# Generalized Greedy Algorithms for Synthesizing Low-depth CNOT Circuits

Tung Chou

Academia Sinica, Taiwan
blueprint@crypto.tw

**Abstract.** A CNOT circuit is a quantum circuit where the only type of gate appeared in the circuit is the CNOT gate. Given an $n \times n$ binary matrix which specifies the relationship between input and output, existing greedy algorithms generate corresponding CNOT circuits of $n$ qubits, with the goal of minimizing depth of the circuits.
This short paper presents two new greedy algorithms, which can be considered as generalized versions of existing greedy algorithms. The new algorithms are inspired by the algorithm presented in the Asiacrypt 2024 paper "Quantum circuits of AES with a low-depth linear layer and a new structure" [1]. Although we have not run large-scale experiments, small-scale experiments suggest that the new algorithms are at least as powerful as existing greedy algorithms.

## 1 Row and Column Operations

We denote by $\mathcal{P}$ the set of permutation matrices in $\mathbb{F}_2^{n\times n}$. We denote by $\mathcal{S}$ the set of binary matrices in $\mathbb{F}_2^{n\times n}$ that can be obtained by adding one row of $I_n$ to another row. Equivalently, $\mathcal{S}$ can be considered as the set of matrices that can be obtained by adding one column of $I_n$ to another. For each element $R \in S$, $R^2 = I_n$. Each elements in $\mathcal{S}$ can be considered as a row operation or a column operation, depending on how it is multiplied with an $n\times n$ matrix. Let $R_1, \ldots, R_k$ be elements in $\mathcal{S}$. If each $R_i$ is obtained by adding row $j_i$ to row $j_i'$ of $I_n$, and if $|\{j_1, j_1', \ldots, j_k, j_k'\}| = 2k$, we say that $R_1, \ldots, R_k$ are independent operations. We define

$$\overline{\mathcal{S}} = \{\mathcal{S}' \subset \mathcal{S} \mid \mathcal{S}' \neq \emptyset \text{ and operations in } \mathcal{S}' \text{ are independent}\}.$$

Given $\mathcal{B} = \{R_1, \ldots, R_k\} \in \overline{\mathcal{S}}$ and $A \in \mathbb{F}_2^{n\times n}$, we define $\mathcal{B}A$ as $R_k \cdots R_1 A$. Note that the order of $R_1, \ldots, R_k$ does not affect the definition of $\mathcal{B}A$. Similarly, given $\mathcal{B} = \{C_1, \ldots, C_k\} \in \overline{\mathcal{S}}$, we define $A\mathcal{B}$ as $AC_1 \cdots C_k$. Finally, we define $\emptyset A$ and $A\emptyset$ as $A$.

## 2 Existing Greedy Algorithms

We start with a type of existing greedy algorithms, which we call row-based greedy algorithms. Given $A \in \mathrm{GL}_n(\mathbb{F}_2)$, each row-based algorithm finds $\mathcal{L}_1, \ldots, \mathcal{L}_d$

**Algorithm 1:** A row-based greedy algorithm

**Input :** $A \in \mathrm{GL}_n(\mathbb{F}_2)$
**Output:** $\mathcal{L}_1, \ldots, \mathcal{L}_d \in \overline{\mathcal{S}}$ for some $d$, and $P \in \mathcal{P}$, such that $\mathcal{L}_d \cdots \mathcal{L}_1 A = P$.

$\hat{A} \leftarrow A$;
$d = 0$;
**if** $A \in \mathcal{P}$ **then**
  **return** $A$;
**end**
**while** *True* **do**
  **if** $\mathcal{B}\hat{A} = P'$ *for some* $\mathcal{B} \in \overline{\mathcal{S}}$ *and* $P' \in \mathcal{P}$ **then**
    $\mathcal{L}_{d+1} \leftarrow \mathcal{B}$;
    $d \leftarrow d + 1$;
    $P \leftarrow P'$;
    break;
  **end**
  $\mathcal{B} \leftarrow \Lambda_{\mathrm{row}}(\hat{A})$ ; `/* Λ_row is presented in Alg. 2 */`
  **if** $\mathcal{B} \neq \emptyset$ **then**
    $\mathcal{L}_{d+1} \leftarrow \mathcal{B}$;
    $\hat{A} \leftarrow \mathcal{L}_{d+1}\hat{A}$;
    $d \leftarrow d + 1$;
  **else**
    `EXCEPTION`;
  **end**
**end**
**return** $(\mathcal{L}_1, \ldots, \mathcal{L}_d)$ and $P$.

