一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

分散量子演算の動的要求に対する最適資源割当

田中　信元†　　上山　憲昭†

† 立命館大学情報理工学　〒565–0456 大阪府吹田市河田 4–5–6
E-mail: †is0677kh@ed.ritsumei.ac.jp, ††kamiaki@fc.ritsumei.ac.jp

あらまし　量子計算資源を複数ノードへ分散配置する分散型量子演算（DQC）は量子ビット規模の制約を克服する有力な手法である．しかしノード間のエンタングルメント資源が有限なため，要求量に応じた動的資源配分が計算性能を大きく左右する．特に，複数アプリケーションが同時実行される環境では，量子回路要求とネットワーク資源需要が時間変動するため，効率的な資源管理が課題となる．そこで本稿では処理時間の短縮を目的として，DQC ネットワークにおける要求受入と論理回路割当の最適化方式を提案する．従来手法では静的状況を前提とし，局所ルールベースで最適化するため動的要求下では性能劣化が起こり得る．そこで本稿では，強化学習により状況に応じて最適な割り当て判断を逐次学習させるアプローチを採用する．

キーワード　分散量子演算，強化学習

Optimal Resource Allocation for Dynamic Demands in Distributed Quantum Computing

Nobuyuki TANAKA† and Noriaki KAMIYAMA†

† R&D Division, Osaka Corporation　4–5–6 Kawada, Suita-shi, 565–0456 Japan
E-mail: †is0677kh@ed.ritsumei.ac.jp, ††kamiaki@fc.ritsumei.ac.jp

Abstract　Distributed quantum computing (DQC), which distributes quantum computing resources across multiple nodes, is a powerful approach to overcoming the limitations imposed by the scale of qubits. However, since entanglement resources between nodes are finite, dynamic resource allocation based on demand significantly impacts computational performance. Particularly in environments where multiple applications run concurrently, quantum circuit demands and network resource requirements fluctuate over time, making efficient resource management a critical challenge. Therefore, this paper proposes an optimization method for request acceptance and logical circuit allocation in DQC networks, aiming to reduce processing time. Conventional methods assume static conditions and optimize using local rule-based approaches, potentially leading to performance degradation under dynamic demands. Consequently, this paper adopts an approach that sequentially learns optimal allocation decisions based on the situation using reinforcement learning.

Key words　Distributed Quantum Computing, Reinforcement Learning

1. ま え が き

近年，量子計算機の発展に伴い，単一の量子計算機では処理が困難な大規模量子回路を，複数の量子処理装置（QPU: Quantum Processing Unit）を協調させて実行する分散量子演算（DQC: Distributed Quantum Computing）が注目されている．DQC 環境では，QPU 間で量子もつれを生成しながら計算を進める必要があり，計算資源の割当や実行順序の決定が性能に大きく影響する [1]．しかし，実際の DQC 環境では，量子回路要求がオンラインに到着し，各 QPU の利用可能状況や量子メモリのコヒーレンス時間が時間とともに変化するため，あらかじめ最適なスケジュールを決定することは困難である．さらに，非局所量子操作に伴う量子通信遅延や忠実度劣化を考慮する必要があり，従来の静的スケジューリング手法や単純なヒューリスティックでは十分な性能を得ることが難しい [2].

本稿では，このような DQC 環境におけるタスク割当問題に対して，強化学習を用いた動的スケジューリング手法を提案する．提案手法では，分散量子回路を論理タスクに分割しタスク間の依存関係を有向非巡回グラフとして表現することで，逐次的な意思決定を可能とする．また，未完了ジョブの待ち時間および量子操作に伴う忠実度劣化を考慮した報酬関数を設計し，長期的な処理性能の向上を目指す．性能評価として，シミュレー

ション実験により，提案手法が従来手法と比較し，平均処理時間および処理可能ジョブ数の観点から有効であることを示す．

2. 関 連 研 究

2.1 分散量子演算における相互接続ネットワークのためのリソース管理と回路スケジューリング

Rahrami らは，複数の量子プロセッサ（QPU）が量子ネットワークによって接続された DQC 環境を対象とし，量子回路の実行要求をネットワーク資源と計算資源の制約下でどのように割り当て・実行するかという問題を定式化している [3]．量子回路を構成するゲート操作を局所ゲートと非局所ゲートに分類し，特に非局所ゲートの実行に必要となるエンタングルメント生成および通信遅延を考慮したスケジューリング手法を提案している．ネットワーク上のリンク容量やエンタングルメント生成率といった物理的制約を明示的にモデル化し，回路全体の実行時間（makespan）を最小化することを目的としている点が特徴である．

