Generalized Greedy Algorithms for Synthesizing Low-depth CNOT Circuits

Tung Chou

Academia Sinica, Taiwan

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 of a CNOT circuit with n qubits, existing greedy algorithms generate an equivalent CNOT circuit, with the goal of minimizing the depth of the circuit.

This short paper presents two new greedy algorithms, which can be considered as generalized versions of existing greedy algorithms. Although we have not run large-scale experiments, small-scale experiments suggest that they are at least as powerful as existing greedy algorithms.

1 Row and Column Operations

We denote by $\mathcal{S} \subset \mathbb{F}_2^{n\times n}$ the set of binary matrices 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. 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 $AC_1 \cdots C_k$. Finall, 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. Row-based greedy algorithms are iterative. Given a matrix $A \in \mathbb{F}_2^{n\times n}$, a row-based algorithm finds a list of matrices $R_1, \ldots, R_k \in \mathcal{C}$, one matrix per iteration, such that

$$R_k \cdots R_2 R_1 A = P, \tag{1}$$

Data: $A \in \mathbb{F}_2^{n \times n}$
Result: $L_1, \ldots, L_d \in \overline{\mathcal{S}}$ for some d, and a permutation matrix P, such that $L_d \cdots L_1 A = P$.

$\hat{A} \leftarrow A$;
$d = 0$;
if *A is a permutation matrix* **then**
  return A;
end
while *True* **do**
  if *$\mathcal{B}\hat{A} = P'$ for some $\mathcal{B} \in \overline{\mathcal{S}}$ and permutation matrix P'* **then**
    $L_{d+1} \leftarrow \mathcal{B}$;
    $d \leftarrow d + 1$;
    $P \leftarrow P'$;
    break;
  end
  $\mathcal{B} \leftarrow$ LF-row($\hat{A}$) ; `/* LF-row is presented in Algorithm 2 */`
  if $\mathcal{B} \neq \emptyset$ **then**
    $L_{d+1} \leftarrow \mathcal{B}$;
    $\hat{A} \leftarrow L_{d+1}\hat{A}$;
    $d \leftarrow d + 1$;
  else
    `Error`;
  end
end
return $L_1, \ldots, L_d$ and P.

Algorithm 1: A row-based greedy algorithm

where P is a permutation matrix. In this way, A can be written as

$$R_1^{-1} \cdots R_k^{-1} P,$$

which represents a circuit where P is applied to the qubits first, and then the CNOT gates represented by $R_k^{-1}, \cdots, R_1^{-1}$. Since application of P is considered as "wiring", it is considered of zero cost.

Similarly, there are column-based greedy algorithms. A column-based greedy algorithms finds a list of matrices $C_1, \ldots, C_k \in \mathcal{C}$, one matrix per iteration, such that

$$AC_k \cdots C_2 C_1 = P.$$

In this way, A can be written as

$$PC_k^{-1} \cdots C_1^{-1}.$$

Withouht loss of generality, let us introduce row-based algorithms in more detail. We mentioned that a row-based algorithm finds $R_1, \ldots, R_k$ such that Equation 1 holds. To be more precise, the algorithm finds $L_1, \ldots, L_d \in \overline{S}$, such

Data: $\hat{A} \in \mathbb{F}_2^{n \times n}$
Result: $\mathcal{B} \in \overline{S} \cup \{\emptyset\}$

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

Algorithm 2: The layer-forming subroutine in row-based greedy algorithms (LF-row).

that $L_d \cdots L_1 A = P$. We call each L_i a layer of oprations. Since there are d layers, d will be the depth of the output circuit.

Algorithm 1 shows the pseudocode of a typical row-based algorithm. In each iteration of the while loop, a layer is formed by calling LF-row, except when $\mathcal{B} \neq \emptyset$, in which case an error is raised. LF-row which is called a layer-forming subroutine. The last layer is always formed by the first if statement in the loop. Given $\hat{A} \in \mathbb{F}_2^{n \times n}$, it is easy to derive $\mathcal{B} \in \overline{S}$ such that $\mathcal{B}\hat{A}$ is a permutation matrix, if such a $\mathcal{B}$ exists.

Forming One 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 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 opeartion 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.

It is easy to rewrite Algorithm 2 to obtain the corresponding layer-forming subroutine in column-based greedy algorithms (which we call LF-col).