$\in \overline{\mathcal{S}}$, one at a time, such that

$$\mathcal{L}_d \cdots \mathcal{L}_1 A = P.$$

Each $\mathcal{L}_i$ is called a layer of operations. We conclude that $A$ can be written as

$$\mathcal{L}_1 \cdots \mathcal{L}_d P,$$

which represents a CNOT circuit of depth $d$ (applying $P$ is considered as "wiring", so it does not increase the depth).

Algorithm 1 shows the pseudocode of a row-based algorithm. In each iteration of the while loop, a new layer is formed by calling $\Lambda_{\mathrm{row}}$, except when $\mathcal{B} \neq \emptyset$, in which case an exception is raised. $\Lambda_{\mathrm{row}}$ is called a layer-forming subroutine. The last layer is always formed by the first if statement in the loop: As mentioned in [1], given $\hat{A} \in \mathbb{F}_2^{n \times n}$, it is easy to derive $\mathcal{B} \in \overline{\mathcal{S}}$ such that $\mathcal{B}\hat{A} \in \mathcal{P}$, if such a $\mathcal{B}$ exists.

### Forming a New Layer of Operations

The layer-forming subroutine used in row-based greedy algorithms is presented in Algorithm 2. In the subroutine, $\mathcal{B}$ serves as a buffer to collect operations in

**Algorithm 2:** $\Lambda_{\text{row}}$, the layer-forming subroutine in row-based greedy algorithms).

**Input :** $\hat{A} \in \mathrm{GL}_n(\mathbb{F}_2)$
**Output:** $\mathcal{B} \in \overline{\mathcal{S}} \cup \{\emptyset\}$

$\tilde{A} \leftarrow \hat{A}$;
$\mathcal{B} \leftarrow \emptyset$;
**while** *True* **do**
  Compute $\mathcal{T} \subset \mathcal{S}$, the set of row operations that are independent of the ones in $\mathcal{B}$;
  **if** $\mathcal{T} = \emptyset$ **then**
    break;
  **end**
  $R \xleftarrow{\$} \{R \in \mathcal{T} \mid c(R\tilde{A}) \leq c(R'\tilde{A}) \text{ for all } R' \in \mathcal{T}\}$;
  **if** $c(R\tilde{A}) \leq c(\tilde{A})$ **then**
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{R\}$;
    $\tilde{A} \leftarrow R\tilde{A}$;
  **else**
    break;
  **end**
**end**
**return** $\mathcal{B}$;

the new layer. $\tilde{A}$ is the result of applying operations in the buffer to $\hat{A}$. In each iteration of the while loop, one new operation can be added to $\mathcal{B}$. To select an operation to be added to $\mathcal{B}$, the subroutine first collects all operations that are independent of the ones in $\mathcal{B}$. If such operations do not exist, the subroutine simply returns $\mathcal{B}$ and ends. Otherwise, the subroutine chooses a random operation $R$ that gives the lowest $c(R\tilde{A})$, and adds $R$ to $\mathcal{B}$ if $c(R\tilde{A}) < c(\tilde{A})$.

The function $c : \mathbb{F}_2^{n \times n} \longrightarrow \mathbb{Z}$ is called the cost function. The lower the cost is, the more the input matrix is considered close to a permutation matrix. As a simple example, the cost function can be defined as the Hamming weight of the input matrix, but there are other ways to define a cost function. The reason to choose an operation $R$ that gives the lowest $c(R\tilde{A})$ is because we hope $R\tilde{A}$ can be as similar to a permutation matrix as possible. If $c(R\tilde{A}) > c(\tilde{A})$, $R\tilde{A}$ is considered worse than $\tilde{A}$ (in terms of similarity to a permutation matrix), so the subroutine simply returns $\mathcal{B}$ and ends. The condition $c(R\tilde{A}) \leq c(\tilde{A})$ can be relaxed though: For example, one may choose to use

$$c(R\tilde{A}) \leq c(\tilde{A}) + \delta$$

for some small $\delta$.

