Symmetry-based quantum circuit mapping

Di Yu[1,†] and Kun Fang[2,*,†]

[1] *Department of Electrical and Electronic Engineering, The University of Hong Kong, 999077 Hong Kong, China*
[2] *School of Data Science, The Chinese University of Hong Kong, Shenzhen, Shenzhen, 518172 Guangdong, China*

(Received 6 November 2023; revised 22 April 2024; accepted 23 July 2024; published 9 August 2024)

Quantum circuit mapping is a crucial process in the quantum circuit compilation pipeline, facilitating the transformation of a logical quantum circuit into a list of instructions directly executable on a target quantum system. Recent research has introduced a postcompilation step known as remapping, which seeks to reconfigure the initial circuit mapping to mitigate quantum circuit errors arising from system variability. As quantum processors continue to scale in size, the efficiency of quantum circuit mapping and the overall compilation process has become of paramount importance. In this work, we introduce a quantum circuit remapping algorithm that leverages the intrinsic symmetries in quantum processors, making it well suited for large-scale quantum systems. This algorithm identifies all topologically equivalent circuit mappings by constraining the search space using symmetries and accelerates the scoring of each mapping using vector computation. Notably, this symmetry-based-circuit-remapping algorithm exhibits linear scaling with the number of qubits in the target quantum hardware and is proven to be optimal in terms of its time complexity. Moreover, we conduct a comparative analysis against existing methods in the literature, demonstrating the superior performance of our symmetry-based method on state-of-the-art quantum hardware architectures and highlighting the practical utility of our algorithm, particularly for large-scale quantum computing.

DOI: 10.1103/PhysRevApplied.22.024029

I. INTRODUCTION

Quantum computing has emerged as a promising avenue for solving intractable problems that are beyond the reach of classical computing, such as integer factoring [1], large database search [2], chemistry simulation [3], and machine learning [4]. To put the theoretically blueprinted quantum advantages into use, it is essential to execute them on real-world quantum computers, moving beyond theoretical concepts and simulation environments. However, quantum algorithms are typically designed at the logical level, assuming ideal qubits and the ability to apply a universal set of quantum gates. These algorithms are not directly executable on quantum hardware devices, which comprise qubits with limited coherence time and support only a specific set of native operations. This disparity underscores the necessity of quantum circuit compilation, which translates high-level quantum algorithms into low-level quantum instructions compatible with the target systems.

Quantum circuit compilation involves various tasks, including gate decomposition into the native operations of the quantum device, adaptation of operations to the hardware's topology, and optimization to reduce circuit depth. Of particular importance is quantum circuit mapping, which aligns the quantum circuit with the hardware architecture, ensuring that any two-qubit operation is applied to physically connected qubits on the device. This alignment is often achieved through the insertion of SWAP operations, allowing one to interchange the position of two logical qubits on the architecture. Chaining such operations permits arbitrary routing among remote qubits. To guarantee the reliable execution of the resulting circuit, it is imperative to minimize the overhead of SWAP operations. Previous research on the circuit-mapping problem predominantly focused on gate-optimal solutions, with the aim of minimizing the number of inserted SWAP gates [5–10]. Other research has concentrated on time-optimal circuit mapping, which seeks to minimize the depth of the entire transformed circuit [11–13].

SWAP gates align quantum circuits with topologies of target quantum hardware but introduce errors. The MAPOMATIC algorithm, introduced in Ref. [14], addresses this by reconfiguring initial circuit mappings to increase fidelity. While effective in mitigating errors, it requires the solving of a nondeterministic-polynomial-time-complete subgraph-matching problem [15]. MAPOMATIC uses standard routines such as VF2 and VF2++ to solve this subgraph-matching problem [16,17], but their polynomial-time complexity may pose scalability issues as quantum

[*] Contact author: kunfang00@outlook.com
[†] D.Y. and K.F. contributed equally to this work.

hardware advances [18,19]. With current quantum computers reaching hundreds of qubits [20] and projections of millions in the future [21], existing remapping algorithms may struggle even with small circuits on near-term processors.

Symmetries play a pivotal role in various applications within the field of quantum information science, including designing quantum machine-learning algorithms [22,23], evaluating quantum channel capacities [24–26], analyzing quantum entropies [27,28], and quantifying all kinds of quantum resources for quantum computation and quantum communication [29–38]. Nevertheless, the potential of harnessing hardware symmetries for quantum circuit compilation remains largely unexplored.

In this work, we address the scalability challenge of quantum circuit remapping by leveraging the inherent symmetries present in scalable quantum architectures. The main idea of this work is illustrated in Fig. 1. We consider a two-step workflow of quantum circuit compilation consisting of precompilation and remapping processes. In the precompilation phase, we determine an initial circuit mapping of the input quantum circuit [Fig. 1(a)]. This process generates a precompiled circuit [Fig. 1(b)] that implements the same unitary operation as the input circuit. The precompiled circuit has a logic qubit connectivity, i.e., interaction graph [Fig. 1(c)], that aligns with the physical connectivity of the target quantum processor [Fig. 1(d)]. Subsequently, during the remapping process, we identify all isomorphic subgraphs of the initial mapping and score the induced mappings using hardware calibration data. In particular, we introduce an algorithm that significantly reduces the search space for all topologically equivalent circuit mappings by leveraging hardware symmetries and that efficiently identifies the optimal circuit mapping through a vectorized scoring method. Notably, this symmetry-based-circuit-remapping algorithm exhibits linear scaling with the number of qubits in the target hardware and is proved to be optimal in terms of its time complexity. Moreover, we benchmark the runtime of our algorithm with three typical quantum processors of grid, octagonal, heavy-hex structures, respectively, demonstrating the superior efficiency of our algorithm over the

