PHYSICAL REVIEW A **89**, 022342 (2014)

Graph isomorphism and adiabatic quantum computing

Frank Gaitan[1] and Lane Clark[2]

[1]*Laboratory for Physical Sciences, 8050 Greenmead Dr, College Park, Maryland 20740, USA*

[2]*Department of Mathematics, Southern Illinois University, Carbondale, Illinois 62901, USA*

(Received 18 April 2013; published 28 February 2014)

In the graph isomorphism (GI) problem two N-vertex graphs G and G' are given and the task is to determine whether there exists a permutation of the vertices of G that preserves adjacency and transforms $G \to G'$. If yes, then G and G' are said to be isomorphic; otherwise they are nonisomorphic. The GI problem is an important problem in computer science and is thought to be of comparable difficulty to integer factorization. In this paper we present a quantum algorithm that solves arbitrary instances of GI and which also provides an approach to determining all automorphisms of a given graph. We show how the GI problem can be converted to a combinatorial optimization problem that can be solved using adiabatic quantum evolution. We numerically simulate the algorithm's quantum dynamics and show that it correctly (i) distinguishes nonisomorphic graphs; (ii) recognizes isomorphic graphs and determines the permutation(s) that connect them; and (iii) finds the automorphism group of a given graph G. We then discuss the GI quantum algorithm's experimental implementation, and close by showing how it can be leveraged to give a quantum algorithm that solves arbitrary instances of the NP-complete subgraph isomorphism problem. The computational complexity of an adiabatic quantum algorithm is largely determined by the minimum energy gap $\Delta(N)$ separating the ground and first-excited states in the limit of large problem size $N \gg 1$. Calculating $\Delta(N)$ in this limit is a fundamental open problem in adiabatic quantum computing, and so it is not possible to determine the computational complexity of adiabatic quantum algorithms in general, nor consequently, of the specific adiabatic quantum algorithms presented here. Adiabatic quantum computing has been shown to be equivalent to the circuit model of quantum computing, and so development of adiabatic quantum algorithms continues to be of great interest.

DOI: 10.1103/PhysRevA.89.022342 PACS number(s): 03.67.Ac, 02.10.Ox, 89.75.Hc

I. INTRODUCTION

An instance of the graph isomorphism (GI) problem is specified by two N-vertex graphs G and G' and the challenge is to determine whether there exists a permutation of the vertices of G that preserves adjacency and transforms $G \to G'$. When such a permutation exists, the graphs are said to be isomorphic; otherwise they are nonisomorphic. GI has been heavily studied in computer science [1]. Polynomial classical algorithms exist for special cases of GI; still it has not been possible to prove that GI is in P. Although it is known that GI is in NP, it has also not been possible to prove that it is NP-complete. The situation is the same for integer factorization (IF)—it belongs to NP, but is not known to be in P or to be NP-complete. GI and IF are believed to be of comparable computational difficulty [2].

IF and GI have also been examined from the perspective of quantum algorithms and both have been connected to the hidden subgroup problem (HSP) [3]. For IF the hidden subgroup is contained in an Abelian parent group ($Z_n^* =$ group of units modulo n), while for GI the parent group is non-Abelian ($S_n =$ symmetric group on n elements). While Fourier sampling allows the Abelian HSP to be solved efficiently [4], strong Fourier sampling does not allow an efficient solution of the non-Abelian HSP over S_n [5]. At this time an efficient quantum algorithm for GI is not known.

A number of researchers have considered using the dynamics of physical systems to solve instances of GI. Starting from physically motivated conjectures, these approaches embed the structure of the graphs appearing in the GI instance into the Hamiltonian that drives the system dynamics. In Ref. [6] the systems considered were classical, while Refs. [7–11] worked with quantum systems.

(1) Building on Refs. [7] and [8], Refs. [9] and [10] proposed using multiparticle quantum random walks (QRW) on graphs as a means for distinguishing pairs of nonisomorphic graphs. Numerical tests of this approach focused on GI instances involving strongly regular graphs (SRG). The adjacency matrix for each SRG was used to define the Hamiltonian $H(G)$ that drives the QRW on the graph G. The walkers can only hop between vertices joined by an edge in G. The propagator $U(G) = \exp[-iH(G)t]$ is evaluated at a fixed time t for each SRG associated with a GI instance. The two propagators are used to define a comparison function that is conjectured to vanish for isomorphic pairs of SRGs, and to be nonzero otherwise. Refs. [9] and [10] examined both interacting and noninteracting systems of quantum walkers and found that (i) no noninteracting QRW with a fixed number of walkers can distinguish all pairs of SRGs; (ii) increasing the number of walkers increases the distinguishing power of the approach; and (iii) two interacting bosonic walkers have more distinguishing power than both one and two noninteracting walkers. No analysis was provided of the algorithm's runtime $T(N)$ versus problem size N in the limit of large problem size $N \gg 1$, and so the computational complexity of this GI algorithm is currently unknown. Note that for isomorphic pairs of graphs, this algorithm cannot determine the permutation(s) connecting the two graphs, nor the automorphism group of a given graph. Finally, no discussion of the algorithm's experimental implementation is given.

(2) The GI algorithm presented in Ref. [11] is based on adiabatic quantum evolution: Here the problem Hamiltonian is an Ising Hamiltonian that identifies the qubits with the vertices of a graph, and qubits i and j interact antiferromagnetically only when vertices i and j in the graph are joined by an

edge. It was conjectured that the instantaneous ground state encodes enough information about the graph to allow suitably chosen measurements to distinguish pairs of nonisomorphic graphs. Physical observables that are invariant under qubit permutations are measured at intermediate times and the differences in the dynamics generated by two nonisomorphic graphs are assumed to allow the measurement outcomes to recognize the graphs as nonisomorphic. The algorithm was tested numerically on (mostly) SRGs, and it was found that combinations of measurements of the (i) spin-glass order parameter, (ii) x magnetization, and (iii) total average energy allowed all the nonisomorphic pairs of graphs examined to be distinguished. Reference [11] did not determine the scaling relation for the runtime $T(N)$ versus problem size N for $N \gg 1$, and so the computational complexity of this algorithm is currently unknown. The algorithm is also unable to determine the permutation(s) that connect a pair of isomorphic graphs, nor the automorphism group of a given graph. Reference [11] discussed the experimental implementation of this algorithm and noted that the measurements available on the D-Wave hardware do not allow the observables used in the numerical tests to be measured on the hardware.

