情報処理学会研究報告
IPSJ SIG Technical Report

決定グラフ型量子シミュレータの性能評価

木村 悠介[1,a)]　李 少文[2,b)]　佐藤 周行[2,c)]　藤田 昌宏[2,d)]

概要：量子コンピュータ向けアルゴリズムの開発のためには、高速で多量子ビットを扱うことが出来る量子シミュレータが重要である。しかし一般的な状態ベクトル型シミュレータでは N 量子ビットに対して 2^N の長さのベクトルを保存する必要があり、手元のコンピュータでシミュレーションするには 30 量子ビット程度が限界である。この問題を解決する手法の 1 つとして決定グラフがある。決定グラフは状態ベクトルをグラフの形で保存するので、状態ベクトルに規則性があれば使用メモリ量を大幅に削減できる可能性がある。本稿では、決定グラフ型シミュレータを独自に実装し様々なアルゴリズムで実験を行い、状態ベクトル型シミュレータとの比較を行った。Grover や Shor などのアルゴリズムでは大幅な高速化と多量子ビット化が見られた一方、パラメータ付き回転ゲートを多く含むようなランダム回路では遅くなることが示された。

Evaluation of Decision Diagram Based Quantum Simulator

Yusuke Kimura[1,a)]　Li Shaowen[2,b)]　Sato Hiroyuki[2,c)]　Masahiro Fujita[2,d)]

Abstract: In order to efficiently develop algorithms for quantum computers, a quantum simulator capable of handling a large number of qubits at high speed is essential. However, a state vector based simulator needs to store a vector of length 2^N for N qubits. In today's typical PC environment at hand, it is evaluated that at most 30 qubits can be handled. A decision diagram-based simulator is expected to solve the problem of memory size. Decision diagram stores state vectors in the form of graphs, which has a possibility to significantly reduce memory usage if there are regularities in the state vector. In this paper, we have implemented our original decision diagram based quantum simulator, and conducted experiments with various algorithms to compare with the state vector based simulator. Algorithms such as Grover and Shor show significant speedup and superiority of performance for larger numbers of qubits, while random circuits with many parameterized rotation gates show slower performance.

1. はじめに

量子コンピュータでは近年目覚ましい開発の進展が見られる。Google は 53 量子ビットを搭載した超伝導型量子チップを開発しており、その量子コンピュータを用いて量子超越性を達成したと主張した [1]。IBM は 433 量子ビットを搭載した量子コンピュータを開発し [2]、様々な量子アルゴリズムの研究開発に役立てている。しかし、このような大規模な量子コンピュータにアクセスできる研究者は限られており、利用のためのコストも高額である。また現行の量子コンピュータはノイズの影響を受けやすく、アルゴリズム開発に用いにくいという問題点もある。したがって、量子コンピュータを模倣することの出来る量子シミュレータを研究開発することは、量子エラー訂正や量子アルゴリズムの開発にあたって非常に重要である。

最も一般的な量子シミュレータは状態ベクトル型である。N 量子ビットの量子状態は 2^N の長さの複素ベクトルで表現することが出来るが、そのまま 2^N 個の複素数分のメモリを確保してシミュレーションを行うのが状態ベクトル型である。IBM の Qiskit Aer[3] や Qulacs[4] などが有名であ

1　富士通株式会社
Fujitsu Limited
2　東京大学
The University of Tokyo
a)　yusuke-kimura@fujitsu.com
b)　li-shaowen879@satolab.itc.u-tokyo.ac.jp
c)　schuko@satolab.itc.u-tokyo.ac.jp
d)　fujita@ee.t.u-tokyo.ac.jp

る。行列ベクトル演算を高速に行うために GPU を活用する事もできる [5]。いずれの場合でも、Double 型を用いて複素数を表現する場合、30 量子ビットの状態ベクトルを保持するには 16GB が必要である。従って手元のコンピュータを用いてシミュレーションするインタラクティブな開発環境では 30 量子ビット程度が限界である。MPI 通信を用いて複数ノードにまたがるメモリ空間をまとめて活用することも可能であり、[6] では 36 量子ビットのシミュレーションが行われている。しかし必要なメモリ量は量子ビット数に対して指数的に増加するため、スーパーコンピュータを用いたとして高々 40 量子ビット程度が限度である。

メモリが不足する問題を解決するために、いくつかの手法が提案されている。テンソルネットワーク型は、状態ベクトルを MPS(Matrix Product State, 行列積状態)[7] として保持し、量子回路をテンソルネットワークに見立ててシミュレーションを行う手法 [8] である。必要なメモリ量を削減できる可能性があるが、エンタングルが多い量子状態や深い回路では逆に動作が遅くなるという欠点がある。特に量子機械学習で活用されている [9], [10]。別の手法として決定グラフ型がある。決定グラフは論理関数を表現するためのデータ構造として古くから用いられてきたが [11], [12]、量子状態（ベクトル）や量子回路（行列）を保存するために利用することも可能である [13]。詳細は後述するが、グラフ構造を用いることで、部分ベクトルに共通部分があったり、0 以外の値が少ない場合（スパースな場合）に使用メモリ量を削減することが出来る。しかし表現すべきベクトルや行列の値がランダムの場合、良いパフォーマンスが得られない可能性が高い。実装として DDSIM[14] や SliQSim[15] が有名である。また筆者らはマルチスレッド化に対応した決定グラフ型量子シミュレータ [16] を開発している。

