Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

量子ビットのエラー情報を用いた最適化コンパイラによる量子プログラムのエラー低減手法

西尾 真 [*1], 潘　宇路 [†2], 佐藤 貴彦 [‡3], 天野 英晴 [§2], バンミーター ロドニー [¶1]

[1] 慶應義塾大学環境情報学部
[2] 慶應義塾大学理工学部情報工学科
[3] 慶應義塾大学量子コンピューティングセンター

2019 年 2 月 11 日

概要

NISQ (Noisy, Intermediate-Scale Quantum) コンピュータにおいて有意義な計算を実行するためにはエラーの低減が必要である．我々のコンパイラ開発では，それぞれの量子ビットによってエラー率が異なることに着目し，加算器のような実世界で利用されるであろうサブルーチンの実行に成功する確率を最大化することを目指している．この論文では，まず量子ゲートをプロセッサ上で離れた位置に配置された変数に対して実行する際に，どの経路を利用するか・どの論理的に等価な回路を使用するかを評価するための定量的基準を定め，それを IBM の 20 量子ビットマシンの一つである Tokyo プロセッサ上でテストした．その結果，我々は一つの数字で個々のゲートの忠実性を表すことは有用ではあるものの完全な基準とはならないことを発見した．

我々はこのサブシステムを用いて量子回路全体をマシンにマップするコンパイラを作成した．このコンパイラはビームサーチを用いたヒューリスティックスを採用することで，プロセッサや量子プログラムの長さに対してスケールするように設計されている．我々は，コンパイル過程全体の評価のために，加算回路をコンパイルし実機で実行したうえで KL ダイバージェンスを計算した．回路サイズがプロセッサ中に収まるサイズの量子回路については，我々のコンパイル手法はエラー情報を利用していないコンパイル手法に比べて推定成功確率が大きく，KL ダイバージェンスは小さくなることを確認した．

1　背景

量子コンピュータの実装は進んでおり [1, 2, 3, 4, 5, 6, 7]，成熟した実装は，さまざまな重要な問題 [8, 9, 10, 11, 12, 13, 14, 15, 16] に対して古典コンピュータを上回る性能を持つと予想されている．近年の実験的研究の進展には目覚ましく，現在 20 量子ビットまでのシステムであれば利用可能であり，49 量子ビットから 128 量子ビットまでのシステムでは現在研究室内で実験中のものや，近い将来に実験される予定のものがある．量子コンピュータは 50–150 量子ビットのどこかで古典コンピュータよりも高い性能を示すであろうが，具体的な値は量子コンピュータの性能（特に，ゲート操作の忠実度）や古典マシンによるシミュレーション手法 [17, 18] の改善の程度次第である．最初に量子計算の優越性が示されるのはそのデモンストレーション用の問題に対してであろうが，その後実際上に有用な問題に対しても示されるに違いない（と信じている）．そのためには，アーキテクチャ [19, 20, 21, 22, 23, 24] やプログラミ

* parton@sfc.wide.ad.jp
† pandaman@am.ics.keio.ac.jp
‡ satoh@sfc.wide.ad.jp
§ hunga@am.ics.keio.ac.jp
¶ rdv@sfc.wide.ad.jp

Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

ングツール [25, 26, 27, 28, 29, 30] の面での工学的な課題が残されている．特に，量子エラー訂正 [31, 32, 33] の完全な実現は依然として手の届かない場所にあるため，今後短期間のノイズ有り・中規模量子コンピュータの時代 [8] においては，エラーを考慮するようなコンパイラを開発することが必要である．

量子プログラムのコンパイル手法は誤り訂正済みの論理量子ビットを用いたフォルトトレラント計算を対象としているのか，それともベアメタルを対象としているのかによって著しく異なる．さらに，ベアメタルの場合でも光子を用いているのか，イオンを用いるのか，もしくは固体量子ビットなのかといった点や，どのような接続制約が存在するのか [23, 24, 34] にも影響される．量子誤り訂正技術の下でのコンパイル手法は近年目覚ましい進展を遂げた分野である（例えば，[35, 36, 37]）が，この論文ではベアメタルマシンに着目し，誤り訂正技術の下でのコンパイルとは異なる目標や制約を対象とする．このマシンレベルでのコンパイルは複数のフェーズから成立する．まず，高級言語で記述された量子プログラムを，対象のマシンで実行可能である 1 もしくは 2 量子ビット操作の列に分解しなければいけない．その後，プログラマが記述した変数をマシン上のどの量子ビットに対応させるかを決定しなければいけない．その際には，配置がうまくいかずマシン上で離れた位置に配置されてしまった変数を移動させるために適切なゲート列を生成することも必要である．また，ハードウェアの低レベル制御も行う必要がある．この論文では，その中でも 2 つめの対応フェーズに着目しており，図1に示すような処理を行う．この論文では我々のアイデアを IBM の 20 量子ビットマシン Tokyo [*1]のみを用いてテストしたが，実際には広範囲の固体量子コンピュータに対して同じ結果は成立すると期待している．

Tokyo はトランズモンを用いた超伝導量子コンピュータであり，量子ビット間の接続は図2に示すとおり限られている [38, 39]．グラフの頂点は 1 つの量子ビットに対応し，それぞれの辺は多量子ビットゲートが物理的に実行可能かどうかを表している．Tokyo マシンなどに挙げられる現在の固体量子ビットプロセッサは，クロストーク等の現象によって制御線および量子ビット同士の結合線を交差させることや，全体で共有のバスを設けることは望ましいことではなくなってしまうため，CNOT ゲート等の多量子ビットゲートの実行について制約が存在する．

現在の量子プロセッサにおいては，さまざまなエラーによって忠実性が悪化してしまい，計算の成功確率に影響が生じる．エラーに対する感受性は量子プロセッサ上のそれぞれの量子ビットによって大きく異なっている．Tokyo プロセッサにおいては，2 量子ビットゲートエラーがその他のエラーよりも大きい．Qiskit ツールによって得られるエラーレートはビット反転エラーと位相反転エラー（量子状態のエラーにおいて基本的な 2 種類のエラー）を区別せず，両者を合わせた値となっている．図2における各辺においては，3% から 12% のエラーレートが報告されている（新しい設計の結合器ではエラーレート

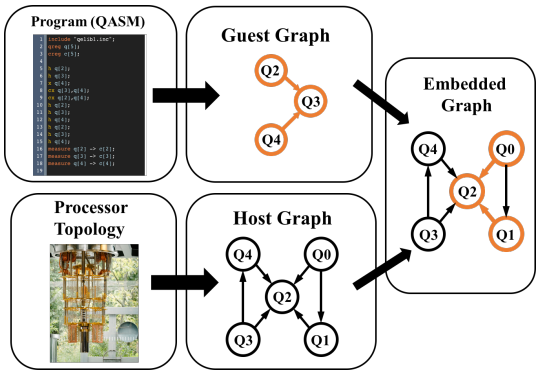


図 1: 回路マッピング: 量子ビットのグラフを考えると，量子プログラム中の量子ゲートは対象の量子ビット同士を結ぶ辺に対応付けられる．また，物理系における制約もグラフとして扱うことができ，量子プログラムのコンパイルとはプログラム中の量子ビット変数をマシン上の量子ビットの位置に対応付けることでプログラム（guest graph）を物理系のトポロジー（host graph）に埋め込むことだと考えられる．

[*1] Tokyo はマシンのコードネームであり，実際はヨークタウンに設置されている．

Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

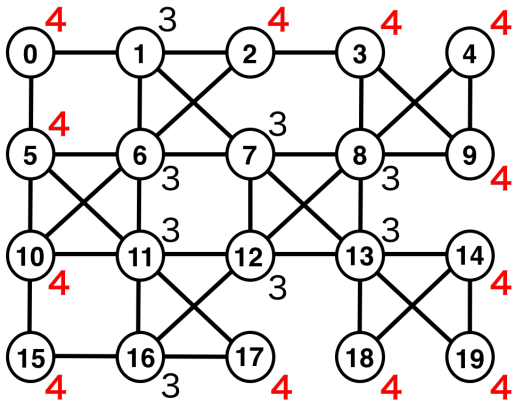


図 2: IBM の 20 量子ビットプロセッサ Tokyo のアーキテクチャ．それぞれの頂点が量子ビットに対応し，それぞれの辺は CNOT ゲートが 2 量子ビット間で実行可能であるかを表している．頂点の外側に配置された数字は各量子ビットの離心率（他の量子ビットとの距離の最大値）を表している．

が 1% 以下になる可能性がある [40]）．中規模のサイズのプログラムをこのようなシステム上で実行するのは大きな課題であり，ゲート数を削減するだけではなく，どの辺が最も使用されるかについて考慮しながらプログラム中の変数を量子ビットに割り当てることが必要不可欠である．いくつかのプロジェクトではゲート数の削減に焦点を当てているものの，我々はプログラムの実行成功確率の向上を第一の目標とした．

システムの理解のため，我々はまずランダマイズドベンチマーキング（RB）と呼ばれるシステムのテストを行った（背景の詳細については2節を見よ）．RB の結果についてはこの論文では述べることはせず，代わりに Qiskit ツール（これも RB を用いている）によって報告されたエラーレートを用いてコンパイルアルゴリズムを開発することにした．なぜならば，アプリケーションの各コンパイルの直前に大規模なテストをマシンに行うのは本質的に実用的ではないからである．

