Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 291410, 8 pages
http://dx.doi.org/10.1155/2013/291410

Research Article

Efficient Algorithms for Optimal 4-Bit Reversible Logic System Synthesis

Zhiqiang Li,[1] Hanwu Chen,[2] Guowu Yang,[3] and Wenjie Liu[4]

[1] *College of Information Engineering, Yangzhou University, Yangzhou 225009, China*
[2] *School of Computer Science and Engineering, Southeast University, Nanjing 211189, China*
[3] *University of Electronic Science and Technology Chengdu, Sichuan, Chengdu 611731, China*
[4] *Nanjing University of Information Science & Technology, Nanjing 210044, China*

Correspondence should be addressed to Zhiqiang Li; yzqqlzq@163.com

Received 8 February 2013; Accepted 23 February 2013

Academic Editor: Xiaoyu Song

Owing to the exponential nature of the memory and run-time complexity, many methods can only synthesize 3-bit reversible circuits and cannot synthesize 4-bit reversible circuits well. We mainly absorb the ideas of our 3-bit synthesis algorithms based on hash table and present the efficient algorithms which can construct almost all optimal 4-bit reversible logic circuits with many types of gates and at mini-length cost based on constructing the shortest coding and the specific topological compression; thus, the lossless compression ratio of the space of n-bit circuits reaches near $2 \times n!$. This paper presents the first work to create all 3120218828 optimal 4-bit reversible circuits with up to 8 gates for the CNT (Controlled-NOT gate, NOT gate, and Toffoli gate) library, and it can quickly achieve 16 steps through specific cascading created circuits.

1. Introduction

Quantum computer is equivalent to quantum Turing machine, and quantum Turing machine is equivalent to a quantum logic circuit. Therefore, the quantum computer can be constructed by cascading and combining the quantum logical gates. Nowadays, many kinds of reversible quantum gates have been proposed, for example, CNOT gate [1], Toffoli gate, and Fredkin gate [2]. How to automatically construct the quantum circuit at small cost using given quantum gates? Several approaches for reversible logic circuit synthesis have been presented. Song et al. [3] presented algebraic characteristics of reversible gates. Iwama et al. [4] introduced transformation rules for CNOT-based circuits. Miller et al. [5] gave a synthesis method based on truth table and used template technology to simplify the circuit. Mishchenko and Perkowski [6] proposed a Reed Muller-based algorithm for optimizing quantum circuit. Gupta et al. [7] also gave a heuristic algorithm based on Reed Muller; Li et al. [8] proposed a general template algorithm. Shende et al. [9, 10] reduced the synthesis for reversible logic circuit to permutation and gave an effective recursive algorithm. Then, Yang et al. [11] reduced the synthesis for reversible logic circuit to group theory and presented a novel algorithm based on group-theory algebraic software GAP, while its performance was better than most others. Until today, we have not found a general and effective algorithm for multivariable quantum circuit synthesis. Owing to the exponential nature of the memory and run-time complexity, many existing methods can only synthesize 3-bit reversible circuits, and they perform only four steps for the CNP library in 4-bit circuit synthesis with mini-length for memory overflow [11–13], however, [14–16] are able to achieve 12 steps by using an enhanced bidirectional synthesis approach. We mainly absorb the ideas of our efficient 3-bit synthesis algorithms based on hash table and present the novel and efficient algorithms which can construct almost all optimal 4-bit reversible logic circuits [17]. Using lossless compression and cascading created circuits, our algorithms can get all the mini-length circuits whose lengths range from 0 to 8 with numbers 1, 28, 576, 9886, 147841, 1986374, 24375385, 274500662, and 2819198076, respectively; it can synthesize all the optimal 16-gate mini-length circuits for the CNT library.

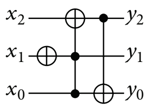


Figure 1: Quantum reversible logic circuit.

Table 1: True table.

Input				Output			
$\langle x_2, x_1, x_0\rangle_2$	x_2	x_1	x_0	$\langle y_2, y_1, y_0\rangle_2$	y_2	y_1	y_0
0	0	0	0	2	0	1	0
1	0	0	1	6	1	1	0
2	0	1	0	0	0	0	0
3	0	1	1	1	0	0	1
4	1	0	0	7	1	1	1
5	1	0	1	3	0	1	1
6	1	1	0	5	1	0	1
7	1	1	1	4	1	0	0

2. Background

Definition 1. Let M be a finite set; then, a permutation of M is a bijection $\sigma : M \rightarrow M$. If $|M| = m$, σ is an m permutation. Normally, the permutation of group theory begins from one, but our program in c++ can work well if the permutation contains zero. Let $M = \{0, 1, \ldots, m-1\}$, and a permutation $\sigma = \left(\begin{smallmatrix} 0 & 1 & \cdots & m-1 \\ p_0 & p_1 & \cdots & p_{m-1} \end{smallmatrix}\right) = (p_0, p_1, \ldots, p_{m-1})$. It can be denoted as the cyclic form; for example, $\tau = (a_1 a_2 \cdots a_i)$, $i \leqslant m$; that is, $a_1 \mapsto a_2 \cdots \mapsto a_i \mapsto a_1$, $\sigma^{-1} = \left(\begin{smallmatrix} p_0 & p_1 & \cdots & p_{m-1} \\ 0 & 1 & \cdots & m-1 \end{smallmatrix}\right)$, and $\pi_e = \left(\begin{smallmatrix} 0 & 1 & \cdots & m-1 \\ 0 & 1 & \cdots & m-1 \end{smallmatrix}\right)$ is the identity permutation of M, such that $\pi_e \circ \sigma = \sigma \circ \pi_e = \sigma$ for any permutation σ of M.

The reversible function can be described by permutation or truth table. The 3-bit reversible logic circuit in Figure 1 can be described by the permutation $\sigma = \left(\begin{smallmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 6 & 0 & 1 & 7 & 3 & 5 & 4 \end{smallmatrix}\right) = (2, 6, 0, 1, 7, 3, 5, 4) = (0\ \ 2)(1\ \ 6\ \ 5\ \ 3)(4\ \ 7)$ or the truth table in Table 1, where the input or output is $\langle x_{n-1}, \ldots, x_1, x_0\rangle_2 = \sum_{i=0}^{n-1}(x_i \cdot 2^i)$ and $\langle y_{n-1}, \ldots, y_1, y_0\rangle_2 = \sum_{i=0}^{n-1}(y_i \cdot 2^i)$, respectively. Let the permutations of NOT gate, Toffoli gate, and CNOT gate be σ_1, σ_2, σ_3; then, $\sigma = \sigma_1 \circ \sigma_2 \circ \sigma_3 = \sigma_1\sigma_2\sigma_3$.