一方で，本論文では回路要求は事前に既知であり，動的に到着する要求や不確実性を伴う環境は考慮されていない．また，最適化は主としてルールベースまたは数理最適化に基づいて行われており，環境変動に適応的に振る舞う学習的手法は導入されていない．そのため，要求が動的に発生する DQC 環境においては，柔軟性の面で課題が残る．

2.2 分散量子演算のための最適確率的資源配分

Ngoenriang らは，量子計算要求の到着や量子リンクの状態が確率的に変動する状況を想定し，長期的な性能指標を最適化する資源割り当て戦略の設計を提案している [4]．量子計算要求を確率過程としてモデル化し，エンタングルメント資源や計算資源の割り当てをマルコフ決定過程（MDP: Markov Decision Process）として定式化している．その上で，期待コスト（例えば遅延や失敗確率）を最小化する最適方策の構造を理論的に解析し，確率的環境下でも性能保証が可能であることを示している点が本稿の特徴である．

しかしながら，この論文で扱われているモデルは比較的抽象化されており，具体的な量子ネットワークトポロジや回路レベルの実行時間までは詳細に考慮されていない．また，最適方策の導出にはモデルの正確な事前知識が必要であり，環境モデルが不完全または時間的に変化する場合への適応性には課題が残る．

3. 提 案 方 式

3.1 概要と特徴

本稿では，DQC 要求を既存研究に基づいて細粒度タスクへ分割し，各タスク間の依存関係から重み付き有向非巡回グラフ（DAG: Directed Acyclic Graph）を生成する．各 DAG ノードは，物理制約に基づいて実行可能な QPU の集合を持つ．強化学習は，複数の DAG が同時に存在する環境において，実行可能な DAG ノードおよびその実行先 QPU を選択し，QPU 内の実行順序を動的に決定する．DQC では大規模な量子回路を複数の QPU に分割して実行するため，非局所ゲートの実行やエンタングルメント生成を伴う複雑な依存関係が発生する．これにより，ジョブの実行時間は単純な回路深さではなく QPU 間通信や待ち時間に大きく依存する．

提案手法の特徴は，DQC ジョブから実行依存関係を表す DAG を自動生成する機構と，その DAG に基づいて強化学習により資源割当および実行判断を最適化する点にある．量子コンパイラによって分散化された量子回路を入力とし，各 QPU 上で実行される量子プログラムを自動生成する．強化学習エージェントはシステム状態として各 QPU の負荷状況，量子ネットワークの混雑度，生成された DAG の構造情報を観測し，QPU の割当やジョブの実行順序に関する行動を選択する．

3.2 処理の流れ

本稿では，DQC における実行管理問題に対し，既存研究で提案されているタスク分割手法 [5] と，強化学習による動的資源配分を組み合わせた設計を採用する．ただし，量子ビットの局在性や非局所ゲートに伴う物理制約を考慮し，強化学習が担う意思決定の範囲を明確に制限することで実行可能性とモデルの整合性を保証する．

まず，DQC ジョブがシステムに到着すると，既存研究に基づき，量子回路は細粒度の論理的タスクへと分割される．この分割には，タスク間の類似性や通信量を考慮したクラスタリング手法が用いられ，各クラスタは量子回路中の意味的にまとまりのある処理単位を表す．

次に，各クラスタに含まれるタスク間の依存関係を解析し，その結果をもとに DAG を自動生成する．この DAG は，タスクの実行順序に関する制約を明示的に表現するものであり，ジョブ内部において守られるべき因果関係のみを表す．したがって，この段階では実行順序の最適化や資源割当は行わず，DAG 生成は決定的な処理として実施される．生成された DAG に含まれる各ノードは，量子ビットの配置や非局所ゲートの実行可能性といった物理的制約に基づき，実行可能な QPU の集合を持つものとして定義される．

