量子プログラミングコンテスト QCoder

雪吉 稀允[1,a)]　高田 隼矢[2,b)]　山田 怜央[3,c)]　芭瀬田 保[4,d)]　加藤 拓己[5,e)]　御手洗 光祐[6,7,f)]
藤井 啓祐[6,7,8,g)]　石川 直樹[1,h)]

概要：本稿では、量子プログラミングコンテスト QCoder の概要とその実装について紹介する。QCoder は、従来の競技プログラミングの形式を量子プログラミングに適用したものであり、参加者は与えられた問題に対して、適切な量子アルゴリズムをプログラムとして実装することが求められる。提出されたプログラムは、シミュレータ上で実行され、その評価結果が参加者にフィードバックされる。このような競技性を介した学習プロセスは、量子情報科学教育においても効果的なアプローチとなる可能性がある。

キーワード：量子計算、量子プログラミング、競技プログラミング

1. はじめに

競技プログラミングは頭脳スポーツの一種であり、参加者は提示された問題文にしたがってプログラムを記述し、その正確さや効率性に応じて得点を競い合う。具体的には、整数や文字列などの入力が与えられ、それらの入力をもとに問題の要求する何らかの結果をテキストとして出力するアルゴリズムをプログラムとして実装する。作成されたプログラムはネットワークを通じてジャッジサーバに提出され、事前に準備されたテストケースに対してプログラムを実行することでその正否が評価される。現在、代表的な競技プログラミングコンテストには Codeforces [1] や Meta Hacker Cup [2]、情報オリンピック [3] などがあり、各コンテストは数万人規模の参加者を集めている。

競技プログラミングで求められるアルゴリズムやデータ構造の知識、そしてそれらをプログラムとして実装する能力は情報科学分野の広範な領域にわたっており、コンテスト参加者はコンテストへの参加を通じてこれらを習得していく。こうした競技プログラミングの教育的な側面は「オンラインジャッジ」や「自動採点」、「e ラーニング」などの文脈で広く研究されてきた [4], [5], [6], [7]。自動採点システムは時間がかかり、誤りの発生しやすい人手によるプログラム評価を自動化し、運用の効率化を可能にする。また、自動採点システムに競技性を取り入れることで、学習者の学習意欲や学習効果が向上することを示唆する結果が得られている [8], [9], [10], [11]。

一般にこれまでの競技プログラミングでは、古典コンピュータ上でプログラムを実行することが前提とされてきた。古典コンピュータの性能向上は 1965 年に「集積回路上のトランジスタ密度は約 2 年ごとに倍増する」と予測したムーアの法則に従っている [12]。ムーアの法則は半世紀にわたり有効であったが、物理的な限界からその終焉が近づいていると考えられている [13]。

このような状況を背景に、古典計算に代わる新たな計算パラダイムとして量子計算が注目されている。これは、最良の古典アルゴリズムよりも効率的な量子アルゴリズムが複数発見されているためである [14], [15]。近年量子コンピュータの性能と規模は急速に向上しており、将来的にはいくつかのタスクにおいて古典コンピュータの性能を上回ることが期待される。

量子コンピュータ上で計算を行う際には古典計算と同様にプログラミングを通じて命令を記述する。このためのツールとして量子ソフトウェア開発キットや量子プログラミング言語があり、Qiskit [16] や PennyLane [17]、Cirq [18] などが知られている。また、量子コンピュータ上で実行可

[1] 横浜国立大学大学院 理工学府
[2] 慶應義塾大学大学院 理工学研究科
[3] 筑波大学 情報学群 情報科学類
[4] Miletos 株式会社
[5] 株式会社 NTT データグループ
[6] 大阪大学 基礎工学研究科
[7] 大阪大学 量子情報・量子生命研究センター
[8] 理化学研究所 量子コンピュータ研究センター
[a)] yukiyoshi-kein-wk@ynu.jp
[b)] s3ta2ka2ta1@keio.jp
[c)] s2413552@u.tsukuba.ac.jp
[d)] bsdwork2425@gmail.com
[e)] Takumi.Kato@nttdata.com
[f)] mitarai.kosuke.es@osaka-u.ac.jp
[g)] fujii.keisuke.es@osaka-u.ac.jp
[h)] ishikawa-naoki-fr@ynu.ac.jp

能な命令セットを定義した量子命令セットアーキテクチャとして OpenQASM [19] や Quil [20] などが存在する。こうした量子コンピュータのためのソフトウェア開発が進展する中、このような開発を実行できる人材の需要が高まっている [21]。

以上の背景を元に、本稿では量子プログラミングコンテスト QCoder を紹介する。QCoder は従来の競技プログラミングの枠組みを量子プログラミングに導入するもので、コンテストの参加者は様々な量子アルゴリズムを実際に動くプログラムとして実装する。こうした取り組みは教育的な効果に加え、人材発掘や新たな量子アルゴリズムの開発、実装提供などの観点からも有用であると考えられる。同様の量子プログラミングコンテストには、IBM Quantum Challenge [22] や QHack Coding Challenges [23]、Microsoft Q# Coding Contest [24] などが既に存在するが、出題される問題の形式や利用可能な言語といった点で違いがある。

本稿は次のとおり構成される。2 章では量子プログラミングの概要について述べ、3 章では関連するプラットフォームを紹介する。4 章では本プラットフォームの概要を説明し、5 章でその実装について詳述する。6 章では量子プログラミングコンテストに伴う課題を提示し、7 章では実施したアンケートの結果を考察する。最後に 8 章で本稿を結論づける。

