2, \dots, \sigma_N^2]$. Then

$$P^1 P^2 = [\sigma_1^1 \sigma_1^2, \dots, \sigma_N^1 \sigma_N^2], E_{op}(P^1 P^2)_i = E(\sigma_i^1 \sigma_i^2)$$

Table 2. Anticommutativity of Pauli Operators

$\sigma_2 \backslash \sigma_1$	$I(\mathbf{0}, \mathbf{0})$	$X(\mathbf{0}, \mathbf{1})$	$Y(\mathbf{1}, \mathbf{0})$	$Z(\mathbf{1}, \mathbf{1})$
$I(\mathbf{0}, \mathbf{0})$	0	0	0	0
$X(\mathbf{0}, \mathbf{1})$	0	0	1	1
$Y(\mathbf{1}, \mathbf{0})$	0	1	0	1
$Z(\mathbf{1}, \mathbf{1})$	0	1	1	0

3.3 Anticommutativity Constraints

To encode the anticommutativity constraint, we first encode the anticommutativity of Pauli operators and then extend to Pauli strings.

Anticommutativity of Pauli Operators: Suppose we use 0 and 1 to denote that two Pauli operators σ_1 and σ_2 are anticommute or not, respectively, and $\text{acomm}(E(\sigma_1), E(\sigma_2))$ to denote the Boolean expression to determine the anticommutativity of σ_1 and σ_2 . Table 2 shows the truth table of $\text{acomm}(E(\sigma_1), E(\sigma_2))$. Notice that I does not anticommute with any operator. Then the Boolean expression of $\text{acomm}(E(\sigma_1), E(\sigma_2))$ is shown in the following:

$$\begin{aligned} \text{acomm}(E(\sigma_1), E(\sigma_2)) = & \\ & (E(\sigma_1).1 \wedge \neg E(\sigma_2).1 \wedge E(\sigma_2).2) \vee \\ & (E(\sigma_1).2 \wedge E(\sigma_2).1 \wedge \neg E(\sigma_2).2) \vee \quad (9) \\ & (E(\sigma_2).1 \wedge \neg E(\sigma_1).1 \wedge E(\sigma_1).2) \vee \\ & (E(\sigma_2).2 \wedge E(\sigma_1).1 \wedge \neg E(\sigma_1).2) \end{aligned}$$

Anticommutativity of Pauli Strings: Note that for any two Pauli operators, they either commute or anticommute:

$$\sigma_i \sigma_j = (+1) \sigma_j \sigma_i \text{ or } \sigma_i \sigma_j = (-1) \sigma_j \sigma_i$$

Then, in the anticommute check for two Pauli strings, each pair of anticommute Pauli operators will introduce one factor of (-1) in the anticommutativity check of two Pauli strings:

$$\begin{aligned} P^1 P^2 + P^2 P^1 = & [\sigma_1^1 \sigma_1^2, \dots, \sigma_N^1 \sigma_N^2] + [\sigma_1^2 \sigma_1^1, \dots, \sigma_N^2 \sigma_N^1] \\ & + [\sigma_1^1 \sigma_1^2, \dots, \sigma_N^1 \sigma_N^2] + (-1)^a [\sigma_1^1 \sigma_1^2, \dots, \sigma_N^1 \sigma_N^2] \end{aligned}$$

where a is the number of anticommute Pauli operator pairs σ_k^1 and σ_k^2 , $1 \leq k \leq N$. The anticommutativity between two distinct Majorana operators P_i and P_j (whether $P_i P_j + P_j P_i = 0$) is equivalent to if they have an odd number a of qubits whose Pauli operators anticommute. For example, the Pauli operator X and Y anticommute. Then, two Pauli strings XX and YY will not anticommute because they share *two* pairs of anticommute operators X and Y . While the Pauli strings XXX and YYY will anticommute because they share *three* pairs of anticommute Pauli operators. This principle is equivalent to all the $\text{acomm}(E_{op}(P_i)_k, E_{op}(P_j)_k)$ of index $1 \leq k \leq N$ xor to 1:

$$\bigoplus_{k=1}^N \text{acomm}(E_{op}(P_i)_k, E_{op}(P_j)_k) = 1$$

Since the problem requires all possible pairs of $2N$ Majorana operators to be anticommute, the model \mathcal{M} of a valid Fermion-to-qubit encoding should satisfy the conjunction of all anticommutativity of Pauli string pairs:

$$\mathcal{M} \models \bigwedge_{1 \leq i < j \leq 2N} \bigoplus_{k=1}^N \text{acomm}(E_{op}(P_i)_k, E_{op}(P_j)_k)$$

3.4 Algebraic Independence Constraints

We first analyze the algebraic independence condition. Suppose we have two sets of Pauli strings S_a and S_b , $S_a \neq S_b$. If the algebraic independence condition is violated, we will have $\prod_{P_a \in S_a} P_a \propto \prod_{P_b \in S_b} P_b$. We can ignore the coefficients in the Pauli string multiplication, and then the condition becomes equivalence checking of the multiplication results:

$$\prod_{P_a \in S_a} P_a = \prod_{P_b \in S_b} P_b \iff \prod_{P_a \in S_a} P_a \prod_{P_b \in S_b} P_b = I_1 I_2 \dots I_N$$

Two Pauli strings are equal if and only if their multiplication is the all-identity string. Note that if S_a and S_b share some Pauli strings, removing those shared Pauli strings from the two sets will not change this result. Consequently, we only need to consider disjoint sets S_a and S_b where $S_a \cap S_b = \emptyset$. In this case, the condition is turned into:

$$\prod_{P \in S_a \cup S_b} P = I_1 I_2 \dots I_N$$

Here $S_a \cup S_b$ can be an arbitrary subset of the $2N$ Majorana operators set, denoted as S_* .

For certain S_* , the deduced algebraic *dependence* could be tested via the equality in bit sequence form. The left side follows the encoding of operator multiplication, which is the xor of all corresponding bits:

$$E_{bit} \left(\prod_{P \in S_*} P \right) = \bigoplus_{P \in S_*} E_{bit}(P)$$

The bit sequence of I implies that all bits are 0, which is enforced by:

$$\bigwedge_{j=1}^N \neg \left(\bigoplus_{P \in S_*} E_{bit}(P)_j \right) = 1 \quad (10)$$

The encoding model \mathcal{M} should not allow any algebraic *dependence* for $S_* \subseteq S$. Thus, for any S_* in the power set of S (denoted as $\mathcal{P}(S)$), logic not of Equation (10) must be satisfied:

$$\begin{aligned} \mathcal{M} \models & \bigwedge_{S_* \in \mathcal{P}(S)} \neg \bigwedge_{j=1}^N \neg \left(\bigoplus_{P \in S_*} E_{bit}(P)_j \right) \\ = & \bigwedge_{S_* \in \mathcal{P}(S)} \bigvee_{j=1}^N \left(\bigoplus_{P \in S_*} E_{bit}(P)_j \right) \end{aligned}$$

Since S has $2N$ elements, its power set $\mathcal{P}(S)$ has 2^{2N} elements. The constraint of algebraic independence thus incurs a large

overhead by generating 2^{2N} clauses. We will later show how to reduce such complexity in Section 4.1.