強化学習は，この制約付き環境のもとで実行管理を担当する．具体的には，強化学習エージェントは複数の DAG が同時に存在する状況において，現在実行可能な DAG ノードの中から次に実行するノードを選択し，さらにそのノードを実行可能 QPU の集合の中からどの QPU に割り当てるかを決定する．また，同一 QPU 上で複数のノードが待機している場合には，それらの実行順序についても強化学習によって動的に決定する．このように，本設計では DAG によって「どのタスクがどの順序で実行可能か」という制約を明示化し，強化学習によって「複数ジョブ間で限られた QPU 資源をどのように共有するか」を最適化する．

3.3 DQC 要求の定義

提案手法の中で用いる記号を表 1 にまとめる．式 (1) に示すように，時刻 t に到着する DQC 要求 J は分散量子回路 C，使用される論理量子ビット集合 Q，非局所ゲート（CatEnt）情報 $\mathcal{L}$，ジョブ到着時刻の情報 t^{arr} を有する．

$$J = (C, Q, \mathcal{L}, t^{\mathrm{arr}}) \tag{1}$$

ただし，C は分散量子回路であり，以下のようにゲートの列として表される．

表 1　提案手法において用いる主な記号の一覧

記号	説明	記号	説明
J	DQC 要求（ジョブ）を表す．分散量子回路や量子ビット集合などを含むタプル	E	論理タスク間の依存関係を表すエッジ集合
		$Q(\tau_k)$	論理タスク τ_k が使用する量子ビットの集合
C	量子ゲート列として表現される分散量子回路	t	離散時間ステップを表すインデックス
Q	DQC 要求 J において使用される論理量子ビットの集合	s_t	時刻 t における強化学習環境の状態
		J_t^{active}	時刻 t においてシステム内で処理中の DQC 要求の集合
$\mathcal{L}$	非局所ゲート（CatEnt）の情報		
t_{arr}	DQC 要求 J の到着時刻	T_t^{ready}	時刻 t において実行可能状態にある論理タスクの集合
g_i	量子回路 C を構成する i 番目の量子ゲート		
type_i	1Q（単一量子ビットゲート），2Q（2 量子ビットゲート），CatEnt（分散操作）のいずれかを意味するゲート g_i の種類	QPU_t	各 QPU の次回使用可能時刻を要素とするベクトル
		D_t^{decoh}	各 QPU における量子メモリの残存コヒーレンス時間を表すベクトル
qubits_i	ゲート g_i が作用する量子ビットの集合	a_t	実行する論理タスクと割り当て先 QPU の組を意味する，時刻 t における行動
loc_i	ゲート g_i の実行に関与可能な QPU の集合		
τ_k	同一 QPU 上で連続して実行される量子ゲート列におけるを表す k 番目の論理タスク	(τ, q)	論理タスク τ を QPU q に割り当てる行動 a_t の具体的内容
T	DAG のノード集合として用いられる論理タスクの集合	r_t	時刻 t における強化学習エージェントの報酬
		α, β, γ	報酬関数の重み係数
$\mathcal{P}(x)$	ゲート x，もしくは論理タスク x が実行可能な QPU の集合	$W_j(t)$	時刻 t における DQC 要求 j の待ち時間
		T_t	時刻 t において評価対象となる論理タスクの集合
$\lvert Q_{\text{cand}}\rvert$	論理タスク τ_k が必要とする量子ビット数		
Q_{th}	1 つの論理タスクが単一 QPU に割り当て可能であることを高い確率で保証するための保守的な上限値	ΔF_τ	論理タスク τ の実行に伴う忠実度の減少量
		F_τ	論理タスク τ に対応する量子状態の忠実度
		F_{th}	許容される忠実度の下限値（忠実度閾値）
$G = (\mathcal{T}, E)$	論理タスクをノードとする有向非巡回グラフ（DAG）	II	条件が真のとき 1，偽のとき 0 を返す指示関数

$$C = (g_1, g_2, ..., g_n) \tag{2}$$

各ゲート g_i は $type_i$（1Q（単一の量子ビットに対して作用する量子ゲート），2Q（2 つの量子ビットに同時に作用する量子ゲート），CatEnt（異なる QPU に配置された量子ビット間で 2Q を実現するための分散操作）），$qubits_i$（使用量子ビット），loc_i（実行に関与する QPU）で表す．

