# Mapping Arbitrarily Sparse Two-body Interactions on One-dimensional Quantum Circuits

Arif Khan[†], Mahantesh Halappanavar[†], Tobias Hagge[†], Karol Kowalski[†],
Alex Pothen[‡], Sriram Krishnamoorthy[†]
[†] Pacific Northwest National Laboratory, Richland, WA, USA
{ariful.khan, hala, tobias.hagge, karol.kowalski, sriram}@pnnl.gov
[‡] Purdue University, West Lafayette, IN, USA; apothen@purdue.edu

***Abstract*—We consider an assignment problem arising in Fermionic-swap based mapping of the one-body and two-body interaction terms in simulating time evolution of a sparse second-quantized electronic structure Hamiltonian on a quantum computer. Relative efficiency of different assignment algorithms depends on the relative costs of performing a swap and computing a Hamiltonian interaction term. Under the assumption that the interaction term cost dominates the computation, we develop an iterative algorithm that uses minimum cost linear assignment (MINLA) and matching for one-body interactions, and hypergraph optimal linear arrangement (HOLA) and partial distance-$2$ coloring for two-body interactions, to exploit arbitrary sparsity in the Hamiltonian for efficient computation. Using a set of $122$ problems from computational chemistry, we demonstrate performance improvements up to 100% relative to the state-of-the-art approach for one-body terms and up to 86% utilization for two-body terms relative to a theoretical peak utilization. To the best of our knowledge, this is the first study to exploit arbitrary sparsity in orbital interactions for efficient computation on one-dimensional qubit connectivity layouts. The proposed algorithms lay a foundation for extension to map general $k$-body interactions that arise in many domains onto generalized qubit connectivity layouts available in current and future quantum systems.**

## I. INTRODUCTION

Molecular simulations play a critical enabling role in domains such as energy-storage materials, efficient catalysis, biomass conversion, and biochemistry. The tools of quantum chemistry and statistical mechanics implemented as advanced parallel computing packages, have proven to be effective and productive for such simulations [35]. Although great effort has been expended in designing numerous methods to describe collective behavior of electrons in correlated systems, the fundamental understanding of these processes is still inhibited by the exponential growth of computational costs associated with representing wave functions on classical computers. In this context, quantum computing as envisioned by Feynman and others can significantly alleviate the situation [10], [31]. Practical quantum simulation of Hamiltonian operator is limited by the depth of the associated quantum circuits. As the technology to build quantum computers improves, one's ability to simulate scientific challenging Hamiltonians is fundamentally limited by the algorithms available to translate the Hamiltonian into an efficient low-depth circuit.

The electronic Hamiltonian represented in a second-quantized form is structured as a collection of terms, each corresponding to interactions involving two or four orbitals. In a typical circuit-level representation, each orbital is mapped to a qubit and each interaction term involving a given list of orbitals is translated into a macro gate (subsequently synthesized into low-level hardware-specific gates) on the corresponding qubits. The compilation procedure needs to account for the underlying limited qubit connectivity.

A key element of optimized mapping is the observation that not all orbital interactions exist in practice. Sparsity is usually a reflection of the nature of inter-particle interactions defining a given quantum system. Sparse Hamiltonian models are prevalent in physics and chemistry. For example, spin Hamiltonians such as Hubbard or Heisenberg models are defined only by interactions between neighboring sites. Also, in chemistry, the dimensionality of the Hamiltonian can be reduced by the localization of molecular orbitals, which enables the development of efficient local approaches [28], [32]. Several specialized tools such as Cholesky decomposition (CD) and singular value decomposition (SVD) have already been employed to simplify the form of the many-body Hamiltonian and to reduce gate depth in the corresponding quantum simulations of the unitary evolution.

The focus of this paper is on improving the efficiency of quantum simulations by *exploiting sparsity* in the Hamiltonian operator in its second-quantized representation, where sparsity is defined as the ratio of actual interactions considered to all possible interactions. If this ratio is one, the problem is dense. In particular, we develop novel heuristic algorithms to minimize interaction depth by maximizing the number of interactions that can be executed concurrently using a minimum linear arrangement (MINLA) and hypergraph optimal linear arrangement problem (HOLA) [3], in conjunction with maximum matching [11] and partial distance-2 coloring [5]. We perform detailed empirical evaluation of this algorithm, and demonstrate its efficacy relative to existing methods. The proposed method is superior to existing work for systems with sparse interactions and can be potentially extended to $k$-body interactions with arbitrary interaction topology for execution on quantum systems with general qubit layouts.

We make the following contributions in this work:

- Formulate the mapping of sparse 1-body interactions onto 1-dimensional qubit layouts iteratively using the minimum linear arrangement problem (MinLA) and maximum matching;

- Formulate the mapping of sparse 2-body interactions onto 1-dimensional qubit layouts iteratively using the hypergraph optimal linear arrangement problem (HOLA) and partial distance-2 coloring; and
- Evaluate the algorithm empirically using a large set of 122 problems with varying degrees of sparsity. We demonstrate up to $\mathbf{86}\%$ utilization relative to a theoretical peak utilization for 2-body interaction problems, and up to $\mathbf{100}\%$ improvement in performance relative to state-of-the-art prior work for 1-body interaction problems.

To the best of our knowledge, this is the first study to explicitly consider arbitrary sparsity patterns in the orbital interactions that are driven by applications, and mapping them efficiently to 1-dimensional qubit layouts.

The rest of the paper is organized as follows: §II introduces the Hamiltonian simulation problem and quantum compilation; §III presents the formulation of the Fermionic swap mapping problem; §IV describes our solution to efficient mapping of arbitrarily sparse second-quantized Hamiltonians; §V and §VI detail our experimental setup and the results, respectively; §VII summarizes related work; and, §VIII concludes the paper.

## II. Background

### A. Simulating Second-quantized Hamiltonian

Consider a molecule with electron spin-orbital interactions, represented by a quantum Hamiltonian. The goal is to simulate evolution of this Hamiltonian (or some related computation involving the Hamiltonian), using a quantum computer. The second-quantization formulation and occupation number representation [9] provide a very efficient way of characterizing many-body effects in the electronic wave function while automatically assuring its anti-symmetry. In the second quantization, all operators and many-body wave function expansion can be represented in terms of creation and annihilation operators $a_p^\dagger$ and $a_p$. Using these operators, one can represent the Hamiltonian operator as:

$$H = \sum_{pq} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{p,q,r,s} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s \qquad (1)$$

For many electronic structure problems, the Hamiltonian may be decomposed with good accuracy into a sum of polynomially many *$k$-local Hamiltonians*, i.e., Hamiltonians involving transfers of at most $k$ electrons between a set of at most $2k$ spin-orbitals. Hamiltonians which are so decomposable are known as *$k$-body Hamiltonians*. These $k$-body Hamiltonians may be evolved, at least approximately, by composing evolutions of the $k$-local Hamiltonians, in any order, via a process known as Trotterization [30], [34]. The time evolution of this Hamiltonian on a quantum computer involves determining a circuit that performs the operation $e^{-iHt}$, for a chosen time $t$ on a given initial quantum state; under Trotterization this is accomplished by evolving the $k$-local Hamiltonians.

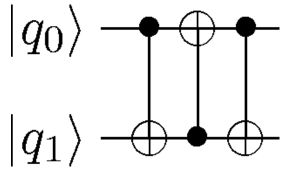

Fig. 1. A quantum circuit illustration a swap operation between two qubits ($|q_0\rangle$ and $|q_1\rangle$) using three CNOT gates, which are more fundamental gates in many architectures.

In typical formulations (e.g., using the Jordan-Wigner transform [2]), this circuit mapping involves: (1) allocating one qubit per spin-orbital, (2) translating each term in $e^{-iHt}$ into a logical gate that preserves Fermion anti-commutation rules. Intuitively, anti-commutation rules are preserved by ensuring a linear ordering of the spin-orbitals (not to be confused with one-dimensional qubit connnectivity), and mapping a term involving spin-orbitals $p$ and $q$ to a logical gate that involves all orbitals between $p$ and $q$ in the second quantization ordering. If the Hamiltonian is second-quantized, there is a natural bijection between the Hamiltonian Hilbert space and a qubit Hilbert space given by using one qubit to represent each spin orbital.