3.5 Vacuum State Preservation

Vacuum state preservation requires the encoding maps the vacuum state $|vac\rangle_{\mathcal{F}}$ to $|0\rangle^{\otimes N}$. It sets requirements for the mapped annihilation operators:

$$a_j |vac\rangle_{\mathcal{F}} = 0 \implies \frac{M_{2j} + iM_{2j+1}}{2} |0\rangle^{\otimes N} = 0 \quad (11)$$

A simple case where Equation (11) holds is when there exists at least one index k such that $(M_{2j})_k + i(M_{2j+1})_k = 0$, which indicates that at such index k , the Pauli operators $(M_{2j})_k$ and $(M_{2j+1})_k$, is a pair of X and Y . This property could be imposed by assuming the final solution produced by the SAT solver is correctly paired; that is, $M_k = P_k$. The creation and annihilation operators are:

$$a_j^\dagger = \frac{P_{2j} - iP_{2j-1}}{2}, a_j = \frac{P_{2j} + iP_{2j-1}}{2} \quad (12)$$

The above condition could thus be encoded via Boolean constraints. The existence of XY pair between Pauli operator σ_1 and σ_2 could be tested by the following in our encoding:

$$\text{pair}(E(\sigma_1), E(\sigma_2)) = \neg E(\sigma_1).1 \wedge E(\sigma_1).2 \wedge E(\sigma_2).1 \wedge \neg E(\sigma_2).2$$

A clause is generated for each Majorana operator pair P_{2j} and P_{2j+1} , which should have at least one XY pair across all possible indexes. The final encoding model \mathcal{M} is expected to satisfy each Majorana operator pair P_{2j}, P_{2j+1} for $j = 1$ to N :

$$\mathcal{M} \models \bigwedge_{j=1}^N \bigvee_{k=1}^N \text{pair}(E_{op}(P_{2j})_k, E_{op}(P_{2j+1})_k)$$

3.6 Hamiltonian-Independent Weight Constraint

A widely used Hamiltonian-independent optimization objective is to minimize the sum of the Pauli weights of all the $2N$ Majorana operators. Recall the Pauli weight of a Pauli string is the number of non-identity Pauli operators in the string (Section 2.1.3). A single Pauli operator will contribute to the total Pauli weight if and only if it is not an identity. Using our Pauli operator encoding in Equation (7), the weight of a Pauli operator is:

$$\text{weight}(E(\sigma)) = E(\sigma).1 \vee E(\sigma).2$$

The total weight then accumulates the weight of each single operator, and this is our optimization target:

$$\min_{P_1, \dots, P_{2N}} \sum_{k=1}^{2N} \sum_{i=1}^N \text{weight}(E_{op}(P_k)_i)$$

subject to all the constraints mentioned earlier in this section.

Note that a SAT solver cannot automatically optimize for such a target. In practice, we set a maximum Pauli weight of

Algorithm 1 Solve \mathcal{M} with Optimal Pauli Weight

Require: C : Clauses

Require: \mathcal{M} : Model $\models C$

Ensure: \mathcal{M} : Model $\models C \wedge \min w$

while $\mathcal{M}' \leftarrow \text{solve}(C) == \text{SAT}$ **do**

$\mathcal{M} \leftarrow \mathcal{M}'$

$S = \{P_k\} \leftarrow \text{decode}(\mathcal{M})$

$w \leftarrow \sum_{k=1}^{2N} \sum_{i=1}^N \text{weight}(E_{op}(P_k)_i)$

$C \leftarrow \sum_{k=1}^{2N} \sum_{i=1}^N \text{weight}(E_{op}(P'_k)_i) < w \wedge C$

end while

return \mathcal{M}

w and encode this as a constraint in the model:

$$\sum_{k=1}^{2N} \sum_{i=1}^N \text{weight}(E_{op}(P_k)_i) < w$$

An SAT solver can determine whether a solution exists for a given w . We start from a larger w and gradually reduce w until the SAT solver cannot find a satisfactory solution in a given time. This will give the minimal w where a valid Fermion-to-qubit encoding could exist. The overall procedure is summarized in Algorithm 1.

To obtain the initial feasible solution, we have to start with a total weight w_0 such that it could produce a solution but also close enough to the minimum weight to reduce the solving time. In practice, we start from the Pauli weight of the baseline Bravyi-Kitaev transformation [4].

3.7 Hamiltonian-Dependent Weight Constraint

The Hamiltonian-independent weight constraint can find the set of Majorana operators with minimal Pauli weight. Generally, we can implement the Hamiltonian simulation using these Majorana operators with relatively small overhead. However, this is still not the optimal Pauli weight for a specific Hamiltonian because different Majorana operators are multiplied and added in different ways with real Hamiltonians. For example, a molecular electron Hamiltonian has the following form:

$$H = \sum_{i,j} h_{ij} a_i^\dagger a_j + \frac{1}{2} \sum_{i,j,k,l} h_{ijkl} a_i^\dagger a_j^\dagger a_k a_l \quad (13)$$

The multiplication $a_i^\dagger a_j$ and $a_i^\dagger a_j^\dagger a_k a_l$ makes it possible to cancel out Pauli operators at specific indices. This is not considered in the Hamiltonian-independent weight constraint. To achieve the optimal Pauli weight for a specific Hamiltonian, the Fermion-to-qubit encoding should consider the structure of the targeted Hamiltonian and encode it in the SAT problem.

We first find the Boolean expression for the actual total weight of a Hamiltonian. A Hamiltonian is in the form of the summation of the multiplication of the creation and annihilation operators. The multiplication of several creation

and annihilation operators is denoted as:

$$a_i^{(\dagger)} a_j^{(\dagger)} a_k^{(\dagger)} \dots, \text{ where } i \neq j \neq k \neq \dots$$

Each creation/annihilation operator would be decomposed into two Majorana operators M_{2i} and M_{2i-1} (Equation 12). Thus, the total Pauli weight of such multiplication would be calculated as:

$$\text{weight}(a_i^{(\dagger)} a_j^{(\dagger)} a_k^{(\dagger)} \dots) = \sum_{x=2i, 2i-1; y=2j, 2j-1; z=2k, 2k-1; \dots} \text{weight}(M_x M_y M_z \dots) \quad (14)$$

This Hamiltonian-dependent weight constraint encodes the total Pauli weight of implementing a specific Hamiltonian. We can use this new weight constraint in the SAT formulation to find the optimal encoding for a targeted Hamiltonian. The other constraints and overall structure of Algorithm 1 remain the same.

3.8 CNF Conversion and Solving

Our entire encoding framework only uses Boolean variables with no natural number theory. This allows us to transform the weight constraints into pure Boolean expressions. Along with other conditions, pure SAT solvers could solve the problem entirely, which would typically be dramatically faster than solving the SMT version of the problem. The last step towards a pure SAT problem is to convert clauses into the Conjunctive Normal Form. Directly unfolding *xor*'s would cause an exponential explosion in clause size and be unbearable even on a small scale. Here, we adopt the Tseitin transformation [38] to solve such conversion in linear time of number of clauses by introducing additional Boolean variables polynomial in the size of the original formula. For a Hamiltonian with N Fermionic modes, the number of variables is $O(4N^2)$ and the number of clauses is $O(4^N)$.

