量子回路設計と最適化

山下　茂，松尾　惇士

汎用的な量子コンピューターが実現されるようになった際には，「量子回路設計」という作業が必要になる．本稿では，最初に量子回路設計とはどのようなものかを説明した後，実用的と考えられる量子回路設計のフローを紹介する．その後，量子回路設計において，最適化の対象になっているコストの指標にどのようなものがあるかを説明する．さらに，量子回路設計において最適化すべきコストの中で，近年多くの研究が行われている Nearest Neighbor Cost (NNC) を最適化するいくつかの手法を紹介する．

キーワード：量子回路設計，最適化，Nearest Neighbor Cost (NNC)

1. はじめに

現在の計算機における「論理回路」という用語から，「量子回路」というと完全なハードウェアのように思われるかもしれない．しかし，量子回路は「量子コンピューターが行う基本的な演算の適用順をグラフィカルに表現した図」と解釈できる．つまり，量子回路はハードウェアというよりは，むしろ量子コンピューターに実行させるソフトウェアと考えることができる．一方で，本特集の高橋氏の記事にあるように，現在の論理回路の回路設計と量子回路設計は異なる点も多いものの，量子回路設計には現在の論理回路の設計技術を利用できる点も多い．

本稿では，最初に量子回路設計とはどのようなものかを説明した後，現実的であると考えられている量子回路設計のフローを説明する．その後，量子回路設計において「最適化」の対象とされているコストの指標を紹介する．その中でも，近年量子回路設計の分野で注目されている Nearest Neighbor Cost (NNC) を最適化するいくつかの手法を紹介する．その理由は，NNC の最適化問題は，量子情報の知識がなくても理解できるため，本誌の読者の方が NNC の最適化問題に興味をもっていただけるのでないかと考えたからである．

2. 量子回路設計

多くの量子ビットを同時に 1 ステップで操作できるような物理的な操作を仮定することは現実的にも計算理論的にもありえない（仮定すれば，どんな問題も 1 ステップで解ける！）．そのため，量子計算を実行するためには，たとえば，1 量子ビットまたは 2 量子ビットのみを同時に操作する演算を基本的な量子ゲートと考えて，実行したい量子アルゴリズムを基本的な量子ゲートのみで実現する必要があると考えられている．つまり，「与えられた量子アルゴリズムを物理的に実現可能な基本的な量子演算（量子ゲート）の列に変換（分解）すること」が必要であり，この作業を「量子回路設計」という．

n 量子ビットを操作する量子アルゴリズムは，n 入力の量子回路で実行することができ，n 入力の量子回路は $2^n \times 2^n$ のユニタリ行列で表現できる．そのため，所望の量子アルゴリズムに対応するユニタリ行列を基本的な量子ゲートに対応するユニタリ行列の積（テンソル積）の形に変形できれば，それが実行したい量子回路に相当する．つまり，行列を分解できれば，量子回路設計ができる [1]．実際，小規模なユニタリ行列の場合や，何らかの望ましい数学的な性質をもった行列に限った場合は，行列分解による量子回路設計は可能である．しかし，一般的には，行列の分解には回路の入力サイズに対して指数個の基本行列を必要とするため，実用的な規模の量子回路設計を行列分解だけで行うのは現実的でないと考えられている．

2.1 実用的な量子回路設計フロー

行列分解による手法は，前述したとおり，一般的な大規模な回路に適用できないという意味で実用的なものとは言いがたい．そこで，以下のような設計フローを仮定して，その Step 3 の量子回路設計の問題を扱う研究が多い．

Step 1. 量子アルゴリズムの形式的な記述（量子プログラミング言語など）

Step 2. アルゴリズムの記述から，あらかじめ効率的な実現方法がわかっている部分と問

やました　しげる
立命館大学情報理工学部
ger@cs.ritsumei.ac.jp
まつお　あつし
a.matsuo1223@gmail.com

題ごとに設計すべき部分に分離

Step 3. 問題ごとに設計すべき部分の量子回路のみを設計

上記の設計フローには，人間が設計したアルゴリズム記述の構造をできるだけ利用して回路を設計しようという思想がある．つまり，あえて全体の量子アルゴリズム（ユニタリ行列）を扱わずに，アルゴリズムレベルで分けられた部分ごとに設計するという考え方である．この考え方は，古典の回路設計では（たぶんどんな分野の設計でも？）ある意味当たり前かもしれないが，量子の回路設計でも明示的に提案されている [2]．また，ほぼすべての量子アルゴリズムでは，「問題ごとに設計すべき部分」は「ブール関数の計算をする部分」だけに限定できる．そのため，「ブール関数の計算をする量子回路設計」についての研究が数多く行われてきている．以下では，ブール関数の計算をする量子回路を，特に量子ブール回路：Quantum Boolean Circuit (QBC) と呼ぶことにする．