3 Generalized Greedy Algorithms

This section presents our generalized greedy algorithms. Each of our algorithm aims to find $R_1, \ldots, R_k$ and $C_1, \ldots, C_{k'}$ such that

$$R_k \cdots R_1 A C_1 \cdots C_{k'} = P.$$

The circuit for A is thus represented by

$$R_1^{-1} \cdots R_k^{-1} P C_{k'}^{-1} \cdots C_1^{-1}.$$

To be more precise, each of our algorithms finds $L_1^{(0)}, \ldots, L_d^{(0)} \in \overline{S}$ and $L_1^{(1)}, \ldots, L_{d'}^{(1)} \in \overline{S}$, such that $L_d^{(0)} \cdots L_1^{(0)} A L_1^{(1)} \cdots L_{d'}^{(1)} = P$. Since there are $d + d'$ layers in total, $d + d'$ is the depth of the output circuit.

3.1 Choosing One of Two Layers

In our first generalized algorithm, to form a new layer, we do not always form a row layer as in row-based algorithms, or always form a column layer as in column-based algorithms. Instead, we generate a new row layer using LF-row, generate a new column layer using LF-col, and then choose one of the two layers as the new layer. Unsurprisingly, we pick the layer that reduces the cost more. 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 row layer and a new column layer at the same time. The way we generate the two layers is different from the first generalized algorithm. In the two-layer-forming subroutine, two buffers $\mathcal{B}^{(0)}$ and $\mathcal{B}^{(1)}$ are maintained, one for the row layer and one for the column layer. 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. The algorithm, along with the corresponding two-layer-forming subroutine, are presented in Algorithm 5 and Algorithm 6.

Data: $A \in \mathbb{F}_2^{n \times n}$
Result: $L_1^{(0)}, \ldots, L_d^{(0)} \in \overline{\mathcal{S}}$ for some d, $L_1^{(1)}, \ldots, L_{d'}^{(1)} \in \overline{\mathcal{S}}$ for some d', and a permutation matrix P, such that $L_d^{(0)} \cdots L_1^{(0)} A L_1^{(1)} \cdots L_{d'}^{(1)} = P$.

$\hat{A} \leftarrow A$;
$d \leftarrow 0$;
$d' \leftarrow 0$;
if *A is a permutation matrix* **then**
  return A;
end
while *True* **do**
  if *$\mathcal{B}\hat{A} = P'$ for some $\mathcal{B} \in \overline{\mathcal{S}}$ and permutation matrix P'* **then**
    $L_{d+1}^{(0)} \leftarrow \mathcal{B}$;
    $d \leftarrow d + 1$;
    $P \leftarrow P'$;
    break;
  end
  $\mathcal{B}, a \leftarrow$ LF-mixed-0($\hat{A}$) ; `/* LF-row is presented in Algorithm 4 */`
  if $\mathcal{B} \neq \emptyset$ **then**
    if $a = 0$ **then**
      $L_{d+1}^{(0)} \leftarrow \mathcal{B}$;
      $\hat{A} \leftarrow L_{d+1}^{(0)} \hat{A}$;
      $d \leftarrow d + 1$;
    end
    if $a = 1$ **then**
      $L_{d+1}^{(1)} \leftarrow \mathcal{B}$;
      $\hat{A} \leftarrow \hat{A} L_{d+1}^{(1)}$;
      $d' \leftarrow d' + 1$;
    end
  else
    `Error`;
  end
end
return $(L_1^{(0)}, \ldots, L_d^{(0)})$, $(L_1^{(1)}, \ldots, L_{d'}^{(1)})$, and P.

Algorithm 3: Our first generalized greedy algorithm

Data: $\hat{A} \in \mathbb{F}_2^{n \times n}$
Result: $\mathcal{B} \in \overline{\mathcal{S}} \cup \{\emptyset\}$ and an integer

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

Algorithm 4: The layer-forming subroutine for our first generalized algorithm (LF-mixed-0)

Data: $A \in \mathbb{F}_2^{n \times n}$
Result: $L_1^{(0)}, \ldots, L_d^{(0)} \in \overline{\mathcal{S}}$ for some d, $L_1^{(1)}, \ldots, L_{d'}^{(1)} \in \overline{\mathcal{S}}$ for some d', and a permutation matrix P, such that $L_d^{(0)} \cdots L_1^{(0)} A L_1^{(1)} \cdots L_{d'}^{(1)} = P$.

