Received December 22, 2021; revised March 8, 2022; accepted March 14, 2022; date of publication March 16, 2022; date of current version April 26, 2022.

Digital Object Identifier 10.1109/TQE.2022.3160015

Topological-Graph Dependencies and Scaling Properties of a Heuristic Qubit-Assignment Algorithm

MATTHEW A. STEINBERG[1,2,3], SEBASTIAN FELD[1,2] (Member, IEEE), CARMEN G. ALMUDEVER[1,4], MICHAEL MARTHALER[3], AND JAN-MICHAEL REINER[3]

[1]QuTech, Delft University of Technology, 2628 CD Delft, The Netherlands
[2]Quantum and Computer Engineering Department, Delft University of Technology, 2628CD Delft, The Netherlands
[3]HQS Quantum Simulations GmbH, D-76131 Karlsruhe, Germany
[4]Computer Engineering Department, Technical University of Valencia, 46022 Valencia, Spain

Corresponding author: Matthew A. Steinberg (e-mail: matt.steinberg3@gmail.com).

ABSTRACT The qubit-mapping problem aims to assign and route qubits of a quantum circuit onto an noisy intermediate-scale quantum (NISQ) device in an optimized fashion, with respect to some cost function. Finding an optimal solution to this problem is known to scale exponentially in computational complexity; as such, it is imperative to investigate scalable qubit-mapping solutions for NISQ computation. In this work, a noise-aware heuristic qubit-assignment algorithm (which assigns initial placements for qubits in a quantum algorithm to qubits on an NISQ device, but does not route qubits during the quantum algorithm's execution) is presented and compared against the optimal *brute-force* solution, as well as a trivial qubit assignment, with the aim to quantify the performance of our heuristic qubit-assignment algorithm. We find that for small, connected-graph algorithms, our heuristic-assignment algorithm faithfully lies in between the effective upper and lower bounds given by the brute-force and trivial qubit-assignment algorithms. Additionally, we find that the topological-graph properties of quantum algorithms with over six qubits play an important role in our heuristic qubit-assignment algorithm's performance on NISQ devices. Finally, we investigate the scaling properties of our heuristic algorithm for quantum processors with up to 100 qubits; here, the algorithm was found to be scalable for quantum-algorithms that admit path-like graphs. Our findings show that as the size of the quantum processor in our simulation grows, so do the benefits from utilizing the heuristic qubit-assignment algorithm, under particular constraints for our heuristic algorithm. This work, thus, characterizes the performance of a heuristic qubit-assignment algorithm with respect to the topological-graph and scaling properties of a quantum algorithm that one may wish to run on a given NISQ device.

INDEX TERMS Quantum computing, qubit-mapping problem.

I. INTRODUCTION

Quantum computation may still be in its infancy, but new advances have allowed for the first experimental demonstrations of quantum computing in recent years [1]–[5]. Quantum computers themselves promise to aid in solving classically intractable problems for such fields as quantum chemistry [6]–[9], quantum machine learning [10], [11], quantum field theory [12]–[15], and quantum cryptography [16], [17], among others. However, such promise comes with a catch: protecting the quantum states in a quantum computer from deleterious noise channels has proven to be a most difficult task, preventing scalability and implementation of most quantum algorithms [18]. Indeed, current prototypes of *quantum processing units* (or QPUs) available from Rigetti, Honeywell, IBM, Google, Intel [1], [3], [5], [19]–[21], and others are considered still too resource-constrained to be able to demonstrate full fault-tolerant quantum computation [22]–[24].

In the *noisy intermediate-scale quantum* (NISQ) era, quantum computers are still rapidly evolving but still present several limitations. First, quantum computing as a field has not yet settled on a particular physical realization for quantum hardware [25]; leading candidate implementations include those constructed from superconducting qubits [19], [26]–[30], trapped-ion qubits [31]–[34], as well as other proposals [35]–[39]. Second, many devices exhibit fixed and finite connectivity constraints between neighboring qubits (a notable exception to this is trapped-ion technology, in which one can *in principle* produce “all-to-all” connectivity [31]). Third, noise considerations have severely hampered developments in quantum devices [40]. As such, efficient methods for executing quantum algorithms on first-generation quantum hardware require special attention.

In light of these difficulties, the goal of efficiently delegating these finite resources in a QPU for usage with near-term quantum algorithms is both exigent and pervasive. Quantum algorithms described as quantum circuits have to be adapted to specific hardware constraints in order to be executed. Many different approaches exist for realizing this aim, which range widely from algorithm compilation [25] and neural-network-based approaches [41], [42], to more theoretically motivated methods such as quantum gate-synthesis techniques adapted for quantum hardware [43]–[46].

Recent work has centered on the *qubit-mapping problem*, which aims to answer the following question. Given a resource-constrained quantum device and a prospective quantum algorithm, what is the optimal strategy for assigning (known as the *assignment of qubits*) and coordinating movements (known as *quantum-state routing*) of qubits along the lattice of the QPU, while making guarantees on the algorithm’s fidelity? Many methods of solving this problem have been studied, wherein the most-promising approaches so far have explicitly taken into account error information from the quantum device (e.g., two-qubit and single-qubit gate fidelities) [41], [42], [47]–[52]. In spite of this progress, much remains to be done in order to understand the general characteristics of such qubit-mapping techniques. Due to the computational hardness of finding optimized solutions for the qubit-mapping problem, one commonly resorts to heuristic algorithms [48], [50]. Therefore, assessing the capabilities of heuristic qubit-mapping algorithms is essential for addressing scalability concerns in NISQ hardware. As the assignment of qubits is a fundamental part of the qubit-mapping process [50], [53], this work will focus on this first step using a heuristic qubit-assignment algorithm (HQAA). For a thorough review of the qubit-mapping problem, we refer the reader to [54] and [55].

Currently, heuristic qubit-assignment algorithms lack a systematic comparative basis by which their performance can be assessed; this includes the notion of providing upper and lower bounds for the performance of a quantum algorithm executed on an NISQ device, evaluating quantum circuits whose entangling two-qubit gates produce interaction graphs with different topological graph-theoretic structures, and understanding whether or not a given heuristic qubit-assignment algorithm will scale well as both the quantum algorithm and the QPU size are increased.

The purpose of this manuscript is to answer the following questions: 1) how does an HQAA compare relative to an optimal brute-force and a trivial approach to the qubit-assignment problem? 2) do the topological features of a quantum algorithm influence the measured success rate for a noise-aware HQAA? 3) what behavior can one expect from a noise-aware HQAA as the size of the QPU is scaled up? To this end, we develop an HQAA which is similar to the noise-aware one introduced in [50], but improve upon this work by incorporating notions of *graph centrality* [56], [57]. We run tests with several realistic benchmarks in order to provide approximate upper and lower bounds to the success rate of the HQAA using implementations of brute-force and trivial assignment algorithms (BFAA and TAA, respectively). We systematically analyze an HQAA with respect to topologically inequivalent graph representations of quantum circuits, and assign them to a $n \times n$ QPU lattice which we keep constant throughout most of the study. An analysis of topologically inequivalent quantum-circuit structures is provided, and it is shown that the topological characteristics of two quantum algorithms (with identical numbers of gates) indeed play a role in the average success rate measured for our HQAA when quantum algorithms consisting of more than six qubits are examined. Finally, we investigate the scaling properties of the HQAA for quantum algorithms whose gate structure gives rise to path-like graph representations (which commonly appear in simulations of fermionic quantum systems [6]–[8]); we find that our HQAA consistently exhibits higher success rates than a TAA, when considering path-like quantum algorithms on larger quantum-device architectures, as long less than 75% of the quantum processor is filled. We additionally find that the benefits of utilizing our HQAA increase with the size of the QPU.

The structure of this article is as follows. Section II provides a background to the qubit-mapping problem and a walk-through of a basic example. Section III provides details on the structure of each of the algorithms employed; additionally, we show how to calculate the success rate for our simulations. We separate our results in Section IV into several parts. We describe the benchmarks utilized in our analysis in Section IV-A. Next, we discuss the results obtained from incorporating nonnearest-neighbor two-qubit gates in the benchmarks we tested (see Section IV-B), using and comparing a *breadth versus depth* analysis of two-qubit gate additions in order to generate topologically inequivalent quantum-algorithmic graphs; and in Section IV-C, we examine the outcomes of our simulations with quantum-algorithmic path-like graph structures, scaled onto QPU connectivity graphs with dimensions $n \times n$ qubits, where $n > 3$. Finally, Section V concludes this article.

II. BACKGROUND

Most quantum circuits that are devised theoretically do not take into account the actual physical hardware constraints of a given quantum device. In order to accommodate NISQ hardware, quantum-programming frameworks such as Qiskit [58] include supports that allow developers to write algorithms without explicitly factoring in hardware limitations. As such, quantum compilers must perform several steps in order to prepare the quantum algorithm for actual execution on a device. Broadly speaking, these steps are: 1) to

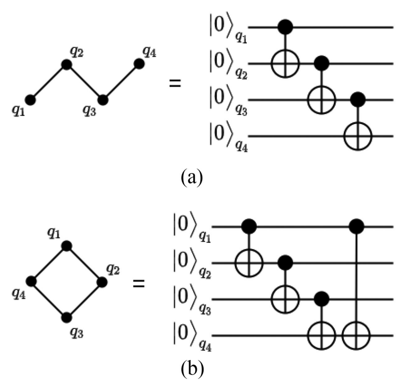

FIGURE 1. **In this work, we introduce the notions of IGs which are *path-like* and *cycle-like*. (a) Nearest-neighbor quantum circuit with a path-like IG. (b) This shows that adding a cyclic edge to the IG is equivalent to the addition of an extra two-qubit gate, which is not nearest-neighbor in the corresponding quantum circuit.**

decompose the quantum gates into *elementary gates*; 2) to assign the qubits of the quantum circuit to the physical qubits of the quantum processor, and inserting SWAP operations (known as *routing*) into the algorithm in order to satisfy the connectivity constraints of a given quantum device; 3) and finally, to optimize the resultant quantum circuit, with an aim to minimize quantities such as the execution time and gate count, among other cost functions. The qubit-mapping problem consists of the second step of quantum compilation and will be the main focus of this article, with an emphasis on the *assignment of the qubits* of a quantum algorithm to the physical qubits of a quantum processor.

In the context of the qubit-mapping problem, we consider two objects: an *interaction graph* (IG), which is a graph representation of the quantum circuit that we would like to execute (wherein vertices represent qubits, and its weights represent the number of single-qubit gates invoked; conversely, the edges correspond to two-qubit gate interaction, with weights designating the quantity of such interactions), and the *coupling graph* (CG), which is a graph representation of the QPU's geometric connectivity (wherein vertices represent the physical qubits used in the device, edges represent the two-qubit gate interactions which are possible, and weights on vertices and edges represent the single and two-qubit gate errors). For the purposes of this manuscript, we define a graph to be an ordered pair $G(V, E)$ of two sets known as the *vertex set* V and *edge set* E. The sets of vertices V and edges E are interpreted *qubits* and *two-qubit gate interactions*, respectively; this relationship is displayed in Fig. 1. Two graphs are termed topologically equivalent if there is a map $f : X \mapsto Y$ between two graphs X and Y such that the following conditions are upheld [59].