QBC の論理レベルの設計では，本特集の高橋氏の記事にもある Toffoli ゲート [3] を一般化した MPMCT (Mixed Polarity Multiple-Control Toffoli) ゲート [4] を用いるのが一般的である．

定義 1. MPMCT ゲートは，複数の正または負の極性をもつ制御ビットと一つの標的ビットをもつ．MPMCT ゲートは，その正極性の制御ビットの値がすべて 1，かつその負極性の制御ビットの値がすべて 0 のときに，標的ビットの値を反転（0 と 1 の入れ替え）する．

MPMCT ゲートを図で表すとき，通常，制御ビットは白丸が負極性を表し，黒丸が正極性を表す．また，標的ビットは $\oplus$ で表し，これらを線で繋いだものが一つの MPMCT ゲートである．例として，図 1 に，四つの MPMCT ゲートよりなる QBC の例を示す．たとえば，一番左の MPMCT ゲートは，x_1 と x_3 が負極性，x_2 と x_4 が正極性の制御ビットである．そのため，入力が $(x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 1)$ のときに，標的ビットの値を反転する．

以下では，$\oplus$ で排他的論理和を表すものとする．二つのブール関数 f と g に対して，$f \oplus g$ は，$g = 1$ となる入力組み合わせのときに f の出力値を反転した関数となる．そのことに注意すると，図 1 において，5 ビット目（図で一番下のビット）の初期値を 0 とすると，一番左のゲートが作用した後の 5 ビット目のビットが表している関数を，$0 \oplus \overline{x_1}\ x_2\ \overline{x_3}\ x_4$ と考えることができる．つまり，MPMCT ゲートを適用すると，そのゲートの標的ビットの状態を表す関数は，制御ビットに対応する積項との排他的論理和となる関数に変化すると考えることができる．この図の例では，四つの MPMCT ゲートを作用させたあとの 5 ビット目は，$f(x) = 0 \oplus \overline{x_1}\ x_2\ \overline{x_3}\ x_4 \oplus \overline{x_1}\ x_2\ x_3\ \overline{x_4} \oplus x_1\ \overline{x_2}\ \overline{x_3}\ x_4 \oplus x_1\ x_2\ x_3\ \overline{x_4}$ と表現できる．この $f(x)$ の論理式の形は，ESOP (Exclusive-or Sum Of Product) と呼ばれ，任意のブール関数が表現できることが知られている．つまり，MPMCT ゲートを複数用いれば，任意の関数を実現する量子回路が設計可能である．そのため，与えられたブール関数を実現する QBC を設計するには，論理レベルではまず MPMCT ゲートを用いて設計することが一般的である．

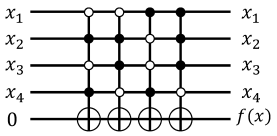

図 1 MPMCT ゲートによる QBC の例

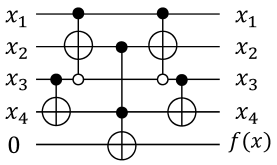

図 2 図 1 と等価な QBC

2.2 量子回路設計における最適化するコスト

以下で，量子回路設計において最適化を目指すべきコストとして，三つのコストの指標を紹介する．

2.2.1 量子コスト (Quantum Cost)

実は，図 2 の回路は，図 1 の回路と論理的に等価である．どちらの量子回路の「コスト」が低いと言えるだろうか？　図 1 の回路のほうがゲート数が少ないのでコストが低いと言えるのだろうか？　量子計算は完全に実現されていないため，量子計算が実現される際の量子回路のコストを厳密に現時点で議論することはできない．しかし，前述したとおり，入力数の多い量子ゲートは直接実現することができないので，2 入力以下のゲートしか 1 ステップでは実行できないと仮定することは現時点でも自然であると考えられている．そのため，量子回路設計の分野では，最終的にはすべて 2 入力以下のゲートに分解して，そのゲートの数を「コスト」と呼ぶのが一般的な考え方であり，文献によっては量子コスト (Quantum Cost) と呼んでいる [4]．

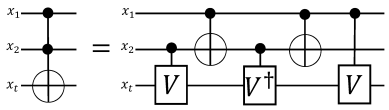

図 3　Toffoli の分解

Toffoli ゲートは 3 入力であるが，図 3 のように，五つの 2 入力以下のゲートに分解できる．そのため，Toffoli ゲートの量子コストは 5 となる．なお，図 3 において，左から一つ目と五つ目のゲートは，制御ビットが 1 のときに標的ビットに，2 回連続して操作を行うと NOT と同じ機能となる V という操作を行うゲートである．三つ目のゲートは，制御ビットが 1 のときに標的ビットに $V^\dagger$ という操作を行うゲートである．ここで $V^\dagger$ は V の逆変換となる操作である．