4 Scaling up the SAT Method

As discussed above, the number of clauses grows exponentially as the number of Fermionic modes increases, which becomes the immediate bottleneck of our framework. To solve larger-scale Fermion-to-qubit encodings, we identify two optimization opportunities and propose corresponding techniques to reduce the number of clauses.

4.1 Ignoring Algebraic Independence

The first cause of the large number of clauses is the algebraic independence constraints (Section 3.4). To deal with this overhead, we made the following key observations:

The constraint of algebraic independence could be ignored when N is large.

We find that the probability for our SAT-based framework to generate an invalid encoding without considering the algebraic independence constraints is only $\frac{1}{4^N}$, where N is the number of Fermionic modes. To understand this, we define

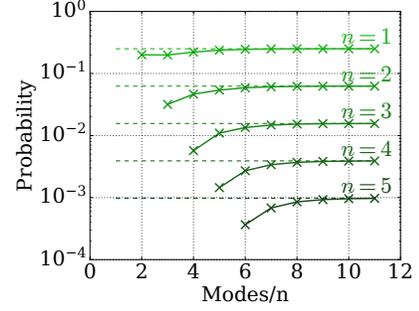


Figure 4. Probability of $n A_k$'s holds simultaneously

solutions found under reduced constraints as *approximate solutions*. Consider the probability that a subset of Majorana operators forms an algebraic dependence. It means all corresponding Pauli operators at each index k multiply to I (coefficient ignored):

$$\forall 1 \leq k \leq N, \prod (M_i)_k = I$$

$$A_k : \prod (M_i)_k = I \quad (15)$$

For a certain k , Equation (15) holds when the quantity of X , Y , and Z are all even or odd, which, the probability is approximately $1/4$ under the assumption that all Pauli operators are distributed uniformly and randomly. Moreover, if the Pauli operator distributions are independent at each index, the probability that all the A_k 's hold simultaneously to break the algebraic independence constraints is $\frac{1}{4^N}$ for N Fermionic modes. This exponentially small failing probability indicates that when N is significant, the $2N$ Majorana operators are most likely algebraic independent, and it would be safe to ignore such constraints to remove their exponentially large number of clauses.

All the discussion above assumes that the Pauli operator distributions are independent at each index. Figure 4 shows numerical evidence for this assumption. For small-scale cases with 1 to 11 Fermionic modes, we select the first 50 optimal Fermion-to-qubit encodings generated from our framework with the algebraic independence constraints applied. We evaluate the probability of $n A_k$'s holds simultaneously over the sampled optimal encodings. The data in Figure 4 perfectly matches our observation that the probability of $n A_k$'s holds simultaneously is $\frac{1}{4^n}$. Therefore, we are confident that the failing probability after removing the algebraic independence constraints is $\frac{1}{4^N}$ for N Fermionic modes.

4.2 Simulated Annealing

We observe that the second cause of the large number of clauses is the Hamiltonian-dependent Pauli weight. Based on Equation (13), second-quantization terms usually grow exponentially with Fermion modes. Our solution to this problem is to remove this part from the SAT formulation. We

Algorithm 2 Solve Hamiltonian Pauli Weight by **Annealing****Require:** \mathcal{H} : N – mode Hamiltonian**Require:** T_0 : initial temperature, T_1 : final temperature**Require:** α : temperature step**Require:** i : iterations**Require:** m : $2N$ Majorana operators**Ensure:** $2N$ Majorana operators $T \leftarrow T_0$ $w = \text{pauli_weight}(m, \mathcal{H})$ **while** $T \geq T_1$ **do** **for** $1 \dots i$ **do** $x, y = \text{randint}(1, N)$ $\text{swap}(x, y, m)$ $w' = \text{pauli_weight}(m, \mathcal{H})$ **if** $\text{random}() \geq e^{-\frac{(w'-w)k}{T}}$ **then** $\text{swap}(x, y, m)$ /* undo the swap */ **end if** **end for** $T \leftarrow T - \alpha$ **end while****return** m

can first find $2N$ Majorana operators with the Hamiltonian-independent Pauli weight, then find the pairing of the Majorana operators to the creation/annihilation operators.

We adopt the simulated annealing algorithm to solve this assignment problem, shown in Algorithm 2. Given an N mode Fermionic system, we could obtain the general Fermion-to-qubit encoding via the original method and try different sequences of Majorana operators to form the corresponding N creation and N annihilation operators. The annealing process takes the Hamiltonian Pauli weight as “energy”, and two Majorana operators are swapped in each iteration. Several classic oracles used in the algorithm are:

- $\text{randint}(i, j)$: return a random integer in $[i, j]$.
- $\text{random}()$: return a random real number in $[0, 1]$.
- $\text{pauli_weight}(m, \mathcal{H})$: calculates the Hamiltonian Pauli weight with given $2N$ strings, where the creation and annihilation operators are paired as Equation (12).
- $\text{swap}(i, j, m)$: swap the i^{th} and j^{th} creation and annihilation operators, that is, swapping the $2i$ with $2j$ string and $2i + 1$ with $2j + 1$ string. This swap does not break the vacuum state preservation property since we are not breaking any existing pairing.

4.3 Complexity Analysis

The SAT problem of finding N -mode Fermion-to-qubit encoding requires $O(N^2)$ Boolean variables, $O(4^N)$ clauses for algebraic independence constraints, $O(N^2)$ clauses for anticommutativity constraints, and $O(N)$ clauses for vacuum state preservation. The number of clauses for Hamiltonian-dependent Pauli weight depends on the specific Hamiltonian.

Table 3. Number of variables and clauses w/ and w/o algebraic independence in the generated SAT instances

Fermionic Modes N	#Vars		#Clauses		Average #Vars/#Clauses	
	w/	w/o	w/	w/o	w/	w/o
2	70	46	459	331	3.65	3.72
3	417	129	2436	1147	3.58	3.72
4	2224	352	10926	3014	3.41	3.98
5	10570	610	46925	5801	3.29	4.03
6	49902	1158	210064	10601	3.23	4.02
7	230503	1687	948732	16608	3.21	4.05
8	1050544	2704	4283375	25693	3.21	4.04
9	N/A	3600	N/A	36037	N/A	4.06
10	N/A	5230	N/A	50798	N/A	4.05
11	N/A	6589	N/A	66593	N/A	4.06
12	N/A	8976	N/A	88440	N/A	4.05
13	N/A	10894	N/A	111129	N/A	4.06
14	N/A	14182	N/A	141504	N/A	4.05
15	N/A	16755	N/A	172132	N/A	4.06
16	N/A	21088	N/A	211938	N/A	4.06
17	N/A	24412	N/A	252025	N/A	4.06
18	N/A	29934	N/A	302793	N/A	4.06

For the benchmark Hamiltonians used later in the Evaluation Section of this paper, the numbers of clauses for Hamiltonian-dependent Pauli weight are $O(N^4)$ for electronic structure problems [19] and SYK model [31], and $O(N^2)$ for the Fermi-Hubbard model [14]. Applying our two optimizations in Section 4 will reduce the number of clauses to $O(N^2)$.