$\hat{A} \leftarrow A$;
$d \leftarrow 0$;
$d' \leftarrow 0$;
if *A is a permutation matrix* **then**
  return A;
end
while *True* **do**
  if *$\mathcal{B}\hat{A} = P'$ for some $\mathcal{B} \in \overline{\mathcal{S}}$ and permutation matrix P'* **then**
    $L_{d+1} \leftarrow \mathcal{B}$;
    $d \leftarrow d + 1$;
    $P \leftarrow P'$;
    break;
  end
  $\mathcal{B}^{(0)}, \mathcal{B}^{(1)} \leftarrow$ LF-mixed-1($\hat{A}$) ; `/* LF-row is presented in Algorithm 6 */`
  if $\mathcal{B}^{(0)} \neq \emptyset$ **then**
    $L_{d+1}^{(0)} \leftarrow \mathcal{B}^{(0)}$;
    $\hat{A} \leftarrow L_{d+1}^{(0)} \hat{A}$;
    $d \leftarrow d + 1$;
  end
  if $\mathcal{B}^{(1)} \neq \emptyset$ **then**
    $L_{d'+1}^{(1)} \leftarrow \mathcal{B}^{(1)}$;
    $\hat{A} \leftarrow \hat{A} L_{d'+1}^{(1)}$;
    $d' \leftarrow d' + 1$;
  end
  if $\mathcal{B}^{(0)} = \mathcal{B}^{(1)} = \emptyset$ **then**
    `Error`;
  end
end
return $(L_1^{(0)}, \ldots, L_d^{(0)})$, $(L_1^{(1)}, \ldots, L_{d'}^{(1)})$, and P.

Algorithm 5: Our second generalized greedy algorithm

Data: $\hat{A} \in \mathbb{F}_2^{n \times n}$
Result: $\mathcal{B}^{(0)}, \mathcal{B}^{(1)} \in \overline{\mathcal{S}} \cup \{\emptyset\}$

$\tilde{A} \leftarrow \hat{A}$;
$\mathcal{B}^{(0)} \leftarrow \emptyset$;
$\mathcal{B}^{(1)} \leftarrow \emptyset$;
while *True* **do**
  Compute $T^{(0)} \subset S$, the set of row operations that are independent of the ones in $\mathcal{B}^{(0)}$;
  Compute $T^{(1)} \subset S$, the set of column operations that are independent of the ones in $\mathcal{B}^{(1)}$;
  if $T^{(0)} = T^{(1)} = \emptyset$ **then**
    break;
  end
  if $T^{(0)} \neq \emptyset$ **then**
    $R \xleftarrow{\$} \left\{R \in T^{(0)} \mid c(R\tilde{A}) \leq c(R'\tilde{A}) \text{ for all } R' \in T^{(0)}\right\}$;
    $\alpha_0 = c(R\hat{A})$
  else
    $\alpha_0 = \infty$
  end
  if $T^{(1)} \neq \emptyset$ **then**
    $C \xleftarrow{\$} \left\{C \in T^{(1)} \mid c(\tilde{A}C) \leq c(\tilde{A}C') \text{ for all } C' \in T^{(1)}\right\}$;
    $\alpha_1 = c(\hat{A}C)$
  else
    $\alpha_1 = \infty$
  end
  if $\alpha_0 > c(\tilde{A})$ **and** $\alpha_1 > c(\tilde{A})$ **then**
    break;
  end
  if $\alpha_0 \leq \alpha_1$ **then**
    $\mathcal{B}^{(0)} \leftarrow \mathcal{B}^{(0)} \cup \{R\}$;
    $\tilde{A} \leftarrow R\tilde{A}$;
  else
    $\mathcal{B}^{(1)} \leftarrow \mathcal{B}^{(1)} \cup \{C\}$;
    $\tilde{A} \leftarrow \tilde{A}C$;
  end
end
return $\mathcal{B}^{(0)}, \mathcal{B}^{(1)}$;

Algorithm 6: The two-layer-forming subroutine for our second generalized algorithm (LF-mixed-1)