# HATT: Hamiltonian Adaptive Ternary Tree for Optimizing Fermion-to-Qubit Mapping

Yuhao Liu[†], Kevin Yao[†], Jonathan Hong[†], Julien Froustey[‡], Ermal Rrapaj[‖], Costin Iancu[‖], Gushu Li[†], Yunong Shi[§]
University of Pennsylvania[†], University of California, Berkeley[‡], AWS Quantum Technologies[§], Lawrance Berkeley National Lab[‖]
Email: {liuyuhao, gushuli}@seas.upenn.edu

***Abstract*—This paper introduces the Hamiltonian-Adaptive Ternary Tree (HATT) framework to compile optimized Fermion-to-qubit mapping for specific Fermionic Hamiltonians. In the simulation of Fermionic quantum systems, efficient Fermion-to-qubit mapping plays a critical role in transforming the Fermionic system into a qubit system. HATT utilizes ternary tree mapping and a bottom-up construction procedure to generate Hamiltonian aware Fermion-to-qubit mapping to reduce the Pauli weight of the qubit Hamiltonian, resulting in lower quantum simulation circuit overhead. Additionally, our optimizations retain the important vacuum state preservation property in our Fermion-to-qubit mapping and reduce the complexity of our algorithm from $O(N^4)$ to $O(N^3)$. Evaluations on various Fermionic systems demonstrate $5 \sim 25\%$ reduction in Pauli weight, gate count, and circuit depth, alongside excellent scalability to larger systems. Experiments on the Ionq device also show the advantages of HATT in noise resistance in quantum simulations.**

## I. Introduction

Simulating Fermionic systems, composed of particles such as electrons, protons, and neutrons [15], is an essential application area for quantum computing. These systems are integral to various critical physics models, including the electronic structure of molecules in quantum chemistry [29], the Fermi-Hubbard lattice model in condensed matter [1], neutrino evolution from astroparticle physics [3], [7], [35], etc. Being able to simulate large-scale Fermionic systems could directly benefit material science, drug discovery, etc. However, Fermionic systems are intrinsically quantum and hard to simulate on classical computers on a large scale due to their exponential and super-exponential complexities. It can take tens of millions of cores in a supercomputer to perform ab initio electronic structure simulation [18], and it is getting harder to scale larger, limiting the scales scientists could study. Thus, using quantum computers to simulate Fermionic systems provides a crucial solution to the complexity barrier as quantum computers can simulate these systems with linear resource requirements [14].

To simulate a Fermionic system on a quantum computer requires an essential step, the Fermion-to-qubit mapping. This is because most quantum computers are built with qubits with different statistical properties than Fermions. A Fermion-to-qubit mapping bridges this gap by creating the Fermionic statistics in a qubit system. As depicted in Figure 1, a Fermionic Hamiltonian $\mathcal{H}_\mathcal{F}$ (on the left) is usually expressed by the creation and annihilation operators $\{a_i^\dagger, a_i\}$, which

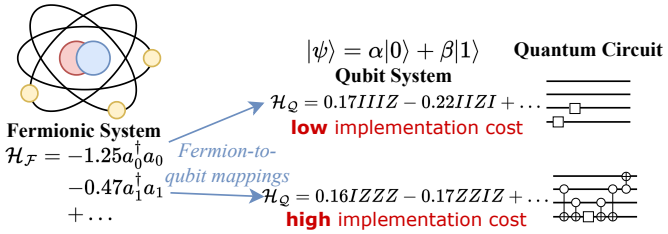



Fig. 1. Overview of simulating Fermionic systems with quantum computers. The mapping on the upper half is better than that in the lower half because it generated a qubit Hamiltonian with fewer Pauli operators, and the circuit implementation cost is lower.

do not naturally arise in a qubit system. A Fermion-to-qubit mapping will find Pauli strings to represent the creation and annihilation operators, converting the Fermionic Hamiltonian into a qubit Hamiltonian $\mathcal{H}_\mathcal{Q}$, which is a weighted sum of Pauli strings as shown in the middle of Figure 1. The qubit Hamiltonian can then be compiled into circuits to simulate the original Fermionic system [26], [43]. Note that Fermion-to-qubit mapping is unique, and different mappings will lead to significantly different circuit implementation costs even for the same input Fermionic system, as shown in Figure 1.

It is naturally desired for people to have Fermion-to-qubit mappings that can minimize the simulation circuit implementation cost. However, existing Fermion-to-qubit mappings are not yet satisfying and have significant further optimization potential. These mappings can be classified into two major types. The first type of approach is the constructive method, such as Jordan-Wigner transformation [22], Bravyi-Kitaev transformation [5], and ternary tree mapping [20], [30]. Although some can generate asymptotically optimal mapping, these methods do not consider the input Hamiltonian and always generate the Fermion-to-qubit mapping using the same operator pattern, leading to suboptimal actual mapping overhead. The second type of approach is the exhaustive search method, such as Fermihedral [27]. Fermihedral formulates the Fermion-to-qubit mapping search into a Boolean satisfiability (SAT) problem and uses an SAT solver to find the optimal mapping with minimal Pauli weight in the final qubit Hamiltonian. Although this approach can find better mappings by encoding the input Hamiltonian in the SAT instance, it has a scalability problem due to the SAT's intrinsic hardness and fails to accommodate large-size practical cases.

In this paper, we aim to push the boundary of Fermion-to-qubit mapping forward and design a new compilation algorithm that can generate lower-cost mapping results while leveraging the input Hamiltonian information in a scalable manner. Formulating a good input-adaptive Fermion-to-qubit mapping is a complicated constrained optimization problem since valid Fermion-to-qubit mappings must satisfy certain constraints and the input Hamiltonions can have different patterns across various Fermionic systems. Our key observation is that existing approaches do not fully exploit the potential of mathematical objects in this domain, including the variants in the data structures and the properties of Pauli algebra. In particular, we notice that the flexibility of the ternary tree data structure can be the key to our new algorithm. Indeed, ternary trees with different structures can yield valid but different Fermion-to-qubit mappings. It is possible to alter the ternary tree structure based on the input Fermionic Hamiltonian to yield low-cost Fermion-to-qubit mappings without exhaustive search.

To this end, we propose the Hamiltonian-Adaptive Ternary Tree (HATT) framework that can compile optimized Fermion-to-qubit mappings tailored to the input Fermionic Hamiltonian. **First**, HATT comes with a new bottom-up ternary tree generation algorithm that can incorporate the input Fermionic Hamiltonian information into the generated tree structure. Starting from the leaf nodes, our algorithm gradually adds parent nodes to construct the tree to maximize the cancellation during Pauli operator multiplication in the input Hamiltonian, thus minimizing the Pauli weight on each qubit. **Second**, we adapt our algorithm with a careful node selection and operator pairing process during tree construction. By traversing down and up the constructed tree, we can ensure that the resulting mapping will map the vacuum state in the Fermionic system to the all-zero state in the qubit mapping. This retains the important *vacuum state preservation* property for a Fermion-to-qubit mapping. **Third**, we further apply a caching strategy to the tree structure during construction to reduce the complexity of traversing up and down during operator pairing from worst $O(N)$ to $O(1)$ when mapping $N$-mode Fermionic systems. Our optimization decreases the total complexity of the algorithm from $O(N^4)$ to $O(N^3)$.

We evaluate HATT extensively against existing construction and exhaustive search methods on various Fermionic system models. The results show that HATT generates close-to-optimal mappings for small-size cases compared with the optimal mapping from Fermihedral [27] and even outperforms the approximately optimal solutions of Fermihedral. For larger-scale cases where Fermihedral fails, HATT outperforms those constructive methods (Jordan-Wigner [22], Bravyi-Kiteav [5], balanced ternary tree [20]) with $\sim 15\%$ reduction in Pauli weight, $5 \sim 20\%$ reduction in circuit depth, and $10 \sim 30\%$ reduction in $CNOT$ gate in the simulation quantum circuit. In addition, HATT achieves the lowest bias and variance in the noisy simulation experiments. On the real system study on IonQ Forte 1 quantum computer, HATT achieves the smallest variance and the second best average result (the best average result comes from Fermihedral's optimal mapping).

The major contributions of this paper can be summarized as follows:

1) We propose HATT, a ternary tree-based Fermion-to-qubit mapping compilation framework to construct input-adaptive ternary tree Fermion-to-qubit mappings based on the Fermionic Hamiltonian in polynomial time.
2) We further improve our tree construction algorithm and successfully implement the *vacuum state preservation*, a desired property for Fermion-to-qubit mapping without increasing the search complexity or affecting the generated mapping performance.
3) By designing a map to cache the structure of the constructed tree, we reduce the overall complexity of our overall algorithm from $O(N^4)$ to $O(N^3)$ ($N$ is system size), achieving significant speed up for large cases.
4) Evaluation on several Fermionic systems shows HATT has much better scalability than exhaustive search and can outperform existing constructive methods on system-dependent performance, including circuit depth, gate count, and higher accuracy in noisy simulation and real-system experiments.

## II. Background

In this section, we introduce the essential background for understanding this paper, including *Fermionic systems*, *Pauli string*, *Fermion-to-qubit mapping*. More fundamental concepts can be found in [34], [37].

### A. Fermionic System

Generally speaking, a Fermionic system contains a set of *Fermionic modes*. Each $i^{th}$ Fermionic mode is captured by a 2D Hilbert space called the *Fock space* $\mathcal{F}_i$, and a pair of creation and annihilation operators $(a_i^\dagger, a_i)$.

*1) Fock Space:* According to the *Pauli exclusion principle* [36], a Fermionic mode has two exclusive states: unoccupied ($|0\rangle_\mathcal{F}$) or occupied by one Fermion ($|1\rangle_\mathcal{F}$). Its *Fock space* $\mathcal{F}$ is the 2D Hilbert space $\mathrm{span}\{|0\rangle_\mathcal{F}, |1\rangle_\mathcal{F}\}$, where $\{|0\rangle_\mathcal{F}, |1\rangle_\mathcal{F}\}$ is called the *Fock basis*.

The Fock space of a $N$-mode Fermionic system is a $2^N$D Hilbert space generated by the tensor products of the Fock space of all $N$ Fermionic modes. The Fock basis is thus the tensor products of each basis:

$$|e_{N-1}, \ldots, e_0\rangle_\mathcal{F} = \bigotimes_{i=0}^{N-1} |e_i\rangle_\mathcal{F} \quad e_i = 0 \text{ or } 1$$

where $|e_i\rangle_\mathcal{F}$ corresponds to the $i^{th}$ Fermionic mode.

*2) Creation and Annihilation Operator:* Each Fermionic mode defines a pair of creation and annihilation operators $(a_i^\dagger, a_i)$. They act by increasing or decreasing the occupation number:

$$a_i^\dagger |0_i\rangle_\mathcal{F} = |1_i\rangle_\mathcal{F} \quad a_i |1_i\rangle_\mathcal{F} = |0_i\rangle_\mathcal{F}$$
$$a_i^\dagger |1_i\rangle_\mathcal{F} = a_i |0_i\rangle_\mathcal{F} = \mathbf{0}$$

Notice that any annihilation operator $a_i$ that applies on the vacuum state $|0\ldots0\rangle_\mathcal{F}$ produces $\mathbf{0}$: $a_i |0\ldots0\rangle_\mathcal{F} \equiv \mathbf{0}$.

*3) Fermionic Hamiltonian:* Finally, the system's Hamiltonian characterizes the time evolution behavior of a Fermionic system. With creation and annihilation operators, we can second quantize [13] the Hamiltonian of a Fermionic System by expressing it with the weighted sum of operator products. A simple 2-mode Fermionic system can have the following Hamiltonian:

$$\mathcal{H}_{\mathcal{F}} = c_0 a_0^\dagger a_0 + c_1 a_1^\dagger a_1 + c_2 a_0^\dagger a_1^\dagger a_0 a_1 \tag{1}$$

### B. Pauli String

*Pauli strings* are the most basic concepts to describe a qubit system, as all $N$-length Pauli strings form an orthogonal basis for $N$-qubit Hermitians [34]. A Pauli string $S$ of a $N$-qubit system is defined as the tensor product of $N$ Pauli operators:

$$S = \sigma_{N-1} \otimes \cdots \otimes \sigma_0 \quad \text{where } \sigma_i \in \{I, X, Y, Z\}$$

We say it has a length $N$. Each Pauli operator $\sigma_i$ operates solely on the qubit $q_i$. $I$ is the identity operator, and $\{X, Y, Z\}$ are the Pauli matrices:

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

*1) Format:* A Pauli string could be written in two forms:

- *$N$-length string*: the string lists operators from $\sigma_{N-1}$ to $\sigma_0$, such as $XYIZ = X \otimes Y \otimes I \otimes Z$
- *Compact form*: identity operators ($I$) are omitted, and each non-identity Pauli operator is subscripted with the qubit it operates, such as $XYIZ = X_3 Y_2 Z_0$. Since operators are annotated, their order does not matter.