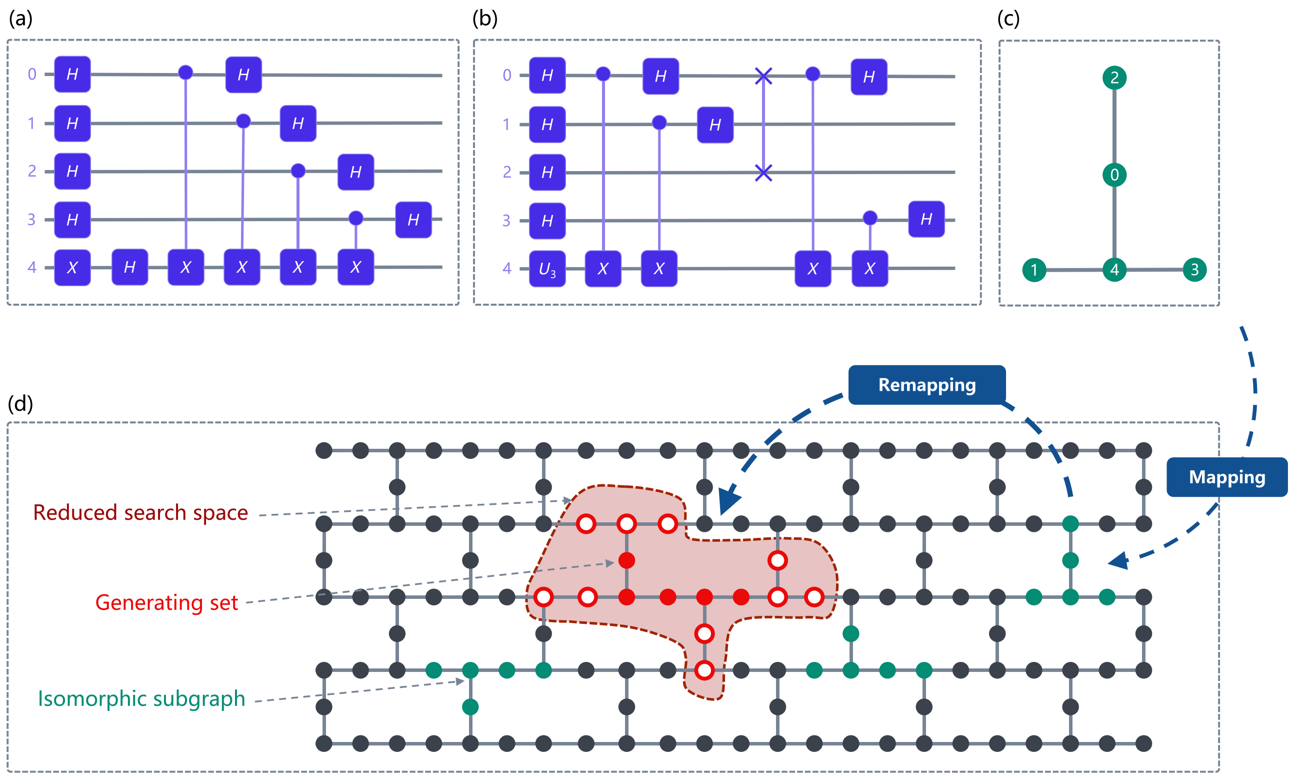

FIG. 1. Illustration of quantum circuit compilation with mapping and remapping processes. (a) The input quantum circuit. (b) The precompiled quantum circuit, assuming $\{U_3, \text{controlled } X, \text{SWAP}\}$ as the set of native operations in this example. (c) The interaction graph of the precompiled circuit. (d) A coupling graph with the heavy-hex structure. The mapping process finds a mapping from logical qubits in the interaction graph to physical qubits in the coupling graph. The remapping process finds a mapping from physical qubits to physical qubits such that the composed mapping-and-remapping process reduces the error of circuit implementation. The search for all isomorphic subgraphs used in the remapping process can be confined to the reduced search space, which is a neighborhood of the generating set, as depicted within the red area.

existing method in practical scenarios. To the best of our knowledge, this study marks the first instance of the utilization of the topological symmetries of quantum processors to address quantum circuit compilation challenges, potentially offering insights into other compilation processes. Moreover, our technical contribution is a purely mathematical result and holds promise for application in various other problem domains.

The remainder of this paper is structured as follows: In Sec. II we introduce the preliminaries to be used throughout this work. In Sec. III we present the symmetry-based-subgraph-matching (SBSM) algorithm and its underlying theorem, justifying the search for all topologically equivalent circuit mappings within a reduced search space. We also outline the vectorized scoring method that expedites the mapping process in practice. In Sec. IV we provide a comparative analysis of our algorithm against existing methods. Finally, Sec. V concludes the paper, and we propose potential directions for future research.

II. PRELIMINARIES

A. Notation

A graph is defined as an ordered pair of the vertex set and the edge set, denoted as $G = (V, E)$, where V [or $V(G)$] denotes the set of vertices and where E [or $E(G)$] represents the set of edges. Each edge is an unordered pair of vertices $e = \{u, v\}$ for some $u, v \in V$. The order of a graph is its number of vertices $|V|$. A graph G' is a subgraph of G, denoted as $G' \subseteq G$, if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. An induced subgraph of a graph is another graph, formed from a subset of the vertices of the graph and all of the edges (from the original graph) connecting pairs of vertices in that subset. The degree of a vertex of a graph is the number of edges that are connected to the vertex. A graph has a bounded degree if the maximum degree of any vertex in the graph is limited by a constant value that remains independent of the graph's order. A walk is a sequence of edges $(e_1, e_2, \dots, e_n)$ for which there is a sequence of vertices $(v_1, v_2, \dots, v_{n+1})$ such that $e_k = \{v_k, v_{k+1}\}$ for $k \in \{1, 2, \dots, n\}$. A path is a walk in which all edges and all vertices are distinct, and the length of the path is defined as the number of edges it contains. Two vertices u and v are connected if there exists a path from vertex u to vertex v in graph G. Two vertices are adjacent if they are connected by a path of length 1, i.e., by a single edge. A graph is connected if every pair of vertices in the graph is connected. The kth-order neighborhood of a vertex v in a graph G, denoted as $N_G^k(v)$, refers to the set of vertices that can be reached from v within k hops. In other words, $u \in N_G^k(v)$ if and only if $u = v$ or there exists a path connecting u and v with a length no greater than k. The kth-order neighborhood of a subset S, denoted as $N_G^k(S)$, is defined as the union of the kth-order neighborhoods of all vertices within the subset S.

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, these graphs are isomorphic, denoted as $G_1 \cong G_2$, if there exists a bijection between the vertex sets $f : V_1 \to V_2$ such that $\{u, v\} \in E_1$ if and only if $\{f(u), f(v)\} \in E_2$. This bijection f is referred to as an isomorphism from G_1 to G_2. In particular, an automorphism is an isomorphism from a graph to itself. Note that f is a mapping from vertices to vertices, but we can naturally extend it to a mapping that operates on both vertices and edges: $\tilde{f} : V_1 \cup E_1 \to V_2 \cup E_2$, defined by $\tilde{f}(v) = f(v)$ and $\tilde{f}(\{u, v\}) = \{f(u), f(v)\}$. Therefore, $\tilde{f}$ can be taken as a mapping from graph to graph and is referred to as a natural extension of f. In this work, we address the subgraph-matching problem, which involves finding all subgraphs in a target graph that are isomorphic to a given pattern graph. This problem is nondeterministic polynomial time complete in general [15,39].

B. Symmetry of graph

In general, the symmetry of an object is associated with a transformation that preserves certain structural aspects of it. In graph theory, the automorphisms of a graph play the role of symmetry transformations. We denote the set of all automorphisms of a graph G as $\mathrm{Sym}(G)$. This set forms a group under the composition of operators, which we refer to as the symmetry group of G. Let H be a subgroup of $\mathrm{Sym}(G)$. For a fixed vertex x in G, the orbit of x under H is defined as $\mathrm{orb}_H(x) = \{y \in V(G) : y = gx \text{ for some } g \in H\}$. In particular, we refer to the orbit of x with respect to a symmetry group $H = \{f^i : i \in \mathbb{Z}\}$ as the orbit of x with respect to automorphism f, denoted as $\mathrm{orb}_f(x)$. It is easy to show that all orbits within G form a partition of the vertices, and two vertices u and v in $V(G)$ belong to the same orbit if and only if there exists an automorphism f of graph G such that $f(u) = v$. The orbit of a set of vertices $V_0 \subseteq V(G)$, denoted as $\mathrm{orb}_f(V_0)$, is defined by the union of the orbits $\mathrm{orb}_f(x)$ for all $x \in V_0$. A generating set of a graph $G = (V, E)$ with respect to the automorphism f is a subset of vertices $V_0 \subseteq V$ such that the orbit of V_0 with respect to f covers all the vertices in G, i.e., $\mathrm{orb}_f(V_0) = V$. Or equivalently, a subset of vertices $V_0 \subseteq V$ is called a generating set of a graph $G = (V, E)$ with respect to an automorphism f if $\{f^i(v) | i \in \mathbb{Z}, v \in V_0\} = V$. Similarly, the generating set of G under a group of automorphisms F is a subset of vertices $V_0' \subseteq V$ such that $\mathrm{orb}_F(V_0') = V$. For an infinite square-grid graph G_{grid}, for instance, this graph has two independent automorphisms—one automorphism f_x maps each node to the right nearest node, while the other one f_y maps each node to the upper nearest node. These two automorphisms are commuting with each other and generate a group of automorphisms $F_{\mathrm{grid}} = \{f_x^i f_y^j | i, j \in \mathbb{Z}\}$. Each single vertex set $\{v\}$ with $v \in V(G_{\mathrm{grid}})$ constitutes a generating set of G_{grid} with respect to F_{grid}.