In this paper we present a quantum algorithm that solves arbitrary instances of GI. The algorithm also provides an approach for determining the automorphism group of a given graph. The GI quantum algorithm is constructed by first converting an instance of GI into an instance of a combinatorial optimization problem whose cost function, by construction, has a zero minimum value when the pair of graphs in the GI instance are isomorphic, and is positive when the pair are nonisomorphic. The specification of the GI quantum algorithm is completed by showing how the combinatorial optimization problem can be solved using adiabatic quantum evolution. To test the effectiveness of this GI quantum algorithm we numerically simulated its Schrödinger dynamics. The simulation results show that it can correctly (i) distinguish pairs of nonisomorphic graphs; (ii) recognize pairs of isomorphic graphs and determine the permutation(s) that connect them; and (iii) find the automorphism group of a given graph. We also discuss the experimental implementation of the GI algorithm, and show how it can be leveraged to give a quantum algorithm that solves arbitrary instances of the (NP-complete) subgraph isomorphism (SGI) problem. As explained in Sec. IV, calculation of the runtime for an adiabatic quantum algorithm in the limit of large problem size is a fundamental open problem in adiabatic quantum computing. It is thus not presently possible to determine the computational complexity of adiabatic quantum algorithms in general, nor, consequently, of the specific adiabatic quantum algorithms presented here. However, because adiabatic quantum computing has been shown to be equivalent to the circuit model of quantum computing [12–14], the development of adiabatic quantum algorithms continues to be of great interest. Just as with the GI algorithms of Refs. [8–11], our GI algorithm also has unknown complexity. However, unlike the algorithms of Refs. [8–11], the GI algorithm presented here (i) encodes the GI instance explicitly into the cost function of a combinatorial optimization problem which is solved using adiabatic quantum evolution without introducing physical conjectures; and (ii) determines the permutation(s) connecting two isomorphic graphs, and the automorphism group of a given graph. As we shall see, our GI algorithm can be implemented on existing D-Wave hardware using established embedding procedures [15]. Such an implementation is in the works and will be reported elsewhere.

The structure of this paper is as follows. In Sec. II we give a careful presentation of the GI problem, and show how an instance of GI can be converted to an instance of a combinatorial optimization problem whose solution (Sec. III) can be found using adiabatic quantum evolution. To test the performance of the GI quantum algorithm introduced in Sec. III, we numerically simulated its Schrödinger dynamics and the results of that simulation are presented in Sec. IV. In Sec. V we describe the experimental implementation of the GI algorithm, and in Sec. VI we show how it can be used to give a quantum algorithm that solves arbitrary instances of the NP-complete problem known as subgraph isomorphism. Finally, we summarize our results in Sec. VII. Two appendices are also included. The first briefly summarizes the quantum adiabatic theorem and its use in adiabatic quantum computing, and the second reviews the approach to embedding the problem Hamiltonian for an adiabatic quantum algorithm onto the D-Wave hardware that was presented in Ref. [15].

II. GRAPH ISOMORPHISM PROBLEM

In this section we introduce the GI problem and show how an instance of GI can be converted into an instance of a combinatorial optimization problem (COP) whose cost function has zero (nonzero) minimum value when the pair of graphs being studied is isomorphic (nonisomorphic).

A. Graphs and graph isomorphism

A graph G is specified by a set of vertices V and a set of edges E. We focus on *simple* graphs in which an edge only connects distinct vertices, and the edges are undirected. The *order of* G is defined to be the number of vertices contained in V, and two vertices are said to be *adjacent* if they are connected by an edge. If x and y are adjacent, we say that y is a *neighbor* of x, and vice versa. The *degree* $d(x)$ of a vertex x is equal to the number of vertices that are adjacent to x. The *degree sequence* of a graph lists the degree of each vertex in the graph ordered from largest degree to smallest. A graph G of order N can also be specified by its *adjacency matrix* A, which is an $N \times N$ matrix whose matrix element $a_{i,j} = 1$ (0) if the vertices i and j are (are not) adjacent. For simple graphs $a_{i,i} = 0$ and $a_{i,j} = a_{j,i}$ since edges only connect distinct vertices and are undirected.

Two graphs G and G' are said to be *isomorphic* if there is a one-to-one correspondence π between the vertex sets V and V' such that two vertices x and y are adjacent in G if and only if their images π_x and π_y are adjacent in G'. The graphs G and G' are nonisomorphic if no such π exists. Since no one-to-one correspondence π can exist when the number of vertices in G and G' is different, graphs with unequal orders are always nonisomorphic. It can also be shown [16] that if two graphs are isomorphic, they must have identical degree sequences.

We can also describe graph isomorphism in terms of the adjacency matrices A and A' of the graphs G and G', respectively. The graphs are isomorphic if and only if there exists a permutation matrix σ of the vertices of G that satisfies

$$A' = \sigma A \sigma^T, \tag{1}$$

where σ^T is the transpose of σ. It is straightforward to show that if G and G' are isomorphic, then a permutation matrix σ exists that satisfies Eq. (1). To prove the "only if" statement, note that the i-j matrix element of the right-hand side (RHS) is A_{σ_i,σ_j}. Equation (1) is thus satisfied when a permutation matrix σ exists such that $A'_{i,j} = A_{\sigma_i,\sigma_j}$. Thus Eq. (1) is simply the condition that σ preserve adjacency. Since a permutation is a one-to-one correspondence, the existence of a permutation matrix σ satisfying Eq. (1) implies G and G' are isomorphic. Thus, the existence of a permutation matrix σ satisfying Eq. (1) is an equivalent way to define graph isomorphism.

The GI problem is to determine whether two given graphs G and G' are isomorphic. The problem is only nontrivial when G and G' have the same order and so we focus on that case in this paper.

B. Permutations, binary strings, and linear maps

A permutation π of a finite set $\mathcal{S} = \{0,\ldots,N-1\}$ is a one-to-one correspondence from $\mathcal{S} \to \mathcal{S}$ which sends $i \to \pi_i$ such that $\pi_i \in \mathcal{S}$, and $\pi_i \neq \pi_j$ for $i \neq j$. The permutation π can be written

$$\pi = \begin{pmatrix} 0 & \cdots & i & \cdots & N-1 \\ \pi_0 & \cdots & \pi_i & \cdots & \pi_{N-1} \end{pmatrix}, \tag{2}$$

where column i indicates that π sends $i \to \pi_i$. Since the top row on the RHS of Eq. (2) is the same for all permutations, all the information about π is contained in the bottom row. Thus we can map a permutation π into an integer string $P(\pi) = \pi_0 \cdots \pi_{N-1}$, with $\pi_i \in \mathcal{S}$ and $\pi_i \neq \pi_j$ for $i \neq j$.

For reasons that will become clear in Sec. II C, we want to convert the integer string $P(\pi) = \pi_0 \cdots \pi_{N-1}$ into a binary string p_π. This can be done by replacing each π_i in $P(\pi)$ by the unique binary string formed from the coefficients appearing in its binary decomposition

$$\pi_i = \sum_{j=0}^{U-1} \pi_{i,j}\,(2)^j\,. \tag{3}$$

Here $U \equiv \lceil \log_2 N \rceil$. Thus the integer string $P(\pi)$ is transformed to the binary string

$$p_\pi = (\pi_{0,0}\cdots\pi_{0,U-1})\cdots(\pi_{N-1,0}\cdots\pi_{N-1,U-1}), \tag{4}$$

where $\pi_{i,j} \in \{0,1\}$. The binary string p_π has length NU, where

$$N \leqslant 2^U \equiv M+1. \tag{5}$$

Thus we can identify a permutation π with the binary string p_π in Eq. (4).