*2) Time Evolution Operator:* The ultimate goal of quantum simulation on a digital quantum computer is to execute the time evolution operator $e^{-i\mathcal{H}_{\mathcal{Q}}t}$ of a given qubit Hamiltonian $\mathcal{H}_{\mathcal{Q}}$ with quantum circuits. It begins with decomposing $\mathcal{H}_{\mathcal{Q}}$ to a unique linear combination of $N$-length Pauli strings:

$$\mathcal{H}_{\mathcal{Q}} = \sum c_j S_j \quad c_j \in \mathbb{R}$$

The time evolution operator $e^{-i\mathcal{H}_{\mathcal{Q}}t}$ is then approximated by Trotterization and number of time steps $n$:

$$e^{-i\mathcal{H}_{\mathcal{Q}}t} = e^{-it\sum c_j S_j} \approx \underbrace{\left(\prod_j e^{itc_j S_j/n}\right)^n}_{①} + \underbrace{O(t^2/n)}_{\text{residue}}$$

① easily converts into a sequence of circuit snippets, each implementing a term $\exp(itc_j S_j/n)$ with basic gates. Figure 2 (a)∼(e) shows how term $\exp(itc_j XYIZ/n)$ is converted: (a) Apply $\mathcal{Y} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}$ gates to qubits that have $Y$ operator in the Pauli string ($q_2$), and apply $H$ to the ones have $X$ ($q_3$). This shifts the basis of the corresponding qubit from $Z$ into $Y$ or $X$. (b) Use $CNOT$ gates to entangle all qubits with non-identity Pauli operators ($q_3, q_2$) to a target qubit ($q_0$). (c) Rotate the target qubit ($q_0$) with $R_z(2c_j t/n)$. (d)∼(e) Apply inverse gates in (b) and (a).

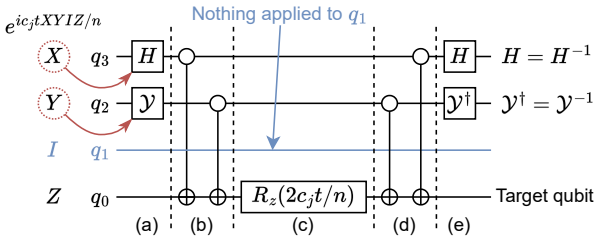



Fig. 2. Circuit snippet of operator $\exp(itc_j XYIZ/n)$. $q_0$ is selected as the target qubit.

*3) Pauli Weight:* The Pauli weight of a Pauli string $w(S)$ is the number of non-identity operators in it. For example, the string $XYIZ$ has weight 3. The Pauli weight of a qubit Hamiltonian $\mathcal{H}_{\mathcal{Q}} = \sum c_i S_i$ is defined as the sum of weights of all decomposed Pauli strings: $w(\mathcal{H}_{\mathcal{Q}}) = \sum w(S_i)$. As discussed in Section II-B2, the identity operators in a Pauli string do not generate any gates. Thus, the Pauli weight of a qubit Hamiltonian is approximately proportional to the number of gates in the circuit of the time evolution operator $e^{-i\mathcal{H}_{\mathcal{Q}}t}$.

### C. Fermion-to-Qubit Mapping

To simulate a Fermionic system on a digital quantum computer, we have to bridge the Fermionic Hamiltonian $\mathcal{H}_{\mathcal{F}}$ to the qubit Hamiltonian $\mathcal{H}_{\mathcal{Q}}$. To do this, we use *Fermion-to-qubit mapping* to find a set of Pauli strings to represent the creation and annihilation operators while satisfying the Fermionic anticommutation relationship. The mapping first pairs the $N$ creation and $N$ annihilation operators into $2N$ Majorana operators $M_j$:

$$a_j^\dagger = \frac{M_{2j} - iM_{2j+1}}{2} \quad a_j = \frac{M_{2j} + iM_{2j+1}}{2} \tag{2}$$

Each Majorana operator $M_j$ maps to a $N$-length Pauli string $S_j$, thus mapping the creation/annihilation operators and the Fermionic Hamiltonian to the qubit Hamiltonian.

For example, using the well-known Jordan-Wigner (JW) Fermion-to-qubit mapping [22], the example Hamiltonian in Equation (1) is mapped to:

$$\mathcal{H}_{\mathcal{F}} \mapsto \mathcal{H}_{\mathcal{Q}} = \frac{2c_0 + 2c_1 - c_2}{4} II + \frac{c_2 - 2c_0}{4} IZ + \frac{c_2 - 2c_1}{4} ZI - \frac{c_2}{4} ZZ$$

with

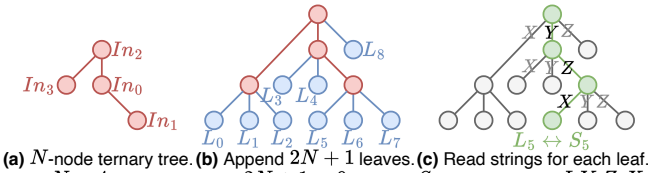
$$a_0^\dagger \mapsto 0.5IX - 0.5iIY \qquad a_0 \mapsto 0.5IX + 0.5iIY$$
$$a_1^\dagger \mapsto 0.5XZ - 0.5iYZ \quad a_1 \mapsto 0.5XZ + 0.5iYZ$$

and the Majorana operators are:

$$M_0 = IX \quad M_1 = IY \quad M_2 = XZ \quad M_3 = YZ$$

## III. Hamiltonian Adaptive Ternary Tree

We aim to design an efficient and scalable compilation framework to generate low-overhead Fermion-to-qubit mappings tailored to the input Hamiltonians. Our approach exploits the properties of Pauli strings and a unique data structure, the




**(a)** $N$-node ternary tree. Here, $N = 4$ **(b)** Append $2N + 1$ leaves. Here, $2N + 1 = 9$ **(c)** Read strings for each leaf. $S_5$ (green path) is $I_3Y_2Z_0X_1$

Fig. 3. Example of a ternary tree and extracting Fermion-to-qubit mapping

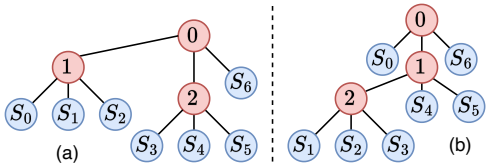



Fig. 4. 3-mode ternary tree Fermion-to-qubit examples. (a) Balanced ternary tree. (b) Unbalanced ternary tree.

ternary tree, to yield Pauli strings satisfying the constraints of a valid Fermion-to-qubit. The Pauli strings are generated to maximize the Pauli operator cancellation during the multiplication of Majorana operators, leading to low Pauli weight post-mapping qubit Hamiltonians.

### A. Extracting Majorana Operators from Ternary Tree

We first introduce the ternary tree and how to extract Majorana operators in the Fermion-to-qubit mapping from a ternary tree.

*1) Ternary Tree:* Ternary trees are a special type of tree. All nodes that are not leaves are denoted as internal nodes. Each internal node of a ternary tree has *at most* three child nodes. A ternary tree is complete if every internal node has exactly three children. If a complete ternary tree has $n$ internal nodes, it has $2n + 1$ leaves. Each path from the root to a leaf is always unique since a ternary tree does not have cycles. For example, the four red nodes in Figure 3 (a) are the internal nodes of a ternary tree, and Figure 3 (b) shows a complete ternary tree after the leaves (blue nodes) are added to ensure each internal node has three child nodes.

*2) Extract Pauli Strings from Ternary Tree:* A ternary tree Fermion-to-qubit mapping for an $N$-mode Fermionic system can be constructed via $2N$ Pauli strings representing the $2N$ Majorana operators. The $2N$ Pauli strings are extracted from a complete ternary tree with $N$ internal nodes and $2N + 1$ leaves. Each Pauli string is represented by a path from the root node to one leaf node in the ternary tree. This process is illustrated using the example in Figure 3.

We denote each internal node by $In_j$ where $j = 0, \ldots, N-1$. The internal node $In_j$ corresponds to the qubit $q_j$ in the qubit system to be mapped onto. The leaf nodes are denoted by $\{L_i\}$. In this example, the ternary tree has four internal nodes $In_0 \sim In_3$ and nine leaves $L_0 \sim L_8$.

Now, we take a path from the root node to one leaf. At each step in this path, there are three possible branches to select the next node because each internal node in a complete ternary tree has three children. The three branches are labeled by $X$, $Y$, and $Z$. Then, a Pauli string can be determined by following the path. Suppose the Pauli string $S_i$ is constructed by the path from the root to the leaf node $L_i$, and the path is:

$$Path = In_a(\text{root}) \to In_b \to In_c \to \cdots \to L_i$$

Each internal node $In_j$ contributes a Pauli operator in the Pauli string $S_i$. For the Pauli operator on qubit $q_j$, If $In_j \notin Path$, the operator is $I_j$. If $In_j \in Path$, based on the branch $In_j \to In_k$ it takes, the node contributes $X_j$, $Y_j$, or $Z_j$ if $In_k$ is the left, middle, or right child of $In_j$. $S_i$ equals the tensor product of all these operators to which the nodes contribute. For example, the green nodes in Figure 3 (c) represent a path from the root node to one leaf in the complete ternary tree. The path starts from node $In_2$ to node $In_0$ via the branch $Y$, so we have $Y_2$ in the Pauli string. Then the path goes from node $In_0$ to node $In_1$ via the branch $Z$, so we have $Z_0$. Similarly, we have $X_1$ because of the $X$ branch from $In_0$ to the leaf, and $I_3$ because $In_3$ is not in this path. This Pauli string is $I_3Y_2X_1Z_0$.

As there are $2N + 1$ leaves, paths from the root to these leaves can construct $2N + 1$ distinct Pauli strings. It can be proved that any $2N$ Pauli strings out of the $2N + 1$ strings satisfy all the constraints required for the $2N$ Majorana operators in a valid Fermion-to-qubit mapping [20], [30]. Thus, we can always construct a Fermion-to-qubit mapping for $N$ Fermionic modes through a complete ternary tree with $N$ internal nodes and $2N + 1$ leaves.

### B. Motivation: Drawback of Balanced Ternary Tree

Since the Pauli strings are determined by the paths from the root to the leaves in a ternary tree, the tree's structure will determine the generated Pauli strings and affect the overhead when simulating the corresponding qubit Hamiltonian. The number of non-identity Pauli operators in one Pauli string is the number of internal nodes in this path. Therefore, to reduce the Pauli weight of the Pauli strings, previous work [20] employs the balanced ternary tree, which has the lowest depth on average. For $N$ Fermonic modes, it generates Pauli strings with an average Pauli weight of $\lceil \log_3(2N+1) \rceil$, which can be proved to be the theoretically optimal Pauli weight per Pauli string.

However, in this paper, we make a key observation that the minimal Pauli weight in the Pauli strings may not lead to the minimal circuit implementation overhead in the qubit Hamiltonian simulation. An actual Hamiltonian can be a complex expression of the Pauli strings. The Pauli operators may cancel with each other and become the identity during the multiplication of Pauli strings. A motivation example is shown in Figure 4.

Suppose we have a toy model 3-mode Fermionic Hamiltonian (in Majorana operators) $\mathcal{H}_\mathcal{F} = c_1 M_0 M_5 + c_2 M_1 M_3$. the mapping generated from a balanced ternary tree, as shown in Figure 4 (a), maps it to $\mathcal{H}_\mathcal{Q} = c_1(X_0X_1)(Y_0Z_2) + c_2(X_0Y_1)(Y_0X_2) = c_1' Z_0X_1Z_2 + c_2' Z_0Y_1X_2$. The Pauli weight is 6. But the unbalanced ternary tree mapping, as shown in Figure 4 (b), can yield a lower Pauli weight. It maps the Hamil-

**Algorithm 1** Hamiltonian Adaptive Ternary Tree Construction

**Require:** $\mathcal{H}_\mathcal{F}$: Hamiltonian of the Fermionic system
**Ensure:** $\{S_i\}$: $2N$ anticommute Pauli strings generated by the ternary tree, used as $2N$ Majorana operators.

1: $\mathcal{H}_\mathcal{Q} \leftarrow \texttt{preprocess}(\mathcal{H}_\mathcal{F})$
2: $\mathcal{U} \leftarrow \{O_0, \ldots, O_{2N}\}$ ▷ *initial node set* $\mathcal{U}$
3: **for** $i$ **from** $0$ **to** $N$ **do**
4: $\quad w \leftarrow \infty$
5: $\quad selection \leftarrow (\text{null}, \text{null}, \text{null})$
6: $\quad$ **for** $O_X, O_Y, O_Z \in \texttt{permutation}(\mathcal{U}, 3)$ **do**
7: $\quad\quad w' \leftarrow \texttt{pauli\_weight}(\mathcal{H}_\mathcal{Q}, (O_X, O_Y, O_Z), i)$ ▷ *Pauli weight on qubit* $i$
8: $\quad\quad$ **if** $w' < w$ **then**
9: $\quad\quad\quad w \leftarrow w'$
10: $\quad\quad\quad selection \leftarrow (O_X, O_Y, O_Z)$
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: $\quad \mathcal{U}.\texttt{remove}(O_X); \mathcal{U}.\texttt{remove}(O_Y); \mathcal{U}.\texttt{remove}(O_Z)$
14: $\quad \mathcal{U}.\texttt{add}(O_{2N+1+i})$
15: $\quad O_{2N+1+i}.(X, Y, Z) \leftarrow (O_X, O_Y, O_Z)$
16: $\quad \mathcal{H}_\mathcal{Q} \leftarrow \texttt{reduce}(\mathcal{H}_\mathcal{Q}, (O_X, O_Y, O_Z) \mapsto O_{2N+1+i})$ ▷ *reduce Hamiltonian* $\mathcal{H}_\mathcal{Q}$
17: **end for**
18: $O_{root} \leftarrow \mathcal{U}.\texttt{pop}()$
19: $\{S_i\} \leftarrow \texttt{walk\_tree}(O_{root})$
20: **return** $\{S_i\}$

tonian to $\mathcal{H}_\mathcal{Q} = c_1(X_0)(Y_0Z_1) + c_2(Y_0X_1X_2)(Y_0X_1Z_2) = c_1''Z_0Z_1 + c_2''Y_2$, which has a Pauli weight of 3.