$$g_i = (type_i, qubits_i, loc_i) \tag{3}$$

3.4　論理タスクへの分割

本稿では，量子回路を構成するゲート列を順に走査し，単一の QPU 上で無理なく実行できる範囲を一つの論理タスクとしてまとめる．論理タスク τ は同一 QPU 上で実行され，外部 QPU との同期を必要としない連続ゲート列として定義する [6]．次の最大部分列として定義する．

$$\tau_k = (g_i, g_{i+1}, ..., g_j) \tag{4}$$

ただし，QPU 一貫性と同期不要性を満たし，CatEnt の通信制御を含む処理は，必ず単独の論理タスクとして切り出す設計とする．また，論理タスクの必要量子ビット数が一定以上を超えたとき，新しい論理タスクを作成するようにする．論理タスクに分割するアルゴリズムを Algorithm1 に示す．

Algorithm 1 論理タスク分割（量子ビット数制約付き）

Require: Quantum circuit $C = \{g_1, g_2, \ldots, g_N\}$
Ensure: Logical task set $\mathcal{T}$

$\mathcal{T} \leftarrow \emptyset$
Initialize empty task τ
Initialize empty qubit set Q_τ
for $i = 1$ **to** N **do**
  $Q_{\text{cand}} \leftarrow Q_\tau \cup Q(g_i)$
  if τ is empty **then**
    $\tau \leftarrow \{g_i\}$
    $Q_\tau \leftarrow Q(g_i)$
  else if g_i is not CatEnt **and** $\mathcal{P}(g_i) = \mathcal{P}(\tau)$ **and** $|Q_{\text{cand}}| \le Q_{\text{th}}$ **then**
    $\tau \leftarrow \tau \cup \{g_i\}$
    $Q_\tau \leftarrow Q_{\text{cand}}$
  else
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\tau\}$
    Insert synchronization boundary
    $\tau \leftarrow \{g_i\}$
    $Q_\tau \leftarrow Q(g_i)$
  end if
end for
$\mathcal{T} \leftarrow \mathcal{T} \cup \{\tau\}$
return $\mathcal{T}$

3.5　DAG の自動生成

DAG 生成は既存研究の回路依存関係解析に基づく [7]．本研究ではそれを動的 RL 環境に適合させるため，論理タスク粒度に再設計した．DAG 生成の入力として，1 つの DQC ジョブに対し必要論理量子ビット数，量子ゲート列（1 量子ビットゲート，2 量子ビットゲート，分散 CNOT（CatEnt）），各ゲートが作用する量子ビットの集合が与えられる．DAG の生成では，量子ビットの使用関係に基づいて依存関係を付与することで，有向非巡回グラフを構築する．このとき，量子回路の時間的な実行順をそのまま直列に並べるのではなく，並列実行可能な操作は同一レベルに配置可能な構造として表現する点が特徴である．

また，DAG の生成に関するアルゴリズムを Algorithm2 に示す．DQC ジョブ到着時に，空の DAG を 1 つ生成し，ノード集合およびエッジ集合を空集合として初期化する．各ゲート操作について，Algorithm1 で得られた各論理タスク τ_k を DAG の

ノードとして用いる．新たに生成した論理タスクノードに対し，同一の量子ビットを使用する直前のタスクが存在する場合，そのタスクから現在のタスクへ有向辺を追加する．ただし，分散CNOT（CatEnt）操作の場合は制御量子ビットおよび標的量子ビットの両方について依存関係を付与する．これにより，量子ビットの排他的使用制約が DAG 上で明示的に表現される．依存関係の追加後，タスクノードを DAG に登録する．この処理をすべてのゲート操作について繰り返すことで，DAG 全体が構築される．

Algorithm 2 DAG Generation from Logical Tasks

Require: Logical task set $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_K\}$
Ensure: DAG $G = (\mathcal{T}, E)$
$E \leftarrow \emptyset$
for $i = 1$ **to** $K - 1$ **do**
  for $j = i + 1$ **to** K **do**
    if $Q(\tau_i) \cap Q(\tau_j) \neq \emptyset$ **then**
      $E \leftarrow E \cup \{(\tau_i \rightarrow \tau_j)\}$
    else if τ_i contains CatEnt **and** $\mathcal{P}(\tau_i) \cap \mathcal{P}(\tau_j) \neq \emptyset$ **then**
      $E \leftarrow E \cup \{(\tau_i \rightarrow \tau_j)\}$
    end if
  end for
end for
Remove transitive edges from E
return $G = (\mathcal{T}, E)$