2. 量子プログラミング

一般に量子計算では古典論理回路に類似した計算モデルである量子回路が利用される[*1]。量子回路では、ゼロ状態に初期化された量子ビットに対して、順に量子ゲートを作用させていくことで、所望の量子状態を生成する。生成された量子状態は測定によって、計算基底状態に対応する一意のビット列を複素振幅に応じた確率で得ることができる。このように、量子状態の生成と測定を繰り返すことで量子アルゴリズムは実行される。図 1 に最も基本的な量子状態の一つである Greenberger–Horne–Zeilinger (GHZ) 状態 [27] を生成する量子回路を示す。GHZ 状態の量子状態は $(|000\rangle + |111\rangle)/\sqrt{2}$ で表され、この状態を測定すると、計算基底状態 $|000\rangle$ もしくは $|111\rangle$ をそれぞれ 1/2 の確率で得る。

量子アルゴリズムを構成する量子回路や測定手順、さらに付随する古典的な情報処理を記述するためには、量子プログラミング言語が利用される。代表的な量子プログラミング言語である Qiskit [16] や PennyLane [17]、Cirq [18] は、Python のパッケージとして開発されており、ユーザは Python を使って量子プログラミングを行う。量子プログラミング言語で記述された量子回路は OpenQASM [19] や Quil [20] といった量子命令セットアーキテクチャにコンパイルされ、なんらかのバックエンド上で実行される[*2]。バックエンドには実機の量子コンピュータや古典コンピュータ上で動作するシミュレータなどが含まれ[*3]、フロントエンドとしての量子プログラミング言語からは、実行基盤となるバックエンドの具体的な実装を意識する必要はない。次のコードに Qiskit を使って図 1 の量子回路を記述するプログラムを示す。

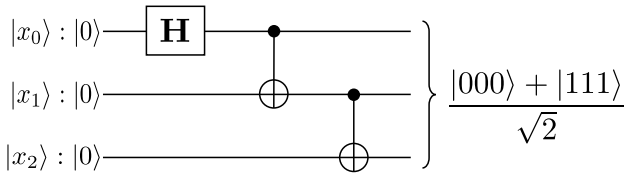

図 1 3 量子ビットの GHZ 状態を生成する量子回路

```
from qiskit import QuantumCircuit

qc = QuantumCircuit(3)
qc.h(0)
qc.cx(0, 1)
qc.cx(0, 2)
```

QCoder では、現状で最も普及している量子プログラミング言語とされる Qiskit を採用しており、コンテスト参加者は Qiskit を通じて量子アルゴリズムを記述する。Qiskit は IBM によって開発されたオープンソースソフトウェアであり、2024 年 9 月時点で 5,100 を超える GitHub スターを獲得している。

3. 関連するプラットフォーム

これまで、量子プログラミング言語を開発する企業によって、いくつかの量子プログラミングコンテストが開催されてきた。代表的なものに、IBM Quantum Challenge [22] や QHack Coding Challenges [23]、Microsoft Q# Coding Contest [24] などが挙げられる。本章では、これらのプラットフォームの特徴を概説し、QCoder との比較を行う。

3.1 IBM Quantum Challenge [22]

IBM Quantum Challenge [22] は 2019 年から IBM により開催されている量子プログラミングコンテストである。参加者は IBM の開発する Qiskit を利用して、期間内に

[*1] 量子計算において、量子回路と比較して設計や解釈が人間にとって容易な高級言語を開発しようとする試みはいくつか存在するが [25], [26]、2024 年 9 月現在、デファクトスタンダードとなる言語は存在していない。

[*2] 正確には、量子命令セットアーキテクチャへコンパイルせずに量子回路を量子プログラミング言語のまま直接実行することもある。

[*3] 古典コンピュータのバックエンドには、事実上 CPU や GPU といった半導体ベースの実装のみが利用されるが、量子コンピュータでは、超伝導方式やイオントラップ方式など、物理的実装は未だ定まっていない。

表 1　量子プログラミングコンテストの比較

名前	主催企業	アルゴリズム/最適化	個人/チーム	言語	コードの参照性
QCoder	-	アルゴリズム	個人	Qiskit	✓
IBM Quantum Challenge [22]	IBM	最適化	個人	Qiskit	-
QHack Coding Challenges [23]	Xanadu	アルゴリズム	チーム	PennyLane	-
Microsoft Q# Coding Contest [24]	Microsoft	アルゴリズム	個人	Q#	✓

提示される課題に取り組む。2022 年に開催された「IBM Quantum Challenge Fall」まで、他の参加者とスコアを競う競技的な問題が含まれていたが、2023 年以降のイベントではそのような問題は提供されていない。

3.2 QHack Coding Challenges [23]

QHack Coding Challenges [23] は Xanadu の主催するハッカソン QHack の一部として、2019 年より開催されている量子プログラミングコンテストである。参加者は複数人のチームを組み、Xanadu の開発する PennyLane を利用する。

3.3 Microsoft Q# Coding Contest [24]

Microsoft Q# Coding Contest [23] は Microsoft の主催する量子プログラミングコンテストであり、古典プログラミングコンテストプラットフォームである CodeForces [1] 上で 2018 年から 2020 年まで開催された。参加者は Microsoft の開発する *Q#* を利用する。

以上 3 つのプラットフォームと QCoder の比較を表 3 に示す。

4. プラットフォーム概要

QCoder は 2024 年 1 月にリリースされた量子プログラミングコンテストプラットフォームである。本章ではプラットフォームの概要について紹介する。

4.1 コンテスト形式

各コンテストでは制限時間内に複数の問題が出題される。従来の競技プログラミングコンテストでは、各問題は互いに独立し、文脈を共有しないことが一般的である。しかし、QCoder では、各大問が最大で 10 問程度の相互に関連した小問の集まりとして構成され、共通の文脈を共有する。小問は段階的に難易度が上昇し、参加者は前半で作成したプログラムを後半の問題で再利用することが求められる。こうすることで事前知識のない参加者でも小問の誘導にしたがっていくことで最終的に複雑な量子回路やアルゴリズムを構築できるようになる。