The example above shows that a mapping generated by unbalanced ternary trees can have larger Pauli weights in the Pauli strings and smaller Pauli weights in the final qubit Hamiltonian because of the multiplication of the Pauli operators. This motivates us to construct the ternary tree adaptively based on the input Fermionic Hamiltonian, which can further decrease the Pauli weight in the final qubit Hamiltonian.

### C. Hamiltonian Adaptive Ternary Tree Construction

This section introduces our ternary tree construction algorithm that can exploit cancellation between Majorana operators for an input Fermionic Hamiltonian. Suppose we have a $N$-mode Fermionic Hamiltonian. Our compilation takes a bottom-up approach, starting with all the $2N+1$ leaves. We gradually append parent nodes to the tree and finally reach the root. These parent nodes are always the internal nodes representing qubits. Our algorithm will search for a tree structure to lower the Pauli weight on each qubit. The pseudo-code is in Algorithm 1, and an algorithm overview is in Figure 5. The algorithm details are explained below.

*1) Setup:* The algorithm compiles a complete ternary tree Fermion-to-qubit mapping for an $N$-mode input Fermionic Hamiltonian with $2N+1$ leaf nodes and $N$ internal nodes. For simplicity, all the nodes are denoted by $O_i$ in the rest of this paper. The $2N+1$ leaf nodes are denoted by $O_0 \sim O_{2N}$. The $N$ internal nodes are denoted by $O_{2N+1} \sim O_{3N}$. Leaf

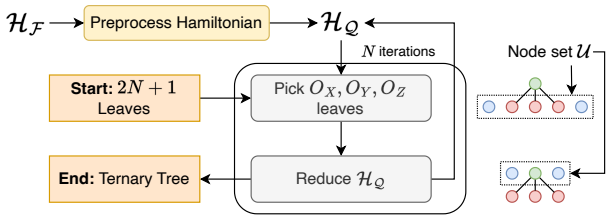



Fig. 5. Algorithm for ternary tree construction

node $O_{0\le \mathbf{i} \le 2N}$ (previously $L_\mathbf{i}$ in the ternary tree example in Figure 3) corresponds to the Pauli string $S_\mathbf{i}$ and is used as the Majorana operator $M_i$ in the Fermion-to-qubit mapping. Internal node $O_{2N+1\le 2N+1+\mathbf{i} \le 3N}$ (previously $In_\mathbf{i}$ in Figure 3) corresponds to qubit $q_\mathbf{i}$. The algorithm starts with $2N+1$ leaves in the ternary tree as the initial node set $\mathcal{U} = \{O_0, \ldots, O_{2N}\}$.

The initial Fermionic Hamiltonian $\mathcal{H}_\mathcal{F}$ is `preprocessed` to substitute all creation and annihilation operators with Majorana operators $\{M_i\}$, then $\mathcal{H}_\mathcal{F}$ is mapped to a qubit Hamiltonian $\mathcal{H}_\mathcal{Q}$ by using Pauli strings to represent Majorana operators: $M_i \rightarrow S_i$. The algorithm finds a concrete Pauli string for each $S_i$. For example, we have a 3-mode Hamiltonian:

$$\begin{aligned}\mathcal{H}_\mathcal{F} &= a_0^\dagger a_0 + 2a_1^\dagger a_2^\dagger a_1 a_2 \\ &= 0.5i \cdot M_0M_1 - 0.5i \cdot M_2M_3 - 0.5i \cdot M_4M_5 \\ &\quad + 0.5 \cdot M_2M_3M_4M_5\end{aligned} \tag{3}$$

The algorithm begins with $\mathcal{U} = \{O_0, \ldots, O_6\}$, each carries $S_0, \ldots, S_6$.The Hamiltonian is preprocessed to $\mathcal{H}_\mathcal{Q} = 0.5i \cdot S_0S_1 - 0.5i \cdot S_2S_3 - 0.5i \cdot S_4S_5 + 0.5 \cdot S_2S_3S_4S_5$.

*2) Iteration:* The algorithm takes $N$ iterations, from 0 to $N-1$. In the $i^{th}$ step, the algorithm picks three nodes from $\mathcal{U}$ and grows a parent $O_{2N+1+i}$ for them. The three nodes are removed from $\mathcal{U}$, and $O_{2N+1+i}$ is added to $\mathcal{U}$, which reduces the size of $\mathcal{U}$ by 2. This settles the operator on qubit $q_i$ for all strings. The three nodes are carefully selected to produce a minimum Hamiltonian Pauli weight on $q_i$. Details are discussed as follows:

- **Node Selection** (line 6 $\sim$ 12 in Algorithm 1): Suppose we select three nodes $O_X$, $O_Y$, $O_Z$ as the $X, Y, Z$ child for node $O_{2N+1+i}$. Based on definition, Pauli strings $S_X, S_Y, S_Z$ correspond to nodes $O_X, O_Y, O_Z$ have $X, Y, Z$ operators on qubit $q_i$ while the rest always have $I$, as shown in Figure 6
  For each possible selection of $O_X, O_Y, O_Z$, we calculate the Pauli weight for the Hamiltonian on qubit $q_i$ (`pauli_weight` in Algorithm 1 line 7). It can be done by only calculating the $i^{th}$ Pauli operator of the Hamiltonian. Based on the Pauli weight and *greedy* strategy, $O_X, O_Y, O_Z$ should give the *minimum Pauli weight* on qubit $q_i$ among all selections.
  For the example Hamiltonian in Equation (3), we will pick $O_0, O_1, O_6$ in the first step. Only $S_0, S_1, S_6$ will

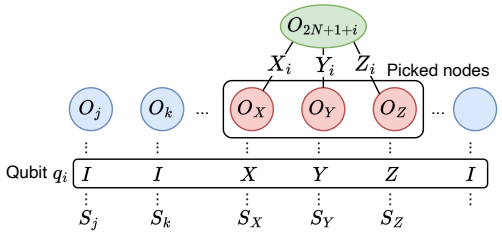



Fig. 6. Picked nodes $O_X, O_Y, O_Z$, their parent $O_{2N+1+i}$ and corresponding Pauli operators on qubit $q_i$

have $X, Y, Z$ operator on qubit $q_0$ while the rest have $I$, as shown in Figure 7 ①. The Pauli weight is:

$$S_0S_1 \mapsto XY = Z(1)$$
$$S_2S_3,\ S_4S_5 \mapsto II = I(0)$$
$$S_2S_3S_4S_5 \mapsto II = I(0)$$
$$\textbf{Total Pauli weight} = 1 + 0 + 0 + 0 = 1$$

- **Update tree and node set** $\mathcal{U}$ (line 13 $\sim$ 15 in Algorithm 1): Remove $O_X, O_Y, O_Z$ from $\mathcal{U}$ and add $O_{2N+1+i}$ to $\mathcal{U}$. Connect $O_X, O_Y, O_Z$ to the $X, Y, Z$ child of node $O_{2N+1+i}$. The node set $\mathcal{U}$ for the example is updated as shown in Figure 7 ②.

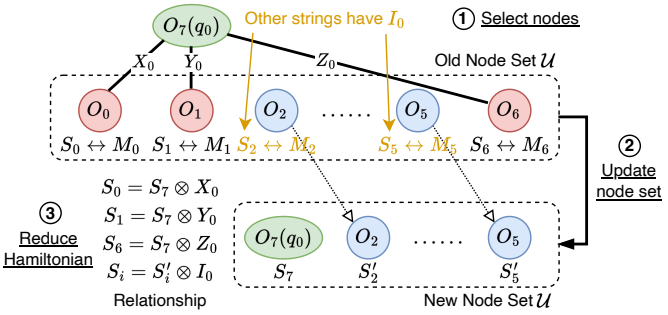



Fig. 7. Example: ① select $O_0, O_1, O_6$ nodes and construct their parent node $O_7(q_0)$. Thus, $S_0, S_1$ and $S_6$ has $X_0, Y_0$ and $Z_0$ in it. All the rest of Pauli strings get $I_0$;② update node set $\mathcal{U}$ by removing $O_0, O_1, O_6$ and inserting $O_7$; ③ reduce Hamiltonian $\mathcal{H}_\mathcal{Q}$ by the relationship: $O_7$ carries $S_7$, which is the *common part* of $S_0, S_1$ and $S_6$, by $\{S_0, S_1, S_6\} = S_7 \otimes \{X, Y, Z\}$. All other nodes carried another Pauli string $S'_i$ such that $S_i = S'_i \otimes I_0$.

- **Reduce Hamiltonian** (`reduce` in Algorithm 1 line 16): The node $O_{2N+1+i}$ carries a Pauli string $S_{2N+1+i}$ such that:

$$S_{\{X,Y,Z\}} = S_{2N+1+i} \otimes \{X, Y, Z\}_i$$

For the rest of the nodes $\forall O_j \notin \{O_X, O_Y, O_Z\}$, we can also find a Pauli string $S'_j$ such that $S_j = S'_j \otimes I_i$. We update $O_j$ to carry $S'_j$ instead of $S_j$.
Then, we substitute $S_j$ with $S'_j$ and $\{S_X, S_Y, S_Z\}$ with $S_{2N+1+i}$ in $\mathcal{H}_\mathcal{Q}$, and remove operators on qubit $q_i$ from $\mathcal{H}_\mathcal{Q}$ since it is settled for all Pauli strings and will not affect total Pauli weight.
In the example, we will introduce node $O_7$ representing qubit $q_0$ that carries $S_7$. The relationships between $S_7$ and $S_0, S_1, S_6$ and between $S'_i$ and $S_i$ are also shown in Figure 7 ③. By substituting using the relationship:

$$S_0 \mapsto S_7 \otimes X_0 \quad S_1 \mapsto S_7 \otimes Y_0 \quad S_6 \mapsto S_7 \otimes Z_0$$
$$S_i \mapsto S'_i \otimes I_0$$

each term of Hamiltonian $\mathcal{H}_\mathcal{Q}$ in Equation (3) reduces to:

$$S_0S_1 \mapsto S_7S_7 = I \otimes Z_0$$
$$S_2S_3 \mapsto S'_2S'_3 \otimes I_0,\ S_4S_5 \mapsto S'_4S'_5 \otimes I_0$$
$$S_2S_3S_4S_5 \mapsto S'_2S'_3S'_4S'_5 \otimes I_0$$

Operator on $q_0$ ($\cdots \otimes \sigma_0$ part) is settled and can be removed since it does not further affect the total Pauli weight. $\mathcal{H}_\mathcal{Q}$ reduces to:

$$\mathcal{H}'_\mathcal{Q} = 0.5i \cdot S'_2S'_3 - 0.5i \cdot S'_4S'_5 + 0.5 \cdot S'_2S'_3S'_4S'_5$$

In the next iteration step, we have node set $\mathcal{U} = \{O_2, O_3, O_4, O_5, O_7\}$ and the reduced Hamiltonian $\mathcal{H}'_\mathcal{Q}$. We can then repeat the above steps with the node set $\mathcal{U}$ and $\mathcal{H}'_\mathcal{Q}$ until all $N = 3$ iterations are performed.

*3) Termination:* The algorithm always terminates after $N$ iterations since the initial size of $\mathcal{U}$ is $2N + 1$ and reduces by two in each step. The final only node in $\mathcal{U}$ is the root of the ternary tree.

*4) Complexity:* We disregard the complexity of calculating the Pauli weight in each iteration step since it is input-dependent. We estimate its overhead to be a constant determined by the input Hamiltonian. In the algorithm, we compared all possible selections of $O_X, O_Y, O_Z$, which are $\binom{N}{3} \sim N^3$ choices. Thus, the complexity of each step is $O(N^3)$. For total $N$ steps, the complexity is $O(N^4)$.

## IV. Vacuum State Preservation and Performance Optimization

The algorithm described in Section III-C decreases the Hamiltonian Pauli weight but fails to retain one desired property of a Fermion-to-qubit mapping: *vacuum state preservation*. In addition, the computation complexity $O(N^4)$ is still high. In this section, we further optimize our algorithm to a) ensure *vacuum state preservation* in the generated mapping and b) reduce its complexity to $O(N^3)$.