In order not to repeat explanation, the following definitions are given.

(1) Let C be any n-bit reversible logic circuit with length l; it is a cascade of quantum gates $g_1, g_2, \ldots, g_l$, denoted by $C = g_1 g_2 \cdots g_l$.

(2) Let $SF = \{1, 2, \ldots, n!\}$, $SM = \{1, 2, \ldots, m\}$, and $SB = \{0, 1\}$.

(3) For permutation $\sigma = (0, 1, \ldots, n-1)$, there are $n!$ kinds of permutations, which are denoted by $\sigma_1, \sigma_1, \ldots, \sigma_{n!}$, respectively.

(4) Let $\pi(g)$ be the permutation of quantum gate g; then, the permutation of circuit C is $\pi(C) \equiv \pi(g_1 g_2 \cdots g_l) = \pi(g_1)\pi(g_2) \cdots \pi(g_l)$.

(5) Let cost(C) be the cost of quantum circuit C and cost(g) be the cost of quantum gate g.

(6) All the circuits in the paper are the reversible logic circuits.

Because permutation is often used in our algorithms, the effective expression of permutation is very important to improve the algorithms. Thus, we present the shortest coding scheme with saving plenty of space, which maps a permutation to an integer with the fewest bytes.

Definition 2. Let O_p^i be ordinal number of p_i in sequence $p_0, p_1, \ldots, p_i, \ldots, p_{2^n-1}$; then, it is the number of members which are both smaller than and before p_i; that is,

$$O_p^i = \sum_{j=0}^{i-1} \operatorname{sgn}\left(p_i - p_j\right), \qquad \operatorname{sgn}(x) = \begin{cases} 1, & (x > 0), \\ 0, & \text{else}. \end{cases} \tag{1}$$

Ordinal numbers of all members in the sequence $p_0, p_1, \ldots, p_i, \ldots, p_{2^n-1}$ construct ordinal number sequence: $(O_p^0, O_p^1, \ldots, O_p^{2^n-1})$, and obviously $O_p^0 \equiv 0$ and $O_p^{2^n-1} \equiv P_{2^n-1}$. For example, the ordinal number sequence of $\sigma = (2, 6, 0, 1, 7, 3, 5, 4)$ is $(0, 1, 0, 1, 4, 3, 4, 4)$.

Lemma 3. *The ordinal number of p_i in Definition 2 can also be defined as*

$$O_p^i = p_i - \sum_{j=i+1}^{2^n-1} \operatorname{sgn}\left(p_i - p_j\right). \tag{2}$$

Proof. From (1), O_p^i is the number of members which are both smaller than and before p_i, and $\sum_{j=i+1}^{2^n-1} \operatorname{sgn}(p_i - p_j)$ is the number of members which are both smaller than and after p_i; so, the sum of them is the number of members which are smaller than p_i in the sequence $O_p^i + \sum_{j=i+1}^{2^n-1} \operatorname{sgn}(p_i - p_j) = |\{0, 1, \ldots, p_i - 1\}| = p_i$. □

Theorem 4. *One can get the ordinal number sequence in Definition 2 by $2^{2n-2} - 2^{n-1}$ comparison times merely.*

Proof. The comparison times using (1) and Lemma 3 are i and $2^n - 1 - i$ for computing O_p^i, respectively, and comparison times are all $\sum_{i=1}^{2^n-1} i = \sum_{i=1}^{2^n-1}(2^n - 1 - i) = 2^{2n-1} - 2^{n-1}$ for computing ordinal number sequence. These two formulas can be chosen depending on condition to reduce the comparisons times. We use (1) if $i < 2^n - 1 - i$; that is, $i \leq 2^{n-1} - 1$, and use Lemma 3 if $i > 2^n - 1 - i$; that is, $i \geq 2^{n-1}$. So, our method is that using (1) when p_i is in the first half part of the sequence, and otherwise using Lemma 3, their comparison times are $\sum_{i=1}^{2^{n-1}-1} i$ and $\sum_{i=2^{n-1}}^{2^n-1}(2^n - 1 - i)$, respectively. Finally, the whole comparison times are $\sum_{i=1}^{2^{n-1}-1} i + \sum_{i=2^{n-1}}^{2^n-1}(2^n - 1 - i) = 2^{2n-2} - 2^{n-1}$; it is lesser than the half of $2^{2n-1} - 2^{n-1}$. □

Definition 5. The shortest coding of the permutation $(p_0, p_1, \ldots, p_{2^n-1})$ is $\text{Code}(p_0, p_1, \ldots, p_{2^n-1}) = \sum_{i=1}^{2^n-1}(O_p^i \cdot i!)$.

Lemma 6. *If the ordinal number sequences of two permutations are the same, then the two permutations must be also the same.*

Proof. Suppose that we get any two permutations of the set $S = \{0, 1, \ldots, 2^n - 1\}$, $P = (p_0, p_1, \ldots, p_{2^n-1})$, and $Q = (q_0, q_1, \ldots, q_{2^n-1})$; if their ordinal number sequences are all $B = (b_0, b_1, \ldots, b_{2^n-1})$, then we will prove that $P = Q$; namely, for all $i \in \{0, 1, \ldots, 2^n - 1\}$, $p_{2^n-1-i} = q_{2^n-1-i}$.

Proof by Mathematical Induction

Basis. Clearly, $O_p^{2^n-1} = b_{2^n-1}$, $O_q^{2^n-1} = b_{2^n-1}$; then, P has b_{2^n-1} elements which are lesser than p_{2^n-1}; that is, p_{2^n-1} is $(b_{2^n-1} + 1)$th small element in S. Similarly, q_{2^n-1} is $(b_{2^n-1} + 1)$th small element in S, and so $p_{2^n-1} = q_{2^n-1} = b_{2^n-1}$; that is, $p_{2^n-1-i} = q_{2^n-1-i}$ for $i = 0$.