To better understand the effect of eliminating the clauses for algebraic independence, Table 3 shows the intermediate data in the formulated SAT instances, including the number of variables, the number of clauses, and the average number of variables per clause w/ and w/o the algebraic independence constraints when using the Hamiltonian-independent weight constraint. (N/A denotes that generating the corresponding SAT instance takes over 1 hour.) It can be observed that eliminating algebraic independence can significantly simplify the generated SAT instance by reducing the number of variables and clauses.

5 Evaluation

This section evaluates our SAT-based optimal Fermion-to-qubit encoding on various benchmarks with compilation output, noisy simulation, and real system testing.

5.1 Experiment Setup

Benchmark: First, we have evaluated Majorana operator sets of various sizes as the Hamiltonian-independent benchmark. Then, we prepare three representative Hamiltonians from different domains, all of which are widely used in quantum simulation studies: a) molecule electron structure problem from quantum chemistry [19], b) 1-D Fermi-Hubbard model with periodic boundary condition from condensed

$$\begin{aligned}
 H_{\text{electronic}} &= \sum_{i,j} h_{ij} a_i^\dagger a_j + \frac{1}{2} \sum_{i,j,k,l} h_{ijkl} a_i^\dagger a_j^\dagger a_k a_l \\
 H_{\text{Fermi-Hubbard}} &= -t \sum_{\sigma \in \{\uparrow, \downarrow\}} \sum_{i,j} (a_{i\sigma}^\dagger a_{j\sigma} + a_{j\sigma}^\dagger a_{i\sigma}) \\
 &\quad + U \sum_i a_{i\uparrow}^\dagger a_{i\uparrow} a_{i\downarrow}^\dagger a_{i\downarrow} \\
 H_{\text{SYK}} &= \frac{1}{4 \times 4!} \sum_{i,j,k,l=1}^N g_{ijkl} M_i M_j M_k M_l
 \end{aligned}$$

Figure 5. The three types of benchmark Hamiltonians used in this paper. a_σ^\dagger and a_σ are the creation and annihilation operators. M_σ is the Majorana operator.

matter physics [14], and c) four-body SYK model from quantum field theory [31]. Their second quantized [10, 16] Hamiltonians, as introduced in Section 2.2, are listed in Figure 5. We generate the multiplication and summation structure of creation and annihilation operators, transform them into Majorana operators, and fit in the optimal Hamiltonian Pauli weight solver.

Implementation: We implement the proposed SAT-based Fermion-to-qubit encoding and execute our experiments with the following key components:

- **SAT Solver:** We use two solvers: a) the Z3 [8] solver for formulating constraints and applying the Tseitin transformation, and b) the Kissat [2] solver as the standalone high-performance SAT solver. Cadical [1] is a backup solver when Kissat behaves abnormally. The SAT solver runs on CPUs only.
- **Simulation Platform:** We use the Qiskit Aer simulator [29] to execute the noisy simulations. The SAT solver and the noisy simulator run on a server with one AMD EPYC CPU (96 cores, frequency 2.4 GHz) and 768 GB memory.
- **Real System Study:** We perform an actual system study on the IonQ Aria-1 quantum computer, available through Amazon Braket. This device has 25 fully connected ion-trap qubits. It has 99.99% single-qubit gate fidelity, 98.91% two-qubit gate fidelity, and a 98.82% readout fidelity.

Baseline: Our baseline is the Jordan-Wigner (JW) [17] and the Bravyi-Kitaev (BK) transformation [36] implemented by Qiskit [29].

Metrics: We generally use the **Pauli Weight** to indicate the performance of a Fermion-to-qubit encoding. Also, we evaluate the **gate count** as another indicator of the quality of our Fermion-to-qubit encoding compilation. We use the **energy** of simulated states for noisy simulation and real system studies and compare it with theoretical results.

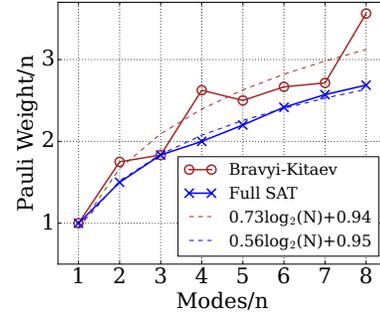


Figure 6. Average Pauli weight per-Majorana operator (small scale)

Configurations: We have three experimental configurations: 1) **Full SAT** is encoding all applicable constraints in SAT (Section 3.7). 2) **SAT w/o Alg.** is to simplify the SAT solving by removing the algebraic constraints (Section 4.1). 3) **SAT + Anl.** is further assigning the Majorana operators with simulated annealing rather than encoding Hamiltonian-dependent weight into the SAT (Section 4.2).

5.2 Hamiltonian-Independent Encoding

In this section, we evaluate the performance of Hamiltonian-independent Fermion-to-qubit encoding. We first compare **Full SAT** with BK at a small scale (up to 8 Fermionic modes). Figure 6 shows the average Pauli weight per Majorana operator from 1 to 8 Fermionic modes. On average, our method shows a 11.16% reduction in per-operator Pauli weight on a small scale. We also plot the regression of the data points.

We then evaluate the larger scale cases (9 to 19 Fermionic modes) where we compare **SAT w/o Alg.** with BK (**Full SAT** requires too many clauses for algebraic independence at this range). The average Pauli weight per Majorana operator is shown in Figure 7. It can be observed that both BK and our method demonstrate a $O(\log N)$ per-Majorana operator Pauli weight, while our method is consistently smaller. The performance of BK varies for different numbers of Fermions, while our method can deliver the optimal performance stably. On average, our method can reduce the Pauli weight by

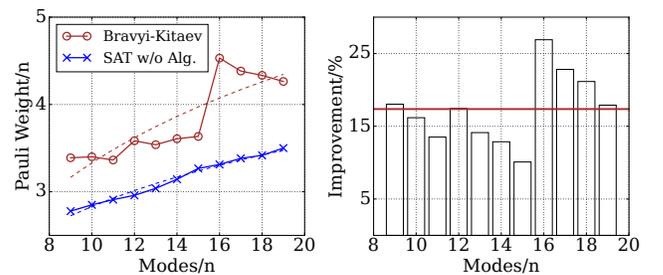


Figure 7. Average Pauli weight per-Majorana operator (larger scale)

Table 4. Hamiltonian-dependent total Pauli weight (small scale), BK vs. **SAT+Anl.** vs. **Full SAT**

Case	Modes (N)	Bravyi-Kitaev	SAT+Anl.	Reduction	Full SAT	Reduction
Electronic Structure	4	934	988	-5.78%	790	15.42%
	6	9004	6878	23.61%	6354	29.43%
Fermi-Hubbard	4	90	104	-15.56%	72	20%
	6	284	202	28.87%	182	35.92%
	8	474	430	9.28%	342	27.85%
Four-Body SYK	3	140	60	57.14%	60	57.14%
	4	496	432	12.90%	312	37.10%
	5	1848	1440	22.08%	896	51.52%
	6	4760	2568	46.05%	2440	48.74%
	7	9876	6148	37.75%	4988	49.50%

17.36% at 9-19 Fermionic modes while our regression coefficient indicates that we can expect around **36.38%** reduction in per-Majorana operator Pauli weight between BK and the optimal Pauli weight as the number of Fermions increases.

