Fast Partitioning of Pauli Strings into Commuting Families for Optimal Expectation Value Measurements of Dense Operators

Nouman Butt, Andrew Lytle, Ben Reggio, Patrick Draper

Department of Physics, University of Illinois, Urbana-Champaign

The Pauli strings P_i appearing in the decomposition of an m-qubit operator H,

$$H = \sum_{i=1}^{4^m} c_i P_i \tag{1}$$

where P_i is a tensor product of Pauli matrices, for example

$$P = \sigma_x \otimes \mathbf{1} \otimes \mathbf{1} \otimes \sigma_y \otimes \ldots \otimes \sigma_z \equiv XIIY \ldots Z\,, \tag{2}$$

can be grouped into commuting families, reducing the number of quantum circuits needed to measure the expectation value of the operator. We detail an algorithm to completely partition the full set of Pauli strings acting on any number of qubits into the minimal number of sets of commuting families, and we provide python code to perform the partitioning. The partitioning method scales linearly with the size of the set of Pauli strings and it naturally provides a fast method of diagonalizing the commuting families with quantum gates. We provide a package that integrates the partitioning into Qiskit, and use this to benchmark the algorithm with dense Hamiltonians, such as those that arise in matrix quantum mechanics models, on IBM hardware. We demonstrate computational speedups close to the theoretical limit of $(3/2)^m$ relative to qubit-wise commuting groupings, for $m = 2, \ldots, 6$ qubits [1].

The cost of a quantum computation depends on several aspects of the computation, including the number of required quantum circuits, the depth of the circuits, and the number of times the same circuits have to be run in order to achieve a level of confidence in the results. For computations involving expectation value measurements, e.g. variational quantum eigensolver (VQE) problems, the naïve approach for a generic operator produces $\mathcal{O}(4^m)$ circuits for m qubits (one for each Pauli string in the operator decomposition). In the NISQ era, the capacity to share the computational burden between classical and quantum computers in an optimal way will be crucial. This is a classical problem, the solution of which can be used to reduce the number of circuits needed to measure an expectation value on a quantum device. An optimal solution partitions all Pauli strings into $2^m + 1$ sets where each set has size $2^m - 1$. This partition reduces the number of circuits from 4^m (3^m), in the naïve (qubit-wise commuting) case, down to $2^m + 1$.

Properties of Pauli Strings

A family is defined as a maximally commuting set of Pauli strings. All families have the same size $2^m - 1$ and can be generated from m generating strings. In other words we only need m generating strings to characterize a family since all other strings in the family are all possible products of these m generating strings. We start with the two canonical families, namely z family with strings of characters Is and Zs and x family with characters Is and Xs. The 2-qubit solution is simple:

	$X \otimes 1$	$1 \otimes X$	$X \otimes X$
$Z \otimes 1$	$Y \otimes 1$	$Z \otimes X$	$Y \otimes X$
$1 \otimes Z$	$X \otimes Z$	$1 \otimes Y$	$X \otimes Y$
$Z \otimes Z$	$Y \otimes Z$	$Z \otimes Y$	$Y \otimes Y$

This table contain all the strings for the 2-qubit case and the families can be read off by picking one generator for the new family from each row and each column. The canonical x and z families can be converted into a Z_2-valued vector space $\mathcal{V}$ with the generating strings playing the role of basis vectors for $\mathcal{V}$. Each family is indexed by a generator matrix A which encodes the commutativity of the strings in that family and furnishes a unique permutation on the x family.

Properties of matrix A

The matrix A is a Z_2 valued $m \times m$ symmetric matrix and has period given by $A^{N-1} = 1$ where $N = 2^m$. On the generating strings(basis vectors $v_i \in \mathcal{V}$) the matrix A act as a permutation:

$$Av_i = v_{P(i)} \tag{3}$$

For each family we have a distinct matrix A which permutes the x family generatring strings into a new set of generating strings. This set of matrices which indices the families, forms a Singer cycle of the form $\{A, A^2, A^3, \ldots A^N = A\}$. Since each matrix realizes a distinct permutation we need to make sure that the ansatz we have for the matrix should have the right characteristics:

- A_i is symmetric
- $A_i - A_j$ is invertible

In order to generate the set of matrices A we use matrix representation of the Galois field $GF(2^m)$. This representation leads to a set of matrices $\{C, C^2, \ldots C^{N-1}\}$ which has the Singer cycle property [2]. However these matrices requires symmetrization which is done using a separate method. With a set of generator matrices at hand we can use them to generate permutations on x family and obtain a new set of unique x family generating strings. We can take the product of these x family generating strings with the z family fixed, and produce generating strings for a new family.