Inductive Assumption. Let $p_{2^n-1-i} = q_{2^n-1-i}$ for $i = 0, 1, \ldots, j$.

Inductive Step. When $i = j + 1$, $O_p^{2^n-2-j} = b_{2^n-2-j}$, and P has b_{2^n-2-j} elements which are both lesser than and before p_{2^n-2-j}; that is, p_{2^n-2-j} is $(b_{2^n-2-j} + 1)$th small element in $\{p_0, p_1, \ldots, p_{2^n-2-j}\}$, and let $T_p = \{p_{2^n-1-j}, p_{2^n-j}, \ldots, p_{2^n-1}\}$, $T_q = \{q_{2^n-1-j}, q_{2^n-j}, \ldots, q_{2^n-1}\}$; then $\{p_0, p_1, \ldots, p_{2^n-2-j}\} = S - T_p$; that is, p_{2^n-2-j} is $(b_{2^n-1-j} + 1)$th small element in $S - T_p$. Similarly, q_{2^n-2-j} is $(b_{2^n-1-j} + 1)$th small element in $S - T_q$. Now, using the inductive assumption, we get $T_p = T_q$; that is, $S - T_p = S - T_q$, and so $p_{2^n-2-j} = q_{2^n-2-j}$; thus, $p_{2^n-1-i} = q_{2^n-1-i}$ for $i = j + 1$. □

Lemma 7. *The coding function Code maps each ordinal number sequence to a distinct coding.*

Proof. Let $c = \text{Code}(p_0, p_1, \ldots, p_{2^n-1}) = \sum_{i=1}^{2^n-1} O_p^i \cdot i!$; that is, the function Code maps ordinal number sequence $B = (O_p^0, O_p^1, \ldots, O_p^{2^n-1})$ to coding c, and so the remainder is O_p^1 when c is divided by 2, the quotient is $\sum_{i=2}^{2^n-1}(O_p^i \cdot i!/2)$, and the remainder is O_p^2 when this quotient is divided by 3. Follow the same steps; the recursive formula is $n_1 = c$, $n_{i+1} = \lfloor n_i/(i+1) \rfloor$, $O_p^i = n_i - (i + 1)n_{i+1}$, $i \in \{1, 2, \ldots, 2^n - 1\}$. Clearly, the coding function Code maps each ordinal number sequence to a distinct coding, and vice versa; so, it is a one-to-one correspondence between each coding and ordinal number sequence. □

Theorem 8. *The function Code is the shortest coding function.*

Proof. 2^n different elements in $\{0, 1, \ldots, 2^n - 1\}$ totally have $2^n!$ permutations. According to Lemma 6, two different ordinal number sequences corresponding to two permutations must be different, and according to Lemma 7, coding function Code maps each ordinal number sequence to a distinct coding. So, the codings of all permutations are different.

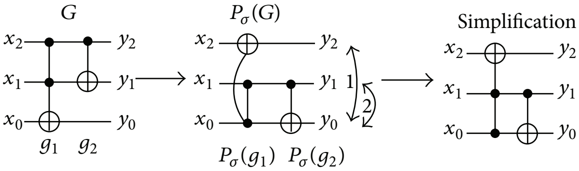


FIGURE 2: The line permutation of quantum circuit.

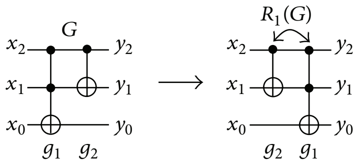


FIGURE 3: The direction transform of quantum circuit.

By Definition 5, $\max(O_p^i) = i$, $\min(O_p^i) = 0$, $\min(H(p_0, p_1, \ldots, p_{2^n-2}, p_{2^n-1})) = \min(\sum_{i=1}^{2^n-1}(O_p^i \cdot i!)) = \sum_{i=1}^{2^n-1}(\min(O_p^i) \cdot i!) = \sum_{i=1}^{2^n-1}(0 \cdot i!) = 0$, $\max(H(p_0, p_1, \ldots, p_{2^n-2}, p_{2^n-1})) = \max(\sum_{i=1}^{2^n-1}(O_p^i \cdot i!)) = \sum_{i=1}^{2^n-1}(\max(O_p^i) \cdot i!) = \sum_{i=1}^{2^n-1}(i \cdot i!) = 2^n! - 1$. So, the number of all codings is $L = \text{MAX}\, H - \text{MIN}\, H + 1 = 2^n! - 1 - 0 + 1 = 2^n!$ $(n > 0)$. There are 2^n different elements in $\{0, 1, \ldots, 2^n - 1\}$ corresponding to $2^n!$ different coding, and thus the minimum number of all the codings is $2^n!$. Therefore, the function Code is the shortest coding function. □

The subject of topology is concerned with those features of geometry which remain unchanged after twisting, stretching, or other deformations of a geometrical space; any continuous change which can be continuously undone is allowed, but cannot be broken.

Definition 9. Line permutation is the operation which permutes quantum lines in quantum circuit without break. Obviously, it is a specific topological transformation. Let $P_\sigma(g)$ be quantum gate g being operated by line permutation σ. The circuit $G = g_1 g_2$, operated by line permutation σ, is $P_\sigma(G) = P_\sigma(g_1 g_2) = P_\sigma(g_1)P_\sigma(g_2)$.

For example, in Figure 2, quantum circuit G is operated by line permutation σ, where $\sigma = \left(\begin{smallmatrix} 0 & 1 & 2 \\ 2 & 0 & 1 \end{smallmatrix}\right) = (0 \quad 2)(0 \quad 1)$.

The circuit after applying topological transformation can be simplified, because the gate's function cannot be changed when the order of its same kind of points is changed.

Consider $P_\sigma(C) = P_\sigma(g_1 g_2 \cdots g_l) = P_\sigma(g_1)P_\sigma(g_2) \cdots P_\sigma(g_l)$. If there are m basal gates in quantum gate library L, $g_1, g_2, \ldots$, and g_m, then $L = \bigcup_{j \in SF, i \in SM} P_{\sigma_j(g_i)}$, and the set of all permutations of the gates in L is $\sigma = \bigcup_{j \in SF, i \in SM} P_{\sigma_j}(\pi(g_i))$.