1) The map f is bijective, i.e., f maps from all edges to all edges and from all vertices to all vertices.
2) f is continuous, i.e., f is an isomorphism from X to Y, allowing for the graph operations of *smoothing out* and *subdivision* of edges.
3) The inverse function f^{-1} is continuous.

These criteria do not form the centerpiece of the present work; rather, the notion of topologically equivalent graphs will be useful for understanding the rest of the article.

Most realistic QPU layouts are accompanied with *noise-calibration statistics* which are added to the graph; these data usually include two-qubit gate error rates, single-qubit error rates, execution times (gate length), relaxation energies, and decoherence characteristic times $T1$ and $T2$ [48]. The goal is to match the geometric connectivity of the IG to that of the CG as closely as possible (in effect defining a *graph isomorphism* in the case of an exact match [59]) while taking into account the noise-calibration statistics of the quantum device as well. In the present work, we do not directly utilize the noise-calibration statistics from a real quantum computer; instead, we have analyzed the statistics from several of the IBM quantum computers [3], [50], [60], and assume random errors on the same order of magnitude (which are typically $\sim 10^{-3}$ for single-qubit errors, and $\sim 10^{-2}$ for two-qubit gate errors and measurement errors [50]), which are then randomly assigned to nodes and edges on the QPU CG. These errors are utilized in a cost function for evaluating the success rate of the quantum algorithm, using our HQAA. The procedure for this analysis is discussed in detail in Section III. Additionally, several proposals show that qubits can be given *initial assignments*, which later may be modified in timestep fashion as the execution of the algorithm progresses. As the basis for this work considers only the *initial assignment* for the qubit-mapping problem, we refer the reader to [49], [51], [52], and [61] for work involving *routing/time-scheduling techniques*.

In order to illustrate how the qubit-assignment problem can be treated, consider the quantum algorithm in Fig. 2(a). Before assigning qubits from the quantum algorithm to the physical QPU, the corresponding circuit is itself decomposed into a graph-theoretic form which we defined as the *interaction graph*; such a decomposition is needed in order to properly assign virtual qubits to a relevant portion of the QPU lattice such that the geometric constraints of the circuit are respected. As shown in Fig. 2(b), the resulting IG is the *complete graph* K_4, and cannot be exactly embedded into the QPU CG shown in Fig. 2(d); equivalently, one may say that no structure-preserving map (i.e., a *graph isomorphism*) f exists from the IG to the CG [59]. In order to correctly assign this algorithm, one may add a SWAP gate operation to the qubits q_1 and q_3, and then perform the required two-qubit gate between q_3 and q_4, as depicted in the modified algorithm of Fig. 2(c); other SWAP gates are added as well in Fig. 2. The SWAP gate itself degrades the final-state fidelity of the algorithm, in accordance with the commensurate two-qubit gate error rates. Due to the disadvantages of utilizing SWAP gates, many qubit-mapping algorithms explicitly attempt to minimize the amount of SWAP gates employed [49], [54]. However, we did not explicitly design our algorithm with such a notion in mind, even though necessary SWAP gates are considered.

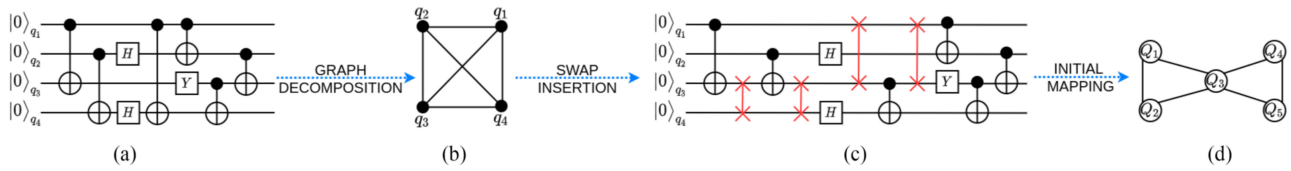

FIGURE 2. **(a) Example quantum circuit. As shown in (b), this circuit is decomposed into a graph-theoretic version of the algorithm [also known as an *interaction graph* (IG)] which illustrates the interaction via two-qubit gates of qubits in the original algorithm; edges represent two-qubit gates, while vertices represent qubits which are acted upon. Weights are added to vertices and edges in order to account for more gate invocations. In (c) the geometric connectivity of the IG is analyzed and SWAP gates are added for any interaction-graph edges which cannot be exactly assigned to the QPU coupling graph (CG); for example, if we define an assignment $\{q_1 \mapsto Q_1, q_2 \mapsto Q_2, q_3 \mapsto Q_3\}$, then the interaction-graph edge between q_1 and q_4 cannot be explicitly assigned, since Q_1 and Q_4 do not share an edge connection in the QPU CG shown in (d). The modified quantum circuit is then assigned to the QPU in (d); in this way, the appropriate vertices, in accordance with some metric to be defined, are assigned to the graph-theoretic object representing the quantum device (referred to as the CG of the QPU). In many types of qubit-mapping algorithms, qubits can then be arranged and mapped *temporally*, as well as *spatially* during the *routing process* [48], [49], [52]; for this reason, the final arrow between (c) and (d) carries the designation *initial assignment*.**

Solutions to both the qubit-assignment and qubit-mapping problems can be separated into two broad categories: 1) *optimal* (or brute-force) optimization and 2) heuristic optimization algorithms [54]. As described before, current studies of qubit-mapping and assignment focus on minimizing the number of SWAP gates [49], [54], [55], [61], [62]. However, our particular solution to the qubit-assignment problem is not the main focus of the present manuscript; rather we concentrate on the evaluation of topologically inequivalent IGs for an HQAA, building and expanding on the work pioneered by [50], with an aim toward understanding the topological-graph properties of IGs and how they influence the performance of an HQAA while keeping the QPU CG *effectively constant*.

For the present work, we introduce the concepts of IGs which are *path-like* and *cycle-like* in the context of the qubit-assignment problem in order to efficiently assess the topological-graph structure of a quantum algorithm. Two examples of this idea can be seen in Fig. 1. In Fig. 1(a), we take a four-vertex IG with three edges (known as a *Hamiltonian path* P_4) to be equivalent to a four-qubit quantum circuit exhibiting nearest-neighbor two-qubit gates. As is shown in Fig. 1(b), if an extra edge is added, the IG becomes a *cycle graph* C_4 and corresponds to the addition of an extra two-qubit gate between the first and last qubits in the quantum circuit. In graph theory, the two objects in Fig. 1 are well known; the problem of identifying suitable *Hamiltonian-path* and *cycle-graph* solutions in a simple undirected graph is related to the *traveling salesman problem* and is known to be NP-complete [63]–[65]; such computational complexity necessitates our present study. Later, we shall further seek to specify interaction-graph structure by introducing edges to a path-like IG either in a *breadth-first* or *depth-first* approach, resulting in several pairing of IGs which have the same number of edges, but are *topologically inequivalent*, cycle-like IGs.[1]

In this article, we shall focus primarily on the two differing cases described above: quantum algorithms whose IGs admit path-like and cycle-like forms (such IGs are briefly discussed in Section IV-A). The emphasis on path-like IGs is justified for two main reasons. First, in fields such as quantum chemistry, the simulation of fermionic quantum systems can be carried out by encoding the qubits via a *Jordan-Wigner transformation* [6]–[8]; such an encoding scheme can give rise to circuits known as *linear SWAP networks* [66], [67], exhibiting path-like IGs. Second, a quantum algorithm with a path-like interacting-graph representation is, in a sense, hardware agnostic; such algorithms exhibit only nearest-neighbor two-qubit gate invocations and are thus adaptable to any architecture. Designing a qubit-assignment algorithm for the goal of considering such quantum algorithms is therefore paramount. Conversely, our precise motivation for investigating *graph cyclicity* as it applies to the qubit-assignment problem is not motivated necessarily with respect to realistic implementations, but rather as a method by which to probe the limits of when our HQAA effectively fails to find an adequate solution. In this way, we set forth a method by which to diagnose and analyze the effectiveness of a general HQAA with respect to a diverse plethora of interaction-graph topological structures, while keeping the coupling-graph connectivity constant; moreover, we utilize a similar framework to investigate the scaling properties of our HQAA, in particular as the size of the QPU increases. In the following section, we will detail our approach for the qubit-assignment algorithm, specifically made with the previously mentioned goals in mind.

III. DESCRIPTION OF THE QUBIT-ASSIGNMENT ALGORITHMS

All of the qubit-assignment algorithms utilized in this work generally function in the following manner. First, an $n \times n$ square lattice (representing the geometric connectivity of the QPU) is initialized, along with single-qubit and two-qubit error rates, as well as measurements, as a NetworkX object [68] which will serve as an approximation for the QPU device's CG. Next, a quantum algorithm written in cQASM [69] is parsed into a NetworkX object as well. The qubit-assignment algorithm is then called, a final-assignment solution is assigned, and the assignment is evaluated using a cost function

[1]One may also consider *tree-like* IGs; however, for the purposes of this manuscript, we will concentrate on path-like and cycle-like IGs, and reserve this discussion for further study.

that is described at the end of this section (we will refer to this cost function as the *metric*). The IGs used in this article for QPU simulations were assigned to CGs which consist of 3×3 QPU lattice grids in the simulations from Section IV-B; for the large path-like simulations of Section IV-C, we assign to $n \times n$ grid lattice QPUs qubits, where $n > 3$. All of the code for this project is freely accessible on Github.[2]

For all of the algorithms described in this section, we will refer to an IG, comprised of some set of vertices V and some set of edges E, as $\tilde{I}(V, E)$; additionally, we shall refer to a QPU CG with vertex set V' and edge set E' as $\tilde{Q}(V,' E')$. For the algorithms that we designed, several assumptions were made.

1) If $\Delta(V) \geq \Delta(V')$ (where $\Delta(V)$ represents the *maximal vertex degree* [63]), then an assignment solution exists for $\tilde{I}(V, E)$, which may or may not require SWAP gates.
2) We assume that $|V| \leq |V'|$ (where $|V|$, $|V'|$ represent the total number of vertices in the interaction and CGs, respectively), i.e., that the number of vertices in the IG is smaller than or equal to the number of vertices in the CG. If $|V| > |V'|$, then the assignment process aborts, and an error message is displayed. An example of such an error would concern the assignment of an n-qubit quantum algorithm to an $(n-1)$-qubit CG.