For completeness, we mention that there are also column-based greedy algorithms. Given $A \in \mathrm{GL}_n(\mathbb{F}_2)$, each column-based algorithm finds $\mathcal{L}_1, \ldots, \mathcal{L}_{d'} \in \overline{\mathcal{S}}$,

one at a time, such that

$$A\mathcal{L}_1 \cdots \mathcal{L}_{d'} = P.$$

Therefore, $A$ can be written as

$$P\mathcal{L}_{d'} \cdots \mathcal{L}_1,$$

which represents a CNOT circuit of depth $d'$. It is easy to modify Algorithm 1 and Algorithm 2 to obtain pseudocode code for column-based greedy algorithms. The layer-forming subroutine in column-based greedy algorithms will be called $\Lambda_{\text{col}}$.

## 3 Generalized Greedy Algorithms

This section presents our generalized greedy algorithms. Given $A \in \text{GL}_n(\mathbb{F}_2)$, each of our algorithms finds $\mathcal{L}_1^{(0)}, \ldots, \mathcal{L}_d^{(0)} \in \overline{\mathcal{S}}$ and $\mathcal{L}_1^{(1)}, \ldots, \mathcal{L}_{d'}^{(1)} \in \overline{\mathcal{S}}$, such that

$$\mathcal{L}_d^{(0)} \cdots \mathcal{L}_1^{(0)} A \mathcal{L}_1^{(1)} \cdots \mathcal{L}_{d'}^{(1)} = P.$$

Therefore, $A$ can be written as

$$\mathcal{L}_1^{(0)} \cdots \mathcal{L}_d^{(0)} P \mathcal{L}_{d'}^{(1)} \cdots \mathcal{L}_1^{(1)},$$

which represents a CNOT circuit of depth $d + d'$.

### 3.1 Choosing One Out of Two Layers

In our first generalized algorithm, to form a new layer, we do not always form a layer of row operations as in row-based algorithms, or always form a layer of column operations as in column-based algorithms. Instead, we generate a new layer of row operations using $\Lambda_{\text{row}}$, generate a new layer of column operations using $\Lambda_{\text{col}}$, and then choose one of the two layers as the new layer. Unsurprisingly, we pick the layer that leads to a lower cost. The algorithm, along with the corresponding layer-forming subroutine, are presented in Algorithm 3 and Algorithm 4.

### 3.2 Forming Two Layers at a Time

In our second generalized algorithm, we do not form one layer at a time. Instead, we form a new layer of row operations and a new layer of column operations at the same time. In the two-layer-forming subroutine, two buffers $\mathcal{B}^{(0)}$ and $\mathcal{B}^{(1)}$ are maintained, one for the new layer of row operations and one for the new layer of column operations. In each iteration, we collect all candidate row operations and all candidate column operations, and select one operation, which can be either a row or a column operation. Once the operation is chosen, it is added into the corresponding buffer, depending on which type of operation it is. In some sense, what this algorithm does is to consider $\mathcal{B}^{(0)}$ and $\mathcal{B}^{(1)}$ together as a big buffer and keep adding new operations to it. The algorithm, along with the corresponding two-layer-forming subroutine, are presented in Algorithm 5 and Algorithm 6.

## References

1. Haotian Shi and Xiutao Feng. Quantum circuits of AES with a low-depth linear layer and a new structure. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 358–395. Springer, 2024.

**Algorithm 3:** Our first generalized greedy algorithm

**Input :** $A \in \mathrm{GL}_n(\mathbb{F}_2)$
**Output:** $\mathcal{L}_1^{(0)}, \ldots, \mathcal{L}_d^{(0)} \in \overline{\mathcal{S}}$ for some $d$, $\mathcal{L}_1^{(1)}, \ldots, \mathcal{L}_{d'}^{(1)} \in \overline{\mathcal{S}}$ for some $d'$, and $P \in \mathcal{P}$, such that $\mathcal{L}_d^{(0)} \cdots \mathcal{L}_1^{(0)} A \mathcal{L}_1^{(1)} \cdots \mathcal{L}_{d'}^{(1)} = P$.