5.3 Hamiltonian-Dependent Encoding

We then evaluate the Pauli weight for Hamiltonian-dependent Fermion-to-qubit encoding. We still start from small-size cases where **Full SAT** can be applied. Table 4 shows the total Pauli weight of the three types of Hamiltonian benchmarks after using the BK encoding and our SAT-based encoding. The **Full SAT** can consistently outperform BK with an average 37.26% Pauli weight reduction. The **SAT+Anl.** is sub-optimal and can provide 21.63% Pauli weight reduction on average against BK. Notice that the **SAT+Anl.** is only worse than BK on extremely small cases with merely four Fermionic modes. This does not matter because we can always use **Full SAT** as this scale. For large-size cases, **SAT+Anl.** is constantly better than BK. For those cases of 9 to 19 Fermionic modes, we compare BK with **SAT+Anl.**. The results are shown in Table 5. Compared with BK, **SAT+Anl.** can reduce the total Pauli weight by 23.71% on average (up to 40%).

5.4 End-to-End Real Hamiltonian Simulation

To understand the end-to-end benefit of optimal Fermion-to-qubit encoding, we evaluate executing quantum Hamiltonian simulation programs with noisy simulation and a real quantum computer. We select three benchmarks: the H_2 molecules (4 qubits), the 3×1 (6 qubits), and 2×2 (8 qubits) Fermi-Hubbard Models with periodic boundary conditions. We compare three encodings: Jordan-Wigner (JW [17]), Bravyi-Kitaev (BK [4]), and our **Full SAT**. The initial state is prepared to the energy eigenstate. All the generated Hamiltonians are optimized and compiled into quantum circuits with Paulihedral [18] and Qiskit level 3 optimization [29] with evolution time $t = 1$. For the real system study, we only evaluated H_2 molecule on the IonQ Aria-1 quantum computer due to the limited fidelity of available real quantum computers.

Table 5. Hamiltonian Pauli weight of different problems (continue, **SAT+Anl.** only)

Case	N	BK	SAT+Anl.	Reduction
Electronic Structure	8	24310	22210	8.64%
	10	54156	43618	19.46%
	12	151548	117534	22.44%
Fermi-Hubbard	10	876	682	22.15%
	12	1404	1094	22.08%
	14	1978	1488	24.77%
	16	2626	2216	15.61%
	18	3436	2738	20.31%
Four-Body SYK	8	17376	12848	26.06%
	9	31952	23328	26.99%
	10	55208	32976	40.27%
	11	85436	54924	35.71%

5.4.1 Compilation Output. Table 6 shows the gate count in the final compiled quantum circuits from different Fermion-to-qubit encoding methods, where *Single* refers to the single-qubit gate and *CNOT* is the two-qubit CNOT gate. On average, our **Full SAT** shows around $\sim 20\%$ reduction on single qubit gates and $\sim 35\%$ reduction on CNOT gates compared with BK. The significant reduction will later translate to an increase in simulation accuracy.

5.4.2 Noisy Simulation. We simulated the time evolution of H_2 and both Fermi-Hubbard models from different energy eigenstates under noise, where single-qubit gate fidelity is fixed to 99.99%, and double-qubits gate varies from 99.99% to 99%. We re-run the circuit for multiple shots for each fidelity setting for a more precise calculation of the observed energy and its variance. The number of shots is 3000 for H_2 and 1000 for Fermi-Hubbard under each setting.

Our noisy simulation starts from the energy eigenstates. Energy eigenstates are stationary under time evolution, meaning the system should retain and measure the same energy after a while. However, due to noise in the quantum circuit, the system would inevitably shift to other states for some

Table 6. Gate count of compiled quantum circuits

Case	Gates	BK	Full SAT	Reduction
H_2	Single	26	22	7.69%
	CNOT	26	21	19.23%
	Total	52	43	17.31%
	Depth	39	33	15.38%
3×1 Fermi-Hubbard	Single	58	41	29.31%
	CNOT	56	31	44.64%
	Total	114	72	36.84%
	Depth	76	26	65.79%
2×2 Fermi-Hubbard	Single	56	41	26.79%
	CNOT	53	31	41.51%
	Total	109	72	33.94%
	Depth	73	26	64.38%

small probabilities and cause the observed energy to drift. Lower noise should reduce such drifting and variance in measuring the final energy.

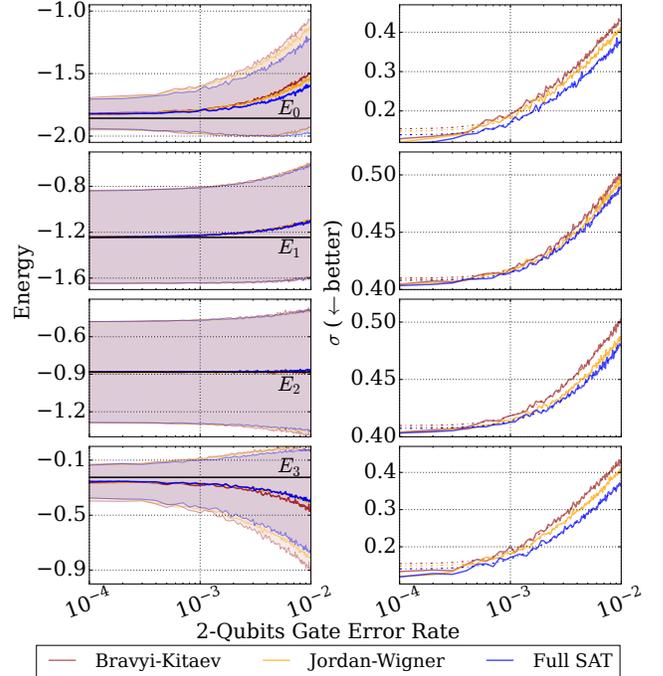
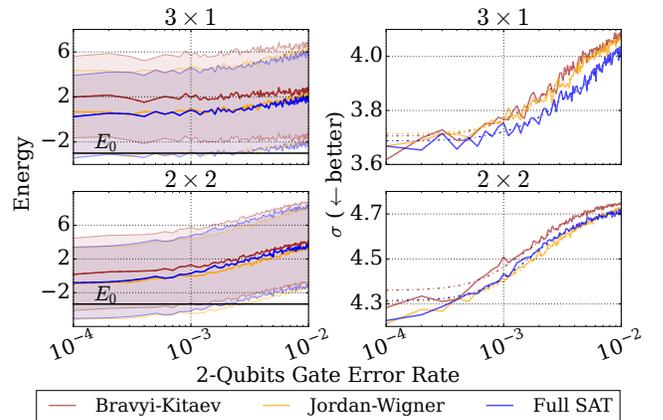
H_2 : Figure 8 shows the simulation result of H_2 molecule. Each row shows the result starting from a different energy eigenstate (E_0 to E_3). The X-axis is the 2-qubit gate error. The Y-axis is the measured energy after the time evolution circuit. The shadows in the left column represent the standard deviation of the measured energy, which is also plotted in the right column. It can be observed that **Full SAT** can outperform BK and JW with the lowest drifting (closer to the energy level) and measuring variance thanks to the fewer gate counts due to lower Pauli weight in the optimal encoding.

