Supporting information for

“Pauli String Partitioning Algorithm with the Ising Model for Simultaneous Measurements”

Tomochika Kurita¹,* Mikio Morita¹ Hirotaka Oshima¹ and Shintaro Sato¹

1. Quantum Laboratory, Fujitsu Research, Fujitsu Limited. 10-1 Morinosato-wakamiya, Atsugi,
Kanagawa, Japan 243-0197; 4-1-1 Kami-odenaka, Nakahara-ku, Kawasaki, Kanagawa, Japan 211-8588

* email: kurita.tomo@fujitsu.com
Appendix 1: Overviews of the Boppana–Halldórsson and Bron–Kerbosch algorithms

Here, we show the detailed descriptions of the Boppana–Halldórsson algorithm (Algorithm S1) and Bron–Kerbosch algorithm (Algorithm S2). They are both based on the maximum clique searching method.

A. Boppana–Halldórsson algorithm

In the Boppana–Halldórsson algorithm\(^1\), a partition \( \mathcal{C}_k \) is created based on a greedy method. Following this algorithm, a partition \( \mathcal{C}(\mathcal{X}) \) is created from a set of Pauli strings \( \mathcal{X} \) with the following steps. The overall description of this algorithm is shown in Algorithm S1.

(i) Randomly choose Pauli string \( X_1 \in \mathcal{X} \) and define \( S_{\mathcal{C}(\{X_1\})}(\mathcal{X}) \) as the subset of \( \mathcal{X} \), which contains all the Pauli strings (other than \( X_1 \)) that commute with \( X_1 \), and define \( S_{\mathcal{A}(\{X_1\})}(\mathcal{X}) \) as the subset of \( \mathcal{X} \), which contains all the Pauli strings that anticommute with \( X_1 \).

(ii) If \( S_{\mathcal{C}(\{X_1\})}(\mathcal{X}) \) is not empty, randomly choose \( X_2 \) from \( S_{\mathcal{C}(\{X_1\})}(\mathcal{X}) \) and then define \( S_{\mathcal{C}(\{X_1\}),\mathcal{C}(\{X_2\})}(\mathcal{X}) \) as the subset of \( S_{\mathcal{C}(\{X_1\})}(\mathcal{X}) \), which contains all the Pauli strings (other than \( X_2 \)) that commute with \( X_2 \), and define \( S_{\mathcal{C}(\{X_1\}),\mathcal{A}(\{X_2\})}(\mathcal{X}) \) as the subset of Pauli strings, all of which anticommute with \( X_2 \).

(iii) Similar to (ii), if \( S_{\mathcal{A}(\{X_1\})}(\mathcal{X}) \) is not empty, randomly choose \( X_3 \) from \( S_{\mathcal{A}(\{X_1\})}(\mathcal{X}) \). Then, define \( S_{\mathcal{A}(\{X_1\}),\mathcal{C}(\{X_2\})}(\mathcal{X}) \) as the subset of \( S_{\mathcal{A}(\{X_1\})}(\mathcal{X}) \), which contains all the Pauli strings (other than \( X_3 \)) that commute with \( X_3 \), and define \( S_{\mathcal{A}(\{X_1\}),\mathcal{A}(\{X_2\})}(\mathcal{X}) \) as the subset of Pauli strings, all of which anticommute with \( X_3 \).

(iv) Divide \( \mathcal{X} \) into subsets \( S_{Q_1,\ldots,Q_m}(\mathcal{X}) \), where \( Q_k \in \{\mathcal{C}(X_k),\mathcal{A}(X_k)\} \) for \( 1 \leq k \leq m \), in the similar manner as mentioned in the aforementioned procedure until all the \( S_{Q_1,\ldots,Q_m}(\mathcal{X}) \) become empty.

(v) With such divided subgroups, define the subsets \( \mathcal{C}(S_{Q_1,\ldots,Q_k}(\mathcal{X})) \) for the descending manner of \( k \). First, \( \mathcal{C}(S_{Q_1,\ldots,Q_k}(\mathcal{X})) \) is determined to be empty when \( S_{Q_1,\ldots,Q_k}(\mathcal{X}) \) is empty. When \( S_{Q_1,\ldots,Q_k}(\mathcal{X}) \) is not empty, determine \( \mathcal{C}(S_{Q_1,\ldots,Q_k}(\mathcal{X})) \) to be a larger subset among \( \{X_{k+1}\} \cup \mathcal{C}(S_{Q_1,\ldots,Q_k,c}(X_{k+1})(\mathcal{X})) \) and \( \mathcal{C}(S_{Q_1,\ldots,Q_k,a}(X_{k+1})(\mathcal{X})) \) after determining \( \mathcal{C}(S_{Q_1,\ldots,Q_k,c}(X_{k+1})(\mathcal{X})) \) and \( \mathcal{C}(S_{Q_1,\ldots,Q_k,a}(X_{k+1})(\mathcal{X})) \). Notably, all the elements commute one another, irrespective of the chosen subset.