多入力の MPMCT ゲートは必ず複数の Toffoli ゲートおよび CNOT ゲートと NOT ゲートに分解可能である（これらのゲートについては，本特集の高橋氏の記事の解説をご参照いただきたい）．一つの MPMCT ゲートをどのように分解するかについては多くの研究が行われている．MPMCT ゲートを分解する際に，途中で行う分解方法の違いによって最終的な量子コストが異なるため，分解方法をどのように適用していくかという最適化問題が考えられる．筆者の知る限り，文献 [4] で報告されている量子コストが最小である．文献 [4] で報告されている量子コストは，論理関数までを考慮した種々の最適化手法を適用して得られるもので自明な結果ではない．興味のある方はぜひ文献 [4] をご参照いただきたい．

また，MPMCT ゲートが複数ある場合は，複数の MPMCT ゲートの分解の仕方によってお互いの分解結果の一部がキャンセルしあって量子コストが削減する場合もある（たとえば，同一の二つの CNOT ゲートはその機能がキャンセルしあうため削除可能である）．そのため，与えられた論理を実現する回路の最適な量子コストを求める問題はかなり難しい設計問題であり，まだ研究すべきことが残されていると考えられる．

2.2.2　*T*-Count（*T* ゲートの数）

2.2.1 節では，任意の 2 量子ビット間の演算ができるという仮定でコストの議論を行ったが，任意の 2 量子ビット間の演算が（全体の計算が問題なく行える程度に）エラーなく実行できるという仮定は当面（もしくはかなり遠い未来においても）成り立たないだろうと考えられている．そのため，一つの量子ゲートにある程度の操作エラーが起こったとしても，いわゆるエラー訂正を行いながら所望の計算を行うフォールトトレラント量子計算の枠組みについて多くの研究がなされている．フォールトトレラント量子計算の枠組みでは，多くの場合，エラー訂正を考慮して，1 量子ビットの演算としては，Clifford+T [5] と呼ばれるゲート集合を用いる．

Clifford+T のゲート集合を用いる枠組みでは，特に T ゲート [5] と呼ばれるゲートの実現コストがほかのゲートに比べて格段に大きいと考えられており，設計された量子回路のコストとしては T ゲートの数（多くの文献で，T-Count と呼ばれている）を考えることが多い．

与えられた任意のユニタリ行列を T-Count 最小に分解するという問題は大変難しい問題と考えられるが，ある種のゲートには，T-Count 最小に分解する手法が知られている [6, 7]．この分解手法を一般の量子ゲートの分解に適用するには，一般の量子ゲートをオイラー分解して文献 [7] の手法が適用できる行列の積の形に変換した後に，それぞれの行列を分解する方法が考えられる．

また，どんな多入力の量子ゲートでも任意の 1 量子ビットの演算と CNOT ゲートに必ず分解できることが知られているため [8]，多くの研究が「任意の 1 量子ビットをできるだけ T-Count の少ない，Clifford+T のゲート集合のゲートに分解する」という最適化問題を扱っている．（任意の行列を Clifford+T のゲート集合に対応する行列の積だけで正確に表現することはできないため，ここでいう「行列の分解」とは「分解した結果が近似的に所望の行列になる」ということを意味している．）

複数の T ゲートを並列に実行できるモデルを想定すると，全体の計算時間は，量子回路の中の並列には実行できない T ゲートの数（T-Depth と呼ばれる）に比例すると考えられる．そのため，T-Count ではなく，T-Depth を最適化する研究も行われており，Matroid Partitioning の考え方を用いて T-Depth を最適化する手法 [9] などが提案されている．本誌の読者の方にも，T-Depth の最適化と Matroid Partitioning が関係するという事実を興味深いと感じられる方がおられるのではないだろうか？

2.2.3　Nearest Neighbor Cost (NNC)

近年，物理的な量子計算の実現の可能性が現実味を帯びてきた．それにより，より具体的に量子ゲートの実現時の制約について，量子回路設計の分野でも議論がされるようになってきた．それは，Nearest Neighbor Architecture 制約（以下，NNA 制約）と呼ばれる制

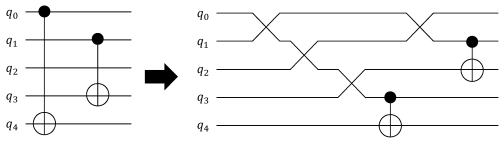

図 4　SWAP ゲートによる NNA への変換例 (1)