A. HQAA TRAFFIC COEFFICIENT

Our HQAA is greedy in nature [70], and pseudocode for the algorithm is described in Algorithm 1. The HQAA functions generally as follows. The IG $\tilde{I}(V, E)$ and the CG $\tilde{Q}(V,' E')$ are initialized as NetworkX objects after being parsed from an input QASM file. Next, the set of traffic coefficients $V_{i,\text{tc}}$ and the maximal traffic coefficient $V_{i,\text{tc}}^{\max}$ are calculated and stored in lists, which can be interpreted as the overall percentage of gate invocations for a given qubit. These mathematical objects will be explained in more detail in the next two paragraphs. Subsequently, the *infimum* vertex (in the present context we refer to the "infimum-vertex qubit" as the maximal-degree qubit with the minimal two-qubit error-rate edge on the CG, which we denote in Algorithm 1 as $\inf_{\Delta(V'),\mathcal{E}(E')}$, where $\Delta(V')$ and $\mathcal{E}(E')$ represent the maximal-degree and the minimal two-qubit error-rate edge for the CG, respectively) of the CG is selected as the first candidate coupling-graph qubit to be assigned to. Afterward, our algorithm defines an initial assignment from the interaction-graph qubit with the maximal traffic coefficient to the infimum coupling-graph qubit; we cycle through both ordered sets of interaction-graph and coupling-graph qubits, defining neighboring interaction-graph qubits in a nearest-neighbor style on the QPU CG. When all nearest neighbors for a given coupling-graph vertex have been defined already, our algorithm uses Dijkstra's shortest-path algorithm in order to find the physically closest coupling-graph qubit (here we take the "shortest path" to mean the edge(s) that constitute the

[2]https://github.com/mattsteinberg13/heuristic-qubit-mapping-algorithm

Algorithm 1: Pseudocode for the HQAA described in Section III-A. Here, we use the abbreviations *nearest-neighbor* (NN) and *next-closet neighbor* (NCN) in the "if" statement below.

```
Input: $\tilde{I}(V,E)$, $\tilde{Q}(V',E')$
// Get traffic coefficients (see Eq. 1)
$V_{i,\text{tc}} \leftarrow f_i(N_{s,i}, N_{d,i})$
// Get infimum-vertex CG-qubit
$V'_{\text{inf}} \leftarrow \inf_{\Delta(V'),\mathcal{E}(E')} V'$
// Get IG-qubit with maximal tc
$V_{i,\text{tc}}^{\max} \leftarrow \max V_{i,\text{tc}}$
// Perform initial assignment
$V_{i,\text{tc}}^{\max} \mapsto V'_{\text{inf}}$
// Remember as last-assigned CG qubit
$V'_{\text{last-assigned}} \leftarrow V'_{\text{inf}}$
// Assign remaining IG-qubits
for $v \in V_{i,tc} \backslash V_{i,tc}^{max}$ do
    // Identify next candidate CG-qubit
    if $NN\_CG\_qubit\_exists(V'_{last\text{-}assigned})$ then
        // find NN CG-qubit
        $V'_N \leftarrow$ get_NN_CG_qubit($V'_{\text{last-assigned}}$)
    else
        // If all NN $V'_N$ assigned, find NCN
        $V'_N \leftarrow$ get_NCN_CG_qubit($V'_{\text{last-assigned}}$)
    end if
    // Perform consecutive assignment
    $v \mapsto V'_N$
    // Remember as last-assigned CG qubit
    $V'_{\text{last-assigned}} \leftarrow V'_N$
end for
```

overall lowest edge error rate); as such, whenever a nearest-neighbor qubit is not able to be located, our policy is to add SWAP gates, in accordance with the shortest path that the Dijkstra's algorithm finds. This process is continued until all of the interaction-graph qubits have been matched to a corresponding coupling-graph qubit. We will provide more detail in the rest of the section.

The HQAA presented here is necessarily similar to the simple heuristic assignment strategy used in [50]; however, our HQAA is novel in the sense that we utilize the traffic coefficients as a fitness function for the interaction-graph qubits. As mentioned earlier, the notion of traffic coefficients is related to the notion of *vertex centrality* in graph theory [56], [57]. Additionally, [50] evaluates and compares several heuristic qubit-assignment algorithms to SMTP-based algorithms. We stress here that our aim is not to make significant improvements to state-of-the-art qubit-mapping strategies, but rather to directly quantify and qualify topological-graph dependencies between an interaction graph and our

HQAA, as well as investigating the scaling properties of our HQAA. We suspect that our findings will have wider applicability to most qubit-mapping algorithms, although we will comment on this in Section V.

The maximal traffic coefficient is calculated as follows. First, for the ith interaction-graph qubit, we sum the total number of single- and two-qubit gate invocations; the *frequency* of an interaction-graph qubit is subsequently labeled f_i, as shown in (1). Here, we have weighted two-qubit gate invocations ($N_{d,i}$) with an extra linear multiplier of 2 in order to weigh interaction-graph qubits which exhibit a large percentage of the two-qubit gates utilized in a given quantum algorithm more heavily than single-qubit gates. In this way, we take into account for *both* the nontriviality of the single-qubit gate invocations ($N_{s,i}$) and the higher error rates of two-qubit gates, which are typically at least one order of magnitude worse than for single-qubit gates [50], [60]. After a bit of algebra (2) and (3), one sees that the frequency of the ith interaction-graph is rewritten as a *traffic coefficient* $v \in V_{i,\text{tc}}$; these traffic coefficients are summed and normalized, such that cv provides a “percentage-wise” overview of the total interactions for the ith interaction-graph qubit in an algorithm. We then take the *maximal traffic coefficient* $V_{i,\text{tc}}^{\max}$ as corresponding to the first interaction-graph qubit to be assigned, as shown in (4).

$$f_i = N_{s,i} + 2N_{d,i} \tag{1}$$

$$1 - \frac{1}{f_i} = V_{i,\text{tc}} \tag{2}$$

$$c \cdot \sum_i V_{i,\text{tc}} = 1 \tag{3}$$

$$\max V_{i,\text{tc}} = V_{i,\text{tc}}^{\max} \tag{4}$$

As a brief aside, one may ask why the single-qubit error rates are factored in at all, given the large difference in magnitude between the two-qubit and single-qubit error rates. The reason for this is the following: consider a quantum circuit, where the ratio of single-qubit gates to double-qubit gates is much higher than 1. Given such a scenario, we expect that the error rates of the single-qubit gates nontrivially factor into the determination of which virtual qubit should be first allocated.

The maximal traffic coefficient $V_{i,\text{tc}}^{\max}$ represents the percentage of interactions for the “most active” qubit in the algorithm and is used as a way to ascertain which interaction-graph qubits must be prioritized for the best-connected, lowest error-rate portions of the QPU CG via our HQAA that is described in Section III-A. It is entirely possible that more than one interaction-graph qubit may have the largest traffic coefficient; in this case, we simply iterate through the set of qubits with maximal traffic coefficient $V_{i,\text{tc}}^{\max}$ and then treat the rest of the qubits in the assignment process.

In order to assign the rest of the IG, a variant of Dijkstra’s algorithm [71] is utilized, which takes into account the error rates of two- and single-qubit gates (represented on the CG as edges and vertices with assigned error rates) in order to find the “shortest path” (in this case the term *shortest path* refers to the particular sequence of gates which lead to the lowest error rate, which is the shortest path since weights are placed on the edges as two-qubit error rates) to the next available qubit in the IG; once the next candidate interaction-graph qubit is designated, the algorithm surveys the interaction-graph qubits that have already been assigned. Finally, the HQAA assigns the new candidate to the QPU CG, as closely as possible (such that the least amount of errors is generated) to the originally assigned interaction-graph qubit. This process continues until the entire algorithm has been assigned to the closest-possible qubits on the IG.

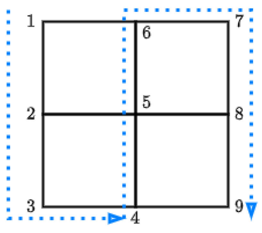

FIGURE 3. Numbering scheme employed by our TAA follows a “snake pattern” across the CG.

B. BRUTE-FORCE AND TRIVIAL QUBIT-ASSIGNMENT ALGORITHMS

The BFAA utilized in this work functions as follows. First, the lattice QPU is initialized, and the error rates for all quantities are defined; next, the BFAA generates a list of all possible permutations for a quantum-algorithm mapping solution. Each permutation is assigned and is evaluated using the metric described in Section III-C. The preceding permutation’s metric value is compared to the current iteration, and the permutation with the highest success-rate metric value is kept, while the inferior one is discarded. This process continues until the best permutation is found.

The TAA functions by sequentially assigning interaction-graph qubits to correspondingly numbered coupling-graph qubits; a quantum algorithm with qubits $q_1 \ldots q_r$, where $r \le n^2$ (n^2 for a $n \times n$ QPU), will be assigned to a CG of a QPU with qubits $Q_1 \ldots Q_s$, where $s \ge r$, by assigning interaction-/coupling-graph qubit pairs as $\{q_1 \mapsto Q_1, q_2 \mapsto Q_2, \ldots, q_r \mapsto Q_r\}$. The success-rate metric is then subsequently evaluated, in order to compare with the other two assignment strategies. In all likelihood, one can imagine many different numbering schemes that will give rise to differing evaluations of the success rate; as such, we fixed our numbering scheme for the QPU CG in order to follow a snake pattern along the processor’s topology, as shown Fig. 3. The numbering scheme used for the IGs can be considered to be arbitrary and follows from the numbering of the qubits in its respective qubit register. As such, our TAA iteratively assigns interaction-graph qubits to a CG while following a snake pattern.

C. EVALUATION OF THE SUCCESS-RATE METRIC

The cost function used to quantify the performance of all algorithms in this work is described below. The purpose of this metric is to approximate the fidelity of the final quantum state after the quantum circuit is assigned and executed, with all gates invoked, in a computationally efficient manner. Other metrics exist [49]–[52]; however, we limit our attention here to a metric that is based on success-rate measures.