この論文における 1 つ目の課題は，量子プログラム中の各量子ゲートの成功確率の積をプログラムの**推定成功確率** (ESP)（3節を見よ）としたときに，**我々は成功確率を精確に予測できるかを評価する**ことである．我々の用いた推定値は実際の値とは離れていたものの，2 つの回路のうちどちらの方が実機において**より良い**のかを判定する問題についてはおよそ 3 分の 2 の確率で成功した．チップ内で長距離間ゲートを実行する際には，理論的には等価な回路でも実行した際の精度に大きな差が生じるため，我々は ESP をどの回路を選択するかの判定に用いた．

2 つ目の課題はこれを利用することで，いかに回路全体をコンパイルしそれをテストするのかということである（4節を見よ）．回路のマッピングフェーズは，ハードウェア設計における配置配線問題と類似しており，残念ながら NP 完全問題であることが示されている [41]．したがって，我々はビームサーチを用いた確率的なヒューリスティックアルゴリズムを用いた．我々は加算器回路 [42] の入力が 1，2 および 4 量子ビットとなっているバージョンをコンパイルした．それぞれの回路は数十から百を大幅に上回る数の 2 量子ビットゲートからできている．性能の評価のため，コンパイルは異なる乱数シードを用いて繰り返し行われた．小規模な回路において量子コンピュータの実機で実験を行った結果，我々のコンパイラは期待される出力分布からの乖離がランダムな変数配置アルゴリズムに比べて著しく小さくなることを確認した．一番大きな回路については，現在のマシンの実現可能な性能を大きく上回っていることも判明した．

結論としては，単純な ESP であっても量子計算の成功確率を改善することはでき，しかもコンパイルに必要な（古典）計算コストはそれほど大きいものではないということが量子コンピュータの実機を用いた実験から分かった．また，この結果は今後数年間で登場するであろうプロセッサのうち最大のものまでスケールことが予期される．我々はより高度な指標を用いることでさらなる改善が望めるであろうと考えている（5節を見よ）．

2 背景

量子計算は 15 年以上にわたってコンピュータアーキテクチャの分野にて扱われているため，分野の紹介は省略する [19, 20, 21, 22]．代わりに，

Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

ここでは高い実行成功確率を達成するために必要であるエラーの測定とモデル化に関連する問題と，それらのコンパイル手法との関係を扱い，最終的には量子コンピュータで古典計算では到達できない範囲の問題を解くことを目指す．

2.1 エラーモデル

我々のモデルでは量子回路におけるエラーを 3 種類に分類する．図3にそれらのエラーが生じる場所を示す．ここでは，量子力学的なモデル [43] を用いるのではなく，エラー確率のみを使用している．

1. 1 量子ビットゲートエラー (G) はユニタリ的なビット反転 ($|0\rangle \rightarrow |1\rangle$ および $|1\rangle \rightarrow |0\rangle$) または位相反転 ($(\alpha|0\rangle+\beta|1\rangle) \rightarrow (\alpha|0\rangle-\beta|1\rangle)$, ただし $\{\alpha,\beta\} \in \mathbb{C}$ は量子状態の複素振幅）であるか，もしくは緩和現象（$|1\rangle$ が低エネルギー状態である $|0\rangle$ に崩壊すること）などの非ユニタリ的なエラーのことである．
2. 2 量子ビットゲートエラー (CNOT エラー) (B) では対象の量子ビットのうち片方または両方の値や位相が反転することがある．2 量子ビットゲートは量子もつれを生成したり，巨大な数を扱うようなアルゴリズムで用いられたりし，片方の量子ビットのエラーがもう片方へと**伝播する**ことになるため，厄介な問題をはらんでいる．
3. 状態初期化および測定エラー (State Preparation And Measurement, SPAM) (S) は重要なエラーの一つであるが，IBM マシンでのプログラムの実行においては 1 回のみかかるので，成功確率についての累積的な影響としては小さい．

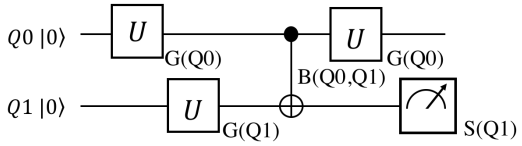


図 3: エラーとそれが発生するゲート

IBM はこれらのエラー確率を各量子ビットまたは各量子ビットの結合子ごとに公表している．IBM Q Network メンバー Qiskit を用いることで Tokyo マシンの値を取得可能である．Tokyo マシンは**ランダマイズドベンチマーキング**を用いて 1 日 1 回キャリブレーションされている．

これらの単純なエラーの他にも，様々なクロストークやシステム内での共鳴によってある量子ビットが他の量子ビットに影響を及ぼしている．これらの完全な特徴づけには広範囲な**トモグラフィー**が必要である．

2.2 トモグラフィー

量子トモグラフィーは状態トモグラフィーとプロセストモグラフィーの二つの基本的な形式が存在する．これらのトモグラフィーは生成した状態がどれぐらい目的の状態に近いか（状態トモグラフィー）や，単一のゲートやゲート列などのプロセスがどれぐらいうまく動作したのか（プロセストモグラフィー）についての情報を与える．これらはエラーの特徴づけに用いることができる．しかし，トモグラフィーには実用的な問題が存在する．まず，システムのとりうる量子状態の数は量子ビットの数に対して指数的に増大してしまう．さらに，我々はビット反転エラーだけではなく位相反転エラーも評価しなければいけない．最終的には，n 量子ビットのシステムについて状態の生成やゲート列を $k3^n$ 回（ただし k は再構築のために必要な精度によって決まる定数であり，数千程度となることもある）実行しなければならないこともある．状態に依存するクロストークやその他の要因についてはトモグラフィーや同等の厳格な（かつ重い）手法を用いなければ調べることはできないものの，それらは 20 量子ビットシステムに対してさえ実用的ではなく，さらに大きなシステムでは不可能である．これらの問題に対処するため，ランダマイズドベンチマーキングが提案された．

2.3 ランダマイズドベンチマーキング

ランダマイズドベンチマーキング（RB）は一定の条件を満たすゲート集合（広く使われるものとしては，**クリフォード群**とよばれるゲートの集合がある）の忠実度を幅広い入力状態を用いることで評価するための手法である [44, 45]．図4にランダマイズドベンチマーキングを実行する回路の概略を示す．

Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

1. クリフォード群から m 個のゲートをランダムに選択し，ランダムな並びの列を作る．
2. 上で選んだ m 個のゲート列全体の逆操作となるようなクリフォードゲート複数量子ビットに対して RB を行う場合には複数のクリフォードゲートの列）を選択し，それを $m+1$ 番目のゲートとして実行する．C_i を i 番目のゲートとすると，

$$C_{m+1} = \left(\prod_{i=1}^{m} C_i \right)^{\dagger} \tag{1}$$

となる．
3. 量子ビットを測定する．もし出力の状態が入力状態とは異なっていた場合，回路全体の実行中にエラーが生じていたことが分かる．
4. 異なる m について 1 から 3 の手順を繰り返す．

上の操作をさまざまな m について行うことで，忠実度の減少度合いが回路の長さに対しての関数としてフィッティングすることができる．これによって，対象のゲート集合について 1 ゲートあたりのエラー率の平均を推定することができる．もちろん，SPAM エラー（状態初期化および測定エラー）はすべての量子回路に含まれてしまうものの，それは減衰率を計算することでキャンセルすることができる．

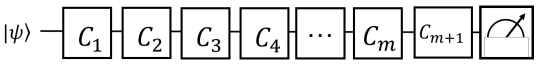


図 4: ランダマイズドベンチマーキング

Qiskit によって報告されている Tokyo マシンのエラーレートは，これと同様の手続きによって毎日報告されている．これらの値は対象のシステムについてすべての状況下でのふるまいを記述するわけではないが，有益な値ではあり，また求めるためのコストもそれほどかからない．この研究の主要な課題の一つとして，ランダマイズドベンチマーキングの値が対象のマシンで回路の実行が成功する確率を最大化するという目標のために**十分有用**であるかを調べる，ということがある．

2.4 アーキテクチャに応じたコンパイル

アーキテクチャに応じて回路を設計しコンパイルすることでプロセッサのトポロジーに適合させる手法は，大規模なシステムやそのアプリケーション [20, 21, 46, 47] についての初期の提案からすでに研究テーマの一つであった．これらの初期の研究は実行時間に対する影響に着目していた．

近年では，忠実度の改善についてより注目が集まるようになってきた．例えば，Zulehner らによる研究では，コスト関数は 2 量子ビットゲートに 10・1 量子ビットゲートに 1 のコストを割り振ることで計算されている [48]．これは量子コンピュータのエラーレートの違いとおおむね対応している．Tannu らや Finigan らは各量子ビットのエラーレートを考慮することで推定実行成功確率を最大化している [49, 50]．Finigan の研究は IBM の 16 量子ビットマシンにおいて回路最適化によって成功確率は上昇することを示した．

3 長距離 CNOT ゲートの選択

上節で登場した G, B, および S エラーはそれらの種類と位置に応じてエラーレート ε を持つ．したがって，ゲート列に対して量子変数をマップする位置を定めるとそのゲート列の推定実行成功確率（Estimated Success Probability, ESP）を下式で定義できる．

$$ESP = \prod_{i} (1 - \varepsilon_i). \tag{2}$$

この値のコスト関数としてのふるまいを調べるため，我々は以下に示す二つの実験を行った．

3.1 遠隔 CNOT における経路選択

ほとんどの量子アルゴリズムは多くの CNOT ゲートを使用している．図2に示されるようなアーキテクチャでは，制御量子ビットと対象量子ビットの両方を近接した位置に配置することはいつも可能であるとは限らない．そのような場合，それらの量子ビットを遠隔 CNOT ゲートによって接続するか，SWAP ゲートによって近接する位置まで移動させる必要がある．我々は，以下に ESP を用いて最適な回路を選ぶ問題の定義およびそれの Tokyo マシンを用いた実験によ

Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

る評価を示す．

問題 1: CNOT 経路選択

プログラマが図5に示す回路を実行したいとする．2 量子ビットゲートの始点（制御量子ビット）と終点（対象量子ビット）が隣接していない場合，どの経路を用いるのが一番忠実に実行できるだろうか？

我々は以下の実験を行った．

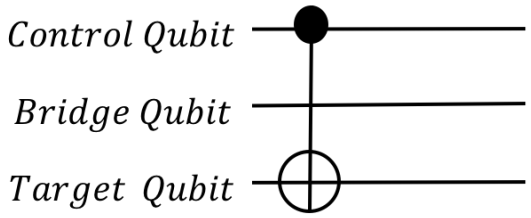
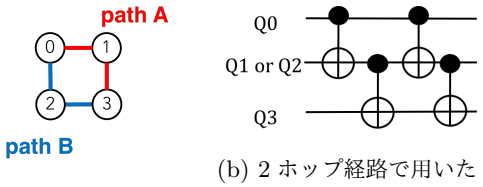


図 5: Circuit A: 量子ビットを一つ飛ばしにかかる CNOT ゲート

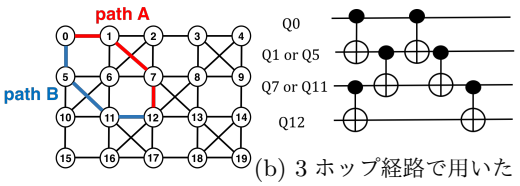
1. 実行可能な経路のうち，我々のコスト関数（ESP）が最大となるものを選択する．図6a, 図7a, 図8aはそれぞれ 2, 3, および 4 ホップの経路選択を示す．それぞれのホップにおいて用いた回路を図6b, 図7b, 図8bに示す．
2. それぞれの経路について実機で 1000 回プログラムを実行する．簡単のため，$|000\rangle$ を入力状態として用いた．実行後，対象量子ビットの値が制御量子ビットのもの（$|0\rangle$）と等しくなかった場合，その経路でエラーが生じたとみなした．これによって，最適な経路（最も回路の実行成功確率が高かった回路）を決定した．
3. 実験によって求めた最適な経路がステップ 1 で求めたものと等しかった場合，経路選択に成功したとみなした．

あるグラフにおいてある頂点の離心率とは，その頂点からほかの頂点への距離のうち最大のものとして定義される．図2に示す通り，IBM が現在公開しているプロセッサにおいて各量子ビットの離心率の最大値は 4 である．したがって，このトポロジーにおいては 4 ホップまでの経路について CNOT 回路の選択ができれば十分なため，実験も 4 ホップまでについて行った．



(a) 2 ホップ経路選択　(b) 2 ホップ経路で用いた回路

図 6: (a) は 2 ホップの CNOT を行う際の経路の例．(b) は (a) の経路で実行する回路．端点の量子ビットは同じであり，中間の量子ビットだけが経路によって異なる．



(a) 3 ホップ経路選択　(b) 3 ホップ経路で用いた回路

図 7: 2 ホップのときと同様に (a) は 3 ホップの CNOT を行う際の経路の例を示し，(b) はそのとき用いた回路を示している．

結果 1

ESP を用いることで，2 つの経路のうち最適な経路を選べる確率は 2 ホップの場合で 70%，3 ホップの場合で 66.6%，4 ホップの場合で 62.5% であった．

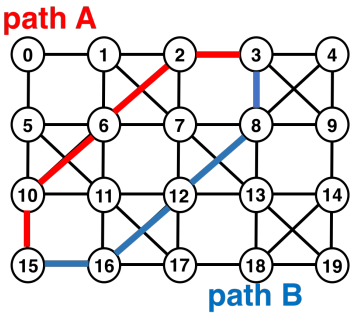
もちろん，このとき 37.5% までの確率で ESP が正しくない経路を選択している．その理由には，以下のものが考えられる．

1. 報告されているエラーの値はビット反転と位相反転の両方が含まれているが，我々の実験ではビット反転のみを扱っている．
2. システム内で長距離間の共鳴およびクロストークが生じるため，それぞれのゲートを別個に実行した場合とはふるまいが異なる．
3. マシンの実際の状態がランダマイズドベンチマーキングによるキャリブレーションによって得たものからドリフトしており，推測に用いたエラーレートが実際の値と異なっている可能性もある．

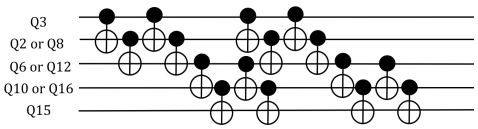
原因 3 が大きな影響を及ぼしている可能性は

Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

低いと考えられる．我々は今後原因 1 と原因 2 のうちどちらが相対的に重要であるかを明らかにしたいと考えている．



(a) 4 ホップ経路選択



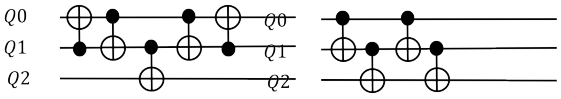
(b) 4 ホップ経路で用いた回路

図 8: (a) は Q3 と Q15 の間で 4 ホップの CNOT を実行する際の経路の例である．(b) はその経路で実行する回路を示す．

3.2 遠隔 CNOT における回路選択

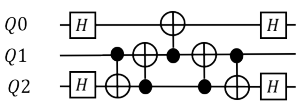
問題 2: 回路選択

プログラマが図5に示す回路を実行したいとする．遠隔の量子ビットに対して CNOT を実行する場合，図9に示すように理論的に等価な回路は複数存在するが，これらを実機で実行した際のエラーのかかり方も同じであるとは限らない．では，これらの回路のうちどの回路が最も高い忠実度を示すだろうか？



(a) 回路 B

(b) 回路 C



(c) 回路 D

図 9: 図5と等価な回路．

我々は以下の実験を Tokyo マシンで行った．

1. 図9に示す回路 B, C, D のうち我々のコスト関数（ESP）が最大となるものを実際に最も忠実度が高くなるだろう回路として選ぶ．
2. これらの回路を Tokyo マシン上で 1000 回実行する．簡単のため，入力状態として $|000\rangle$ を用いた．出力の対象量子ビットの値が制御量子ビットの値（$|0\rangle$）と異なっていた場合，回路の実行中にエラーが生じたと判定した．それによって，どの回路が最適な回路（最も成功確率が高い回路）であるかを決定した．
3. 実機において最適と判定された回路がステップ 1 で選択した回路と同じであった場合，回路の選択に成功したと判定した．

結果 2

ESP を用いることで，3 つの回路から最適なものを選ぶことができた確率は 40% であった．

ESP を用いた結果，回路 B は 35% の確率で，回路 C は 10% の確率で，回路 D は 55% の確率で選択された．実験的に，回路 B が最適であった確率は 50%，回路 C は 15%，回路 D は 35% であった．ランダムに回路を選択していた場合，最適なものを選ぶ確率は 3 分の 1 であるため，ESP を用いることでランダム選択よりは改善されることが分かった．

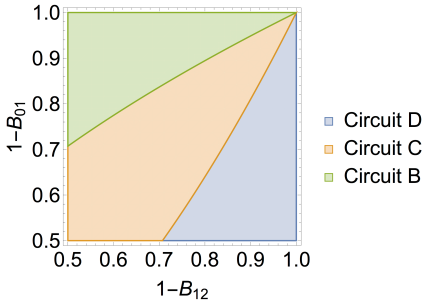


図 10: 2 ホップ回路の推定最適回路．量子ビット間のエラー率が拮抗している場合，回路 C が推定実行成功確率が最大となる．Tokyo マシンの場合は回路 B や回路 D が最大の ESP を持つことが多かった．

Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

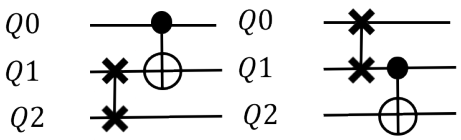
ESP を用いた際にどの回路が最適な 2 量子ビットゲートの実現として推定されるかを図10に示す．B_{01} は Q0 と Q1 の間で 2 量子ビットゲートを実行するときのエラー率，B_{12} は Q1 と Q2 の間で 2 量子ビットゲートを実行するときのエラー率である．

もし，各回路の CNOT ゲートの数を忠実性の推定に用いた際には，実際に最適な回路を選べる確率は 15% にすぎない．

問題 2': 回路選択 (SWAP も含む)

問題 2 において，図11に示すような SWAP ゲートによる量子ビットの移動も考慮した際に最も高い忠実性を示す回路はどれであろうか？

Tokyo マシンでは SWAP ゲートはネイティブサポートされていないため，我々は SWAP ゲートを 3 つの CNOT ゲートに分解した．CNOT ゲートの向きも考慮すれば，SWAP ゲートの分解方法は 2 種類存在する．図11aに示す回路は図12aおよび図12bに示す回路として実装することができる．また，図11bに示す回路は図12cおよび図12dに示す回路として実装することができる．我々は問題 2 と同様の実験を回路 B–H まですべてを考慮に入れて行った．



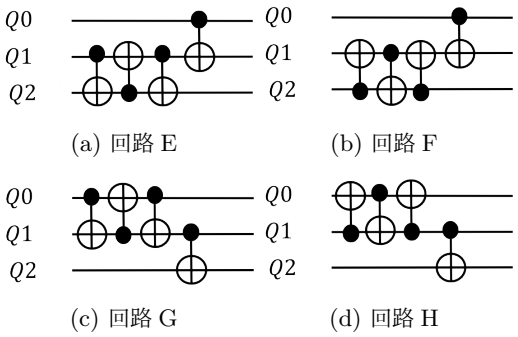
(a) SWAP Q1, Q2　(b) SWAP Q0, Q1

図 11: SWAP ゲートを用いた回路

結果 2'