C. Graph distance

The distance between two vertices v and u within a graph, denoted as $d(v,u)$, corresponds to the shortest path connecting them, measured in terms of the number of edges. In cases where no such path exists, the distance is conventionally regarded as infinite. The eccentricity $\epsilon(v)$ for a given vertex v within a graph $G=(V,E)$ is defined as the greatest distance between vertex v and any other vertex, i.e., $\epsilon(v)=\max_{u\in V} d(v,u)$. This value tells how distant a node is from the farthest node in the entire graph. The radius of a graph $G=(V,E)$, denoted as $r(G)$, represents the minimum eccentricity among all vertices, i.e., $r(G)=\min_{v\in V}\epsilon(v)$. Then the minimizer is called a central vertex of the graph.

D. Quantum circuit mapping

The interaction graph of a quantum circuit serves as a representation of the qubits and the required interactions between them. Each node in the interaction graph corresponds to a logical qubit in the quantum circuit, and an edge connects two qubits if there is a two-qubit gate in the circuit acting on both of those qubits. However, existing quantum computers typically provide a native gate set that includes a family of single-qubit gates along with some two-qubit gates. The interactions between qubits in a quantum processor are also constrained by the connectivity of its architecture, which can be expressed through the coupling graph, denoted as $G=(V,E)$. In this graph, V represents the set of physical qubits, and an edge $\{u,v\}\in E$ signifies that a two-qubit gate can be directly executed between physical qubits u and v.

To execute a quantum circuit on a target quantum device, the set of logical qubits, denoted as Q_l, in the interaction graph of the circuit must first be mapped to the physical qubits, represented as Q_p, according to the coupling graph of the quantum processor. This mapping involves establishing an injective function $g: Q_l \to Q_p$, meaning that each logical qubit is uniquely assigned to a specific physical qubit. The purpose of circuit mapping is to ensure that the two-qubit gates in the circuit can be executed with use of the available physical qubits in accordance with the constraints defined by the coupling graph of the quantum device. In graph theory, this is the same as finding a subgraph of the coupling graph that is isomorphic to the interaction graph of the circuit.

Quantum circuit remapping is introduced as a subsequent step that follows circuit mapping and entails the mapping of $h: Q_p \to Q_p$. Therefore, the combined effect of the circuit mapping and remapping, denoted as $h\circ g$, is a mapping from logical qubits to physical qubits. This mapping aims to reduce errors during the implementation of the quantum circuit. The most-crucial step in the remapping process is to identify all subgraphs in the coupling graph that are isomorphic to the interaction graph of the circuit, essentially solving a subgraph-matching problem.

III. SYMMETRY-BASED QUANTUM CIRCUIT MAPPING

In this section, we introduce a subgraph-matching algorithm that leverages the topological symmetry of quantum processors. We also present a circuit-mapping-scoring algorithm that uses vector computation. These routines are then integrated into a compilation pipeline, forming a complete quantum circuit mapping algorithm. We then provide a complexity analysis of our algorithm, highlighting its optimality with respect to time complexity.

A. Symmetry-based subgraph matching

The idea behind our subgraph-matching algorithm, as depicted in Fig. 1, is to narrow down the search space for isomorphic graphs by focusing on a neighborhood of the generating set within the target graph. This approach is supported by Theorem 1, which indicates that all isomorphic graphs can be obtained by one applying the corresponding symmetry transformations to certain subgraphs within the reduced space.

Theorem 1. Let T be a graph and S be a generating set of T with respect to the automorphism f. Let G be a subgraph of T with radius r. Then for any subgraph G'' of T that is isomorphic to G, there exists an integer n and subgraph $G'\subseteq T$ such that $G''=\tilde{f}^{\,n}(G')$, with $G'\cong G$, $V(G')\subseteq N_T^r(S)$, and $\tilde{f}$ being the natural extension of f.

Here $N_T^r(S)$ is the rth-order neighborhood of the generating set S within the graph T, f is a mapping from vertex to vertex, and $\tilde{f}$ is a mapping from vertex (or edge) to vertex (or edge). The proof of this result requires the following lemmas.

Lemma 1. Let T be a graph with an automorphism f. Let u and v be two vertices in T. Then there exists a path connecting u and v of length l if and only if $f^n(u)$ and $f^n(v)$ are connected by a path of length l for any integer n.

Proof. Suppose that $(w_0,w_1,\dots,w_l)$ is a path connecting vertices $w_0=u$ and $w_l=v$. Then $\{w_k,w_{k+1}\}$ is an edge in graph T for any $k\in\{0,\dots,l-1\}$. Since f is an automorphism, we know that $\{f(w_k),f(w_{k+1})\}$ is also an edge in T. This implies that $(f(w_0),f(w_1),\dots,f(w_l))$ is a path in T, which connects $f(u)$ and $f(v)$ and has a length of l. By repeating this process, we can demonstrate that for all positive integers n, $f^n(u)$ and $f^n(v)$ are connected by a path of length l. Furthermore, since f^{-1} is also an automorphism of T, we can extend the range of n to all integers and conclude the proof. ■

The next lemma suggests that the operation of taking a neighborhood of a vertex set commutes with an automorphism.

Lemma 2. Let T be a graph with an automorphism f, and let $S \subseteq V(T)$ be a subset of vertices. Then for any integer n and positive integer m, $N_T^m(f^n(S)) = f^n(N_T^m(S))$.

Proof. For any $x \in N_T^m(f^n(S))$, there exists $y \in f^n(S)$ such that x and y are connected by a path of length l in T with $l \leq m$. Since $y \in f^n(S)$, there exists $z \in S$ such that $y = f^n(z)$. By Lemma 1, we know that there exists a path of length l connecting $f^{-n}(x)$ and $f^{-n}(y)$. Note that $f^{-n}(y) = f^{-n}(f^n(z)) = z \in S$. This implies $f^{-n}(x) \in N_T^m(S)$. Hence, there exists s in $N_T^m(S)$ such that $f^{-n}(x) = s$, or equivalently $x = f^n(s)$. This gives $x \in f^n(N_T^m(S))$ and concludes that $N_T^m(f^n(S)) \subseteq f^n(N_T^m(S))$. Now we prove the other direction. Consider any $x \in f^n(N_T^m(S))$. There exists s in $N_T^m(S)$ such that $x = f^n(s)$. Since s is in $N_T^m(S)$, we know that there exists y in S and a path of length l connecting s and y with $l \leq m$. By Lemma 1, we know that there exists a path of length l connecting $x = f^n(s)$ and $f^n(y) \in f^n(S)$. This implies $x \in N_T^m(f^n(S))$ and concludes that $f^n(N_T^m(S)) \subseteq N_T^m(f^n(S))$. ■

We are now ready to prove the main theorem.