The set of all circuits of S operated by line permutation σ is denoted by $P_\sigma(S) = \{P_\sigma(G) \mid G \in S\}$.

Definition 10. Direction transform (Figure 3) has two kinds of operations, obverse direction transform and reverse direction transform. It is also a specific topological transformation;

the obverse direction transform b of gate g is denoted by $R_b(g)$. Thus, $R_0(g) = g$, $R_1(g) = g^{-1}$. Gate g is symmetric, if $g = g^{-1}$, and otherwise it is asymmetric. Let circuit $G = g_1g_2$, and the two gates are symmetric; then, $R_1(G) = R_1(g_1g_2) = g_2^{-1}g_1^{-1} = g_2g_1$.

Consider $R_1(C) = R_1(g_1g_2\cdots g_l) = (g_1g_2\cdots g_l)^{-1} = g_l^{-1}g_{l-1}^{-1}\cdots g_2^{-1}g_1^{-1}$. $R_1(C) = g_lg_{l-1}\cdots g_2g_1$, if all the gates in C are symmetric. Direction transform can be used in our synthesis algorithms only if all quantum gates are symmetric or their inverse gates are all in the quantum gate library.

The set of all circuits of S operated by direction transform b is denoted by $R_b(S) = \{R_b(G) \mid G \in S\}$.

From Definitions 9 and 10, the following properties can be gotten.

Property 1. $P_\sigma(P_\tau(C)) = P_{\sigma\circ\tau}(C)$ and $P_\sigma(P_{\sigma^{-1}}(C)) = P_{\sigma^{-1}}(P_\sigma(C)) = C$, but $P_\sigma(P_\tau(C)) \neq P_\tau(P_\sigma(C))$, where $P_\sigma(P_\tau(C))$ denotes the circuit C being operated by line permutation τ and line permutation σ; so, $P_\sigma(P_\tau(C)) = P_{\tau\circ\sigma}(C)$. $\sigma\circ\sigma^{-1} = \sigma^{-1}\circ\sigma = \pi_e$, and thus $P_{\sigma^{-1}}(P_\sigma(C)) = P_\sigma(P_{\sigma^{-1}}(C)) = P_{\pi_e}(C) = C$. Normally, $\tau\circ\sigma \neq \sigma\circ\tau$; hence, $P_\sigma(P_\tau(C)) \neq P_\tau(P_\sigma(C))$.

Property 2. $R_{b1}(R_{b2}(C)) = R_{b2}(R_{b1}(C)) = R_{b1\oplus b2}(C)$, where $R_{b1}(R_{b2}(C))$ denotes the circuit C being operated by direction transform $b1$ and direction transform $b2$. The circuit is not changed, if $b1 = b2$, and otherwise, it is operated by reverse direction transform; thus, $R_{b1}(R_{b2}(C)) = R_{b1\oplus b2}(C)$. In the same way, $R_{b2}(R_{b1}(C)) = R_{b1\oplus b2}(C)$.

Property 3. $R_b(P_\sigma(C)) = P_\sigma(R_b(C))$, where $R_b(P_\sigma(C))$ denotes the circuit C being operated by line permutation σ and direction transform b, and $P_\sigma(R_b(C))$ denotes that the circuit C is operated by direction transform b and line permutation σ. Line permutation is vertical transform, and direction transform is horizontal transform; so, the order of the two operations does not affect their compound function, and thus $R_b(P_\sigma(C)) = P_\sigma(R_b(C))$.

We show by Properties 1, 2, and 3 that the order of the previous topological transformations except the line permutations in quantum circuits can not affect their compound functions.

Lemma 11. *For all $k \in SF$, for all $b \in SB$, cost(C) = cost$(R_b(P_{\sigma_k}(C)))$.*

Proof. Consider any quantum gate g, operated by any line permutation σ_k, whose type is not changed so, $\mathrm{cost}(P_{\sigma_k}(g)) = \mathrm{cost}(g)$. When any quantum circuit C is operated by any direction transform, its direction may be changed, but its cost can not be changed; thus, for all $b \in SB$, $\mathrm{cost}(R_b(C)) = \mathrm{cost}(C)$, and then $\mathrm{cost}(R_b(P_\sigma(C))) = \mathrm{cost}(P_\sigma(C)) = \sum_{i=1}^{l}\mathrm{cost}(P_\sigma(g_i)) = \sum_{i=1}^{l}\mathrm{cost}(g_i) = \mathrm{cost}(C)$, where $\mathrm{cost}(C)$ of optimal C should be minimal with the cost function. If for all $i \in \{1, 2, \ldots, l\}$, $\mathrm{cost}(g_i) = 1$, then $\mathrm{cost}(C) = l$; that is, the length of C is minimal, and thus minimum cost standard degenerates to mini-length standard. □

Lemma 12. *Any optimal circuit after any specific topological transformation must be optimal.*

Proof. Let the circuit C be optimal, and circuit $D = R_b(P_\sigma(C))$. Assume that circuit D is not optimal, and so there is an optimal circuit E with $\pi(E) = \pi(D)$ and $\mathrm{cost}(E) < \mathrm{cost}(D)$; thus, $\pi(R_b(P_{\sigma^{-1}}(E))) = \pi(R_b(P_{\sigma^{-1}}(D))) = \pi(R_b(P_{\sigma^{-1}}(R_b(P_\sigma(C))))) = \pi(C)$. Let circuit $F = R_b(P_{\sigma^{-1}}(E))$; by the Lemma 11, there is $\pi(F) = \pi(C)$, and $\mathrm{cost}(F) = \mathrm{cost}(E)$, $\mathrm{cost}(D) = \mathrm{cost}(C)$. Thus, $\mathrm{cost}(F) < \mathrm{cost}(C)$; that is, the functions of circuit F and circuit C are the same, but F is more optimal than C in contradiction to C being optimal. □