各問題には得点が設定されており、正答した問題の得点の合計によって順位が決定する。順位表はリアルタイムに更新され、他の参加者がどの問題に正答したかを確認できる。コンテスト終了後には、解答例や詳細な解説が公開され、問題を解くための思考プロセスを学習する機会が提供される。また、コンテスト後には他の参加者が提出したプログラムを閲覧可能になり、他者の解答を参考にすることができる。

4.2 問題形式

それぞれの問題は問題文と制約から構成され、必要に応じて入力例やヒントが記される。問題文には提出されたプログラムがどのような入力を受け取り、どのような出力を行うことが期待されるかが記述される。制約にはプログラムに課される具体的な制限が記述される。古典計算における競技プログラミングでは実行時間やメモリ使用量の制限を通じて、アルゴリズムの時間計算量や空間計算量に制約を課すことが一般的である。量子コンピュータ上での実行を想定すると、これらに代わる指標として、回路の深さや量子ビット数、ゲート数などの制約を設けることができる。

4.3 フィードバック

提出されたプログラムはサーバ上で自動的に採点され、その結果が参加者にフィードバックされる。結果はテストケースごとに判定され、主に以下の 6 つのステータスに分類される。

(1) AC (Accepted): 期待された出力を返し、正答したことを表す。
(2) WA (Wrong Answer): 誤った出力を返したことを表す。
(3) QLE (Qubits Limit Exceeded): 使用された量子ビット数が制約を超過したことを表す。
(4) DLE (Depth Limit Exceeded): 量子回路の深さが制約を超過したことを表す。
(5) TLE (Time Limit Exceeded): 実行時間が制約を超過したことを表す。
(6) MLE (Memory Limit Exceeded): メモリの使用量が制約を超過したことを表す。

4.4 問題例

QCoder ではこれまで計 27 問の問題が出題されてきた。ここでは 2024 年 8 月に開催された QCoder Programming Contest 002 より A5 問題「Generate State $\frac{1}{\sqrt{2}}(|0\rangle - |2^n - 1\rangle)$ III」を問題例として紹介する。この問題は量子ビット数 n に対して、$O(\log n)$ 程度の深さで一般化 GHZ 状態 $(|0\rangle - |2^n - 1\rangle)/\sqrt{2}$ の状態準備回路を構成

する問題である。コンテスト時間中の正答率は約 46 % で参加者 122 人のうち 56 人が正答した。具体的な問題の出題形式は以下の通りである。

問題文

整数 n が入力として与えられる。ゼロ状態から量子状態 $|\psi\rangle$ を作り出す操作を、n 量子ビットをもつ量子回路 qc 上に実装せよ。量子状態 $|\psi\rangle$ は次式で定義される。

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|\underbrace{0\cdots0}_{n}\rangle - |\underbrace{1\cdots1}_{n}\rangle) \tag{1}$$

制約

- $2 \leq n \leq 15$
- 量子回路の深さは 6 を超えてはならない。
- グローバル位相の変化は問わない。
- 提出されるコードは次のフォーマットにしたがうこと

```
from qiskit import QuantumCircuit


def solve(n: int) -> QuantumCircuit:
    qc = QuantumCircuit(n)
    # Write your code here:

    return qc
```

入力例

- $n = 4$
  実装された量子回路 qc は次式の遷移を満たす。
  $$|0000\rangle \xrightarrow{\text{qc}} \frac{1}{\sqrt{2}}(|0000\rangle - |1111\rangle) \tag{2}$$

4.5 期待される効果

量子プログラミングにおける競技プログラミングは未だ発展途上の取り組みである。その効果は体系的に整理されていないが、我々は主に以下の 4 点の効果が期待されると考える。

1 つ目は教育的効果である。コンテストではさまざまな難易度の量子アルゴリズムを実装する問題が出題される。参加者はこうした問題を解く過程を楽しみながら、量子アルゴリズムの理解を深め、実装力を高めることができる。

2 つ目はコミュニティの拡大と活発化である。参加者は問題に挑戦し、得点をもとに他の参加者と競い合う。こうした競技プログラミングのゲーム性は娯楽としての側面をもち、これまで量子情報科学に接点のなかった層を新たに取り込むことで、新たな人材の発掘や量子コミュニティの成長、それに伴う量子情報科学分野全体のプレゼンス向上が期待される。実際、7 章で紹介するアンケート結果から、参加者の多くは量子計算に馴染みのない初学者に近い状態であったことが確認された。

3 つ目は新たな量子アルゴリズムの開発である。コンテストを通じて、既存の手法と比較してより効率的な量子アルゴリズムが発見される可能性がある。例えば、4.4 節にて紹介した問題「Generate State $\frac{1}{\sqrt{2}}(|0\rangle - |2^n - 1\rangle)$ III」に対する解法は [28] にて 2019 年に提案されている。すなわち問題の解法はアカデミアにおける貢献としても有用な新規性の高い手法であるにも関わらず、コンテストでは参加者 122 人のうち 56 人がこの問題を正答した[*4]。今後、同様の問題の出題を通じてこれまでに提案されていない新たな手法の発見につながる可能性がある。