$\hat{A} \leftarrow A$;
$d \leftarrow 0$;
$d' \leftarrow 0$;
**if** $A \in \mathcal{P}$ **then**
  **return** $A$;
**end**
**while** *True* **do**
  **if** $\mathcal{B}\hat{A} = P'$ *for some* $\mathcal{B} \in \overline{\mathcal{S}}$ *and* $P' \in \mathcal{P}$ **then**
    $\mathcal{L}_{d+1}^{(0)} \leftarrow \mathcal{B}$;
    $d \leftarrow d + 1$;
    $P \leftarrow P'$;
    break;
  **end**
  $\mathcal{B}, t \leftarrow \Lambda_{\mathrm{choose}}(\hat{A})$ ; `/* Λ_choose is presented in Alg. 4 */`
  **if** $t = 0$ **then**
    $\mathcal{L}_{d+1}^{(0)} \leftarrow \mathcal{B}$;
    $\hat{A} \leftarrow \mathcal{L}_{d+1}^{(0)} \hat{A}$;
    $d \leftarrow d + 1$;
  **end**
  **if** $t = 1$ **then**
    $\mathcal{L}_{d'+1}^{(1)} \leftarrow \mathcal{B}$;
    $\hat{A} \leftarrow \hat{A} \mathcal{L}_{d'+1}^{(1)}$;
    $d' \leftarrow d' + 1$;
  **end**
  **if** $t = -1$ **then**
    `EXCEPTION`;
  **end**
**end**
**return** $(\mathcal{L}_1^{(0)}, \ldots, \mathcal{L}_d^{(0)})$, $(\mathcal{L}_1^{(1)}, \ldots, \mathcal{L}_{d'}^{(1)})$, and $P$.

**Algorithm 4:** $\Lambda_{\text{choose}}$, the layer-forming subroutine for our first generalized algorithm

**Input :** $\hat{A} \in \mathrm{GL}_n(\mathbb{F}_2)$
**Output:** $\mathcal{B} \in \overline{\mathcal{S}} \cup \{\emptyset\}$ and an integer in $\{-1, 0, 1\}$

$\mathcal{B}^{(0)} \leftarrow \Lambda_{\text{row}}(\hat{A})$;
$\mathcal{B}^{(1)} \leftarrow \Lambda_{\text{col}}(\hat{A})$;
**if** $\mathcal{B}^{(0)} \neq \emptyset$ **and** $c(\mathcal{B}^{(0)}\hat{A}) \leq c(\hat{A}\mathcal{B}^{(1)})$ **and** $c(\mathcal{B}^{(0)}\hat{A}) \leq c(\hat{A})$ **then**
  **return** $\mathcal{B}^{(0)}, 0$;
**end**
**if** $\mathcal{B}^{(1)} \neq \emptyset$ **and** $c(\hat{A}\mathcal{B}^{(1)}) \leq c(\mathcal{B}^{(0)}\hat{A})$ **and** $c(\hat{A}\mathcal{B}^{(1)}) \leq c(\hat{A})$ **then**
  **return** $\mathcal{B}^{(1)}, 1$;
**end**
**return** $\emptyset, -1$;

**Algorithm 5:** Our second generalized greedy algorithm

**Input :** $A \in \mathrm{GL}_n(\mathbb{F}_2)$

**Output:** $\mathcal{L}_1^{(0)}, \ldots, \mathcal{L}_d^{(0)} \in \overline{\mathcal{S}}$ for some $d$, $\mathcal{L}_1^{(1)}, \ldots, \mathcal{L}_{d'}^{(1)} \in \overline{\mathcal{S}}$ for some $d'$, and $P \in \mathcal{P}$, such that $\mathcal{L}_d^{(0)} \cdots \mathcal{L}_1^{(0)} A \mathcal{L}_1^{(1)} \cdots \mathcal{L}_{d'}^{(1)} = P$.