The single-qubit gate, two-qubit gate, and SWAP-gate product metrics are calculated as shown below in (5)–(8). These product metrics, as explained above, relate specifically to the CGs that we utilize in this work.

$$\sigma_s = \prod_{i}^{n'<n} (1 - \xi_{s,i})^{N_{s,i}} \tag{5}$$

$$\sigma_d = \prod_{i}^{\delta_d} (1 - \xi_{d,i})^{N_{d,i}} \tag{6}$$

$$\sigma^{\mathrm{SW}} = \prod_{j}^{\delta^{\mathrm{SW}}} \prod_{i}^{l} (1 - \xi_i^{\mathrm{SW}})_j^{(2N_i^{\mathrm{SW}})} \tag{7}$$

$$\sigma_{\mathrm{total}} = \sigma_s \cdot \sigma_d \cdot \sigma^{\mathrm{SW}} \tag{8}$$

where on the first line, σ_s, n', n, $\xi_{s,i}$, $N_{s,i}$ are the total single-qubit gate metric value; the total number of qubits in the IG; the total number of qubits in the CG; the single-qubit gate error rate for each qubit; and the number of single-qubit gate invocations per qubit, respectively. On the second line, σ_d, δ_d, $\xi_{d,i}$, $N_{d,i}$ are the total two-qubit gate metric value; the total number of edges on the NISQ device; the two-qubit gate error rate per edge; and the number of two-qubit gate invocations per edge, respectively. On the final line, σ^{SW}, δ^{SW}, l, ξ_i^{SW}, $2N_i^{\mathrm{SW}}$ represent the total SWAP-gate metric value; the total number of separate edges that need SWAP gates; the total number of edges which physically separate the SWAP-corrected pair of coupling-graph qubits; the two-qubit gate error rate per edge; and the number of two-qubit gate invocations, for which the error rates overall are squared (this takes into account the cost of moving the qubit both to and from the closest unoccupied region, using the SWAP gate), respectively. Finally, σ_{total} represents the total metric calculated from the product of all other success rates, as mentioned above. For simplicity, we do not explicitly take into account how SWAP gates may be invoked on real QPU devices [45], [46].

Since the HQAA itself functions by allocating interaction-graph qubits to coupling-graph qubits based on a measure of the success rate, one may ask how useful it is to utilize a metric based on the same measures. The reason for this can be seen as follows: when considering the optimal solution for any qubit-assignment algorithm, there are a variety of different cost functions that one may utilize. However, *any assignment* that is based on an exact calculation of the fidelity of the final quantum state after running it on a physical QPU will give the best indication of performance for an assignment solution [72]. Since direct computations of the fidelity are computationally resource-intensive and scale exponentially with the size of the QPU [18], [73], we opt to utilize a cost

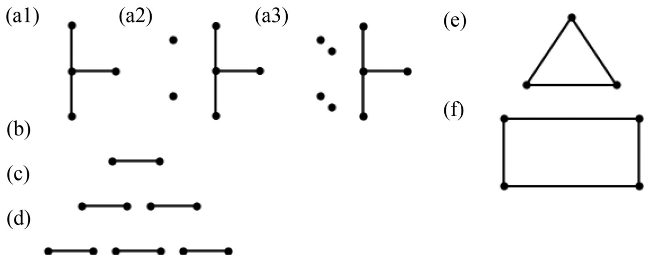

FIGURE 4. IGs of several realistic benchmarks which were tested in Section IV-A; the benchmarks themselves were taken from [50]. a1)–a3) show the BV4, BV6, and BV8 benchmarks, respectively; (b) represents the QFT and HS2 benchmark algorithms; the IGs in (c) and (d) were used for the HS4 and HS6 algorithms; (e) depicts the Fredkin, Or, Peres, and Toffoli algorithms; and (f) displays the Adder benchmark.

function that may be regarded as related to the calculation of the fidelity (i.e., a success rate measurement, based on the error rates as presented). In this way, we attempt to employ a cost function that is as relevant as possible, without the computational demands incurred by large-scale simulations.

In agreement with [50], a precise formulation of the noise model is not warranted in this work. One main reason for this discrepancy is due to the inherent nature of the cost function used, which takes into account the experimentally determined noise-calibration statistics, not the specific details of the quantum channels acting upon the system. Further details are not necessary, as such inclusions will impact the scalability of the qubit-assignment algorithm.

In the next section, we will discuss in full the results obtained from studying several realistic interaction-graph benchmarks, benchmarks whose IGs exhibit high degrees of cycle-like edges, and benchmarks that increase in size and sequentially occupy more and more of a given QPU coupling-graph's qubits.

IV. RESULTS

The results are organized as follows. Section IV-A details the results obtained from realistic benchmark IGs, assigned to a 3×3 CG. Section IV-B discusses the results from assigning quantum benchmarks with IGs that exhibit increasing amounts of nonnearest neighbor two-qubit gate combinations. The results from scaling the size of path-like IGs onto ever-increasing QPU CGs are described in Section IV-C. All benchmarks were tested on a Dell Latitude 7400 laptop with a 1.9 GHz $\times$ 4 Intel i7-8665 U quadcore processor and 8.0 GB of RAM. Each benchmark was assigned using our simulation 100, 100, and 1000 times in Sections IV-A and IV-C, respectively; success rates were averaged over all trials. Simulations of Sequences I and II (as shown in Fig. 6) took approximately 60 hours of continuous runtime.

A. REALISTIC INTERACTION-GRAPH BENCHMARKS

The benchmarks from [50] were utilized in this section. The corresponding IGs for each of the benchmarks listed are shown in Fig. 4. All of the benchmarks were tested on a 3×3 lattice CG, and the results are reported in Fig. 5. In this section, we did not explicitly take into account whether

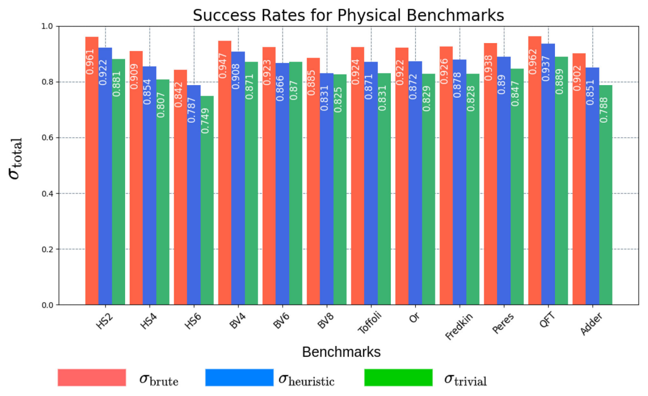

FIGURE 5. **Success rate for the benchmarks in [50]. The BFAA results are shown in red, followed by blue and green bars, which represent the HQAA and TAA results, respectively.**

the IGs are path-like or cycle-like; path-like and cycle-like interaction-graph benchmarks will be investigated and compared in detail in Sections IV-B and IV-C.

The results obtained from the BFAA, HQAA, and TAA with respect to the benchmarks detailed in Fig. 4 are shown in Fig. 5. The algorithms themselves are grouped: the first three groupings of results pertain to the *hidden-shift algorithmic* benchmarks; the second three groupings relate the results obtained for the *Bernstein–Vazirani* benchmarks; as for the next grouping, the results belonging to the *Toffoli, Or, Fredkin*, and *Peres* benchmarks exhibit triangular IGs; and finally, the *quantum Fourier transform* (QFT) and *Adder* benchmarks admit IGs as shown in Fig. 4(b) and (f), respectively. Red bars denote the results from the BFAA; blue and green bars denote the results from the commensurate HQAA and TAA, respectively. The y-axis of Fig. 5 displays the calculated success rates.

As is evidenced in the graphs, the BFAA provides an effective upper bound for the performance of the HQAA (we use the term *effective upper bound* with the view that better solutions may exist if one utilizes *time-scheduling techniques*, as mentioned in Section II). Additionally, the TAA allows for an interpretation as an effective lower bound, with the HQAA pivoting between both of these effective bounds. The HQAA outperforms the TAA in virtually all cases except for those involving disjointness in the interaction-graph quantum algorithms, as indicated by the success-rate values of Fig. 4 a2) and Fig. 4 a3).

One may naïvely conclude from the results obtained that the HQAA used in this work performs generally well for all types of interaction-graph topologies, both cycle-like and path-like.[3] However, one may expect that topological-graph effects play a limited role in such small-scale quantum algorithms. Therefore, our results from this section serve to motivate a more systematic investigation of the topological-graph effects of an IG on the performance of our HQAA. We will show in the next section that topological-graph structure plays a role in the performance of our HQAA for IGs larger than six virtual qubits, and thus, were not apparent in the results of this section.

[3]The only exception to this rule can be seen in the *disjoint* IGs that were utilized for the HS4 and HS6 algorithms. Such a complication is hardly unexpected, as our HQAA was intended for connected-IGs only. One can in principle consider additions in order address quantum algorithms that prepare product states; however, we reserve the discussion on disjoint IGs for future work.

B. TOPOLOGICALLY INEQUIVALENT INTERACTION-GRAPH BENCHMARKS

The results of this section detail comparison between the HQAA, BFAA, and TAA. These two additional qubit-assignment algorithms provide *effective* upper and lower bounds on the performance of the HQAA, just as in the previous section. The 4-, 6-, and 8-qubit IGs were used as benchmarks to be assigned onto a 3×3 lattice CG. Two sequences of two-qubit gate additions for 6- and 8-qubit IGs are utilized and are shown in Fig. 6. We sequentially add cycle-like edges to IGs, starting from their path-like counterparts. Our aim is twofold. First, we wish to characterize and associate the performance of our HQAA with the *number of cycle-like edges* in the corresponding IG; furthermore, we are interested in the specific topological properties of said IGs. In this section, we added cycle-like edges in two different patterns: following a *depth-first* pattern, which seeks to saturate the vertex degree of one vertex of the IG before adding more edges to the next, and *breadth-first* pattern, in which edges are added in such a way that the vertex degrees of all nodes stay approximately equal.

In the small-n regime (for a $n \times n$ QPU CG), BFAAs can be utilized, as the size of the coupling and IGs are sufficiently small. Even so, it must be stated that in all of our simulation results, our BFAA requires several orders of magnitude more time in order to complete each trial than the HQAA or TAA.

Cycle-like chords that were sequentially added to s-qubit path-like IGs are shown in Fig. 6; here, $s \in \{4, 6, 8\}$. Due to the fact that there are several different ways in which one may add chords to the six- and eight-qubit quantum algorithms, two different ways of edge addition were used for the respective s-qubit cycles, in order to explore and compare two sets of IGs that are equivalent in all ways, except when viewed through the lens of *topological graph equivalence*. We name these resulting sets of IGs generated by each iterative procedure as *breadth-first* or *depth-first* sequences. As shown in Sequence I (see Fig. 6), a cycle is immediately created upon adding an edge between the first and last vertices of a path-like IG; subsequently, chords are added to the s-qubit cycle such that the degree of every sequentially numbered vertex is maximized before proceeding to add chords to the consecutively numbered vertices. We refer to this style of adding edges as a *depth-first* approach. Sequence II, however, exhibits a *breadth-first* approach to cycle-like edge addition, as chords are added in a manner such that the vertex degree of all vertices remain approximately equal. The red-highlighted edges represent newly added chords to a particular sequence.