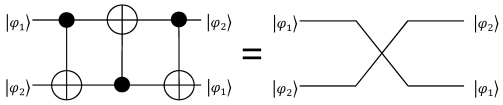

図 5　SWAP ゲート

約である．

NNA 制約とは，「物理的に隣接する量子ビットのみでしか演算することができない」という制約である．たとえば，量子ビットが一次元に並ぶ状況を考えると，図 4 の左側の回路は二つの CNOT ゲートからなるが，どちらのゲートも制御ビットと標的ビットが隣接していないため NNA 制約を満たしていない．しかし，図 4 の左側の回路は，右側の回路のように変形すると NNA 制約を満たすようにできる．ここで，量子ビットの線がクロスしている部分は，SWAP ゲートと呼ばれ，二つの量子ビットの状態を交換する操作である．SWAP ゲートの操作は，図 5 のように CNOT ゲート三つで実現できる．この例では，一つ目のゲートを実行する前に，SWAP ゲートにより q_0 の内容を q_4 に隣接する量子ビット q_3 に移動している．そのため，右側の回路では一つ目の CNOT ゲートの制御ビットと標的ビットを隣接する量子ビットとすることができている．二つ目の CNOT ゲートに対しても同様に SWAP ゲートを挿入して NNA 制約を満たしている．

この例からもわかるように，SWAP ゲートを用いれば NNA 制約を満たすような回路に変形できることがわかる．次節で見るが，SWAP ゲートの挿入方法により，NNA 制約を満たすために必要な SWAP 数は異なる．そのため，NNA 制約を満たすために必要な SWAP 数のことを，Nearest Neighbor Cost (NNC) と呼び，このコストを最適化する研究が多く研究されている．それらの研究について次節で紹介する．

3.　NNC の最適化手法

NNC の最適化手法に関してはさまざまな研究があるが，本節では，代表的なものをその特徴で分類して紹介する．

3.1　対象とするアーキテクチャによる分類

NNC の最適化問題は，当初 1 次元のアーキテクチャを対象としているものが主流であった．たとえば，イオントラップ [10] などによる量子計算を想定して，量子ビットが 1 次元に並び，隣接した量子ビット間でしか演算ができないモデルに特化した研究が行われた．このモデルは，Linear Nearest Neighbor (LNN) Architecture とも呼ばれる．1 次元のアーキテクチャを対象とした NNC の最適化手法として，文献 [11–18] などの研究がある．

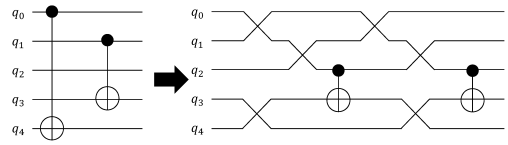

図 6　SWAP ゲートによる NNA への変換例 (2)

量子ドット [19] や中性原子 [20] による量子計算では，量子ビットを 2 次元格子上に配置するのが一般的である．そのため，2 次元格子上に量子ビットを配置し，上下左右の隣接量子ビット間でしか演算ができない量子計算のモデルも提案されている．実際，2 次元のアーキテクチャを対象とした NNC の最適化手法の研究も多く行われており，文献 [21–25] などの研究がある．

また，IBM Q システム [26] のように 2 次元の完全な格子上に量子ビットが配置されていないような量子計算のモデルもある．そのため，最も一般的なモデルとして，任意のグラフの頂点を量子ビットと考えて，隣接している頂点間のみでしか演算が行えないという制約の下で NNC を最適化する手法も研究されている [27–29]．

3.2　最適化する項目による分類

NNC の最適化手法は，最終的に必要となる SWAP ゲートの数をできるだけ減らすことを目標としている．そのために，多くの既存の研究では，(1) SWAP ゲートの挿入方法，(2) 量子ビットの初期配置，(3) 可換なゲートの適用順序，を考慮している．（手法によっては，複数を同時に考慮している．）以下ではこれらの項目に関して，最適化手法の分類を簡単に紹介する．

3.2.1　SWAP ゲートの挿入方法の最適化

前節で説明したように，SWAP ゲートを挿入することによって，図 4 のように NNA 制約を満たすようにすることができる．この例では，一つ目のゲートを実行する前に q_0 と q_4 の量子ビットの内容が隣接するようにしなければならない．そのために，SWAP ゲートにより q_0 の内容を q_4 に隣接するまで移動している．

一つ目のゲートに関して NNA 制約を満たすために，q_0 と q_4 の量子ビットの内容を隣接させる方法は複数

あり，たとえば，図 6 のように SWAP ゲートを挿入する方法も考えられる．この二つの図を比べればわかるように，一つ目のゲートの実行に関して NNA 制約を満たすための SWAP ゲートの挿入方法が異なると，一つ目のゲートを実行した後の量子ビットの内容の隣接関係が異なるため，その後のゲートに対して NNA 制約を満たすようにするための SWAP ゲートの数が変化する．

