Vol.2021-QS-2 No.9
2021/3/29

量子計算機をアクセラレータとして用いるスケジューリングアルゴリズム設計

多和田 雅師[1,a)]　戸川 望[2,b)]

概要：スケール性能や実行速度の点から任意の計算を量子計算機で実行するのではなく, 古典計算機に対する優位となる特定の計算に対し量子計算機をアクセラレータとして用いることが考えられる. 複数のアクセラレータを実行可能な場合にはプログラムに対し部分的なアクセラレータの割当を行うだけでなく, 周辺の処理と協調する統合的な割当をする必要がある. 本稿ではプログラムの処理の流れに適したアクセラレータ割当・実行するスケジューリングアルゴリズムを提案する. 提案アルゴリズムはプログラムに対し部分的な単位ではなく周辺の処理を考慮してアクセラレータを割当し, 大域的に最適となるようプログラムを実行する.

1. はじめに

1.1 研究背景と目的

量子計算機をプログラムから呼び出して実行する形式の専用アクセラレータとして用いる研究が存在する. 量子計算機は特定のアプリケーションに対し古典計算機より優位な計算ができると期待されている. 一方で, 量子計算機はスケーリングや実行速度といった技術的課題から汎用計算機として実用可能な領域に達していない. スケール性能や実行速度の点から, 任意の計算を量子計算機で実行することは現実的ではない. 古典計算機に対して優位となる計算のみを量子計算機で実行することで双方の性能を引き出せる. 古典計算機の FPGA や GPU にも計算内容によって得意不得意が存在し, 量子古典に関わらず得意な計算をする計算機をアクセラレータと呼ぶ.

プログラムからアクセラレータを呼び出す方法は自明ではないため, 最適な戦略を考える必要がある. アクセラレータで実行可能なプログラムの単位を部分プログラムと呼ぶ. アクセラレータに複数の種類がある場合, 部分プログラムをどのアクセラレータで実行するかの組合せは多数存在する. 特にアクセラレータがどんなアプリケーションを得意としているかを判別するには専門知識が必要となるため, 人の手を介さず ソフトウェアが自動で部分プログラムとアクセラレータを割当てることが望ましい.

実行直前の推定結果に基づきアクセラレータと部分プログラムを割当てる動的なスケジューリングが研究されている [1]. 量子計算機の多くは計算機そのものが遠隔地にあり, ネットワークごしに実行するリモート API 形式である. 計算機の実行可否や実行に掛かる時間はリアルタイムに変化するため, プログラムの実行時に動的な推定が必要となる. 既存手法 [1] は動的な推定を用いて, 部分プログラムに対して割当てるアクセラレータを最適化しているが, 注目している部分プログラムに対する瞬時的な最適化のみ行われ周辺の部分プログラムを通しての最適化はされていない.

本稿は複数の部分プログラムの処理フローを考慮して, 割当てるアクセラレータを決定する手法を提案する.

1.2 本稿の貢献

本稿の貢献は以下の通りである.

- 部分プログラムに対するアクセラレータ割当手法を提案する.

2. 量子・古典アクセラレータ

本章では特定アプリケーションに特化した専用計算機として量子・古典アクセラレータを紹介する (図 1). イジング計算機, Noisy Intermediate-Scale Quantum Computer (NISQ), Fault-Tolerant Quantum Computation (FTQC) が存在する. これらは特定のアプリケーションを得意とする専用アクセラレータである. 一方で計算結果に誤りや近似を含む可能性があり, 近似の量を考慮して実行する必要がある.

[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] 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

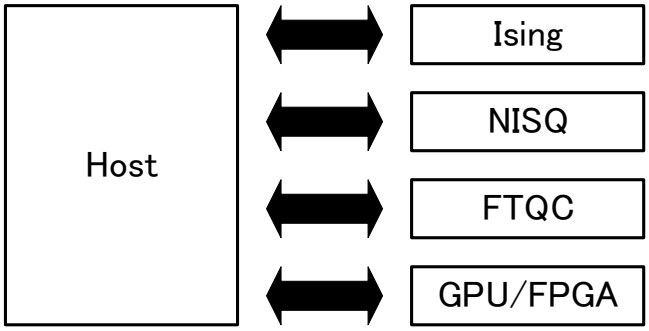

図 1 アクセラレータを用いた実行環境.