Diagonalizing to z family

With a unique solution with 2^m+1 families, we need to run only $O(2^m)$ circuits rather than 4^m. The computational basis which is used for measurement is the eigen-basis for the z family. For each family we need to find a unitary transformation that can transform the strings in the family to strings in z family modulo an overall sign. The generating strings of the canonical x family can be used to find the unitary transformation $U = exp(i\frac{\pi}{4}\Sigma_m x_m)$ where x_m the generating strings. The overall sign can be evaluated by keeping track of sign change accumulated for each generating string. Surprisingly the set of matrices $\{A, A^2, A^3, \ldots\}$ can be used to find the diagonalizing strings.

For the $i-$th family given by A^i the diagonalizing strings can be found via computing $(A^i)^{\frac{N}{2}}$.

$$(A^i)^{N/2} = \left\{ \begin{array}{ll} A^{i/2} & \text{if } i \bmod 2 = 0 \\ A^{\frac{N+i-1}{2}} & \text{if } i \bmod 2 = 1 \end{array} \right\}. \tag{4}$$

The set of unitary transformation that diagonalizes all family to the canonical z family leads to additional circuit depth which is approximately quadratic in the number of qubits. In QWC(qubit-wise commuting) families the circuit depth only increases by a unit.

Integrating into QISKIT

We developed a python package for generating the optimal solution and the diagonalizing circuits [4]. We also developed a `QISKIT` extension `dense_ev` [3] which contain two classes. The first class `DenseGrouper` works as an analog of native `QISKIT` class `AbelianGrouper` (which generates qubit-wise commuting solutions) and the second class `DensePauliExpectation` builds upon the native `QISKIT PauliExpectation` and contains the method to compute expectation values on hardware and quantum simulators. Both packages are publicly available.

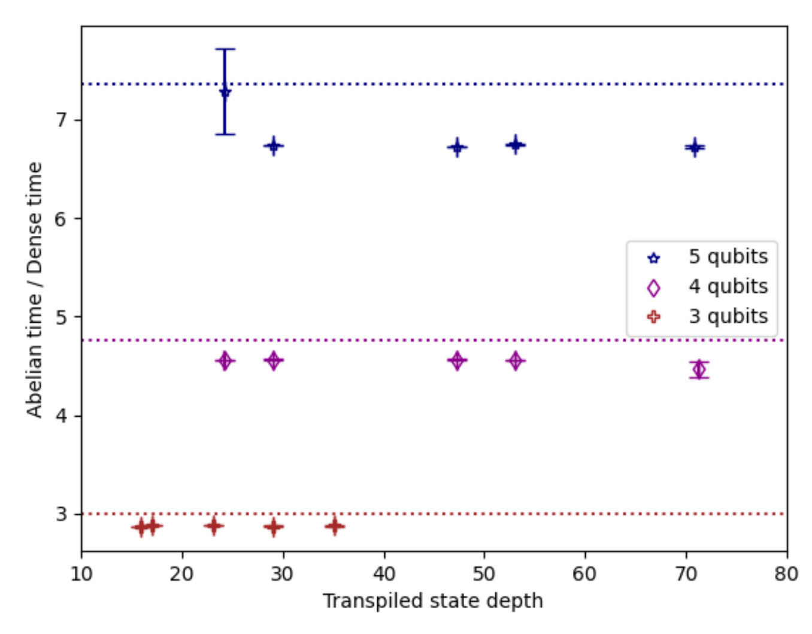
Computational Cost

We compare the computational cost of qubit-wise commuting (QWC) vs. optimal grouping using a simple model for the circuit runtime, $\tau = \tau_{over} + \tau_{circ}(D)$, where τ_{over} is the overhead time and τ_{circ} depends on the depth D of the circuit needed to generate the state $|\psi\rangle$ in the desired expectation value $\langle\psi|H|\psi\rangle$. For a prepared m-qubit state of depth D we have