この例でわかるように，SWAP ゲートの挿入方法により最終的な結果が変わることは容易にわかる．そのため，SWAP ゲートの最適な挿入方法を求める手法が提案されている [28]．文献 [28] では，Pseudo-Boolean Optimization problem (PBO problem) が用いられている．ただ，明らかにこの問題は NP 困難な問題のため，ゲート数が多い場合は，厳密な最適化は現実的ではない．そのため，さまざまなヒューリスティックが考えられている．1 次元向けには，文献 [13–15]，2 次元向けには，文献 [21–23] などの研究がある．

3.2.2 量子ビットの初期配置の最適化

n 量子ビットの配置の仕方は，1 次元であっても 2 次元であっても，$n!$ 通りある．容易に想像できるが，量子ビットの初期配置によって NNC は異なる可能性がある．量子ビットの初期配置の決定方法については，たとえば，量子回路中の各 CNOT ゲートの制御ビットと標的ビットのマンハッタン距離を初期配置段階でできるだけ小さくするように配置する手法が提案されている [17, 24]．そのほかにも，量子回路の部分回路作成時に，各量子ゲートで使用する量子ビットが隣接するように順番に量子ビットを配置していき，もう隣接するように量子ビットを配置できなくなった時点で，初期配置を決定する手法 [27] や，初期配置も含めて最適な SWAP ゲートの挿入方法を計算する手法なども提案されている [12, 28]．

3.2.3 可換なゲートの適用順序の最適化

二つの CNOT ゲートは，一方の標的ビットと他方の制御ビットが同じ量子ビットでない限り可換（その実行順序を交換しても計算結果は変わらない）である．3.2.1 節で見たように SWAP ゲートの挿入方法によって結果が変わることから，実行するゲートの順序を交換すると結果が変わることも容易にわかる．ゲートの順序まで考慮して NNC を最適化する手法としては，文献 [18] の研究がある．この内容については 4.3 節で簡単に紹介する．

今まで紹介した手法はすべて，「与えられた量子回路に SWAP ゲートを挿入することによって，NNA 制約を満たすように変換するときに，必要な SWAP ゲートをできるだけ少なくする」という問題を解いていることになる．そのため，これらの手法で得られる結果は，与えられた初期回路に依存する．言い換えると，今まで紹介した手法では，例え，与えられた問題を厳密に解いたとしても，得られた回路が真の意味で「NNA 制約を満たした最小の量子回路」になっているかの保証はできない．一方，少し異なるアプローチとして，与えられた量子回路（QBC に限定している）の論理機能（ブール関数）を，隣接量子ビットのみに作用する基本ゲートのみを使って設計する問題を，SAT 問題に定式化している研究がある [30]．そのアプローチでは，SAT 問題を解くことにより，真の意味で NNA 制約を満たした「最小の量子回路」の設計が可能となるが，小規模な回路にしか適用できない．

4. NNC の最適化手法の具体例

前節で，NNC 最適化手法では主に三つの項目を考慮することを述べた．本節では，それぞれの項目について，具体的に既存研究でどのような手法が使われているかを簡単に紹介する．

4.1 SWAP ゲートの挿入方法の最適化

NNC の最適化を行ううえで，どのように SWAP ゲートを挿入すればよいかについては，1 次元のアーキテクチャや 2 次元のアーキテクチャに対してヒューリスティックを含めさまざまな手法が提案されている．その中で文献 [28] では，NNC の最適化問題を Pseudo-Boolean Optimization problem (PBO problem) に定式化し，PBO solver を用いて解くことで NNC の最適化を行う手法が提案されている．文献 [28] で提案されている定式化では，1 次元もしくは 2 次元のアーキテクチャだけに限らず，PBO problem の制約の条件を変更することで 3 次元以上のアーキテクチャに対しても NNC の最適化を行うことが可能である．ただし，文献 [28] の提案手法では，挿入可能な SWAP ゲートの組み合わせを全探索して最適解を求めているため小規模な量子回路の場合は問題ないが，大規模な量子回路の場合は現実的な時間で最適解を求めることはできない．また，文献 [28] では，初期配置の最適化も行われているが，量子回路内の各ゲートの順序の最適化は行われていない．

4.2 量子ビットの初期配置の最適化

最適な量子ビットの初期配置を見つける問題は NP 困難であることが示されている [21]．そのため，ヒューリスティックが必要と考えられ，たとえば，配置・配線

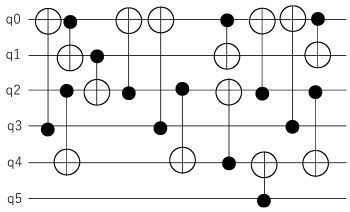

図 7　PAQCS の説明のための CNOT ゲートによる回路

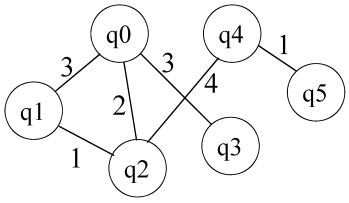

図 8　図 7 に対する interaction graph

の考え方を利用した PAQCS (Physical Design-Aware Fault-Tolerant Quantum Circuit Synthesis) と呼ばれる 2 次元のアーキテクチャ向けのヒューリスティック [24] が提案されている．この手法にはさまざまな改善方法が考えられそうであり，たとえば，文献 [25] で改善手法が提案されている．ここでは，PAQCS で使われている量子ビットの初期配置を見つけるためのヒューリスティックの考え方を簡単な例で紹介する．