(vi) Finally, \( \mathcal{C}(\mathcal{X}) \) is determined as a larger subset among \( \{X_1\} \cup \mathcal{C}(\mathcal{X}_{X_1}) \) and \( \mathcal{C}(\mathcal{X}_{X_1}) \).
Algorithm S1: Boppana–Halldórsson algorithm

1: **Input:** a set of Pauli strings $\mathcal{P} = \{P_1, \ldots, P_N\}$, interaction coefficients $\{c_{i,j}|1 \leq i \leq N, 1 \leq j \leq N\}$ ($c_{i,j} = 0$ if $P_i P_j = P_j P_i$, $c_{i,j} = 1$ otherwise)

2: **function** Ramsey($\mathcal{X}$)

3: if $\mathcal{X} = \emptyset$ then
4:      return $\emptyset$
5: else

6:     Randomly choose $X = P_i$ from the elements of $\mathcal{X}$
7:     **Set:** $\mathcal{S}_{c(X)}(\mathcal{X}) \equiv \{P_j | P_j \in \mathcal{X}, j \neq i, c_{i,j} = 0\}$
8:     **Set:** $\mathcal{S}_{d(X)}(\mathcal{X}) \equiv \{P_j | P_j \in \mathcal{X}, j \neq i, c_{i,j} = 1\}$
9:     return larger of $\{P_i\} + \text{Ramsey}(\mathcal{S}_{c(X)}(\mathcal{X})), \text{Ramsey}(\mathcal{S}_{d(X)}(\mathcal{X}))$

10: end if

11: **Set:** $k = 1$
12: **Set:** $\mathcal{P}_1 = \mathcal{P}$
13: while $\mathcal{P}_k \neq \emptyset$ do
14:     **Set:** $c_k = \text{Ramsey}(\mathcal{P}_k)$
15:     **Set:** $\mathcal{P}_{k+1} = \mathcal{P}_k - c_k$
16:     $k \leftarrow k + 1$
17: end while
18: **Output:** $c_1, c_2, \ldots$

B. Bron–Kerbosch algorithm

In the Bron–Kerbosch algorithm\(^2^3\), many partition candidates $\{c_k(\mathcal{X})\}$ are created and, among these candidates, a partition with the maximum number of elements is chosen. We show the steps of creating $\{c_k(\mathcal{X})\}$ below and in Algorithm S2:

(i) Let $k = 1$.

(ii) Let $c_k(\mathcal{X})$ be empty; then, randomly choose a Pauli string $X_1 \in \mathcal{X}$, and add it to $c_k(\mathcal{X})$.

(iii) Define $N_{X_1}(\mathcal{X})$ as the subset of $\mathcal{X}$, where all the elements commute with $X_1$; then, randomly choose $X_2$ from $N_{X_1}(\mathcal{X})$, and add it to $c_k(\mathcal{X})$.

(iv) Define $N_{X_2}(\mathcal{X})$ as the subset of $\mathcal{X}$, where all the elements commute with $X_2$. Then, randomly choose $X_3$ from $N_{X_1}(\mathcal{X}) \cap N_{X_2}(\mathcal{X})$, and add it to $c_k(\mathcal{X})$.

(v) Repeat step (iv) until $N_{X_1}(\mathcal{X}) \cap \cdots \cap N_{X_m}(\mathcal{X})$ becomes empty, and finally let $c_k(\mathcal{X}) = \{X_1, \ldots, X_m\}$.

(vi) Let $k \leftarrow k + 1$, and repeat steps (ii)–(v) with different Pauli strings $X_1, X_2, \ldots$.

(vii) Repeat step (vi) for all considerable Pauli strings $X_1, X_2, \ldots$.

The corresponding module in NetworkX\(^4\) incorporates protocols to efficiently narrow down the candidate partitions $\{c_k(\mathcal{X})\}$\(^3\).
Algorithm S2: Bron–Kerbosch algorithm

1: **Input**: a set of Pauli strings $\mathcal{P} = \{P_1, \ldots, P_n\}$, interaction coefficients $\{c_{i,j} | 1 \leq i \leq n, 1 \leq j \leq n\}$ ($c_{i,j} = 0$ if $P_iP_j = P_jP_i$, $c_{i,j} = 1$ otherwise).