### B. Quantum Circuit Model and Mapping

A quantum circuit is a popular model among the most widely used logical models of computation involving quantum devices. Figure1 illustrates a quantum circuit. Each qubit is represented by a horizontal line, and each operation on one more more qubits is represented by a box. The boxes may correspond to elementary gates natively supported by the hardware technology, or abstract operations that are compiled at a later compilation stage into sequences of elementary gates. The primary challenge in near-term quantum computing devices is the limit on the time duration for which a quantum state can be preserved without interacting with the environment. This motivates the design of low-depth circuits that can best utilize a given quantum device.

In addition to the native gate set, the hardware technology imposes on a constraint on the qubit connectivity; not every qubit can interact directly with every other. In addition, implementing multi-qubit gates is much more expensive than those involving one or two qubits. The limited connectivity and the preference toward one or two-qubit gates gives rise to a scheduled reordering problem; qubits must be shuffled in such a way that the Hamiltonian-local operations can be performed as qubit-local operations. Typically it is desirable to perform as many such operations as possible simultaneously in parallel.

For electronic-structure problems this shuffling requires a sign change when the occupancy states of two occupied spin orbitals are exchanged in order to preserve the electrons' Fermi statistics; the resulting operation is called a *Fermionic swap*. From the perspective of the reordering problem, however, regular and Fermionic swaps are not different.

In this paper, we focus on mapping the interactions in a sparse Hamiltonian into a one-dimensional quantum circuit where the qubits are linearly ordered and quantum gates can only be applied on neighboring qubits. Ideally, the Hamiltonian interaction operations will be mapped to neighboring qubits with a small number of swaps to shuffle them as needed.

## III. Problem Formulation

We define an interaction graph $G = (V, E)$ with vertex set $V$ representing orbitals and edge set $E$ representing interactions between two orbitals corresponding to one-body interactions. Similarly, we define an interaction hypergraph $H = (V_o, E_o)$, where the vertices $V_o$ represent orbitals and hyperedges $E_o$ represent interactions between a set of up to four orbitals (corresponding to two-body interactions). Since the hypergraph representation is a generalization of the graph, we present this section in terms of $H$ for simplicity. We define a *qubit layout* graph $L = (V_q, E_q)$, where the vertices $V_q$ represent qubits and the edges $E_q$ represent quantum gates involving a pair of qubits. Without loss of generality, we assume that the number of orbitals is equal to the number of qubits for formulation as well as empirical evaluation.

Given an interaction hypergraph $H$ and a qubit layout graph $L$, *our goal is to map the edges in $H$ to the edges in $L$.* As illustrated in Figures 2 and 3, our goal is to map orbital interactions to qubit gates for computation. Since not all edges in $H$ can be mapped to edges in $L$ in a single instance due to conflicts arising from shared vertices, we propose an iterative algorithm that maps a *maximal* subset of edges in $H$ to edges in $L$ at each iteration. The overall objective is to: $(i)$ Minimize the cost of mapping, known as the *swap depth* (§IV-A); and $(ii)$ minimize the number of iterations, known as the *interaction depth* (§IV-B). In the following discussion, we provide illustrative examples, first of a 1-body problem and then of a 2-body problem. We provide formal definitions for the costs involved in §IV.

Consider a Nitrogen ($N_2$) molecule with 10 orbitals and the following six 1-body (2-electron) interactions: $\{(1,3),(1,7),(2,6),(2,10),(3,7),(6,10)\}$. Let us consider a simple 1-dimensional qubit layout for this example, which represents a set of $n$ qubits that are physically placed along a line. Each qubit $q_i$ has gates to $q_{i-1}$ and $q_{i+1}$, except the terminal qubits that have only one neighbor (graph $L$ in Figure 2).

Our goal is to map orbital interactions to qubit gates in an efficient manner. For simplicity, we assume that orbital $i$ will be mapped to qubit $i$. For our example, the given ordering is not a feasible solution since no interaction can be computed by the underlying qubit layout directly. For instance, interaction $(1,3)$ needs to be mapped to $q_1$ and $q_3$, but these qubits do not have any gates between them that are necessary to perform the computation. Thus the optimization problem can be formulated as the renumbering (or reordering) of vertices in $H$, where a reordering denotes a specific mapping of interactions to qubits.

Let us now consider a different ordering of the orbitals: $[2,5,1,7,8,4,3,9,10,6]$, where we have renumbered $o_1$ to $o_2$, $o_2$ to $o_5$, $o_3$ to $o_1$, and so on. Consequently the interactions

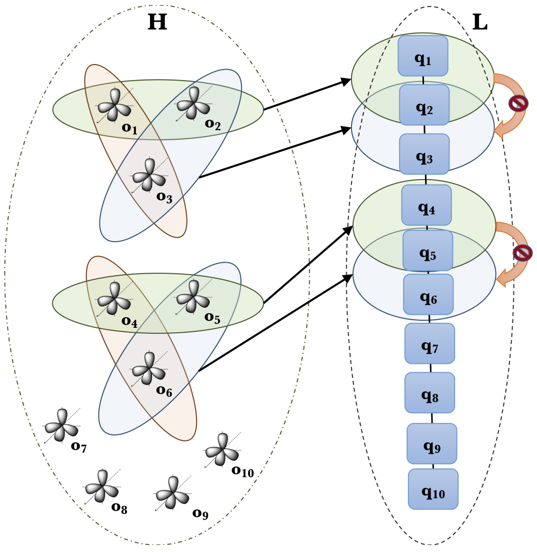

Fig. 2. Illustration of mapping 1-**body** interactions to 1D qubit layout architecture. Note that interactions $\{(\mathbf{2},\mathbf{1}),(\mathbf{2},\mathbf{3})\}$ lead to a conflict if executed concurrently since they share qubit $q_2$.

in $H$ now become: $\{(\mathbf{2},\mathbf{1}),(\mathbf{2},\mathbf{3}),(\mathbf{5},\mathbf{4}),(\mathbf{5},\mathbf{6}),(1,3),(6,4)\}$, where the bold faced interactions that we term as *eligible interactions* are directly computable on the quantum machine. Eligible interactions that share qubits cannot be executed concurrently in a given iteration. Two such conflicts in our example are illustrated in Figure 2, where we can either execute interaction $\{(2,1)\}$ or $\{(2,3)\}$ in an iteration since qubit $q_2$ is common to both the interactions. Similarly, we can either execute $\{(4,5)\}$ or $\{(5,6)\}$ since qubit $q_5$ is common to both. Therefore, given a reordering (mapping), we need to find a mutually exclusive set of interactions that do not share qubits. For instance, we can pick $(\mathbf{2},\mathbf{3})$ and $(\mathbf{5},\mathbf{6})$ for concurrent execution. We can then remove these interactions (edges) from $H$ that leaves us with the following interactions: $\{(\mathbf{2},\mathbf{1}),(\mathbf{5},\mathbf{4}),(1,3),(6,4)\}$. Although, two edges in this set are eligible edges, we ignore them to compute a new ordering $[2,1,3,5,4,6,7,8,9,10]$ with the corresponding edges in $H$: $\{(\mathbf{1},\mathbf{2}),(\mathbf{2},\mathbf{3}),(\mathbf{4},\mathbf{5}),(\mathbf{5},\mathbf{6})\}$, where all edges are eligible edges. However, only a subset of these eligible edges can be executed concurrently, for example, $(\mathbf{1},\mathbf{2})$ and $(\mathbf{4},\mathbf{5})$. We repeat the process until all the interactions are computed. We describe this iterative process in §IV.