ESP を用いることで，7 つの回路のうちから最適な回路を選ぶことは 25% の確率で成功した．図13に選択された回路の成功確率とほかの回路のものの比較を示す．

選択された回路の ESP の平均値は 0.8255 であった．一方で，Tokyo マシンで回路を実行した場合に得られた成功確率は 0.8609 であった．ランダムに回路を選択した場合，最適な回路を選択できる確率は 7 分の 1 であるが，ESP を用いることで 25% とそれよりも高い確率で最適な回路を選択することができた．ほとんどの場合で，平均よりもよい回路を選択することができた．



(a) 回路 E　(b) 回路 F

(c) 回路 G　(d) 回路 H

図 12: 図11に等価な回路．SWAP ゲートは 3 つの CNOT ゲートを用いて実装できる．

選択された回路は実際には ESP よりも高い忠実度を示した．これは，初期状態として 0 状態のみを用いたことが影響しているかもしれない．もしくは，我々の実験は完全なトモグラフィーよりも不完全であり，振幅エラーを考慮していないことが原因となっている可能性もある．

これを示すため，我々は入力状態として $|0\rangle$, $|1\rangle$, $|+\rangle$ を用いて $3 \times 3 \times 3 = 27$ 状態を用いた実験を計画している．

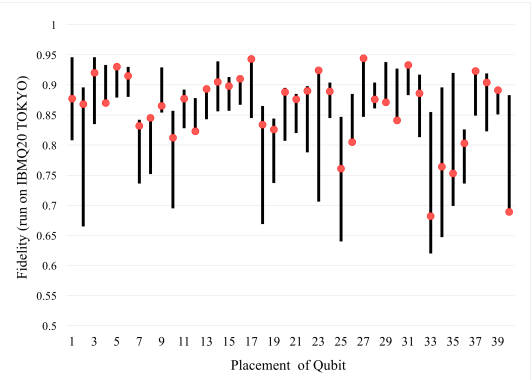


図 13: 選択された回路の忠実度．7 つの回路（回路 B–H）を Tokyo プロセッサ上の 40 個の異なる位置にて実行した．赤い点は ESP を用いて最適であると推定された回路が実機で示した忠実度である．グラフの棒の上端は 7 つの回路の成功確率のうち最大値であり，下端は最小値を示している．

Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

4 プログラム全体のコンパイル

前節では，一つの CNOT ゲートを IBM の量子コンピュータアーキテクチャ上で実現する種々の方法について議論し，それらの実行成功確率についての推定値と観測値の比較を行った．量子プログラムを一般にコンパイルするためには，コンパイラはこれらの実現方法を組み合わせて，多数の CNOT ゲートからなるより複雑な回路を取り扱わなければいけない．実現方法の組み合わせ方には複数の可能性があることから，コンパイラはその中からより信頼性の高い実現方法を選びださなければいけない．それらの組み合わせについての忠実度の尺度を与えるため，式2に示した推定実行成功確率（ESP）の定義を回路全体へと拡大する．つまり，量子回路 C の推定実行成功確率をそれらの構成要素の推定実行成功確率の積として下式の通り定義する．

$$ESP(C) = \prod_{g \in C} ESP(g) = \prod_{g \in C} (1 - \varepsilon_g). \quad (3)$$

このモデルにおいて，我々は

1. 量子回路中のそれぞれのゲートおよび測定の結果は実行に成功するまたは失敗により回路全体の実行が停止するかのいずれかである．
2. それらの失敗確率は操作ごとに独立であり，対象の量子ビットにのみ依存して定まる

この定式化の下で，コンパイラの仕事は $ESP(C)$ の最大化と定義する．そのためのアルゴリズムとして，我々は以下の小節で紹介する組合せ最適化手法を適用した．

式3は最適化プロセスの中で繰り返し計算するうえで十分シンプルかつ計算量が小さいが，ノイズあり・中規模量子コンピュータ（NISQ コンピュータ）を用いてコンパイル後の回路の信頼性を評価するためにはこの指標を用いることはできない．なぜかというと，量子回路の出力は一般的には単一の値ではなく確率分布となるからである．例えば，各量子ビットにアダマールゲートをかけたのちにそれぞれの量子ビットを測定するような回路を考えると，この回路の出力分布は一様分布となる．したがって，この回路を実機で一回実行したとしても，その実行は成功だったのか失敗だったのかを測定結果から判定することはできない．代わりに，回路の実行結果を多数サンプリングすることで，出力結果の経験分布を構成し，それをノイズ無し量子コンピュータでその回路を実行したときの理想分布と比較するべきである．我々は，そのようなコンパイル後の信頼性を示すような確率分布に基づいた回路指標として経験分布と理想分布の間の KL ダイバージェンスを提案する．また，IBM の Tokyo マシンを用いて ESP と KL ダイバージェンスの間の関係を調査した．

4.1 探索空間

この節では，ESP の最大化問題の探索空間を精査する．

最適化アルゴリズムが訪問するノードはゲートの実行状態および量子ビットのマッピングの 2 つによって決定される．ゲートの実行状態とは，現在の状態においてどのゲートがすでに実行済みかそうではないかを表すビット列である．N をゲートの数とすれば，全体で 2^N の実行状態が存在する．また，各ゲートの実行状態について，量子変数の物理量子ビットへのマッピングが $P(Q,V) = Q \times (Q-1) \times \cdots \times (Q-V+1)$ 通り存在する [51]．

探索空間はハイパーキューブの集合として図示することができる．それぞれのハイパーキューブはゲートの実行状態全体に対応し，それぞれのマッピングについて一つのハイパーキューブが対応する．状態間の遷移はゲートの実行と SWAP の挿入という 2 種類が存在する．

ゲートの実行を行った場合，そのゲートの実行状態のフラグを立てた状態へと遷移が生じる．したがって，ゲートの実行を繰り返すことで初期状態の $000\ldots0$ 状態から最終状態である $111\ldots1$ 状態まで遷移していくこととなる．一方で，SWAP ゲートの挿入を行った場合，ゲートの実行状態は変化しないものの，量子ビットのマッピングは別のものとなる．したがって，SWAP ゲートの挿入は異なるハイパーキューブへの遷移とみなすことができる．これらより，$000\ldots0$ 状態から $111\ldots1$ 状態への任意の経路

Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

はそれぞれコンパイル後の量子回路と対応するので，コンパイラの仕事はそのような経路の中で最適なものを探索することであると定義できる．

これまでの節で述べてきた通り，CNOT ゲートは制御量子ビットと対象量子ビットが物理実装のトポロジー上で隣接している場合のみ実行することができる．この隣接制約によって，特定のマッピングにおいては CNOT ゲートの実行が不可能となり，ハイパーキューブのいくつかの辺は遷移不可能となる．さらに，入力として与えられた量子回路とコンパイル後の回路は等価でなくてはいけないため，ゲート間には依存関係が存在する．ゲート依存関係もハイパーキューブの遷移辺のうち一部を無効化する．

単純な力任せ法では $2^N \times P(Q,V)$ 個の状態を探索する必要があり，これは明らかに実用的ではない．また，[41] では最適化のコスト関数を SWAP の挿入数の最小化とした場合，最適解の探索は NP 困難な問題となることを証明している．したがって，最適解を求めることは実用的ではない．

4.2 最適化アルゴリズム

したがって，本論文ではビームサーチを用いたヒューリスティック最適化アルゴリズムを提案する．ビームサーチとは幅優先探索の変種であり，それぞれの探索ステップにおいて幅優先探索では全ての取りうる状態を探索するのに対して，ビームサーチでは望ましいと推測される一定の数の状態のみを探索する．この定数をビーム幅 B と呼ぶ．ビーム幅が大きくなればなるほどビームサーチで探索する状態数が増えるので，より良い解を見つける蓋然性が高まるものの，一方で時間・空間計算量も増大してしまう．特に，$B = 1$ の場合のビームサーチは貪欲法に等しく，$B \to \infty$ の場合は幅優先探索に等しくなる．

また，ビームサーチをどの初期状態から始めるのかを決定するという点で，量子ビットの最初のマッピングをいかにして決めるのかというのが重要な問題となる．一つの方法としては，ランダムな初期マッピングを生成してしまうというものがある．その他にも，いくつかの論文では [41, 51] コンパイル後の回路サイズを小さくするような初期マッピングを生成するヒューリスティクスが提案されている．我々の手法では，これらの方法を組み合せている．すなわち，ビームサーチの初期状態としてヒューリスティクスによって計算した状態に加えてランダムに生成した初期マッピングも採用している．これによって，ヒューリスティクスの提案する良いマッピングを探索空間に加えた上で，ランダム生成したマッピングがヒューリスティクスよりも良いマッピングをもたらす可能性も取り入れている．以下では，ランダムに生成する初期マッピングの数を M とおく．

量子回路のコンパイルアルゴリズムをアルゴリズム1に示す．このアルゴリズムにおいて，S_i は一番外側のループの i 番目のイテレーションにおいて探索する状態の集合を指す．ループの中では，S_i 内の全ての状態について，ゲート依存関係を満たしている実行可能なゲートを探索し，それぞれの実行可能なゲートについて，そのゲートをスケジュールした遷移後の状態を S_{i+1} に加える．したがって，S_i の全ての状態は i 個のゲートをスケジュールした状態となるため，S_N に含まれる状態のうち，ESP が最も高いものがこのアルゴリズムが出力する結果となる．