Proof of Theorem 1. Since G'' is isomorphic to G, they have the same radius r. Let v be a central vertex of $V(G'')$. By the definition of central vertices, all vertices of G'' must be reachable from v within r hops. That is, G'' is contained by the rth-order neighborhood of v, namely, $V(G'') \subseteq N_T^r(v)$. According to the definition of generating set, there exists an integer n such that $v \in f^n(S)$. Since $v \in f^n(S)$ and $V(G'') \subseteq N_T^r(v)$, we then get $V(G'') \subseteq N_T^r(f^n(S))$. By Lemma 2, we have $N_T^r(f^n(S)) = f^n(N_T^r(S))$. This gives $V(G'') \subseteq f^n(N_T^r(S))$. Now let us choose $G' = \tilde{f}^{-n}(G'')$. We have $V(G') = V(\tilde{f}^{-n}(G'')) = f^{-n}(V(G'')) \subseteq f^{-n}(f^n(N_T^r(S))) = N_T^r(S)$, where the second equality follows by the definition of the natural extension of an automorphism. Since $\tilde{f}$ maps graphs to isomorphic graphs, we have $G' = \tilde{f}^{-n}(G'') \cong G''$. Note that $G'' \cong G$, and we get $G' \cong G$. This concludes the proof. ■

The automorphism in Theorem 1 can be replaced with a group of automorphisms. This result is formally presented in Corollary 1 and applies to various practical scenarios where a graph has multiple independent symmetry transformations. For example, a two-dimensional grid possesses two independent symmetry transformations: translation in both the horizontal direction and the vertical direction.

Corollary 1. Let T be a graph and S be a generating set of T with respect to a group of automorphisms F. Let G be a subgraph of T with radius r. Then for any subgraph G'' of T that is isomorphic to G, there exists an automorphisms $f \in F$ and subgraph $G' \subseteq T$ such that $G'' = \tilde{f}(G')$, with $G' \cong G$, $V(G') \subseteq N_T^r(S)$, and $\tilde{f}$ being the natural extension of f.

ALGORITHM 1. Symmetry-based subgraph matching

```
Input  : A pattern graph $G$, a target graph $T$, a group of
         automorphisms $F$ of the target graph and the
         associated generating set $S$.
Output: All subgraph isomorphisms of $G$ in $T$.

1  Let $r$ be the radius of the pattern graph $G$;
2  Let $N_T^r(S)$ be the $r$-th order neighborhood of $S$;
3  Let $R$ be the induced subgraph of $N_T^r(S)$ in $T$;
4  Obtain the set of all isomorphic graphs of $G$ within $R$ and
     denote this set as $H_0$;
5  Let $H$ be an empty list;
6  for $G' \in H_0$ do
7      Let $M = \{\tilde{f}(G') : f \in F, \tilde{f}(G') \subseteq T\}$;
8      Append $M$ to $H$;
9  end
10 Return $H$;
```

Theorem 1 and Corollary 1 indicate that to locate all the subgraphs in T that are isomorphic to G, one needs to search over only the subgraph induced by the vertex set $N_T^r(S)$. In light of this reduction, we propose the SBSM algorithm (Algorithm 1), enabling efficient searching for all circuit mappings given a coupling graph.

The SBSM algorithm consists of three major steps. First, the algorithm identifies a reduced search space for initial subgraph matching. To this end, the algorithm calculates the radius of G, denoted as r, and computes the rth-order neighborhood of G, serving as the reduced search space. Second, it uses a standard subgraph-matching algorithm, such as VF2 [16], to search for isomorphic graphs within the reduced search space. Lastly, the searched-for patterns are transformed with use of the symmetry transformations to get all the isomorphisms.

As an explicit example, Algorithm 1 can be illustrated by one considering the problem of finding all subgraph isomorphisms of the interaction graph G [Fig. 1(c)] in the hardware coupling graph T [Fig. 1(d)]. In graph G, nodes 0 and 4 are central vertices, each with an eccentricity of 2, indicating that the radius of G is 2. The solid red points in Fig. 1(d) represent a generating set S of the hardware coupling graph, and the subset of vertices enclosed by the dashed red line represents its neighborhood $N_T^2(S)$. The subgraph of T induced by the vertices in $N_T^2(S)$, denoted by R, serves as the reduced space for the SBSM algorithm. The algorithm searches for isomorphisms of G within R. According to Corollary 1, all subgraph isomorphisms of G in T are related to those found in R through an automorphism of the coupling graph T. Traversing through the automorphisms via a for-loop gives all the isomorphisms we need.

Note that the coupling graph of a quantum processor may not inherently possess symmetries, particularly along its boundary. Nevertheless, we can effectively apply the SBSM algorithm by embedding the coupling graph within a larger graph that does possess symmetries. It is also worth mentioning that the SBSM algorithm can accommodate various types of symmetry, which may include, but are not limited to, translational, mirror, and rotational symmetries. For the numerical experiments discussed in the following sections, we primarily focus on translational symmetries for simplicity.

B. Circuit-mapping scoring through vectorization

Quantum circuit remapping involves the assessment of all topologically equivalent circuit mappings. The existing approach in QISKIT accomplishes this by using a for-loop to estimate the overall circuit fidelity for each mapping one by one [40]. However, the for-loop implementation is time consuming, particularly when one is dealing with a large number of mappings. To mitigate the computational cost of scoring these mappings, we use the technique of vectorization, which transforms the evaluation process into vector computations. As we demonstrate later, the use of vectorization substantially speeds up the circuit-evaluation process, which would otherwise be dominant in the runtime of our remapping algorithm.

Our circuit-mapping-scoring algorithm is presented in Algorithm 2, where we emphasize the vectors in bold symbols. In this algorithm, we denote the array of circuit mappings and the error map by $\boldsymbol{L}$ and E, respectively. Specifically, $\boldsymbol{L}$ is a mapping from a logical gate to an array of physical gates. E is a function that takes these physical gates as input and returns their error rates. The initial phase of the circuit-mapping-scoring process involves the transformation of the precompiled circuit into a sequence of individual logical gates, represented as $\boldsymbol{G}$. For each logical

ALGORITHM 2. Circuit-mapping scoring through vectorization

```
Input  : An array of circuit mappings $\boldsymbol{L}$, a logical
         quantum circuit $C$, and an error map $E$.
Output: The estimated fidelity of the quantum circuit
         under the circuit mappings.
1 Convert $C$ to a list of gates $\boldsymbol{G}$;
2 Initiate $\boldsymbol{S}$ to be a vector of ones $\mathbf{1}$;
3 for $g$ in $\boldsymbol{G}$ do
4     Compute $\boldsymbol{L}' = \boldsymbol{L}(g)$;
5     Compute $\boldsymbol{E}' = E(\boldsymbol{L}')$;
6     Update $\boldsymbol{S}$ as $\boldsymbol{S} \odot (\mathbf{1} - \boldsymbol{E}')$;
7 end
8 Return $\boldsymbol{S}$;
```

gate g within $\boldsymbol{G}$, we access the error-rate vector of the corresponding physical gates from the hardware calibration data, $\boldsymbol{E}' = E(\boldsymbol{L}(g))$. Note that $\boldsymbol{E}'$ encapsulates the error rates associated with the specific gate G' across all circuit mappings. Subsequently, by harnessing the elementwise vector multiplication supported by the NumPy library [41], we perform vector computations to evaluate the circuit-scoring vector $\boldsymbol{S}$, recording the estimated fidelity values of all circuit mappings. That is, we perform elementwise multiplication, denoted by $\odot$, iteratively on $\mathbf{1} - \boldsymbol{E}'$ for all g, where $\mathbf{1}$ is the vector of 1's. The resulting $\boldsymbol{S}$ is used to guide the selection of the optimal mapping that maximizes the overall circuit fidelity.