Let $\mathcal{H}$ be the Hamming space of binary strings of length NU. This space contains 2^{NU} strings, and we have just seen that $N!$ of these strings p_π encode permutations π. Our last task is to define a mapping from $\mathcal{H}$ to the space of $N \times N$ matrices σ with binary matrix elements $\sigma_{i,j} = 0,1$. The mapping is constructed as follows:

(1) Let $s_b = s_0 \cdots s_{NU-1}$ be a binary string in $\mathcal{H}$. We parse s_b into N substrings of length U as follows:

$$s_b = (s_0\cdots s_{U-1})(s_U\cdots s_{2U-1})\cdots(s_{(N-1)U}\cdots s_{NU-1}). \tag{6}$$

(2) For each substring $s_{iU}\cdots s_{(i+1)U-1}$, construct the integer

$$s_i = \sum_{j=0}^{U-1} s_{iU+j}(2)^j \leqslant 2^U - 1 = M. \tag{7}$$

(3) Finally, introduce the integer string $s = s_0 \cdots s_{N-1}$, and define the $N \times N$ matrix $\sigma(s)$ to have matrix elements

$$\sigma_{i,j}(s) = \begin{cases} 0, & \text{if } s_j > N-1 \\ \delta_{i,s_j}, & \text{if } 0 \leqslant s_j \leqslant N-1, \end{cases} \tag{8}$$

where $i,j \in \mathcal{S}$, and $\delta_{x,y}$ is the Kronecker delta. Note that when the binary string s_b corresponds to a permutation, the matrix $\sigma(s)$ is a permutation matrix since the s_i formed in step (2) will obey $0 \leqslant s_i \leqslant N-1$ and $s_i \neq s_j$ for $i \neq j$. In this case, if A is the adjacency matrix for a graph G, then $A' = \sigma(s)A\sigma^T(s)$ will be the adjacency matrix for a graph G' isomorphic to G. On the other hand, if s_b does not correspond to a permutation, then the adjacency matrix $A' = \sigma(s)A\sigma^T(s)$ must correspond to a graph G' which is not isomorphic to G.

The result of our development so far is the establishment of a map from binary strings of length NU to $N \times N$ matrices (viz., linear maps) with binary matrix elements. When the string is (is not) a permutation, the matrix produced is (is not) a permutation matrix. Finally, recall from Stirling's formula that $\log_2 N! \sim N\log_2 N - N$, which is the number of bits needed to represent $N!$. Our encoding of permutations uses $N\lceil \log_2 N \rceil$ bits and so approaches asymptotically what is required by Stirling's formula.

C. Graph isomorphism and combinatorial optimization

As seen above, an instance of GI is specified by a pair of graphs G and G' (or equivalently, by a pair of adjacency matrices A and A'). Here we show how a GI instance can be transformed into an instance of a COP whose cost function has a minimum value of zero if and only if G and G' are isomorphic.

The search space for the COP is the Hamming space $\mathcal{H}$ of binary strings s_b of length NU which are associated with the integer strings s and matrices $\sigma(s)$ introduced in Sec. II B. The COP cost function $C(s)$ contains three contributions:

$$C(s) = C_1(s) + C_2(s) + C_3(s). \tag{9}$$

The first two terms on the RHS penalize integer strings $s = s_0 \cdots s_{N-1}$ whose associated matrix $\sigma(s)$ is not a permutation matrix,

$$C_1(s) = \sum_{i=0}^{N-1}\sum_{\alpha=N}^{M} \delta_{s_i,\alpha}, \tag{10}$$

$$C_2(s) = \sum_{i=0}^{N-2}\sum_{j=i+1}^{N-1} \delta_{s_i,s_j}, \tag{11}$$

where $\delta_{x,y}$ is the Kronecker delta. We see that $C_1(s) > 0$ when $s_i > N-1$ for some i, and $C_2(s) > 0$ when $s_i = s_j$ for some $i \neq j$. Thus $C_1(s) + C_2(s) = 0$ if and only if $\sigma(s)$

is a permutation matrix. The third term $C_3(s)$ adds a penalty when $\sigma(s)A\sigma^T(s) \neq A'$:

$$C_3(s) = \|\sigma(s)A\sigma^T(s) - A'\|_i. \tag{12}$$

Here $\|M\|_i$ is the L_i norm of M. In the numerical simulations discussed in Sec. IV, the L_1 norm is used, though any L_i norm would be acceptable. Thus, when G and G' are isomorphic, $C_3(s) = 0$, and $\sigma(s)$ is the permutation of vertices of G that maps $G \rightarrow G'$. Putting all these remarks together, we see that if $C(s) = 0$ for some integer string s, then G and G' are isomorphic and $\sigma(s)$ is the permutation that connects them. On the other hand, if $C(s) > 0$ for all strings s, then G and G' are nonisomorphic.

We have thus converted an instance of GI into an instance of the following COP:

Graph isomorphism COP. Given the N-vertex graphs G and G' and the associated cost function $C(s)$ defined above, find an integer string s_* that minimizes $C(s)$.

By construction (i) $C(s_*) = 0$ if and only if G and G' are isomorphic and $\sigma(s_*)$ is the permutation matrix mapping $G \rightarrow G'$; and (ii) $C(s_*) > 0$ if and only if G and G' are nonisomorphic.

Before moving on, notice that if $G = G'$, then $C(s_*) = 0$ since G is certainly isomorphic to itself. In this case $\sigma(s_*)$ is an automorphism of G. We shall see that the GI quantum algorithm to be introduced in Sec. III provides an approach for finding the automorphism group of a graph.

III. ADIABATIC QUANTUM ALGORITHM FOR GRAPH ISOMORPHISM

A quantum algorithm is an algorithm that can be run on a realistic model of quantum computation [17]. One such model is adiabatic quantum computation [18] which is based on adiabatic quantum evolution [19,20]. The adiabatic quantum optimization (AQO) algorithm [21] is an example of adiabatic quantum computation that exploits the adiabatic dynamics of a quantum system to solve a COP (see Appendix A for a brief overview). The AQO algorithm uses the optimization problem cost function to define a problem Hamiltonian H_P whose ground-state subspace encodes all problem solutions. The algorithm evolves the state of an L-qubit register from the ground state of an initial Hamiltonian H_i to the ground state of H_P with probability approaching 1 in the adiabatic limit. An appropriate measurement at the end of the adiabatic evolution yields a solution of the optimization problem almost certainly. The time-dependent Hamiltonian $H(t)$ for global AQO is

$$H(t) = \left(1 - \frac{t}{T}\right) H_i + \left(\frac{t}{T}\right) H_P, \tag{13}$$

where T is the algorithm runtime, and adiabatic dynamics corresponds to $T \rightarrow \infty$.