Fermi-Hubbard: Figure 9 shows the simulation result of 3×1 and 2×2 square lattice Fermi-Hubbard model with periodic boundary condition. For both models, we start the simulation from the ground energy eigenstate. The representation of the X/Y-axis is the same as H_2 . Similarly, **Full SAT** can demonstrate the lower drifting.

5.4.3 Real System Study. Figure 10 shows the time evolution of H_2 molecule from ground energy E_0 eigenstate on IonQ Aria-1 quantum computer. The Y-axis is the measured energy. The circle size represents the number sampled to a certain value. **Full SAT** can achieve the closest average energy (-1.56 vs. -1.54 vs. -1.49) with the smallest variance compared with BK and JW.

5.5 Time-to-Solution and Scalability Discussion

We also compare the time cost to solve the problem with and without algebraic independence using the SAT solver. Figure 11 (a), (b) shows the time consumption of constructing and solving the SAT problem with/without algebraic independence. Note that we exclude the time for the SAT solver to prove that a Pauli weight lower than optimal is unsatisfiable since it usually triggers a fixed timeout termination. Removing the algebraic independence constraints


Figure 8. Noisy simulation of H_2 , initial state $E_0 \sim E_3$ (black line marks the theoretical energy, shadow marks $\pm 1\sigma$ measurement S.D.)

Figure 9. Noisy simulation of 3×1 and 2×2 Fermi-Hubbard Model, initial state E_0 (black line marks the theoretical energy, shadow marks $\pm 1\sigma$ measurement S.D.)

significantly reduced the time to construct and solve the SAT problem.

Scalability is a major concern of SAT as it is NP-complete, and the time-to-solution, even on small scales, is rather long. However, we only apply a single-thread SAT solver on a small server. Employing a distributed SAT solver on a high-performance cluster could help solve larger-size cases, but confirming such speed-up requires further experiments.

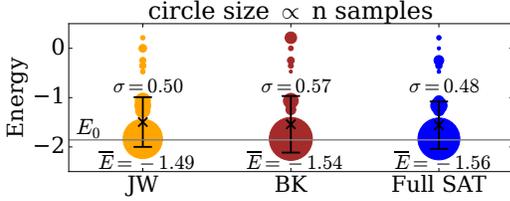


Figure 10. Measured energy of H_2 simulation on IonQ Aria-1, initial state E_0 (gray line)

More importantly, our method can reveal the structure of optimal Fermion-to-qubit encoding for different Hamiltonians and thus guide future work in similar directions, e.g., approximate-optimal encodings with lower complexity, architecture-aware compilations, etc.

6 Related Works

Fermion-to-Qubit Encoding: Previously, Fermion-to-qubit encoding has been studied from a theoretical and constructive perspective. General encoding schemes include Jordan-Wigner transformation [17], Bravyi-Kitaev transformation [4], Parity [3], ternary tree [15, 22], and they have been widely implemented in many quantum software frameworks like Qiskit [29] and OpenFermion [20]. Among them, the Bravyi-Kitaev transformation [4] ([36] compares Bravyi-Kitaev with Jordan-Wigner) and ternary tree [15, 22] have achieved the asymptotical optimal Pauli weight per Majorana operator $O(\log N)$ for N Fermionic modes. But they are still far from the actual optimal solutions. Moreover, the actual Hamiltonian structure is not considered in these approaches. This paper formulates the entire Fermion-to-qubit encoding along with the Hamiltonian-dependent cost into an SAT problem and thus can provide the optimal solution. Recently, the superfast encoding [5, 32] is proposed for a particular type of Fermionic systems with only local Fermionic mode and cyclic structured interactions. This work, however, can process any weakly (Fermi-Hubbard) and strongly (SYK, electronic structure) interacted systems, especially since it considers the Hamiltonian structure in the SAT formulation.

Generic Quantum Compiler Optimization: Today’s quantum compilers (e.g., Qiskit [29], Cirq [9], Q# [21]) usually run multiple passes to optimize a quantum circuit. Typical passes include gate cancellation [26], gate rewrite [33], and routing [23, 24]. These optimization methods (passes) are usually small-scale local circuit transformations. They cannot optimize the Fermion-to-qubit encoding as it will rewrite the entire Hamiltonian simulation circuit.

Compilation for Quantum Simulation: Recently, several works have identified domain-specific compiler optimization opportunities in quantum simulation. These optimizations include the Pauli string ordering [12, 13], architectural-aware synthesis for Pauli string [18, 39], simultaneous diagonalization on commutative Pauli strings [6, 7, 40]. All

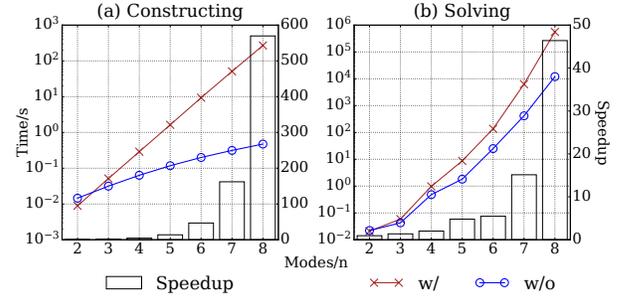


Figure 11. Time to construct and solve **w, w/o Alg.** (Proving unsatisfactory time is excluded)

these works happen at the Pauli string level after finishing the Fermion-to-qubit encoding. This work optimizes the Fermion-to-qubit encoding, which happens before the simulation problem is turned into Pauli strings, and can naturally be combined with all these works.

Formal Methods in Quantum Compilation: Several previous works bring the SAT method to qubit mapping and routing on connectivity-constrained architectures by different constraint encoding, including [11, 23, 34, 35, 41], and [25] (SMT based). These methods focus on different aspects (fidelity, gate count, and circuit depth) and still optimize the gates in the final quantum circuit. To the best of our knowledge, this work is the first to formalize the Fermion-to-qubit encoding finding into an SAT problem.

7 Conclusion

In summary, this paper introduces Fermihedral, a novel compiler framework that addresses the challenge of efficiently simulating complex Fermionic systems on quantum computers. Fermihedral converts the problem of finding optimal Fermion-to-qubit encoding into an SAT problem with carefully designed constraint/objective encoding and solves it with high-performance SAT solvers. Moreover, it offers approximate optimal outcomes for larger-scale scenarios by overcoming the complexities of the exponentially large number of clauses involved in this encoding process. The extensive evaluation across diverse Fermionic systems shows its superiority over existing methods, notably reducing implementation costs and enhancing efficiency in quantum simulations. This work marks a substantial step forward, providing a robust solution for encoding Fermionic systems onto quantum computers and advancing quantum simulation capabilities across scientific domains.

Acknowledgments

We thank the anonymous reviewers for their insightful feedback and our shepherd Yipeng Huang for guidance. This work was in part supported by NSF CAREER Award 2338773 and Amazon Web Service Cloud Credit.

A Artifact Appendix

A.1 Abstract

The artifact contains the code of our Fermihedral framework and other scripts to prepare the environment and reproduce our key results and figures (Figures 4, 6, 7, 8, 9, and 10; Tables 4 and 5). It requires a regular x86-64 Linux server with at least 32GB RAM and 10GB available harddrive. A CUDA GPU is recommended to speed up some of the experiments. Gate counts in Table 5 cannot be automatically reproduced due to break changes and version conflict in Qiskit with Paulihedral. We also provide instructions on how to run the circuit on the IonQ quantum computer device, which requires extra manual work, as well as the entire software dependencies list and generated Hamiltonian model. Since randomization is applied in simulated annealing, the result in Tables 3 and 4 could be deviated.

A.2 Artifact check-list (meta-information)

- **Algorithm:** Fermihedral has following core algorithms:
 - Solving the Fermion-to-qubit encoding problem with SAT (Section 3) is implemented in the file 'fermihedral/majorana.py', DescentSolver and MajoranaModel.
 - Solving Hamiltonian-dependent Pauli weight with **Full SAT** or **SAT+Anl.** method (Section 3.7) is implemented in the file 'fermihedral/majorana.py', HamiltonianSolver.
- **Program:** Noisy simulation subroutines are implemented in the file 'fermihedral/fock.py'.
- **Run-time environment:** Python, Jupyter Notebook
- **Hardware:** Single x86-64 CPU Linux server, preferably with a CUDA GPU, to run the noisy simulation faster.
- **Metrics:** We consider the following metrics and could be directly observed from the output figures:
 - Pauli weight
 - Observed system energy
- **Output:** The output contains the Fermion-to-qubit encoding schema. The output is adapted to the FermionicMapper interface in Qiskit.
- **Experiments:** There are three notebooks for different sets of experiments:
 1. `singleshot.ipynb`: Figures 4, 6, and 7. The time to run this notebook could be extremely long.
 2. `simulation.ipynb`: Figures 8, 9, and 10.
 3. `hamiltonian-weight.ipynb`: Table 4 and 5. The time to run this notebook could be extremely long.
- **Disk space required:** 10GB
- **Time to prepare workflow:** 5 minutes
- **Time to complete experiments:** 1 to 2 weeks
- **Publicly available:** Yes
- **Code licenses:** MIT License
- **Workflow framework used:** Python, Qiskit, Jupyter Notebook
- **Archived:** 10.5281/zenodo.10854557

A.3 Description

A.3.1 How to access. The artifact is available at the following GitHub repository <https://github.com/acasta-yhliu/fermihedral.git> or with DOI <https://doi.org/10.5281/zenodo.10854557>. You can clone the repository or submit issues if there is any problem.

A.3.2 Hardware dependencies. A regular server with a single CPU and preferably a CUDA GPU can run our artifact. RAM is not strictly limited but is recommended to 32GB or above. We also recommend using a powerful CPU to speed up the SAT solver.

A small part of our results (Figure 10) requires execution on the real IonQ quantum computer (available through Amazon Web Service). If you wish to run the quantum circuit simulating H_2 molecule on this real IonQ quantum computer, the cost should be around \$600 (charged by Amazon) with our setup. Consequently, we exclude this part from artifact evaluation, and the exclusion will not affect the major results of this paper.

A.3.3 Software dependencies. The artifact is implemented in Python 3.10. We also require Python packages, including `z3-solver 4.12.2.0`, `Qiskit 1.0.1` accompanied by `Qiskit-nature 0.7.2`, `Qiskit-algorithm 0.3.0`, and `OpenFermion 1.5.1`. Both Qiskit packages require `Numpy 1.25.2`. The list of Python packages and other assistant packages can be found in the `requirements.txt`.

We require a SAT solver to solve the model. To install and compile the `kissat` SAT solver, `make` and a C compiler are required. We used `GCC 11.4.0` in our experiments.

A.4 Installation

You can clone the repository to your local machine and prepare the environment needed by our artifact with the following command (script):

A.4 Installation

You can clone the repository to your local machine and prepare the environment needed by our artifact with the following command (script):

```
$ python3 prepare.py
```

A.5 Experiment workflow

After cloning the repository and preparing the environment in the installation setup, you can execute the Jupyter Notebooks in the virtual environment to reproduce the result (order does not matter).

1. `singleshot.ipynb`: Figures 4, 6, and 7.
2. `simulation.ipynb`: Figures 8, 9, and 10.
3. `hamiltonian-weight.ipynb`: Table 4 and 5.

The results are produced in the exact order of the figures and tables. Please note that the time to execute experiments in `singleshot.ipynb` and `hamiltonian-weight.ipynb` could be extremely long (around weeks).

A.6 Evaluation and expected results

Notebooks should reproduce the exact figures as in our paper. The results are directly printed for Tables 4 and 5.

A.7 Experiment customization

If a CUDA GPU is available, uncomment the following line in the first code cell in `simulation.ipynb` to enable GPU acceleration:

```
# uncomment this line to enable GPU support
# config_device("GPU")
```

Solving the SAT problem at a large scale takes an extremely long time. If the time is a problem, it is possible to limit the problem scale in `singleshot.ipynb` by lowering the number of `MAX_MODES` at the beginning of cells. The timeout number of the SAT solver can also be adjusted to fit the local machine's performance.

For `hamiltonian-weight.ipynb`, you can also adjust the models it solves. `FULL_SAT_MODELS` controls **Full SAT** method and `ANNEALING_MODELS` controls **SAT+AnI.** method. Your model must follow the format:

1. First line marks the model name, modes, and format (ac or mj). Hamiltonians characterized by creation and annihilation operators are marked by ac, while mj indicates Majorana operators characterize the Hamiltonian.
2. Each line is considered as a production of operators. Creation operators are positive numbers, while annihilation operators are negative numbers. Majorana operators are all positive operators.