Consider an example where the input logical quantum circuit C consists of only two gates: a single-qubit gate g_1 and a two-qubit gate g_2. Algorithm 2 begins by determining the indices of the physical gates that correspond to g_1, which are denoted as $\boldsymbol{L}_1' = \boldsymbol{L}(g_1)$, on the basis of the circuit mappings provided . It then retrieves the error rates of these physical gates and stores them in a NumPy array $\boldsymbol{E}_1' = E(\boldsymbol{L}_1')$. Next, the algorithm applies the same process to g_2, obtaining a NumPy array $\boldsymbol{E}_2'$ that contains the error rates of the corresponding two-qubit physical gates. Finally, to estimate the circuit fidelity, the algorithm calculates the elementwise product of $(\mathbf{1} - \boldsymbol{E}_1')$ and $(\mathbf{1} - \boldsymbol{E}_2')$, where $\mathbf{1}$ represents a NumPy array of 1's with the same shape as $\boldsymbol{E}_1'$ and $\boldsymbol{E}_2'$, respectively. The resulting product serves as an estimate of the circuit fidelity and is returned as the output of the algorithm.

C. Symmetry-based quantum circuit mapping

The SBSM algorithm and the circuit-mapping-scoring algorithm can work together to facilitate efficient quantum circuit remapping. Now we integrate these steps into a complete quantum circuit mapping algorithm, as presented in Algorithm 3. This symmetry-based-circuit-mapping (SBCM) algorithm comprises several main steps. First, the algorithm uses existing methods, such as those supported by QISKIT, to compile the quantum circuit, giving the interaction graph G and an initial circuit mapping L_{pre}. Note that this precompilation should be chosen as independent of the order of T. Second, the SBSM algorithm is used to search for isomorphic subgraphs of G within the hardware coupling graph T. These discovered subgraphs are recorded in the vector $\boldsymbol{L}$. Note that each isomorphic subgraph is associated with a circuit mapping from the precompiled circuit onto the given quantum hardware. Third, the algorithm uses vectorized computation, as stated in Algorithm 2, to score all circuit mappings. Finally, the circuit mappings are sorted on the basis of their scores (the estimated circuit fidelity), and the circuit mapping that maximizes the scores is selected and returned.

ALGORITHM 3. Symmetry-based circuit mapping

```
Input  : A logical quantum circuit $C$, a coupling graph
         $T$ of the target quantum processor, a group of
         automorphisms $F$ of the coupling graph and the
         associated generating set $S$, and an error map $E$
         of the quantum processor.
Output: A compiled quantum circuit and its circuit
         mapping.
1 Use $C, T, E$ as inputs to precompile the circuit for
    selected physical qubits and obtain a precompiled circuit
    $C'$, its interaction graph $G$ and an initial mapping $L_{\text{pre}}$;
2 Use $T, F, S, G$ as the inputs of Algorithm 1 to get all
    isomorphic subgraphs of $G$ in $T$ and record them in $\boldsymbol{L}$;
3 Compute the element-wise composition $\tilde{\boldsymbol{L}} = \boldsymbol{L} \circ L_{\text{pre}}$;
4 Use $\tilde{\boldsymbol{L}}, C', E$ as the inputs of Algorithm 2 to score the
    circuit mappings and record the mapping-and-score pair
    in $(\tilde{\boldsymbol{L}}, \boldsymbol{S})$;
5 Return the compiled circuit $C'$ and the circuit mapping
    with the highest score;
```

D. Computational complexity

We conduct a computational-complexity analysis for our symmetry-based algorithms, with a particular focus on scalable quantum processors that feature sparse coupling graphs with bounded degrees. That is, the maximum degree of any vertex in the graph is constrained by a constant value that remains independent of the graph's order. This ensures that the size of the reduced search space remains constant. We are aware that this constraint may not hold for trapped-ion quantum systems whose coupling graph is complete. Nevertheless, such systems inherently exhibit scalability issues, and we believe that the bounded-degree assumption is reasonable for all scalable quantum processors, especially in the context of superconducting quantum systems. Our complexity analysis focuses on the scaling of the coupling graph, while we consider the size of the circuit to be compiled as fixed.

The following result shows that the SBSM algorithm exhibits a time complexity of $O(n)$, which stands in stark contrast to VF2, a widely used subgraph-matching algorithm with a time complexity of $O(n^2)$ in the best-case scenario and a much less efficient time complexity of $O(n!n)$ in the worst-case scenario [16].

Theorem 2. Let T be a target graph with order n that has a bounded degree and is characterized by a group of automorphisms F along with a constant-size generating set S. Then the SBSM algorithm has a time complexity of $O(n)$, which is optimal for the subgraph-matching problem.

Proof. The subgraph-matching routine consists of three steps. First, we determine the reduced search space $N_T^r(S)$, which is the rth-order neighborhood of a generating set S, where r is the radius of the pattern graph. Calculation of the graph radius is efficiently performed with use of a breadth-first search algorithm, with a complexity that is independent of the order of the target graph [42]. Similarly, as the cardinality of the generating set is constant, identification of its rth-order neighborhood requires at most $O(n)$ steps. Second, subgraph matching is conducted within this reduced search space, and we consider using the VF2 algorithm, known for its efficiency in subgraph matching and widely used in applications such as graph isomorphism tests within NetworkX [43]. Since the target graph has a bounded degree, the size of the reduced search space is independent of n. In this case, the time complexity of subgraph matching using VF2 in the reduced space remains constant [16]. Finally, symmetry transformations are applied to obtain all isomorphic subgraphs. By Lemma 3, there are at most $O(n)$ subgraphs isomorphic to a connected pattern graph in T. So finding all isomorphic subgraphs requires only a number of steps linear in n. As a result, the SBSM algorithm has an overall time complexity of $O(n)$. Note that the subgraph-matching problem entails finding all isomorphic subgraphs in the target graph, necessitating a minimum of visiting each vertex at least once, resulting in a lower bound of $\Omega(n)$ steps in total. Our SBSM algorithm meets this trivial lower bound, thus concluding the proof of its optimality. ■

Lemma 3. Let T be a target graph with order n that has a bounded degree of d. Let G be a connected pattern graph with order m. Then there are at most $O(n)$ subgraphs in the target graph T that are isomorphic to G.