$\hat{A} \leftarrow A$;
$d \leftarrow 0$;
$d' \leftarrow 0$;
**while** *True* **do**
  **if** $\hat{A} \in \mathcal{P}$ **then**
    $P \leftarrow \hat{A}$;
    break;
  **end**
  **if** $\mathcal{B}\hat{A} = P'$ *for some* $\mathcal{B} \in \overline{\mathcal{S}}$ *and* $P' \in \mathcal{P}$ **then**
    $\mathcal{L}_{d+1} \leftarrow \mathcal{B}$;
    $d \leftarrow d + 1$;
    $P \leftarrow P'$;
    break;
  **end**
  $\mathcal{B}^{(0)}, \mathcal{B}^{(1)} \leftarrow \Lambda_{\text{two-layer}}(\hat{A})$ ;  `/* Λ_two-layer is presented in Alg. 6 */`
  **if** $\mathcal{B}^{(0)} \neq \emptyset$ **then**
    $\mathcal{L}_{d+1}^{(0)} \leftarrow \mathcal{B}^{(0)}$;
    $\hat{A} \leftarrow \mathcal{L}_{d+1}^{(0)} \hat{A}$;
    $d \leftarrow d + 1$;
  **end**
  **if** $\mathcal{B}^{(1)} \neq \emptyset$ **then**
    $\mathcal{L}_{d'+1}^{(1)} \leftarrow \mathcal{B}^{(1)}$;
    $\hat{A} \leftarrow \hat{A} \mathcal{L}_{d'+1}^{(1)}$;
    $d' \leftarrow d' + 1$;
  **end**
  **if** $\mathcal{B}^{(0)} = \mathcal{B}^{(1)} = \emptyset$ **then**
    `EXCEPTION`;
  **end**
**end**
**return** $(\mathcal{L}_1^{(0)}, \ldots, \mathcal{L}_d^{(0)})$, $(\mathcal{L}_1^{(1)}, \ldots, \mathcal{L}_{d'}^{(1)})$, and $P$.

**Algorithm 6:** $\Lambda_{\text{two-layer}}$, the two-layer-forming subroutine for our second generalized algorithm

**Input :** $\hat{A} \in \mathrm{GL}_n(\mathbb{F}_2)$
**Output:** $\mathcal{B}^{(0)}, \mathcal{B}^{(1)} \in \overline{\mathcal{S}} \cup \{\emptyset\}$

$\tilde{A} \leftarrow \hat{A}$;
$\mathcal{B}^{(0)} \leftarrow \emptyset$;
$\mathcal{B}^{(1)} \leftarrow \emptyset$;
**while** *True* **do**
  Compute $\mathcal{T}^{(0)} \subset \mathcal{S}$, the set of row operations that are independent of the ones in $\mathcal{B}^{(0)}$;
  Compute $\mathcal{T}^{(1)} \subset \mathcal{S}$, the set of column operations that are independent of the ones in $\mathcal{B}^{(1)}$;
  **if** $\mathcal{T}^{(0)} = \mathcal{T}^{(1)} = \emptyset$ **then**
    break;
  **end**
  $\mathcal{T} \leftarrow \{(R, 0) \mid R \in \mathcal{T}^{(0)}\} \cup \{(C, 1) \mid C \in \mathcal{T}^{(1)}\}$;
  $(X, t) \xleftarrow{\$} \{(X, t) \in \mathcal{T} \mid c(X^{1-t}\tilde{A}X^t) \leq c((X')^{1-t'}\tilde{A}(X')^{t'}) \text{ for all } (X', t') \in \mathcal{T}\}$;
  **if** $c(X^{1-t}\tilde{A}X^t) \leq c(\tilde{A})$ **then**
    $\mathcal{B}^{(t)} \leftarrow \mathcal{B}^{(t)} \cup \{X\}$;
    $\tilde{A} \leftarrow X^{1-t}\tilde{A}X^t$;
  **else**
    break;
  **end**
**end**
**return** $\mathcal{B}^{(0)}, \mathcal{B}^{(1)}$;