3.6 強化学習の手法

強化学習の目的は DAG で表現された複数の論理的タスクを，複数 QPU に対して順序・割当を最適化し全体の処理時間を最小化することである．本稿では強化学習の手法として PPO（Proximal plicy Optimization）を使用する [8]．シミュレーションでは状態・行動が高次元かつ連続的に変化し，DAG という制約構造を持つ．動的到着ジョブを扱うため，学習の安定性が最重要である．PPO は方策更新が制限されており学習が安定し，即時報酬ベース設計と相性が良く，状態・行動の拡張に強いという特徴を持つ．短期的に遅延を増やす行動を低報酬とし，長期的に DAG のクリティカルパスを短縮する行動を高累積報酬として学習することで，強化学習は DAG のクリティカルパスを短縮する方向に自然に誘導される．

3.6.1 状　　態

時刻 t のときの状態 s_t は式（5）で定義される．

$$s_t = (\mathcal{J}_t^{\text{active}}, \mathcal{T}_t^{\text{ready}}, \mathcal{QPU}_t, \mathcal{D}_t^{\text{dechoh}}) \tag{5}$$

$\mathcal{QPU}_t$ により，各 QPU がいつ使用可能になるかを把握することで，将来の待ち時間を考慮した割り当て判断が可能となる．また $\mathcal{D}_t^{\text{dechoh}}$ により，量子状態の劣化が進行している QPU を回避するなど，忠実度低下を抑制する行動を学習できる．これにより，s_t によって，強化学習は古いジョブを優先し，ボトルネックを回避し，量子状態が壊れにくくする．

3.6.2 行　　動

行動としては到着している DAG の中から 1 つの実行可能ノードを選び，割り当てる QPU を選ぶ．

$$a_t = (\tau, q) \tag{6}$$

3.6.3 エピソード

ポアソン分布に従ってランダムに到着する 200 個の DQC 要求を 1 エピソードとし，1000 エピソード行う．

3.6.4 報　　酬

本報酬は待ち時間最小化と忠実度維持を通じて，間接的にクリティカルパス短縮を誘導する．報酬関数は式 (7) で表される．

$$r_t = -\alpha \sum_{j \in J_t^{\text{active}}} W_j(t) - \beta \sum_{\tau \in T_t} \Delta F_\tau - \gamma \sum_{\tau \in T_t} \mathit{II}\,(F_\tau < F_{\text{th}}) \tag{7}$$

本稿では，忠実度閾値違反を最優先で回避するため $\gamma \gg \alpha, \beta$ と設定した．第一の項は，未完了ジョブの待ち時間に比例したペナルティであり，平均処理時間の削減を直接的に目的とする．DQC 環境では複数のジョブが同時に滞留するため処理遅延が累積しやすく，待ち時間を明示的に抑制することが全体性能の向上に不可欠である．この項を導入することで，エージェントはジョブ滞留を避ける行動を学習し，平均完了時間の短縮が促進される．

第二の項は，タスク実行に伴う忠実度劣化量に応じたペナルティである．量子状態の忠実度はタスクの進行に伴って連続的に低下するため，最終結果のみを評価する報酬設計では劣化の兆候を十分に捉えることができない．そこで，本稿では微小な忠実度劣化も逐次的に評価対象とすることでエージェントが将来的な劣化を予測し，先回りして抑制するような資源割り当て方策を学習できるように設計した．

第三の項は，忠実度が所定の閾値を下回った場合に与えられる大きな負の報酬である．忠実度閾値の違反は，量子状態の崩壊や計算結果の無効化を意味し，処理時間の短縮と比較しても許容できない失敗である．そのため，本稿では閾値違反を強く排除する目的で，この項を導入した．これにより，致命的な忠実度劣化を伴うスケジューリング方策は学習過程で自然に排除され，安全性を確保した方策探索が可能となる．

以上の三項は，それぞれ平均処理時間の短縮，忠実度劣化の抑制，致命的失敗の回避という異なる役割を担っており，DQC スケジューリングにおける主要な要求を網羅的に反映している．これらを同一の報酬関数内で評価することで，安全性を維持しつつ高速なスケジューリングを実現することを目指す．