To map the GI COP onto an adiabatic quantum computation, we begin by promoting the binary strings s_b to computational basis states (CBS) $|s_b\rangle$. Thus each bit in s_b is promoted to a qubit so that the quantum register contains $L = NU = N\lceil\log_2 N\rceil$ qubits. The CBS are defined to be the 2^L eigenstates of $\sigma_z^0 \otimes \cdots \otimes \sigma_z^{L-1}$. The problem Hamiltonian H_P is defined to be diagonal in the CBS with eigenvalue $C(s)$, where s is the integer string associated with s_b:

$$H_P|s_b\rangle = C(s)|s_b\rangle. \tag{14}$$

Note that (see Sec. II C) the ground-state energy of H_P will be zero if and only if the graphs G and G' are isomorphic. We will discuss the experimental realization of H_P in Sec. V. The initial Hamiltonian H_i is chosen to be

$$H_i = \sum_{l=0}^{L-1} \frac{1}{2}\left(I^l - \sigma_x^l\right), \tag{15}$$

where I^l and σ_x^l are the identity and x-Pauli operator for qubit l, respectively. The ground state of H_i is the easily constructed uniform superposition of CBS.

The quantum algorithm for GI begins by preparing the L-qubit register in the ground state of H_i and then driving the qubit register dynamics using the time-dependent Hamiltonian $H(t)$. At the end of the evolution the qubits are measured in the computational basis. The outcome is the bit string s_b^* so that the final state of the register is $|s_b^*\rangle$ and its energy is $C(s^*)$, where s^* is the integer string derived from s_b^*. In the adiabatic limit, $C(s^*)$ will be the ground-state energy, and if $C(s^*) = 0$ (>0) the algorithm decides G and G' are isomorphic (nonisomorphic). Note that any real application of AQO will only be approximately adiabatic. Thus the probability that the final energy $C(s^*)$ will be the ground-state energy will be $1 - \epsilon$. In this case the GI quantum algorithm must be run $k \sim O(\ln(1-\delta)/\ln\epsilon)$ times so that, with probability $\delta > 1 - \epsilon$, at least one of the measurements will return the ground-state energy. We can make δ arbitrarily close to 1 by choosing k sufficiently large.

IV. NUMERICAL SIMULATION OF ADIABATIC QUANTUM ALGORITHM

In this section we present the results of a numerical simulation of the dynamics of the GI adiabatic quantum algorithm (AQA). Because the GI AQA uses $N\lceil\log_2 N\rceil$ qubits, and these simulations were carried out using a classical digital computer, we were limited to GI instances involving graphs with order $N \leqslant 7$. Although we would like to have examined larger graphs, this simply was not practical. Note that the $N = 7$ simulations use 21 qubits. These simulations are at the upper limit of 20–22 qubits at which simulation of the full adiabatic Schrödinger dynamics is feasible [22–24]. To simulate a GI instance with graphs of order $N = 8$ requires a 24-qubit simulation which is well beyond what can be done practically. The protocol for the simulations presented here follows Refs. [22–25].

As explained in Appendix A, the runtime for an adiabatic quantum algorithm is related to the minimum energy gap arising during the course of the adiabatic quantum evolution. Thus, determining the runtime scaling relation $T(N)$ versus problem size N in the asymptotic limit $(N \gg 1)$, reduces to determining the minimum gap scaling relation $\Delta(N)$ for large N. This, however, is a well-known, fundamental open problem in adiabatic quantum computing, and so it was not possible to determine the asymptotic runtime scaling relation for our GI AQA. Although our numerical simulations could be used to compute a runtime for each of the GI instances considered below, we did not do so for two reasons. First, as noted above, the GI instances that can be simulated using a digital computer are limited to graphs with no more than seven vertices. These instances are thus far from the large problem-size limit $N \gg 1$, and so the associated runtimes tell us nothing about the asymptotic performance of the GI AQA. Second, it is well known that the minimum energy gap encountered during adiabatic quantum evolution (and which largely determines the runtime) is sensitive to the particular Hamiltonian path followed by the adiabatic quantum evolution [26,27]. Determining the optimal Hamiltonian path which yields the largest minimum gap, and thus the shortest possible runtime, is another fundamental open problem in adiabatic quantum computing. As a result, the Hamiltonian path used in the numerical simulations [i.e., the linear interpolating Hamiltonian $H(t)$ in Eq. (13)] will almost certainly be nonoptimal, and so the runtime it produces will also, almost certainly, be nonoptimal, and thus a poor indicator of GI AQA performance.

In Sec. IV A we present simulation results for simple examples of isomorphic and nonisomorphic graphs. These examples allow us to illustrate the analysis of the simulation results in a simple setting. Section IV B then presents our simulation results for nonisomorphic instances of (i) isospectral graphs; and (ii) strongly regular graphs. Finally, Sec. IV C considers GI instances where $G' = G$. Clearly, all such instances correspond to isomorphic graphs since the identity permutation will always map $G \to G$ and preserve adjacency. The situation is more interesting when G has symmetries which allow nontrivial permutations as graph isomorphisms. These self-isomorphisms are referred to as graph automorphisms, and they form a group known as the automorphism group Aut(G) of G. In this final section we use the GI AQA to find Aut(G) for a number of graphs. In all GI instances considered in this section, the GI AQA correctly (i) distinguished nonisomorphic pairs of graphs; (ii) recognized isomorphic pairs of graphs; and (iii) determined the automorphism group of a given graph.

A. Illustrative examples

Here we present the results of a numerical simulation of the GI AQA applied to two simple GI instances. In Sec. IV A 1 (Sec. IV A 2) we examine an instance of two nonisomorphic (isomorphic) graphs. For the isomorphic instance we also present the permutations found by the GI AQA that transforms G into G' while preserving adjacency.

1. *Nonisomorphic graphs*

Here we use the GI AQA to examine a GI instance in which the two graphs G and G' are nonisomorphic. The two graphs are shown in Fig. 1. Each graph contains four vertices and four edges, however, they are nonisomorphic. Examining Fig. 1 we see that the degree sequence for G is {2,2,2,2}, while that for G' is {3,2,2,1}. Since these degree sequences are different we know that G and G' are nonisomorphic. Finally, the adjacency

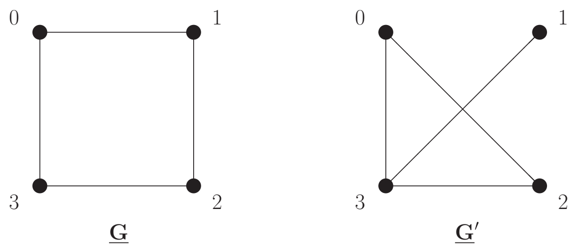

FIG. 1. Two nonisomorphic four-vertex graphs G and G'.

matrices A and A' for G and G', respectively, are

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}; \quad A' = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}. \tag{16}$$

The GI AQA finds a nonzero final ground-state energy $E_{gs} = 4$, and so correctly identifies these two graphs as nonisomorphic. It also finds that the final ground-state subspace has a degeneracy of 16. The linear maps associated with the 16 CBS that span this subspace give rise to the lowest cost linear maps of G relative to G'. As these graphs are not isomorphic, these lowest cost maps have little inherent interest and so we do not list them.