2: **function** Expand($d, l, \mathcal{P}', \mathcal{N}, \mathcal{F}$)

3: **Set**: $l \leftarrow l_1$

4: Choose $P_m \in \mathcal{N}$ which maximize the number of $P_{m'} \in \mathcal{N}$ satisfying $c_{m,m'} = 0$ and $m \neq m'$

5: **Set**: $\mathcal{N}' = \{P_{m'} \in \mathcal{N} | c_{m,m'} = 1 \text{ or } m' = m\}$

6: for $P_i \in \mathcal{N}' \cap \mathcal{F}$ do

7: **Set**: $d' \leftarrow d + \{P_i\}$

8: **Set**: $N_{P_i} = \{P_j \in \mathcal{P}' | j \neq i, c_{i,j} = 0\}$

9: $\mathcal{F} \leftarrow \mathcal{F} + \{P_i\}$

10: if $N_{P_i} \cap \mathcal{N} = \emptyset$ then

11: $c_i(\mathcal{X}) \leftarrow d'$

12: $l \leftarrow l + 1$

13: else

14: $[c_i(\mathcal{X}), \ldots, c_{l'}(\mathcal{X})], l' = \text{Expand}(d', l, \mathcal{P}', N_{P_i} \cap \mathcal{N}, \mathcal{F})$

15: $l \leftarrow l' + 1$

16: end if

17: end for

18: return $[c_i(\mathcal{X}), \ldots, c_{l-1}(\mathcal{X})], l = 1$

19: **Set**: $k = 1$

20: **Set**: $\mathcal{P}_1 = \mathcal{P}$

21: while $\mathcal{P}_k \neq \emptyset$ do

22: **Set**: $l_0 = 0$

23: **Set**: $\mathcal{F} = \emptyset$

24: for $P_i \in \mathcal{P}_k$ do

25: **Set**: $d = \emptyset$

26: $l_0 \leftarrow l_0 + 1$

27: $d \leftarrow d + \{P_i\}$

28: $\mathcal{F} \leftarrow \mathcal{F} + \{P_i\}$

29: **Set**: $N_{P_i} = \{P_j \in \mathcal{P}_k | j \neq i, c_{i,j} = 0\}$

30: $[c_i, \ldots, c_{l}], l = \text{SunFunc}(d, l_0, \mathcal{P}_k, N_{P_i}, \mathcal{F})$

31: $l_0 \leftarrow l$

32: **Set** $C_k$ to be the set with the largest elements among $[c_i(\mathcal{X}), \ldots, c_{l}(\mathcal{X})]$

33: **Set**: $\mathcal{P}_{k+1} = \mathcal{P}_k - C_k$

34: $k \leftarrow k + 1$

35: end while
Appendix 2: Proof that $m > 1$ guarantees that the global minimum of $f(x_1, ..., x_N)$ in eq 11 is realized when the second term is zero.

We prove this by contradiction. Suppose $Q$ to be the subset of $P = \{P_1, ..., P_n\}$ and the corresponding $x_Q = \{x^Q_1, ..., x^Q_N\}$ satisfies the global minimum of $f(x_1, ..., x_N)$ and its second term of the right hand remains nonzero. Here we define $R$ as the subset of $Q$ with the maximum number of elements whose corresponding $x_R = \{x^R_1, ..., x^R_N\}$ satisfies that the second term of the right hand of eq 11 is zero. Let $S = Q - R$. Based on the supposition about $R$, for every element $S \in S$, there is at least one element in $R$ which does not commute with $S$.

Here, for convenience, we define integers $q$ and $r$ such that $1 \leq r < q \leq N$ and

\[
(x^Q_i, x^R_j) = (1, 1) \text{ for } i = 1, ..., r, \tag{S1}
\]

\[
(x^Q_i, x^R_j) = (1, 0) \text{ for } i = r + 1, ..., q, \tag{S2}
\]

\[
(x^Q_i, x^R_j) = (0, 0) \text{ for } i = q + 1, ..., N. \tag{S3}
\]

Obviously, $q = |Q|$ and $r = |R|$. From eq 11, we can write

\[
f(x_Q) = -\sum_{1 \leq i \leq N} b_i x^Q_i + \sum_{1 \leq i < j \leq N} m_{c_{ij}} x^Q_i x^Q_j = -q + \frac{1}{2} \sum_{1 \leq i < j \leq q} m_{c_{ij}}, \tag{S4}
\]

\[
f(x_R) = -\sum_{1 \leq i \leq N} b_i x^R_i + \sum_{1 \leq i < j \leq N} m_{c_{ij}} x^R_i x^R_j = -r + \frac{1}{2} \sum_{1 \leq i < j \leq r} m_{c_{ij}} = -r. \tag{S5}
\]

where we set $b_i = 1$ for all $i$, and set $c_{ij} = 0$ if $P_i P_j = P_j P_i$ and $c_{ij} = 1$ otherwise.