4. 性能評価方法

4.1 ネットワーク

本稿では，DQC を想定した量子ネットワーク上において，動的に到着する複数の DQC ジョブを処理する環境を構築する．ネットワークは Python の NetworkX ライブラリを用いて生成し，各ノードを量子プロセッサユニット（QPU），各エッジを量子通信リンクとしてモデル化する．

QPU 数は 30 とし，各 QPU が保有する計算可能な量子ビット数は一様分布に従い 15 から 25 の範囲でランダムに割り当てる．ネットワークトポロジは Erdős–Rényi（ER）ランダムグラフモデル $G(N, p_{\text{ed}})$ に基づいて生成する [3]．ここで $N = 30$ はノード数，$p_{\text{ed}} = 0.6$ は任意の 2QPU 間に量子通信リンクが存在する確率を表す．

4.2 比較対象

本稿では，提案手法の有効性を評価するため，既存手法および異なる強化学習ベースの手法を比較する．比較対象として，以下の二つの手法を採用した．

MIPS に基づく資源割り当て手法

一つ目の比較対象として，関連研究において提案されている MIPS（Mixed Integer Programming-based Scheduling）手法を用いる [3]．この手法では，DQC における資源割り当ておよび回路スケジューリング問題を，混合整数計画問題として定式化し，最適解を求めることにより資源を割り当てる．MIPS 手法は，与えられた要求集合に対して理論的に最適な解を導出できる点に特徴があり，静的または準静的な要求環境において高い性能を示す．本稿では，この MIPS 手法を，各スナップショットで最適解を求めた準静的手法として用いる．

強化学習による戦略選択手法

二つ目の比較対象として，強化学習を用いて，あらかじめ定義された複数の資源割り当て戦略の中から最適な戦略を選択する手法を用いる．この手法では，資源割り当て自体は既存のルールベースまたはヒューリスティックな戦略に基づいて行われ，強化学習エージェントはネットワーク状態に応じて，どの戦略を適用するかを選択する役割を担う．これにより，環境の変化に応じた柔軟な戦略切り替えが可能となる一方で，個々の資源割り当て行動は事前に設計された戦略に制約される．本稿では，この手法を「間接的に資源割り当てを最適化する強化学習手法」と位置づけ，提案手法との比較を行う．

4.3 評価手法

本稿では，提案する強化学習による直接資源割り当て手法の有効性を検証するため，DQC 要求の処理性能および計算コストの観点から評価を行う．評価指標として，DQC 要求の発生から処理完了までに要する時間の平均値に着目する．DQC 要求が発生してから処理が完了するまでの総処理時間を測定し，その平均値の変化を評価する．評価は，DQC 要求の特性を変化させた複数の条件下で行う．第一に，1 要求当たりに必要な量子ビット数の平均値を変化させる．第二に，DQC 要求の到着率を変化させる．これにより，負荷の異なる動的環境下において，要求規模の違いが処理時間に与える影響を比較する．

4.3.1 量子ビット数の変化

1 要求当たりに必要な量子ビット数の違いが DQC 要求の処理時間および資源割り当て計算時間に与える影響を評価するため，要求規模を段階的に変化させた比較実験を行う．要求規模は各 DQC 要求が必要とする論理量子ビット数 $|Q|$ によって定義する．本稿のネットワークモデルでは，各 QPU が保持する量子ビット数をその QPU 上で同時に使用可能な論理量子ビット数とみなす．各 QPU の論理量子ビット数は一様分布 $U(15, 25)$ に従って割り当てられており，平均的には 1QPU あたり約 20 量子ビットを使用可能である．

DQC 要求は必ず複数の QPU を用いるものとし，最小構成として 2QPU を使用する要求から全体の約 3 分の 1 に相当する 10QPU を使用する大規模要求までを対象とする．この条件に基づき，必要論理量子ビット数 $|Q|$ と使用 QPU 数の関係を対応付け，段階的な要求規模を設定する．具体的には平均的な QPU の量子ビット数に基づき，必要論理量子ビット数 $|Q|$ の平均を 40, 60, 80, 100, 120, 150, 180 の 7 段階に設定する．また，この時の DQC 要求の到着率を 10要求 $/s$ と設定する．