図 7 の回路は，q0 から q5 の六つの量子ビットからなっている．この六つの量子ビットの初期の 2 次元配置を，最終的な NNC をできるだけ少なくするように PAQCS では以下のようなヒューリスティックで決定する．まず，interaction graph と呼ばれる図 8 に示すようなグラフを作成する．各ノードは量子ビットに対応し，二つのノード間の辺の重みは，その二つのノードに対応する量子ビットに作用する CNOT ゲートの数とする．この図を用いて，図 9 のように 3×3 の 2 次元配列の各セルに量子ビットを配置していく．

PAQCS で用いられている考え方は，LSI 設計におけるモジュール配置問題などにも共通するところがある．基本的には，(1) 多くの辺をもつノードに対応する量子ビットをできるだけ“自由度”の高いセルに配置する，(2) 接続関係が強いものほど近くに配置する，という二つの方針である．この二つの指標を反映させたコスト関数を定義し，そのコスト関数を最小化するセルに優先度が高い量子ビットから配置する．ここで，「セルの自由度」とは，そのセルに隣接するまだ量子ビットが配置されていないセルの数である．自由度の高いセルに量子ビットを配置すれば，その量子ビットと CNOT ゲートにより相互作用している量子ビットを空いている隣接セルに将来配置できる可能性が高くなるであろうという考え方である．

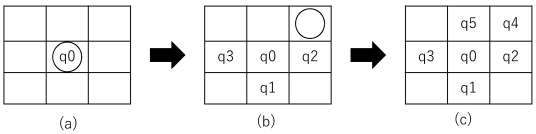

図 9　PAQCS による量子ビットの配置

まずはじめに，接続している辺の数が最も多いノードを一つ選択する．図 8 では，ノード q0 とノード q2 が辺の数が最も多い．そのようなノードが複数ある場合は，辺の重みの合計値が多い q0 を選択する．最初は，まだすべてのセルに量子ビットが配置されていないため，中央のセルの自由度が 4 で一番高い．そのため，図 9(a) のように，q0 が中央に配置される．これは，上記の (1) に基づくヒューリスティックである．

続いて，順に優先度の高い量子ビットを選択して，その時点で空いているセルの中で，上記の (1) と (2) の方針で最もよさそうなセルに配置する．図 9(b) は，q3 まで配置が終わった状態である．残る量子ビットは，q4 と q5 であるが，q4 のほうが接続辺が多いため，q4 を配置する．この時点で自由度が最も高いセルは，上段の真ん中のセルであるが，PAQCS では図 9(b) で ○で示した右上のセルを選択する．その理由は，上記の (1) の指標だけでなく (2) の指標も含めたコスト関数を用いているため，q4 と q2 の間に辺があることも考慮に入れて総合的に右上のセルのほうがよいという判断を行うからである．

この例では，PAQCS では最終的には図 9(c) を初期配置とする．このままでは，NNA 制約を満たしていない CNOT ゲートがあるため，そのようなゲートを実行する前に，SWAP ゲートを挿入して制御ビットと標的ビットを隣接させる．この SWAP ゲート数をできるだけ少なくするような量子ビットの初期配置を，上記のヒューリスティックで探していると考えることができる．

4.3　可換なゲートの適用順序の最適化

文献 [18] では，実行するゲートの順序を考慮するために Gate Dependency Graph が提案されている．Gate Dependency Graph では，量子回路中の順序が交換不可能なゲート間には依存関係があると定義し，ゲート間の依存関係を有向グラフを用いて表現している．たとえば，図 10 の量子回路の Gate Dependency Graph は図 11 になる．図 11 では，ゲート B とゲー