アルゴリズムの一番内側のループはたかだか N 個のゲートの実行可能性をチェックするので，その内部のコードは各 i について $N|S_i|$ 回実行される．最内ループのうち，ゲート依存関係の充足性チェック（8 行目）は $state$ に補助的な情報を加えることで定数時間で判定することができる．さらに，10 行目の BEST_SWAP 関数も実行結果をキャッシュしておくことで定数時間の計算となる．したがって，このアルゴリズムにおいて最も計算時間を必要とする部分は，13 行目の状態のスコアを計算する部分である．UPDATE_SCORE 手続きは $O(N)$ の計算時間を必要とする（4.3.1節を見よ）ため，コストの計算にかかる計算量は各 i について $O(N^2|S_i|)$ である．$i = 0$ のとき（最初のイテレーションのとき），S_0 は初期状態なのでその状態の数は $|S_0| = M$ である．$i > 0$ のとき（その後のイテレーションのとき），ビームサーチの枝刈りによって $|S_i| \leq B$ であることが保証されるので，アルゴリズム全体の計算量として

Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

Algorithm 1 コンパイルアルゴリズム

```
1: function COMPILE(gates, topology)
2:   N ← |gates|
3:   S_0 ← INITIAL_STATE(gates, topology)
4:   for i = 0 upto N (N は含まない) do
5:     S_{i+1} ← {}
6:     for all state ∈ S_i do
7:       for all g ∈ gates do
8:         if g の依存関係が充足 then
9:           qubits ← state.mapping[g.qubits]
10:          swap ← BEST_SWAP(qubits, topology)
11:          state' ← state に swap を挿入し g を実行した状態
12:          state'.esp ← state.esp × ESP(swap) × ESP(g)
13:          UPDATE_SCORE(state', gates, topology)
14:          S_{i+1} ← S_{i+1} ∪ {state'}
15:        end if
16:      end for
17:    end for
18:    S_{i+1} ← top-B states of S_{i+1}
19:  end for
20:  return state ∈ S_N のうち，最善なもの
21: end function
```

$O(N^2M + N^3B)$ を得る．この式のうち，左の項は $i = 0$ のときの計算量に相当し，右の項は $i > 0$ のときの計算量に相当する．

上の解析では 18 行目で行っている状態の枝刈りの計算量を無視したが，Floyd-Rivest のアルゴリズム [52] を用いることで枝刈りを $O(|S_{i+1}|) = O(N|S_i|)$ の計算量で行うことができるため，状態のスコアを計算する部分よりも計算量が小さくなり，問題は生じない．

4.3 コンパイルアルゴリズムで用いるサブルーチン

以下の小節では，コンパイルアルゴリズムで用いた 3 つのサブルーチンについて紹介する．

Algorithm 2 状態のスコアの更新

```
1: function UPDATE_SCORE(state, gates, topology)
2:   score ← state.esp
3:   for all g ∈ gates do
4:     if g が実行済みでない then
5:       swap ← BEST_SWAP(state.mapping, g, topology)
6:       score ← score × ESP(swap) × ESP(g)
7:     end if
8:   end for
9:   state'.score ← score
10: end function
```

4.3.1 状態のスコアの計算

この論文で提案するアルゴリズムでは，UPDATE_SCORE 関数を用いて各状態のスコアを更新し，その値を用いて状態の枝刈りを行っている．

状態の評価において，最も単純なスコア関数は既に実行済みのゲートの ESP の積で計算される現在の ESP である．しかし，これから実行するゲートについての推測値もそこに加えることで，状態選択をより効率化することができる．このコンパイラにおいては，それぞれの状態について，現在の ESP に今後実行するゲートの仮想 ESP を掛けることでスコア関数を計算している．仮想 ESP とは，現在のマッピングにおいて対象のゲートを実行する際に挿入することが必要な SWAP ゲートの ESP に対象ゲートの ESP を掛けたものである．今後のスケジューリングによって量子ビットのマッピングが変わる可能性があるため，仮想 ESP は実際にそのゲートをスケジュールする際に用いられる ESP とは異なる可能性がある．

スコア関数の計算アルゴリズムをアルゴリズム2に示す．このアルゴリズムはまだ実行済みではないゲート全体を探索するので，時間計算量は $O(N)$ である．

Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

4.3.2 初期マッピング生成ヒューリスティクス: 最大接続辺マッピング

上で述べたとおり，このコンパイラはランダムに生成した初期マッピングとヒューリスティクスで生成した初期マッピングの双方を探索の初期状態として利用している．このコンパイラで採用した初期マッピング生成ヒューリスティクスは，プログラム中で関係の強い（それらの変数に掛かる CNOT ゲートが多い）量子変数同士をマシン上でノイズが少ない量子ビット同士に写すことを目標として設計されている．これは，IBM の量子コンピュータアーキテクチャにおいて CNOT ゲートのエラー率は他のゲートのエラー率よりも 10 倍程度の値となることを用いている [39].

このヒューリスティクスでは，まずプログラム中の量子変数のそれぞれの組についてそれらに加えられる CNOT ゲートの個数を数える．それによって，量子変数を頂点とし，変数間の CNOT ゲートの数を辺とするようなグラフ（ゲストグラフ）を構築する．

次に，ゲストグラフの各辺をプリムのアルゴリズム [53] と同様の順番で訪問する．ただし，プリムのアルゴリズムは最小の辺から最大の辺へと訪問していたが，このヒューリスティクスでは逆順の最大から最小の順で訪問する．これによって，ゲストグラフの最大全域木が構築されることとなる．

最初の辺においては，まだ量子変数のマッピングは行われていないため，まずはそれらの変数をマシン内でもエラーが一番少ない量子ビットへとマップする．

その後訪問する辺では，片方の量子変数は既にマッピング済みであり，もう片方はまだマッピングされていない．しかも，その辺はマッピング済みの頂点の集合とマッピングされていない頂点の集合を結ぶような辺のうち CNOT ゲートの個数が最大のものである．したがって，我々はこのアルゴリズムを最大接続辺マッピングと呼んでいる．片方の量子変数は既にマッピング済みであるため，もう片方の量子変数をマップした量子ビットの隣にマッピングすれば回路の信頼性は向上すると考えられる．そこで，我々はマップ後の量子ビットに隣接する量子ビットのうち，最も小さいエラーで到達できるような量子ビットを選択し，マッピング済みではない量子変数をその量子ビットへとマップした．このアルゴリズムをアルゴリズム3に示す．

Algorithm 3 最大接続辺マッピング

function INITIAL_STATE($gates, topology$)
  $gates$ からゲストグラフ $G_V = (V_V, E_V)$ を構築
  $Mapped = \{\}$
  $(v_1, v_2) \leftarrow$ 最大辺 $\in E_V$
  $(q_1, q_2) \leftarrow$ 最小辺 $\in topology$
  $M \leftarrow \{v_1 \mapsto q_1, v_2 \mapsto q_2\}$
  $Mapped \leftarrow Mapped \cup \{v_1, v_2\}$
  $E_V \leftarrow E_V \setminus \{(v_1, v_2)\}$
  while マップされてない端点が E_V 内に存在 **do**
    $v \in S$ かつ $v' \notin S$ が成立するような辺 $(v, v') \in E_V$ のうち最大のものを選択
    if $M[v]$ に隣接する頂点でマップされてないものが存在 **then**
      q がマップされておらず，$(M[v], q') \in topology$ を満たすような q' を選択
      $M \leftarrow M \cup \{v' \mapsto q'\}$
      $Mapped \leftarrow Mapped \cup \{v'\}$
    end if
    $E_V \leftarrow E_V \setminus \{(v, v')\}$
  end while
  for all $v \in V_V \setminus Mapped$ **do**
    ランダムに $q \in topology$ を選択
    $M \leftarrow M \cup \{v \mapsto q\}$
  end for
  return M
end function

4.3.3 最適 SWAP 経路探索

このコンパイラでは，SWAP ゲートの挿入のみを用いた遠隔 CNOT ゲートの実現のみを考慮している．コンパイラは隣接制約を満たすためにどのように SWAP ゲートの列を挿入すればよいのかを計算できる必要がある．BEST_SWAP

Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

関数はそのような SWAP ゲートの列をトポロジー内のそれぞれの物理量子ビットについて計算する関数である．

計算アルゴリズムをアルゴリズム4に示す．このアルゴリズムの主要なアイデアは，それぞれの量子ビットの対について「合流辺」を計算するということである．それぞれの量子ビットは SWAP ゲートによって合流辺の端点に移動し，最終的に合流辺上で CNOT ゲートが実行される．SWAP による量子ビットの移動の ESP と最後の CNOT ゲートの ESP の積をとることで，全体の ESP を計算することができる．コスト最小（ESP 最大）となる合流辺およびその端点への移動経路は，最短経路アルゴリズムをトポロジーに適用することで計算できる．

アルゴリズム4にそのアルゴリズムを示す．アルゴリズム4ではループ内で最短経路アルゴリズムをループ内で実行しているが，実装においては Warshall-Floyd アルゴリズム [54, 55] を最初に 1 回走らせることで全ての量子ビット対に対して最短経路を求めておき，その結果を利用している．また，BEST_SWAP 関数の実行結果もキャッシュすることが可能である．したがって，コンパイルアルゴリズム（アルゴリズム1）では最初に BEST_SWAP 関数を全ての組み合わせについて計算しておき，その結果をループ内で用いている．