Lemma 13. *By Definition 9, for all $k \in SF$, $P_{\sigma_k}(L) = L$.*

Proof. By permutation group theory, for all $i, j \in SF$, $\sigma_i\sigma_j \in \{\sigma_1, \sigma_2, \ldots, \sigma_{n!}\}$, and $i \neq j \rightarrow \sigma_i \neq \sigma_j$, and thus for all $k \in SF$, $\sigma_i\sigma_k \neq \sigma_j\sigma_k$, $|\{\sigma_1\sigma_k, \sigma_2\sigma_k, \ldots, \sigma_{n!}\sigma_k\}| = n!$, $\{\sigma_1\sigma_k, \sigma_2\sigma_k, \ldots, \sigma_{n!}\sigma_k\} = \{\sigma_1, \sigma_2, \ldots, \sigma_{n!}\}$. Similarly, we have

$$\forall k \in SF, \quad \{\sigma_k\sigma_1, \sigma_k\sigma_2, \ldots, \sigma_k\sigma_{n!}\} = \{\sigma_1, \sigma_2, \ldots, \sigma_{n!}\}. \tag{3}$$

For all $k \in SF$, $L_{\text{new}} = P_{\sigma_k}(L) = P_{\sigma_k}(\bigcup_{j\in SF, i\in SM}P_{\sigma_j}(g_i)) = \bigcup_{j\in SF, i\in SM}P_{\sigma_j\sigma_k}(g_i) \overset{(3)}{=} \bigcup_{i\in SM}(\bigcup_{j\in SF}P_{\sigma_j}(g_i)) = \bigcup_{j\in SF, i\in SM}P_{\sigma_j}(g_i) = L$. □

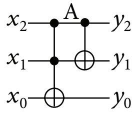
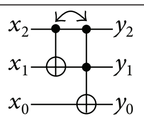
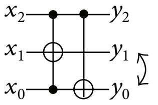
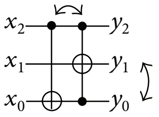
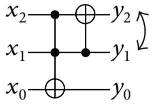
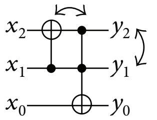
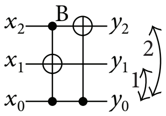
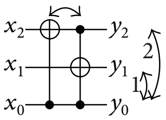
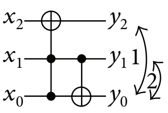
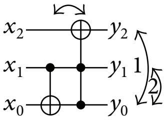
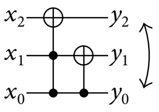
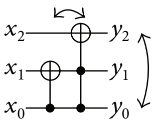
Definition 14. Let G be a circuit, $\mathrm{Min}(\pi(G)) = \min\{R_b(P_{\sigma_j}(\pi(G))) \mid j \in SF, b \in SB\}$, and thus $\exists k \in SF$, $\exists c \in SB$, $R_c(P_{\sigma_k}(\tau)) = \mathrm{Min}(\pi(G))$. Let $\mathrm{Min}(G) = R_c(P_{\sigma_k}(G))$; $\mathrm{Min}(G)$ is the minimal permutation circuit of G. All circuits gotten by all the specific topological transformations of G compose the set $S = \bigcup_{j\in SF, b\in SB}R_b(P_{\sigma_j}(G))$. The function GetMin$(\tau, \sigma, b)$ returns the minimal permutation π, line permutation σ and direction transform b, where $\pi = R_b(P_\sigma(\tau)) = \mathrm{Min}(\tau)$.

Theorem 15. *From Definition 14,* Min(G) *is a circuit of S, and S can be gotten only by* Min(G)*; that is, the lossless compression ratio of the space is near $2 \times n!$.*

Proof. Obviously, all the minimal permutation circuits of S must be $\mathrm{Min}(G)$, $S = \bigcup_{j\in SF, b\in SB}R_b(P_{\sigma_j}(G))$, $\mathrm{Min}(G) = R_c(P_{\sigma_k}(G))$. Let S' be the set of all circuits gotten by all the specific topological transformations of $\mathrm{Min}(G)$; thus, $S' = \bigcup_{j\in SF, b\in SB}R_b(P_{\sigma_j}(\mathrm{Min}(G))) = \bigcup_{j\in SF, b\in SB}R_b(P_{\sigma_j}(R_c(P_{\sigma_k}(G)))) \overset{\text{Property 3}}{=} \bigcup_{j\in SF, b\in SB}R_{b\oplus c}(P_{\sigma_k\sigma_j}(G)) \overset{(3)}{=} \bigcup_{j\in SF, b\in SB}R_b(P_{\sigma_j}(G)) = S$, where $|SF| = n!$, $|SB| = 2$; that is, $|SF| \times |SB| = 2 \times n!$, and few specific topological transformation circuits may be the same. Thus $|S| \leq 2 \times n!$ and $|S| \approx 2 \times n!$. Then, the lossless compression ratio of the set S is near $2 \times n!$. □

Let $L_{n,G}$ be an n-bit quantum gate library with gates G, and $T(L_{n,G})$ a set of all n-bit reversible circuits synthesized by any gates in $L_{n,G}$. For example, $2 \times n!|_{n=4} = 48$. The average compression ratio of $T(L_{4,\mathrm{CNT}})$, whose circuits are synthesized 8 layers at minimal cost, is 47.95.

Table 2: All the specific topological transformations of 3-bit logic circuit.

Line permutation	Obverse direction	Coding	Reverse direction	Coding
$\begin{pmatrix}0&1&2\\0&1&2\end{pmatrix}=\pi_e$		23759		28679
$\begin{pmatrix}0&1&2\\1&0&2\end{pmatrix}=\begin{pmatrix}0&1\end{pmatrix}$		25199		34439
$\begin{pmatrix}0&1&2\\0&2&1\end{pmatrix}=\begin{pmatrix}1&2\end{pmatrix}$		11951		16985
$\begin{pmatrix}0&1&2\\1&2&0\end{pmatrix}=\begin{pmatrix}0&1\end{pmatrix}\begin{pmatrix}0&2\end{pmatrix}$	 B x_2 x_1 x_0 y_2 y_1 y_0 2 1	8949	 x_2 x_1 x_0 y_2 y_1 y_0 2 1	18903
$\begin{pmatrix}0&1&2\\2&0&1\end{pmatrix}=\begin{pmatrix}0&2\end{pmatrix}\begin{pmatrix}0&1\end{pmatrix}$	 x_2 x_1 x_0 y_2 y_1 y_0 1 2	14975	 x_2 x_1 x_0 y_2 y_1 y_0 1 2	32969
$\begin{pmatrix}0&1&2\\2&1&0\end{pmatrix}=\begin{pmatrix}0&2\end{pmatrix}$		9333		29241

Given two quantum logic circuits, D and G, $\exists h, j \in SF$, $\exists b, c \in SB$, $R_b(P_{\sigma_h}(G)) = R_c(P_{\sigma_j}(D)) \rightarrow G, D \in S \rightarrow \mathrm{Min}(D) = \mathrm{Min}(G)$.