In a similar manner, we illustrate 2-body interactions involving four electrons in Figure 3. There are four eligible interactions: $\{(1,2,3,4),(2,3,4,5),(3,4,5,6),(5,6,7,8)\}$. Since these interactions share orbitals/qubits, our best option is to schedule $\{(1,2,3,4),(5,6,7,8)\}$ for concurrent execution to maximize qubit utilization in the first iteration. Since we can use eight of the ten available qubits, we note that we have $80\%$ utilization of the system for the first iteration. In the subsequent iterations, the remaining interactions need to be executed one after another resulting in $40\%$ utilization of the system for the next two iterations. We provide details of utilization in §VI.

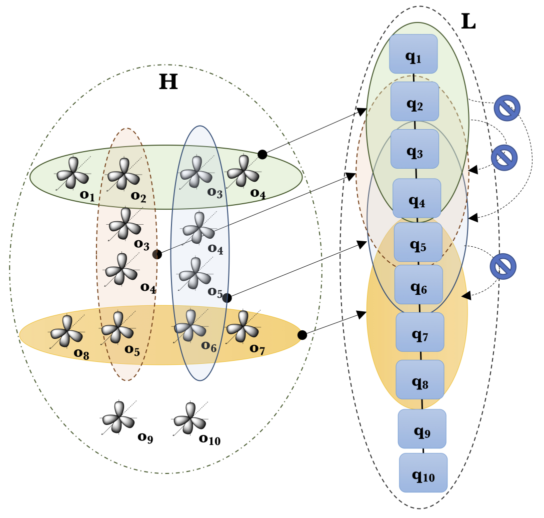

Fig. 3. Illustration of mapping 2-**body** interactions to 1D qubit layout architecture. Note that interactions sharing orbitals (and consequently, qubits) lead to a conflict if executed concurrently. Thus, in the first iteration, only two interactions, $\{(1, 2, 3, 4), (5, 6, 7, 8)\}$, can be executed concurrently. The two remaining interactions can be concurrently executed in the second iteration.

TABLE I
SUMMARY OF PROBLEMS AND ALGORITHMS CONSIDERED IN THIS WORK.

| **Interactions** | **Layout** | **Algorithm** | **Proposed by** |
|---|---|---|---|
| 1-body | 1D | LNN, FSN, ITRMAP | [21], [33], This work |
| 2-body | 1D | ITRMAP | This work |
| k-body | kD | – | Future work |

## IV. Algorithms for Efficient Mapping

In this section, we present an iterative algorithm for mapping 1-body and 2-body interactions to a one-dimensional qubit layout (§III). We first provide an overview of the iterative algorithm, and then provide details of two key steps of the algorithm, the first step to reorder the interactions to minimize the swap depth, and the second step to find a maximal set of eligible interactions for concurrent execution. We summarize the types of interactions and proposed solutions for mapping in Table I.

While Algorithm ITRMAP-1B takes the graph $G$ representing 1-body orbital interactions as input, Algorithm ITRMAP-2B takes the hypergraph $H$ representing 2-body orbital interactions as its input. Both algorithms take the graph $L$ representing qubit layout as a second input, and generate intermediate data structures, $G'$ and $H'$, to store eligible edges in $G$ and $H$, respectively. A maximally independent set of edges in $G'$ and $H'$ are then selected for execution at a given iteration. It is to be noted that for 1D layout, the graph L is simply a line graph. The three main tasks performed at each iteration are:

1) **Reorder:** To compute an ordering to align edges in $G$ (or, hyperedges in $H$) with corresponding edges in $L$ using the shortest path distance measure $d(e, L)$ (or, $d(h, L)$) as a metric. An ordering is computed using Algorithm MINLA (or, Algorithm HOLA) (Line 2). The permutation $\pi$ returned by MINLA (or, HOLA) is used to renumber the vertices in graph $G$ (or hypergraph $H$) (Line 3).
2) **Select:** Based on the reordered graph $G$ (or hypergraph $H$), select all the interactions that can be computed on $L$, denoted by $E_l$ (Line 4). An intermediate graph $G'$ (or, hypergraph $H'$) is then constructed to represent the subgraph induced by the eligible edges (Line 5). For Algorithm ITRMAP-1B, a maximum matching $M$ is computed on $G'$. The edges in $M$ represent all the edges that can concurrently scheduled for execution for the given iteration (Line 6 in Algorithm 1). However, For Algorithm ITRMAP-1B, the induced subgraph $H'$ is stored as a bipartite graph $B$ (Line 6 in Algorithm 2). A partial distance-2 coloring of $B$ is computed using Algorithm PARTIALDIST2COLORING to identify a maximal set of eligible (hyper)edges that can be scheduled for concurrent execution (Line 7).
3) **Repeat:** The final step is to remove all the interactions (edges in $G$ or hyperedges in $H$) that have been executed. At this step, we can either process all the remaining eligible edges or reorder the remaining edges once again. Empirically, we observed that reordering the remaining edges in $G$ (or, $H$) provides better performance for the inputs used in the study (results are provided in § VI).

**Algorithm 1 ITRMAP-1B**: Iterative algorithm to map and execute 1-body interactions on 1-dimensional layouts
**Input**: Interaction graph $G$ and Layout graph $L$

```
1: while ($G \neq \emptyset$) do
2:     $\pi \leftarrow$ MINLA($G$)                ▷ Compute reordering
3:     $G \leftarrow \pi(G)$                      ▷ Reorder interactions
4:     $E_l \leftarrow$ Eligible edges ($G$)
5:     $G' \leftarrow$ Induced subgraph ($G, E_l$)
6:     $M \leftarrow$ MAXIMUMMATCHING($G'$)
7:     $G \leftarrow G \setminus M$               ▷ Remove completed edges
```

**Algorithm 2 ITRMAP-2B**: Iterative algorithm to map and execute 2-body interactions on 1-dimensional layouts
**Input**: Interaction hypergraph $H$ and Layout graph $L$

```
1: while ($H \neq \emptyset$) do
2:     $\pi \leftarrow$ HOLA($H$)                 ▷ Compute reordering
3:     $H \leftarrow \pi(H)$                      ▷ Reorder interactions
4:     $E_l \leftarrow$ Eligible edges ($H$)
5:     $H' \leftarrow$ Induced subgraph ($H, E_l$)
6:     $B \leftarrow$ Bipartite ($H'$)
7:     $M \leftarrow$ PARTIALDIST2COLORING($B$)
8:     $H \leftarrow H \setminus M$               ▷ Remove completed edges
```

In the rest of this section, we discuss the metrics and approaches to minimize the swap depth and interaction depth. For simplicity, we will provide the definitions for the 2-body problem using hypergraphs and note that the definitions can be trivially extended to the 1-body problem where all the edges in the hypergraph consists of only two vertices (edges in a graph). The problem can be alternatively formulated as a bi-objective optimization problem that simultaneously optimizes for swap depth and interaction depth. However, considering the overall simplicity and scalability of the proposed two-step approach, our goal is to develop a computationally efficient heuristic that can also provide high quality solutions for efficient mapping.

## *A. Minimizing Swap Depth*

In this section, we will introduce a few concepts before formally defining swap depth. We define a layout function $\phi(.)$ that maps the hyperedges in $H$ to a non-negative number defined with respect to the graph $L$ as follows:

$$\phi(h) = \begin{cases} 0 & \text{if } h \in E_L \\ d(h, L) & \text{otherwise,} \end{cases} \tag{2}$$

where $E_L$ represents the set of hyperedges in $L$, and the function $d(h, L)$ is the distance (in the graph $L$) between the minimum-numbered and the maximum-numbered vertices in a hyperedge $h = (i_1, \ldots, i_k)$ in $H$. Assuming $u = \min(h) = \min\{i_1, \ldots, i_k\}$ and $v = \max(h) = \max\{i_1, \ldots, i_k\}$,

$$d(h, L) = \delta_L(u, v) - 1, \tag{3}$$

where $\delta_L(x, y)$ is the length of a shortest path joining vertices $x$ and $y$ in $L$.