Proof. A subgraph isomorphism $f: G \to T$ embeds the vertex set $V(G)$ into $V(T)$. Let v_1 be a vertex in G. There are n possible values for $f(v_1)$, corresponding to all vertices in T. Once $f(v_1)$ is fixed, we choose a vertex $v_2 \in V(G)\backslash\{v_1\}$ that is adjacent to v_1. The existence of v_2 is ensured by the connectivity of G. Since f is a subgraph isomorphism from G to T, $f(v_2)$ must be a vertex in T adjacent to $f(v_1)$. As T has a bounded degree of d, each vertex in T has at most d adjacent vertices. Therefore, there are at most d possible values for $f(v_2)$. Continuing this process, we consider $f(v_3)$, where $v_3 \in V(G)\backslash\{v_1, v_2\}$ and is adjacent to at least one node in $\{v_1, v_2\}$. Without loss of generality, we assume that v_2 is adjacent to v_3 for simplicity. Then $f(v_3)$ must be a vertex in T adjacent to $f(v_2)$ because f is a subgraph isomorphism. As T has a bounded degree of d, there are at most d possible values for $f(v_3)$. By interating this procedure, we can traverse all the vertices in G and find their images under f, concluding that there are at most nd^{m-1} ways to match vertices that ensure f is a subgraph isomorphism. In other words, the total number of subgraph isomorphisms from G to T is at most nd^{m-1} with the dominant factor $O(n)$. ■

Theorem 2 demonstrates that our SBSM algorithm exhibits an optimal time complexity for solving the subgraph-matching problem with symmetries, making it an ideal choice for the remapping step. This optimality extends to quantum circuit mapping consisting of the standard mapping and the incorporation of an additional remapping process, as exemplified in Algorithm 3. This is because the initial mapping is independent of the size of the target quantum processor, and the remapping process dominates the overall circuit mapping process in the runtime. This optimality is formally presented in Corollary 2.

Corollary 2. Consider a quantum processor with a coupling graph of order n that has a bounded degree and is characterized by a group of automorphisms F along with a constant-size generating set S. Let C be a quantum circuit to be compiled. Then any quantum circuit mapping scheme that comprises circuit-mapping and circuit-remapping steps requires at least $\Omega(n)$ time. Moreover, the SBCM algorithm achieves this optimal time complexity.

Proof. Since the remapping process requires at least $\Omega(n)$ steps, the overall mapping has a time complexity no less than $\Omega(n)$. In terms of our SBCM algorithm, the precompilation is independent of the order of the coupling graph and it requires only $O(n)$ steps to find all isomorphic graphs. By Lemma 3, there are at most $O(n)$ subgraphs isomorphic to a connected pattern graph in T. This implies that the scoring process and the search for the mapping with the highest score each consume a maximum of $O(n)$ steps. Hence, the SBCM algorithm has time complexity of $O(n)$ and it achieves the lower bound. ■

IV. BENCHMARKING RESULTS

In this section, we report numerical experiments conducted to assess the performance of our algorithms and compare them with the existing algorithms such as VF2, VF2++, and MAPOMATIC. In particular, we consider quantum processors with grid, octagonal, and heavy-hex architectures, as depicted in Fig. 2, which are the state-of-the-art architectures used by Google [18], Rigetti [44], and IBM [45], respectively. We specifically consider layouts with equal rows and columns. The test circuit we use is the five-qubit Deutsch-Jozsa algorithm [46], which is the pioneering example of a quantum algorithm that outperforms classical counterparts and exemplifies the advantages of using quantum computing for specific problems. All of the numerical experiments were performed on a MacBook Pro computer (2020) with an Intel i5 processor and 8 GB of memory. The source codes are available on Ref. [47].

A. Benchmarking the subgraph-matching algorithms

We compared the runtimes of VF2, VF2++, and our SBSM algorithm for solving the subgraph-matching problem. The results are presented in Fig. 3, where the horizontal axis represents the size of the quantum processors and the vertical axis indicates the runtime for identifying all isomorphic graphs within the target graph. Notably, as the size of the hardware increases, the runtime of VF2 and VF2++ experiences significant growth due to the expansion of the search space for subgraph matching. In contrast, the SBSM algorithm's runtime exhibits a gradual increase with the number of qubits in the hardware. Specifically, for an 88 200-qubit octagonal-shaped quantum processor, the SBSM algorithm runs approximately 18 times faster than VF2 and 13 times faster than VF2++. These benchmarking results underscore a remarkable performance advantage of the SBSM algorithm over VF2 and VF2++, making it a more-suitable choice for the search of circuit mappings to large-scale quantum computers.

B. Benchmarking the circuit-mapping-scoring methods

We conducted benchmark tests comparing the runtimes of both the conventional for-loop method used by MAPOMATIC and our vectorized method for evaluating all

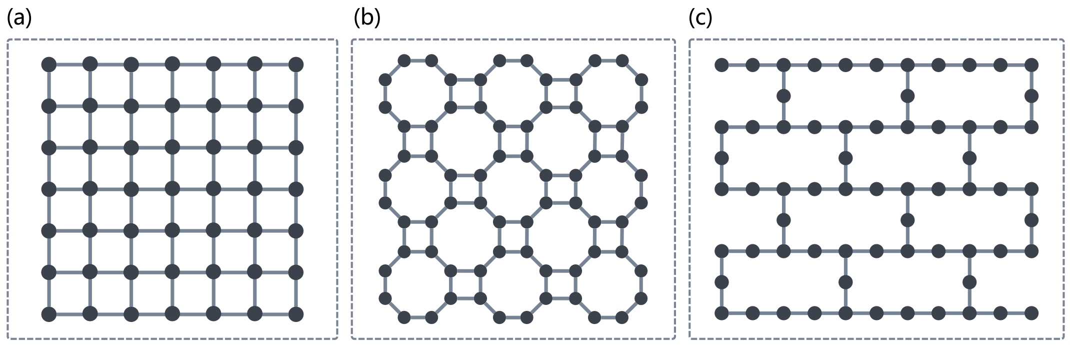

FIG. 2. Three typical quantum chip structures. Each node represents a physical qubit, and each edge corresponds to a coupler between two qubits. (a) 7×7 grid coupling graph with 49 qubits. (b) 3×3 octagonal coupling graph with 72 qubits. (c) 4×2 heavy-hex coupling graph with 67 qubits.

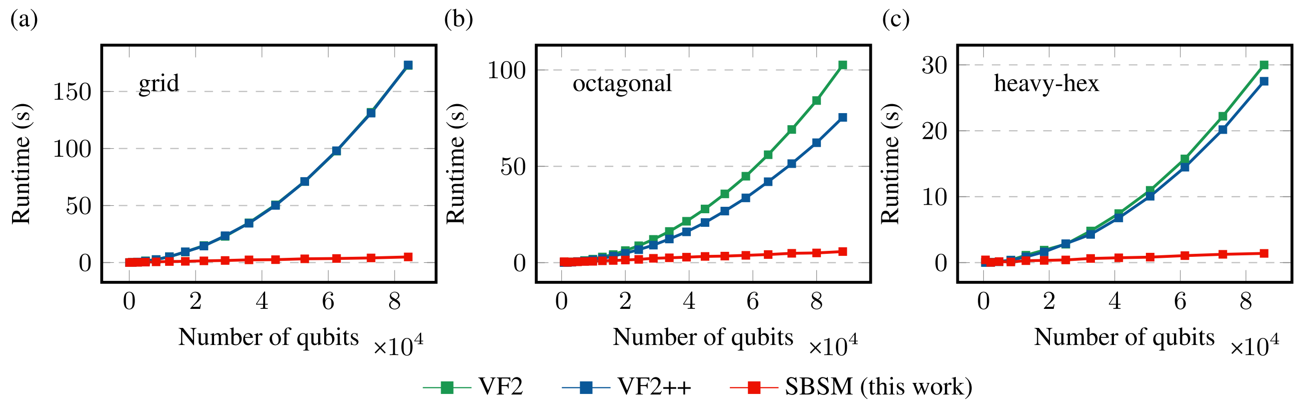

FIG. 3. Runtime of three subgraph-matching algorithms: VF2 (green) [16], VF2++ (blue) [17], and the SBSM algorithm (red). These experiments solved the subgraph-matching problem on three quantum hardware architectures. Note that the results for VF2 and VF2++ coincide in (a).

mappings of a five-qubit Deutsch-Jozsa algorithm on the three quantum hardware architectures. To account for the impact of imperfect fabrication and noisy environments, we assumed that the fidelity of quantum gates follows a two-dimensional spatial normal distribution, with the highest gate fidelity assumed to be at the center of the layout. The results of the numerical experiments are presented in Fig. 4, where the blue line represents the runtime of the scoring method used by MAPOMATIC and the red line shows the runtime of our vectorized scoring. Evidently, for all three hardware architectures, the vectorized method's runtime is substantially shorter than that of the conventional for-loop scoring method. Specifically, in the case of an octagonal-shaped quantum processor with 88 200 qubits, the vectorized method accomplishes this task approximately 36 times faster than the conventional scoring function, resulting in an impressive 97% reduction in runtime. This comparison illustrates the efficiency gains achieved through vectorization and the substantial time-saving potential in the circuit-mapping process.