最後に 4 つ目は量子アルゴリズムの実装提供である。一般に量子アルゴリズムは論文中で擬似コードの形で提供され、実機やシミュレータ上で動作するプログラムが公開されることはほとんどない。こうした状況に対し、量子プログラミングコンテストでは、コンテストを通じて参加者が量子アルゴリズムを実際のプログラムとして記述する。このコードはインターネット上に公開され、誰でも参照することが可能になる。こうして、アルゴリズムを提案する理論家とアルゴリズムの実装を必要とするユーザの橋渡しを担うことができる可能性がある。ただし、提出されたプログラムの所有権および著作権は、プログラムの作成者であるユーザに帰属するため、その利用には許諾が必要となる。

5. 実装

自動採点システムの実装については、これまで数多くの研究が行われてきた [4], [8], [10], [29], [30], [31]。その中でも QCoder はクライアントサーバモデルに基づく標準的なクラウドアーキテクチャを採用している。図 2 にシステム全体のアーキテクチャを示す。

提出から採点の流れは以下の通りである。はじめに、コンテスト参加者は、与えられた問題に対して作成したプログラムをクライアントから提出する。プログラムは HTTP リクエストを介して、API (application programming interface) サーバに送信され、API サーバはプログラムとそのメタデータをメッセージキューに追加する。追加されたプログラムは、空いているジャッジサーバによって取り出され、プログラムがテストケースに対して実行される。実行結果はデータベースに記録され、API サーバがその結果をクライアントに返すことで、実行結果のフィードバックが行われる。

5.1 スケーラビリティとコスト効率性

コンテスト中には、1 秒間で最大数十から数百のプログラムが提出され、それぞれのプログラムが採点のために一台のサーバを数十秒の間占有する。このため、自動採点シ

[*4] コンテスト中に論文 [28] を参照し、正答した参加者が存在した可能性はあるが、ほとんどの提出の実装は論文と一致しておらず、大半の参加者は自力で解法にたどり着いたものと考えられる。

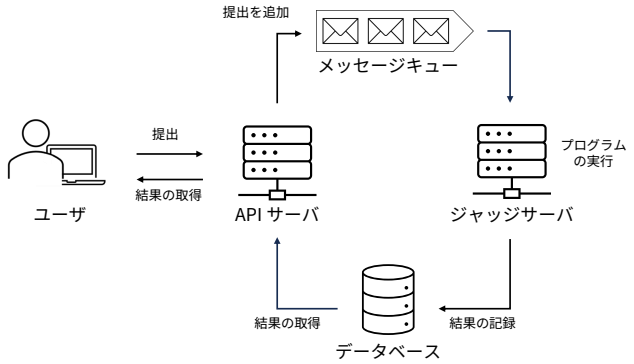

図 2 システムアーキテクチャ

ステムには非常に高い負荷がかかり、スケーラビリティが重要な課題となる。QCoder では、これらの課題に対応するために二つの対策を講じている。

一つ目は、API サーバとジャッジサーバの間にメッセージキューを導入し、採点処理を非同期化した点である。API サーバは一台で数千のリクエストを同時に処理できるが、ジャッジサーバは一台で一つのリクエストしか同時に処理できない。このため、API サーバとジャッジサーバの台数比率は最大で数千に達することもある。メッセージキューによる非同期処理の導入によって、この処理速度の差を吸収し、スケーラビリティの問題を効果的に解決している。

二つ目は、すべてのコンポーネントに対して水平スケールが可能なサーバーレスシステム [32] を採用している点である。従来の常設型システムでは、開発者が手動で必要な計算リソースの量を推定し、あらかじめ確保する必要があった。メッセージキューの導入により、プログラムが確実に採点されるようになったが、需要予測に失敗すると、キューが際限なく伸び、採点結果のフィードバックに大幅な遅延が生じる可能性がある。サーバーレスシステムでは、計算リソースが需要に応じて動的に割り当てられ、高速に制限なくスケールするため、こうした心配が不要になる。さらに、サーバーレスの利点はスケーラビリティにとどまらない。図 3 にコンテスト中に動作するジャッジサーバのインスタンス数の時間推移を示す。図から明らかなように、自動採点システムはコンテストの開催時に負荷が集中する特殊なサービスであり、常設型システムをピーク負荷に合わせて常時稼働させることは、コスト効率の面で大変非効率である。サーバーレスシステムでは、時間単位ではなく使用量に応じて課金されるため、コストの大幅な削減が可能になる。

QCoder では、API サーバとジャッジサーバには Google Cloud Run [33]、メッセージキューには Google Cloud Pub/Sub [34]、データベースには TiDB Cloud Serverless [35] を利用している。

5.2 セキュリティ

自動採点システムでは、ユーザが任意のプログラムを提出し、そのプログラムを実際に実行して採点を行う。このとき、悪意のあるユーザが攻撃を仕掛ける可能性も考慮する必要がある。たとえば、サーバ内の機密ファイルを不正に読み取って外部に送信したり、既存のファイルを改ざん・削除したりする攻撃が想定される。このような自動採点システムのセキュリティに関しては、すでに広く研究が行われており [29], [30], [31]、QCoder ではコンテナによるサンドボックス化を用いてこれらの攻撃を対処している。コンテナとはオペレーティングシステムレベルで仮想化された隔離環境を提供する技術であり、特に Docker [36] がよく知られている。コンテナはサーバーレスシステムとの相性もよく、提出されたプログラムをコンテナ内部で実行することで、たとえ悪意のあるコードが含まれていても被害はコンテナ内に限定される。また、外部へのネットワークアクセスも容易に遮断でき、さらなる安全性が確保される。

5.3 テスト

それぞれの問題では提出するプログラムの形式が指定され、提出されるプログラムには solve 関数が含まれる。多くの問題では、solve 関数は与えられた引数に対して量子回路のインスタンスを返す関数として定義される。