### A. Vacuum State Preservation

In Fermion-to-qubit mappings, it is desired to map the vacuum state of a Fermionic system $|0, \ldots, 0\rangle_\mathcal{F}$ to the zero qubit state $|0\rangle^{\otimes N}$ of the qubits. It is called *Vacuum State Preservation*, allowing lower overhead for state preparation in quantum simulation. This property is achieved by ensuring that the annihilation operators always produce $\mathbf{0}$ when applying on the vacuum state:

$$\forall j, a_j |0, \ldots, 0\rangle_\mathcal{F} = \mathbf{0} \Leftrightarrow \frac{M_{2j} + iM_{2j+1}}{2} |0\rangle^{\otimes N} = \mathbf{0}$$

To satisfy the right-hand-side equation, the corresponding Pauli strings $S_{2j}, S_{2j+1}$ of Majorana operators $M_{2j}, M_{2j+1}$ should have a $(X, Y)$ Pauli operator pair on a qubit, since

$((X+iY)/2)\,|0\rangle \equiv \mathbf{0}$, and all the rest operators act the same on $|0\rangle$, i.e., the $k^{th}$ Pauli operator $\sigma_k^{2j}, \sigma_k^{2j+1}$ of $S_{2j}, S_{2j+1}$ satisfies $\sigma_k^{2j}\,|0\rangle = \sigma_k^{2j+1}\,|0\rangle$ [30]. In the vanilla balance ternary tree [20], [30], this is achieved by re-assigning the Pauli strings to Majorana operators.

### *B. Operator Pairing In Tree Construction*

However, we cannot re-assign Pauli strings to Majorana operators in our Hamiltonian-Adaptive ternary tree construction because we have already assumed the Pauli string $S_j$ is assigned to the Majorana operator $M_j$. Reassignment destroys the Pauli operator cancellation created in our algorithm.

Instead of re-assigning Pauli strings, we improve our tree construction algorithm by enforcing *vacuum state preservation* during the node selection. That is, we only freely select $O_X$ and $O_Z$, and $O_Y$ is determined based on $O_X$ and $O_Z$. With careful construction leveraging the property of ternary trees, we can guarantee that all the Majorana operator pairs $(M_{2j}, M_{2j+1})$ have an $(X, Y)$ Pauli operator pair on one qubit, and operators on other qubits act the same on $|0\rangle$, and thus ensure vacuum state preservation.

We first define two new concepts in our new algorithm:

I) *Z-descendant*: The *Z-descendant* of a node $O$, denoted as $desc_Z(O)$, is defined as the *rightmost leaf* of the subtree with root $O$. It can be reached by traversing down all the $Z$ branches. If $O$ is a leaf, then $desc_Z(O) = O$.

II) *Valid Pair*: Two Pauli strings $(S_{2j}, S_{2j+1})$ form a *valid pair* if they share a $(X, Y)$ pair on a qubit and all other operators act the same on $|0\rangle$. If we have $(S_{2j}, S_{2j+1})$ is a *valid pair* forall $0 \leq j < N$, then *vacuum state preservation* is guranteed.

We now introduce our improved tree construction algorithm 2. The overall structure is similar to Algorithm 1, but node selection is changed. In each step $i$, as shown in Algorithm 2 instead of selecting all $O_X, O_Y, O_Z$ nodes, we now only select nodes $O_X, O_Z$ as the $X, Z$ children of node $O_{2N+1+i}$ and find an appropriate $O_Y$ to ensure exists $l$, such that $desc_Z(O_X)$ corresponds to $S_{2l}$ and $desc_Z(O_Y)$ corresponds to $S_{2l+1}$. Thus, $(S_{2l}, S_{2l+1})$ is valid paired since they have a $(X, Y)$ pair on qubit $q_i$, and for all $k \neq i$, the $k^{th}$ operators for $S_{2l}, S_{2l+1}$ are either the same or a $Z, I$ pair, whereas $Z\,|0\rangle \equiv I\,|0\rangle$, as shown in Figure 8.

Suppose $O_X$ and $O_Z$ are selected. We traverse down along the $Z$ child from node $O_X$ until reaching a leaf of the final leaf $O_x = desc_Z(O_X)$ (where $0 \leq x < 2N+1$), as shown in Figure 8 ①. Then, we select its nearby leaf $O_y$ as $desc_Z(O_Y)$ based on $O_x$ to ensure the corresponding Pauli strings $S_x, S_y$ form a valid pair $(S_{x=2l}, S_{y=2l+1})$, as shown in Figure 8 ②:

- If $x = 2N$, then $O_x$ is the rightmost leaf. Discard this selection and pick $O_X, O_Z$ again ($S_{2N}$ is always discarded in the ternary tree mapping and does not pair).
- If $x$ is *even* ($x = 2l$), let $y = x + 1$ ($y = 2l + 1$).
- If $x$ is *odd* ($x = 2l + 1$), let $y = x - 1$ ($y = 2l$). We must also exchange $O_X$ with $O_Y$ when we get $O_Y$ (Line 15, Algorithm 2) to ensure $(S_x, S_y)$ has a $(X, Y)$ Pauli operator pair, instead of $(Y, X)$.

**Algorithm 2** Optimized HATT

**Require:** $\mathcal{H}_\mathcal{F}$: Hamiltonian of the Fermionic system
**Ensure:** $\{S_i\}$: $2N$ Majorana operators generated by the ternary tree

```
1: ...                                          ▷ Algorithm 1, line 1 ∼ 2
2: for i from 0 to N do
3:     ...                                      ▷ Algorithm 1, line 4 ∼ 5
4:     for O_X, O_Z ∈ permutation(U, 2) do
5:         O_x ← desc_z(O_X)
6:         if x = 2N then
7:             continue ▷ O_x is the last operator, no pairing
8:         end if
9:         if x is even then
10:            O_y ← O_{x+1}
11:            O_Y ← traverse_up(O_y, U)
12:        else
13:            O_y ← O_{x−1}
14:            O_Y ← traverse_up(O_y, U)
15:            swap(O_X, O_Y)                   ▷ swap Y, X to X, Y
16:        end if
17:        ...                                  ▷ Algorithm 1, line 7 ∼ 11
18:    end for
19:    ...                                      ▷ Algorithm 1, line 13 ∼ 16
20: end for
21: ...                                         ▷ Algorithm 1, line 18 ∼ 20
```

```
1: procedure desc_z(O)
2:     while O is not leaf do
3:         O ← O.Z
4:     end while
5:     return O
6: end procedure
```

```
1: procedure traverse_up(O, U)
2:     while O ∉ U do
3:         O ← O.parent
4:     end while
5:     return O
6: end procedure
```

To find $O_Y$ based on $O_y = desc_Z(O_Y)$, we can traverse up from $O_y$ until we reach a node $O_Y$ in the current node set $\mathcal{U}$, as shown in Figure 8 ③. $O_Y$ is the ancestor of $O_y$, as it is the root of the subtree in which $O_y$ is located.

Finally, $O_Y$ is chosen as the $Y$ child of $O_{2N+1+i}$ to ensure valid string pairing. For each selection of $O_X, O_Z$ in step $i$, we calculate the Pauli weight similarly to the original algorithm and construct the node based on the selection of $O_X, O_Y, O_Z$ that gives a minimum Pauli weight on qubit $q_i$, then reduce the Hamiltonian similarly to the original algorithm.

Consider our previous example, where $\mathcal{H}_\mathcal{F} = a_0^\dagger a_0 + 2a_1^\dagger a_2^\dagger a_1 a_2 = 0.5i \cdot M_0 M_1 - 0.5i \cdot M_2 M_3 - 0.5i \cdot M_4 M_5 + 0.5 \cdot M_2 M_3 M_4 M_5$. We have already selected $O_0, O_1, O_6$ as the children of $O_7$ in the first step, and the Hamiltonian is

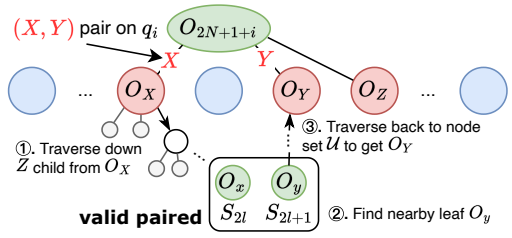



Fig. 8. Procedure of finding $O_Y$ based on $O_X, O_Z$

reduced to $\mathcal{H}_{\mathcal{Q}} = 0.5iS_2'S_3' - 0.5iS_4'S_5' + 0.5S_2'S_3'S_4'S_5'$.

In the second step, we first choose $O_7, O_2$ as $O_X, O_Z$, as shown in Figure 9 (a). However, $O_x = desc_Z(O_7) = O_6$ is the rightmost operator, so we discard this selection and move on. Then, we choose $O_2, O_7$ as $O_X, O_Z$, as shown in Figure 9 (b). Here, $O_x = desc_Z(O_2) = O_2$, thus $O_y = O_{2+1} = O_3$ and traverse back to $S$ gives $O_Y = O_3$. This choice also minimizes the Pauli weight. $O_2, O_3, O_7$ are the children of $O_8$.

We then follow the original procedure and reduce the Hamiltonian:

$$S_2'S_3' \mapsto XY \to Z(1),\ S_4'S_5' \mapsto II \to I(0)$$
$$S_2'S_3'S_4'S_5' \mapsto XYII \to Z(1)$$
$$\textbf{Total Pauli weight} = 1 + 0 + 1 = 2$$

We can check that Majorana operators are paired validally. $(M_2, M_3) \mapsto (S_2, S_3)$ have a $(X, Y)$ pair on qubit 1, and $(M_0, M_1) \mapsto (S_0, S_1)$ have a $(X, Y)$ pair on qubit $q_0$.

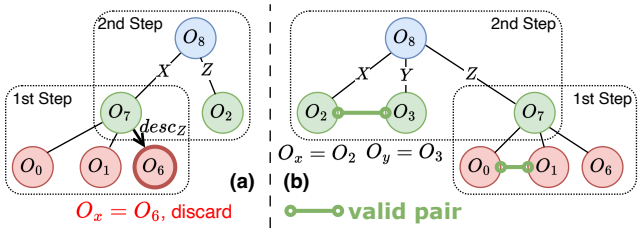



Fig. 9. An example of operator pairing in tree construction. In (a), we select $O_7, O_2$ as $O_X, O_Z$, but $O_x = desc_Z(O_7) = O_6$ is the rightmost leaf, thus we discard this selection. In (b), we select $O_2, O_7$ as $O_X, O_Z$ and found $O_3$ as $O_Y$. Green lines show the valid pair formed by Pauli strings.

### C. Optimizing Operator Pairing

In the algorithm above, the complexity at the node selection is reduced because we only select two operators $O_X, O_Z$ in each step. The computation complexity is $\binom{N}{2} \sim O(N^2)$. However, it introduces new overhead traversing to $O_x$ and up to $O_Y$. In the worst case, the complexity of traversing up and down can be $O(i)$, leading to a total complexity of $O(N^4)$ (there are $N$ steps).

In this section, we further optimize the algorithm by reducing the traversing time from $O(N)$ to $O(1)$. This improvement is made based on the following *statement*:

> In step $i$, the $O_y$ we find and its ancestor $O_Y \in \mathcal{U}$ must always satisfy $O_y = desc_Z(O_Y)$

*Proof*: We first prove two lemmas.

> **Lemma 1**: Before step $i$, a leaf $O$ is not paired with another leaf if and only if it is a Z descendant of a node in $\mathcal{U}$. In other words, $\exists O' \in \mathcal{U}, O = desc_Z(O')$.
>
> *Proof*: We prove it by induction.
>
> (I) Before step $i = 0$, $\mathcal{U}$ includes all leaves. They are all unpaired and Z descendants of themselves.
>
> (II) Suppose Lemma 1 is satisfied before step $i$. We prove it holds after step $i$ (holds before step $i+1$). After step $i$, we find $O_X$, $O_Y$, and $O_Z$ as the $X, Y$, and $Z$ children of node $O_{2N+1+i}$. Based on the induction hypothesis, all unpaired leaves in the $O_X, O_Y$, and $O_Z$ subtree are $O_x = desc_Z(O_X), O_y = desc_Z(O_Y)$, and $O_z = desc_Z(O_Z)$. Our algorithm pairs $O_x$ with $O_y$ while leaving $O_z$ unpaired. Since $O_Z$ is the $Z$ child of $O_{2N+1+i}$, we have $O_z = desc_Z(O_Z) = desc_Z(O_{2N+1+i} \in \mathcal{U})$ and it is the only unpaired leaf in the $O_{2N+1+i}$ subtree.
> Now, we can check the lemma. $O_x$ and $O_y$ are paired and no longer Z descendants since their path to $O_{2N+1+i}$ takes $X$ or $Y$ branch. $O_z$ is unpaired and still Z descendant of $O_{2N+1+i} \in S$. The lemma holds on $O_x, O_y$, and $O_z$. All other leaves are untouched, and the lemma holds on them. □

> **Lemma 2**: $O_y$ is not paired in step $i$.
>
> *Proof*: Notice that $\forall 0 \le l < N$, leaf $O_{2l}$ must be paired with $O_{2l+1}$. $O_x$ is not paired in step $i$ due to Lemma 1. Then, based on how we find $O_y$ in Section IV-B, $O_y$ must be unpaired. Otherwise, it leads to a contradiction. □

Based on Lemma 1 and $O_x = desc_Z(O_X)$, we have $O_x$ must be unpaired. Then, based on Lemma 2, $O_y$ is also unpaired. Again, by Lemma 1, $O_y$ is a Z descendant, indicating $O_y = desc_Z(O_Y)$. □

The observation hints that the traversing up and down procedure (`descz` and `traverse_up` in Algorithm 2) only involves the $Z$ descendants and their ancestors. Thus, we could keep two maps: $O \mapsto desc_Z(O)$ and $desc_Z(O) \mapsto O$ to reduce the complexity of traversing from $O(N)$ to $O(1)$. The map updates when constructing a new parent in each step, as shown in Algorithm 3. Overall, this optimization reduced the total complexity from $O(N^4)$ to $O(N^3)$.