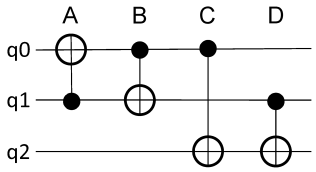

図 10　ゲート間の依存関係がある CNOT ゲートによる回路

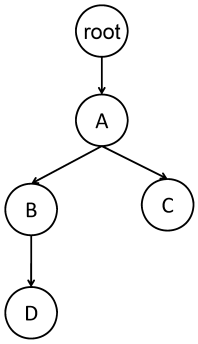

図 11　図 10 の Gate Dependency Graph

ト C はゲート A の実行後に実行される必要があり，ゲート A の実行後はゲート B とゲート C どちらを先に実行しても問題ないことを表している．また，ゲート D はゲート B の実行後に実行される必要があるが，ゲート B の実行後ならばゲート C とゲート D どちらを先に実行しても問題ないことも表している．図 11 の root ノードはスタートを表す特別なノードで，量子回路中の実際のゲートを表しているわけではない．root ノードと直接つながっているノードのゲートは量子回路中のほかのどのゲートにも依存していないことを表している．

具体的にどのように Gate Dependency Graph を用いて，NNA 制約を満たす量子回路に変換を行うかについて以下で説明する．変換を行う際，まず最初に Gate Dependency Graph 上の root ノードと直接つながっているノードのゲートを SWAP ゲート挿入の対象とする．root ノードと直接つながっているノードのゲートが複数ある場合，どのゲートを SWAP ゲート挿入の対象としても問題ない．図 11 では，root ノードと直接つながっているノードのゲートは，ゲート A しかないため，まずゲート A を SWAP ゲート挿入の対象とする（Gate Dependency Graph は変換を行う際のゲートの順序を考慮するために用いるグラフである．root ノードと直接つながっているノードのゲートが複数ある場合の選択方法や実際に変換を行う際の SWAP ゲートの挿入方法には，別途ヒューリスティックなどを用いる）．ゲート A の制御ビットと標的ビットは隣接しており，NNA 制約を満たしているため次はゲート B もしくはゲート C を SWAP ゲート挿入の対象とする．図 10 を見るとゲート B の制御ビットと標的ビットは隣接しており，NNA 制約を満たしている．図 10 の量子回路では，ゲート B の次にゲート C，ゲート D の順で並んでいるが，図 11 の Gate Dependency Graph を参照するとゲート B の次にゲート D，ゲート C の順に SWAP ゲート挿入の対象としても問題ないことがわかる．図 10 を見るとゲート D の制御ビットと標的ビットはすでに隣接しており，追加の SWAP ゲートを挿入する必要はない．そのためゲート順序を入れ替えてゲート B の次にゲート D を SWAP ゲート挿入の対象とする（ただし，この例では実際には SWAP ゲートを挿入しなくても NNA 制約を満たす）．最後にゲート C を SWAP ゲート挿入の対象として q0 と q2 が隣接するように SWAP ゲートを一つ挿入する．

上記の例のように SWAP ゲート挿入の対象となるゲートの順序を初めから固定するのではなく，Gate Dependency Graph 上の依存関係を崩さないようにゲートの順序を入れ替えながら与えられた量子回路を NNA 制約を満たす量子回路に変換することで，NNC が少なくなるようなゲートの順序を考慮した変換を行うことが可能である．

5. おわりに

本稿では，量子回路の最適化問題について簡単に紹介を行った．特に，本誌の多くの読者にも興味をもっていただけると考えた「NNC の最適化問題」について具体的な手法を含めて紹介した．この記事をきっかけに，OR の分野の方が NNC の最適化問題を含めて，量子回路設計に関する研究に興味をもっていただければ，筆者としてはこの上ない幸せである．

参考文献

[1] R. R. Tucci, “A rudimentary quantum compiler (2nd ed.),” arXiv: 9902062, 2001.
[2] K. Iwama, S. Yamashita and Y. Kambayashi, “Transformation rules for designing CNOT-based quantum circuits,” In *Proceedings of Design Automation Conference*, pp. 419–429, 2002.
[3] T. Toffoli, “Reversible computing,” In J. de Bakker and J. van Leeuwen (eds.), *Automata, Languages and Programming*, pp. 632–644, 1980.
[4] Z. Sasanian and D. M. Miller, “Reversible and quantum circuit optimization: A functional approach,” In *Proceedings of Reversible Computation 4th International Workshop, RC 2012*, pp. 112–124, 2013.
[5] X. Zhou, D. W. Leung and I. L. Chuang, “Method-