提出されたプログラムの正否を判定するテストの手順は以下の通りである。はじめに、各問題に対して準備されたテスト用のプログラムをジャッジサーバ上で実行する。このテストプログラムは提出されたプログラムから solve 関数をインポートした後、事前に準備された入力に対して solve 関数を実行し、返り値として量子回路のインスタンスを取得する。次にシミュレータを利用して、生成された量子回路を評価する。

量子回路を評価する手法の一つとして、量子回路をユニタリ行列で表現する方法が考えられる。具体的には、まず量子回路はユニタリ行列によって一意に定義できることから、対象となる量子回路のユニタリ行列を計算する。次に、得られた行列を期待される行列と比較し、その誤差が十分に小さい場合に提出を正答と判定する。この手法は少ないテストケース数で正確な検証を可能にするが、量子ビット数 n に対するユニタリ行列のサイズは $2^n \times 2^n$ となるため、テストプログラムのメモリ利用量が膨大になり、実行時間も長くなる。その結果、評価可能な問題サイズが制限されてしまう。この課題に対して、QCoder では、より軽量な評価手法として状態ベクトルを用いた方法を採用している。具体的には、任意の初期状態に対して量子回路を実行し、結果として得られる状態ベクトルを計算する。その後、得られた状態ベクトルを期待される状態ベクトルの値と比較するというものである。正確な検証を行うには、複数の初期状態に対して評価を行う必要があり、テストケー

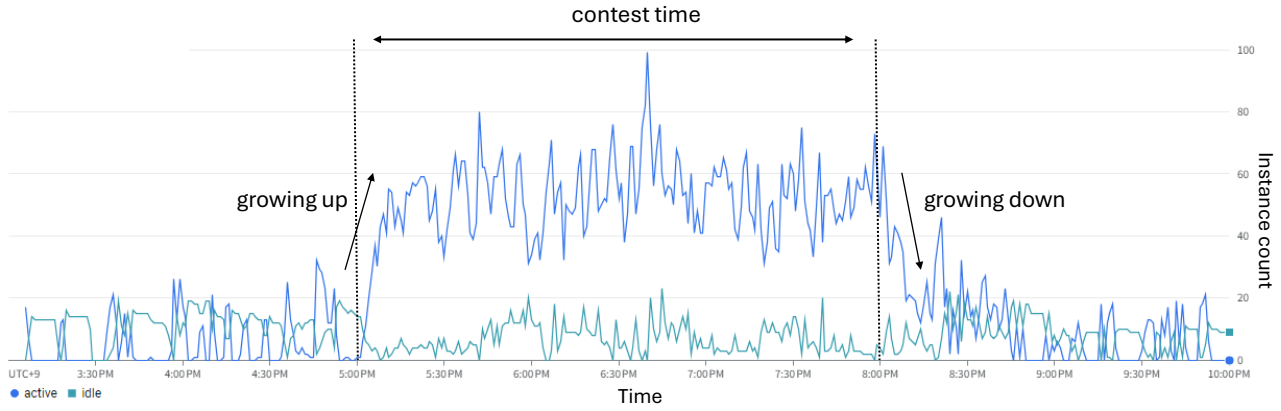

図 3 ジャッジサーバのインスタンス数の時間推移

スの数は増えてしまうものの、状態ベクトルのサイズは 2^n であることから、実行時間が短縮し、より大きな問題サイズを扱うことができる。次のコードに 4.4 節で示した問題に対するテストプログラムの例を示す。

```
import numpy as np
from qiskit.quantum_info import
    Statevector
from submission import solve

n = 2  # number of qubits
qc = solve(n)

if qc.depth() > 6:
    exit(1)  # Depth Limit Exceeded

statevector = Statevector(qc)
expected_statevector = np.array([1, 0, 0,
    -1]) / np.sqrt(2)

if np.allclose(statevector,
    expected_statevector):
    exit(0) # Accepted

exit(1) # Wrong Answer
```

6. 課題

量子プログラミングコンテストには量子プログラミングに特有の課題がいくつか存在する。本章では、それぞれの課題を紹介し、考えられる解決策について議論する。

6.1 入出力形式

従来の古典競技プログラミングでは、提出するプログラムの入出力は標準入出力を介したテキスト形式が一般的である。しかし、現在の量子プログラミングコンテストでは、こうしたテキストベースの入出力形式は不正防止の観点から困難である。例として、整数 N の素因数分解をする問題を考える。量子プログラミングコンテストではショアのアルゴリズムが解法として考えられるが、現在の量子コンピュータやシミュレータの性能では、ショアのアルゴリズムで大きな整数を扱うことはできない [37]。よって、テキストベースの入出力を採用した場合、ショアのアルゴリズムではなく、量子計算を介さない古典アルゴリズムによって結果を計算するプログラムを提出できる。しかし、これでは単なる古典プログラミングコンテストである。こうした出題意図に反する解答を防ぐために、QCoder では提出に含まれる solve 関数の引数と返り値を通じて入出力を行う形式を採用している。例えば、ショアのアルゴリズムで利用する量子回路のインスタンスを出力（返り値）として指定することで、不正な解答は事実上排除できる。しかし、こうした関数ベースの提出形式を採用する場合、各問題ごとにその関数を呼び出してテストするテストプログラムを別途準備する必要が生じる。さらに、このテストプログラムは言語ごとに用意する必要があり、多言語対応が難しくなるという課題もある。将来的にハードウェアの進化によって古典アルゴリズムでは解けない規模の問題を量子コンピュータが解けるようになれば、こうした不正は原理的に不可能になり、入出力形式に関する課題は解決されるだろう。