Current architectural limitations enforce the constraint that once orbitals are assigned to qubits, we can move orbital information to other qubits only by swaps involving adjacent qubits. For instance if $q_i$ has information of an arbitrary orbital and we want to move the information to $q_{i+2}$ then we need to first swap the information between $q_i$ and $q_{i+1}$ followed by a second swap from $q_{i+1}$ to $q_{i+2}$. Since each reordering is essentially moving the orbital information around, it is associated with a cost. The swaps are performed by qubit gates, and the cost of reordering is *the number of swaps* required to change one sequence of interactions to another, also known as the *Kendall-Tau* distance or the *Bubble Sort* distance [15]. Formally, we define swap depth as follows. Given two ranked lists $o$ and $l$, swap depth:

$$\begin{aligned} S_d(o, l) = |\{(i, j) : i < j,\\ ((o(i) < o(j)) \text{ AND } (l(i) > l(j))) \text{ OR}\\ ((o(i) > o(j)) \text{ AND } (l(i) < l(j)))\}|, \end{aligned} \tag{4}$$

where $l(i)$ represents the rank of $i$ in $l$. In other words, the metric counts the number of disagreements between the relative ranking of two elements in the list. Since disjoint swaps can be performed independently on a quantum system, we adapt the definition of swap depth such that we count it only once for all the swaps that can be performed concurrently at a given step. Thus, swap depth reduces to the number of steps in the odd-even sort variant of the Bubble Sort algorithm [22].

Since our goal is to minimize the *swap depth*, the intuition is to find an ordering, $\pi$, that aligns $H$ as best as possible to $L$ to reduce the number of remaps of edges in $H$, and thereby, the total cost of swapping. For an arbitrary qubit layout, the ordering $\pi$ is defined as follows:

$$\pi = \arg\min_{\pi'} \sum_{h \in H} d(\pi'(h), L), \tag{5}$$

where the permutation $\pi'$ ranges over all possible orderings.

It is to be noted that for 1-dimensional layout, the graph L reduces to a line graph. For 1-body interactions with 1-dimensional layout, a reordering that minimizes swap depth can be formulated as a *Minimum Linear Arrangement* (MINLA) problem [18], [23], where for an edge $h = (i, j)$ and a layout function $\phi(h) = |i - j|$, an ordering $\pi_m$ is defined as follows:

$$\pi_m = \arg\min_{\pi'} \sum_{(i,j) \in H} |\pi'(i) - \pi'(j)|. \tag{6}$$

MINLA is known to be NP-hard [17], and techniques such as dynamic programming and mixed-integer linear programming based approaches to solve the problem optimally require exponential time [29]. Heuristic approaches based on spectral sequencing and simulated annealing have been proposed in literature. Empirical evaluations demonstrate that while spectral sequencing based methods are faster, their quality is inferior to those computed using simulated annealing based methods [29]. We therefore implement a simulated annealing based algorithm of Nahar *et al.* as illustrated in Algorithm 3 [26].

**Algorithm 3 MINLA**: Algorithm to compute a minimum linear arrangement based permutation
**Input**: A graph $G = (V, E)$, and a parameter for maximum number of iterations ($maxcount$)
**Output**: An ordering $\pi_{best}$ of $G$

```
1:  $\pi_{best} \leftarrow \emptyset$
2:  $\pi \leftarrow$ random initial ordering
3:  $N \leftarrow$ number of vertices in $G$
4:  $counter \leftarrow 0$
5:  while $counter < maxcount$ do
6:      $i = random(1, N-1)$
7:      $\pi \leftarrow exchange(\pi[i], \pi[i+1])$
8:      if $cost(\pi) < cost(\pi_{best})$ then
9:          $\pi_{best} \leftarrow \pi$
10:         $counter \leftarrow 0$
11:     else
12:         $counter \leftarrow counter + 1$
13: return $\pi_{best}$
```

Algorithm 3 starts with an initial random ordering $\pi$ (Line 2). At each iteration the algorithm randomly picks an element of the current ordering and exchanges it with the next element to generate a new ordering (Line 7). If the new ordering has a lower cost than the best ordering observed so far, then the best ordering is updated with this solution (Line 9). Otherwise, the algorithm produces another ordering through random exchanges. If no better ordering is found after a specified number of attempts (`maxcount`), the algorithm returns the best ordering that it has observed so far, $\pi_{best}$ (Line 13).

**Algorithm 4 HOLA**: Compute a Hypergraph Optimal Linear Arrangement (HOLA) of a hypergraph
**Input**: 2-body interaction yypergraph $H$, and a parameter for maximum number of iterations ($maxcount$)
**Output**: An ordering $\pi$ of $H$

```
1: $G \leftarrow$ Reduced form of $H$          ▷ Hyperedges to paths
2: $\pi \leftarrow$ MinLa($G, maxcount$)
3: return $\pi$
```

Let us now consider the problem of mapping 2-body interactions to a 1-dimensional layout. A reordering that minimizes swap depth can be formulated as a *Hypergraph Optimal Linear Arrangement* (HOLA) problem [3], [16]. Each hyperedge $h = \{i_1, i_2, \ldots, i_k\}$ is sorted from the smallest to largest vertex id, i.e., $i_1 \leq i_2 \leq \ldots \leq i_k$. The layout function is $\phi(h) = |i_1 - i_4|$, and the ordering $\pi_h$ is defined as follows:

$$\pi_h = \arg\min_{\pi'} \sum_{(i,j)\in H} |\max(\{pi'(h)\} - \min\{\pi'(h)\}|. \quad (7)$$

Jin *et al.* demonstrate that an efficient approach to solve HOLA is to first transform a hypergraph to a graph, and then solve MinLa on the reduced graph [16]. The reduced graph $G$ has the same number of vertices as the original hypergraph $H$. Then an ordered path $P(i_1, i_2 \ldots i_k)$ is created in $G$ for each hyperedge $h \in H$. Jin *et al.* prove that this reduction leads to an equivalent result, in a sense that the objective cost of solving the original hypergraph reordering problem directly will be equal to cost of solving the reduced graph reordering problem using MinLa [16]. The worst-case time complexity of the reduction is $O(|V_0||E_0|)$, where $|V_0|$ is the number of vertices and $|E_0|$ is the number of hyperedges in $H$.

Algorithm 4 first reduces the edges in the input hypergraph $H$ to paths in the graph $G$ (Line 1), and then computes the best ordering by a call to Algorithm MinLa on the graph $G$ (Line 2).

Reordering of $H$ results in a set of eligible edges that can be mapped for execution on a 1-dimensional qubit circuit. The next step is to identify an optimal number of disjoint edges (interactions) that can be executed concurrently, which will result in the minimization of the interaction depth.

## B. Minimizing Interaction Depth

Let us consider the example scenario of 1-body interactions presented in §III. We observe that after reordering in the first iteration, four out of six interactions become eligible for execution. However, not all of these six interactions can be scheduled for concurrent execution. Since our goal is to minimize the *interaction depth*, i.e., to minimize the number of iterations to complete the execution of all the interactions, we need to identify a maximal set of interactions that can be executed concurrently in each iteration. Since we consider eligible edges for a given ordering, computed with the primary goal of minimizing the swap cost (using Algorithm 4), we might not be able to identify the maximum number of interactions that might be available for concurrent execution during a given iteration. However, given a set of eligible edges, we can optimize to find a maximum number of disjoint edges that can be executed concurrently by using a maximum matching. A matching $M$ in a graph is a subset of edges such that no two edges in $M$ are incident on the same vertex [11].

For 1-body problems, the lower bound on interaction depth is the minimum number of colors needed to color the edges of the interaction graph $G$ (§III). Given a graph, the problem of assigning colors to edges such that adjacent edges receive distinct colors is the edge coloring problem [4]. The minimum number of colors in an edge coloring is the chromatic index of a graph. Since any pair of interactions that share an orbital cannot be executed in the same iteration (equivalently edges that share an endpoint cannot be assigned the same color), the chromatic index represents the fewest iterations necessary to execute all the interactions (the interaction depth). The chromatic index of a graph is either the maximum vertex degree or this degree plus one.