## V. Evaluation

In this section, we evaluate the proposed Hamiltonian-Adaptive Fermion-to-qubit mapping compilation framework HATT against existing Fermion-to-qubit mappings. We also test the scalability and performance of HATT and its optimization.

### A. Benchmark Physics Models

We select the following physical models as our benchmark Fermionic Hamiltonians. They come from various application

**Algorithm 3** Optimized Algorithm 2 with $O \mapsto desc_Z(O)$ and $desc_Z(O) \mapsto O$ maps

**Require:** $m_{down}$: $O \mapsto desc_Z(O)$ map
$m_{up}$: $desc_Z(O) \mapsto O$ map

**for** $i$ **from** $0$ **to** $2N$ **do** ▷ *initialize maps*
  $m_{down}[O_i] \leftarrow O_i$
  $m_{up}[O_i] \leftarrow O_i$
**end for**
$\cdots$ ▷ *Algorithm 1, line* $1 \sim 2$
**for** $i$ **from** $0$ **to** $N$ **do**
  $\cdots$ ▷ *Algorithm 2, line* $3 \sim 19$
  $O_{2N+1+i}.(X,Y,Z) \leftarrow (O_X, O_Y, O_Z)$
  $Zdesc \leftarrow m_{down}[O_Z]$
  $m_{down}[O_{2N+1+i}] \leftarrow Zdesc$
  $m_{up}[Zdesc] \leftarrow O_{2N+1+i}$
  $\cdots$ ▷ *Algorithm 1, line* $16$
**end for**
$\cdots$ ▷ *same as Algorithm 1, line* $18 \sim 20$

**procedure** $\texttt{desc}_\texttt{Z}(O)$ ▷ $O(1)$ *time*
  **return** $m_{down}[O]$
**end procedure**

**procedure** $\texttt{traverse\_up}(O, \mathcal{U})$ ▷ $O(1)$ *time*
  **return** $m_{up}[O]$
**end procedure**

domains of quantum simulation and have different Fermionic mode coupling structures.

1) **Electronic structure model** from quantum chmistry [29]. Hamiltonian describes the electrons in a molecule:

$$\mathcal{H}_{\text{e}} = \sum_{p,q} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{p,q,r,s} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s$$

Geometric data of molecules are from PubChem [23] to determine the coefficients using PySCF [42].

2) **Fermi-Hubbard model** in condensed-matter physics [1]. Hamiltonian describes a position lattice model of Fermions:

$$\mathcal{H}_{\text{fh}} = \sum_{i,j} \sum_{\sigma=\uparrow,\downarrow} t_{ij} a_{i,\sigma}^\dagger a_{j,\sigma} + U \sum_i a_{i,\uparrow}^\dagger a_{i,\uparrow} a_{i,\downarrow}^\dagger a_{i,\downarrow}$$

3) **Collective neutrino oscillations** from astroparticle physics [3], [7], [35]. The Hamiltonian is formulated on a 1D momentum lattice:

$$H_\nu = \sum_{i=1}^{N} \sum_{a}^{3} \sqrt{p_i^2 + m_a^2}\, \hat{a}_{a,i}^\dagger \hat{a}_{a,i} + \sum_{i_1,i_2,i_3}^{N} \sum_{a,b}^{3} C_{i_1,i_2,i_3} \hat{a}_{a,i_1}^\dagger \hat{a}_{a,i_3} \hat{a}_{b,i_2}^\dagger \hat{a}_{b,4}$$

where $p_{*,*}$ and $m_a$ are the momentum and static mass of neutrino and $C_{i_1,i_2,i_3} = \mu \left[\hat{p}_{i_2,x} - \hat{p}_{i_1,x}\right] \left[\hat{p}_{4,x} - \hat{p}_{i_3,x}\right]$ ($\mu$ is the two-body coupling strength).

### B. Experiment Setup

*1) Implementation:* We implemented our Hamiltonian-Adaptive Ternary Tree method (HATT) as described in Section III and IV using Python. We leveraged some Pauli operator processing modules from Qiskit [19] and Qiskit Nature [12].

*2) Baseline Fermion-to-Qubit Mappings:* We compared against a) the Jordan-Wigner transformation (JW) [22], and b) the Bravyi-Kitaev transformation (BK) [5] in Qiskit Nature [12], c) the balanced ternary tree mapping [30] (BTT), and Fermihedral [27] (FH) that gives optimal Hamiltonian Pauli weight at small scales using a SAT solver.

*3) Compilation:* The time evolution operator is compiled and optimized with the quantum simulation kernel compiler Paulihedral [26], Rustiq [10] or Tetris [21] followed by Qiskit L3 optimization. We chose $\{CNOT, U3\}$ as the basis gates for noisy simulation. For Tetris [21], we tested against the IBM *Manhattan*, *Sycamore*, and *Montreal* superconducting architectures.

*4) Noisy Simulation:* We use the Qiskit Aer [19] to simulate circuits with depolarizing noise on single- and two-qubit gates.

*5) Real-System Study:* We also executed the compiled circuit on the IonQ Forte 1 quantum computer. It has 36 ion trap qubits with all-to-all connectivity, $99.98\%$ single-qubit gate fidelity, $98.99\%$ double-qubit gate fidelity, and $99.02\%$ readout fidelity. Due to current hardware limitations, only the $H_2$ molecule simulation is executed.

*6) Metrics:* We use the following metrics: 1) Pauli weight of the mapped qubit Hamiltonian, 2) $CNOT$ gate count and depth of the compiled circuit, and 3) the simulated system energy in noisy simulation and real-system study.

### C. Pauli Weight and Circuit Metrics

We evaluate the Pauli weight of the qubit Hamiltonians generated by different Fermion-to-qubit mapping methods and the circuits to simulate these Hamiltonians.

*1) Electronic Structure Model:* Table I shows the Pauli weight and circuit metrics of different molecules. For the smallest case where Fermihedral (FH) can find the optimal Pauli weight, HATT achieves similar results. For larger cases where FH can only provide approximate solutions or even fail to solve, HATT consistently shows the best results in all metrics except for the $H_2$ 631g and $H_2$ 631g frz where Jordan-Wigner transformation (JW) is slightly better. Compared with (JW), HATT, on average, reduces Pauli weight by $9.53\%$, number of $CNOT$ gates by $15.40\%$, and circuit depth by $13.08\%$. Compared with the Bravyi-Kitaev transformation (BK), HATT reduces $15.47\%$ Pauli weight, $24.54\%$ $CNOT$ count and $19.50\%$ circuit depth. Compared gainst the balanced ternary tree (BTT), HATT reduces $14.42\%$ Pauli weight, $25.51\%$ $CNOT$ count and $25.88\%$ circuit depth. Compared with JW on more sophisticated workflows, like Rustiq [10] and Tetris [21], HATT still demonstrates better performance for almost all cases (results shown in Table V and IV). Compared with JW+Rustiq, HATT+Rustiq further reduces up to $18.2\%$ $CNOT$ count, $21.83\%$ $U3$ count, and $13.5\%$

TABLE I

EVALUATION RESULT OF ELECTRONIC STRUCTURE MODEL. THE BEST RESULTS OF EACH METRIC ARE HIGHLIGHTED IN BOLD. '–' INDICATES THE CASE IS TOO LARGE TO SOLVE BY FERMIHEDRAL (FH). '*' INDICATES THAT FERMIHEDRAL ONLY FINDS AN APPROXIMATELY OPTIMAL SOLUTION. 'FRZ' INDICATES FREEZE CORE TRANSFORMATION. 'STO3G' [41] INDICATES THE BASIS.

| Case | Modes | Pauli Weight | | | | | $CNOT$ Gate Count | | | | | Circuit Depth | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | JW | BK | BTT | FH | HATT | JW | BK | BTT | FH | HATT | JW | BK | BTT | FH | HATT |
| $H_2$ sto3g | 4 | **32** | 34 | 36 | **32** | **32** | **21** | 25 | 32 | 22 | **21** | 34 | 35 | 53 | **33** | 34 |
| $H_2$ sto3g frz | 4 | **32** | 34 | 36 | **32** | **32** | **21** | 25 | 32 | 22 | **21** | 34 | 35 | 53 | **33** | 34 |
| $H_2$ 631g | 8 | **728** | 756 | 834 | 912* | 768 | **508** | 646 | 628 | 707* | 618 | **753** | 878 | 911 | 1003* | 860 |
| $H_2$ 631g frz | 8 | **728** | 756 | 834 | 912* | 768 | **508** | 646 | 628 | 707* | 622 | **753** | 878 | 911 | 1003* | 881 |
| $LiH$ sto3g | 12 | 3248 | 3660 | 3536 | 3842* | **2926** | 2377 | 2373 | 2298 | 2985* | **1637** | 3174 | 3249 | 3306 | 3987* | **2425** |
| $LiH$ sto3g frz | 10 | 1240 | 1410 | 1358 | 1230* | **1082** | 944 | 1162 | 1016 | 769* | **725** | 1215 | 1500 | 1424 | 1092* | **992** |
| $NH$ sto3g frz | 10 | 1240 | 1410 | 1358 | 1230* | **1082** | 944 | 1162 | 1016 | 769* | **725** | 1215 | 1500 | 1424 | 1092* | **992** |
| $NH$ 631g | 22 | 58220 | 56161 | 50522 | – | **42855** | 26999 | 29074 | 28619 | – | **18082** | 38043 | 40873 | 40617 | – | **27347** |
| $NH$ 631g frz | 20 | 34576 | 35106 | 30964 | – | **26000** | 17867 | 18514 | 19809 | – | **11949** | 25146 | 25965 | 27454 | – | **17614** |
| $BeH_2$ sto3g frz | 12 | 5624 | 6360 | 6128 | 6448* | **5318** | 4054 | 3311 | 3883 | 4391* | **2850** | 5458 | 4834 | 5668 | 6213* | **4236** |
| $O_2$ sto3g | 20 | 17096 | 17320 | 15472 | – | **13088** | 11520 | 12459 | 10485 | – | **8701** | 14468 | 15854 | 14170 | – | **11102** |

circuit depth. For compilation onto superconducting architectures (*Manhattan*, *Sycamore* and *Montreal*), HATT+Tetris can outperform JW+Tetris with reduction on $CNOT$ count, $U3$ count, and circuit depth by up to $18.20\%$, $21.83\%$ and $13.55\%$, respectively. Note that both Rustiq and Tetris are developed using JW. It is possible to further optimize the workflow for the Pauli strings patterns generated by HATT.

*2) Fermi-Hubbard Model:* Table II shows the Pauli weight and two metrics of the compiled circuits for the Fermi-Hubbard model benchmarks. It can be observed that although FH achieved the minimal Pauli weight for up to 18 modes, it cannot solve more significant cases. HATT achieves a lower Pauli weight consistently and a lower circuit complexity most of the time against JW, BK, and BTT as our method is tailored according to a specific Hamiltonian and does capture the optimization opportunities within.

On Fermi-Hubbard models, compared to JW, HATT on average reduces Pauli weight by $20.90\%$, number of $CNOT$ gates by $22.90\%$, and circuit depth by $7.88\%$. Compared to BK, HATT reduces $6.48\%$ Pauli weight, $12.11\%$ $CNOT$ count and $8.16\%$ circuit depth. Against BTT, HATT reduces $4.77\%$ Pauli weight, $16.58\%$ $CNOT$ count and $18.11\%$ circuit depth correspondingly. On small scales ($8 \sim 18$ modes), HATT shows the results closest to the optimal solution by FH.

*3) Collective Neutrino Oscillation:* Table III shows the Pauli weight and metrics of compiled circuits of different neutrino oscillation test cases. Fermihedral (FH) is not evaluated since all the cases are too large for FH. Among all the cases, HATT always achieves the lower Pauli weight. Although HATT shows slightly higher circuit overhead compared with JW, the trend shows that HATT has more advantage as the problem size increases. Compared to JW, HATT on average reduces Pauli weight by $15.65\%$, number of $CNOT$ gates by $4.01\%$. Compared to BK, HATT reduces $14.58\%$ Pauli weight, $17.69\%$ $CNOT$ count and $14.92\%$ circuit depth. Against BTT, HATT reduces $11.95\%$ Pauli weight, $17.30\%$ $CNOT$ count and $17.31\%$ circuit depth correspondingly.