このように要求規模を段階的に増加させることで，小規模要求では通信や待ち時間の影響が支配的となる一方，大規模要求では資源競合や割り当て戦略の違いが顕著に現れると考えられる．本稿ではこれらの条件下で提案手法と比較手法を適用し，要求規模の違いが処理時間および計算時間に与える影響を定量的に評価する．

4.3.2 到着率の変化

DQC 要求の到着率がシステム性能に与える影響を評価するため，到着率を変化させた場合の比較を行う．4.4 節の量子ビット数の変化と同様に，到着率を 7 段階に設定し，低負荷状態から高負荷状態までを段階的に変化させる．DQC 要求の到着はポアソン過程に従うものとし，到着率を 5, 8, 11, 14, 17, 20, 23 要求 $/s$ とする．

10要求 $/s$ は 4.4 節における評価条件と一致する基準値であり，それを中心として到着頻度が低い場合および高い場合の両方を評価対象とする．このとき，各 DQC 要求が必要とする要求量子ビット数の平均は 100 に固定する．これにより，要求規模の影響を排除し，到着率の変化が資源競合，待ち時間，および実行時間に与える影響を純粋に評価できる．以上の条件のもと，到着率の増加に伴う平均実行時間の変化を測定し，要求が集中する高負荷環境においても提案手法が安定して性能を維持できるかを既存手法との比較により検証する．

5. 数値結果

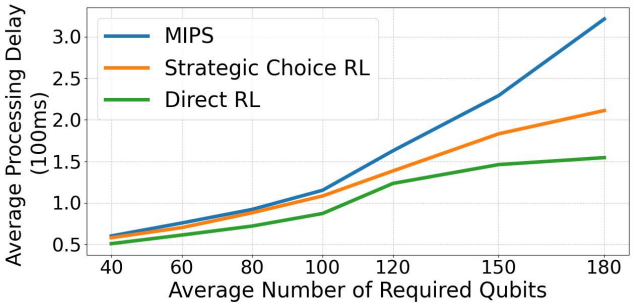

図 1　平均要求量子ビットの変化に対する平均処理時間の変化

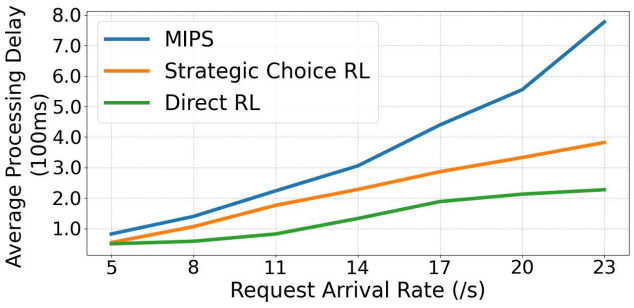

図 2　到着率の変化に対する平均処理時間の変化

量子ビット数を増加させた場合（図 1），すべての手法において平均処理時間は増加する傾向が確認された．量子ビット数の

増加は，使用する QPU 数の増大や非局所ゲート（CatEnt）の増加を引き起こし，量子通信時間および同期待ち時間を押し上げる要因となる．そのため，処理時間が増加すること自体は避けられない．一方で，量子ビット数の変化に対する処理時間の上昇は比較的緩やかであり，各ジョブ内部の DAG 構造の拡大にほぼ比例した増加に留まっている．

これに対して，到着率を増加させた場合には（図 2），量子ビット数変化の場合と比べて処理時間の上昇率が大きく，特に高到着率条件では急激な性能劣化が観測された．到着率の上昇は，同時に存在する未完了ジョブ数の増加を引き起こし，QPU の待ち行列が形成されやすくなる．その結果，待ち時間が累積的に増大し，処理時間全体に対して非線形な影響を与える．したがって，処理時間に対する影響という観点では，量子ビット数の増加よりも到着率の増加の方が支配的であるといえる。