The first step towards this computation is the construction of a bipartite graph $G = (S \cup T, E)$ from the reordered graph $H'$, as indicated in Algorithm ItrMap-2B (Line 6). The vertex set $V(G) = S \cup T$ is formed from the vertices and edges in $H'$. We reduce $H'$ to $G$ as follows: For each unique vertex $v_i \in H'$, we add a vertex to $S(G)$, $S \leftarrow S \cup \{v_i\}$. For each hyperedge $e_i \in H'$, we add a unique vertex to $T(G)$, $T \leftarrow T \cup \{t_i\}$ representing edge $e_i$. For each vertex in the hyperedge $e_i$, we add a corresponding edge $(s_i, t_i)$ to $E(G)$.

In order to identify the maximum number of disjoint edges for concurrent execution from an eligible set, we compute a *partial distance-2 coloring* in $G$, where we only color the vertices in $T$. Given a graph $G = (V, E)$, a *distance-$k$* coloring assigns unique identities (colors) to each vertex such that no two vertices at a distance $k$ from each other receive the same color. A *partial distance-2 coloring* of $G$ is an assignment of colors to every vertex in $T$ such that no two vertices in $T$ that share a neighbor in $S$ are assigned the same color. Therefore, we can observe that two hyperedges in $H'$ that share a vertex (orbital/qubit) would receive distinct colors.

In order to find a maximal set of hyperedges that can be scheduled for concurrent execution, we pick the color class (color id) with the maximum number of $T$-vertices (hyperedges). Since any two hyperedges in a color class are guaranteed to be disjoint (do not share a vertex), they will not generate conflicts for execution. Since graph coloring is known to be NP-hard [5], we adapt the heuristics proposed by Çatalyürek *et al.* for computing partial distance-2 coloring in parallel [5], as presented in Algorithm 5.

Algorithm 5 maintains a data structure, `forbidden`, for each vertex in $T$ to mark all the forbidden colors that have already been used by its distance-2 neighbors (Lines 4 to 7), and then picks the minimum available color for $t \in T$ (Line 8). Since the algorithm processes vertices in parallel, there could be conflicts that are created in the concurrent coloring phase, and need to be resolved. The conflicts are detected in the second phase (Lines 11 to 15). The algorithm iterates until all the vertices in $T$ are colored.

**Algorithm 5 PARTIALDIST2COLORING**: Compute a partial distance-2 coloring
**Input**: A bipartite graph $G = (S \cup T, E)$
**Output**: Color assignment `color` for each vertex in $T$

```
1: color← $\emptyset$; forbidden← $\emptyset$;
2: $Q \leftarrow T$
3: while $Q \neq \emptyset$ do
4:     for each $t \in Q$ in parallel do
5:         for each $s \in adj(t)$ do
6:             for each $\hat{t} \in adj(s)$ do
7:                 forbidden[color[$\hat{t}$]] $\leftarrow t$
8:         $c \leftarrow \min\{i > 0$: forbidden[color[$i$]] $\neq t\}$
9:         color[$t$]← $c$
10:    $R \leftarrow \emptyset$
11:    for each $t \in Q$ in parallel do
12:        for each $s \in adj(t)$ do
13:            for each $\hat{t} \in adj(s)$ do
14:                if color[$t$]=color[$\hat{t}$] and $t > \hat{t}$ then
15:                    $R \leftarrow R \cup \{t\}$
16:    $Q \leftarrow R$
17: return $M$
```

An alternative way to formulate the computation of a maximal set of disjoint hyperedges is the *maximum set packing* (MSP) problem, which is one of Karp's 21 NP-complete problems [19]. Given a finite set $\mathbf{S}$ of $n$ elements and a list of subsets of $\mathbf{S}$, the MSP problem finds a collection $M$ of subsets of maximum cardinality that cover $n$ elements with the constraint that the subsets in $M$ are pairwise mutually disjoint. In our case, the qubits are the elements and the eligible interactions (hyperedges) form the list of subsets. We use a greedy heuristic algorithm [12] in our current implementation that finds a maximal set of mutually exclusive set of interactions instead of the maximum number of interactions.

We empirically evaluated the performance of Algorithm ITRMAP using a large set of problems from quantum chemistry. We present the evaluation results in the next section.

## V. EXPERIMENTAL SETUP

Our test set consists of 122 problems from quantum chemistry with varying degrees of sparsity and generalizations. We evaluated the proposed algorithms on all the 122 problems for 1-body and 2-body interactions. We present detailed results in this section on ten representative problems for 2-body interactions. These problems are chosen with varying degree of molecular complexity described next; some statistics on these problems are summarized in Table II. Detailed information on the entire dataset is also available for download [1].

As benchmarks to illustrate the performance of new algorithms we have chosen several molecular systems that epitomize basis problems encountered in studies of chemical processes. In particular, the main emphasis is on establishing dependencies between the gate depth and $(i)$ strength of the correlation effects, $(ii)$ size of the active space (i.e., the number of correlated orbitals), and $(iii)$ the number of correlated electrons. Excellent benchmarks to study these problems are:

- H4 system, where the quasi-degeneracy of the ground-state wave function can be controlled by a single geometrical parameter $\alpha$ (for $\alpha = 0.0$ the four hydrogen atoms constitute a square geometry, for $\alpha = 0.5$ H4 assumes a linear configuration);
- H4 dimer to study the separation processes in molecular systems;
- potential energy curve of the LiH system;
- breaking the triple bond in the $N_2$ system;
- studies of ozone molecule opening relevant to studies of conical intersections; and
- $C_{20}$ (bowl, fullerene, and ring configurations), *beta*-carotene, and cytosine systems for various choices of active spaces and the number of correlated electrons.

We also considered typical systems used in studies of many-body techniques: beryllium, boron, and argon atoms, $BeH_2$, $F_2$, and $CH_4$ systems, and stretching bond in the water molecule. To illustrate the effect of system size growth we used linear chain of the $H_2$ molecules – $(H_2)_n$ for $n = 2, \ldots, 12$. We also employed various types of basis sets: STO-3G (LiH, $H_2O$, $N_2$) [14], STO-6G (H4 system, H4 dimer, and $(H_2)_n$) [14], 6-31G (Be, B, $BeH_2$, $F_2$, cytosine) [14], 6-311G ($\beta$-carotene) [24], cc-pVDZ ($H_2O$, $C_{20}$, Ar, Be) [8], cc-pVTZ ($O_3$) [8].

We quantify the benefits of exploiting arbitrary sparsity using two metrics: *swap depth* $C_r$, and *interaction depth* $C_d$ in the context of the quantum molecular simulation, the chemistry application that we are considering. Both these metrics are important and define the complexity of a quantum circuit. However, we focus on *interaction depth* which has direct impact on quality and accuracy of the quantum molecular simulation. Since simulated annealing is a probabilistic technique, we repeat each experiment $10\times$ and report the median value for each metric. We implement our algorithms using Python $(v3.7)$. The code is parallelized using Python *multiprocessor* package. We perform the experiments on an 80-core Intel Xeon $E7-8860$ system with processor speed of 2.27 GHz and equipped with 1 TB of primary memory.

## VI. EXPERIMENTAL RESULTS

We provide results from empirical evaluation of the proposed algorithm in this section. We evaluate the efficiency of the proposed methods using two metrics – swap depth (swap cost) and interaction depth (interaction cost) – that were defined in §IV. We also provide relative performance of the proposed algorithm with previous work for 1-body interaction problems. Since no prior work exists for 2-body interactions, we provide insight into the utilization of a circuit relative to an ideal situation where all the qubits are used.

