1 Fast Algorithm for Overlap

As we stated in Theorem [**?**], we can calculate the all overlap of the stabilizer states efficiently. In this section, we will explain the algorithm in detail and introduce some heuristics to improve the efficiency.

1.1 Efficient Enumeration of Stabilizer States

As we stated in the proposition **??**, there are the form of the stabilizer states. Here, we will show a more efficient way to enumerate all the stabilizer states by modifying the form slightly.

Theorem 1. *The form* (**??**) *with the following conditions enumerates all the stabilizer states without any duplication or omission:*

- Q *is a upper triangular* $\mathbb{F}_2^{k\times k}$ *matrix.*
- R *is a* rank k $\mathbb{F}_2^{k\times(n-k)}$ *rref (reduced row echelon form) matrix.*
- t *belongs to the complement of the row space of* R*.*

Proof. Main Ideas come from [**?**]. Firstly, we show that the mapping $\{(Q,c,R,t)\} \to \mathcal{S}_n$ is injective. We can say that

$$\begin{aligned} &\left\{R_1x+t_1 \;\middle|\; x\in\mathbb{F}_2^{n-k}\right\} = \left\{R_2x+t_2 \;\middle|\; x\in\mathbb{F}_2^{n-k}\right\} \\ \iff &\mathrm{Im}(R_1)=\mathrm{Im}(R_2)\wedge(t_2-t_1)\in\mathrm{Im}(R_1) \\ \iff &R_1=R_2\wedge t_1=t_2. \end{aligned}$$

The last equivalence is due to the property of the rref matrix and the complement condition. Since Q is a upper triangular matrix, we can uniquely determine Q and c for given state $|\phi\rangle$. Thus, if two states are the same, then the corresponding (Q,c,R,t) are also the same, which means that the mapping is injective.

Next, we show that the mapping is surjective. Since the mapping is injective, we only have to show that the cardinality of the domain is equal to that of the codomain, i.e., $|\mathcal{S}_n|$. It is known that the number of rank k $\mathbb{F}_2^{k\times(n-k)}$ rref matrices is $\begin{bmatrix}n\\k\end{bmatrix}_2$, which is a q-binomial coefficient with $q=2$. Therefore, the number of Q,c,R,t is $2^{k(k+1)/2}, 2^k, \begin{bmatrix}n\\k\end{bmatrix}_2, 2^{n-k}$, respectively, and the total number of states is

$$2^n+\sum_{k=1}^{n} 2^{k(k+1)/2}2^k\begin{bmatrix}n\\k\end{bmatrix}_2 2^{n-k} = 2^n\sum_{k=0}^{n}\begin{bmatrix}n\\k\end{bmatrix}_2 2^{k(k+1)/2} = 2^n\prod_{k=1}^{n}(2^k+1) = |\mathcal{S}_n|.$$

In the second last equation, we used the q-binomial theorem. Therefore, the mapping is surjective, which concludes the proof. □

In theorem 1, we used $\mathbb{F}_2$. However, in the viewpoint of the dfs algorithm, it is more practical to use $\{0,1\}\subset\mathbb{Z}$ and allow the term $c^\top x$ to be any integer. Therefore, the following corollary is useful.

Corollary 1. *In theorem 1, We can substitute* $\mathbb{F}_2$ *with* $\{0,1\}\subset\mathbb{Z}$*.*

Proof. By changing $\mathbb{F}_2$ to $\{0,1\}\subset\mathbb{Z}$, the term $(-1)^{x^\top Qx}$ is invariant, and the term $i^{c^\top x}$ is multiplied by -1 iff $p\equiv 2,3 \pmod 4$, where p is the number of i such that $c_i=1$ and

$x_i = 1$. Now, we consider the following form:

$$\begin{cases} |\phi\rangle := |t\rangle & \text{if } k = 0, \\ |\phi\rangle := \dfrac{1}{2^{k/2}} \displaystyle\sum_{x=0}^{2^k-1} (-1)^{x^\top (Q+Q')x} i^{c^\top x} |Rx + t\rangle & \text{if } k > 0, \end{cases} \tag{1}$$

where $Q \in \{0,1\}^{k\times k}, c \in \{0,1\}^k, R \in \{0,1\}^{k\times(n-k)}, t \in \{0,1\}^{n-k}, \mathrm{rank}(R) = k$ and $Q'_{ij} = 1$ iff $(i < j) \wedge (c_i = c_j = 1)$. Now, if the pair (Q, c, R, t) in (1) is the same as that of the original form (**??**), then the two states are representing the exactly same state since

$$(-1)^{x^\top Q' x} = (-1)^{\binom{p}{2}} = \begin{cases} 1 & \text{if } p \equiv 0, 1 \pmod 4, \\ -1 & \text{if } p \equiv 2, 3 \pmod 4. \end{cases}$$