2.1 イジング計算機

イジング計算機はイジングモデル [2] を入力とし，そのエネルギが最小となる基底状態あるいはエネルギが小さい準基底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．

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

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，実際には多くの場合にイジング計算機は基底状態に近似したエネルギが最小ではない準基底状態を出力する．

組合せ最適化問題をイジングモデルに変換しイジング計算機に入力することで, 基底状態あるいは準基底状態が出力される．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 [3], [4], [5], [6].

2.2 NISQ

NISQ 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 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 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 で有効と考えられているアルゴリズムに Variational Quantum Eigensolver (VQE)[7]，Quantum Approximate Optimazation Algorithm (QAOA)[8], Quantum Circuit Learning (QCL)[9] が存在する．

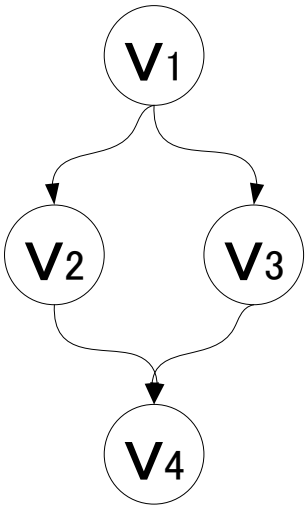

図 2 プログラムを表す DAG.

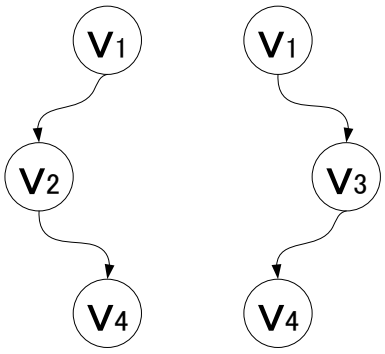

図 3 プログラムのパス p_1 と p_2.

2.3 FTQC

FTQC 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 [10] 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．FTQC で有効な量子アルゴリズムに HHL アルゴリズム [11]，Shor の素因数分解アルゴリズム [12] が存在する．

2.4 古典アクセラレータ

本稿では CPU, GPU, FPGA を古典アクセラレータと呼ぶ. 古典アクセラレータは古典アルゴリズムに対して高速に動作し, 精度の保証されたアルゴリズムでは誤りなく計算を実行できる.

3. アクセラレータ割当問題

プログラムの動作を高速化させるため，量子計算機を専用アクセラレータとして用いる需要が存在する．専用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当てる必要がある．本章ではアクセラレータを用いてプログラムを実行するための, 部分プログラムとアクセラレータの割当をアクセラレータ割当問題 [1] として定義する．

3.1 アクセラレータ割当問題の入力

本節ではアクセラレータ割当問題の入力を定義する．アクセラレータ割当問題の入力は複数のアクセラレータをまたがって実行されるプログラムであり, データ構造は有向非巡回グラフ (directed acyclic graph; DAG) で表される．DAG の粒度はアクセラレータで実行可能な最小単位とし, この最小単位を部分プログラムと呼ぶ．

部分プログラム v, v' に対し, v と v' の実行順序に直接的な依存関係があるとき有向辺 $e = (v, v')$ で表す．部分プログラムの集合 V と有向辺集合 E を用いてプログラムを表す DAG を $G = (V, E)$ とする．有向辺の出力とならない部分プログラムを始点と呼び, 始点集合を I とする．有向辺の入力とならない部分プログラムを終点と呼び, 終点集合を O とする．

始点から終点まで有向辺を辿った有向道（directed path）をプログラムのパスと呼ぶ．図 3 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 は依存関係を考慮した部分プログラムの列とみなせる (式 (2))．

$$p = \langle v_{p_1}, v_{p_2}, \ldots, v_{p_{|p|}} \rangle \qquad (v_{p_i} \in V, v_{p_1} \in I, v_{p_{|p|}} \in O) \tag{2}$$

ただし，$|p|$ はパス p に含まれる部分プログラムの数である．

3.2 アクセラレータ割当問題の出力

本節ではアクセラレータ割当問題の出力であり最適化の対象である決定変数を説明する．アクセラレータを $a \in A$ とする．部分プログラム v を実行するアクセラレータを $w(v)$ とする．部分プログラム v をアクセラレータ a で実行すると割当するとき $w(v) = a$ である．アクセラレータ割当問題の出力はすべての部分プログラム v に対する $w(v)$ の値である．