### A. *Relative Performance for 1-body Interaction Problems*

In order to demonstrate the efficacy of the proposed methods, we perform relative comparison of our approach with

[1] https://hpc.pnl.gov/people/hala/files/hipc_2_1D_full.xlsx

| Problem | # Orbitals | # Interaction | # Unique Interactions | Swap depth | Interaction depth | Run time (Sec) |
|---|---|---|---|---|---|---|
| **h2o_oh3.0_sto3g** | 7 | 274 | 71 | 154 | 39 | 1.1 |
| **be_6-31g_fci** | 9 | 417 | 136 | 300 | 62 | 1.82 |
| **n2_4_00Re_sto3g** | 10 | 364 | 143 | 282 | 55 | 2.13 |
| **ch4_sto6g_fci** | 9 | 837 | 222 | 481 | 100 | 8.21 |
| **beh2_6-31g_fci** | 13 | 1050 | 397 | 789 | 123 | 11.98 |
| **o3_13_6_6_110deg_ccpvtz** | 13 | 2028 | 700 | 1429 | 219 | 21.87 |
| **cytosine_6-31g** | 13 | 4186 | 1079 | 2293 | 352 | 48.10 |
| **beta_carotene_6_311G** | 16 | 4708 | 1604 | 3173 | 404 | 97.90 |
| **h2_11_sto6g_1.0au** | 22 | 16159 | 5456 | 10306 | 1012 | 554.71 |
| **c20_bowl_ccpvdz_22_9_9** | 22 | 27245 | 8324 | 16356 | 1605 | 703.62 |

TABLE II
SUMMARY OF TEN 2-BODY INTERACTION PROBLEMS REPRESENTING DIFFERENT CHEMICAL PROCESSES. THESE TEN PROBLEMS ARE CHOSEN FROM A LARGER SET OF 122 PROBLEMS USED FOR EVALUATION.

state-of-the-art work of Kivlichan *et al.* for dense problems [21]. Relative performance of ITRMAP to FSN using a set of 122 problems is captured in Figure 4. Performance improvements are presented as percentages relative to FSN ($\frac{(T_{FSN}-T_{ITRMAP})}{T_{FSN}}$), and are plotted along the Y-axis. Density of an input is plotted along the X-axis. A problem with $100\%$ density means that all pairs of orbital interactions are considered. Since FSN was designed for dense interactions, we expect FSN to perform better than ITRMAP for denser problems. The scatter plot in Figure 4 shows a strong (negative) correlation between density and swap depth (with a correlation coefficient of $-0.77$), and between density and interaction depth (correlation coefficient of $-0.82$). Thus, the sparser the problem, the higher is the gain in performance for ITRMAP relative to FSN. We also observe significant performance gains for problems that are more than $50\%$ sparse. The use of heuristics to solve MINLA leads to a loss in performance for denser problems (density $\geq 30\%$) with respect to the swap depth. However, we do not observe loss in the performance for interaction depth for the same set of dense problems. Although, we test the performance of ITRMAP for a wide range of problems, we expect stronger performance for complex sparsity structures and qubit layouts beyond simple 1-D structures.

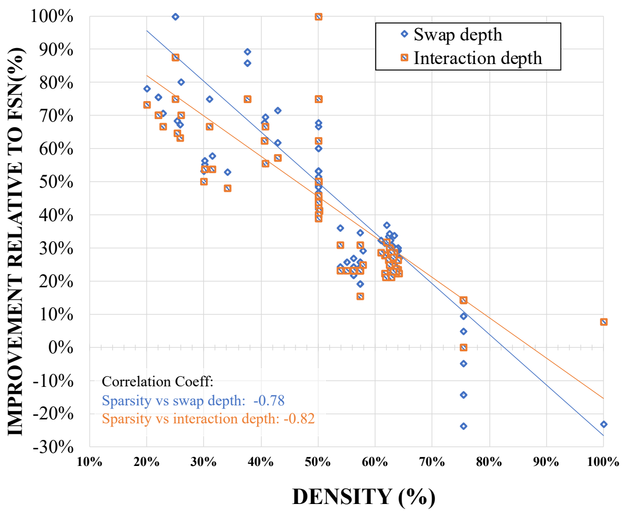

Fig. 4. Performance improvements of ITRMAP relative to FSN by exploiting sparsity. Density of interactions is plotted along X-axis and the benefits in swap depth and interaction depth are plotted along Y-axis (as percentage improvements). Bottom right of the figure represents dense problems with loss in performance for ITRMAP.

## B. *Performance for* 2*-body Interaction Problems*

We compute the utilization, $u_p$ as the ratio of the number of qubits in $M$ (the interactions that could be computed at this iteration) over the total number of qubits for a given iteration $p$. The variable $u_p$ indicates what percent of the total qubits are being used in a given iteration. We define the *qubit utilization* $u_q$ as the geometric mean of $u_p$ over all iterations $p$. We argue that if $u_q$ is (high) then the interaction depth is optimal or close to optimal. This is so because an optimal algorithm will assign a maximum number of concurrent eligible edges (i.e., maximize $u_p$) in order to minimize interaction depth. We show in Figure 5 that the geometric mean of qubit utilization across all the iterations for Algorithm ITRMAP is at least $70\%$ for all the test problems. We also show that $u_q$ increases almost linearly with the number of interactions (problem size), and utilization increases to about $90\%$ for the larger problems. Intuitively, as the number of interactions increases the algorithm has more choices for finding concurrent eligible edges at each iteration, and therefore, it is easier for a heuristic algorithm to find $M$ with higher values of $u_p$. Broadly, we observe that the proposed method is capable of computing high quality solutions that are closer to optimal solutions.

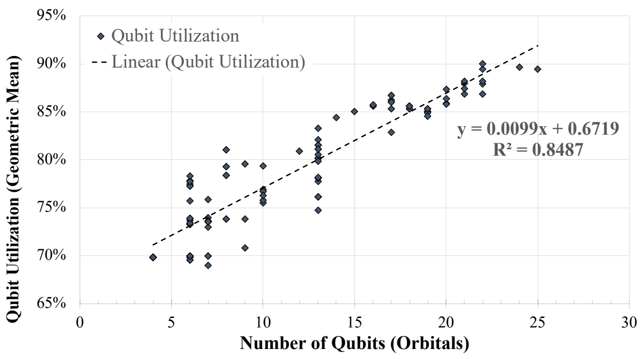

Fig. 5. Geometric mean of qubit utilization across all the iterations for a given problem is plotted along the Y-axis. Problems plotted along the X-axis are ordered by the number of orbitals. A linear trend line is shown using dashed black line.

We also plot the per-iteration qubit utilization $u_p$ in Figures 6, 7 and 8. We observe that initial iterations have high utilization; as the algorithms progress there are fewer interactions available for qubit assignment, and $u_p$ goes down.

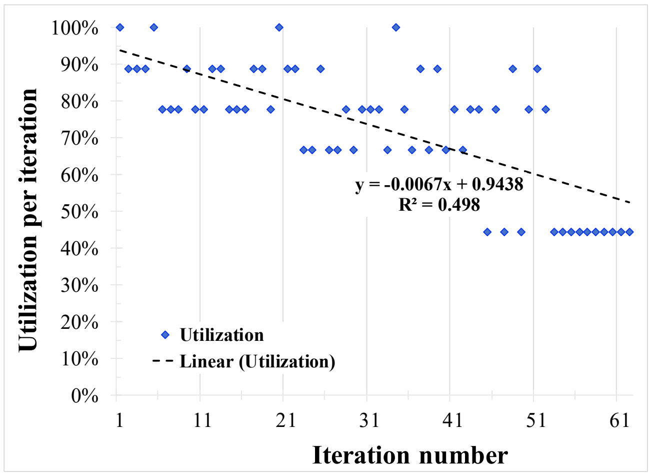

Fig. 6. Utilization per iteration for problem `be_6-31g_fci` with 62 iterations. We observe a mean of 73.12 and standard deviation of 17.26 across all the values.