他の出力形式としては、OpenQASM [19] や Quil [20] などの量子命令セットアーキテクチャが考えられる。すなわち、参加者は自分の好みの量子プログラミング言語を使い、実装した量子回路を量子命令セットアーキテクチャにコンパイルしてテキストとして出力するような形式が考えられる。現状、量子命令セットアーキテクチャの表現力やそのコンパイラとパーサの実装に量子プログラミング言語ごとの差があることなどの課題は存在するが、将来的には検討の価値がある。

6.2 不安定なソフトウェア

古典的なプログラミング言語が約半世紀の歴史を持つ一

方で、量子プログラミング言語やその関連ソフトウェアの多くは、この 10 年ほどで開発が始まった。そのため、これらのソフトウェアやインターフェースは未だ成熟しておらず、後方互換性を伴わない大規模な変更が頻繁に行われる [16]。このため、コンテストで実装されたプログラムはソフトウェアのバージョン間での互換性が保証されず、ユーザはアップデートのたびに変更を追う必要がある。こうしたソフトウェアの不安定性は時間とともに改善されることが期待されるものの、プログラムの寿命やユーザの学習負荷といった観点から依然として重要な課題である。

6.3 人工知能の利用

近年、生成言語モデルをはじめとする人工知能の競技プログラミングへの適用に関する研究が盛んに進められている [38], [39], [40]。現状では、人工知能の成績は大半の人間を上回っていないとされているが [40]、その性能向上は急速に進んでいる。参加者が人工知能を利用してコンテストに参加する場合、人工知能以下のスキルレベルにおける競技性は失われてしまう。競技ルールにおいて人工知能の利用を禁止することも可能であるが、インターネット上でコンテストを実施する場合には、その厳密な運用は困難である。ただし、量子プログラミングに関しては、古典プログラミングと比較してプログラムの流通が少ないため、学習に必要なデータが不足し、人工知能の性能が低下する可能性がある。また、仮に人工知能の性能が大多数の人間を超えたとしても、競技目的ではなく学習目的の自動採点システムとしての利用は引き続き有用である。

7. アンケート結果

2024 年 8 月に開催された QCoder Programming Contest 002 の終了後、任意参加のアンケートを実施し、参加者 122 人のうち 45 人から回答を得た。アンケート結果を表 7 に示す。各設問に対する回答の百分率は整数に丸めてあり、設問 3 から設問 5 までは、5 段階評価のリッカート尺度 [41] を使用した。リッカート尺度では、選択肢 5 が「強い同意」を、選択肢 1 は「強い不同意」を表す。

7.1 参加者の背景

設問 1「現在の状況を教えてください」に対する回答では、48 % が社会人、31 % が大学院生、11 % が大学生、4 % が高校生、4 % が小学生または中学生という結果が得られた。一般に競技プログラミングコンテストでは学生の参加が多く社会人の参加は少ないとされる。そのため今回の、年齢が高いほど参加率が高く、社会人の割合が約半数に達する結果は意外であった。この傾向は、産業界および学術界における量子コンピューティング技術への期待と関心の高まりを反映していると考えられる。また同時に、大学生や中高生を対象とした量子情報科学の普及活動を一層促進する必要性が浮き彫りとなった。

設問 2「量子情報科学分野との関係性を教えてください。」では、2 % が関連する職に就いている（いた）、24 % が研究活動に従事している (いた) 、33 % が大学の講義や自分で学習したことがある、33 % があまり馴染みがない、7 % が馴染みがないと回答した。全体の 60 % が少なくとも量子情報科学の学習経験がある一方で、40% が学習経験がないことから、参加者が多様な学習経験に分散していることがわかる。

7.2 難易度

設問 3「コンテストの難易度について教えてください。」に対する回答の平均値は 3.7、標準偏差は 0.6 であった。コンテストは初学者にも取り組める問題を含むが、難易度が「易しい」と答えた参加者はいなかった。今後は参加者が自身のレベルに応じた問題に取り組めるよう、より適切な難易度調整が課題となる。

7.3 評価と意欲

設問 4「コンテストの楽しさについて教えてください。」に対する回答の平均値は 4.6、標準偏差は 0.53 と非常に高い結果とが得られた。設問 5「次回のコンテストにも参加したいですか。」に対する回答の平均値は 4.9、標準偏差は 0.31 であり、こちらも非常に高い結果となった。どちらの設問も非常に高い値が得られており、参加者がコンテストを大いに楽しんだことが確認できる。この結果から、競技プログラミングのゲーム性が参加者の学習意欲を高める効果は、従来の古典的な競技プログラミングと同様に、量子プログラミングにおいても有効であると考えられる。

8. おわりに

本稿では、量子プログラミングコンテスト QCoder の概要や実装、課題について紹介した。量子プログラミングコンテストの実装は古典プログラミングコンテストと多くの点で類似する一方で、問題の出題形式やテスト方法などに違いがある。アンケート結果からは、量子プログラミングにおいても競技プログラミングのゲーム性が参加者の学習意欲を高める効果がある可能性が示された。現在のプラットフォームではシミュレータによる提出の評価を前提としているが、今後は量子コンピュータの実機を使った評価の実施も期待される。

謝辞

本研究の遂行にあたっては、未踏ターゲット事業ならびに Unitary Fund の支援を受けた。

表 2 アンケート調査の結果 ($n = 45$)