3.3 アクセラレータ割当問題の目的関数

本節ではアクセラレータ割当問題の目的関数となる実行時間と近似度を説明する．

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．

$$T_p = \sum_{i=1}^{|p|} t(v_{p_i}, w(v_{p_i})) \tag{3}$$

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．

$$T_{all} = \max_{p \in P} T_p \tag{4}$$

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．

$$E_p = \prod_{i=1}^{|p|} e(v_{p_i}, w(v_{p_i})) \tag{5}$$

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．

$$E_{all} = \max_{p \in P} E_p \tag{6}$$

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．

アクセラレータ割当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当 $w(v)$ $(v \in V)$ を得る多目的最適化問題である．

3.4 動的な割当の必要性

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．現在多くの量子計算機からなる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．今後もアクセラレータはリモート API 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．

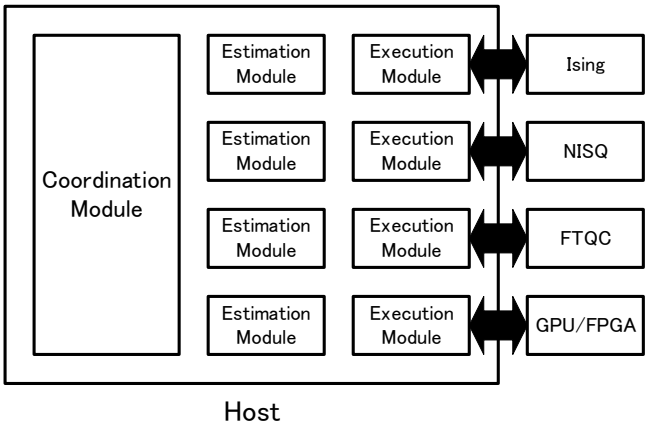

図 4 アクセラレータ割当問題に対するフレームワーク.

4. アクセラレータ割当問題のフレームワークと既存手法 [1]

アクセラレータ割当問題では部分プログラムとアクセラレータの割当を最適化するが, その割当による目的関数への影響は動的に変化するため予めスケジューリングすることは難しい. 目的関数に対する割当の寄与を動的に推定する推定モジュールを有するフレームワークを用いる.

4.1 アクセラレータ割当問題のフレームワーク

アクセラレータ割当問題を考えるフレームワークとして図 4 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アクセラレータ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存在するアクセラレータへは実行モジュールがリモート API を用いてアクセスし, 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.

4.2 既存手法 [1]

アクセラレータ割当問題は多目的最適化問題であり, 目的関数が実行時間と近似度の複数存在する. [1] は目的関数の重みの比 α を用いて, 目的関数をスカラー値として定義し直している. アクセラレータ割当問題のフレームワークに対し, 推定モジュールと実行モジュールは与えられると仮定している.

[1] のアルゴリズムをアルゴリズム 1 に示す. 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. アクセラレータ割当問題として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.

$$F = t(v, w(v)) + \alpha e(v, w(v)) \tag{7}$$

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.

アルゴリズム 1 [1] のアルゴリズム.

Input: $v \in V$, $t(v, a_i), e(v, a_i) \quad (1 \leq i \leq |A|)$
Output: $a_{opt} = w(v)$
  $F \Leftarrow t(v, a_1) + \alpha e(v, a_1)$
  $opt \Leftarrow 1$
  for $i = 2 \ldots |A|$ **do**
    $F_{tmp} \Leftarrow t(v, a_i) + \alpha e(v, a_i)$
    if $F_{tmp} < F$ **then**
      $F \Leftarrow F_{tmp}$
      $opt \Leftarrow i$
    end if
  end for

4.3 既存手法の問題点

既存手法では部分プログラムに対して動的にアクセラレータを割当しており, 周辺の部分プログラムを考慮していないためプログラム全体では最適な割当とならない可能性がある.