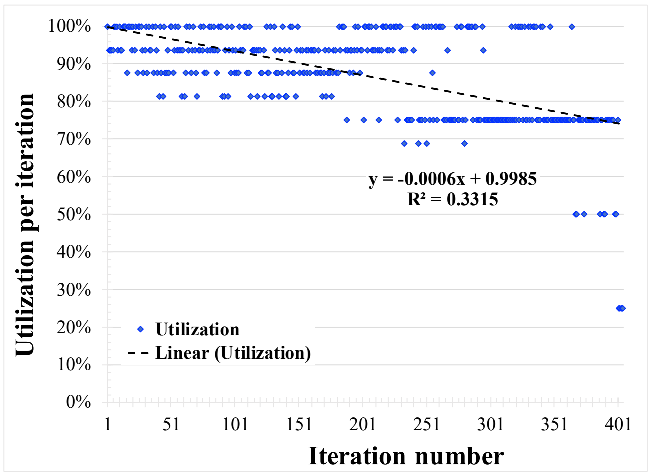

Fig. 7. Utilization per iteration for problem `beta_carotene_6_311G` with 404 iterations. We observe a mean of 86.88 and standard deviation of 13.00 across all the values.

We show the relationship between the problem size (the number of orbitals), and the interaction depth and swap depth in Figure VI-B. We observe that both the swap depth and interaction depth increase linearly with the number of electron orbitals (with high confidence). We note that the number of possible $k$-body interactions with $n$ orbitals is $O(n^k)$. However, our algorithm successfully exploits the sparsity structure and solves the problem in linear depths with the increase of $n$. We believe that this relationship will also hold for higher $k$-body interaction cases.

Finally, we report the run times for each problem in Table II. We implemented the pipeline using Python $v3.7$. The most time consuming computation of the pipeline is cost evaluation of an ordering (line 8 in Algorithm 3). Since this step is embarrassingly parallel, we employed the Python *multiprocessing* package to parallelize the evaluation.

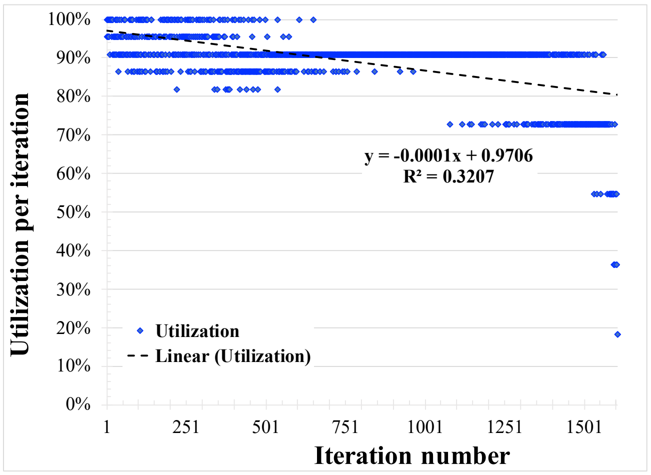

Fig. 8. Utilization per iteration for problem `c20_bowl_ccpvdz_22_9_9` with 1605 iterations. We observe a mean of 88.16 and standard deviation of 8.52 across all the values.

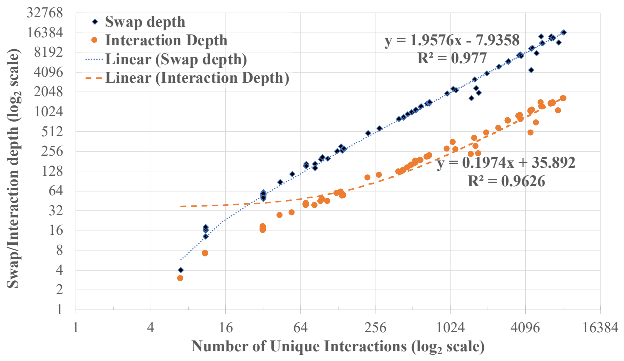

Fig. 9. Depth versus Number of Interactions: Swap depth (blue diamonds) and Interaction depth (orange circles) are plotted along Y-axis for different numbers of unique interactions (X-axis). Both the axis are $\log_2$ scale. Linear trend lines are shown in blue dotted and orange dashed lines for swap depth and interaction depth respectively.

## VII. Related Work

Mapping dense (all-pairs) 1-body interactions for execution on simple 1-dimensional or very simple qubit layouts have been explored in literature. Two closely related works are the following. Kivlichan *et al.* present the FSN Algorithm designed to efficiently compute all-pairs of orbital interactions [21]. The algorithm is optimal in the sense that there is no swapping method which brings all pairs of qubits to adjacency which has asymptotically shorter reordering cost or circuit depth.

For a given number of orbitals $n$, FSN has a reordering cost of $C_r = n(n-1)/2$ with an interaction depth of $C_d = n$. While the interaction depth is linear in the number of the orbitals, the quadratic cost for reordering becomes a limitation if the underlying interactions are sparse; even if circuit depth is not increased the nontrivial operations can be expected to contribute to circuit noise. We provide relative performance of our approach with FSN in §VI to demonstrate the benefits of exploiting sparsity. Although the LNN Algorithm of Shafaei *et al.* improves the reordering cost by taking advantage of sparsity patterns, it performs global reordering only once [33].

We have observed from our experiments that performing reordering only once fails to solve the problem completely when additional swaps become necessary. Maximizing concurrent execution of eligible interactions is not considered in LNN.

For the $k$-body dense interaction case, [27] provides a recursive algorithm for producing asymptotically swap-depth optimal networks in which the interaction terms are executed as $2k$-qubit local operators. The required swap depth is $O(n^{2k-1})$, with the $k = 1$ case reducing to the method of [21]. For general $k$, unlike the $k = 1$ case, the method seems unlikely to be reordering-optimal for dense interactions.

For an in-depth discussion of Hamiltonian sparsity issues as they relate to quantum computing, see [25].

The decomposition of a Hamiltonian into local Hamiltonian operators is usually followed by an evolution of that Hamiltonian expressed in terms of local Hamiltonian operators. The cost metrics we have defined are most relevant when the local operators are evaluated sequentially and that the order of the operators doesn't matter. This is the case when Hamiltonians are evolved via Trotterization [30], [34]. Other Hamiltonian evolution algorithms, such as linear combinations of unitaries [7] and qubitization [13], require a separate performance analysis beyond the scope of this paper.

Our work focuses on electron interactions in second quantized Hamiltonians. In this case, electron orbitals have a localized qubit representation. There are also first-quantized approaches to electronic structure calculation (see e.g. [20]), in which each electron has a localized multi-qubit representation which encodes the orbital-occupancy state of that electron. For problems with small, fixed electron number and a large number of orbitals, it may be more efficient to use a first-quantized representation. In this case, the Hamiltonian will require interactions between all of the electrons and one obtains the dense interaction problem but with orbital-occupancy-qubits replaced by multi-qubit localized storage units. The sparse interaction case does not occur for first-quantized electronic structure Hamiltonians, as it is occupancy rather than identity which prevents electrons from interacting.

For some problems, it is possible to work in another Hamiltonian basis, such as the plane wave basis [1] in order to reduce the number of Hamiltonian interaction terms.

The order in which local Hamiltonians are evolved does affect the accuracy of a Trotter decomposition, but it is infeasible to determine which order is best. Empirically, it has been shown that reshuffling the order of local Hamiltonians between Trotter steps can improve the accuracy of a Hamiltonian evolution [6]. Since our method is heuristic, some variant of this approach is possible, but its performance would need to be tested empirically.

## VIII. Conclusions and Future Work

We presented a novel formulation for explicitly considering arbitrary sparsity in one-body and two-body interactions for execution on one-dimensional qubit layouts using hypergraph optimal linear arrangements and partial distance-2 coloring. We demonstrated up to $100\%$ improvement while measuring two key metrics for performance relative to the state-of-the-art solutions for one-body problems, and up to $86\%$ improvement in utilization relative to a theoretical peak utilization for two-body problems.