Algorithm 4 最適 SWAP 経路探索

```
function BEST_SWAP(qubits, topology)
  if qubits.len() == 1 then
    return []
  end if
  ESP_max ← 0
  swap_max ← []
  for all (q_0, q_1) ∈ topology do
    swap_0 ← qubits[0] と q_0 の間の ESP(swap_0) を最大化する SWAP 列
    swap_1 ← qubits[1] と q_1 の間の ESP(swap_1) を最大化する SWAP 列
    ESP ← ESP(swap_0) × ESP(swap_1) × ESP(CNOT over q_0 and q_1)
    if ESP > ESP_max then
      ESP_max ← ESP
      swap_max ← swap_0 + swap_1
    end if
  end for
  return swap_max
end function
```

4.4 コンパイル結果の実験的評価

4.4.1 確率分布に基づいた評価

この節では，IBM の Tokyo マシンを用いたコンパイルした量子回路の実験的評価方法について述べる．式3で述べたとおり，我々のコンパイラでは単純な乗法的エラーモデルを採用し，量子回路の ESP を最大化している．しきあし，ESP ははコンパイル後の量子回路が実際に実機でどれほど信頼性を示すのかを評価するためには問題がある．

どのような問題かというと，量子回路を実行して観測された結果は，一般的には成功・失敗の 2 つに分けることはできないという問題がある．各量子ビットにアダマールゲートをかけた後，全ての量子ビットを測定するような量子回路を考える．この回路の測定結果の分布は一様分布であるので，その回路を一回実機で測定してどのような結果を得られたとしても，その回路の実行に成功しているのか失敗しているのかは分からない．であるから，実行結果から回路の成功・失敗を計算するのではなく，実行結果を NISQ マシンで多数サンプルして得られた経験分布と理想的なノイズなし量子コンピュータが生成するであろう理想分布の差を考えるべきである．

この問題に対処するために，我々は確率分布に基づいたコンパイル後の回路の信頼性の指標を提案する．量子回路 C について，P_{ideal} を C の回路を実行した結果の理想分布と定義し，$P_{empirical}$ を実機で C を実行した際に観測された経験分布と定義する（P_{ideal} は定数であるのに対し $P_{empirical}$ は確率変数となる）．これらの分布の間の KL ダイバージェンス $D_{KL}(P_{ideal}||P_{empirical})$ を下式で定義する．

Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

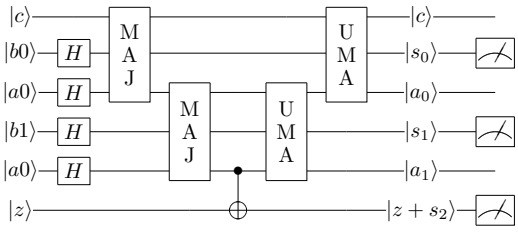


図 14: 2 量子ビット入力の Cuccaro の加算器．$|c\rangle$ はキャリーインであり，$|z\rangle$ はキャリーアウトとなる．MAJ (MAJority) ゲートはトフォリゲート一つと 2 つの CNOT ゲートからなるゲートであり，入力のうち 2 つ以上が $|1\rangle$ 状態であるかチェックすることでキャリービットの計算を行う．UMA (UnMajority and Add) ゲートもトフォリゲート一つと 2 つの CNOT ゲートからなるゲートであり，この桁の加算結果を $|b\rangle$ の量子ビットに書き込み，その他の量子ビットを初期状態へと戻す．ここでは，$|s_i\rangle$ を用いて加算結果の i ビット目を表している．トフォリゲートは 6 つの CNOT ゲートと 9 つの 1 量子ビットゲートからなる．この回路では $|c\rangle$ と $|z\rangle$ の両方の初期状態として $|0\rangle$ を与えているため，この回路の測定結果は $a + b$ の加算結果となる．

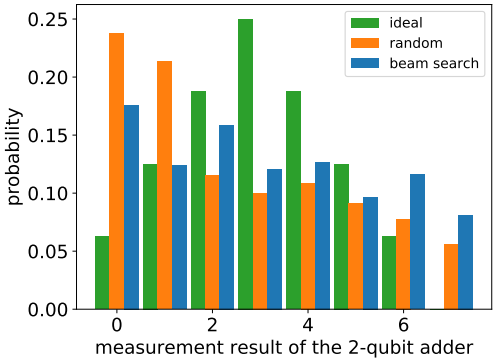


図 15: 2 量子ビット加算器の実行結果の例．緑の棒グラフは理想分布に対応し，オレンジおよび青の棒グラフは IBM の Tokyo マシンでコンパイル後の回路を実行した結果である．オレンジの棒グラフは乱択コンパイラによってコンパイルした回路の実行結果であり，青の棒グラフはビームサーチを用いたコンパイラによってコンパイルした回路の実行結果である．この実験では，乱択コンパイラの回路について $ESP = 0.169$ と $D_{KL} = 0.297$ を得た．ビームサーチコンパイラの回路については $ESP = 0.091$ および $D_{KL} = 0.216$ を得た．

$$D_{KL}(P_{ideal}||P_{empirical}) = \sum_x P_{ideal}(x) \log \frac{P_{ideal}(x)}{P_{empirical}(x)} \tag{4}$$

ただし，x は全ての観測結果について走る．KL ダイバージェンスについては $P_{ideal} = P_{empirical}$ $a.e. \Leftrightarrow D_{KL}(P_{ideal}||P_{empirical}) = 0$ となることが知られている．すなわち，D_{KL} が小さければ小さいほど，コンパイル後の回路の信頼性は高いと言うことができる．

Boixo らはクロスエントロピーという KL ダイバージェンスと定数のオフセットを除いて等しい量を NISQ マシンにおける量子超越性 [56] のベンチマークに用いている [57]．しかし，量子回路のコンパイルについては，そのような指標を用いている論文は我々の知る限り存在しない．

4.4.2 実験結果

コンパイラのテストのため，我々は Cuccaro のリップルキャリー加算器 [42] を用いた．図14に加算器の模式図を示す．テストにおいて，まず入力の量子ビット全てにアダマールゲートをかけて初期化した後，加算器を用いてそれらの和を計算した．最後に，加算結果およびキャリーアウトを計算基底で測定した．我々は，以下の理由から加算器を用いた．

1. 単純で理解しやすい計算であり，理想分布の計算も簡単である．
2. 加算器は CNOT ゲートが複雑に組み合わされてできている．
3. 加算器は多くの量子アルゴリズムにおいて重要な構成要素であるので，それの性能は当然重要である．

この実験では，我々は以下の 2 つのコンパイルアルゴリズムを用いて評価を行った．

1. ビームサーチを用いたコンパイラ．パラメータとして $B = 10000$ と $M = 1000$ を用いた．
2. 乱択コンパイラ．このコンパイラでは，ス

Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

表 1: それぞれの加算器についてのゲート数とコンパイルに要した時間．Q_{input} は加算器の入力の量子ビット数を示している．g_{ori} はコンパイル前の回路に含まれるゲートの数である．$g_{min}, g_{median}, g_{max}$ はそれぞれコンパイル後の回路のゲート数において最小値・中央値・最大値を表す．$T_{min}, T_{median}, T_{max}$ はそれぞれコンパイルにかかった所要時間の最小値・中央値・最大値を表す．我々は Intel Core i7-8550U and 16GB RAM を用いてコンパイルを行った．コンパイラは Rust 言語を用いて 1894 行のコードで記述されている．

		beam search						random selection					
Q_{input}	g_{ori}	g_{min}	g_{median}	g_{max}	T_{min}	T_{median}	T_{max}	g_{min}	g_{median}	g_{max}	T_{min}	T_{median}	T_{max}
1	45	48	66	81	2.96	3.11	3.34	69	96	129	0.022	0.032	0.040
2	82	100	121	145	9.47	10.2	11.6	103	127	211	0.032	0.042	0.053
4	156	255	317	390	31.3	33.6	36.1	239	290	380	0.051	0.063	0.131

コア関数を用いて状態を評価することをせずに，ランダムに状態遷移先を選択している．また，初期状態もランダムなマッピングから開始している．

この実験において，加算器に入力する初期状態は全ての値が等確率で出現するような状態となっているので，この回路の観測結果の分布は，サイコロを 2 回振った目の和の分布と等しくなる．図15にこの回路の理想分布とこれら 2 つのコンパイラを用いた際の観測分布を示している．

我々は 1･2･4 量子ビットの入力量子ビット数についてそれぞれ Tokyo マシン上で実験を行った．表1にゲート数とコンパイルに要した時間を示す．

図16a–16cに実験結果を示す．横軸はコンパイル後の ESP の値を常用対数軸で表している．縦軸は観測結果の経験分布と理想分布の間の KL ダイバージェンスを表している．各点の KL ダイバージェンスを計算するために，我々は実機で実行した結果を 5000 回サンプリングした．また，各試行間でノイズによって実行結果が変化することを考慮するため，同じ回路を用いて複数回試行を行った．図の中の各点はそれぞれの回路を実行して得られた KL ダイバージェンスの中央値を表しており，エラーバーは最大値及び最小値を表している．また，理想分布と一様分布の間の KL ダイバージェンスを補助線として追加してある．

まず，ビームサーチを用いたコンパイラの点がグラフの右側の方に現れていることが分かる．これはビームサーチによって回路全体の ESP が劇的に上昇するような回路の実現方法を選択できていることを示している．横軸は常用対数軸なので，我々の方法によって多くの場合 10 倍程度の ESP が得られていることが分かる．

次に，図16aを見ると，1 量子ビット加算器のほとんどの実験で一様分布よりはよい結果を得られたということが分かる．一方，量子ビットの数が増え回路が複雑になるにつれて，KL ダイバージェンスの値は悪化している．2 量子ビット加算器の場合には，試行のうち半分程度が一様分布よりよい結果を得ることができた．4 量子ビットの場合では，全ての試行が一様分布よりも悪い結果となってしまった．これについては，コンパイルアルゴリズムに関わらず同じ傾向性を示した．