Taking all specific topological transformations of a 3-bit circuit as an example, there are $2 \times n!|_{n=3} = 12$ kinds of transformations. In Table 2, the bidirectional arrow expresses exchanging the two quantum lines, and the minimal permutation circuit of all circuits is circuit B by computing coding.

The way of computing the shortest coding of circuit A in Table 2 is given.

(1) By Definition 1, the permutation of circuit A is $\sigma = (0, 1, 2, 3, 6, 7, 5, 4)$.

(2) By Definition 2, the ordinal number sequence of σ is $(0, 1, 2, 3, 4, 5, 4, 4)$.

(3) By Definition 5, the shortest coding is 23759.

Definition 16. Given two circuit sets S and Q, the set of all circuits of S cascaded by all circuits of Q is denoted by $S \times Q = \{G_1G_2 \mid G_1 \in S \wedge G_2 \in Q\}$.

Lemma 17. *Given two circuit sets S and Q,* $P_\sigma(S \times Q) = P_\sigma(S) \times p_\sigma(Q)$.

Proof. By Definition 16, $S \times Q = \{G_1G_2 \mid G_1 \in S \wedge G_2 \in Q\}$, thus $P_\sigma(S \times Q) = P_\sigma(\{G_1G_2 \mid G_1 \in S \wedge G_2 \in Q\}) = \{P_\sigma(G_1G_2) \mid G_1 \in S \wedge G_2 \in Q\} = \{P_\sigma(G_1)P_\sigma(G_2) \mid P_\sigma(G_1) \in P_\sigma(S) \wedge P_\sigma(G_2) \in P_\sigma(Q)\} = P_\sigma(S) \times P_\sigma(Q)$. □

Theorem 18. *Given the set* S_l *of all optimal circuits with length* l, $\mathrm{Min}(S_{l+1}) = \mathrm{Min}((\bigcup_{b\in SB} R_b(\mathrm{Min}(S_l))) \times L) = \mathrm{Min}(S_l \times L)$.

For saving memory, we only store minimal permutation circuits $\mathrm{Min}(S_l)$. The common computing method is that, firstly, S_l can be gotten by decompressing $\mathrm{Min}(S_l)$, secondly, S_{l+1} can be gotten by $S_l \times L$, and lastly, Min (S_{l+1}) can be gotten by compressing $S_l \times L$. Its number of decompressing circuit is $2 \times n! \times |\mathrm{Min}(S_l)|$, but ours is zero based on Theorem 15. The number of cascading circuit and the number of compressing circuit in the common method all are $n!$ times than ours; so, our method is better.

3. New Synthesis Algorithm

A quantum logic gate realizes certain permutation in essence; quantum circuit is the cascade of some quantum gates, and so the basic function of quantum circuit can be represented the multiplication of permutations. The basic idea of the hash-based 3-bit synthesis algorithm which we previously

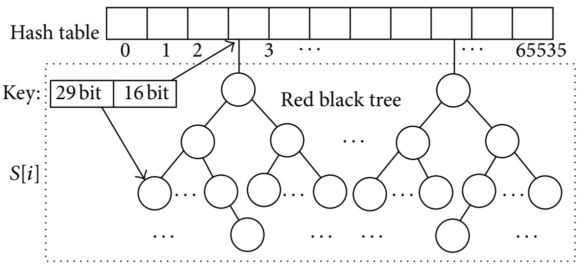


Figure 4: The data structure of the minimal permutation circuit with length i.

advanced is constructing an optimal circuit using WFS, making a one-to-one correspondence between the elements in the hash table and different circuits. Thus, we need only one step to check whether the circuit of the same function has already been found to decide whether it is optimal. But it is unfeasible if we use this algorithm directly in 4-bit circuit synthesis, since the length of the hash table is at least $n^2!$; so, it is extremely memory consuming. Otherwise, if we use DFS, it will be meaningless that the algorithm runs too slowly or can not realize optimal. To sum up, we adopt the BFS, with higher speed and more synthesis capability, and some lossless data compression methods to reduce memory consumption.

(1) Represent the permutation using the shortest encoding. Only $\lceil \log_2^{2^n!}|_{n=4} \rceil = 45$ bits is required to represent 4-bit circuit permutations using the permutation encoding method in Definition 5, rather than 64 bits as usual.

(2) Compress the optimal circuits without loss. Using line permutation alone, nearly $n!|_{n=4} = 24$ times compression can be reached, while using both line permutation and direction transformation, nearly $n! \times 2|_{n=4} = 48$ times compression can be reached, which is the backbone of this paper.

(3) Use hash table with length 2^{16} in the top of the data structure, with each element pointing to a different RB tree in Figure 4 and each node in the tree reduces 2 bytes. Hash table and RB tree always remain effective, and RB tree can also allocate the memory dynamically. Therefore the data structure can not only permit a faster access speed, but also save memory space.

How to determine the length of hash table? Let the permutation of n-bit circuit be π, whose shortest coding is $\text{Code}(\text{Min}(\pi))$, $\{0,1\}^{\lceil \log_2^{n!} \rceil}$ in binary. Use the rear $k \times 8$ bits as the hash address, the rest $\lceil \log_2^{n!} \rceil - k \times 8$ bits as the value of the nodes in the RB tree. Suppose that the length of the elements in hash table is j bytes; so, hash table uses $j \times 2^{8k}$ bytes while the RB tree saves $k \times |S[i]|$ bytes, and the maximum space this structure saves is max $fm(k, j, i) = k \times |S[i]| - j \times 2^{8k}$. Take the 4-bit circuit based on CNT quantum gate library as an example; each element in hash table is a pointer pointing to the different tree, taking 4 bytes. The number of minimum permutation circuits with length 8 is 58777916, when $k = 2$, and the maximum of the function is $fm(2, 4, 8) = 117293688$; so, the optimal length of hash table is $2^{8k}|_{k=2} = 65536$.