このような条件下において，提案手法は他の二手法と比較して一貫して短い平均処理時間を達成している．MIPS に基づく手法では，各スナップショットで最適化問題を解くため高負荷環境では資源割り当て計算時間そのものが増大し，さらに割り当て結果が即時的な混雑状況を十分に反映できない場合がある．そのため，要求規模増大や高到着率などの条件下では，処理時間が大幅に悪化する．また，間接 RL（戦略選択型強化学習）手法は，MIPS と比べて計算時間を抑制できるものの，実際の資源割り当ては事前に定義された戦略に依存している．そのため，DAG の進行状況や QPU の局所的な混雑といった細粒度な状態変化に十分に追従できず，負荷が高い条件では処理時間の増加を抑えきれない．

これに対して提案手法は，実行可能な論理タスクと割り当て先 QPU を強化学習によって逐次決定するため，DAG の進行状態や QPU の利用状況を直接反映した柔軟な資源割り当てが可能である．特に到着率が高い状況では，QPU の混雑を回避する行動や，クリティカルパス上のタスクを優先する行動が学習されることで，待ち時間の増加が相対的に抑制されている．結果，厳しい動的環境においても処理時間の増加を抑制できており，他の二手法と比較して優れたスケーラビリティと安定性を有しているといえる．

6. ま と め

本稿では，DQC を対象とした資源割り当て問題に着目し，動的に到着する DQC 要求に対して，DAG の進行状態および各 QPU の利用状況を状態として直接入力し，実行可能タスクと割り当て先 QPU を逐次決定する強化学習手法を提案した．提案手法はあらかじめ定義された戦略の選択に依存せず，細粒度な資源割り当てをオンラインで行える点に特徴がある．シミュレーション評価では，量子ビット数および DQC 要求の到着率を変化させた条件下において提案手法を MIPS に基づく資源割り当て手法および戦略選択型強化学習手法と比較した．その結果，提案手法は全ての評価条件において平均処理時間を最も短く抑えることが確認された．特に，大規模 DQC 要求や高到着率条件といった高負荷環境下においても処理時間の増加が比較的緩やかであり，高い安定性とスケーラビリティを有することが示された．処理時間の内訳分析から MIPS 手法では資源割り当て計算時間および待ち時間が性能劣化の主因となる一方，間接 RL 手法では計算時間は抑制されるものの，戦略選択に基づく制御の制約により待ち時間の削減に限界があることが明らかとなった．これに対して，提案手法は割り当て計算時間を一定に保ちながら動的な資源競合を適切に緩和できる点で優れている．以上の結果から，本稿で提案した手法は動的な DQC 要求を扱う将来の量子ネットワーク環境において有効かつ実用的なアプローチであると結論付けられる．

謝辞　本研究は JSPS 科研費（25K03113, 23K28078）の助成を受けたものである．

文　献

[1] J. C. Boschero, N. M. Neumann, W. van der Schoot, T. Sijpesteijn, and R. Wezeman, “Distributed quantum computing: Applications and challenges,” in *Intelligent Computing-Proceedings of the Computing Conference*. Springer, 2025, pp. 100–116.

[2] N. K. Chandra, E. Kaur, and K. P. Seshadreesan, “Network operations scheduling for distributed quantum computing,” in *2024 IEEE 6th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA)*. IEEE, 2024, pp. 506–515.

[3] S. Bahrani, R. D. Oliveira, J. M. Parra-Ullauri, R. Wang, and D. Simeonidou, “Resource management and circuit scheduling for distributed quantum computing interconnect networks,” *arXiv preprint arXiv:2409.12675*, 2024.

[4] N. Ngoenriang, M. Xu, S. Supittayapornpong, D. Niyato, H. Yu *et al.*, “Optimal stochastic resource allocation for distributed quantum computing,” *arXiv preprint arXiv:2210.02886*, 2022.

[5] R. Parekh, A. Ricciardi, A. Darwish, and S. DiAdamo, “Quantum algorithms and simulation for parallel and distributed quantum computing,” in *2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS)*, 2021, pp. 9–19.

[6] F. Burt, K.-C. Chen, and K. K. Leung, “A multilevel framework for partitioning quantum circuits,” *Quantum*, vol. 10, p. 1984, 2026.

[7] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, “Resource management with deep reinforcement learning.” New York, NY, USA: Association for Computing Machinery, 2016, p. 50–56.

[8] M. Sgambati, A. Vakanski, and M. Anderson, “Decentralized distributed proximal policy optimization (dd-ppo) for high performance computing scheduling on multi-user systems,” *arXiv preprint arXiv:2505.03946*, 2025.