最後に，我々は ESP の変化につれて KL ダイバージェンスがどのように変化するのかを調べた．各グラフに ESP と KL ダイバージェンスの線形回帰を破線を用いてプロットした．1 量子ビット加算器の実験においては，図16aに示すとおり，ESP と KL ダイバージェンスの間には明確な負の相関が存在した．相関係数は −0.475 であった．KL ダイバージェンスは小さければ小さいほどコンパイル後の回路を実行した結果が理想の分布に近いということを表しているので，この負の相関は我々の ESP を最大化するような回路をコンパイルするというアプローチによって実際にコンパイル後の回路の信頼性を 1 量子ビット加算器の場合には向上させることができたということを意味している．

Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

一方で，2 量子ビット加算器の場合（図16b）には，ESP と KL ダイバージェンスの間の相関係数は −0.0279 であり，ほとんど相関が無いことが分かる．

また，4 量子ビット加算器の場合は，ESP と KL ダイバージェンスの間に負の相関は存在はするものの，そもそも KL ダイバージェンスの値は一様分布のそれよりもはるかに悪くなってしまっている．したがって，この場合は KL ダイバージェンスの改善はできたものの，その改善の度合いは回路を信頼性高く実行するには不足していたと考えられる．

5 結論・今後の課題

この論文では，量子ゲートと量子回路について 2 種類の信頼性に関わる指標を提案した．片方は推定実行成功確率（Estimated Success Probability, ESP）であり，量子回路のコンパイル中に用いられる．ESP は量子操作の信頼性を表す合成可能な指標であり，量子ビットの初期化・1 量子ビットゲート・CNOT ゲートの ESP はランダマイズドベンチマーキングによって算出される．また，回路全体の ESP をその構成要素の ESP の積として定義した．このような ESP の合成性より，コンパイラは ESP を簡単に計算できるので，回路全体の ESP をコンパイラによる最適化の対象とした．

加算器や量子フーリエ変換などの回路は量子アルゴリズムにおいて重要な構成要素であり，それらのうち最終状態が重ね合わせ状態となるような回路については，実験的に ESP を実験的に観測することはできない．そのような場合にも対応するため，我々は KL ダイバージェンスを用いて出力分布の信頼性を定量化し，コンパイルアルゴリズム間の比較に用いた．KL ダイバージェンスは完全なトモグラフィーの代わりに使えるような，中間的かつ有用な道具である．

経路選択と回路選択における実験により，ESP を用いることによってランダムな選択より低い（より良い）KL ダイバージェンスを得ることができた．しかし，2 ホップ経路選択のようなシンプルな問題に対しても，選択に成功した確率は最高でも 70% に過ぎなかった．

経路選択においていくつかの困難はあったものの，我々のビームサーチを用いたコンパイラの実験では，回路全体の ESP の改善を見ることができた．また，1 量子ビット加算器を用いた実験では，ESP を改善することで KL ダイバージェンスも減少することが分かった．これは，ESP を最適化するという我々の手法が少なくとも小規模な回路においては実際に NISQ マシン上でエラーを低減することに成功したということを意味している．しかし，回路がより複雑になるにつれて，ESP と KL ダイバージェンスの相関は消失してしまったり，KL ダイバージェンスの値が一様分布とのそれより悪くなってしまったりする現象が見られた．これは現在の NISQ マシンを用いた量子計算の現状における限界を示している一方で，エラー低減を重視したコンパイル手法の重要性も提示している．

ESP はランダムな回路選択よりは良い結果を示したものの，7 つの回路から最適なものを選ぶ問題では 25% の確率でしか最適な回路を選択することはできなかった．これは ESP における仮定と実際の物理系で生じている現象についてずれが生じているからである．

この論文で提案する手法の欠点の一つとして，我々の手法はメモリエラーを考慮に入れていないということがある．Qiskit ツール内で行われているゲートスケジューリングの複雑さの結果，T_1（エネルギー緩和時間）や T_2（位相緩和時間）デコヒーレンスを考慮して ESP を補正することは難しい．我々は，実行時間を取り入れたデコヒーレンスについての包括的な項を導入することを検討している．

我々のコンパイラのより包括的なベンチマーキングのために，量子フーリエ変換などの他の回路もテストすることを検討している．しかし，我々の評価手法は量子超越性の文脈で用いられるような量子回路については使用することはできない．何故ならば，我々の手法では，事前に回路の出力する分布を古典コンピュータによって計算しておく必要があるからだ．したがって，この手法を一般化するためには，うまく KL ダイバージェンスの値を推測できるような手法を開発しなければいけない．また，量子アルゴリ

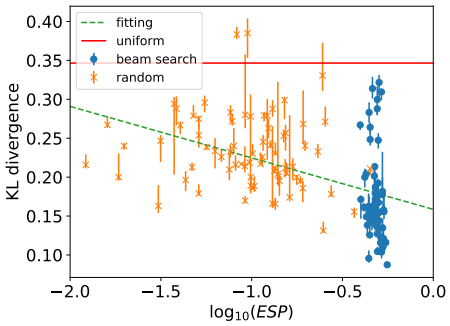
Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

ズムの実行がうまくいくような量子的な干渉パターンが生じる見込みがあるかどうか KL ダイバージェンスから推定するような手法も必要である．そのような手法はビット反転エラーだけでなく符号反転エラーも取り扱える必要があるだろう．

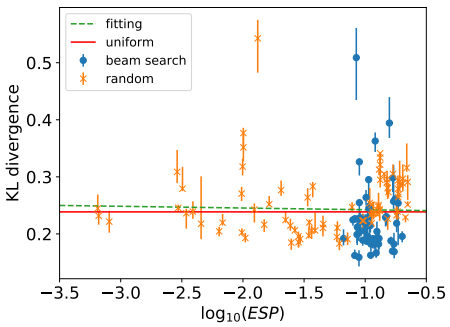
最後に，この論文でエラーを取り扱うコンパイル手法の必要性を示したので，Qiskit コンパイラやその他の言語・システムにおける設計においても同様の手法が取り入れられることを期待している，もしくはこの論文で提案する最適化のプログラムをそれらのプロジェクトに組み入れることを予定している．回路の実行に成功したかを推測する技術が発達するにつれて，このツールは様々な量子ビット結合子や量子ビットのレイアウト間においてアーキテクチャのトレードオフを評価するのにも使うことができるであろうし，それによって次世代の量子コンピュータの設計に影響を及ぼすことが期待される．

謝辞

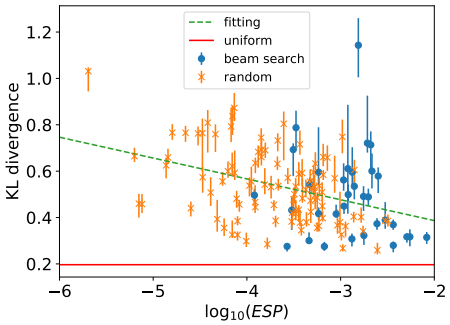
この論文で紹介した実験結果は IBM Q Network メンバーとして IBM Q の量子計算システムを使用することで得られた．この論文における意見は著者のものであり，IBM や IBM Q チームの方針や立場を反映したものではない．



(a) 1-qubit adder



(b) 2-qubit adder



(c) 4-qubit adder

図 16: ESP と KL ダイバージェンスの関係．この実験では，同じ加算器回路に対して一定回数コンパイルを実行した（1・4 量子ビット加算器は 70 回，2 量子ビットは 100 回）．各コンパイル後の回路について Tokyo マシンで一定回数実験を行った（1・2 量子ビットのときは 5 回，4 量子ビットでは 10 回）．各実験において，それぞれ 5000 回回路を実行し，その観測結果から得られた経験分布を用いて KL ダイバージェンスを計算した．プロット中のそれぞれの点はそれぞれの回路で得られた KL ダイバージェンスの中央値を表しており，エラーバーは KL ダイバージェンスの最大値と最小値に対応している．

Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

参考文献

[1] E. Grumbling and M. Horowitz, eds., *Quantum Computing: Progress and Prospects*. National Academies Press, 2018.

[2] T. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, and J. O'Brien, "Quantum computers," *Nature*, vol. 464, pp. 45–53, Mar. 2010.

[3] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, p. 242, 2017.

[4] J. Zhang, G. Pagano, P. W. Hess, A. Kyprianidis, P. Becker, H. Kaplan, A. V. Gorshkov, Z.-X. Gong, and C. Monroe, "Observation of a many-body dynamical phase transition with a 53-qubit quantum simulator," *Nature*, vol. 551, no. 7682, p. 601, 2017.

[5] H. Bernien, S. Schwartz, A. Keesling, H. Levine, A. Omran, H. Pichler, S. Choi, A. S. Zibrov, M. Endres, M. Greiner, *et al.*, "Probing many-body dynamics on a 51-atom quantum simulator," *Nature*, vol. 551, no. 7682, p. 579, 2017.

[6] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. White, J. Mutus, A. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O'Malley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland, and J. M. Martinis, "Superconducting quantum circuits at the surface code threshold for fault tolerance," *Nature*, vol. 508, no. 7497, pp. 500–503, 2014.

[7] E. Martín-López, A. Laing, T. Lawson, R. Alvarez, X. Zhou, and J. O'Brien, "Experimental realization of Shor's quantum factoring algorithm using qubit recycling," *Nature Photonics*, 2012.

[8] J. Preskill, "Quantum computing in the NISQ era and beyond," July 2018. arXiv:1801.00862v3.