Therefore, by identifying the $Q+Q'$ in $\mathbb{Z}$ with new Q'' in $\mathbb{F}_2$, we can conclude the proof. □

1.2 Calculating the Overlap

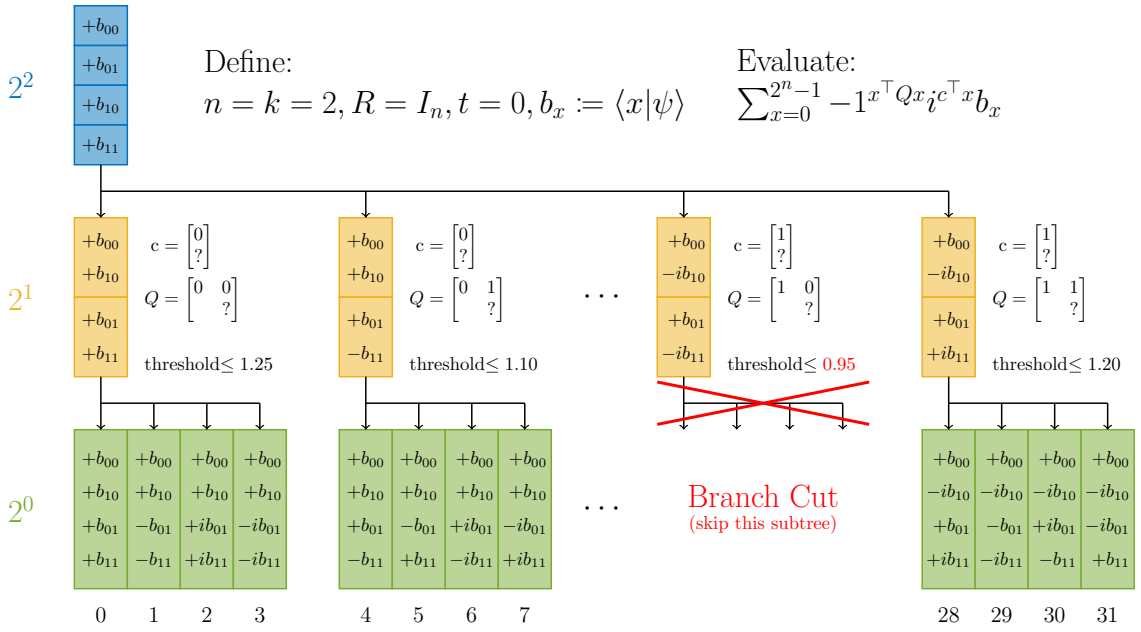

Figure 1: The visualization of the DFS algorithm. The DFS algorithm is a recursive algorithm that calculates the overlap of the stabilizer states by recursively calculating the overlap of the substates. The algorithm is efficient since the overlap of the substates can be calculated with very small computational cost.

Thanks to the corollary 1, we can prove the following theorem.

Theorem 2. *Fix k, R, t in the standard form* (**??**)*. Then, we can compute the overlap $\langle\phi|\psi\rangle$ efficiently.*

Proof. We only consider the case $k > 0, R = 0, t = 0$ for the simplicity. Other cases are trivial or can be reduced to this case. Define $x := \begin{bmatrix} x_0 \\ \overline{x} \end{bmatrix}$, $c := \begin{bmatrix} c_0 \\ \overline{c} \end{bmatrix}$, and $Q := \begin{bmatrix} Q_{00} & Q_0^\top \\ 0 & \overline{Q} \end{bmatrix}$

(x_0, c_0 and Q_{00} are all in $\{0,1\}$). Since $x^\top Q x = x_0(Q_{00} + Q_0^\top \overline{x}) + \overline{x}^\top \overline{Q}\overline{x}$ and $c^\top x = c_0 x_0 + \overline{c}^\top \overline{x}$, we can rewrite the state as

$$\begin{aligned}
|\phi\rangle &= \sum_{x=0}^{2^k-1} (-1)^{x^\top Q x} i^{c^\top x} |x\rangle \\
&= \sum_{\overline{x}=0}^{2^{k-1}-1} (-1)^{\overline{x}^\top \overline{Q}\overline{x}} i^{\overline{c}^\top \overline{x}} \Big(|2\overline{x}\rangle + (-1)^{Q_{00}+Q_0^\top \overline{x}} i^{c_0} |2\overline{x}+1\rangle \Big) \\
&= \sum_{\overline{x}=0}^{2^{k-1}-1} (-1)^{\overline{x}^\top \overline{Q}\overline{x}} i^{\overline{c}^\top \overline{x}} |\overline{x}'\rangle
\end{aligned}$$

by defining $|\overline{x}'\rangle := |2\overline{x}\rangle + (-1)^{Q_{00}+Q_0^\top \overline{x}} i^{c_0} |2\overline{x}+1\rangle$. (Question: Is it natural to equate integer $2\overline{x}+1$ to the vector $\begin{bmatrix} 1 \\ \overline{x} \end{bmatrix}$?)