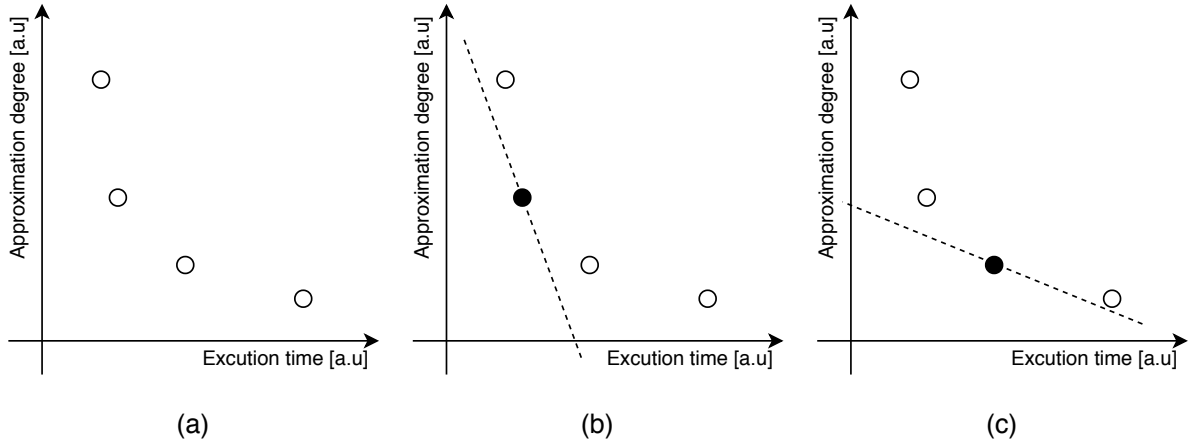

図 5 [1] で提案された重みパラメタ α を用いた多目的関数のスカラ値化手法．(a) アクセラレータごとの推定された実行時間と近似度．(b) 実行時間に大きい重みをつけた場合に選択されるアクセラレータ．(c) 近似度に大きい重みをつけた場合に選択されるアクセラレータ．

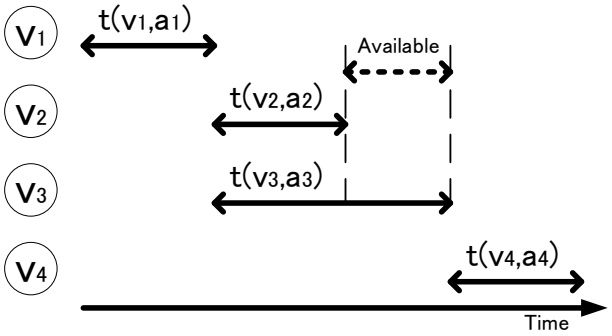

図 6 提案手法のスケジューリング．

5. 提案手法

本章では割当対象の部分プログラムだけでなく後続の処理を考慮し割当するスケジューリングアルゴリズムを提案する．既存手法の問題点は部分プログラムに対するアクセラレータの割当に周辺の部分プログラムの割当情報を用いていないことである．実際にはプログラムの実行には依存関係が存在するため，待ち時間は発生する可能性があるため実行時間が小さいよりも計算制度が高い割当を優先するべき状況がある．図 6 に提案手法のスケジューリングを示す．図 6 は部分プログラム v_1, v_2, v_3, v_4 に対し，アクセラレータ a_1, a_2, a_3, a_4 を割当てていることを示している．v_2, v_3 の実行に対し，v_4 の実行は依存関係があるため先んずることができない．v_4 の実行のためには v_2 と v_3 の両方の終了を待つ必要がある．複数パスの共有部分プログラムで律速されるためパスごとの実行時間を揃えるようスケーリングする割当アルゴリズムを提案する．

6. おわりに

本稿では部分プログラムに対するアクセラレータ割当問題に対し，処理の流れを考慮したスケジューリングアルゴリズムを提案した．今後の課題はスケジューリングアルゴリズムの実行時間と近似度から最適性を定量化することが挙げられる．

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] 多和田雅師，田中宗，戸川望：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，情報処理学会研究報告, vol. 2020-QS-1, no. 21，pp. 1–6 (2020).
[2] Ising, E.: Beitrag zur Theorie des Ferromagnetismus, *Zeitschrift für Physik*, Vol. 31, pp. 253–258.
[3] Lucas, A.: Ising formulations of many NP problems, *Frontiers in Physics*, Vol. 2, pp. 1–15 (2014).
[4] Lucas, A.: Hard combinatorial problems and minor embeddings on lattice graphs, *Quantum Information Processing*, Vol. 18, pp. 1–38 (2019).
[5] 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).
[6] Tanaka, S., Tamura, R. and Chakrabarti, B. K.: *Quantum spin glasses, annealing and computation*, Cambridge University Press (2017).
[7] 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).
[8] Farhi, E., Goldstone, J. and Gutmann, S.: A quantum approximate optimization algorithm, *arXiv preprint arXiv:1411.4028* (2014).
[9] Mitarai, K., Negoro, M., Kitagawa, M. and Fujii, K.: Quantum circuit learning, *Physical Review A*, Vol. 98, No. 3, p. 032309 (2018).

[10] Gottesman, D.: Stabilizer codes and quantum error correction, *arXiv preprint quant-ph/9705052* (1997).
[11] 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).
[12] 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).