本研究では決定グラフ型に着目し、その性能を調査することを目的とする。既存研究 [13], [14], [15], [16] でも実験結果は報告されているが、決定グラフ型が高速に動作するアルゴリズムに偏っている場合が多く、広範なアルゴリズムについて調査してその特性を明らかにしたものがなかった。本研究の主な貢献は、QCBM (ランダム回路)[4], QASM Bench[17], VQE[18], Shor[19], Grover[20] などの様々な種類のアルゴリズムを用いて決定グラフ型と状態ベクトル型の実行時間を明らかにし、特性の違いを明らかにした点である。このような違いを理解することで、種々のシミュレータを組合せて利用し、アルゴリズム開発をより効率的に行うことが出来るようになる。

本稿の構成は以下の通りである。2 節で決定グラフ型シミュレータの仕組みを説明する。3 節では、実験環境について説明した上で、ベンチマークごとの実験結果を紹介する。各ベンチマーク回路の特徴についても概説するが、詳細は参考文献を当たられたい。4 節で本稿をまとめ、今後の展望を述べる。また詳細な実験結果は付録にも添付した。

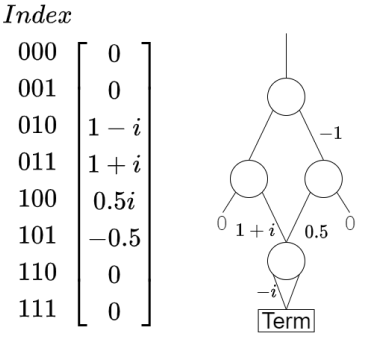

図 1 決定グラフによる状態ベクトル表現

2. 決定グラフ型量子シミュレータ

本節では量子状態や量子ゲートを表現するための決定グラフについて説明する。既存研究で利用されている決定グラフにはいくつかの種類があるが、本研究では DDSIM[14] などが採用している QMDD[13] を用いることにした。従ってこの手法について説明する。なお、本稿ではどのようにして行列・ベクトルを決定グラフで表現するかに絞って説明することとし、決定グラフが省メモリになりやすい理由を理解することを目的とする。紙面の都合上行列ベクトル積や測定は取り扱わないため、参考文献を参照されたい [11], [12], [13], [14], [15], [16]。

2.1 量子状態（ベクトル）の表現方法

N 量子ビットの状態ベクトルは 2^N の長さの複素ベクトルで表現できる。例として図 1 の左側にあるようなベクトルを考えよう。このベクトルは、右側に示された決定グラフでも表現することが出来る。決定グラフからベクトルの各要素の値を知るためには、インデックスの値に応じてエッジを辿り、エッジ重みの積を計算すれば良い。0 は左に、1 は右に進むこととし、値のないエッジ重みは 1 とする。たとえばインデックスが 101 の値を知りたい場合、右・左・右の順に上からエッジをたどる。その際エッジの重みは順に $(1, -1, 0.5, 1)$ であるから、積は -0.5 となる。インデックスが 110 の場合、途中で 0 のエッジ重みが登場するため、積は 0 となる。

元のベクトルには 8 つの複素数が含まれているが、決定グラフには 4 つのノードしか含まれていない点に着目されたい。このようにして、共有するノードがある決定グラフではメモリを削減することができ、アルゴリズムによっては大幅な省メモリ化が期待できる。また、値として 0 が多い場合もノード数を削減できる。

ランダムなベクトルを表現する場合、共有するノードが一切作成できず 2 分木となる場合がある。この場合、量子ビット数に対してノード数が指数的に増えることになる

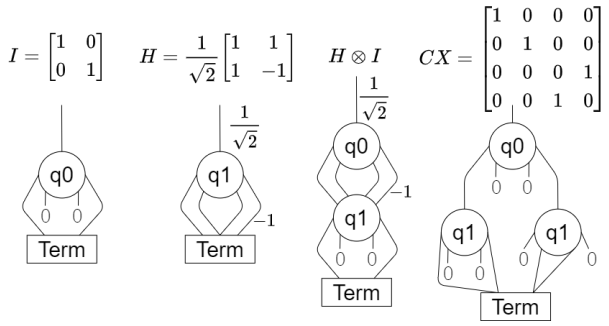

図 2 決定グラフによる量子ゲート表現

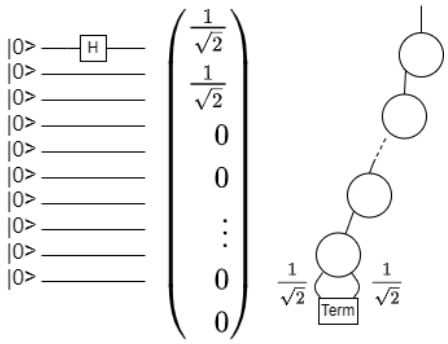

図 3 決定グラフの例

ため、決定グラフを利用するメリットは失われる。複雑なデータ構造を用いている分、状態ベクトル型よりもより多くのメモリを必要とし、処理も遅くなることが予想される。

2.2 量子ゲート（行列）の表現方法

N 量子ビットに作用する量子ゲートは 2^N の大きさの正方行列で表現できる。ベクトルを表現する際には 2 分割するために二分グラフを用いたが、正方行列の場合には 4 分割するため 1 つのノードが 4 つの子ノードへのエッジを持つ。いくつかの量子ゲートの例を図 2 に示す。

2 つの 1 量子ゲートから 2 量子ゲート分のユニタリ行列を得るには、状態ベクトル型の場合にはクロネッカー積を計算する。決定グラフの場合には、2 つの決定グラフを単純に結合することで実現できる。CX ゲートのような 2 つの量子ビットをエンタングルさせるようなゲートの場合、その限りではない。ベクトルと同様に、行列の場合でもノードが共有される場合がある。従って、量子ゲートを決定グラフで表現する場合も、メモリ削減効果が期待できる。