Thus, we can compute the overlap recursively with very small computational cost per each step. This leads to the efficient calculation of the overlaps, which concludes the proof. □

1.3 Branch Cut For The DFS

In this section, we will explain the branch cut methods used in the dfs search. Firstly, please recall that we are maximizing the following:

$$\max_{c,Q} \left\{ \left| \sum_{x=0}^{2^n-1} -1^{x^\top Q x} i^{c^\top x} P_x \right| \right\}$$

This value can easily evaluate as the following:

$$\max_{c,Q} \left\{ \left| \sum_{x=0}^{2^n-1} -1^{x^\top Q x} i^{c^\top x} P_x \right| \right\} \leq \max_{c,Q} \left\{ \sum_{x=0}^{2^n-1} \left| -1^{x^\top Q x} i^{c^\top x} P_x \right| \right\} = \sum_{x=0}^{2^n-1} |P_x|$$

However, since each coefficient takes only $1, -1, i$ or $-i$, we can obtain more tight bound by

$$\max_{c,Q} \left\{ \left| \sum_{x=0}^{2^n-1} -1^{x^\top Q x} i^{c^\top x} P_x \right| \right\} \leq \max_{c,Q} \left\{ \left| \sum_{x=0}^{2^n-1} i^{c_x} P_x \right| \right\} \tag{2}$$

where $c_x \in \{0,1,2,3\}$ is the independent variable for each x. Let $\mathcal{P} := \sum_{x=0}^{2^n-1} i^{c_x^*} P_x$ be the one of optimal solutions for the problem (2). Then, without loss of generality, we can assume that $\frac{\pi}{2} \leq \arg \mathcal{P} < \frac{3\pi}{2}$, and by sorting and multiplying $i, -1$ or $-i$ to P_x appropriately, we can also assume that

$$0 \leq \arg(P_0) \leq \arg(P_1) \leq \cdots \leq \arg(P_{2^n-1}) < \pi/2. \tag{3}$$

If all c_x satisfies $\arg(\mathcal{P}) - \pi/4 \leq \arg(i^{c_x} P_x) < \arg(\mathcal{P}) + \pi/4$, then c_x is optimal for $\mathcal{P}$. Therefore, we can justify the following Algorithm 1 by moving $\arg(\mathcal{P})$ in the range of $[\pi/2, 3\pi/2)$. Also refer to the figure 2 for the visualization of this algorithm. The time complexity of this algorithm is $O(n2^n)$ due to the sorting of 2^n elements.

Algorithm 1: Branch Cut Algorithm

Input: Coefficients P_x for $x = 0, 1, \ldots, 2^n - 1$

Output: The answer for the problem (2)

Sort and modify the coefficients P_x so that the condition (3) is satisfied

ans $\leftarrow 0$, $\quad c_x \leftarrow 0 \quad$ for all x

for $x \leftarrow 0$ **to** $2^n - 1$ **do**

  ans $\leftarrow \max\left(\text{ans}, \left|\sum_{x=0}^{2^n-1} i^{c_x} P_x\right|\right)$

  $c_x \leftarrow c_x + 1$

return ans

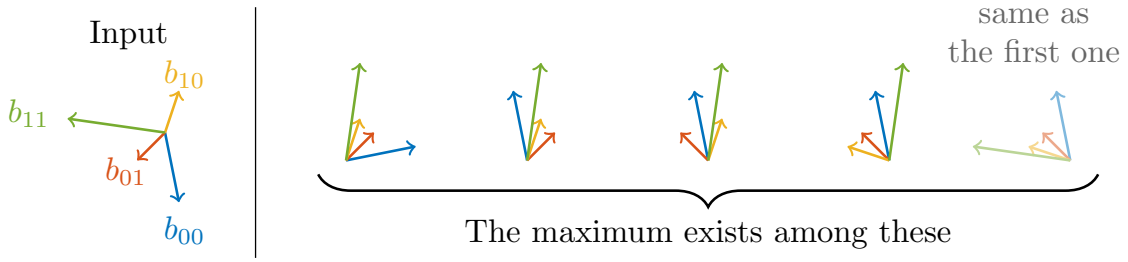

Figure 2: The visualization of Algorithm 1. Suppose that $n = 2$ and P_x are represented as the vectors in the complex plane (for example, $P_{00} = 1 - 5i$) by the left figure. Then, by sorting and iterating the loop in Algorithm 1, we can obtain 2^n patterns of the coefficients c_x as the right figure. The maximum of for the problem (2) exists among these 2^n patterns.