2. *Isomorphic graphs*

Here we examine the case of two isomorphic graphs G and G' which are shown in Fig. 2. Each graph contains four vertices and five edges, and both graphs have degree sequence {3,3,2,2}. By inspection of Fig. 2, the associated adjacency matrices are, respectively,

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}; \quad A' = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}. \tag{17}$$

For these two graphs, the GI AQA finds a vanishing final ground-state energy $E_{gs} = 0$ and so recognizes G and G' as isomorphic. It also finds that the final ground-state subspace has a degeneracy of 4. The four CBS that span this subspace give four binary strings s_b. These four binary strings in turn generate four integer strings s which are, respectively, the

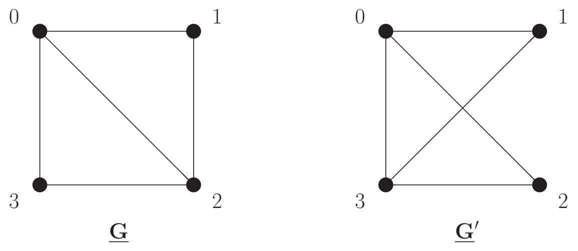

FIG. 2. Two isomorphic four-vertex graphs G and G'.

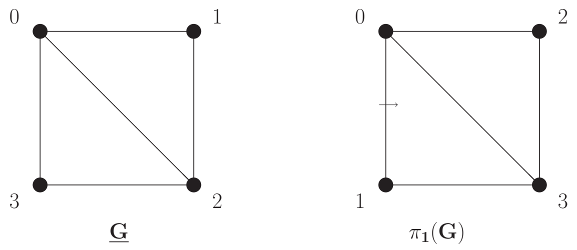

FIG. 3. Transformation of G produced by the permutation π_1.

bottom row of four permutations [see Eq. (2)]. Thus, not only does the GI AQA recognize G and G' as isomorphic, but it also returns the four graph isomorphisms $\pi_1, \ldots ,\pi_4$ that transform $G \to G'$ while preserving adjacency:

$$\pi_1 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 2 & 3 & 1 \end{pmatrix}; \quad \pi_2 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 3 & 2 & 0 & 1 \end{pmatrix};$$
$$\pi_3 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 3 & 1 & 0 & 2 \end{pmatrix}; \quad \pi_4 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 3 & 2 \end{pmatrix}. \tag{18}$$

We will next explicitly show that π_1 is a graph isomorphism; the reader can easily check that the remaining three permutations are also graph isomorphisms.

The permutation matrix σ_1 associated with π_1 is

$$\sigma_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{19}$$

Under π_1, G is transformed to $\pi_1(G)$, which is shown in Fig. 3. It is clear from Fig. 3 that vertices x and y are adjacent in G if and only if $\pi_{1,x}$ and $\pi_{1,y}$ are adjacent in $\pi_1(G)$. The adjacency matrix for $\pi_1(G)$ is $\sigma_1 A \sigma_1^T$, which is easily shown to be

$$\sigma_1 A \sigma_1^T = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}. \tag{20}$$

This agrees with the adjacency shown in $\pi_1(G)$ in Fig. 3. Comparison with Eq. (17) shows that $\sigma_1 A \sigma_1^T = A'$. Thus the permutation π_1 does map G into G' and preserve adjacency and so establishes that G and G' are isomorphic.

Finally, note that G and G' are connected by exactly four graph isomorphisms. Examination of Fig. 2 shows that the degree-3 vertices in G are vertices 0 and 2, while in G' they are vertices 0 and 3. Since the degree of a vertex is preserved by a graph isomorphism [16], vertex 0 in G must be mapped to vertex 0 or 3 in G'. Then vertex 2 (in G) must be mapped to vertex 3 or 0 (in G'), respectively. This then forces vertex 1 to map to vertex 1 or 2, and vertex 3 to map to vertex 2 or 1, respectively. Thus only four graph isomorphisms are possible and these are exactly the four graph isomorphisms found by the GI AQA which appear in Eq. (18).

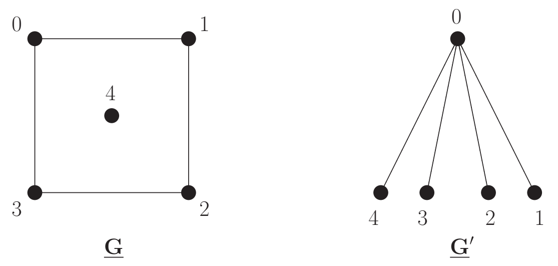

FIG. 4. Two nonisomorphic five-vertex isospectral graphs G and G'.

B. Nonisomorphic graphs

In this section we present GI instances involving pairs of nonisomorphic graphs. In Sec. IV B 1 we examine two instances of isospectral graphs, and in Sec. IV B 2 we look at three instances of strongly regular graphs. We shall see that the GI AQA correctly distinguishes all graph pairs as nonisomorphic.

1. *Isospectral graphs*

The spectrum of a graph is the set containing all the eigenvalues of its adjacency matrix. Two graphs are isospectral if they have identical spectra. Nonisomorphic isospectral graphs are believed to be difficult to distinguish [8,9]. Here we test the GI AQA on pairs of nonisomorphic isospectral graphs. The nonisomorphic pairs of isospectral graphs examined here appear in Ref. [28].

$N = 5$. It is known that no pair of graphs with less than five vertices is isospectral [28]. Figure 4 shows a pair of graphs G and G' with five vertices which are isospectral and yet are nonisomorphic. Although both have five vertices and four edges, the degree sequence of G is {2,2,2,2,0}, while that of G' is {4,1,1,1,1}. Since they have different degree sequences, it follows that G and G' are nonisomorphic. The adjacency matrices A and A' for G and G', respectively, are

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix};$$
$$A' = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{21}$$

It is straightforward to show that A and A' have the same characteristic polynomial $P(\lambda) = \lambda^5 - 4\lambda^3$ and so G and G' are isospectral. Numerical simulation of the dynamics of the GI AQA finds a nonzero final ground-state energy $E_{gs} = 5$, and so the GI AQA correctly distinguishes G and G' as nonisomorphic graphs.