Since $c_{ij} = 0$ for $1 \leq i \leq r$ and $1 \leq j \leq r$, the second term of eq S4 can be converted as follows:

\[
\sum_{1 \leq i < j \leq q} m_{c_{ij}} = \sum_{1 \leq i < j \leq r} m_{c_{ij}} + \sum_{r+1 \leq i < j \leq q} m_{c_{ij}} + \sum_{1 \leq i < j \leq r} m_{c_{ij}} + \sum_{r+1 \leq i < j \leq q} m_{c_{ij}} \tag{S6}
\]

\[
= \sum_{r+1 \leq i < j \leq q} m_{c_{ij}} + \sum_{1 \leq i < j \leq q} m_{c_{ij}} + \sum_{r+1 \leq i < j \leq q} m_{c_{ij}}. \tag{S7}
\]

Here, for every $i = r + 1, ..., q$, there is at least one $j \in \{1, ..., r\}$ such that $c_{ij} = 1$. Therefore,

\[
\sum_{r+1 \leq i \leq q} m_{c_{ij}} \geq m(q - r). \tag{S8}
\]

Similarly,

\[
\sum_{r+1 \leq i \leq q} m_{c_{ij}} \geq m(q - r). \tag{S9}
\]
From eqs S7, S8 and S9,
\[
\sum_{1 \leq i \leq q} m c_{i,j} \geq \sum_{r+1 \leq i \leq q} m c_{i,j} + 2m(q-r) \geq 2m(q-r).
\] (S10)

Therefore, from eqs S4, S5 and S10, and \( m > 1 \),
\[
f(x_q) \geq -q + m(q-r) = (m-1)(q-r) - r > -r = f(x_R),
\] (S11)
which abuses the supposition that \( f(x_q) \) is the global minimum.

**Appendix 3:** Proof that the \( D \) dependence of \( p_D \) is represented by \( O(D^{-1}) \)

When the Ising model–based algorithm is used, the maximum number of elements in each resultant partition is \( n_{\text{bit}} \). Therefore, the reduction factor \( F_D \) does not exceed \( n_{\text{bit}} \). Here, we define a factor \( \gamma_D \leq 1 \), which satisfies
\[
\gamma_D = \frac{F_D}{n_{\text{bit}}}.
\] (S12)

Therefore, \( \gamma_D = 1 \) if all the resultant partitions contain \( n_{\text{bit}} \) elements; otherwise \( \gamma_D < 1 \). When \( n_{\text{bit}} \) is close to 1 (i.e., \( D \) is close to \( N \)), a reasonable assumption is that each partition that contains \( C_k \) also contains \( n_{\text{bit}} \) (or very close to \( n_{\text{bit}} \)) elements. Thus, the factor \( \gamma_D \) increases and becomes close to 1 (Figure S1). Therefore, by applying this to eq 14, we obtain
\[
p_D^{-1} = \frac{F_1}{F_D} = \frac{F_1}{\gamma_D n_{\text{bit}}} \geq \frac{F_1}{n_{\text{bit}}} = \frac{F_1 D}{N}
\] (S13)
and
\[
p_D^{-1} = \frac{F_1}{F_D} = \frac{\gamma_D n_{\text{bit}}}{\gamma_D n_{\text{bit}}} \leq \frac{N}{n_{\text{bit}}} = D.
\] (S14)

Therefore, we can confirm that \( p_D^{-1} = O(D) \) (Figure 13).
Figure S1. Plots of $\gamma_D$ along the relative problem dimension $D$

References

[1] Boppana, R.; Halldórsson, M. Approximating Maximum Independent Sets by Excluding Subgraphs. *BIT Numerical Math.* 1992, 32, 180–196.

[2] Bron, C.; Kerbosch, J. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM* 1973, 16(9), 575–577.

[3] Tomita, E.; Tanaka, A.; Takahashi, H. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Comput. Sci.* 2006, 363, 28-42.

[4] NetworkX Home Page. [https://networkx.org/](https://networkx.org/) (accessed 2022-03-03).