In this paper, we use BFS to get all the first N layers of optimal circuits. If the gate library is determined, then the optimal circuits are determined. In order to save time, we store all minimum permutation circuits in the first N layers in a file. If CNT gate library is used, then $N = 8$, and we get a 700 MB file. As there are considerable gates in the 4-bit circuit quantum gate library, thus lots of optimal circuits are generated at each length. When a circuit reaches a certain length, the memory will overflow, and so N has practical upper lower limit related to memory. After the lossless compression mentioned earlier, only the minimum permutation circuits are stored, saving 47.95 times less of circuits. The length of the overall synthesis circuits reaches 8 instead of 6, 117.7 times larger; the synthesizable length grows form 12 to 16 [14].

3.1. Minimal Length Algorithm. The node type of our RB tree is defined as follows:

```
struct rbtnode {
        gate; // the gate is in the end of the circuit G
        cpm; // the permutation of the circuit Min(G)
        binv; // the current circuit is inverted or not
        lnpm; // Min(π(G)) ≡ GetMin(π(G); binv; lnpm)
        pcpm; // the permutation of the previous circuit
        pbinv; // the previous circuit is inverted or not
}
```

To enhance readability, two ways are used to simplify the algorithms.

(1) All permutations are not transformed into the relevant shortest codings.

(2) The hash table in the top of Figure 4 is omitted, and all the minimum permutation circuits in each layer are only saved in a RB tree; for example, $S[i]$ is the RB tree which has saved all minimum permutation circuits with length i.

3.2. Algorithm for the Minimal Permutation Quantum Reversible Logic Circuits Representation in QML. For more details, see Algorithm 2.

3.3. General Algorithm for the Quantum Reversible Logic Circuits Representation in QML (Algorithm 1). For more details, see Algorithm 3.

4. Experimental Results

Our experiments were conducted using many benchmark functions for 4-bit reversible logic circuits synthesis, and [14, 15] dealt with the synthesis of 4-qubit circuit for the high complexity of the algorithm. Based on CNP quantum gate library, using the mini-length as criteria, [14] added 4 layers

```
Input: Quantum Gate Library $L$
Output: $\max l, N[0\ldots\max l], S[0\ldots\max l]$
1: $S[0] = \{\mathrm{cmp} : \pi_e\}, j = 0, N[0] = 1$
2: while $N[j] \neq 0$ do
3:    $j = j + 1, S[j] = \emptyset$
4:    for each node Node $x$ in $S[j-1]$ do
5:       $c = \mathrm{Node}x.\mathrm{cpm}$
6:       for $v = 0$ to $1$ do
7:          if $v = 1$ then
8:             $c = c^{-1}$
9:          end if
10:         for each gate $g$ in $L$ do
11:            $p = c \circ \pi(g), \pi = \mathrm{GetMin}(p, \sigma, b)$
12:            if $\pi \notin \bigcup_{i=0}^{j} S[i]$ then
13:               $S[j] = S[j] \cup \{(\mathrm{gate} : g, \mathrm{cpm} : \pi, \mathrm{lnpm} : \sigma, \mathrm{binv} : b, \mathrm{pcpm} : p, \mathrm{pbinv} : v)\}$
14:               if memory overflow error then
15:                  free $S[j]$, $j = j - 1$, go to 23
16:               end if
17:            end if
18:         end for
19:      end for
20:   end for
21:   $N[j] = \sum_{G\in S[j]} \left|\bigcup_{i\in SF, b\in SB} R_b(P_{\sigma_i}(G))\right|$
22: end while
23: $\max l = j$
```

Algorithm 1: Quantum_Minimum_Length (QML).

```
Input: Quantum Gate Library $L$, minimal permutation $p$,
       circuit length $l$
Output: Circuit of minimal permutation $p$ with mini-length
1: compute $S[0\ldots\max l]$ as in QML($L$) for the first time;
2: $i = l$, mynode $[i] = s\,[i]$.find($p$)
3: pcpm $x$ = mynode $[i]$.pcpm
4: while $i > 1$ do
5:    $i = i - 1$
6:    mynode $[i] = S\,[i]$.find(pcpm $x$)
7:    pcpm $x$ = mynode $[i]$.pcpm
8: end while
9: $\sigma$ = mynode [1].lnpm, $b$ = mynode [1].binv
10: $G_1 = P_\sigma R_b$(mynode [1].gate)
11: for $i = 2$ to $l$ do
12:    $\sigma$ = mynode $[i]$.lnpm, $b$ = mynode $[i]$.binv
13:    $c$ = mynode $[i]$.pbinv, $g$ = mynode $[i]$.gate
14:    $G_i = R_b\left(P_\sigma\left(R_c\left(G_{i-1}\right) g\right)\right)$
15: end for
16: return $G_l$
```

Algorithm 2: Minimal_Quantum_Representation (MQR).

```
Input: Quantum Gate Library $L$, permutation $p$
Output: Circuit of permutation $p$ with mini-length
1: if $\exists l \in \{0, 1, 2, \ldots, \max l\}$, GetMin$(p, \sigma, b) \in S[l]$ then
2:    $G$ = MQR(GetMin$(p, \sigma, b), l$)
3:    return $R_b(P_{\sigma^{-1}}(G))$
4: else if $\exists i \in SF, \exists b \in SB, \exists l \in \{1, 2, \ldots, \max l\}$,
      $\exists$Node $x \in S[l]$,
      GetMin$(R_b\left(P_{\sigma_i}\left(p\right)\right) \circ (\mathrm{Node}\,x.\mathrm{cpm})^{-1}, \tau, c)$
      $\in S[\max l]$ then
5:    $G_2$ = MQR(Node $x$.cpm, $l$)
6:    $\rho = \mathrm{GetMin}\left(R_b\left(P_{\sigma_i}\left(P\right)\right) \circ (\mathrm{Node}\,x.\mathrm{cpm})^{-1}, \tau, c\right)$
7:    $G_1$ = MQR$\left(\rho, \max l\right)$
8:    return $R_b\left(P_{\sigma_i^{-1}}\left(R_c\left(P_{\tau^{-1}}\left(G_1\right)\right) G_2\right)\right)$
9: else
10:    return NULL
11: end if
```

Algorithm 3: Quantum_Minimum _Representation (QMR).

in the basis of [11] through bidirection synthesis, and another 4 layers by combining DFS, thus realized a 12 layers synthesis of arbitrary circuits. Yet [14] can only synthesize the first 4 layers of the optimal circuit at a time, while in our previous study, for example, the hash-based 3-bit synthesis algorithm, the average speed of the mini-length and minimum cost are 49.15 and 365.13 times of [11], respectively. In this paper, we used CNT quantum gate library and mini-length criteria, creating all optimal circuits with up to 8 gates. By bidirection cascading the generated circuits, we can quickly synthesize the optimal quantum circuits within the length of 16, without consuming more memory. Our bidirection cascading is quite different with the bidirection synthesis used in [14]; they calculated the head and tail of the circuit, respectively, then moved forward to the middle. In order to avoid repeated computation, we first synthesize the former parts of the circuit, then perform specific topology transformations on them and reuse them in the latter part.