状態ベクトル型のメモリ削減効果を分かりやすく説明するため、図 3 の量子回路を考える。10 量子ビットの回路で、1 ビットにだけアダマールゲートを作用させるものである。状態ベクトル型では $2^{10} = 1024$ 個の複素数を保存する必要があるが、決定グラフの場合には 10 個のノードを保持するだけで良い。以上のように、決定グラフを用いることで必要メモリ量が大幅に削減されることがある。

表 1 実験に用いたベンチマーク一覧

Name	Explanation	Code source
QCBM	Random circuits	[4][*1]
QASM Bench	Various circuits	[17]
VQE[18]	Solving maxcut problem	[21][*2]
Shor[19]	Prime factorization	[22]
Grover[20]	Search algorithm	[14][*3]

3. 実験

3.1 実装・実験環境

本研究では QMDD[13] を採用した決定グラフ型量子シミュレータを開発した。実装に当たっては DDSIM[14] を参考にしたが、今後のマルチスレッド化を考慮してノードを保存するテーブルの実装等は独自に行なった。シミュレーション実行部分は 3000 行程度の C++言語で実装されている。また実験しやすさのために Qiskit Backend として動作するようになっており、その部分は 500 行程度の Python 言語で実装されている。筆者らの既存研究 [16] ではマルチスレッド化した実験結果を紹介している。しかしマルチスレッド版は Qiskit backend への対応等が不十分であるため、本研究ではシングルスレッドでの実験結果を紹介する。

実験に用いた計算環境は以下の通りである。実験はすべてシングルスレッド実行である点には注意されたい。

- OS: Ubuntu 22.04 (Kernel 6.2.0)
- Software: GCC 11.4.0, Python 3.8.16
- CPU: AMD Ryzen 0 7950X (シングルスレッド実行)
- RAM: 128GB

実験に用いたベンチマークは表 1 にある 5 種類である。実験の際には実行時間制限を設定しており、約 10 分である。

実験に用いたシミュレータは表 2 の通りである。State Vector 型のシミュレータにはマルチスレッド・マルチノード・GPU などで実行する機能があるが、すべて無効化してシングルスレッド・シングルノードで実行した。なお DDSIM と Qulacs は 3.2 節の QCBM 実験にだけ用い、以降の実験は我々の実装と Qiskit Aer に限定した。これは Decision Diagram 型と State Vector 型では実行時間等に大きな違いが期待できないことや実験時間を考慮してのことである。

以降の実験結果では「ノード数 (nNodes)」を記載する場合がある。これは、実験の最後に得られる量子状態を表す決定グラフに含まれるノードの数である。

[*1] https://github.com/qulacs/benchmark-qulacs
[*2] https://github.com/cda-tum/mqt-bench/blob/v1.0.4/src/mqt/bench/benchmarks/vqe.py
[*3] https://github.com/cda-tum/mqt-ddsim/blob/v1.19.0/src/GroverSimulator.cpp

表 2　実験に用いた量子シミュレーター一覧

Name	Version	Type	Reference
Ours		Decision Diagram	
DDSIM	1.19.0	Decision Diagram	[14]
Qiskit Aer	0.12.2	State Vector	[3]
Qulacs	0.6.1	State Vector	[4]

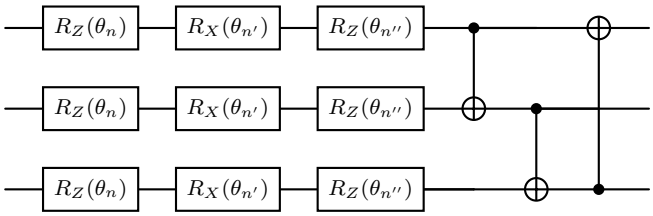

図 4　QCBM (繰り返し部分、3 量子ビットの場合)

3.2　QCBM

QCBM は Qulacs[4] の評価で用いられているランダム回路である。図 4 で示されるランダムなパラメータを持った RX, RZ ゲートと CX ゲートで構成されるサブ回路が 8 回繰り返される構造となっている。最終的に得られる状態ベクトルの値の中身はランダムになっていた。また、決定グラフ型の場合には一切ノードを共有しない木構造となっており、N 量子ビットに対して 2^N 個のノードを持った形になった。

4 つのシミュレータの実行時間を比較するグラフを図 5 に示す。なお詳細な実験結果は付録の表 A·1 も参照されたい。すべてのシミュレータが、おおよそ直線を描いている。これは、Qubit 数が 1 増えるごとに状態ベクトルの大きさは 2 倍になるため、計算すべき量が指数的に増えるからである。また、決定グラフ型は状態ベクトル型と比較して同じ量子ビット数に対して 100 倍以上程度遅いこともわかった。ランダムな状態ベクトルの場合、共有されたノードを作ることができないため、計算時間が短縮できない。複雑なデータ構造を採用している決定グラフが不利になっており、量子ビット数が増えるに連れて決定グラフ型との実行時間の差が開いている。

3.3　QASMBench

QASM Bench[17] には、様々なアルゴリズムから生成された量子回路が含まれており OpenQASM2.0 形式 [23] で保存されている。回路は量子ビット数に応じて small (43 個), medium (24 個), large (57 個) の 3 つに分けられている。本実験では small, medium のすべてと large5 個を用いて Qiskit Aer と我々の実装の比較を行った。large には最大 433 量子ビットの量子回路が含まれており、それらを状態ベクトル型でシミュレーションすることはメモリ使用量の観点から不可能である。従って今回は、large からは 33 量子ビット以下の 5 個だけを利用することにした。実行

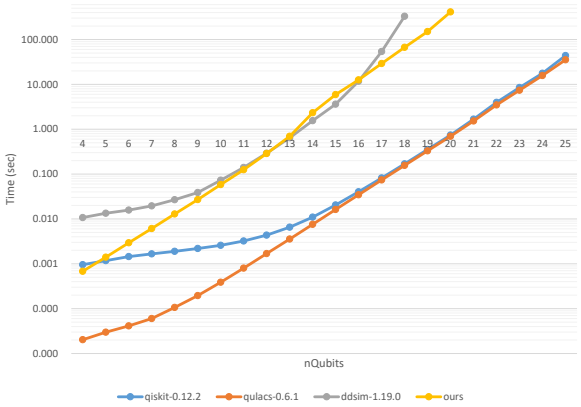

図 5　QCBM 実験結果 (sec, log10 スケール)

時間制限として 1000 秒を設定した。

実験結果として、Qiskit Aer の実行時間と我々の実装の実行時間を散布図にしたものを図 6 に示す。なお詳細な実験結果は付録の表 A·2 も参照されたい。実行時間 1000 秒を超えたものは、上辺・右辺に張り付く形で描画した。点線上にプロットされた回路は、Qiskit Aer と我々の実装で実行時間が変わらないものである。点線より左側にプロットされた回路は Qiskit Aer の方が高速な回路であり、右側にプロットされた回路は我々の実装のほうが高速なものである。

Qiskit Aer が高速な方の回路には、DNN, VQE が多く含まれた。一方で我々の実装が高速な方には、QFT, IQFT, ADD, Multiplier などが含まれた。これらの回路を目視で確認すると、DNN や VQE にはたくさんのパラメータ付き回転ゲート (RX,RY,RZ 等) が含まれており、一方にはほとんど含まれていなかったことがわかった。QFT にはパラメータ付き回転ゲートも含まれるが、回転角のバリエーションは少ないため VQE 等と比較するとノード数は多くなりにくい。以上より、ランダムなパラメータを与えられた回転ゲートがたくさんある場合、決定グラフが複雑になって実行時間が長くなることが示唆された。

3.4　VQE

VQE (Variational Quantum Eigensolver)[18] は古典コンピュータと量子コンピュータの実行を繰り返しながら回路中の Ansatz の適切なパラメータを求めるアルゴリズムである。具体的には、量子回路を作用させて得られた量子状態を古典コンピュータ上の最適化器によって評価し、より良い量子回路内のパラメータを得ることが繰り返される。無制限に量子状態を探索するのは不可能であるから、予め Ansatz と呼ばれる回路を与えておき、その中にあるパラメータを調整することで所望の量子状態が得られるようになっている。

Ansatz には様々なものが提案されているが [24], [25]、一般的に多数の RX, RY, RZ ゲートが含まれており、それら

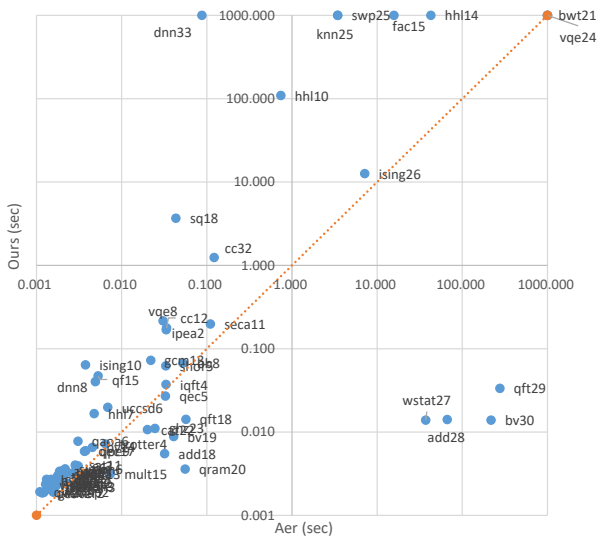

図 6 QASM Bench 実験結果の散布図 (sec, log10 スケール)

のパラメータが調整すべきものである。Ansatz を含む量子回路は複雑な量子状態を作りやすいと予想され、決定グラフ型が遅くなるアルゴリズムのはずである。

この実験では、決定グラフ型シミュレータ DDSIM[14] の研究チームが公開しているベンチマークプログラム [21] に含まれる最大カット問題を用いた。このプログラムは、ランダムなグラフから Maxcut 問題を生成して VQE 問題に変換することが出来る。最適化器には SLSQP を、Ansatz には RealAmplitude を用いた。なお、Qiskit Aer は AerPauliExpectation 命令に対応しており、高速に VQE を行えるようになっている。しかし我々の実装はこれには対応しないため、AerPauliExpectation は用いずに状態ベクトルを最適化器の入力とする設定を行った。

実験結果を表 3 に記す。Qiskit Aer がメモリ使用量に関する警告を発生させるため、15 量子ビット以上の実験は行わなかった。nNodes は、最終的な量子状態を表現するために必要な決定グラフのノード数である。nIterations は量子回路が評価された回数である。実行時間は Aer のほうが 30-40%ほど高速であることが分かった。今回の問題はゲート数が少ないため決定グラフのノード数が高々 2 万程度となったが、量子ビット数や Ansatz によってはノード数がより大きくなる可能性がある点には注意したい。

3.5 Shor

Shor のアルゴリズム [19] は因数分解を行うためのものである。たとえば 253 は 23×11 に分解することができる。

本実験で用いる Shor アルゴリズムのための量子回路は、$4n+2$ の量子ビット数を必要とする。ここで n は因数分解したい数のビット長である。たとえば 15(2 進数:1111) を因数分解する場合、$n=4$ であるから必要な量子ビット数は 18 である。なお既存研究では、アルゴリズムの終盤で登場

表 3 VQE 実験結果 (sec)

nQubits	nGates	nNodes (Ours)	nIterations	Aer	Ours
4	18	15	131	0.23	0.09
5	23	31	177	0.16	0.14
6	28	32	324	0.34	0.33
7	33	124	353	0.41	0.69
8	38	254	629	0.54	2.13
9	43	214	590	0.84	3.86
10	10	48	715	1.16	12.38
11	53	1929	786	7.62	40.47
12	58	4077	709	28.21	78.94
13	63	6525	847	117.75	190.48
14	68	16383	1083	504.69	759.90

表 4 Shor 実験結果 (sec)

N	a	nQubits	nGates	nNodes	Aer	Ours
15	2	18	12600	31	1.360	0.110
21	2	22	25330	3350	57.875	0.653
33	5	26	44466	21530	2107.088	4.017
123	2	30	118342	161735	OoM	26.556
253	2	34	204637	3620822	OoM	1197.188
511	3	38	400123	1569830	OoM	818.831

する QFT 回路の量子ビットを節約するなどし、$2n+3$[26] や $2n+2$[27] にすることが提案されている。これらのアルゴリズムは理想的な量子シミュレータ上では正しく動作するはずだが、本研究ではコードの入手のしやすさの観点から $4n+2$ の量子ビット数を必要とするアルゴリズムを用いた。

主要な数の実験結果を表 4 に記す。なお、詳細な実験結果は付録の表 A·3 も確認されたい。a は Shor アルゴリズムに与えられる入力で、N と互いに素な整数になるようになっている。本実験では適切と考えられる a を自動で計算して与えている。nNodes は、最終的な量子状態を表現するために必要な決定グラフ内のノード数である。実験ではノード数が数百万で非常に多いが、それでも我々の実装は 253 や 511 の因数分解を 10-20 分程度で終了することができた。一方で Aer の場合は 18,22,26 量子ビットでも実行時間が非常に長く、30 量子ビットを超えた時点でメモリ不足で終了してしまった。以上のように、Shor アルゴリズムは決定グラフ型が高速に処理できるアルゴリズムの 1 つであることが確認できた。

追加して、ノード数と実行時間の関係をプロットした散布図を図 7 に示す。同じアルゴリズムの場合、決定グラフ型では最終的なノード数が実行時間に比例していることを示唆する結果となっている。ゲート数ではなくノード数と実行時間に相関がある点は興味深い。しかし、最終的に必要なノード数は実行してみるまで分からないため、決定グラフ型シミュレータの実行時間を予測するとは難しいと考

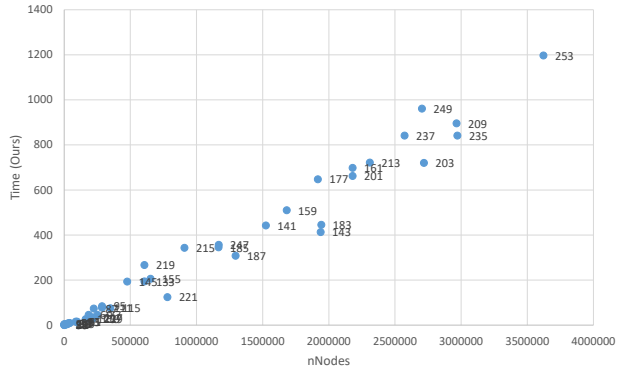

図 7 Shor: ノード数と実行時間の関係

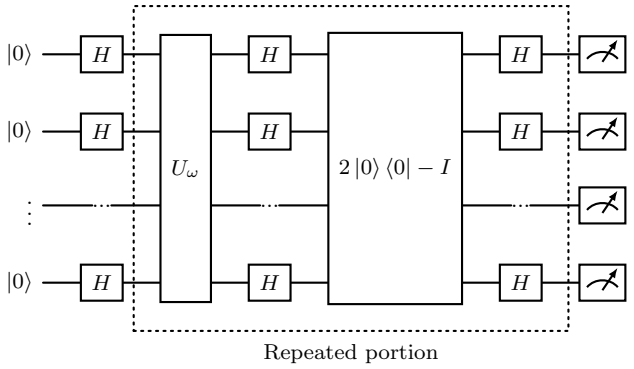

図 8 Grover アルゴリズム

えられる。

3.6 Grover

Grover のアルゴリズム [20] は、オラクルを満たす量子状態を見つけ出すためのアルゴリズムである。アルゴリズムの概略を図 8 に記す。図中でオラクルに当たる部分は U_ω である。このアルゴリズムの特徴は、同じ回路を量子状態に何度も作用させることで、徐々に目的となる状態の振幅だけを増幅させていくことである。VQE アルゴリズムと違い、繰り返し作用させられる回路にはパラメータがない。従って繰り返し部分を 1 つのユニタリゲートとしてまとめておくことができれば、シミュレーションを高速化することが出来る。

ユニタリ行列の場合、N 量子ビットに対して 2^{2N} 個の複素数分を記憶する必要がある。状態ベクトル型の場合、10 量子ビット程度でも莫大なメモリを消費することになり、事前に 1 つのユニタリ行列に回路をまとめることは困難である。一方決定グラフ型の場合、省メモリで保存できる可能性がある。従って本研究では、回路を左から右に順に実行する手法の他に、事前に繰り返し部分を 1 つにまとめておく手法の実行時間についても調査することとした。なお、Oracle は単一のランダムな量子状態とし、長さ N ビットのオラクルに対する Grover アルゴリズムの量子ビット数は $N+1$ となった。この実験設定は [14] を参考にしたものである。

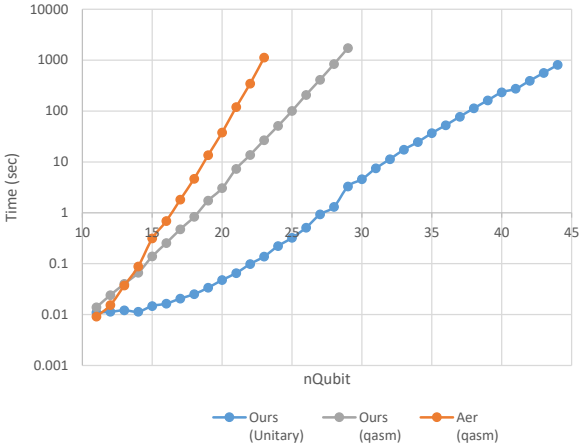

図 9 Grover 実験結果

実験結果を図 9 に示す。また、詳細な実行時間等は付録の表 A·4 も参照されたい。橙色線 (Aer, qasm) が Aer の実行時間であり、灰色線 (Ours, qasm) が我々の実装の実行時間である。さらに、繰り返し部分を事前にまとめた場合の実験結果は青色線 (Ours, Unitary) で示した。実行時間制限を 1000 秒程度としたところ、Aer は 23 量子ビットまで実験することができた。我々の実装の場合、Aer と同様のアルゴリズムならば 29 量子ビット、回路をまとめた場合には 44 量子ビットまで実行することができた。以上のように、Grover アルゴリズムは Shor アルゴリズムと同様に決定グラフ型が高速に動作するアルゴリズムであることが分かった。ただし、オラクルにパラメータ付き回転ゲートが含まれる場合には遅くなる可能性があり、注意が必要である。

4. 結論と今後の課題

4.1 結論

本研究では決定グラフ型量子シミュレータに着目し、その性能を調査することを目的として様々な実験を行った。3.2 節のランダム回路の実験結果より、たくさんのパラメータ付回転ゲートを持つようなランダム回路に適用した場合、決定グラフ型の実行時間は状態ベクトル型と比較して 100 倍以上遅くなることが分かった。これはサブグラフの共有が出来ずノード数が膨大となってしまうためである。

VQE 実験では、我々の決定グラフ型シミュレータは Qiskit Aer よりも 3,4 割程度低速であることが示された。ただし用いた例題はゲート数が数十程度と少なかったため、より複雑な Ansatz を適用した場合には Qiskit Aer と比較してより低速になる可能性がある。

一方で Shor や Grover は 40 量子ビット超でも高速に計算が終わることが判明した。Qiskit Aer は高々 30 量子ビット程度しか扱えないので、決定グラフ型が特に有用なアルゴリズムと言える。また、計算時間はゲート数に比例するわけではなく、状態ベクトルが含むノード数におよそ比例することも示された。

以上のように、決定グラフ型や状態ベクトル型量子シミュレータの利用には、前段までで明らかにした特性を理解して適用することが必要である。

4.2 今後の課題

本稿で紹介した実験結果は、すべてシングルスレッド・シングルノード環境で行われたものである。しかし、状態ベクトル型にはマルチスレッド化・マルチノード化が施されたものが多く、高速化の恩恵を受けることが出来る。筆者らは [16] で決定グラフ型量子シミュレータのマルチスレッド化手法を提案しているが、マルチノード化も検討中である。これらの技術を適用した上で、状態ベクトル型との実行時間の比較も行いたい。

実験で用いた VQE はグラフの Maxcut 問題であった。VQE は量子化学計算にも多用されているから、それらに対しても適用したい。また、決定グラフ型シミュレータで VQE 計算を高速化するための手法についても検討したい。

参考文献

[1] Arute, F., Arya, K., Babbush, R. et al.: Quantum supremacy using a programmable superconducting processor, *Nature*, Vol. 574, No. 7779, pp. 505–510 (online), DOI: 10.1038/s41586-019-1666-5 (2019).

[2] IBM: The IBM Quantum Development Roadmap, https://www.ibm.com/quantum/roadmap (2022).

[3] contributors, Q.: Qiskit: An Open-source Framework for Quantum Computing (2023).

[4] Suzuki, Y., Kawase, Y., Masumura, Y. et al.: Qulacs: a fast and versatile quantum circuit simulator for research purpose, *Quantum*, Vol. 5, p. 559 (online), DOI: 10.22331/q-2021-10-06-559 (2021).

[5] Bayraktar, H., Charara, A., Clark, D. et al.: cuQuantum SDK: A High-Performance Library for Accelerating Quantum Science (2023).

[6] Imamura, S., Yamazaki, M., Honda, T. et al.: mpiQulacs: A Distributed Quantum Computer Simulator for A64FX-based Cluster Systems (2022).

[7] Vidal, G.: Efficient Classical Simulation of Slightly Entangled Quantum Computations, *Physical Review Letters*, Vol. 91, No. 14 (online), DOI: 10.1103/physrevlett.91.147902 (2003).

[8] Orús, R.: A practical introduction to tensor networks: Matrix product states and projected entangled pair states, *Annals of Physics*, Vol. 349, pp. 117–158 (online), DOI: https://doi.org/10.1016/j.aop.2014.06.013 (2014).

[9] McClean, J. R., Boixo, S., Smelyanskiy, V. N. et al.: Barren plateaus in quantum neural network training landscapes, *Nature Communications*, Vol. 9, No. 1, p. 4812 (online), DOI: 10.1038/s41467-018-07090-4 (2018).

[10] Huggins, W., Patil, P., Mitchell, B. et al.: Towards quantum machine learning with tensor networks, *Quantum Science and Technology*, Vol. 4, No. 2, p. 024001 (online), DOI: 10.1088/2058-9565/aaea94 (2019).

[11] Bryant, R. E.: Graph-Based Algorithms for Boolean Function Manipulation, *IEEE Trans. Comput.*, Vol. 35, No. 8, p. 677–691 (online), DOI: 10.1109/TC.1986.1676819 (1986).

[12] Fujita, M., McGeer, P. C. and Yang, J. C.-Y.: Multi-Terminal Binary Decision Diagrams: An Efficient Data Structure for Matrix Representation, *Formal Methods in System Design*, Vol. 10, No. 2, pp. 149–169 (online), DOI: 10.1023/A:1008647823331 (1997).

[13] Miller, D. and Thornton, M.: QMDD: A Decision Diagram Structure for Reversible and Quantum Circuits, *36th International Symposium on Multiple-Valued Logic (ISMVL'06)*, pp. 30–30 (online), DOI: 10.1109/ISMVL.2006.35 (2006).

[14] Zulehner, A. and Wille, R.: Advanced Simulation of Quantum Computations, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 38, No. 5, pp. 848–859 (online), DOI: 10.1109/TCAD.2018.2834427 (2019).

[15] Tsai, Y.-H., Jiang, J.-H. R. and Jhang, C.-S.: Bit-Slicing the Hilbert Space: Scaling Up Accurate Quantum Circuit Simulation, *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 439–444 (online), DOI: 10.1109/DAC18074.2021.9586191 (2021).

[16] Li, S., Kimura, Y., Sato, H., Yu, J. and Fujita, M.: Parallelizing quantum simulation with decision diagrams, *2023 IEEE International Conference on Quantum Software (QSW)*, pp. 149–154 (online), DOI: 10.1109/QSW59989.2023.00026 (2023).

[17] Li, A., Stein, S., Krishnamoorthy, S. and Ang, J.: QASMBench: A Low-Level Quantum Benchmark Suite for NISQ Evaluation and Simulation, *ACM Transactions on Quantum Computing*, Vol. 4, No. 2 (online), DOI: 10.1145/3550488 (2023).

[18] Peruzzo, A., McClean, J., Shadbolt, P. et al.: A variational eigenvalue solver on a photonic quantum processor, *Nature Communications*, Vol. 5, No. 1, p. 4213 (online), DOI: 10.1038/ncomms5213 (2014).

[19] Shor, P.: Algorithms for quantum computation: discrete logarithms and factoring, *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134 (online), DOI: 10.1109/SFCS.1994.365700 (1994).

[20] Grover, L. K.: A Fast Quantum Mechanical Algorithm for Database Search, *Proceedings of the 89th Annual ACM Symposium on Theory of Computing*, STOC '96, Association for Computing Machinery, p. 212–219 (online), DOI: 10.1145/237814.237866 (1996).

[21] Quetschlich, N., Burgholzer, L. and Wille, R.: MQT Bench: Benchmarking Software and Design Automation Tools for Quantum Computing, *Quantum*, (online), available from ⟨https://www.cda.cit.tum.de/mqtbench/⟩ (2023).

[22] Yamaguchi, J., Yamazaki, M., Tabuchi, A. et al.: Estimation of Shor's Circuit for 2048-bit Integers based on Quantum Simulator, Cryptology ePrint Archive, Paper 2023/092 (2023).

[23] Cross, A. W., Bishop, L. S., Smolin, J. A. and Gambetta, J. M.: Open Quantum Assembly Language (2017).

[24] Peruzzo, A., McClean, J., Shadbolt, P. et al.: A variational eigenvalue solver on a photonic quantum processor, *Nature Communications*, Vol. 5, No. 1, p. 4213 (online), DOI: 10.1038/ncomms5213 (2014).

[25] Kandala, A., Mezzacapo, A., Temme, K. et al.: Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, *Nature*, Vol. 549, No. 7671, pp. 242–246 (online), DOI: 10.1038/nature23879 (2017).

[26] Beauregard, S.: Circuit for Shor's Algorithm Using 2n+3 Qubits, *Quantum Info. Comput.*, Vol. 3, No. 2, p. 175–185 (2003).

[27] Takahashi, Y. and Kunihiro, N.: A Quantum Circuit for Shor's Factoring Algorithm Using 2n + 2 Qubits, *Quantum Info. Comput.*, Vol. 6, No. 2, p. 184–192 (2006).

付　　録

表 A·1 QCBM 実験結果 (sec)

nQubits	Qiskit Aer	Qulacs	DDSIM	Ours
4	0.001	0.000	0.011	0.001
5	0.001	0.000	0.013	0.001
6	0.001	0.000	0.016	0.003
7	0.002	0.000	0.019	0.006
8	0.002	0.000	0.027	0.013
9	0.002	0.000	0.039	0.027
10	0.003	0.000	0.073	0.058
11	0.003	0.001	0.140	0.125
12	0.004	0.002	0.291	0.288
13	0.007	0.004	0.642	0.698
14	0.011	0.008	1.544	2.337
15	0.020	0.016	3.620	5.926
16	0.040	0.035	11.731	12.622
17	0.082	0.074	53.745	29.036
18	0.169	0.156	330.002	66.702
19	0.354	0.329		149.665
20	0.742	0.694		412.654
21	1.680	1.531		
22	3.971	3.488		
23	8.514	7.410		
24	17.649	15.751		
25	43.828	35.273		

表 A·2 QASM Bench 実験結果 (紙面の都合上 4 量子ビット以下の実験は掲載しない, sec)

Name	nQubits	nGates	nNodes (Ours)	Aer	Ours
pea5	5	96	5	0.002	0.003
qec5	5	27	8	0.002	0.003
lpn5	5	12	8	0.003	0.003
qec5	5	14	5	0.033	0.027
ec3	5	118	13	0.003	0.004
shor5	5	29	8	0.033	0.062
simon6	6	24	12	0.002	0.004
uccsd6	6	1505	33	0.007	0.020
qaoa6	6	152	63	0.003	0.008
hhl7	7	489	127	0.005	0.017
sat7	7	29	9	0.001	0.002
vqe8	8	7175	129	0.031	0.216
dnn8	8	528	255	0.005	0.040
bb8	8	27	8	0.052	0.067
qpe9	9	98	66	0.004	0.006
ising10	10	245	1023	0.004	0.064
hhl10	10	138259	1023	0.740	108.805
add10	10	35	10	0.002	0.003
sat11	11	76	25	0.003	0.004
seca11	11	72	11	0.110	0.198
cc12	12	61	12	0.034	0.177
gcm13	13	1852	57	0.022	0.072
mult13	13	21	13	0.002	0.003
bv14	14	55	14	0.005	0.007
hhl14	14	3726509		42.781	TO
qf15	15	251	1028	0.005	0.047
fac15	15	660575		15.708	TO
mult15	15	73	15	0.007	0.003
qec17	17	61	29	0.004	0.006
sq18	18	481	18	0.043	3.668
add18	18	67	18	0.032	0.006
qft18	18	802	18	0.057	0.014
bv19	19	75	19	0.041	0.009
qram20	20	45	20	0.056	0.004
bwt21	21	112833		TO	TO
cat22	22	45	43	0.020	0.011
ghz23	23	47	45	0.025	0.011
vqe24	24	2306100		TO	TO
knn25	25	40		3.438	TO
swp25	25	40		3.433	TO
ising26	26	152	22756	7.153	12.615
wstat27	27	133	53	37.189	0.014
add28	28	117	28	66.401	0.014
qft29	29	2089	29	276.569	0.033
bv30	30	108	30	217.365	0.014
cc32	32	161	32	0.122	1.240
dnn33	33	344		0.088	TO

表 A·3 Shor 実験結果 (253 まで, sec)

N	a	nQubits	nGates	nNodes (Ours)	Ours
15	2	18	12600	31	0.111
21	2	22	25330	3350	0.654
33	5	26	44466	21530	4.0172
35	2	26	55392	24223	6.305
39	2	26	61946	24223	6.874
51	2	26	55760	72	0.984
55	2	26	61904	40282	8.848
57	5	26	51365	37908	9.924
65	3	30	82681	96977	13.068
69	2	30	98779	184343	45.700
77	2	30	104290	249905	45.895
85	2	30	99412	83	2.472
87	2	30	120032	223079	73.678
91	2	30	116239	96981	15.617
93	2	30	108075	86053	15.745
95	2	30	125965	285909	83.904
111	2	30	124964	285665	77.211
115	2	30	109927	348645	74.966
119	2	30	122965	191470	29.555
123	2	30	118342	161735	26.556
129	7	34	154326	213029	27.439
133	2	34	171052	606272	193.731
141	2	34	185594	1523619	442.710
143	2	34	210041	1937432	413.271
145	6	34	160706	475205	193.335
155	2	34	200852	651755	205.479
159	2	34	220383	1680862	509.652
161	3	34	156983	2179150	698.038
177	5	34	170854	1917018	647.519
183	2	34	209924	1942908	445.453
185	3	34	182838	1166027	345.262
187	2	34	210795	1295002	307.664
201	7	34	171837	2179188	661.788
203	2	34	195403	2718391	720.366
205	3	34	180079	95	4.608
209	3	34	167050	2965174	895.473
213	2	34	186345	2309935	721.668
215	2	34	207060	908949	343.160
217	5	34	180694	213032	34.375
219	2	34	206577	606275	266.684
221	2	34	202536	779543	124.590
235	2	34	200796	2970951	841.300
237	2	34	195625	2572037	841.071
247	2	34	210574	1166847	356.437
249	11	34	188597	2702997	960.818
253	2	34	204637	3620822	1197.188

表 A·4 Grover 実験結果 (sec)

nQubits	nGates	nNodes (Unitary)	nIter	Ours (Unitary)	Ours (qasm)	Aer (qasm)
11	56	20	25	0.011	0.014	0.009
12	61	22	35	0.011	0.024	0.015
13	66	24	50	0.012	0.040	0.037
14	71	26	71	0.011	0.066	0.088
15	76	28	100	0.015	0.139	0.314
16	81	30	142	0.016	0.254	0.688
17	86	32	201	0.021	0.473	1.815
18	91	34	284	0.025	0.836	4.687
19	96	36	402	0.034	1.734	13.648
20	101	38	568	0.048	3.052	37.863
21	106	40	804	0.065	7.298	119.372
22	111	42	1137	0.098	13.731	344.673
23	116	44	1608	0.139	26.770	1120.155
24	121	46	2274	0.223	51.063	TO
25	126	48	3216	0.322	100.587	
26	131	50	4549	0.512	207.933	
27	136	52	6433	0.930	412.141	
28	141	54	9099	1.313	836.258	
29	146	56	12867	3.306	1734.069	
30	151	58	18198	4.572	TO	
31	156	60	25735	7.532		
32	161	62	36396	11.307		
33	166	64	51471	17.472		
34	171	66	72792	24.613		
35	176	68	102943	36.958		
36	181	70	145584	52.847		
37	186	72	205887	77.541		
38	191	74	291168	113.248		
39	196	76	411774	162.588		
40	201	78	582337	233.435		
41	206	80	823549	274.106		
42	211	82	1164675	394.433		
43	216	84	1647099	562.885		
44	221	86	2329350	806.320		