C. Benchmarking the complete-circuit-mapping algorithms

We conducted benchmarking for the complete quantum circuit mapping problem, comparing the total runtime of our SBCM algorithm with that of MAPOMATIC, as they map the five-qubit Deutsch-Jozsa algorithm onto the three quantum hardware architectures. The results are depicted in Fig. 5, where the red line represents the runtime of the SBCM algorithm and the blue line represents that of MAPOMATIC. Notably, our SBCM algorithm outperforms MAPOMATIC for all three hardware architectures, with the runtime of the former being substantially shorter. For instance, when mapping the input circuit onto an 88 200-qubit octagonal-shaped quantum processor, MAPOMATIC took approximately 279 s to find the optimal circuit mapping, while our SBCM algorithm completed the same task in just 11 s. This represents an impressive 96% reduction in runtime. The comparison results clearly demonstrate the superiority of the SBCM algorithm over MAPOMATIC for mapping circuits onto

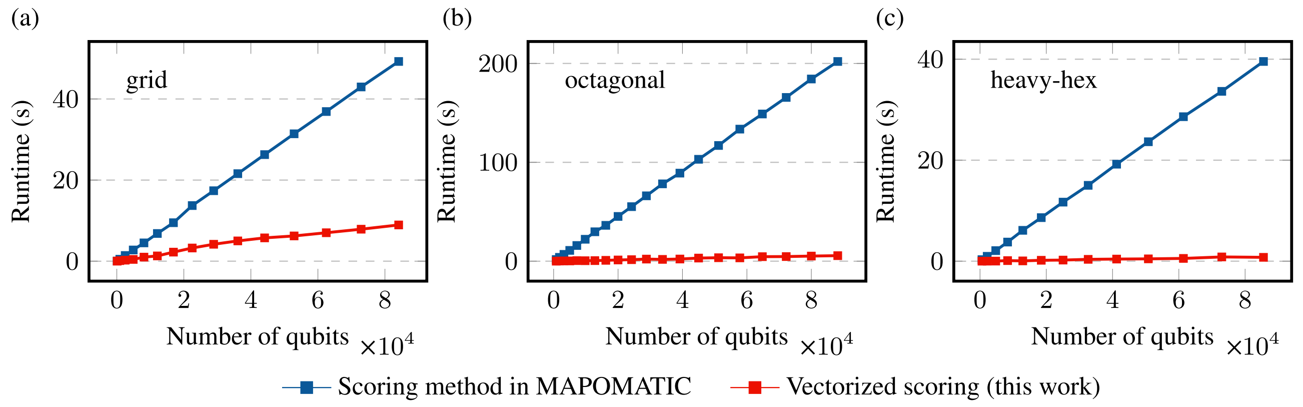

FIG. 4. Runtime for scoring circuit mappings with the conventional for-loop method in MAPOMATIC (blue) [14] and our vectorized approach (red). The experiments were conducted on three distinct quantum hardware architectures.

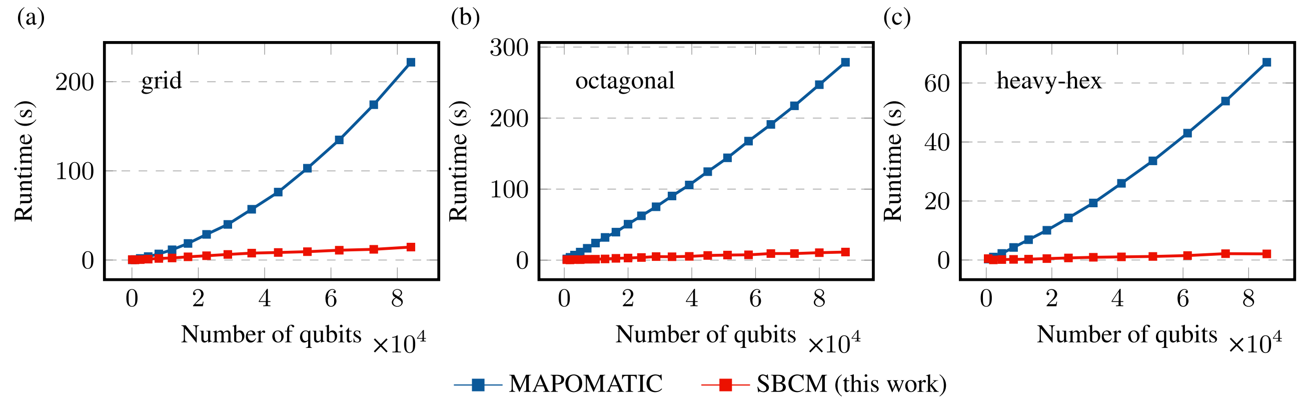

FIG. 5. Runtime of two circuit-remapping algorithms: MAPOMATIC (blue) [14] and SBCM (red). The experiments were conducted on three distinct quantum hardware architectures.

quantum hardware with typical architectures, underscoring the significance of using hardware symmetries in finding the optimal circuit mapping.

V. CONCLUSIONS

In this work, we have introduced an efficient algorithm for identifying all circuit mappings within scalable quantum systems by leveraging its inherent symmetries. This algorithm is based on a subgraph-matching approach that harnesses hardware symmetries to significantly reduce the size of the search space. We have provided theoretical proof of the optimality of the symmetry-based algorithms and conducted numerical experiments to confirm their advantages in practical scenarios. It is worth noting that the symmetry-based circuit mapping can be integrated with other existing compilation techniques [48] and helps to bridge the gap between theoretical quantum algorithms and their physical implementation on quantum computers with operational constraints and limited resources.

As scalable quantum processors necessarily possess symmetries, we envision the potential to leverage hardware symmetries across various aspects of quantum circuit compilation and beyond. For instance, the intrinsic symmetries of quantum devices could be exploited to increase the efficiency of quantum gate optimizations, dynamic circuit compilation, and distributed quantum computing across multiple quantum processors. We believe that ample opportunities for future research exist to delve deeper into these promising possibilities.

ACKNOWLEDGMENTS

D.Y. acknowledges valuable discussions with Zhiping Liu. K.F. is supported by the University Development Fund (Grant No. UDF01003565) from the Chinese University of Hong Kong, Shenzhen.

[1] P. W. Shor, in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, edited by S. Goldwasser (IEEE Computer Society, 1994), p. 124.