References

- [1] Armin Biere, Katalin Fazekas, Mathias Fleury, and Maximilian Heisinger. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In Tomas Balyo, Nils Froyleyks, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors, *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, pages 51–53. University of Helsinki, 2020.
- [2] Armin Biere and Mathias Fleury. Gimsatul, IsaSAT and Kissat entering the SAT Competition 2022. In Tomas Balyo, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors, *Proc. of SAT Competition 2022 – Solver and Benchmark Descriptions*, volume B-2022-1 of *Department of Computer Science Series of Publications B*, pages 10–11. University of Helsinki, 2022.
- [3] Sergey Bravyi, Jay M. Gambetta, Antonio Mezzacapo, and Kristan Temme. Tapering off qubits to simulate fermionic hamiltonians, 2017.
- [4] Sergey Bravyi and Alexei Kitaev. Fermionic quantum computation. *Annals of Physics*, 298(1):210–226, May 2002. arXiv:quant-ph/0003137.
- [5] Riley W. Chien, Sha Xue, Tarini S. Hardikar, Kanav Setia, and James D. Whitfield. Analysis of superfast encoding performance for electronic structure simulations. *Physical Review A*, 100(3), sep 2019.
- [6] Alexander Cowtan, Silas Dilkes, Ross Duncan, Will Simmons, and Seyon Sivarajah. Phase gadget synthesis for shallow circuits. *Electronic Proceedings in Theoretical Computer Science*, 318:213–228, may 2020.
- [7] Alexander Cowtan, Will Simmons, and Ross Duncan. A generic compilation strategy for the unitary coupled cluster ansatz, 2020.
- [8] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
- [9] Cirq Developers. Cirq, July 2023.
- [10] P. A. M. Dirac. The quantum theory of the emission and absorption of radiation. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 114(767):243–265, 1927.
- [11] Daniel Große, Robert Wille, Gerhard W. Dueck, and Rolf Drechsler. Exact multiple-control toffoli network synthesis with sat techniques. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(5):703–715, May 2009.
- [12] Kaiwen Gui, Teague Tomesh, Pranav Gokhale, Yunong Shi, Frederic T. Chong, Margaret Martonosi, and Martin Suchara. Term grouping and travelling salesperson for digital quantum simulation, 2021.
- [13] Matthew B. Hastings, Dave Wecker, Bela Bauer, and Matthias Troyer. Improving quantum algorithms for quantum chemistry. *Quantum Info. Comput.*, 15(1–2):1–21, jan 2015.
- [14] J. Hubbard and Brian Hilton Flowers. Electron correlations in narrow energy bands. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 276(1365):238–257, 1963.
- [15] Zhang Jiang, Amir Kalev, Wojciech Mruzekiewicz, and Hartmut Neven. Optimal fermion-to-qubit mapping via ternary trees with applications to reduced quantum states learning. *Quantum*, 4:276, June 2020.
- [16] P. Jordan and E. Wigner. Über das paulische äquivalenzverbot. *Zeitschrift für Physik*, 47(9):631–651, Sep 1928.
- [17] P. Jordan and E. Wigner. Über das Paulische Äquivalenzverbot. *Zeitschrift für Physik*, 47(9):631–651, September 1928.
- [18] Gushu Li, Anbang Wu, Yunong Shi, Ali Javadi-Abhari, Yufei Ding, and Yuan Xie. Paulihedral: A generalized block-wise compiler optimization framework for quantum simulation kernels. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '22*, page 554–569, New York, NY, USA, 2022. Association for Computing Machinery.
- [19] Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C. Benjamin, and Xiao Yuan. Quantum computational chemistry. *Rev. Mod. Phys.*, 92:015003, Mar 2020.
- [20] Jarrod R McClean, Nicholas C Rubin, Kevin J Sung, Ian D Kivlichan, Xavier Bonet-Monroig, Yudong Cao, Chengyu Dai, E Schuyler Fried, Craig Gidney, Brendan Gimby, Pranav Gokhale, Thomas Häner, Tarini Hardikar, Vojtěch Havlíček, Oscar Higgott, Cupjin Huang, Josh Izaac, Zhang Jiang, Xinle Liu, Sam McArdle, Matthew Neeley, Thomas O'Brien, Bryan O'Gorman, Isil Ozfidan, Maxwell D Radin, Jhonathan Romero, Nicolas P D Sawaya, Bruno Senjean, Kanav Setia, Sukin Sim, Damian S Steiger, Mark Staudtner, Qiming Sun, Wei Sun, Daochen Wang, Fang Zhang, and Ryan Babbush. Openfermion: the electronic structure package for quantum computers. *Quantum Science and Technology*, 5(3):034014, jun 2020.
- [21] Microsoft. *Q# Language Specification*, 2020.
- [22] Aaron Miller, Zoltán Zimborás, Stefan Knecht, Sabrina Maniscalco, and Guillermo García-Pérez. The Bonsai algorithm: grow your own fermion-to-qubit mapping, December 2022. arXiv:2212.09731 [quant-ph].
- [23] Abtin Molavi, Amanda Xu, Martin Diges, Lauren Pick, Swamit Tannu, and Aws Albarghouthi. Qubit Mapping and Routing via MaxSAT, August 2022. arXiv:2208.13679 [quant-ph].
- [24] Prakash Murali, Jonathan M. Baker, Ali Javadi Abhari, Frederic T. Chong, and Margaret Martonosi. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers, 2019.
- [25] Prakash Murali, Ali Javadi-Abhari, Frederic T. Chong, and Margaret Martonosi. Formal constraint-based compilation for noisy intermediate-scale quantum systems. *Microprocessors and Microsystems*, 66:102–112, April 2019.
- [26] Yunseong Nam, Neil J. Ross, Yuan Su, Andrew M. Childs, and Dmitri Maslov. Automated optimization of large quantum circuits with continuous parameters. *npj Quantum Information*, 4(1), may 2018.
- [27] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University

- Press, 2010.
- [28] Oak Ridge National Lab. ALCC program awards nearly 6 million summit node hours across 31 projects. <https://www.olcf.ornl.gov/2020/08/05/alcc-program-awards-nearly-6-million-summit-node-hours-across-31-projects/>, 2020. Accessed: 2020-08-16.
- [29] Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2023.
- [30] Venkat K. Raman. Handbook of Computational Quantum Chemistry By David B. Cook. Oxford University Press: New York, 1998. 743 pp. ISBN 0-19-850114-5. \$140.00. *Journal of Chemical Information and Computer Sciences*, 40(3):882–882, May 2000. Publisher: American Chemical Society.
- [31] Subir Sachdev and Jinwu Ye. Gapless spin-fluid ground state in a random quantum heisenberg magnet. *Physical Review Letters*, 70(21):3339–3342, May 1993.
- [32] Kanav Setia, Sergey Bravyi, Antonio Mezzacapo, and James D. Whitfield. Superfast encodings for fermionic quantum simulation. *Phys. Rev. Res.*, 1:033033, Oct 2019.
- [33] Mathias Soeken and Michael Kirkedal Thomsen. White dots do matter: Rewriting reversible logic circuits. In *Proceedings of the 5th International Conference on Reversible Computation*, RC'13, page 196–208, Berlin, Heidelberg, 2013. Springer-Verlag.
- [34] Bochen Tan and Jason Cong. Optimal layout synthesis for quantum computing. In *Proceedings of the 39th International Conference on Computer-Aided Design*, ICCAD '20. ACM, November 2020.
- [35] Bochen Tan and Jason Cong. Optimal qubit mapping with simultaneous gate absorption. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, November 2021.
- [36] Andrew Tranter, Peter J. Love, Florian Mintert, and Peter V. Coveney. A comparison of the bravyi–kitaev and jordan–wigner transformations for the quantum simulation of quantum chemistry. *Journal of Chemical Theory and Computation*, 14(11):5617–5630, 2018. PMID: 30189144.
- [37] H. F. Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10(4):545–551, 1959.
- [38] G. S. Tseitin. *On the Complexity of Derivation in Propositional Calculus*, pages 466–483. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983.
- [39] Arianne Meijer van de Griend and Ross Duncan. Architecture-aware synthesis of phase polynomials for nisq devices, 2020.
- [40] Ewout van den Berg and Kristan Temme. Circuit optimization of hamiltonian simulation by simultaneous diagonalization of pauli clusters. *Quantum*, 4:322, sep 2020.
- [41] Robert Wille, Lukas Burgholzer, and Alwin Zulehner. Mapping quantum circuits to ibm qx architectures using the minimal number of swap and h operations. In *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, New York, NY, USA, 2019. Association for Computing Machinery.