### D. Noisy Simulation and Real-System Study

We performed the noisy simulation and real-system study of compiled circuits for $H_2$ and $LiH$ sto3g frz cases from the electronic structure model. 'frz' means that the inner layer electrons are fixed.

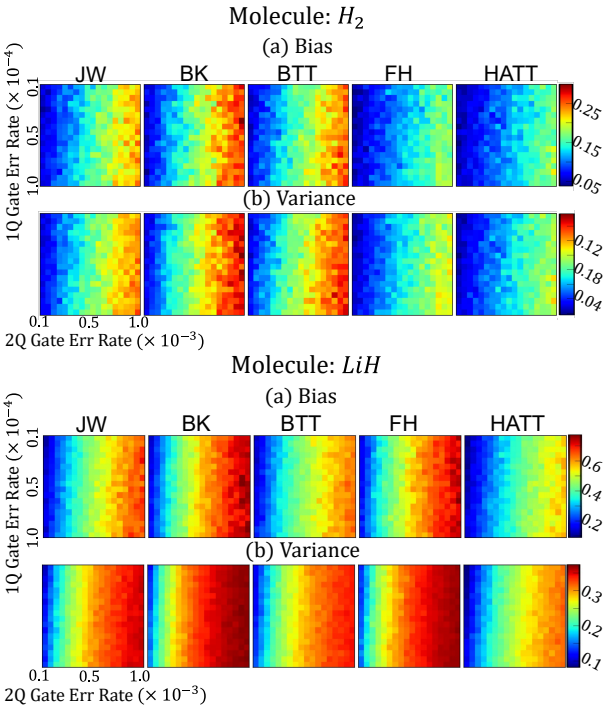



Fig. 10. Noisy simulation result of $H_2$ and $LiH$ sto3g frz molecule. Energy is measured for 1000 shots. Bias is calculated against the theoretical results.

*1) Noisy Simulation:* We simulated the circuit generated by different Fermion-to-qubit mappings under depolarizing errors. The error rate range of single-qubit gates is $10^{-5} \sim 10^{-4}$ and $10^{-4} \sim 10^{-3}$ for double-qubit gates. We simulate each circuit for 1000 shots, and the final system energy is measured and compared against the theoretical results.

Figure 10 shows the simulation results of $H_2$ (upper half) and $LiH$ sto3g frz (lower half) molecules. Bias and variance are calculated against the theoretical values based on 1000 shots. It can be observed that HATT has the lowest deviation and variance (heatmap closer to blue), which outperforms JW, BK, and BTT, achieving similar results with the optimal FH.

*2) Real-System Study:* Figure 11 shows the results of $H_2$ ground state energy simulation on the IonQ Forte 1 device. The

TABLE II
EVALUATION RESULT OF FERMI-HUBBARD MODEL. '–' INDICATES THE CASE IS TOO LARGE TO SOLVE BY FERMIHEDRAL (FH).

| Geometry | Modes | Pauli Weight | | | | | $CNOT$ Gate Count | | | | | Circuit Depth | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | JW | BK | BTT | FH | HATT | JW | BK | BTT | FH | HATT | JW | BK | BTT | FH | HATT |
| $2 \times 2$ | 8 | 80 | 80 | 86 | **56** | 76 | 51 | 71 | 77 | **37** | 62 | 60 | 99 | 108 | **36** | 73 |
| $2 \times 3$ | 12 | 212 | 200 | 199 | **161** | 187 | 159 | 172 | 161 | **123** | 146 | 161 | 223 | 229 | **160** | 196 |
| $2 \times 4$ | 16 | 304 | 263 | 260 | **230** | 256 | 228 | **183** | 208 | 195 | 189 | **225** | 239 | 256 | 230 | 249 |
| $3 \times 3$ | 18 | 492 | 428 | 408 | **352** | 410 | 378 | 296 | 317 | 266 | **260** | 328 | 391 | 427 | **270** | 356 |
| $2 \times 5$ | 20 | 396 | 348 | 356 | – | **330** | 287 | 270 | 266 | – | **224** | 305 | **275** | 320 | – | 283 |
| $3 \times 4$ | 24 | 704 | 620 | 580 | – | **524** | 528 | 460 | 433 | – | **364** | 462 | 529 | 496 | – | **401** |
| $2 \times 7$ | 28 | 580 | 493 | 502 | – | **473** | 405 | 380 | 373 | – | **320** | 465 | 374 | 399 | – | **333** |
| $3 \times 5$ | 30 | 916 | 756 | **706** | – | **706** | 661 | 523 | 563 | – | **490** | 621 | **550** | 629 | – | 582 |
| $4 \times 4$ | 32 | 1152 | 790 | 784 | – | **760** | 842 | 531 | 651 | – | **483** | 712 | 562 | 749 | – | **553** |
| $3 \times 6$ | 36 | 1128 | 932 | 876 | – | **806** | 794 | 662 | 668 | – | **544** | 780 | 684 | 694 | – | **487** |
| $4 \times 5$ | 40 | 1504 | 1030 | **986** | – | **986** | 1055 | 665 | 782 | – | **601** | 941 | **561** | 795 | – | 618 |

TABLE III
EVALUATION RESULT OF COLLECTIVE NEUTRINO OSCILLATION. FERMIHEDRAL (FH) IS NOT INCLUDED SINCE ALL THE BENCHMARKS IN THIS APPLICATION ARE TOO LARGE FOR FERMIHEDRAL.

| Case | Modes | Pauli Weight | | | | $CNOT$ Gate Count | | | | Circuit Depth | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | JW | BK | BTT | HATT | JW | BK | BTT | HATT | JW | BK | BTT | HATT |
| $3 \times 2F$ | 12 | 1424 | 1568 | 1556 | **1290** | **776** | 986 | 1092 | 850 | **1137** | 1421 | 1554 | 1229 |
| $4 \times 2F$ | 16 | 4048 | 4011 | 4244 | **3720** | **2115** | 2742 | 2657 | 2203 | **3003** | 3763 | 3788 | 3110 |
| $3 \times 3F$ | 18 | 5550 | 5770 | 5548 | **5153** | 2912 | 3667 | 3391 | **2703** | **3927** | 4911 | 4801 | 3949 |
| $5 \times 2F$ | 20 | 9240 | 9800 | 9016 | **7852** | 4630 | 5285 | 5685 | **4487** | **6261** | 7476 | 8021 | 6414 |
| $4 \times 3F$ | 24 | 16216 | 16462 | 14806 | **14267** | 7996 | 8952 | 8243 | **7141** | 10530 | 12098 | 11786 | **10440** |
| $6 \times 2F$ | 24 | 18280 | 18594 | 16992 | **15047** | 8868 | 9168 | 9612 | **8382** | **11571** | 13338 | 13693 | 11986 |
| $7 \times 2F$ | 28 | 32704 | 31088 | 28876 | **25074** | 15440 | 14733 | 15358 | **13322** | **19281** | 21278 | 22148 | 19400 |
| $5 \times 3F$ | 30 | 37690 | 33776 | 32154 | **31418** | 17872 | 17460 | 16957 | **15204** | 21958 | 23708 | 23872 | **21697** |
| $6 \times 3F$ | 36 | 75540 | 66262 | 60576 | **58229** | 34697 | 30193 | 29361 | **26298** | 41198 | 41702 | 41995 | **38502** |
| $7 \times 3F$ | 42 | 136486 | 114833 | 101717 | **99334** | 60414 | 48846 | 48155 | **45045** | 69117 | 67548 | 67600 | **64686** |

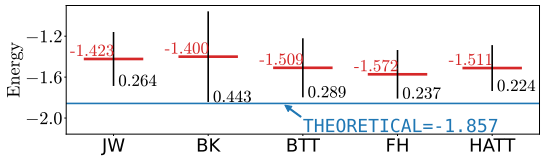



Fig. 11. $H_2$ molecule simulation results on IonQ Forte 1 quantum computer

red horizontal lines indicate the average measured energy of 1000 shots using the corresponding mapping. The associated black vertical lines indicate the variance of the measured energy across all shots. The blue line indicates the theoretical result. HATT achieved the second closest to theory, with the closest being the small-scale optimal FH solution. HATT also has the lowest variance.

### E. Execution Time and Scalability

To understand the scalability of the proposed tree construction algorithm and the impact of our performance optimization, we compared our unoptimized tree construction algorithm in Section III-C (HATT (unopt)), optimized algorithm in Section IV-C (HATT) and the optimal exhaustive search method Fermihedral [27] (FH) on the time consumption to produce a Fermion-to-qubit mapping for a simple Hamiltonian: $\mathcal{H}_{\mathcal{F}} = \sum_{i=0}^{2N-1} M_i$ at different sizes. As shown in Figure 12, both HATT (unopt) and HATT scales to large cases in polynomial time, but FH show its exponential complexity. Also, HATT has a shorter execution time compared with unoptimized HATT (unopt) as our optimization reduces the algorithm's complexity from $O(N^4)$ to $O(N^3)$.

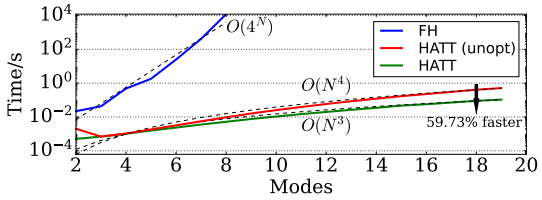



Fig. 12. Scalability of Fermihedral (FH), unoptimized HATT (HATT (unopt)) and HATT with optimization (HATT). Black dotted lines (---) show the regression of complexity orders.

### F. Performance Cost of Optimized HATT

We also compared the Pauli weight in Table VI of HATT (unopt) and HATT to show how optimization could affect performance. The result shows that HATT and HATT (unopt) generate Hamiltonians with $\sim$ 0.43% difference in Pauli weight on average. Thus, the optimization does not come with a significant performance trade-off.

## VI. RELATED WORKS

**Fermion-to-Qubit Mappings**: Previous Fermion-to-qubit mappings includes the Jordan-Wigner transformation [22], Bravyi-Kitaev transformation [5], parity mapping [4] and ternary tree mapping [20], [30]. The Bravyi-Kitaev transformation and ternary tree mapping generate the theoretical minimum Pauli weight per Majorana operator ($O(\log N)$).

TABLE IV
EVALUATION RESULT OF ELECTRONIC STRUCTURE MODEL WITH **TETRIS** [21] ON MANHATTAN, SYCAMORE AND MONTREAL

| Case | $CNOT$ Gate Count | | $U3$ Gate Count | | Circuit Depth | |
|---|---|---|---|---|---|---|
| | JW | HATT | JW | HATT | JW | HATT |
| Manhattan | | | | | | |
| $H_2$ sto3g | 40 | **38** | 47 | **37** | 62 | **56** |
| $H_2$ sto3g frz | 42 | **36** | **40** | 43 | 61 | **57** |
| $H_2$ 631g | 1479 | **1420** | 1302 | **1172** | 1744 | **1659** |
| $H_2$ 631g frz | 1506 | **1314** | 1290 | **1147** | 1769 | **1540** |
| $LiH$ sto3g | 7167 | **6517** | 5029 | **4616** | 7302 | **6890** |
| $LiH$ sto3g frz | 2753 | **2282** | 2000 | **1560** | 2915 | **2346** |
| $NH$ sto3g frz | 2824 | **2415** | 1999 | **1610** | 2881 | **2563** |
| $NH$ 631g | 145396 | **138719** | 80484 | **72264** | 126398 | **123177** |
| $NH$ 631g frz | 85256 | **80046** | 48451 | **46037** | 76164 | **73964** |
| $BeH_2$ sto3g frz | 11646 | **11513** | **8911** | 8980 | 12711 | **11959** |
| $O_2$ sto3g | 48660 | **46483** | 24231 | **23460** | **38985** | 39446 |
| Sycamore | | | | | | |
| $H_2$ sto3g | 42 | **38** | 40 | **37** | 60 | **56** |
| $H_2$ sto3g frz | 42 | **38** | 40 | **37** | **56** | **56** |
| $H_2$ 631g | 1120 | **1032** | 1017 | **965** | 1348 | **1228** |
| $H_2$ 631g frz | 1122 | **1040** | 1044 | **988** | 1383 | **1225** |
| $LiH$ sto3g | 5760 | **5665** | 4114 | **3614** | 6031 | **5941** |
| $LiH$ sto3g frz | 2098 | **2001** | 1472 | **1295** | 2202 | **2183** |
| $NH$ sto3g frz | 2214 | **1976** | 1540 | **1364** | 2356 | **2140** |
| $NH$ 631g | 98096 | **94642** | 61052 | **57630** | 89638 | **86487** |
| $NH$ 631g frz | 60009 | **56304** | 38446 | **33891** | 56151 | **52170** |
| $BeH_2$ sto3g frz | 9607 | **9350** | 7155 | **6895** | 10427 | **9805** |
| $O_2$ sto3g | 31986 | **29690** | 19296 | **17390** | 28301 | **27031** |
| Montreal | | | | | | |
| $H_2$ sto3g | 42 | **36** | **40** | 43 | 61 | **57** |
| $H_2$ sto3g frz | 42 | **38** | 40 | **37** | 61 | **56** |
| $H_2$ 631g | 1409 | **1271** | 1324 | **1140** | 1633 | **1510** |
| $H_2$ 631g frz | 1435 | **1417** | 1300 | **1160** | 1699 | **1641** |
| $LiH$ sto3g | 7057 | **6465** | 4834 | **4731** | 7171 | **6811** |
| $LiH$ sto3g frz | 2865 | **2522** | 1949 | **1628** | 3013 | **2553** |
| $NH$ sto3g frz | 2862 | **2467** | 1939 | **1524** | 2999 | **2523** |
| $NH$ 631g | 146957 | **144169** | 77300 | **74026** | 127955 | **124624** |
| $NH$ 631g frz | 87792 | **84268** | 47537 | **45356** | 77307 | **74992** |
| $BeH_2$ sto3g frz | **11798** | 12350 | 8913 | **8412** | 12783 | **12318** |
| $O_2$ sto3g | 50641 | **46443** | 24294 | **22969** | 40857 | **38649** |