One may ask what the reasoning behind such an elaborate edge-generation procedure may be; after all, is it not

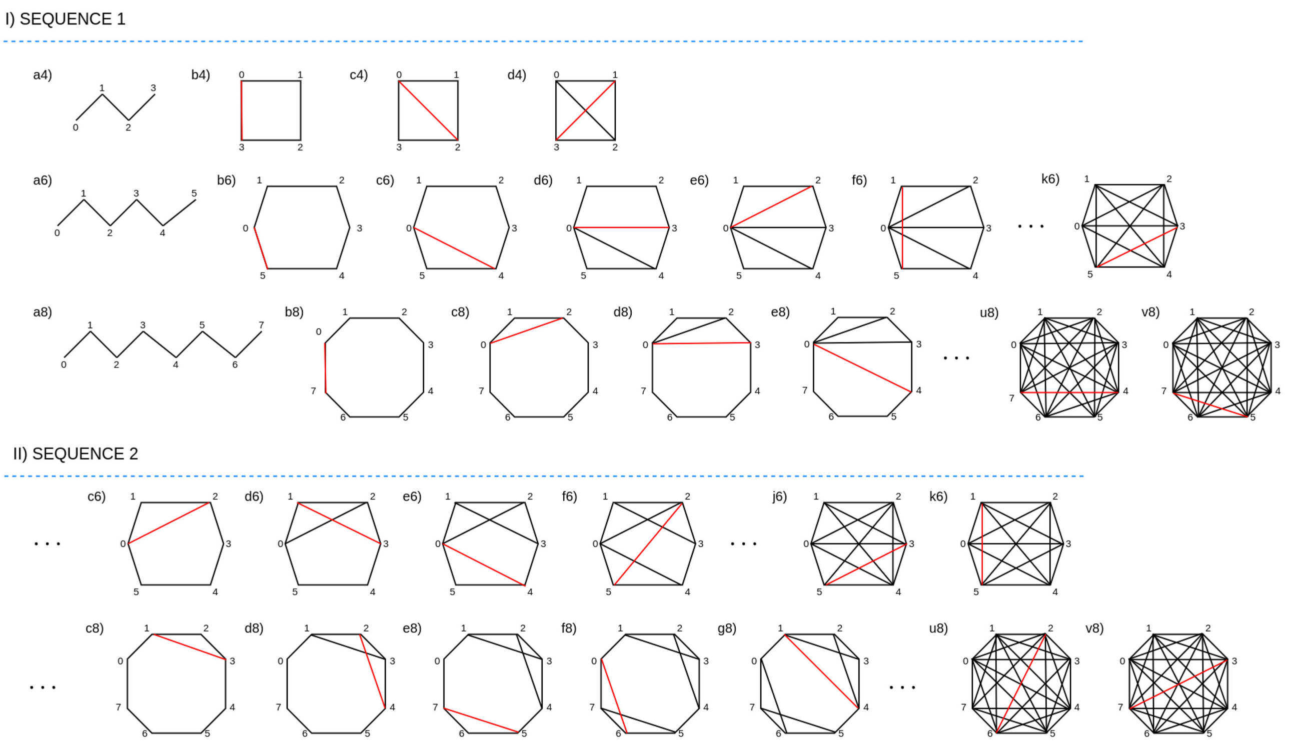

FIGURE 6. **IGs for the first and second benchmark sequences. Red edges depict newly added cycle-like edges to the IGs that were tested with our HQAA. In the IGs from Sequence I, cycle-like edges are appended such that every vertex degree is maximized before subsequently appending cycle-like edges to consecutively higher-numbered vertices; this process continues until a K_s graph is attained for Sequence I (We refer to this technique as a *depth-first* edge-assignment procedure). As for Sequence II, a *breadth-first* edge-assignment procedure is utilized, as cycle-like edges are appended such that every vertex exhibits approximately the same degree until a K_s graph is reached.**

sufficient to only consider the numbers of virtual qubits and gate overhead for understanding the qubit-assignment properties of a quantum algorithm using an HQAA? One of the main purposes of this manuscript is to show that topological-graph properties such as *degree centrality* play an important role in qubit-assignment when using an HQAA; as such, it was imperative to construct sets of interaction-graph benchmarks which exhibit the same numbers of two- and single-qubit gates, exhibiting their only differences in their topological-graph properties. As an example, consider a map f from graph d6) in Sequence I (which for this example we shall denote X for simplicity) to graph d6) in Sequence II (we here we shall denote Y for simplicity). According to the definition of *topological equivalence* introduced in Section II, three main conditions must be fulfilled, in order to define such a map $f : X \mapsto Y$. It should be obvious that condition 1 does not hold, as their graph Laplacians [63] do not share the same eigenvalue spectrum.[4] In our simulations, we hold the total numbers of single- and two-qubit gates to be the same for our benchmarks but allow for different topological structures to emerge between IGs, as shown by Fig. 6. In this way, we have positioned ourselves such that we can now evaluate the performance of an HQAA relative to varying the topological-graph properties of the IGs while keeping the coupling-graph structure constant.

The results for 4-, 6-, and 8-qubit IGs that were assigned in accordance with the IGs in Sequence I and II are depicted in Fig. 7. Fig. 7(a)–(c) represents the results procured from IGs in Sequence I; Fig. 7(d) and (e) represent the Sequence II results. In each subfigure, the BFAA, HQAA, and TAA results are shown in red, blue, and green, respectively. The success rates of each assignment algorithm $\{\sigma_{\text{brute}}, \sigma_{\text{heuristic}}, \sigma_{\text{trivial}}\}$ are graphed as a function of the number of cycle-like edges that have been added from the original sequence start in Ia4), Ia6), and Ia8) in Fig. 6. The average runtime per trial for the BFAA, HQAA, and TAA are on the order of about 60 s for the BFAA versus 0.5 and 0.2 ms for the HQAA and TAA, respectively. These figures largely stay the same for the Sequence II benchmarks, as the BFAA exhibits an average solution time of several orders of magnitude higher than the other two qubit-assignment algorithms.

Fig. 7(a) shows that our HQAA's success rate can approach the BFAA's for 4-qubit IGs, no matter how many cyclic chords are added; indeed, it is observed that even a quantum algorithm representing a K_4 IG can be effectively assigned to the 3×3 lattice CG in our simulations. In contrast, Fig. 7(b) and (c) depict performance decreases for our heuristic

[4]It is well known in spectral graph theory [63], [74], [75] that two graphs X and Y are isomorphic (i.e., exhibit an edge-set and vertex-set bijection) if a permutation matrix $\mathcal{P}$ exists for which $\mathcal{P}X\mathcal{P}^{-1} = Y$; a necessary consequence of this is that both graph Laplacians share the same eigenvalue spectrum. Since we know *a priori* that X and Y do not share the same eigenvalue spectrum, we can conclude that no permutation matrix exists such that a vertex- and edge-set bijection exists; thus, condition 1 does not hold.

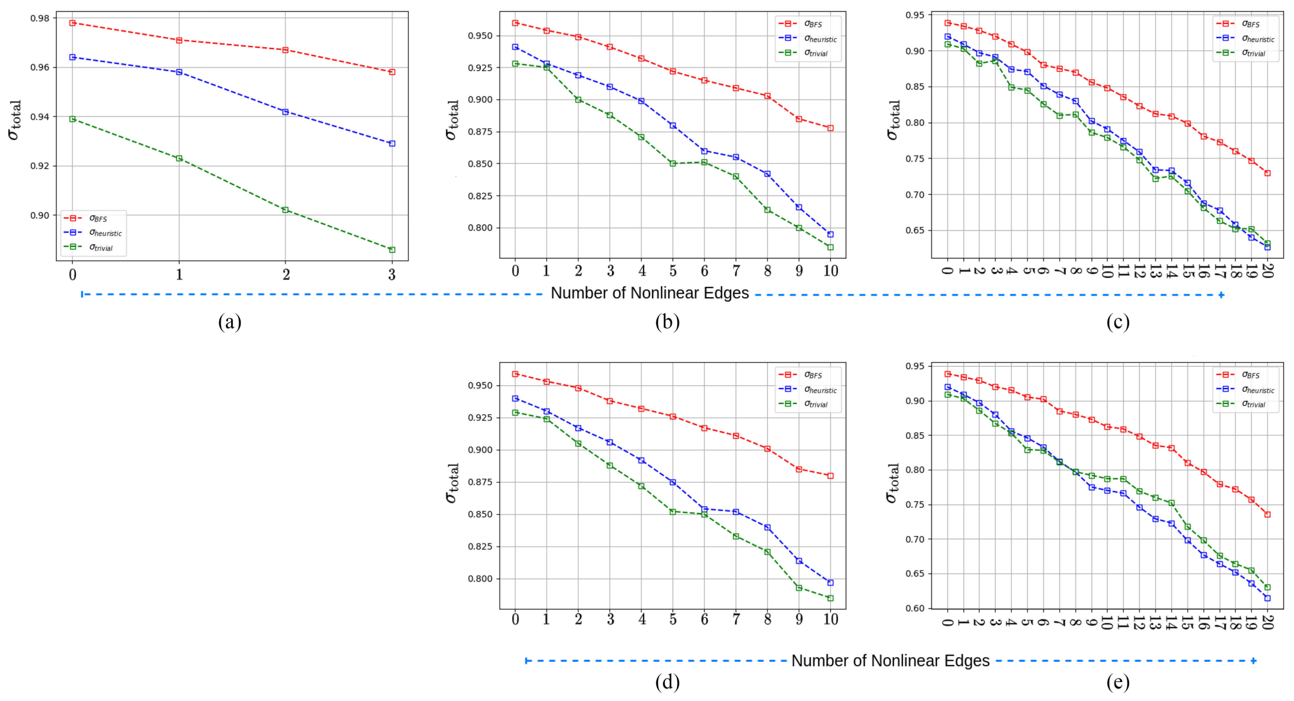

FIGURE 7. **(a)–(c) Results for the cycle-like benchmarks in Fig. 6 for Sequence I (the depth-first benchmarks); the *x*-axis of the figure follows the sequence of cycle-like edge additions. The average runtime per trial for all of the benchmarks was approximately 60 s, 0.5 ms, and 0.2 ms for the BFAA, HQAA, and TAA, respectively. The total runtime for the entire simulation was approximately 60 h. (d) and (e) Results for the cycle-like benchmark IGs in Fig. 6 for Sequence II. These breadth-first benchmarks exhibited an average runtime per trial were approximately equal to those stated for the depth-first benchmarks.**

optimization as more and more cycle-like edges are added to the IG. Fig. 7(c), however, is unique in that the performance of the HQAA achieves roughly analogous performance to that of the TAA, until finally, after 18 cycle-like edges have been added, a “critical point” is reached. We define this critical point to be the point at which the HQAA’s success rate matches that of the TAA’s. This behavior is not witnessed in the 6-qubit IGs; indeed, although the 6-qubit benchmarks do exhibit steadily decaying performance, our results indicate that this tendency is more prevalent with larger quantum circuits, such as the 8-qubit benchmarks.

The breadth-first benchmarks of Sequence II exhibit a distinct trend from the first-sequence IGs. Although Fig. 7(d) largely appears to agree with Fig. 7(b), the second-sequence IGs in Fig. 7(e) exhibit much worse HQAA success rates, compared to the first-sequence alternatives. In point of fact, the critical point appears sooner in our simulations while following Sequence II. Approximately 18 cycle-like edges were added in the depth-first analysis of Sequence I; in contrast, the point wherein the HQAA’s success rate is equal to that of the TAA’s is observed after approximately 7–8 cycle-like edges have been added from Sequence II (breadth-first).