質問	回答	回答数	回答の百分率
設問 1: 現在の状況を教えて下さい。	社会人	22	49 %
	大学院生	14	31 %
	大学生	5	11 %
	高校生	2	4 %
	小・中学生	2	4 %
設問 2: 量子情報科学分野との関係性を教えてください。	関連する職に就いている (いた)	1	2 %
	研究活動に従事している (いた)	11	24 %
	大学の講義や自分で学習したことがある	15	33 %
	あまり馴染みがない	15	33 %
	馴染みがない	3	7 %
設問 3: コンテストの難易度について教えてください。	5: とても難しい	4	9 %
	4: 難しい	25	56 %
	3: どちらともいえない	16	36 %
	2: 易しい	0	0 %
	1: とても易しい	0	0 %
設問 4: コンテストの楽しさについて教えてください。	5: とても楽しい	29	64 %
	4: 楽しい	15	33 %
	3: どちらともいえない	1	2 %
	2: 楽しくない	0	0 %
	1: まったく楽しくない	0	0 %
設問 5: 次回のコンテストにも参加したいですか。	5: ぜひ参加したい	40	89 %
	4: 参加したい	5	11 %
	3: どちらともいえない	0	0 %
	2: 参加したくない	0	0 %
	1: まったく参加したくない	0	0 %

参考文献

[1] Codeforces (online), available from ⟨https://codeforces.com/⟩.
[2] Meta Hacker Cup (online), available from ⟨https://www.facebook.com/codingcompetitions/hacker-cup⟩.
[3] Dagiene, V. and Skupiene, J.: Learning by Competitions: Olympiads in Informatics as a Tool for Training High-Grade Skills in Programming, *2nd International Conference Information Technology: Research and Education ITRE 2004*, pp. 79–83 (2004).
[4] Kurnia, A., Lim, A. and Cheang, B.: Online Judge, *Computers & Education*, Vol. 36, No. 4, pp. 299–315 (2001).
[5] Law, K. M. Y., Lee, V. C. S. and Yu, Y. T.: Learning Motivation in E-Learning Facilitated Computer Programming Courses, *Computers & Education*, Vol. 55, No. 1, pp. 218–228 (2010).
[6] Wasik, S., Antczak, M., Badura, J., Laskowski, A. and Sternal, T.: A Survey on Online Judge Systems and Their Applications, *ACM Computing Surveys*, Vol. 51, No. 1, pp. 3:1–3:34 (2018).
[7] Paiva, J. C., Leal, J. P. and Figueira, Á.: Automated Assessment in Computer Science Education: A State-of-the-Art Review, *ACM Transactions on Computing Education*, Vol. 22, No. 3, pp. 34:1–34:40 (2022).
[8] Lawrence, R.: Teaching Data Structures Using Competitive Games, *IEEE Transactions on Education*, Vol. 47, No. 4, pp. 459–466 (2004).
[9] Regueras, L. M., Verdu, E., Munoz, M. F., Perez, M. A., de Castro, J. P. and Verdu, M. J.: Effects of Competitive E-Learning Tools on Higher Education Students: A Case Study, *IEEE Transactions on Education*, Vol. 52, No. 2, pp. 279–285 (2009).
[10] Verdú, E., Regueras, L. M., Verdú, M. J., Leal, J. P., de Castro, J. P. and Queirós, R.: A Distributed System for Learning Programming On-Line, *Computers & Education*, Vol. 58, No. 1, pp. 1–10 (2012).
[11] Yuen, K. K. F., Liu, D. Y. W. and Leong, H. V.: Competitive Programming in Computational Thinking and Problem Solving Education, *Computer Applications in Engineering Education*, Vol. 31, No. 4, pp. 850–866 (2023).
[12] Moore, G. E.: Cramming More Components onto Integrated Circuits, *Electronics*, Vol. 38, No. 8 (1965).
[13] Hennessy, J. L. and Patterson, D. A.: *Computer Architecture, Sixth Edition: A Quantitative Approach*, sixth edition (2017).
[14] Shor, P. W.: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, *SIAM Journal on Computing*, Vol. 26, No. 5, pp. 1484–1509 (1997).
[15] Grover, L. K.: A Fast Quantum Mechanical Algorithm for Database Search, *Proceedings of ACM Symposium on Theory of Computing*, pp. 212–219 (1996).
[16] Javadi-Abhari, A., Treinish, M., Krsulich, K., Wood, C. J., Lishman, J., Gacon, J., Martiel, S., Nation, P. D., Bishop, L. S., Cross, A. W., Johnson, B. R. and Gambetta, J. M.: Quantum Computing with Qiskit, *arXiv:2405.08810* (2024).
[17] Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Ahmed, S., Ajith, V., Alam, M. S., Alonso-Linaje, G., Akash-Narayanan, B., Asadi, A., Arrazola, J. M., Azad, U.,

Banning, S., Blank, C., Bromley, T. R., Cordier, B. A., Ceroni, J., Delgado, A., Di Matteo, O., Dusko, A., Garg, T., Guala, D., Hayes, A., Hill, R., Ijaz, A., Isacsson, T., Ittah, D., Jahangiri, S., Jain, P., Jiang, E., Khandelwal, A., Kottmann, K., Lang, R. A., Lee, C., Loke, T., Lowe, A., McKiernan, K., Meyer, J. J., Montañez-Barrera, J. A., Moyard, R., Niu, Z., O'Riordan, L. J., Oud, S., Panigrahi, A., Park, C.-Y., Polatajko, D., Quesada, N., Roberts, C., Sá, N., Schoch, I., Shi, B., Shu, S., Sim, S., Singh, A., Strandberg, I., Soni, J., Száva, A., Thabet, S., Vargas-Hernández, R. A., Vincent, T., Vitucci, N., Weber, M., Wierichs, D., Wiersema, R., Willmann, M., Wong, V., Zhang, S. and Killoran, N.: PennyLane: Automatic Differentiation of Hybrid Quantum-Classical Computations, *arXiv:1811.04968* (2018).

[18] Isakov, S. V., Kafri, D., Martin, O., Heidweiller, C. V., Mruczkiewicz, W., Harrigan, M. P., Rubin, N. C., Thomson, R., Broughton, M., Kissell, K., Peters, E., Gustafson, E., Li, A. C. Y., Lamm, H., Perdue, G., Ho, A. K., Strain, D. and Boixo, S.: Simulations of Quantum Circuits with Approximate Noise Using Qsim and Cirq, *arXiv:2111.02396* (2021).