$N = 6$. The pair of graphs G and G' in Fig. 5 are isospectral and nonisomorphic. To see this, note that although both graphs

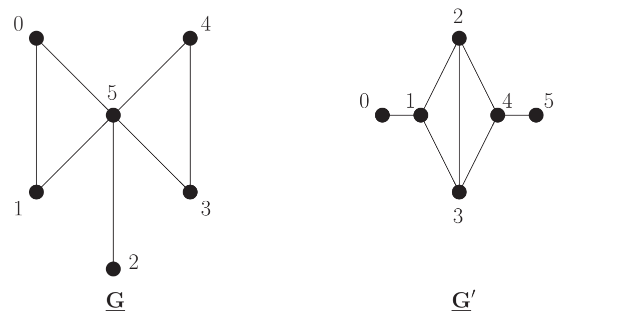

FIG. 5. Two nonisomorphic six-vertex isospectral graphs G and G'.

have six vertices and seven edges, the degree sequence of G is $\{5,2,2,2,2,1\}$, while that of G' is $\{3,3,3,3,1,1\}$. Since their degree sequences are different, G and G' are nonisomorphic. The adjacency matrices A and A' of G and G', respectively, are

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}; \tag{22}$$

$$A' = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Both A and A' have the same characteristic polynomial $P(\lambda) = \lambda^6 - 7\lambda^4 - 4\lambda^3 + 7\lambda^2 + 4\lambda - 1$, which establishes that G and G' are isospectral. Numerical simulation of the dynamics of the GI AQA finds a nonzero final ground-state energy $E_{gs} = 7$, and so the GI AQA correctly distinguishes G and G' as nonisomorphic graphs.

2. *Strongly regular graphs*

As we saw in Sec. II A, the degree $d(x)$ of a vertex x is equal to the number of vertices that are adjacent to x. A graph G is said to be *k regular* if, for all vertices x, the degree $d(x) = k$. A graph is said to be *regular* if it is k regular for some value of k. Finally, a *strongly regular* graph is a graph with ν vertices that is k regular, and for which (i) any two adjacent vertices have λ common neighbors; and (ii) any two nonadjacent vertices have μ common neighbors. The set of all strongly regular graphs is divided up into families, and each family is composed of strongly regular graphs having the same parameter values (ν,k,λ,μ). In this section we apply the GI AQA to pairs of nonisomorphic strongly regular graphs. An excellent test for this algorithm would be two nonisomorphic strongly regular graphs belonging to the *same* family as such graphs would then have the same order and degree sequence. Unfortunately, to find a family containing *at least* two nonisomorphic strongly

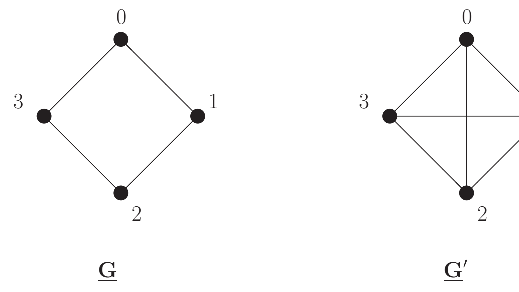

FIG. 6. Two four-vertex nonisomorphic strongly regular graphs G and G'.

regular graphs requires going to a family containing 16-vertex graphs. For example, the family (16,9,4,6) contains two nonisomorphic strongly regular graphs. Since the GI AQA uses $\nu\lceil\log_2 \nu\rceil$ qubits, simulation of its quantum dynamics on 16-vertex graphs requires 64 qubits. This is hopelessly beyond the 20–22 qubit limit for such simulations discussed in the introduction to Sec. IV. As noted there, this hard limit restricts our simulations to graphs with no more than seven vertices. Now the number of connected strongly regular graphs with $\nu = 4,5,6,7$ is 3,2,5,1, respectively. For each of the values $\nu = 4,5,6$, the strongly regular graphs are nonisomorphic as desired, however, each graph belongs to a *different* family. In light of the above remarks, the simulations reported in this section are restricted to pairs of connected nonisomorphic strongly regular graphs with $\nu = 4,5,6$. Ideally we would have simulated the GI AQA on the above pair of 16-vertex strongly regular graphs, however, the realities of simulating quantum systems on a classical computer made this test well beyond reach.

$N = 4$. In Fig. 6 we show two strongly regular four-vertex graphs G and G'. The parameters for G are $\nu = 4$, $k = 2$, $\lambda = 0$, and $\mu = 2$; while for G' they are $\nu = 4$, $k = 3$, $\lambda = 2$, and $\mu = 0$. It is clear that G and G' are nonisomorphic since they contain an unequal number of edges and different degree sequences. The adjacency matrices A and A' for G and G' are, respectively,

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}; \qquad A' = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}. \tag{23}$$

Numerical simulation of the GI AQA dynamics finds a nonzero final ground-state energy $E_{gs} = 4$ and so the GI AQA correctly distinguishes G and G' as nonisomorphic.

$N = 5$. In Fig. 7 we show two strongly regular five-vertex graphs G and G'. The parameters for G are $\nu = 5$, $k = 2$, $\lambda = 0$, and $\mu = 1$; while for G' they are $\nu = 5$, $k = 4$, $\lambda = 3$, and $\mu = 0$. It is clear that G and G' are nonisomorphic since they contain an unequal number of edges and different degree sequences. The adjacency matrices A and A' for G and G' are,

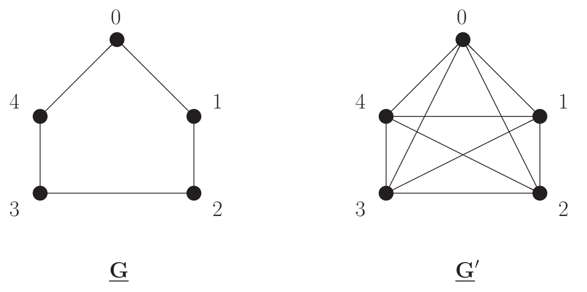

FIG. 7. Two five-vertex nonisomorphic strongly regular graphs G and G'.

respectively,

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix};$$
$$A' = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}. \tag{24}$$

Numerical simulation of the GI AQA dynamics finds a nonzero final ground-state energy $E_{gs} = 10$ and so the GI AQA correctly distinguishes G and G' as nonisomorphic.

$N = 6$. In Fig. 8 we show two strongly regular six-vertex graphs G and G'. The parameters for G are $\nu = 6$, $k = 3$, $\lambda = 0$, and $\mu = 3$; while for G' they are $\nu = 6$, $k = 4$, $\lambda = 2$, and $\mu = 4$. It is clear that G and G' are nonisomorphic since they contain an unequal number of edges and different degree sequences. The adjacency matrices A and A' for G and G' are, respectively,

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix};$$
$$A' = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}. \tag{25}$$

Numerical simulation of the GI AQA dynamics finds a nonzero final ground-state energy $E_{gs} = 10$ and so the GI AQA correctly distinguishes G and G' as nonisomorphic.