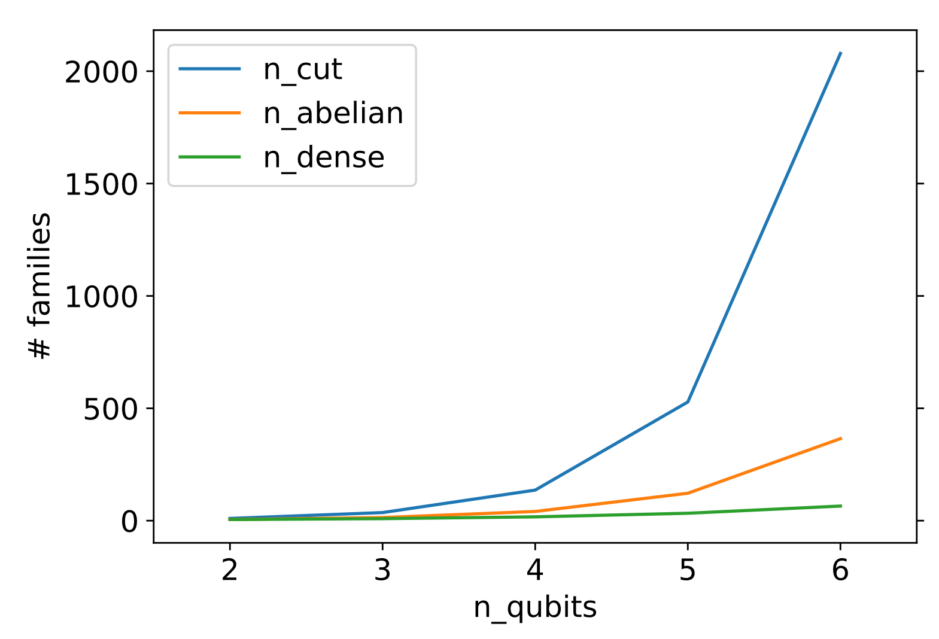
$$\frac{\tau_{\text{QWC}}}{\tau_{\text{optimal}}} = \frac{3^m\Big[\tau_{\text{over}} + \tau_{\text{circ}}(D+1)\Big]}{(2^m+1)\Big[\tau_{\text{over}} + \tau_{\text{circ}}(D + am^2)\Big]}\,. \tag{5}$$

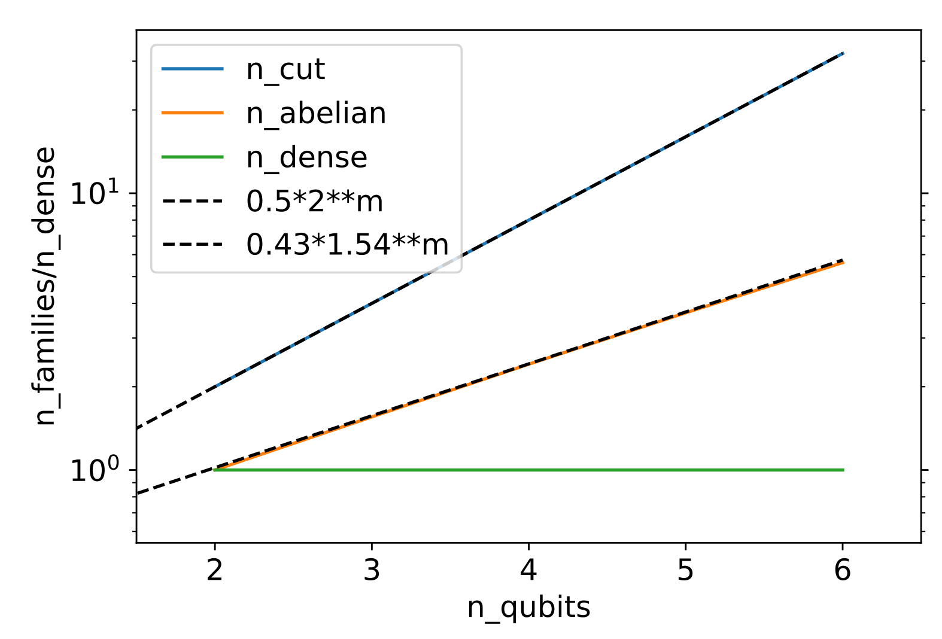
If the circuit overhead is much greater than the circuit runtimes ($\tau_{over} >> \tau_{\text{circ}}$), or the state circuit depth is much greater than the average diagonalization circuit ($D >> am^2$), the runtime improvement will be close to the ideal $(3/2)^m$.

Numerical Results



In the figure above we show the ratios of the computational times between different methods. The circuits ran on `ibmq_quito` using both dense (optimal) and abelian (QWC) grouping methods for 3 to 5 qubits. The ideal speedup factor is the ratio of the number of circuits, $\frac{3^m}{2^m+1}$ shown by dotted lines. The states measured are constructed using `EfficientSU2`, and the `reps` parameter is varied from 1 to 5 to show the effects of increased circuit depth.





On the left figure we show the number of family groupings generated by different grouping methods for the $A_1^+(g = 0.8)$ femtouniverse Hamiltonian [5], as a function of number of qubits m. We compare the naïve decomposition into individual Pauli strings, the `AbelianGrouper`, and the dense grouping. On the right we have the same data but plotted as a ratio to the number of families from the dense grouping $(2^m + 1)$, showing the improvement factor of the dense grouping compared to measuring individual Pauli strings (blue) and grouping generated by `AbelianGrouper` (orange). The dotted lines give an indication of the exponential improvement observed using the dense vs. other methods.

References

[1] *arXiv:2305.11847*

[2] *Andrew Jena (2019). Partitioning Pauli Operators: in Theory and in Practice. UWSpace. http://hdl.handle.net/10012/15017*

[3] *https://github.com/Benjreggio/Psfam*

[4] *https://github.com/atlytle/dense-ev*

[5] *arXiv:2211.10870*

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN