Vol.2020-QS-1 No.21
2020/10/16

量子計算機と古典計算機を動的に協調動作する実行システム設計

多和田 雅師[1,a)]　田中 宗[1,2]　戸川 望[3,b)]

概要：近年，量子計算機や量子インスパイアド計算機といった次世代アクセラレータのハードウェア開発が進んでいる．これら次世代アクセラレータはインタフェースや得意な計算が異なりハードウェアごとの特性を把握して使用する必要がある．ユーザには高い専門性が求められプログラミングの設計コストが高い．ハードウェアに依存する設計段階をユーザに隠蔽する設計環境を提供することでプログラミングの設計コストを下げることができる．本稿ではハードウェアごとの特性により次世代アクセラレータを協調して動作させる実行システムを設計する．協調動作に必要な情報が与えられたときの部分動作を次世代アクセラレータ・古典アクセラレータに振り分けるアルゴリズムを提案し，システムの有効性を検証する．

1. はじめに

1.1 研究背景と目的

量子を利用した計算機や量子現象をもとに設計されたアルゴリズムを使用する計算機が次世代アクセラレータとして注目されている．特に量子アルゴリズムによっては古典計算機の限界の性能を上回ることが期待されている．次世代アクセラレータはあらゆる計算に万能に有効ではなくそれぞれ得意とするアプリケーションが異なると考えられるが，その判定は専門的な知識を必要とするため使用する上での障壁となっている．

次世代アクセラレータごとにソフトウェア開発は進められおり，専門的な知識を必要とせず特定の問題に対して特定の次世代アクセラレータを使用することは比較的容易になっている．しかしながら，大きなスケールのアプリケーションを実行できる実機が存在しないため，あらゆるアプリケーションに対してソフトウェア開発が十分進んでいるとは言えない．特に種別をまたいで実行する次世代アクセラレータを選択するソフトウェアフレームワークは研究されておらず，次世代アクセラレータの使いやすさを高めることが急務である．

ユーザが専門知識を持たないまま使用するアクセラレータの選択を気にすることなくプログラムの実行が可能となるソフトウェアの開発が求められている．本稿では複数種類の次世代アクセラレータを統合的に使用可能な協調動作システムの設計を研究目的とする．

1.2 本稿の貢献

本稿の貢献は以下の通りである．

- 次世代アクセラレータ・コデザイン問題を定義する．
- 次世代アクセラレータ・コデザイン問題に取り組む実行システムのフレームワークを示す．
- 実行システムのアルゴリズムを提案・評価する．

1.3 本稿の構成

本稿の構成は以下の通りである．2 章で次世代アクセラレータおよび古典アクセラレータを定義し，その性質を紹介する．3 章で本稿が取り組むべき次世代アクセラレータ・コデザイン問題を定式化し，次世代アクセラレータの課題を述べる．4 章で次世代アクセラレータ・コデザイン問題に対する実行システムの設計及びそのアルゴリズムを提案する．5 章で提案した手法の有効性を検証する．6 章で本稿をまとめ，今後の課題を述べる．

2. 次世代アクセラレータと古典アクセラレータ

本稿では次世代アクセラレータとしてイジング型コンピュータ，Noisy Intermediate-Scale Quantum Computer (NISQ コンピュータ), 誤り耐性ゲート型量子コンピュータを対象とする．これら次世代アクセラレータは特定の問題

[1] 早稲田大学グリーン・コンピューティング・システム研究機構
Waseda Univesity, Shinjuku, Tokyo 162–0042, Japan
[2] 慶應義塾大学理工学部物理情報工学科
Keio University, Yokohama, Kanagawa 223–8522, Japan
[3] 早稲田大学基幹理工学部情報通信学科
Waseda Univesity, Shinjuku, Tokyo 169–8555, Japan
[a)] tawada@togawa.cs.waseda.ac.jp
[b)] togawa@togawa.cs.waseda.ac.jp

に対して高速に動作すると考えられている．一方で計算結果に誤りや近似を含むため，その特性を理解して使用しなければならない．

2.1 イジング型コンピュータ

イジング型コンピュータはイジングモデル [1] を入力とし，その基底状態あるいは基底状態に近いエネルギが小さい状態を出力する計算機である．イジングモデルは頂点と辺に重みのある無向グラフとして表現される．イジングモデルの各頂点に $+1$ と -1 の 2 値状態を割り当てたとき，イジングモデルのエネルギが式 (1) で定義される．頂点 i に割り当てた状態を s_i とする．頂点の重み h_i は磁場係数と呼ばれる．h_i が正の値のとき，頂点 i の状態 s_i を正の値にする強制力が働く．h_i が負の値のとき，頂点 i の状態 s_i を負の値にする強制力が働く．辺の重み $J_{i,j}$ は相互作用係数と呼ばれる．$J_{i,j}$ が正の値のとき，頂点 i の状態 s_i と頂点 j の状態 s_j を等しい値にしようとする強制力が働く．$J_{i,j}$ が負の値のとき，頂点 i の状態 s_i と頂点 j の状態 s_j を異なる値にしようとする強制力が働く．

$$\mathcal{H} = -\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} J_{i,j}s_i s_j - \sum_{i=1}^{n} h_i s_i \qquad (1)$$

ただし，頂点の数を n とする．

基底状態はエネルギ $\mathcal{H}$ を最小化する各頂点の状態 $\{s_i\}$ である．イジングモデルからその基底状態を得るためには NP 困難な計算量がかかるため，実際には多くのイジング型コンピュータは基底状態に近似したエネルギが最小ではない状態を出力する．

組合せ最適化問題をイジングモデルに変換しイジング型コンピュータに入力することで，基底状態あるいは基底状態に近いエネルギが小さい状態が出力される．出力された状態を変換して組合せ最適化問題の近似解を得る手法が研究・実用化されている [2], [3], [4], [5].

2.2 NISQ コンピュータ

NISQ コンピュータは短期的な実現可能性を見据えた誤りを許容する量子計算機である．量子を計算機に用いるとき，量子状態の不安定さが問題となる．NISQ コンピュータでは量子ビットが誤りを含むことを前提としたアルゴリズムを動作させる．特に NISQ コンピュータは量子ビットの数が小中規模なため，古典計算機と組合せて使用するアルゴリズムが注目されている．NISQ コンピュータで有効と考えられているアルゴリズムに Variational Quantum Eigensolver (VQE)[6]，Quantum Approximate Optimazation Algorithm (QAOA)[7], Quantum Circuit Learning (QCL)[8] が存在する．

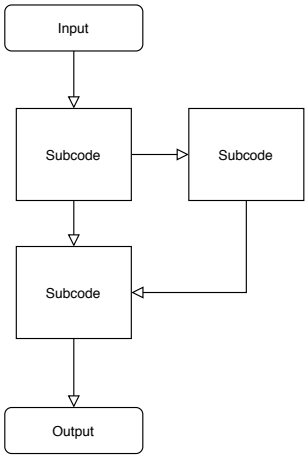

図 1 プログラムを表す DAG.

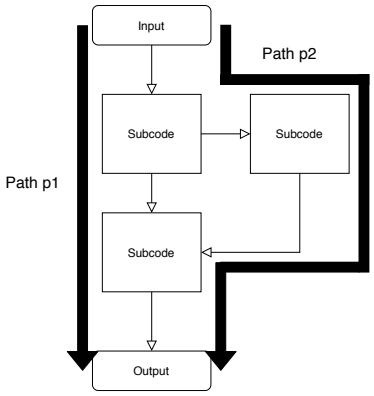

図 2 プログラムのパス p_1 と p_2.

2.3 誤り耐性ゲート型量子コンピュータ

誤り耐性ゲート型量子コンピュータは量子アルゴリズムを誤りなく実現する計算機である．量子の状態は不安定であり時間経過や演算により誤りが発生する．複数の量子ビットを誤り訂正符号化 [9] し，論理的な量子ビットを作り出すことで誤りなく量子アルゴリズムが実現できると考えられている．誤り耐性ゲート型量子コンピュータで有効な量子アルゴリズムに HHL アルゴリズム [10]，Shor の素因数分解アルゴリズム [11] が存在する．

2.4 古典アクセラレータ

古典アクセラレータは次世代アクセラレータ以外の既存の計算機デバイスを指す．古典アクセラレータは CPU, GPU, FPGA が存在する．

3. 次世代アクセラレータ・コデザイン問題

プログラムの動作を高速化させるアクセラレータとして，特に次世代アクセラレータに期待が集まっている．次世代アクセラレータは動作させるプログラムに対して得手不得手が存在するため，実行前に適したアクセラレータを判定して動作を割り当てる必要がある．本章では次世代アクセラレータ・古典アクセラレータへの動作割り当てを次

世代アクセラレータ・コデザイン問題として定式化する．

3.1 次世代アクセラレータ・コデザイン問題の定式化

本節では次世代アクセラレータ・コデザイン問題の計算モデルを与える．プログラムを有向非巡回グラフ (directed acyclic graph; DAG) で表す．DAG の粒度はアクセラレータで実行可能な塊の最小単位とし，この最小単位を部分プログラムと呼ぶ．DAG を $G = (V, E)$ とする．

プログラムの入力集合を I，出力集合を O とする．入力 $i \in I$ から出力 $o \in O$ までの有向道（directed path）をプログラムのパスと呼ぶ．図 2 ではプログラムのパスは p_1 と p_2 の 2 個存在する．プログラムのパス集合を P，プログラムのパスを $p \in P$ とする．プログラムのパス p は入力と出力を含めた部分プログラムの列とみなせる (式 (2))．

$$p = \langle i, v_1, v_2, \ldots, v_{|p|}, o \rangle \qquad (v_i \in V) \qquad (2)$$

ただし，$|p|$ は入力と出力を除いたパス p に含まれる部分プログラムの数である．

次世代アクセラレータ・古典アクセラレータを $a \in A$ とする．部分プログラム v からアクセラレータ a への割り当てを表す関数を $w(v)$ とする．このとき $w(v) = a$ とする．

部分プログラム v をアクセラレータ a で実行したときの動作時間を $t(v, a)$ とする．あるパス p の実行時間 T_p は式 (3) で与えられる．

$$T_p = \sum_{i=1}^{|p|} t(v_i, w(v_i)) \qquad (3)$$

プログラムの動作時間 T_{all} は式 (4) で与えられる．ただし，max は最大値関数とする．

$$T_{all} = \max_{p \in P} T_p \qquad (4)$$

次世代アクセラレータではプログラムの実行結果として誤りが含まれることがある．実行結果の誤りを定量的に示す指標として近似度を定義する．近似度は最適な結果に対する誤りの含まれる結果の比に相当し，1 以上の値をとる．このとき誤りの含まれる結果が最適な結果に一致する場合は近似度を 1 と定義する．部分プログラム v をアクセラレータ a で実行したときの近似度を $e(v, a)$ とする．あるパス p の近似度 E_p は式 (5) で計算される．

$$E_p = \prod_{i=1}^{|p|} e(v_i, w(v_i)) \qquad (5)$$

このとき部分プログラム間の計算誤りは累積されるため，パスの近似度は部分プログラムの近似度の積として計算モデルを定義する．プログラムの解の近似度 E_{all} は式 (6) で計算される．

$$E_{all} = \max_{p \in P} E_p \qquad (6)$$

このとき複数のパスの近似度を比較すると，特定パスの近似度の大きさが支配的になると考えられるため，プログラム全体の近似度はパスの近似度の最大値として計算モデルを定義する．

次世代アクセラレータ・コデザイン問題はプログラムの DAG G が与えられたときに動作時間 T_{all} と近似度 E_{all} を同時に最小化する割り当て $w(v)$ を得る多目的最適化問題である．

3.2 次世代アクセラレータ・コデザインの課題

実際には部分プログラム v をアクセラレータ a で動作させたときの動作時間 $t(v, a)$ と近似度 $e(v, a)$ は自明ではない．現在多くの次世代アクセラレータは汎用の計算機から離れた外部の処理系に存在する．今後も次世代アクセラレータは外部インタフェースの形式で提供されると考えられ，動作時間 $t(v, a)$ や近似度 $e(v, a)$ は待ち時間などにより動的に変化する．アクセラレータが動的に協調動作可能な実行システムを設計する必要がある．

4. アクセラレータが協調動作する実行システム設計

4.1 システム

次世代アクセラレータ・コデザイン問題を考えるフレームワークとして図 3 のシステムを考える．システムは協調モジュール, 推定モジュール, 実行モジュールの 3 個のモジュールによって構成されている．システムはプログラムを入力とし，部分プログラムごとに各モジュールを実行し，最終的にプログラムの実行結果を出力する．

(1) 協調モジュール
　協調モジュールは部分プログラムに対し各次世代アクセラレータ・古典アクセラレータでの動作割当を担うモジュールである．

(2) 推定モジュール
　推定モジュールはアクセラレータごとに存在し，部分プログラムに対してアクセラレータ上で実行する場合の計算時間と計算精度を推定し，出力するモジュールである．

(3) 実行モジュール
　アクセラレータそのものは外部に存在し，実行モジュールはアクセラレータ上で部分プログラムを実行するインタフェースとなるモジュールである．

システムの外部モジュールとして次世代アクセラレータ・古典アクセラレータが存在する．アクセラレータは実行モジュールを通してのみアクセスされ，アクセラレータ間のインタフェースの差や実行前後の変換処理は実行モジュールによって隠蔽される．

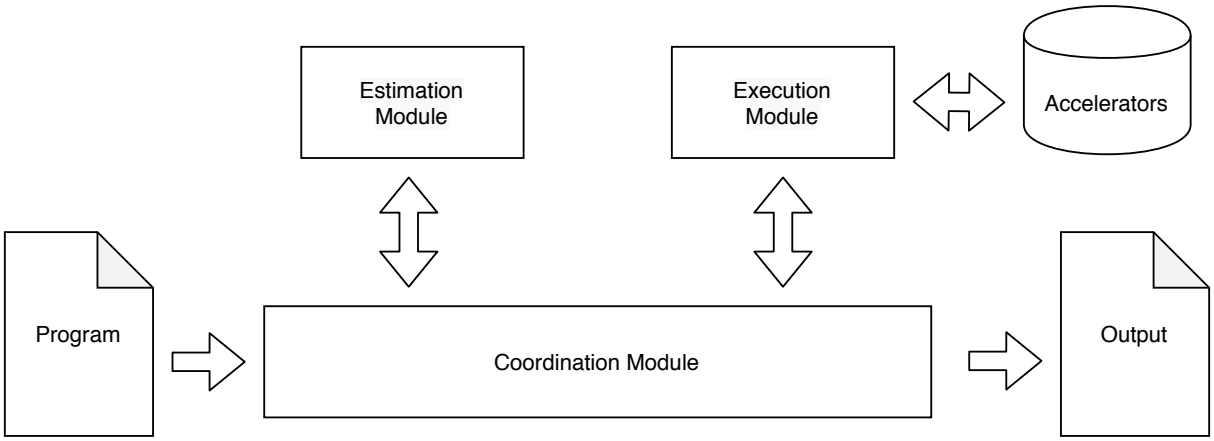

図 3 次世代アクセラレータ・古典アクセラレータを協調して動作させるシステム.

4.2 提案手法

本稿では協調モジュールを提案の対象とし，推定モジュールと実行モジュールは与えられると仮定する．部分プログラムごとに推定モジュールの推定結果をもとに動的にアクセラレータを決定し実行する協調モジュールの動作を提案する．

提案手法は以下のプロセスに従って動作する．図 4 に部分プログラムに対する協調モジュールの動作フローを示す．部分プログラムおよび前後の部分プログラムの入出力をインタフェースとして持つ．

1) 推定プロセス
2) アクセラレータ決定プロセス
3) 実行プロセス
4) 出力プロセス

アクセラレータ決定プロセスではアルゴリズム 1 を用いて使用するアクセラレータを決定する．

アルゴリズム 1 アクセラレータ決定アルゴリズム.

Input: $v \in V$, $t(v, a_i), e(v, a_i) \quad (1 \leq i \leq |A|)$
Output: $a_{opt} = w(v)$

```
F ⇐ t(v, a_1) + αe(v, a_1)
opt ⇐ 1
for i = 2 ... |A| do
    F_tmp ⇐ t(v, a_i) + αe(v, a_i)
    if F_tmp < F then
        F ⇐ F_tmp
        opt ⇐ i
    end if
end for
```

推定プロセスでは部分プログラムとその入力を各アクセラレータの推定モジュールに入力し，実行時間と近似度を推定する．アクセラレータによっては実行パラメタにより実行時間と近似度が異なるため，推定モジュールは複数の実行時間と近似度の推定結果を返すことを仮定する．推定モジュールが複数の実行パラメタの推定結果を返したら，協調モジュールは仮想的にアクセラレータが複数存在するように扱う．実際に実行するアクセラレータは単一のため 1 種類のアクセラレータを仮想的に複数種類として扱っても問題ない．

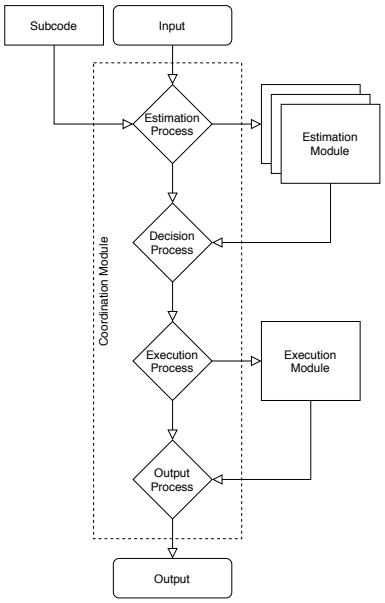

図 4 部分プログラムに対する協調モジュールの動作フロー（提案手法）.

アクセラレータ決定プロセスでは推定モジュールが推定した実行時間と近似度をもとに，アルゴリズム 1 を用いて使用するアクセラレータを決定する．推定結果は実行時間と近似度のペアで返されるため，図 5(a) のように実行時間と近似度の座標平面上の点として表せる．ここで 3.1 節で定式化された多目的最適化問題の目的関数である実行時間と近似度の重み和を計算し，目的関数をスカラ値で与える最適化問題に変換する．実行時間に対する近似度の重みの比を α とする．α が小さいとき，図 5(b) に示すように実行時間の重みが大きいことを表し，実行時間が短く近似度が大きいと推定されるアクセラレータが選ばれる．α が大きいとき，図 5(c) に示すように近似度の重みが大きいことを表し，実行時間が長く近似度が小さいと推定されるアク

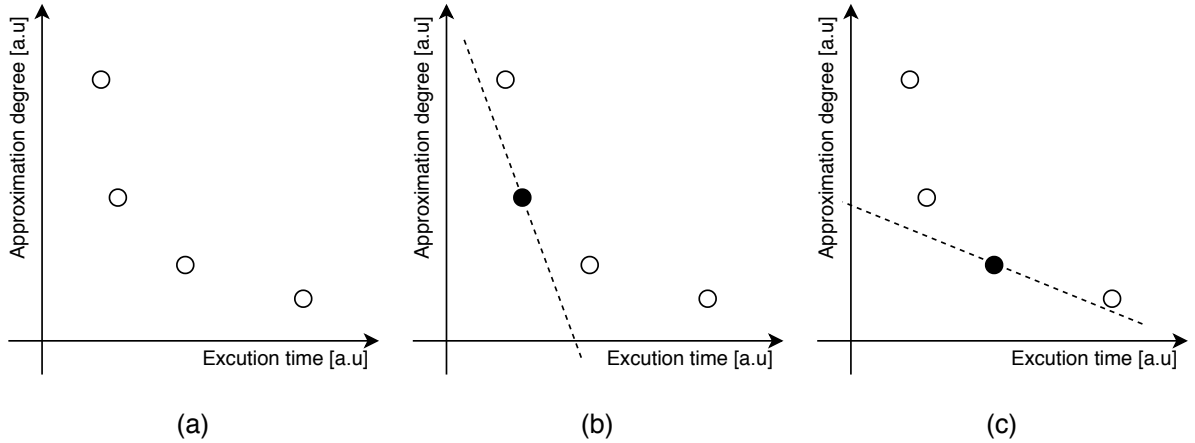

図 5 (a) アクセラレータごとの推定された実行時間と近似度．(b) 実行時間に大きい重みをつけた場合に選択されるアクセラレータ．(c) 近似度に大きい重みをつけた場合に選択されるアクセラレータ．

セラレータが選ばれる．実際には部分プログラムごとにアクセラレータ決定プロセスを実行するため，プログラム全体の実行時間と近似度を目的関数に使用できない．部分プログラムの実行時間 $t(v, w(v))$ と近似度 $e(v, w(v))$ を用いて，単一の目的関数 F を持つ最適化問題を解く．目的関数 F を式 (7) で与える．ただし，部分プログラムを v，部分プログラムからアクセラレータの割り当てを $w(v)$ とする．

$$F = t(v, w(v)) + \alpha e(v, w(v)) \qquad (7)$$

アルゴリズム 1 により，推定モジュールの出力 $t(v, a_i)$,$e(v, a_i)$ に対して，F を最小化する割り当て $w(v)$ が得られる．

実行プロセスでは実行モジュールを通じてアクセラレータを動作させる．実行モジュールは部分プログラムをアクセラレータに入力できる形式へ変換する．次世代アクセラレータのアプリケーションに相当する内容であり，2 章に示すように多くの変換方法・アルゴリズムが存在する．

出力プロセスではアクセラレータの実行結果を出力する．

5. 評価実験

5.1 計算機環境

提案手法を実行する計算機の OS は Windows 10, プロセッサは Intel Core i7 2.7GHz, メモリは 16.0GB を用いた．提案手法の実装に用いた言語は Python 3.6.9 である．

使用した次世代アクセラレータは以下の通りである．2 通りのイタレーション回数に応じて仮想的に 2 種類の次世代アクセラレータを使用した．

- 次世代アクセラレータ: イジング型コンピュータ [12]
- 実行アルゴリズム: レプリカ交換方式 [13]
- レプリカ数: 80 個
- イタレーション回数: 1,000,000 回 または 100,000,000 回

古典アクセラレータとして OS は CentOS 7.6, プロセッサは Intel Xeon Platinum 2.5GHz, メモリは 1.0TB を用いた．古典アクセラレータの実装に用いた言語は C である．

5.2 実験内容

アルゴリズム 1 を用いて協調モジュールのアクセラレータ決定プロセスを実装し，式 (7) の重み α が異なるとき使用するアクセラレータが変化するのかを検証した．

システムへ入力するプログラムとして，QAPLIB[14] ベンチマークを解くプログラムを用いた．DAG に含まれる部分プログラムの数は 1 個である．

実行モジュールは古典アクセラレータでは総当り法を実装し，次世代アクセラレータでは二次割当問題を解くイジングモデルの作成プログラムを実装した．推定モジュールは予め実行しておいた出力の実行時間と近似度そのものを仮想的に推定結果として用いた．

5.3 実験結果

重み $\alpha = 1, 10000$ のときの式 (7) で定義される目的関数 F と選択されたアクセラレータを表 1，表 2 に示す．近似度の重みにより選択される次世代アクセラレータが変化し，目的関数 F に最適なアクセラレータを選ぶのに提案手法が有効であることを確認できた．推定モジュールの出力となる実行時間と近似度の推定結果を表 3 と表 4 に示す．

6. おわりに

本稿ではハードウェアごとの特性により次世代アクセラレータを協調して動作させる実行システムを設計した．協調動作に必要な情報が与えられたときの部分動作を次世代アクセラレータ・古典アクセラレータに振り分けるアルゴリズムを提案し，システムの有効性を検証した．

今後の課題は協調モジュールへ入力されたプログラムを

表 1 アルゴリズム 1 実行結果（$\alpha = 1$[s]）.

ベンチマーク	目的関数 F [s]	選択されたアクセラレータ
chr12a	7.242	次世代アクセラレータ（イタレーション回数 1,000,000 回）
chr12b	6.724	次世代アクセラレータ（イタレーション回数 1,000,000 回）
chr12c	7.196	次世代アクセラレータ（イタレーション回数 1,000,000 回）

表 2 アルゴリズム 1 実行結果（$\alpha = 10000$[s]）.

ベンチマーク	目的関数 F [s]	選択されたアクセラレータ
chr12a	10006.242	次世代アクセラレータ（イタレーション回数 1,000,000 回）
chr12b	10022.262	次世代アクセラレータ（イタレーション回数 100,000,000 回）
chr12c	10021.561	次世代アクセラレータ（イタレーション回数 100,000,000 回）

表 3 次世代アクセラレータと古典アクセラレータの実行時間 [s].

ベンチマーク	古典アクセラレータ	次世代アクセラレータ イタレーション回数 1,000,000 回	次世代アクセラレータ イタレーション回数 100,000,000 回
chr12a	229.184	6.242	20.896
chr12b	229.167	5.651	22.262
chr12c	229.170	6.190	21.561

表 4 次世代アクセラレータと古典アクセラレータの近似度．括弧内の数値は出力された解の値を表す.

ベンチマーク	最適解	古典アクセラレータ	次世代アクセラレータ イタレーション回数 1,000,000 回	次世代アクセラレータ イタレーション回数 100,000,000 回
chr12a	9552	1 (9552)	1 (9552)	1 (9552)
chr12b	9742	1 (9742)	1.073 (10458)	1 (9742)
chr12c	11156	1 (11156)	1.006 (11224)	1 (11156)

部分プログラムに分解し，各部分プログラムに対して次世代アクセラレータを適用することである．

謝辞 本研究の一部は，内閣府総合科学技術・イノベーション会議の戦略的イノベーション創造プログラム（SIP）「光・量子を活用した Society 5.0 実現化技術」（管理法人：量子科学技術研究開発機構）によって実施されました。

参考文献

[1] Ising, E.: Beitrag zur Theorie des Ferromagnetismus, *Zeitschrift für Physik*, Vol. 31, pp. 253–258.
[2] Lucas, A.: Ising formulations of many NP problems, *Frontiers in Physics*, Vol. 2, pp. 1–15 (2014).
[3] Lucas, A.: Hard combinatorial problems and minor embeddings on lattice graphs, *Quantum Information Processing*, Vol. 18, pp. 1–38 (2019).
[4] Tanahashi, K., Takayanagi, S., Motohashi, T. and Tanaka, S.: Application of Ising Machines and a Software Development for Ising Machines, *Journal of the Physical Society of Japan*, Vol. 88, No. 6, pp. 061010–1–10 (2019).
[5] Tanaka, S., Tamura, R. and Chakrabarti, B. K.: *Quantum spin glasses, annealing and computation*, Cambridge University Press (2017).
[6] Peruzzo, A.，McClean, J.，Shadbolt, P.，Yung, M.-H.，Zhou, X.-Q.，Love, P. J.，Aspuru-Guzik, A.，Jeremy LO' brien : A variational eigenvalue solver on a photonic quantum processor, *Nature communications*, Vol. 5, p. 4213 (2014).
[7] Farhi, E., Goldstone, J. and Gutmann, S.: A quantum approximate optimization algorithm, *arXiv preprint arXiv:1411.4028* (2014).
[8] Mitarai, K., Negoro, M., Kitagawa, M. and Fujii, K.: Quantum circuit learning, *Physical Review A*, Vol. 98, No. 3, p. 032309 (2018).
[9] Gottesman, D.: Stabilizer codes and quantum error correction, *arXiv preprint quant-ph/9705052* (1997).
[10] Harrow, A. W., Hassidim, A. and Lloyd, S.: Quantum algorithm for linear systems of equations, *Physical review letters*, Vol. 103, No. 15, p. 150502 (2009).
[11] Shor, P. W.: Algorithms for quantum computation: discrete logarithms and factoring, *Proceedings 35th annual symposium on foundations of computer science*, Ieee, pp. 124–134 (1994).
[12] Tsukamoto, S., Takatsu, M., Matsubara, S. and Tamura, H.: An accelerator architecture for combinatorial optimization problems, *Fujitsu Sci. Tech. J*, Vol. 53, No. 5, pp. 8–13 (2017).
[13] Hukushima, K. and Nemoto, K.: Exchange Monte Carlo method and application to spin glass simulations, *Journal of the Physical Society of Japan*, Vol. 65, No. 6, pp. 1604–1608 (1996).
[14] Burkard, R. E., Karisch, S. E. and Rendl, F.: QAPLIB–a quadratic assignment problem library, *Journal of Global optimization*, Vol. 10, No. 4, pp. 391–403 (1997).