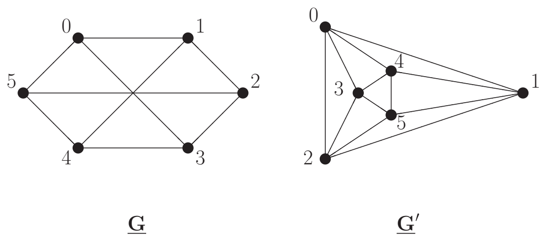

FIG. 8. Two 6-vertex non-isomorphic strongly regular graphs G and G'.

C. Graph automorphisms

As noted in the introduction of Sec. IV, we can find the automorphism group Aut(G) of a graph G using the GI AQA by considering a GI instance with $G' = G$. Here the self-isomorphisms are permutations of the vertices of G that map $G \to G$ while preserving adjacency. Since G is always isomorphic to itself, the final ground-state energy will vanish: $E_{gs} = 0$. The set of CBS $|s_b\rangle$ that span the final ground-state subspace give rise to a set of binary strings s_b that determine the integer strings $s = s_0 \cdots s_{N-1}$ (see Sec. II B) that then determine the permutations $\pi(s)$,

$$\pi(s) = \begin{pmatrix} 0 & \cdots & i & \cdots & N-1 \\ s_0 & \cdots & s_i & \cdots & s_{N-1} \end{pmatrix}, \tag{26}$$

which are all the elements of Aut(G). By construction, the order of Aut(G) is equal to the degeneracy of the final ground-state subspace. In this section we apply the GI AQA to the (i) cycle graphs $C_4, \ldots, C_7$; (ii) grid graph $G_{2,3}$; and (iii) wheel graph W_7, and show that it correctly determines the automorphism group for all of these graphs.

1. Cycle graphs

A *walk* W in a graph is an alternating sequence of vertices and edges $x_0, e_1, x_1, e_2, \ldots, e_l, x_l$, where the edge e_i connects x_{i-1} and x_i for $0 < i \leqslant l$. A walk W is denoted by the sequence of vertices it traverses $W = x_0 x_1 \cdots x_l$. Finally, a walk $W = x_0 x_1 \cdots x_l$ is a *cycle* if $l \geqslant 3$, $x_0 = x_l$, and the vertices x_i with $0 < i < l$ are distinct from each other and x_0. A cycle with n vertices is denoted C_n.

The automorphism group of the cycle graph C_n is the dihedral group D_n [16]. The order of D_n is $2n$, and it is generated by the two elements α and β that satisfy the following relations:

$$\alpha^n = e; \qquad \beta^2 = e; \qquad \alpha\beta = \beta\alpha^{n-1}, \tag{27}$$

where e is the identity element. Because α and β are generators of D_n, each element g of D_n can be written as a product of appropriate powers of α and β:

$$g = \alpha^i \beta^j \qquad (0 \leqslant i \leqslant n-1;\; 0 \leqslant j \leqslant 1). \tag{28}$$

We now use the GI AQA to find the automorphism group of the cycle graphs C_n for $4 \leqslant n \leqslant 7$. We will see that the GI AQA correctly determines Aut(C_n) $= D_n$ for these graphs. We will work out C_4 in detail, and then give more abbreviated presentations for the remaining cycle graphs as their analysis is identical.

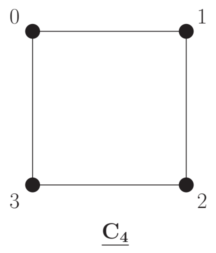

FIG. 9. Cycle graph C_4.

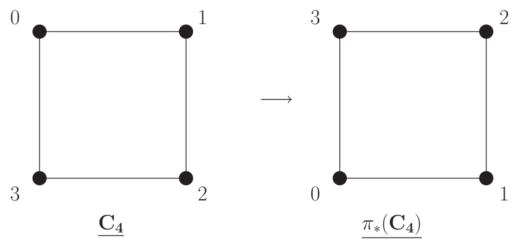

FIG. 10. Transformation of C_4 produced by the permutation π_*.

$N = 4$. The cycle graph C_4 appears in Fig. 9. It contains four vertices and four edges, has degree sequence {2,2,2,2}, and adjacency matrix

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}. \tag{29}$$

Numerical simulation of the GI AQA applied to the GI instance with $G = C_4$ and $G' = G$ found a vanishing final ground-state energy $E_{gs} = 0$. The GI AQA thus correctly identifies C_4 as being isomorphic to itself. The simulation also found that the final ground-state subspace is eightfold degenerate. Table I lists the integer strings $s = s_0s_1s_2s_3$ that result from the eight CBS $|s_b\rangle$ that span the final ground-state subspace (see Secs. II B and III). Each integer string s determines the bottom row of a permutation $\pi(s)$ [see Eq. (26)]. We explicitly show that $\pi(s)$, for the integer string $s = 3210$, is an automorphism of C_4. The reader can repeat this analysis to show that the seven remaining integer strings in Table I also give rise to automorphisms of C_4. Note that the CBS that span the final ground-state subspace determine *all* the graph automorphisms of C_4. This follows from the manner in which the GI AQA is constructed since each automorphism of C_4 must give rise to a CBS with vanishing energy, and each CBS in the final ground-state subspace gives rise to an automorphism of C_4. These automorphisms form a group Aut(C_4) and the GI AQA has found that the order of Aut(C_4) is 8 which is the same as the order of the dihedral group D_4.

TABLE I. Automorphism group Aut(C_4) of the cycle graph C_4 as found by the GI AQA. The first row lists the integer strings $s = s_0s_1s_2s_3$ determined by the labels of the eight CBS $|s_b\rangle$ that span the final ground-state subspace (see text). Each string s determines a graph automorphism $\pi(s)$ via Eq. (26). The second row associates each integer string s in the first row with a graph automorphism $\pi(s)$. It identifies the two strings that give rise to the graph automorphisms α and β that generate Aut(C_4), and writes each graph automorphism $\pi(s)$ as a product of an appropriate power of α and β. Note that e is the identity automorphism, and the product notation assumes the rightmost factor acts first.