ology for quantum logic gate construction," *Physical Review A*, **62**, article number: 052316, 2000.
[6] P. Selinger, "Efficient Clifford+T approximation of single-qubit operators," *Quantum Information & Computation*, **15**, pp. 159–180, 2015.
[7] N. J. Ross and P. Selinger, "Optimal ancilla-free Clifford+T approximation of z-rotations," arXiv: 1403.2975, 2014.
[8] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin and H. Weinfurter, "Elementary gates for quantum computation," *Physical Review A*, **52**, pp. 3457–3467, 1995.
[9] M. Amy, D. Maslov and M. Mosca, "Polynomial-time T-depth optimization of Clifford+T circuits via matroid partitioning," *IEEE Transactions on CAD of Integrated Circuits and Systems*, **33**, pp. 1476–1489, 2014.
[10] H. Häffner, W. Hänsel, C. F. Roos, J. Benhelm, D. Chek-al-kar, M. Chwalla, T. Körber, U. D. Rapol, M. Riebe, P. O. Schmidt, C. Becher, O. Gühne, W. Dür and R. Blatt, "Scalable multiparticle entanglement of trapped ions," *Nature*, **438**(7068), pp. 643–646, 2005.
[11] R. Wille, M. Saeedi and R. Drechsler, "Synthesis of reversible functions beyond gate count and quantum cost," arXiv: 1004.4609, 2010.
[12] R. Wille, A. Lye and R. Drechsler, "Optimal swap gate insertion for nearest neighbor quantum circuits," In *Proceedings of 19th Asia and South Pacific Design Automation Conference (ASP-DAC 2014)*, pp. 489–494, 2014.
[13] M. Saeedi, R. Wille and R. Drechsler, "Synthesis of quantum circuits for linear nearest neighbor architectures," *Quantum Information Processing*, **10**, pp. 355–377, 2011.
[14] Y. Hirata, M. Nakanishi, S. Yamashita and Y. Nakashima, "An efficient conversion of quantum circuits to a linear nearest neighbor architecture," *Quantum Information & Computation*, **11**, pp. 142–166, 2011.
[15] A. Shafaei, M. Saeedi and M. Pedram, "Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures," In *Proceedings of the 50th Annual Design Automation Conference*, pp. 1–6, 2013.
[16] M. Rahman and G. W. Dueck, "Synthesis of linear nearest neighbor quantum circuits," arXiv: 1508.05430, 2015.
[17] A. Chakrabarti, S. Sur-Kolay and A. Chaudhury, "Linear nearest neighbor synthesis of reversible circuits by graph partitioning," arXiv: 1112.0564, 2011.
[18] A. Matsuo and S. Yamashita, "Changing the gate order for optimal lnn conversion," In *Proceedings of International Workshop on Reversible Computation*, pp. 89–101, 2011.
[19] J. M. Taylor, J. R. Petta, A. C. Johnson, A. Yacoby, C. M. Marcus and M. D. Lukin, "Relaxation, dephasing, and quantum control of electron spins in double quantum dots," *Physical Review B*, **76**, article number: 035315, 2007.
[20] M. Saffman and T. G. Walker, "Analysis of a quantum logic device based on dipole-dipole interactions of optically trapped rydberg atoms," *Physical Review A*, **72**, article number: 022347, 2005.
[21] A. Shafaei, M. Saeedi and M. Pedram, "Qubit placement to minimize communication overhead in 2D quantum architectures," In *Proceedings of 19th Asia and South Pacific Design Automation Conference (ASP-DAC 2014)*, pp. 495–500, 2014.
[22] D. Ruffinelli and B. Barán, "Linear nearest neighbor optimization in quantum circuits: A multiobjective perspective," *Quantum Information Processing*, **16**(9), pp. 1–26, 2017.
[23] R. Wille, O. Keszocze, M. Walter, P. Rohrs, A. Chattopadhyay and R. Drechsler, "Look-ahead schemes for nearest neighbor optimization of 1D and 2D quantum circuits," In *Proceedings of 21st Asia and South Pacific Design Automation Conference (ASP-DAC 2016)*, pp. 292–297, 2016.
[24] C. C. Lin, S. Sur-Kolay and N. K. Jha, "PAQCS: Physical design-aware fault-tolerant quantum circuit synthesis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **23**, pp. 1221–1234, 2015.
[25] A. Farghadan and N. Mohammadzadeh, "Quantum circuit physical design flow for 2D nearest-neighbor architectures," *International Journal of Circuit Theory and Applications*, **45**, pp. 989–1000, 2017.
[26] IBM Q, https://www.research.ibm.com/ibmq/
[27] A. Zulehner, A. Paler and R. Wille, "An efficient mapping of quantum circuits to the IBM QX architectures," arXiv: 1712.04722, 2017.
[28] A. Lye, R. Wille and R. Drechsler, "Determining the minimal number of swap gates for multi-dimensional nearest neighbor quantum circuits," In *Proceedings of 20th Asia and South Pacific Design Automation Conference (ASP-DAC 2015)*, pp. 178–183, 2015.
[29] D. Bhattacharjee and A. Chattopadhyay, "Depth-optimal quantum circuit placement for arbitrary topologies," arXiv: 1703.08540, 2017.
[30] D. Große, R. Wille, G. W. Dueck and R. Drechsler, "Exact synthesis of elementary quantum gate circuits for reversible functions with don't cares," In *Proceedings of 38th International Symposium on Multiple Valued Logic (ISMVL 2008)*, pp. 214–219, 2008.