To evaluate the ability of the algorithm while synthesizing complicated circuits, we have run our program on a great number of circuits, and none of them has been found not to be synthesized. Then, we only give two examples. (1) We cascaded the two optimal circuits 4_49 and Hwb4 to get one circuit in Figure 5A. By using the generated all minimal permutation circuits with up to 8 gates, it took only 35 s to generate circuit B. It is easy to prove that the permutation of both A and B are (15,2,3,12,5,9,1,11,0,10,14,6,4,8,7,13). (2) Synthesized Alhagi01 (2,12,8,13,0,9,6,15,10,11,14,4,5,3,1,7) circuit is given in Figure 6.

5. Conclusions

Based on the idea that the synthesis of reversible logic circuit is a permutation problem in essence, we present the novel and efficient quantum circuit synthesis algorithms. Among them, we elaborately construct a shortest encoding method of the permutation and compress the memory space of the n-bit optimal circuits to $2 \times n!$ times less using certain topology transformation of quantum circuits. By

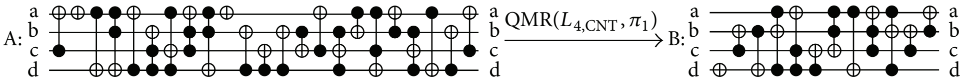


Figure 5: The 4-bit reversible logic circuits synthesis.

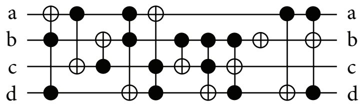


Figure 6: Alhagi01 circuit synthesis.

bidirectional cascading of the generated optimal circuits, using several quantum gates and the mini-length cost metric, our algorithms can efficiently generate most optimal 4-bit reversible logic circuits.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (nos. 61070240, 61170321, 61272175, and 61103235), the Natural Science Foundation of College of Jiangsu Province (no. 10KJB520021), and Specialized Research Fund for the Doctoral Program of Higher Education (no. 20110092110024).

References

[1] R. P. Feynman, "Quantum mechanical computers," *Foundations of Physics*, vol. 16, no. 6, pp. 507–531, 1986.

[2] E. Fredkin and T. Toffoli, "Conservative logic," *International Journal of Theoretical Physics*, vol. 21, no. 3-4, pp. 219–253, 1982.

[3] X. Song, G. Yang, M. Perkowski, and Y. Wang, "Algebraic characterization of reversible logic gates," *Theory of Computing Systems*, vol. 39, no. 2, pp. 311–319, 2006.

[4] K. Iwama, Y. Kambayashi, and S. Yamashita, "Transformation rules for designing CNOT-based quantum circuits," in *Proceedings of the 39th Annual Design Automation Conference (DAC '02)*, pp. 419–424, New Orleans, La, USA, June 2002.

[5] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Proceedings of the 40th Design Automation Conference*, pp. 318–323, San Jose, Calif, USA, June 2003.

[6] A. Mishchenko and M. Perkowski, "Logic synthesis of reversible wave cascades," in *Proceedings of 11th IEEE International Workshop on Logic Synthesis*, pp. 197–202, New Orleans, La, USA, 2002.

[7] P. Gupta, A. Agrawal, and N. K. Jha, "An algorithm for synthesis of reversible logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 11, pp. 2317–2330, 2006.

[8] W. Li, H. Chen, and Z. Li, "Application of semi-template in reversible logic circuit," in *Proceedings of the 11th International Conference on Computer Supported Cooperative Work in Design (CSCWD '07)*, pp. 332–336, Melbourne, Australia, April 2007.

[9] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Reversible logic circuit synthesis," in *IEEE/ACM International Conference on Computer Aided Design (ICCAD '02)*, pp. 353–360, San Jose, Calif, USA, November 2002.

[10] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 6, pp. 710–722, 2003.

[11] G. Yang, X. Song, W. N. N. Hung, M. A. Perkowski, and C.-J. Seo, "Synthesis of reversible circuits with minimal costs," *Calcolo*, vol. 45, no. 3, pp. 193–206, 2008.

[12] G. Yang, X. Song, and M. Perkowski, "Fast synthesis of exact minimal reversible circuits using group theory," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '05)*, vol. 2, pp. 18–21, Shanghai, China, 2005.

[13] W. N. N. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "Optimal synthesis of multiple output Boolean functions using a set of quantum gates by symbolic reachability analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 9, pp. 1652–1663, 2006.

[14] G. Yang, X. Song, W. N. N. Hung, and M. A. Perkowski, "Bi-direction synthesis for reversible circuits," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI: New Frontiers in VLSI Design (ISVLSI'05)*, pp. 14–19, Tampa, Fla, USA, May 2005.

[15] G. Yang, X. Song, W. N. N. Hung, and M. A. Perkowski, "Bi-directionalsynthesis of 4-bit reversible circuits," *Computer Journal*, vol. 51, no. 2, pp. 207–215, 2008.

[16] G. Yang, F. Xie, W. N. N. Hung, X. Song, and M. A. Perkowski, "Realization and synthesis of reversible functions," *Theoretical Computer Science*, vol. 412, no. 17, pp. 1606–1613, 2011.

[17] Z. Li, H. Chen, B. Xu, and W. Liu, "Fast algorithm for 4-qubit reversible logic circuits synthesis," in *Proceedings of the IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence) (CEC '08)*, pp. 2202–2207, Hongkong, China, 2008.