Explicit consideration of sparsity is not only reflective of real-world molecular simulations, but also has significant impact on performance. We therefore plan to extend our work to include $k$-body interactions on general $m$-dimensional qubit layout architectures. We observe that the hypergraph based formulation presented in this paper can be used to formulate $k$-body interactions for an arbitrary value of $k \geq 1$. However, extending the qubit layout from 1-dimension to $m$-dimensions, for $m \geq 2$, will necessitate a major change in the mapping of eligible edges to qubits. Further, qubit layouts can also feature arbitrary connection topologies. We plan to extend our work to address this combinatorial problem in our future work.

To the best of our knowledge, this is the first work to exploit arbitrary sparsity of orbital interactions. We therefore believe that this work will be of interest to a broad set of researchers and practitioners looking to exploit the potential of quantum computing for molecular simulations.

## Acknowledgment

This work is funded by QUASAR, a Laboratory Funded Research and Development (LDRD) Initiative at the Pacific Northwest National Laboratory, and by a grant from the Integrated Data Science Initiative of Discovery Park at Purdue University. We thank the anonymous reviewers for providing critical feedback to improve the manuscript.

## References

[1] R. Babbush, N. Wiebe, J. McClean, J. McClain, H. Neven, and G. Kin-Lic Chan. Low Depth Quantum Simulation of Electronic Structure. *ArXiv e-prints*, May 2017.
[2] C. Batista and G. Ortiz. Generalized Jordan-Wigner transformations. *Physical Review Letters*, 86(6):1082, 2001.
[3] J. Bhasker and S. Sahni. Optimal linear arrangement of circuit components. *J. VLSI COMP. SYST.*, 2(1):87–109, 1987.
[4] J. Bondy and U. Murty. *Graph Theory*. Springer Publishing Company, Incorporated, 1st edition, 2008.
[5] U. V. Catalyurek, J. Feo, A. H. Gebremedhin, M. Halappanavar, and A. Pothen. Graph coloring algorithms for multi-core and massively multithreaded architectures. *Parallel Computing*, 38(10):576 – 594, 2012.
[6] A. M. Childs, A. Ostrander, and Y. Su. Faster quantum simulation by randomization. *ArXiv e-prints*, May 2018.
[7] A. M. Childs and N. Wiebe. Hamiltonian simulation using linear combinations of unitary operations. *Quantum Info. Comput.*, 12(11-12):901–924, Nov. 2012.
[8] T. H. Dunning Jr. Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen. *The Journal of Chemical Physics*, 90(2):1007–1023, 1989.
[9] A. L. Fetter and . Walecka, John Dirk. *Quantum theory of many-particle systems*. Mineola, N.Y. : Dover Publications, 2003. Originally published: San Francisco, McGraw-Hill, c1971.
[10] R. P. Feynman. There's plenty of room at the bottom. In A. J. G. Hey, editor, *Feynman and Computation*, pages 63–76. Perseus Books, Cambridge, MA, USA, 1999.
[11] H. N. Gabow. An efficient implementation of Edmonds' algorithm for maximum matching on graphs. *Journal of the ACM*, 23(2):221–234, Apr. 1976.
[12] M. M. Halldórsson. Approximating discrete collections via local improvements. In *SODA*, volume 95, pages 160–169, 1995.
[13] G. Hao Low and I. L. Chuang. Hamiltonian simulation by Qubitization. *ArXiv e-prints*, Oct. 2016.

[14] W. J. Hehre, R. F. Stewart, and J. A. Pople. Self-consistent molecular-orbital methods. I. Use of Gaussian expansions of Slater–type atomic orbitals. *The Journal of Chemical Physics*, 51(6):2657–2664, 1969.

[15] Y. Hirata, M. Nakanishi, S. Yamashita, and Y. Nakashima. An efficient conversion of quantum circuits to a linear nearest neighbor architecture. *Quantum Information & Computation*, 11(1&2):142–166, 2011.

[16] R. Jin, Y. Xiang, D. Fuhry, and F. F. Dragan. Overlapping matrix pattern visualization: A hypergraph approach. In *2008 Eighth IEEE International Conference on Data Mining*, pages 313–322. IEEE, 2008.

[17] D. Johnson, M. Garey, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[18] M. Juvan and B. Mohar. Optimal linear labelings and eigenvalues of graphs. *Discrete Appl. Math.*, 36(2):153–168, Apr. 1992.

[19] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.

[20] I. Kassal, S. P. Jordan, P. J. Love, M. Mohseni, and A. Aspuru-Guzik. Polynomial-time quantum algorithm for the simulation of chemical dynamics. *Proceedings of the National Academy of Science*, 105(48):18681–18686, Dec 2008.

[21] I. D. Kivlichan, J. McClean, N. Wiebe, C. Gidney, A. Aspuru-Guzik, G. K.-L. Chan, and R. Babbush. Quantum simulation of electronic structure with linear depth and connectivity. *Physical Review Letters*, 120(11):110501, 2018.

[22] D. E. Knuth. *The Art of Computer Programming, Volume 1 (3rd Ed.): Fundamental Algorithms*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1997.

[23] Y. Koren and D. Harel. A multi-scale algorithm for the linear arrangement problem. In *Revised Papers from the 28th International Workshop on Graph-Theoretic Concepts in Computer Science*, WG '02, pages 296–309, London, UK, UK, 2002. Springer-Verlag.

[24] R. Krishnan, J. S. Binkley, R. Seeger, and J. A. Pople. Self-consistent molecular orbital methods. XX. A basis set for correlated wave functions. *The Journal of Chemical Physics*, 72(1):650–654, 1980.

[25] J. R. McClean, R. Babbush, P. J. Love, and A. Aspuru-Guzik. Exploiting locality in quantum computation for quantum chemistry. *The Journal of Physical Chemistry Letters*, 5(24):4368–4380, 2014.

[26] S. Nahar, S. Sahni, and E. Shragowitz. Simulated annealing and combinatorial optimization. In *23rd ACM/IEEE Design Automation Conference*, pages 293–299. IEEE, 1986.

[27] B. O'Gorman, W. J. Huggins, E. G. Rieffel, and K. B. Whaley. Generalized swap networks for near-term quantum computing. *arXiv e-prints*, page arXiv:1905.05118, May 2019.

[28] F. Pavošević, C. Peng, P. Pinski, C. Riplinger, F. Neese, and E. F. Valeev. SparseMaps A systematic infrastructure for reduced scaling electronic structure methods. V. Linear scaling explicitly correlated coupled-cluster method with pair natural orbitals. *The Journal of Chemical Physics*, 146(17):174108, 2017.

[29] J. Petit. Experiments on the minimum linear arrangement problem. *Journal of Experimental Algorithmics (JEA)*, 8:2–3, 2003.

[30] D. Poulin, M. B. Hastings, D. Wecker, N. Wiebe, A. C. Doherty, and M. Troyer. The trotter step size required for accurate quantum simulation of quantum chemistry. *arXiv preprint arXiv:1406.4920*, 2014.

[31] E. Rieffel and W. Polak. An introduction to quantum computing for non-physicists. *ACM Computing Surveys*, 32(3):300–335, Sept. 2000.

[32] C. Riplinger and F. Neese. An efficient and near linear scaling pair natural orbital based local coupled cluster method. *The Journal of Chemical Physics*, 138(3):034106, 2013.

[33] A. Shafaei, M. Saeedi, and M. Pedram. Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures. In *Proceedings of the 50th Annual Design Automation Conference*, page 41. ACM, 2013.

[34] H. F. Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10(4):545–551, 1959.

[35] M. Valiev, E. J. Bylaska, N. Govind, K. Kowalski, T. P. Straatsma, H. J. Van Dam, D. Wang, J. Nieplocha, W. A. de Jong, E. Apra, and T. L. Windus. NWChem: a comprehensive and scalable open-source solution for large scale molecular simulations. *Computer Physics Communications*, 181(9), 1 2010.