A few comments are in order here. First, the data obtained from these simulations reveal that our HQAA can adequately assign IGs that exhibit a low degree of cycle-like edges, provided that the quantum circuit in question is not very deep. As we increase the sheer amount of the two-qubit gate invocations between interaction-graph qubits, it can be seen that our HQAA can tolerate a small amount of interaction-graph cyclicity, especially if this cyclicity is centralized on a subset of the total vertices. Next, and perhaps more importantly, for a given number of cycle-like edges added to a path-like IG, the success rate varies significantly for our HQAA. However, if we take two IGs from Sequences I and II with equivalent numbers of edges, we found that the local distribution of edge assignments play a role in the success rates observed, as these localized edge-assignment discrepancies are the only difference between any two pairs of IGs from Sequences I and II. This local distribution discrepancy generates the observed topological inequivalence *globally* for the IGs and plays a significant role in the performance of the HQAA.

These same simulations were performed for the same benchmarks, with the only difference being that the number of total gates for each benchmark was multiplied by two and four. Our results largely conform with those shown in Fig. 7, as only the scaling of the success-rate metric changes. This observation is indeed expected, as the design of the experiment and the cost function themselves facilitate such a rescaling.

Lastly, as shown in Fig. 7(c) and (e), the heuristic’s and TAA’s success rates are relatively close. One may surmise that a reason for this proximity is related to the high occupancy of the coupling-graph qubits during our simulations (8/9 of the available QPU qubits were utilized). The next section will provide information related to the large path-like interactions graphs that were tested and how they scale in the

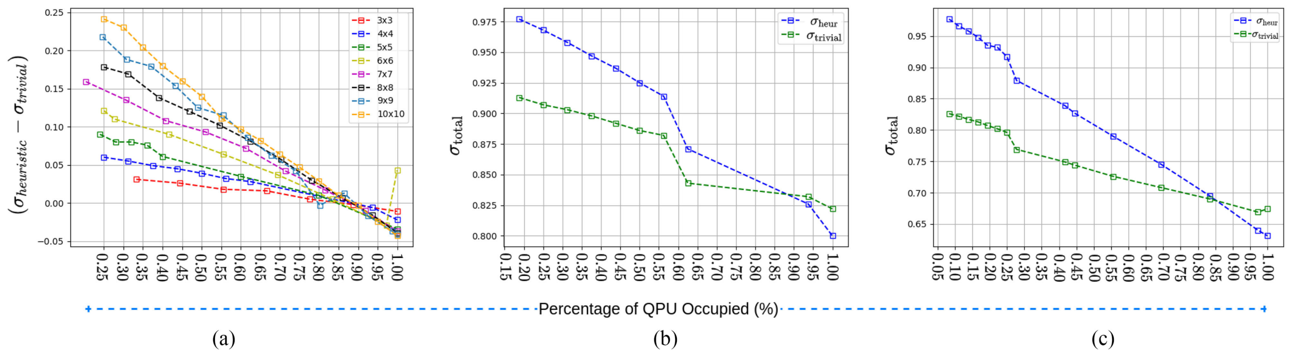

FIGURE 8. **Success-rate differences between the success rates of the HQAA and TAA ($\sigma_{heurstic}, \sigma_{trivial}$) taken from *n*-qubit path-like IGs which have been assigned to $n \times n$ QPU CGs. a) Success rates differences without any noise scaling. (b) and (c) Success rate σ_{total} calculated for both $n = 4$ and $n = 6$ QPU CGs, respectively.**

regime $n > 3$ for a corresponding $n \times n$ QPU CG. We specifically address the question of how the occupancy percentage of the QPU affects our HQAA's solution.

C. SCALING PROPERTIES OF OUR HQAA ON LARGER QPU LATTICES

The benchmarks tested in this section are all path-like, but vary in size so as to occupy different percentages of an $n \times n$ QPU CG, where $n = 3, \ldots, 10$. A CG of lattice dimensions $n \times n$ is initialized, and sequentially larger and larger path-like IGs are assigned to a CG until it is 100% occupied; as in the previous simulations, vertices of the CG are taken to be qubits. Additionally, no explicit noise scaling was utilized as the CG increases; the magnitudes of noise for two-qubit and single-qubit error rates, as well as measurements, are kept the same as described in Section III. As BFAAs were not utilized for this section (becoming practically intractable if $n > 3$), success rates were instead compared between the HQAA and TAA, as shown in Fig. 8(b) and (c).

The results from our study are displayed in Fig. 8. Fig. 8(a) shows the success-rate difference $\{\sigma_{\text{heuristic}} - \sigma_{\text{trivial}}\}$, calculated between our HQAA and the TAA, with respect to path-like IGs that progressively fill the entire QPU CG. The different colors in Fig. 8(a) denote differing n-values for the lattice dimensions of the CG. Fig. 8(b) and (c) displays the actual success rates calculated under no noise scaling behavior, for 4×4 and 6×6 coupling-graph dimensions, respectively.

In these graphs, one can observe several trends: first, as the n-value of the CG increases, the success-rate difference becomes steadily larger for path-like IGs that occupy the same percentage of the CG when fully assigned, until about 75% of the QPU is occupied. Next, after approximately 75% of the CG has been filled, the success-rate difference becomes negligible; this success-rate difference implies that an effectively negligible difference between the heuristic- and the TAA is observed. Lastly, after roughly 85%–90% of the lattice CG is occupied, the success-rate difference is not only negligible but starts to dip below zero, the main observation here being that the HQAA cannot find a solution that outperforms the success rate measured from the TAA. We will discuss the consequences of this observation in Section V.

Additionally, we performed the same tests for CGs with increasing n-values such that a uniform increase in noise of order four times larger than the noise in Fig. 8, as well as under exponentially increasing noise. The purpose of these rescaled-noise simulations was to investigate potentially more realistic noise, as device error rates are expected to increase due to crosstalk as quantum processors become larger [60], [76]. Our results from these simulations largely confirm the same trends that were mentioned in the preceding paragraphs, albeit with steeper linear decays.

V. DISCUSSION

In this article, we have introduced an HQAA for the purpose of exploring the advantages and limitations of assigning topologically-inequivalent IGs to quantum hardware, in addition to systematically investigating the scaling behavior of our HQAA as the QPU size increases. The HQAA itself is *noise-aware*, and the success rate is high, relative to a BFAA and TAA, which serves to effectively bound the performance of our HQAA from above and below. For small, low-depth quantum circuits, the HQAA is shown to provide a significant performance gain over a TAA solution, and in some cases for realistic benchmarks even approaches that of a BFAA.

Two main stages of our work have led us to novel results about the behavior of simple HQAAs. First, we have investigated the performance of our BFAA, HQAA, and TAA with low-depth quantum circuits which admit cycle-like IGs. Two particular edge-addition sequences were considered, both of which highlight the inherent limitations of the HQAA that we devised, albeit in different ways. The topological-graph properties of the cycle-like edges added to the circuit (i.e., depth-first or breadth-first edge assignments), as well as the sheer amount of cycle-like edges utilized, both were found to play a role in our HQAA's calculated success rate. These two observations are evidenced from an analysis of Fig. 7(c) and (e): although our HQAA performs better than the TAA solution for most of the depth-first edge additions (Sequence I), we see that the performance is *not much better*, implying that our HQAA can tolerate a relatively low amount of interaction-graph cyclicity, in the case of the number of qubits in the IG being relatively high. This observation comes as no surprise, seeing as the algorithm was designed to accommodate path-like IGs. Furthermore, for two IGs with the same number of cycle-like edges, it was found that our HQAA's performance depends on the particular manner in which the cycle-like interaction-graph edges are distributed. We verified that this fact is reinforced for deeper algorithms by running simulations with larger-depth IGs of the same form as described in Fig. 6. More specifically, Sequence I and Sequence II differ from each other due to their *vertex centrality*. When looked at from this perspective, it is unsurprising that our HQAA, tailored to take into account vertex centrality, performs better for IGs that additionally exhibit. What is in fact novel about these results is *how quickly* the performance of our HQAA falls off in comparison to another, topologically inequivalent IG, and also in comparison to our approximate upper- and lower-bounds, provided by the BFAA and TAA. This falloff demonstrates that a topological-graph dependence is exhibited for our particular assignment strategy employed.

Second, we investigated the scaling behavior of our HQAA in the regime $n > 3$ for $n \times n$ QPU CGs. In this regime, our BFAA solutions to the assignment problem become intractable; as such, the HQAA was compared to the TAA described in Section III. Our novel results indicate that the HQAA scales well for quantum circuits with path-like IGs, as long as less than approximately 75% of the QPU CG is filled (in comparison to our TAA). If one occupies more of the available space on the QPU, one can expect a trivial benefit from utilizing our HQAA, until approximately 85% of the processor is allocated; after this marker, performance losses can be expected (with respect to our TAA), as our results denoted. Additionally, for comparable percentages of the CG that are filled, one can expect higher success rates relative to the TAA solution as n is steadily increased. These same simulations were additionally employed for uniformly and exponentially increasing noise parameters, concomitant with observations that larger QPU devices experience more problems with error rates due to crosstalk [60], [76]. Our results confirm and reinforce the remarks stated above.

One may ask why our HQAA tends to underperform as we occupy larger percentages of a QPU CG. An answer to this question can be seen when one considers that, as the CG is filled up, fewer and fewer nearest-neighbor choices are left for the HQAA to evaluate. As the algorithm itself functions by selecting first the maximum-degree vertex of the minimal error-rate edge, the heuristic-based solution will necessitate more SWAP gates as the processor is further allocated. In this regime (i.e., when over 75% of the QPU is occupied), the TAA used in this study would be expected to outperform our HQAA for quantum circuits that give rise to path-like IGs. In this sense, using notions of vertex centrality for an HQAA's strategy may lead to detrimental results in the limit when over 75% of the processor is filled. An example of this difficulty can be seen in Fig. 9; indeed, in Fig. 9(a) and (b), one sees the result of a greedy shortest-path assignment applied to a 5 × 5 CG. In Fig. 9(a), we may start with a high-degree vertex (shown in green); as the algorithm proceeds to find nearest-neighbor solutions, the algorithm's attempt to assign the 15-qubit path-like IG in our example may run into a portion of the device that is less highly connected (shown by the blue-dotted line which terminates at the blue-labeled vertex). When such an event happens, the HQAA will continue searching for the nearest possible neighbor that is free; unfortunately, in this case, several SWAP gates would be needed in order to realize the mapping shown (denoted by the magenta-dotted line that terminates at the red vertex q_{14}). In Fig. 9(b), we see an example of our TAA that would make better use of the space requirements for the 15-qubit quantum circuit. One solution to this issue with our HQAA may be to utilize *look-ahead* or *look-behind* techniques, which would serve to match not only the nearest neighbor on the CG but to additionally analyze several interaction-graph vertices [62], [77], [78]. Such a qubit-mapping algorithm may improve overall success rates. In any case, it is clear that the choice of HQAA may or may not perform well for a given CG and cost function; we leave further discussion to future work.