TABLE V
EVALUATION RESULT OF ELECTRONIC STRUCTURE MODEL WITH **RUSTIQ** [10]

| Case | $CNOT$ Gate Count | | $U3$ Gate Count | | Circuit Depth | |
|---|---|---|---|---|---|---|
| | JW | HATT | JW | HATT | JW | HATT |
| $H_2$ sto3g | **7** | **7** | 6 | **2** | **7** | **7** |
| $H_2$ sto3g frz | **7** | **7** | 6 | **2** | **7** | **7** |
| $H_2$ 631g | **217** | 227 | 180 | **164** | 164 | **161** |
| $H_2$ 631g frz | 217 | **216** | 180 | **170** | 164 | **149** |
| $LiH$ sto3g | 1348 | **1230** | 1031 | **872** | 625 | **561** |
| $LiH$ sto3g frz | 423 | **413** | 318 | **306** | 252 | **250** |
| $NH$ sto3g frz | 423 | **346** | 318 | **308** | 252 | **225** |
| $NH$ 631g | **38988** | 40266 | **30514** | 32005 | **8169** | 8273 |
| $NH$ 631g frz | 22666 | **18993** | 17726 | **13856** | 5268 | **4554** |
| $BeH_2$ sto3g frz | 2200 | **1960** | 1705 | **1555** | 1000 | **936** |
| $O_2$ sto3g | 12012 | **10592** | 9526 | **7529** | 2766 | **2582** |

However, all of them fail to achieve optimal solutions with a specific Hamiltonian since the pattern of the Fermionic system is disregarded in their construction. Recent superfast encoding [6], [38] captures structures of the Fermionic system to produce optimized mappings. However, it is only restricted to systems with local Fermionic modes and cyclic interaction patterns. Fermihedral [27] solves a given Hamiltonian's theoretical optimum Pauli weight mapping with an SAT problem. However, SAT is NP-complete, and Fermihedral fails to scale to cases larger than $\sim$ 20 qubits. In contrast, HATT combines the goal of optimizing Pauli weight and Hamiltonian information with a scalable ternary tree construction process. It works on any Fermionic system and scales to larger cases with polynomial complexity $O(N^3)$.

TABLE VI
COMPARISON OF PAULI WEIGHT BETWEEN HATT (UNOPT) AND HATT (UP TO 24 MODES)

| Case | HATT (unopt) | HATT | Case | HATT (unopt) | HATT |
|---|---|---|---|---|---|
| $H_2$ sto3g | 32 | 32 | $3 \times 3$ | 404 | 410 |
| $LiH$ sto3g frz | 1082 | 1082 | $2 \times 5$ | 338 | 330 |
| $LiH$ sto3g | 2880 | 2850 | $3 \times 4$ | 558 | 524 |
| $H_2O$ sto3g | 5545 | 5545 | $3 \times 2F$ | 1266 | 1290 |
| $CH_4$ sto3g | 37182 | 37077 | $3 \times 3F$ | 4976 | 5153 |
| $O_2$ sto3g | 13082 | 13370 | $4 \times 2F$ | 3595 | 3720 |
| $2 \times 2$ | 82 | 76 | $4 \times 3F$ | 14330 | 14267 |
| $2 \times 3$ | 194 | 187 | $5 \times 2F$ | 7844 | 7852 |
| $2 \times 4$ | 261 | 256 | $6 \times 2F$ | 15005 | 15047 |

**Quantum Simulation Compilers**: Some recent works successfully identify the computation pattern in quantum simulation to further optimize quantum circuits, including architectural-aware synthesis [25], [26], [44], reordering Pauli strings for gate cancellation [16], [17], and simultaneous diagonalization [8], [9], [45]. These works happen after Fermion-to-qubit mappings and do not rely on a specific Fermion-to-qubit mapping. HATT is compatible with these works.

**Quantum Compilers and Optimizations**: Modern quantum compilers, such as Qiskit [19], Cirq [11], Braket [2] and t|ket$\rangle$ [39] apply multiple optimization passes to quantum circuits. These optimizations, including gate cancellations [33] and rewriting [40], qubit routing [24], [31], [32], are applied after the Fermion-to-qubit mapping stage where the Hamiltonian simulation is converted to circuits. Thus, they are compatible with HATT.

## VII. CONCLUSION

In this work, we presented the Hamiltonian-Adaptive ternary tree (HATT) compilation framework, a novel optimization for compiling Fermion-to-qubit mapping that leverages the high-level structure of the input Fermionic Hamiltonian. HATT efficiently compiles ternary trees to optimize the mapping process, significantly reducing the Pauli weight and circuit complexity compared to existing methods. Our approach retains the *vacuum state preservation* property and achieves a polynomial ($O(N^3)$) complexity, making it suitable for large-scale simulations that classical and exhaustive methods struggle to handle.


## ACKNOWLEDGEMENT

This work was supported by 1) the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under Contract No. DE-AC05-00OR22725 through the Accelerated Research in Quantum Computing Program MACH-Q project and 2) the Network for Neutrinos, Nuclear Astrophysics and Symmetries (N3AS), through the National Science Foundation Physics Frontier Center award No. PHY-2020275. YL and GL are supported in part by NSF CAREER Award No. CCF-2338773 and ExpandQISE Award No. OSI-2427020. GL is also supported by the Intel Rising Star Faculty Award. The IonQ Forte 1 experiments are supported by Amazon Web Service Cloud Credit. KY and JH are supported by Penn Undergraduate Research Mentoring Program (PURM).

## APPENDIX

### A. Abstract

The artifact contains the code of HATT Fermion-to-qubit mapping and scripts to figures and tables (Figures 10, 11, 12; Tables I, II, III, IV, V and VI. It requires an x86-64 Linux server with at least 32GB RAM and 10GB available hard drive. A CUDA GPU is recommended to speed up the simulation. We also provide instructions on how to run the circuit on the IonQ quantum computer device, which requires extra credits towards Amazon Braket. Due to internal randomization, the result could be deviated.

### B. Artifact check-list (meta-information)

- **Algorithm:** HATT has following core algorithms:
  - `HATTNaiveMapper`: Generate ternary tree Fermion-to-qubit mapping based on Algorithm 1, corresponds to **HATT (unopt)**.
  - `HATTMapper`: Generate ternary tree Fermion-to-qubit mapping with optimization from Algorithm 2 and 3, corresponds to **HATT**.
- **Program:** `HATTMapper` is implemented in the file 'hattmapper/hatt_mapper.py'. `HATTNaiveMapper` is implemented in the file 'hattmapper/hatt_naive_mapper.py'.
- **Run-time environment:** Python, Jupyter Notebook
- **Hardware:** Single x86-64 CPU Linux server, preferably with a CUDA GPU, to accelerate the noisy simulation.
- **Metrics:** The following metrics are tested:
  - Pauli weight
  - Circuit complexity (number of gates, depth)
  - Observed system energy
- **Output:** The code creates a Fermion-to-qubit mapper adapted to the `FermionicMapper` interface in Qiskit-Nature.
- **Experiments:** There are four notebooks for different experiments:
  1) `ae-forte.ipynb`: Figure 11.
  2) `ae-lattice.ipynb`: Tables II, III and VI.
  3) `ae-molecule.ipynb`: Figure 10; Tables I, IV and V.
  4) `ae-scalability.ipynb`: Figure 12.
- **Disk space required:** 10GB
- **Time to prepare workflow:** 5 minutes
- **Time to complete experiments:** half to one day
- **Publicly available:** Yes
- **Code licenses:** MIT License
- **Workflow framework used:** Python, Qiskit, Qiskit-Nature, Amazon Braket, Jupyter Notebook, Rust
- **Archived:** 10.5281/zenodo.14238346

### C. Description

*1) How to access:* The artifact is available at the following GitHub repository https://github.com/acasta-yhliu/hattmapper.git on branch `ae` or with DOI https://doi.org/10.5281/zenodo.14238346. You can clone the repository or submit issues if there is any problem.

*2) Hardware dependencies:* Our artifact can be run on a regular Linux server with a single CPU and, preferably, a CUDA GPU. RAM is not strictly limited, but it is recommended to be 32GB or more.

Figure 11 requires execution on the IonQ Forte-1 quantum computer (available through reservation on Amazon Web Service). If you wish to run the quantum circuit simulating $H_2$ molecule on this real quantum computer, the cost should be around $7000 (charged by Amazon) with our setup through reservation. Consequently, we exclude this part from artifact evaluation, and the exclusion will not affect the major results of this paper.

*3) Software dependencies:* The artifact is implemented in Python 3.10. We also require Python packages that are listed in `requirements.txt`. Some essential packages are Qiskit 1.1.0, Qiskit-Nature pre-release, Rustiq [10]. We also included necessary Tetris [21] and Paulihedral [26] files. Rustiq [10] requires Rust [28] to be installed.

### D. Installation

Rust [28] is required for Rustiq [10]. A general way to install Rust is to use `rustup` by https://www.rust-lang.org/tools/install. You may also refer to Rust's other manuals.

You can clone the repository locally and prepare the Python environment by the following commands:

```
$ python3 -m venv venv
$ source venv/bin/activate
$ pip3 install -r requirements.txt
```

### E. Experiment workflow

After unzipping our code and preparing the environment in the installation setup, you can execute the Jupyter Notebooks in the virtual environment to reproduce the result.

1) `ae-forte.ipynb`: Figure 11.
2) `ae-lattice.ipynb`: Tables II, III and VI.
3) `ae-molecule.ipynb`: Figure 10; Tables I, IV and V.
4) `ae-scalability.ipynb`: Figure 12.

Notebooks are annotated in each section for each figure and table. The tables are pretty printed.

### F. Evaluation and expected results

Notebooks should reproduce the exact figures as in our paper. All tables are printed in the notebook. However, due to internal randomization, the result could be deviated.

### G. Experiment customization

To use `HATTMapper` with a custom Fermionic system, replace other Fermion-to-qubit mappers in Qiskit-Nature with `HATTMapper` and initialize with a Hamiltonian:

```
mapper = HATTMapper(hamiltonian)
```

## REFERENCES

[1] A. Altland, B. Simons, and B. Simons, Condensed Matter Field Theory. Cambridge University Press, 2006. [Online]. Available: https://books.google.com/books?id=0KMkfAMe3JkC

[2] Amazon Web Services, "Amazon Braket," 2020. [Online]. Available: https://aws.amazon.com/braket/

[3] V. Barger, D. Marfatia, and K. Whisnant, The Physics of Neutrinos. Princeton University Press, 2012. [Online]. Available: https://books.google.com/books?id=qso8NEr3XY8C

[4] S. Bravyi, J. M. Gambetta, A. Mezzacapo, and K. Temme, "Tapering off qubits to simulate fermionic hamiltonians," 2017.