[2] L. K. Grover, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, edited by Gary L. Miller (Association for Computing Machinery, 1996), p. 212.

[3] B. P. Lanyon, J. D. Whitfield, G. G. Gillett, M. E. Goggin, M. P. Almeida, I. Kassal, J. D. Biamonte, M. Mohseni, B. J. Powell, M. Barbieri, A. Aspuru-Guzik, and A. G. White, Towards quantum chemistry on a quantum computer, Nat. Chem. **2**, 106 (2010).

[4] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature **549**, 195 (2017).

[5] G. Li, Y. Ding, and Y. Xie, in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, edited by I. Bahar and M. Herlihy (Association for Computing Machinery, New York, 2019), p. 1001.

[6] M. Y. Siraichi, V. F. dos Santos, C. Collange, and F. M. Q. Pereira, Qubit allocation as a combination of subgraph isomorphism and token swapping, Proc. ACM Program. Lang. **3**, 1 (2019).

[7] M. Y. Siraichi, V. F. dos Santos, C. Collange, and F. M. Q. Pereira, in *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, edited by J. Knoop and M. Schordan (Association for Computing Machinery, New York, 2018), p. 113.

[8] R. Wille, L. Burgholzer, and A. Zulehner, in *Proceedings of the 56th Annual Design Automation Conference 2019*, edited by R. Aitken (Association for Computing Machinery, New York, 2019), p. 1.

[9] A. Zulehner, S. Gasser, and R. Wille, in *International Conference on Reversible Computation*, edited by I. Phillips and H. Rahaman (Springer Cham, 2017), p. 185.

[10] 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. **38**, 1226 (2018).

[11] B. Tan and J. Cong, in *Proceedings of the 39th International Conference on Computer-Aided Design*, edited by Y. Xie (Association for Computing Machinery, New York, 2020), p. 1.
[12] 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. **41**, 359 (2021).
[13] C. Zhang, A. B. Hayes, L. Qiu, Y. Jin, Y. Chen, and E. Z. Zhang, in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (2021), p. 360.
[14] P. D. Nation and M. Treinish, Suppressing quantum circuit errors due to system variability, PRX Quantum **4**, 010327 (2023).
[15] S. A. Cook, in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, series and number STOC '71, edited by M. A. Harrison, R. B. Banerji, and J. D. Ullman (Association for Computing Machinery, New York, NY, USA, 1971), p. 151.
[16] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, A (sub) graph isomorphism algorithm for matching large graphs, IEEE Trans. Pattern Anal. Mach. Intell. **26**, 1367 (2004).
[17] A. Jüttner and P. Madarasi, VF2++—An improved subgraph isomorphism algorithm, Discrete Appl. Math. **242**, 69 (2018).
[18] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell *et al.*, Quantum supremacy using a programmable superconducting processor, Nature **574**, 505 (2019).
[19] Y. Wu, W.-S. Bao, S. Cao, F. Chen, M.-C. Chen, X. Chen, T.-H. Chung, H. Deng, Y. Du, D. Fan *et al.*, Strong quantum computational advantage using a superconducting quantum processor, Phys. Rev. Lett. **127**, 180501 (2021).
[20] https://newsroom.ibm.com/2022-11-09-IBM-Unveils-400-Qubit-Plus-Quantum-Processor-and-Next-Generation-IBM-Quantum-System-Two (2022).
[21] https://www.ibm.com/quantum/roadmap (2022).
[22] C. Lyu, X. Xu, M.-H. Yung, and A. Bayat, Symmetry enhanced variational quantum spin eigensolver, Quantum **7**, 899 (2023).
[23] J. J. Meyer, M. Mularski, E. Gil-Fuster, A. A. Mele, F. Arzani, A. Wilms, and J. Eisert, Exploiting symmetry in variational quantum machine learning, PRX Quantum **4**, 010328 (2023).
[24] X. Wang, K. Fang, and R. Duan, Semidefinite programming converse bounds for quantum communication, IEEE Trans. Inf. Theory **65**, 2583 (2018).
[25] K. Fang and H. Fawzi, Geometric Rényi divergence and its applications in quantum channel capacities, Commun. Math. Phys. **384**, 1615 (2021).
[26] K. Fang, X. Wang, M. Tomamichel, and M. Berta, Quantum channel simulation and the channel's smooth max-information, IEEE Trans. Inf. Theory **66**, 2129 (2019).
[27] K. Fang, G. Gour, and X. Wang, Towards the ultimate limits of quantum channel discrimination, preprint arXiv:2110.14842.
[28] K. Fang, O. Fawzi, R. Renner, and D. Sutter, Chain rule for the quantum relative entropy, Phys. Rev. Lett. **124**, 100501 (2020).
[29] M. Hayashi, K. Fang, and K. Wang, Finite block length analysis on quantum coherence distillation and incoherent randomness extraction, IEEE Trans. Inf. Theory **67**, 3926 (2021).
[30] K. Fang and Z.-W. Liu, No-go theorems for quantum resource purification: New approach and channel theory, PRX Quantum **3**, 010337 (2022).
[31] K. Fang and Z.-W. Liu, No-go theorems for quantum resource purification, Phys. Rev. Lett. **125**, 060405 (2020).
[32] B. Regula, K. Fang, X. Wang, and M. Gu, One-shot entanglement distillation beyond local operations and classical communication, New J. Phys. **21**, 103017 (2019).
[33] K. Fang, X. Wang, M. Tomamichel, and R. Duan, Non-asymptotic entanglement distillation, IEEE Trans. Inf. Theory **65**, 6454 (2019).
[34] K. Fang and H. Fawzi, The sum-of-squares hierarchy on the sphere and applications in quantum information theory, Math. Program. **190**, 331 (2021).
[35] M. G. Díaz, K. Fang, X. Wang, M. Rosati, M. Skotiniotis, J. Calsamiglia, and A. Winter, Using and reusing coherence to realize quantum processes, Quantum **2**, 100 (2018).
[36] K. Fang, X. Wang, L. Lami, B. Regula, and G. Adesso, Probabilistic distillation of quantum coherence, Phys. Rev. Lett. **121**, 070404 (2018).
[37] B. Regula, K. Fang, X. Wang, and G. Adesso, One-shot coherence distillation, Phys. Rev. Lett. **121**, 010401 (2018).
[38] W. Xie, K. Fang, X. Wang, and R. Duan, Approximate broadcasting of quantum correlations, Phys. Rev. A **96**, 022302 (2017).
[39] M. Qiao, H. Zhang, and H. Cheng, Subgraph matching: On compression and computation, Proc. VLDB Endow. **11**, 176 (2017).
[40] QISKIT, https://qiskit.org (2023).
[41] https://numpy.org/
[42] L. Roditty and V. Vassilevska Williams, in *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing* (2013), p. 515.
[43] https://networkx.org/documentation/stable/reference/algorithms/isomorphism.html (2023).
[44] https://qcs.rigetti.com/qpus (2022).
[45] https://research.ibm.com/blog/heavy-hex-lattice (2021).
[46] D. Deutsch and R. Jozsa, Rapid solution of problems by quantum computation, Proc. R. Soc. Lond. Ser. A: Mathe. Phys. Sci. **439**, 553 (1992).
[47] D. Yu and K. Fang, Symmetry-based quantum circuit mapping, arXiv:2310.18026 [quant-ph]
[48] K. Fang, M. Zhang, R. Shi, and Y. Li, Dynamic quantum circuit compilation, preprint arXiv:2310.11021.