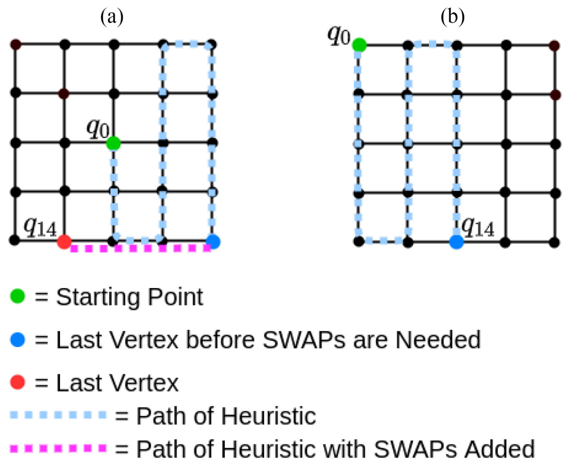

FIGURE 9. **Example scenario in which our HQAA could perform worse in comparison to a TAA. On the 5 × 5 CG pictured above, (a) and (b) represent possible final-assignment solutions for the HQAA and TAA, respectively. In (a), one may initiate the assignment process in a highly-connected region of the CG; however, as one proceeds using the algorithm to consecutively assign the 15 qubits in our quantum-circuit example, one may encounter a situation in which the heuristic-based solution may involve several more SWAP gates than normally anticipated. This situation is shown as a dotted line in blue (tracing out the nearest-neighbor assignment path of our HQAA) which essentially runs into a corner in the lower-right portion of the QPU CG, stopping at the vertex shown in blue. From this point forward, the next shortest-path distance would not be a nearest-neighbor vertex, and SWAPs will undoubtedly be needed in order to realize such an assignment solution (shown by the dotted line in magenta, which terminates at the red-labeled vertex q_{14}). In (b), a TAA solution would better utilize the space and connectivity available for a QPU when fewer choices are available.**

As a final comment, it was observed that our BFAA scaled badly after $n = 3$ for the CGs tested; this does not necessarily imply that *any exact simulation* is intractable. Although we note that finding an exact solution to the qubit-mapping problem is NP-hard, one may be able to write an exact mapping algorithm that does in fact scale better than ours for larger n-values.

Various other qubit-mapping algorithm proposals exist as well [41], [42], [47]–[49], [51], [52], [54], [79]; many of these could be analyzed and classified, as it is imperative to better understand which types of qubit-assignment algorithms may be most amenable for certain structural classes of quantum algorithms. In the interim, we expect that most (if not all) HQAAs will exhibit a topological-graph dependence via the IGs utilized, although this must be explored and verified in future work. Additionally, profiling the topological-graph properties of further benchmark sets such as those from [80] and [81] may serve to inform future proposals for QAAs, as well as more general qubit-mapping approaches. In this way, understanding the topological-graph properties of the qubit-mapping problem may serve to elucidate a more fundamental mathematical metric to quantify the performance of qubit-mapping strategies.

ACKNOWLEDGMENT

The authors would like to thank J. Eisert for useful discussions. The authors would also like to thank HQS Quantum Simulations GmbH (where a portion of this work was completed), and P. Murali for providing the realistic benchmark files from [50] which were utilized in Section IV-A.

REFERENCES

[1] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019, doi: 10.1038/s41586-019-1666-5.

[2] D. Leibfried, R. Blatt, C. Monroe, and D. Wineland, "Quantum dynamics of single trapped ions," *Rev. Mod. Phys.*, vol. 75, pp. 281–324, Mar. 2003, doi: 10.1103/RevModPhys.75.281.

[3] P. Jurcevic *et al.*, "Demonstration of quantum volume 64 on a superconducting quantum computing system," *Quantum Sci. Technol.*, vol. 6, no. 2, 2020, Art. no. 025020, doi: 10.1088/2058-9565/abe519.

[4] H. Zhong *et al.*, "Quantum computational advantage using photons," *Science*, vol. 370, no. 6523, pp. 1460–1463, 2020, doi: 10.1126/science.abe8770.

[5] J. M. Pino *et al.*, "Demonstration of the QCCD trapped-ion quantum computer architecture," 2020, *arXiv:2003.01293*, doi: 10.48550/arXiv.2003.01293.

[6] J. Reiner, F. Wilhelm-Mauch, G. Schön, and M. Marthaler, "Finding the ground state of the Hubbard model by variational methods on a quantum computer with gate errors," *Quantum Sci. Technol.*, vol. 4, no. 3, May 2019, Art. no. 035005, doi: 10.1088/2058-9565/ab1e85.

[7] J. Reiner *et al.*, "Effects of gate errors in digital quantum simulations of fermionic systems," *Quantum Sci. Technol.*, vol. 3, no. 4, Aug. 2018, doi: 10.1088/2058-9565/aad5ba.

[8] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, "Quantum computational chemistry," *Rev. Modern Phys.*, vol. 92, no. 1, Mar. 2020, doi: 10.1103/RevModPhys.92.015003.

[9] T. R. Bromley *et al.*, "Applications of near-term photonic quantum computers: Software and algorithms," *Quantum Sci. Technol.*, vol. 5, no. 3, May 2020, Art. no. 034010, doi: 10.1088/2058-9565/ab8504.

[10] J. Biamonte *et al.*, "Quantum machine learning," *Nature*, vol. 549, pp. 195–202, 2017, doi: 10.1038/nature23474.

[11] V. Dunjko and H. J. Briegel, "Machine learning & artificial intelligence in the quantum domain: A review of recent progress," *Rep. Prog. Phys.*, vol. 81, no. 7, Jun. 2018, Art. no. 074001, doi: 10.1088/1361-6633/aab406.

[12] S. P. Jordan, K. S. M. Lee, and J. Preskill, "Quantum computation of scattering in scalar quantum field theories," 2019, *arXiv:1112.4833*, doi: 10.48550/arXiv.1112.4833.

[13] S. P. Jordan, K. S. M. Lee, and J. Preskill, "Quantum algorithms for quantum field theories," *Science*, vol. 336, no. 6085, pp. 1130–1133, Jun. 2012, doi: 10.1126/science.1217069.

[14] L. Funcke *et al.*, "Towards quantum simulations in particle physics and beyond on noisy intermediate-scale quantum devices," *Phil. Trans. R. Soc. A*, vol. 380, no. 2216, 2022, doi: 10.1098/rsta.2021.0062.

[15] M. Bagherimehrab, Y. R. Sanders, D. W. Berry, G. K. Brennen, and B. C. Sanders, "Nearly optimal quantum algorithm for generating the ground state of a free quantum field theory," 2021, *arXiv:2110.05708*, doi: 10.48550/arXiv.2110.05708.

[16] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," *Rev. Mod. Phys.*, vol. 74, pp. 145–195, Mar. 2002, doi: 10.1103/RevModPhys.74.145.

[17] S. Pirandola *et al.*, "Advances in quantum cryptography," *Adv. Opt. Photon.*, vol. 12, no. 4, Dec. 2020, Art. no. 1012, doi: 10.1364/AOP.361502.

[18] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018, doi: 10.22331/q-2018-08-06-79.

[19] E. A. Sete, W. J. Zeng, and C. T. Rigetti, "A. functional architecture for scalable quantum computing," in *Proc. IEEE Int. Conf. Rebooting Comput.*, 2016, pp. 1–6, doi: 10.1109/ICRC.2016.7738703.

[20] J. Hsu, "Ces 2018: Intel's 49-qubit chip shoots for quantum supremacy," *IEEE Spectr.*, 2018. [Online]. Available: https://spectrum.ieee.org/intels-49qubit-chip-aims-for-quantum-supremacy

[21] Intel PR, "Intel delivers 17-qubit superconducting chip with advanced packaging to qutech," *Intel Newsroom*, 2017. [Online]. Available: https://newsroom.intel.com/news/intel-delivers-17-qubit-superconducting-chip-advanced-packaging-qutech/

[22] S. J. Devitt, W. J. Munro, and K. Nemoto, "Quantum error correction for beginners," *Rep. Prog. Phys.*, vol. 76, Jun. 2013, Art. no. 076001, doi: 10.1088/0034-4885/76/7/076001.

[23] L. Egan *et al.*, "Fault-tolerant operation of a quantum error-correction code," 2020, *arXiv:2009.11482*, doi: 10.48550/arXiv.2009.11482.

[24] N. M. Linke *et al.*, "Fault-tolerant quantum error detection," *Sci. Adv.*, vol. 3, no. 10, Oct. 2017, Art. no. 1701074, doi: 10.1126/sciadv.1701074.

[25] P. Murali, N. M. Linke, M. Martonosi, A. J. Abhari, N. H. Nguyen, and C. H. Alderete, "Full-stack, real-system quantum computer studies: Architectural comparisons and design insights," in *Proc. 46th Int. Symp. Comput. Architecture*, New York, NY, USA, 2019, pp. 527–540. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8980312

[26] J. You and F. Nori, "Atomic physics and quantum optics using superconducting circuits," *Nature*, vol. 474, pp. 589–597, 2011, doi: 10.1038/nature10122.

[27] C. Neill *et al.*, "A blueprint for demonstrating quantum supremacy with superconducting qubits," *Science*, vol. 360, no. 6385, pp. 195–199, 2018, doi: 10.1126/science.aao4309.

[28] R. Barends *et al.*, "Superconducting quantum circuits at the surface code threshold for fault tolerance," *Nature*, vol. 508, pp. 500–503, 2014, doi: 0.1038/nature13171.

[29] H. L. Huang *et al.*, "Superconducting quantum computing: A review," *Sci. China Inf. Sci.*, vol. 63, 2020, Art. no. 180501, doi: 10.1007/s11432-020-2881-9.

[30] A. Wallraff *et al.*, "Strong coupling of a single photon to a superconducting qubit using circuit quantum electrodynamics," *Nature*, vol. 431, pp 162–167, 2004, doi: 10.1038/nature02851.

[31] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-ion quantum computing: Progress and challenges," *Appl. Phys. Rev.*, vol. 6, no. 2, Jun. 2019, Art. no. 021314, doi: 10.1063/1.5088164.

[32] P. Schindler *et al.*, "A quantum information processor with trapped ions," *New J. Phys.*, vol. 15, no. 12, Dec. 2013, Art. no. 123012, doi: 10.1088/1367-2630/15/12/123012.

[33] J. I. Cirac and P. Zoller, "Quantum computations with cold trapped ions," *Phys. Rev. Lett.*, vol. 74, pp. 4091–4094, May 1995, doi: 10.1103/PhysRevLett.74.4091.