[9] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, "Defining and detecting quantum speedup," *Science*, vol. 345, no. 6195, pp. 420–424, 2014.

[10] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and weaknesses of quantum computing," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1510–1523, 1997. arXiv:quant-ph/9701001.

[11] S. J. Aaronson, *Limits on Efficient Computation in the Physical World*. PhD thesis, U.C. Berkeley, 2004.

[12] A. W. Harrow and A. Montanaro, "Quantum computational supremacy," *Nature*, vol. 549, no. 7671, p. 203, 2017.

[13] M. Mosca, "Quantum algorithms," 2008. arXiv:0808.0369.

[14] D. Bacon and W. van Dam, "Recent progress in quantum algorithms," *Communications of the ACM*, vol. 53, pp. 84–93, Feb. 2010.

[15] A. Montanaro, "Quantum algorithms: an overview," *npj Quantum Information*, vol. 2, p. 15023, 2016.

[16] A. P. Lund, M. J. Bremner, and T. C. Ralph, "Quantum sampling problems, BosonSampling and quantum supremacy," *npj Quantum Information*, vol. 3, p. 15, Apr. 2017.

[17] I. L. Markov and Y. Shi, "Simulating quantum computation by contracting tensor networks," *SIAM Journal on Computing*, vol. 38, no. 3, pp. 963–981, 2008.

[18] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, T. Magerlein, E. Solomonik, and R. Wisnieff, "Breaking the 49-Qubit

Barrier in the Simulation of Quantum Circuits," 2017. arXiv:1710.05867v1.

[19] M. Oskin, F. T. Chong, I. L. Chuang, and J. Kubiatowicz, "Building quantum wires: The long and short of it," in *Computer Architecture News, Proc. 30th Annual International Symposium on Computer Architecture*, ACM, June 2003.

[20] D. D. Thaker, T. Metodi, A. Cross, I. Chuang, and F. T. Chong, "CQLA: Matching density to exploitable parallelism in quantum computing," in *Computer Architecture News, Proc. 33rd Annual International Symposium on Computer Architecture* [58].

[21] N. Isailovic, Y. Patel, M. Whitney, and J. Kubiatowicz, "Interconnection networks for scalable quantum computers," in *Computer Architecture News, Proc. 33rd Annual International Symposium on Computer Architecture* [58].

[22] R. Van Meter, W. J. Munro, K. Nemoto, and K. M. Itoh, "Distributed arithmetic on a quantum multicomputer," in *Computer Architecture News, Proc. 33rd Annual International Symposium on Computer Architecture* [58], pp. 354–365.

[23] R. Van Meter and C. Horsman, "A blueprint for building a quantum computer," *Communications of the ACM*, vol. 53, pp. 84–93, Oct. 2013.

[24] R. Van Meter and S. Devitt, "The path to scalable distributed quantum computing," *IEEE Computer*, vol. 49, pp. 31–42, Sept. 2016.

[25] S. Gay, "Quantum programming languages: Survey and bibliography," *Bulletin of the European Association for Theoretical Computer Science*, June 2005.

[26] A. S. Green, P. L. Lumsdaine, N. J. Ross, P. Selinger, and B. Valiron, "Quipper: a scalable quantum programming language," in *ACM SIGPLAN Notices*, vol. 48, pp. 333–342, ACM, 2013.

[27] A. J. Abhari, A. Faruque, M. J. Dousti, L. Svec, O. Catu, A. Chakrabati, C.-F. Chiang, S. Vanderwilt, J. Black, F. Chong, M. Martonosi, M. Suchara, K. Brown, M. Pedram, and T. Brun, "Scaffold: Quantum programming language," Tech. Rep. TR-934-12, Princeton University, July 2012.

[28] A. JavadiAbhari, S. Patil, D. Kudrow, J. Heckey, A. Lvov, F. T. Chong, and M. Martonosi, "ScaffCC: A framework for compilation and analysis of quantum computing programs," in *Proceedings of the 11th ACM Conference on Computing Frontiers*, CF '14, (New York, NY, USA), pp. 1:1–1:10, ACM, 2014.

[29] D. Wecker and K. M. Svore, "LIQUi—¿: A software design architecture and domain-specific language for quantum computing," 2014. arXiv:1402.4467.

[30] J. Heckey, S. Patil, A. JavadiAbhari, A. Holmes, D. Kudrow, K. R. Brown, D. Franklin, F. T. Chong, and M. Martonosi, "Compiler management of communication and parallelism for quantum computation," in *ACM SIGARCH Computer Architecture News*, vol. 43, pp. 445–456, ACM, 2015.

[31] S. J. Devitt, W. J. Munro, and K. Nemoto, "Quantum error correction for beginners," *Reports on Progress in Physics*, vol. 76, no. 7, p. 076001, 2013.

[32] D. Gottesman, "An introduction to quantum error correction and fault-tolerant quantum computation," 2009. arXiv:0904.2557.

[33] B. M. Terhal, "Quantum error correction for quantum memories," *Rev. Mod. Phys.*, vol. 87, pp. 307–346, Apr 2015.

[34] N. C. Jones, R. Van Meter, A. G. Fowler, P. L. McMahon, J. Kim, T. D. Ladd, and Y. Yamamoto, "Layered architecture

for quantum computing," *Phys. Rev. X*, vol. 2, p. 031007, Jul 2012.

[35] P. Selinger, "Efficient Clifford+T approximation of single-qubit operators," *Quantum Information & Computation*, vol. 15, no. 1-2, pp. 159–180, 2015. arXiv:1212.6253.

[36] C. Jones, "Low-overhead constructions for the fault-tolerant Toffoli gate," *Phys. Rev. A*, vol. 87, p. 022328, Feb 2013.

[37] A. G. Fowler and C. Gidney, "Low overhead quantum computation using lattice surgery," 2018. arXiv:1808.06709.

[38] J. Koch, T. M. Yu, J. Gambetta, A. A. Houck, D. I. Schuster, J. Majer, A. Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf, "Charge-insensitive qubit design derived from the Cooper pair box," *Phys. Rev. A*, vol. 76, p. 042319, Oct 2007.

[39] IBM, "IBMQ experience device." https://quantumexperience.ng.bluemix.net/qx/devices, 2018. (Accessed: 2018-06-07).

[40] M. Roth, M. Ganzhorn, N. Moll, S. Filipp, G. Salis, and S. Schmidt, "Analysis of a parametrically driven exchange-type gate and a two-photon excitation gate between superconducting qubits," *Phys. Rev. A*, vol. 96, p. 062323, Dec 2017.

[41] M. Y. Siraichi, V. F. d. Santos, S. Collange, and F. M. Q. Pereira, "Qubit allocation," in *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, CGO 2018, (New York, NY, USA), pp. 113–125, ACM, 2018.

[42] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, "A new quantum ripple-carry addition circuit," 2004. arXiv:quant-ph/0410184.

[43] D. A. Lidar and T. A. Brun, "Introduction to decoherence and noise in open quantum systems," in *Quantum Error Correction*, pp. 3–45, Cambridge Press, 2013.

[44] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland, "Randomized benchmarking of quantum gates," *Phys. Rev. A*, vol. 77, p. 012307, Jan 2008.

[45] E. Magesan, J. M. Gambetta, and J. Emerson, "Characterizing quantum gates via randomized benchmarking," *Phys. Rev. A*, vol. 85, p. 042311, Apr 2012.

[46] A. G. Fowler, S. J. Devitt, and L. C. Hollenberg, "Implementation of Shor's algorithm on a linear nearest neighbor qubit array," *Quantum Information and Computation*, vol. 4, no. 4, p. 237, 2004.

[47] R. Van Meter and K. M. Itoh, "Fast quantum modular exponentiation," *Physical Review A*, vol. 71, p. 052320, May 2005.

[48] A. Zulehner, A. Paler, and R. Wille, "An efficient methodology for mapping quantum circuits to the IBM QX architectures," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems (TCAD)*, 2018.

[49] S. S. Tannu and M. K. Qureshi, "A case for variability-aware policies for NISQ-era quantum computers," 2018. arXiv:1805.10224.

[50] W. Finigan, M. Cubeddu, T. Lively, J. Flick, and P. Narang, "Qubit allocation for noisy intermediate-scale quantum computers," 2018. arXiv:1810.08291.

[51] W. Finigan, M. Cubeddu, T. Lively, J. Flick, and P. Narang, "Qubit allocation for noisy intermediate-scale quantum computers," 2018. arXiv:1810.08291.

[52] R. W. Floyd and R. L. Rivest, "Algorithm 489: The algorithm select – for finding the ith smallest of n elements [m1]," *Commun. ACM*, vol. 18, p. 173, Mar. 1975.

[53] R. C. Prim, "Shortest connection net-

Vol.2019-ARC-235 No.29
Vol.2019-SLDM-187 No.29
Vol.2019-EMB-50 No.29
2019/3/18

works and some generalizations," *The Bell System Technical Journal*, vol. 36, pp. 1389–1401, Nov 1957.

[54] S. Warshall, "A theorem on boolean matrices," *J. ACM*, vol. 9, pp. 11–12, Jan. 1962.

[55] R. W. Floyd, "Algorithm 97: Shortest path," *Commun. ACM*, vol. 5, pp. 345–, June 1962.

[56] J. Preskill, "Quantum computing and the entanglement frontier," 2012. arXiv:1203.5813.

[57] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, "Characterizing quantum supremacy in near-term devices," *Nature Physics*, vol. 14, pp. 595–600, June 2018.

[58] ACM, *Computer Architecture News, Proc. 33rd Annual International Symposium on Computer Architecture*, June 2006.