[19] Cross, A., Javadi-Abhari, A., Alexander, T., De Beaudrap, N., Bishop, L. S., Heidel, S., Ryan, C. A., Sivarajah, P., Smolin, J., Gambetta, J. M. and Johnson, B. R.: OpenQASM 3: A Broader and Deeper Quantum Assembly Language, *ACM Transactions on Quantum Computing*, Vol. 3, No. 3, pp. 12:1–12:50 (2022).

[20] Smith, R. S., Curtis, M. J. and Zeng, W. J.: A Practical Quantum Instruction Set Architecture, *arXiv:1608.03355* (2017).

[21] Akbar, M. A., Khan, A. A., Mahmood, S. and Rafi, S.: Quantum Software Engineering: A New Genre of Computing, *Proceedings of the 1st ACM International Workshop on Quantum Software Engineering: The Next Evolution*, pp. 1–6 (2024).

[22] IBM Quantum Challenge (online), available from ⟨https://research.ibm.com/projects/quantum-challenges/⟩.

[23] QHack Coding Challenge (online), available from ⟨https://qhack.ai/⟩.

[24] Microsoft Q# Coding Contest (online), available from ⟨https://codeforces.com/msqs2020⟩.

[25] Bichsel, B., Baader, M., Gehr, T. and Vechev, M.: Silq: A High-Level Quantum Language with Safe Uncomputation and Intuitive Semantics, *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 286–300 (2020).

[26] Seidel, R., Bock, S., Zander, R., Petrič, M., Steinmann, N., Tcholtchev, N. and Hauswirth, M.: Qrisp: A Framework for Compilable High-Level Programming of Gate-Based Quantum Computers, *arXiv:2406.14792* (2024).

[27] Greenberger, D. M., Horne, M. A. and Zeilinger, A.: Going beyond Bell's Theorem, *Bell's Theorem, Quantum Theory and Conceptions of the Universe*, pp. 69–72 (1989).

[28] Cruz, D., Fournier, R., Gremion, F., Jeannerot, A., Komagata, K., Tosic, T., Thiesbrummel, J., Chan, C. L., Macris, N., Dupertuis, M.-A. and Javerzac-Galy, C.: Efficient Quantum Algorithms for GHZ and W States, and Implementation on the IBM Quantum Computer, *Advanced Quantum Technologies*, Vol. 2, No. 5-6, p. 1900015 (2019).

[29] Imajo, K.: Contest Environment Using Wireless Networks: A Case Study from Japan, *Olympiads in Informatics*, Vol. 5, pp. 26–31 (2011).

[30] Petit, J., Giménez, O. and Roura, S.: Jutge.Org: An Educational Programming Judge, *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, pp. 445–450 (2012).

[31] Peveler, M., Maicus, E. and Cutler, B.: Comparing Jailed Sandboxes vs Containers within an Autograding System, *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pp. 139–145 (2019).

[32] Shafiei, H., Khonsari, A. and Mousavi, P.: Serverless Computing: A Survey of Opportunities, Challenges, and Applications, *ACM Computing Surveys*, Vol. 54, No. 11s, pp. 239:1–239:32 (2022).

[33] Cloud Run (online), available from ⟨https://cloud.google.com/run⟩.

[34] Cloud Pub/Sub (online), available from ⟨https://cloud.google.com/pubsub⟩.

[35] Huang, D., Liu, Q., Cui, Q., Fang, Z., Ma, X., Xu, F., Shen, L., Tang, L., Zhou, Y., Huang, M., Wei, W., Liu, C., Zhang, J., Li, J., Wu, X., Song, L., Sun, R., Yu, S., Zhao, L., Cameron, N., Pei, L. and Tang, X.: TiDB: A Raft-Based HTAP Database, *Proc. VLDB Endow.*, Vol. 13, No. 12, pp. 3072–3084 (2020).

[36] Merkel, D.: Docker: Lightweight Linux Containers for Consistent Development and Deployment, *Linux J.*, Vol. 2014, No. 239, p. 2:2 (2014).

[37] Skosana, U. and Tame, M.: Demonstration of Shor's Factoring Algorithm for N=21 on IBM Quantum Processors, *Scientific Reports*, Vol. 11, No. 1, p. 16599 (2021).

[38] Alexander, F., Abdiwijaya, E. A., Pherry, F., Gunawan, A. A. S. and Anderies: Systematic Literature Review on Solving Competitive Programming Problem with Artificial Intelligence (AI), *2022 1st International Conference on Software Engineering and Information Technology (ICoSEIT)*, pp. 85–90 (2022).

[39] Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Dal Lago, A., Hubert, T., Choy, P., de Masson d'Autume, C., Babuschkin, I., Chen, X., Huang, P.-S., Welbl, J., Gowal, S., Cherepanov, A., Molloy, J., Mankowitz, D. J., Sutherland Robson, E., Kohli, P., de Freitas, N., Kavukcuoglu, K. and Vinyals, O.: Competition-Level Code Generation with AlphaCode, *Science*, Vol. 378, No. 6624, pp. 1092–1097 (2022).

[40] Koubaa, A., Qureshi, B., Ammar, A., Khan, Z., Boulila, W. and Ghouti, L.: Humans Are Still Better than ChatGPT: Case of the Ieeextreme Competition, *Heliyon*, Vol. 9, No. 11 (2023).

[41] Likert, R.: A Technique for the Measurement of Attitudes, *Archives of Psychology*, Vol. 22 140, pp. 55–55 (1932).