[34] T. Monz *et al.*, "14-qubit entanglement: Creation and coherence," *Phys. Rev. Lett.*, vol. 106, no. 13, Mar. 2011, Art. no. 130506, doi: 10.1103/PhysRevLett.106.130506.

[35] R. Maurand *et al.*, "A CMOS silicon spin qubit," *Nature Commun.*, vol. 7, 2016, Art. no. 13575, doi: 10.1038/ncomms13575.

[36] S. Slussarenko and G. J. Pryde, "Photonic quantum information processing: A concise review," *Appl. Phys. Rev.*, vol. 6, no. 4, Dec. 2019, Art. no. 041303, doi: 10.1063/1.5115814.

[37] J. Eli Bourassa *et al.*, "Blueprint for a scalable photonic fault-tolerant quantum computer," *Quantum*, vol. 5, p. 392, Feb. 2021, doi: 10.22331/q-2021-02-04-392.

[38] S. Bartolucci *et al.*, "Fusion-based quantum computation," 2021, *arXiv:2101.09310*, doi: 10.48550/arXiv.2101.09310.

[39] C. Chamberland *et al.*, "Building a fault-tolerant quantum computer using concatenated cat codes," *PRX Quantum*, vol. 3, 2022, Art. no. 010329, doi: 10.1103/PRXQuantum.3.010329.

[40] E. Knill, "Quantum computing with realistically noisy devices," *Nature*, vol. 434, pp. 39–44, 2005, doi: 10.1038/nature03350.

[41] S. Herbert and A. Sengupta, "Using reinforcement learning to find efficient qubit routing policies for deployment in near-term quantum computers," 2019, *arXiv:1812.11619*, doi: 10.48550/arXiv.1812.11619.

[42] M. G. Pozzi, S. J. Herbert, A. Sengupta, and R. D. Mullins, "Using reinforcement learning to perform qubit routing in quantum compilers," 2020, *arXiv:2007.15957*, doi: 10.48550/arXiv.2007.15957.

[43] V. V. Shende, S. S. Bullock, and I. L. Markov, "Synthesis of quantum-logic circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 25, no. 6, pp. 1000–1010, Jun. 2006, doi: 10.1109/TCAD.2005.855930.

[44] M. Saeedi and I. L. Markov, "Synthesis and optimization of reversible circuits—A survey," *ACM Comput. Surv.*, vol. 45 no. 2, Mar. 2013, Art. no. 21, doi: 10.1145/2431211.2431220.

[45] M. Saeedi, R. Wille, and R. Drechsler, "Synthesis of quantum circuits for linear nearest neighbor architectures," *Quantum Inf. Process*, vol. 10, pp. 355–377, 2011, doi: 10.1007/s11128-010-0201-2.

[46] P. Niemann, R. Wille, and R. Drechsler, "Efficient synthesis of quantum circuits implementing clifford group operations," in *Proc. 19th Asia South Pacific Des. Autom. Conf.*, 2014, pp. 483–488, doi: 10.1109/ASP-DAC.2014.6742938.

[47] P. Narang, W. Finigan, M. Cubeddu, and J. Flick, "Qubit allocation for noisy intermediate-scale quantum computers," 2020, *arXiv:1810.08291*, doi: 10.48550/arXiv.1810.08291.

[48] S. Niu, A. Suau, G. Staffelbach, and A. Todri-Sanial, "A hardware-aware heuristic for the qubit mapping problem in the NISQ era," *IEEE Trans. Quantum Eng.*, vol. 1, 2020, Art. no. 3101614, doi: 10.1109/TQE.2020.3026544.

[49] L. Lao, H. van Someren, I. Ashraf, and C. G. Almudever,, "Timing and resource-aware mapping of quantum circuits to superconducting processors," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 41, no. 2, pp. 359–371, Feb. 2022, doi: 10.1109/TCAD.2021.3057583.

[50] P. Murali, J. M. Baker, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, "Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2019, pp. 1015–1029, doi: 10.1145/3297858.3304075.

[51] D. Venturelli, M. Do, E. Rieffel, and J. Frank, "Compiling quantum circuits to realistic hardware architectures using temporal planners," *Quantum Sci. Technol.*, vol. 3, no. 2, Feb. 2018, Art. no. 025004, doi: 10.1088/2058-9565/aaa331.

[52] A. Cowtan, S. Dilkes, R. Duncan, A. Krajenbrink, W. Simmons, and S. Sivarajah, "On the qubit routing problem," in *Proc. 14th Conf. Theory Quantum Comput., Commun. Cryptography*, 2019, pp. 5:1–5:32, doi: 10.48550/arXiv.1902.08091.

[53] M. Y. Siraichi, V. Fernandes, C. S. dos Collange, and F. M. Quintão Pereira, "Qubit allocation," in *Proc. Int. Symp. Code Gener. Optim.*, Vienna, Austria, 2018, pp. 1–12, doi: 10.1145/3168822.

[54] M. Bandic, H. Zarein, E. Alarcon, and C. G. Almudever, "On structured design space exploration for mapping of quantum algorithms," in *Proc. 35th Conf. Des. Circuits Integr. Syst.*, 2020, pp. 1–6, doi: 10.1109/DCIS51330.2020.9268670.

[55] C. G. Almudever, L. Lao, R. Wille, and G. G. Guerreschi, "Realizing quantum algorithms on real quantum computing devices," in *Proc. Des., Autom. Test Europe Conf. Exhib.*, 2020, pp. 864–872, doi: 10.23919/DATE48585.2020.9116240.

[56] D. Simmons, J. Coon, and A. Datta, "The quantum Theil index: Characterizing graph centralization using von Neumann entropy," *arXiv: 1707.07906*, doi: 10.48550/arXiv.1707.07906.

[57] M. E. J. Newman. *Networks: An Introduction*, 2nd ed. London, U.K.: Oxford Univ. Press, 2010.

[58] H. Abraham *et al.*, "Qiskit: An open-source framework for quantum computing," 2019. [Online]. Available: https://qiskit.org/

[59] J. L. Gross, J. L. Gross, T. W. Tucker, and B. G. Osgood, *Topological Graph Theory*. Hoboken, NJ, USA: Wiley, 1987.

[60] P. Murali, D. C. Mckay, M. Martonosi, and A. Javadi-Abhari, "Software mitigation of crosstalk on noisy intermediate-scale quantum computers," in *Proc. 25th Int. Conf. Architectural Support Program. Lang. Oper. Syst.*, 2020, doi: 10.1145/3373376.3378477.

[61] B. Tan and J. Cong, "Optimality study of existing quantum computing layout synthesis tools," *IEEE Trans. Comput.*, vol. 70, no. 9, pp. 1363–1373, Sep. 2020, doi: 10.1109/TC.2020.3009140.

[62] A. Zulehner, A. Paler, and R. Wille, "An efficient methodology for mapping quantum circuits to the IBM QX architectures," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 38, no. 7, pp. 1226–1236, Jul. 2019, doi: 10.1109/TCAD.2018.2846658.

[63] R. J. Wilson. *Introduction to Graph Theory*. Hoboken, NJ, USA: Wiley, 1986.

[64] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1982.

[65] D. Ferrari, I. Tavernelli, and M. Amoretti, "Deterministic algorithms for compiling quantum circuits with recurrent patterns," *Quantum Inf. Process.*, Feb. 2021, Art. no. 213, doi: 10.1007/s11128-021-03150-9.

[66] Ian D. Kivlichan *et al.*, "Quantum simulation of electronic structure with linear depth and connectivity," *Phys. Rev. Lett.*, vol. 120, no. 11, Mar. 2018, Art. no. 110501, doi: 10.1103/PhysRevLett.120.110501.

[67] B. O' William, J. Gorman, E. G. H. Rieffel, and K. B. Whaley, "Generalized swap networks for near-term quantum computing," 2019, *arXiv:1905.05118*, doi: 10.48550/arXiv.1905.05118.

[68] A. Hagberg, P. Swart, and D. Chult, "Exploring network structure, dynamics, and function using NetworkX," U.S. Dept. of Energy Office Sci. Tech. Inf., Washington, DC, USA, Jan. 2008.

[69] N. Khammassi, G. Guerreschi, I. Ashraf, J. Hogaboam, C. G. Almudéver, and K. Bertels, "cQASM v1.0: Towards a common quantum assembly language," 2018, *arXiv:1805.09607*, doi: 10.48550/arXiv.1805.09607.

[70] H. T. Charles, E. Cormen, R. L. L. Rivest, and C. Stein. *Introduction to Algorithms*. 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[71] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[72] S. Nishio, Y. Pan, T. Satoh, H. Amano, and R. Van Meter, "Extracting success from IBM's 20-qubit machines using error-aware compilation," *J. Emerg. Technol. Comput. Syst.*, vol. 16 no. 3, May 2020, doi: 10.1145/3386162.

[73] A. M. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*, 10th ed. Cambridge, U.K.: Cambridge Univ. Press, 2010.

[74] U. Knauer and K. Knauer. *Algebraic Graph Theory*. Berlin, Germany: De Gruyter, 2019.

[75] R. Merris, "Laplacian matrices of graphs: A survey," *Linear Algebra Appl.*, vol. 197/198, pp. 143–176, 1994, doi: 10.1016/0024-3795(94)90486-3.

[76] D. Hangleiter, "Crosstalk diagnosis for the next generation of quantum processors," *Quantum Views*, vol. 4, pp. 46, Oct. 2020, doi: 10.22331/qv-2020-10-29-46.

[77] R. Wille, O. Keszocze, M. Walter, P. Rohrs, A. Chattopadhyay, and R. Drechsler, "Look-ahead schemes for nearest neighbor optimization of 1 D and 2 D quantum circuits," in *Proc. 21st Asia South Pacific Des. Autom. Conf.*, 2016, pp. 292–297, doi: 10.1109/ASPDAC.2016.7428026.

[78] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for NISQ-Era quantum devices," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Oper. Syst. (ASPLOS '19)*, 2019, pp. 1001–1014, doi: 10.1145/3297858.3304023

[79] J. X. Lin, E. R. Anschuetz, and A. W. Harrow, "Using spectral graph theory to map qubits onto connectivity-limited devices," *ACM Trans. Quantum Comput.*, vol. 2, no. 1, Apr. 2021, Art. no. 3, doi: 10.1145/3436752.

[80] R. Wille, D. Große, L. Teuber, Gerhard W. Dueck, and R. Drechsler, "Revlib: An online resource for reversible functions and reversible circuits," in *Proc. 38th Int. Symp. Mult. Valued Log.*, 2008, pp. 220–225, doi: 10.1109/ISMVL.2008.43.

[81] B. Tan and J. Cong, "Optimality study of existing quantum computing layout synthesis tools," *IEEE Trans. Comput.*, vol. 70, no. 9, pp. 1363–1373, Sep. 2021, doi: 10.1109/TC.2020.3009140.