[5] S. B. Bravyi and A. Y. Kitaev, "Fermionic quantum computation," Annals of Physics, vol. 298, no. 1, pp. 210–226, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003491602962548

[6] R. W. Chien, S. Xue, T. S. Hardikar, K. Setia, and J. D. Whitfield, "Analysis of superfast encoding performance for electronic structure simulations," Physical Review A, vol. 100, no. 3, sep 2019. [Online]. Available: https://doi.org/10.1103%2Fphysreva.100.032337

[7] V. Cirigliano, S. Sen, and Y. Yamauchi, "Neutrino many-body flavor evolution: the full hamiltonian," 2024. [Online]. Available: https://arxiv.org/abs/2404.16690

[8] A. Cowtan, S. Dilkes, R. Duncan, W. Simmons, and S. Sivarajah, "Phase gadget synthesis for shallow circuits," Electronic Proceedings in Theoretical Computer Science, vol. 318, pp. 213–228, may 2020. [Online]. Available: https://doi.org/10.4204%2Feptcs.318.13

[9] A. Cowtan, W. Simmons, and R. Duncan, "A generic compilation strategy for the unitary coupled cluster ansatz," 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2007.10515

[10] T. G. de Brugière and S. Martiel, "Faster and shorter synthesis of hamiltonian simulation circuits," 2024. [Online]. Available: https://arxiv.org/abs/2404.03280

[11] C. Developers, "Cirq," Jul. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.8161252

[12] T. Q. N. developers and contributors, "Qiskit nature 0.6.0," Apr. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7828768

[13] P. A. M. Dirac and N. H. D. Bohr, "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, vol. 114, no. 767, pp. 243–265, 1927. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1927.0039

[14] R. P. Feynman, "Simulating physics with computers," International Journal of Theoretical Physics, vol. 21, no. 6, pp. 467–488, Jun 1982. [Online]. Available: https://doi.org/10.1007/BF02650179

[15] P. D. Group, R. L. Workman, V. D. Burkert, V. Crede, E. Klempt, U. Thoma, L. Tiator, K. Agashe, G. Aielli, B. C. Allanach, C. Amsler, M. Antonelli, E. C. Aschenauer, D. M. Asner, H. Baer, S. Banerjee, R. M. Barnett, L. Baudis, C. W. Bauer, J. J. Beatty, V. I. Belousov, J. Beringer, A. Bettini, O. Biebel, K. M. Black, E. Blucher, R. Bonventre, V. V. Bryzgalov, O. Buchmuller, M. A. Bychkov, R. N. Cahn, M. Carena, A. Ceccucci, A. Cerri, R. S. Chivukula, G. Cowan, K. Cranmer, O. Cremonesi, G. D'Ambrosio, T. Damour, D. de Florian, A. de Gouvêa, T. DeGrand, P. de Jong, S. Demers, B. A. Dobrescu, M. D'Onofrio, M. Doser, H. K. Dreiner, P. Eerola, U. Egede, S. Eidelman, A. X. El-Khadra, J. Ellis, S. C. Eno, J. Erler, V. V. Ezhela, W. Fetscher, B. D. Fields, A. Freitas, H. Gallagher, Y. Gershtein, T. Gherghetta, M. C. Gonzalez-Garcia, M. Goodman, C. Grab, A. V. Gritsan, C. Grojean, D. E. Groom, M. Grünewald, A. Gurtu, T. Gutsche, H. E. Haber, M. Hamel, C. Hanhart, S. Hashimoto, Y. Hayato, A. Hebecker, S. Heinemeyer, J. J. Hernández-Rey, K. Hikasa, J. Hisano, A. Höcker, J. Holder, L. Hsu, J. Huston, T. Hyodo, A. Ianni, M. Kado, M. Karliner, U. F. Katz, M. Kenzie, V. A. Khoze, S. R. Klein, F. Krauss, M. Kreps, P. Križan, B. Krusche, Y. Kwon, O. Lahav, J. Laiho, L. P. Lellouch, J. Lesgourgues, A. R. Liddle, Z. Ligeti, C.-J. Lin, C. Lippmann, T. M. Liss, L. Littenberg, C. Lourenço, K. S. Lugovsky, S. B. Lugovsky, A. Lusiani, Y. Makida, F. Maltoni, T. Mannel, A. V. Manohar, W. J. Marciano, A. Masoni, J. Matthews, U.-G. Meißner, I.-A. Melzer-Pellmann, M. Mikhasenko, D. J. Miller, D. Milstead, R. E. Mitchell, K. Mönig, P. Molaro, F. Moortgat, M. Moskovic, K. Nakamura, M. Narain, P. Nason, S. Navas, A. Nelles, M. Neubert, P. Nevski, Y. Nir, K. A. Olive, C. Patrignani, J. A. Peacock, V. A. Petrov, E. Pianori, A. Pich, A. Piepke, F. Pietropaolo, A. Pomarol, S. Pordes, S. Profumo, A. Quadt, K. Rabbertz, J. Rademacker, G. Raffelt, M. Ramsey-Musolf, B. N. Ratcliff, P. Richardson, A. Ringwald, D. J. Robinson, S. Roesler, S. Rolli, A. Romaniouk, L. J. Rosenberg, J. L. Rosner, G. Rybka, M. G. Ryskin, R. A. Ryutin, Y. Sakai, S. Sarkar, F. Sauli, O. Schneider, S. Schönert, K. Scholberg, A. J. Schwartz, J. Schwiening, D. Scott, F. Sefkow, U. Seljak, V. Sharma, S. R. Sharpe, V. Shiltsev, G. Signorelli, M. Silari, F. Simon, T. Sjöstrand, P. Skands, T. Skwarnicki, G. F. Smoot, A. Soffer, M. S. Sozzi, S. Spanier, C. Spiering, A. Stahl, S. L. Stone, Y. Sumino, M. J. Syphers, F. Takahashi, M. Tanabashi, J. Tanaka, M. Taševský, K. Terao, K. Terashi, J. Terning, R. S. Thorne, M. Titov, N. P. Tkachenko, D. R. Tovey, K. Trabelsi, P. Urquijo, G. Valencia, R. Van de Water, N. Varelas, G. Venanzoni, L. Verde, I. Vivarelli, P. Vogel, W. Vogelsang, V. Vorobyev, S. P. Wakely, W. Walkowiak, C. W. Walter, D. Wands, D. H. Weinberg, E. J. Weinberg, N. Wermes, M. White, L. R. Wiencke, S. Willocq, C. G. Wohl, C. L. Woody, W.-M. Yao, M. Yokoyama, R. Yoshida, G. Zanderighi, G. P. Zeller, O. V. Zenin, R.-Y. Zhu, S.-L. Zhu, F. Zimmermann, and P. A. Zyla, "Review of Particle Physics," Progress of Theoretical and Experimental Physics, vol. 2022, no. 8, p. 083C01, 08 2022. [Online]. Available: https://doi.org/10.1093/ptep/ptac097

[16] K. Gui, T. Tomesh, P. Gokhale, Y. Shi, F. T. Chong, M. Martonosi, and M. Suchara, "Term grouping and travelling salesperson for digital quantum simulation," 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2001.05983

[17] M. B. Hastings, D. Wecker, B. Bauer, and M. Troyer, "Improving quantum algorithms for quantum chemistry," Quantum Info. Comput., vol. 15, no. 1–2, p. 1–21, jan 2015. [Online]. Available: https://doi.org/10.48550/arXiv.1403.1539

[18] W. Hu, H. An, Z. Guo, Q. Jiang, X. Qin, J. Chen, W. Jia, C. Yang, Z. Luo, J. Li, W. Wu, G. Tan, D. Jia, Q. Lu, F. Liu, M. Tian, F. Li, Y. Huang, L. Wang, S. Liu, and J. Yang, "2.5 million-atom ab initio electronic-structure simulation of complex metallic heterostructures with dgdft," in SC22: International Conference for High Performance Computing, Networking, Storage and Analysis, 2022, pp. 1–13.

[19] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, "Quantum computing with Qiskit," 2024.

[20] Z. Jiang, A. Kalev, W. Mruczkiewicz, and H. Neven, "Optimal fermion-to-qubit mapping via ternary trees with applications to reduced quantum states learning," Quantum, vol. 4, p. 276, Jun. 2020. [Online]. Available: http://dx.doi.org/10.22331/q-2020-06-04-276

[21] Y. Jin, Z. Li, F. Hua, T. Hao, H. Zhou, Y. Huang, and E. Z. Zhang, "Tetris: A compilation framework for vqa applications in quantum computing," 2024. [Online]. Available: https://arxiv.org/abs/2309.01905

[22] P. Jordan and E. Wigner, "Über das paulische äquivalenzverbot," Zeitschrift für Physik, vol. 47, no. 9, pp. 631–651, Sep 1928. [Online]. Available: https://doi.org/10.1007/BF01331938

[23] S. Kim, J. Chen, T. Cheng, A. Gindulyte, J. He, S. He, Q. Li, B. A. Shoemaker, P. A. Thiessen, B. Yu, L. Zaslavsky, J. Zhang, and E. E. Bolton, "PubChem 2023 update," Nucleic Acids Research, vol. 51, no. D1, pp. D1373–D1380, 10 2022. [Online]. Available: https://doi.org/10.1093/nar/gkac956

[24] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for nisq-era quantum devices," in Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ser. ASPLOS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1001–1014. [Online]. Available: https://doi.org/10.1145/3297858.3304023

[25] G. Li, Y. Shi, and A. Javadi-Abhari, "Software-hardware co-optimization for computational chemistry on superconducting quantum processors," in Proceedings of the 48th Annual International Symposium on Computer Architecture, ser. ISCA '21. IEEE Press, 2021, p. 832–845. [Online]. Available: https://doi.org/10.1109/ISCA52012.2021.00070

[26] G. Li, A. Wu, Y. Shi, A. Javadi-Abhari, Y. Ding, and Y. 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, ser. ASPLOS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 554–569. [Online]. Available: https://doi.org/10.1145/3503222.3507715

[27] Y. Liu, S. Che, J. Zhou, Y. Shi, and G. Li, "Fermihedral: On the optimal compilation for fermion-to-qubit encoding," in Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3, ser. ASPLOS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 382–397. [Online]. Available: https://doi.org/10.1145/3620666.3651371

[28] N. D. Matsakis and F. S. Klock II, "The rust language," in ACM SIGAda Ada Letters, vol. 34, no. 3. ACM, 2014, pp. 103–104.

[29] D. McQuarrie, Quantum Chemistry. University Science Books, 2008. [Online]. Available: https://books.google.com/books?id=zzxLTIljQB4C

[30] A. Miller, Z. Zimborás, S. Knecht, S. Maniscalco, and G. García-Pérez, "Bonsai algorithm: Grow your own fermion-to-qubit mappings," PRX Quantum, vol. 4, no. 3, Aug. 2023. [Online]. Available: http://dx.doi.org/10.1103/PRXQuantum.4.030314

[31] A. Molavi, A. Xu, M. Diges, L. Pick, S. Tannu, and A. Albarghouthi, "Qubit Mapping and Routing via MaxSAT," Aug. 2022, arXiv:2208.13679 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2208.13679

[32] P. Murali, J. M. Baker, A. J. Abhari, F. T. Chong, and M. Martonosi, "Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers," 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1901.11054

[33] Y. Nam, N. J. Ross, Y. Su, A. M. Childs, and D. Maslov, "Automated optimization of large quantum circuits with continuous parameters," npj Quantum Information, vol. 4, no. 1, may 2018. [Online]. Available: https://doi.org/10.1038%2Fs41534-018-0072-4

[34] M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press, 2010. [Online]. Available: http://dx.doi.org/10.1017/CBO9780511976667

[35] A. V. Patwardhan, M. J. Cervia, E. Rrapaj, P. Siwach, and A. B. Balantekin, Many-Body Collective Neutrino Oscillations: Recent Developments. Singapore: Springer Nature Singapore, 2020, pp. 1–16. [Online]. Available: https://doi.org/10.1007/978-981-15-8818-1_126-1

[36] W. Pauli, "Über den zusammenhang des abschlusses der elektronengruppen im atom mit der komplexstruktur der spektren," Zeitschrift für Physik, vol. 31, no. 1, pp. 765–783, Feb 1925. [Online]. Available: https://doi.org/10.1007/BF02980631

[37] J. J. Sakurai and J. Napolitano, Modern Quantum Mechanics, 3rd ed. Cambridge University Press, 2020.

[38] K. Setia, S. Bravyi, A. Mezzacapo, and J. D. Whitfield, "Superfast encodings for fermionic quantum simulation," Phys. Rev. Res., vol. 1, p. 033033, Oct 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.1.033033

[39] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, "t—ket⟩: a retargetable compiler for nisq devices," Quantum Science and Technology, vol. 6, no. 1, p. 014003, nov 2020. [Online]. Available: https://dx.doi.org/10.1088/2058-9565/ab8e92

[40] M. Soeken and M. K. Thomsen, "White dots do matter: Rewriting reversible logic circuits," in Proceedings of the 5th International Conference on Reversible Computation, ser. RC'13. Berlin, Heidelberg: Springer-Verlag, 2013, p. 196–208. [Online]. Available: https://doi.org/10.1007/978-3-642-38986-3_16

[41] R. F. Stewart, "Small Gaussian Expansions of Slater-Type Orbitals," The Journal of Chemical Physics, vol. 52, no. 1, pp. 431–438, 01 1970. [Online]. Available: https://doi.org/10.1063/1.1672702

[42] Q. Sun, T. C. Berkelbach, N. S. Blunt, G. H. Booth, S. Guo, Z. Li, J. Liu, J. D. McClain, E. R. Sayfutyarova, S. Sharma, S. Wouters, and G. K.-L. Chan, "Pyscf: the python-based simulations of chemistry framework," WIREs Computational Molecular Science, vol. 8, no. 1, p. e1340, 2018. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wcms.1340

[43] H. F. Trotter, "On the product of semi-groups of operators," Proceedings of the American Mathematical Society, vol. 10, no. 4, pp. 545–551, 1959. [Online]. Available: http://www.jstor.org/stable/2033649

[44] A. M. van de Griend and R. Duncan, "Architecture-aware synthesis of phase polynomials for nisq devices," 2020. [Online]. Available: https://doi.org/10.4204/EPTCS.394.8

[45] E. van den Berg and K. Temme, "Circuit optimization of hamiltonian simulation by simultaneous diagonalization of pauli clusters," Quantum, vol. 4, p. 322, sep 2020. [Online]. Available: https://doi.org/10.22331%2Fq-2020-09-12-322