$s = s_0s_1s_2s_3$	3012	2301	1230	0123	0321	3210	2103	1032
$\pi(s)$	α	α^2	α^3	$\alpha^4 = e$	β	$\alpha\beta$	$\alpha^2\beta$	$\alpha^3\beta$

The permutation $\pi_* \equiv \pi(s = 3210)$ is

$$\pi(3210) = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 3 & 2 & 1 & 0 \end{pmatrix}, \tag{30}$$

and the associated permutation matrix $\sigma_* \equiv \sigma(3210)$ is

$$\sigma(3210) = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}. \tag{31}$$

Under π_*, C_4 is transformed to $\pi_*(C_4)$, which is shown in Fig. 10. It is clear from Fig. 10 that x and y are adjacent in C_4 if and only if $\pi_{*,x}$ and $\pi_{*,y}$ are adjacent in $\pi_*(C_4)$. Thus π_* is a permutation of the vertices of C_4 that preserves adjacency and so is a graph automorphism of C_4. We can also show this by demonstrating that $\sigma_* A \sigma_*^T = A$. Using Eqs. (29) and (31) it is easy to show that

$$\sigma_* A \sigma_*^T = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}. \tag{32}$$

This agrees with the adjacency of edges in $\pi_*(C_4)$ appearing in Fig. 10, and comparison with Eq. (29) shows that $\sigma_* A \sigma_*^T = A$, confirming that π_* is an automorphism of C_4.

We now show that Aut(C_4) is isomorphic to the dihedral group D_4 by showing that the automorphisms $\pi(3012) \equiv \alpha$ and $\pi(0321) \equiv \beta$ generate Aut(C_4), and satisfy the generator relations [Eqs. (27)] for D_4. The second row of Table I establishes that α and β are the generators of Aut(C_4) as it shows that each element of Aut(C_4) is a product of an appropriate power of α and β, and all possible products of powers of α and β appear in that row. Now notice that

$$\alpha = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 3 & 0 & 1 & 2 \end{pmatrix} \tag{33}$$

corresponds to a clockwise rotation of C_4 by 90° (see Fig. 11). Thus four applications of α corresponds to a 360° rotation of C_4 which leaves it invariant. Thus $\alpha^4 = e$. This can also be checked by composing α with itself four times using Eq. (33). This establishes the first of the generator relations in Eqs. (27). Similarly,

$$\beta = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 3 & 2 & 1 \end{pmatrix} \tag{34}$$

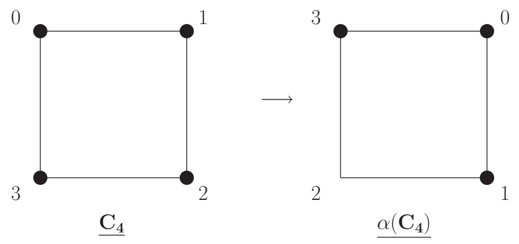

FIG. 11. Transformation of C_4 produced by the automorphism α.

corresponds to reflection of C_4 about the diagonal passing through vertices 0 and 2 (see Fig. 12). Thus two applications of β leaves C_4 invariant, and so $\beta^2 = e$. This establishes the second of the generator relations in Eqs. (27). Finally, to show the third generator relation $\alpha\beta = \beta\alpha^3$, we simply evaluate both sides of this relation and compare results. Using Table I we find that

$$\alpha\beta = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 3 & 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 3 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 3 & 2 & 1 & 0 \end{pmatrix}, \tag{35}$$

$$\beta\alpha^3 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 3 & 2 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 2 & 3 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 3 & 2 & 1 & 0 \end{pmatrix}. \tag{36}$$

It is clear that $\alpha\beta$ does equal $\beta\alpha^3$. Thus we have shown that α and β (i) generate Aut(C_4) and (ii) satisfy the generator relations [Eqs. (27)] for the dihedral group D_4, and so generate a group isomorphic to D_4. In summary, we have shown that the GI AQA found all eight graph automorphisms of C_4, and that the group formed from these automorphisms is isomorphic to the dihedral group D_4, which is the correct automorphism group for C_4.

$N = 5$. The cycle graph C_5 appears in Fig. 13. It has five vertices and five edges, degree sequence {2,2,2,2,2}, and adjacency matrix

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{37}$$

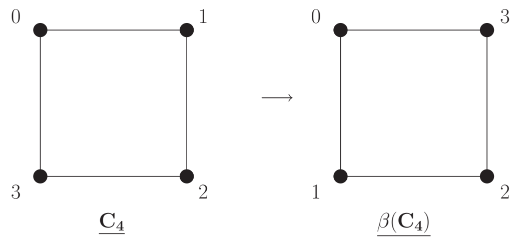

FIG. 12. Transformation of C_4 produced by the automorphism β.

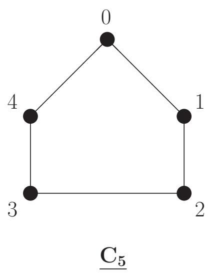

FIG. 13. Cycle graph C_5.

Numerical simulation of the GI AQA applied to the GI instance with $G = C_5$ and $G' = G$ found a vanishing final ground-state energy $E_{gs} = 0$. The GI AQA thus correctly identifies C_5 as being isomorphic to itself. The simulation also found that the final ground-state subspace is tenfold degenerate. Table II lists the integer strings $s = s_0 \cdots s_4$ that result from the ten CBS $|s_b\rangle$ that span the final ground-state subspace (see Secs. II B and III). Each integer string s fixes the bottom row of a permutation $\pi(s)$ [see Eq. (26)], which is a graph automorphism of C_5. The demonstration of this is identical to the demonstration given for C_4 and so will not be repeated here. Just as for Aut(C_4), the graph automorphisms in Table II are all the elements of Aut(C_5), which is seen to have order 10. Note that this is the same as the order of the dihedral group D_5.

We now show that Aut(C_5) is isomorphic to the dihedral group D_5 by showing that the graph automorphisms $\alpha = \pi(40123)$ and $\beta = \pi(04321)$ generate Aut(C_5), and satisfy the generator relations [Eqs. (27)] for D_5. The second row of Table II establishes that α and β are the generators of Aut(C_5) as it shows that each element of Aut(C_5) is a product of an appropriate power of α and β, and that all possible products of powers of α and β appear in that row. Following the discussion for C_4, it is a simple matter to show that α corresponds to a 72° clockwise rotation of C_5. Thus five applications of α rotates C_5 by 360°, which leaves it invariant. Thus $\alpha^5 = e$, which is the first of the generator relations in Eqs. (27). Similarly, β can be shown to correspond to a reflection of C_5 about a vertical axis passing through vertex 0 in Fig. 13. Thus two applications of β leave C_5 invariant. Thus $\beta^2 = e$, which

TABLE II. Automorphism group Aut(C_5) of the cycle graph C_5 as found by the GI AQA. The odd rows list the integer strings $s = s_0 \cdots s_4$ determined by the labels of the ten CBS $|s_b\rangle$ that span the final ground-state subspace (see text). Each string s determines a graph automorphism $\pi(s)$ via Eq. (26). Each even row associates each integer string s in the odd row preceding it with a graph automorphism $\pi(s)$. It identifies the two strings that give rise to the graph automorphisms α and β that generate Aut(C_5), and writes each graph automorphism $\pi(s)$ as a product of an appropriate power of α and β. Note that e is the identity automorphism, and the product notation assumes the rightmost factor acts first.

$s = s_0 \cdots s_4$	40123	34012	23401	12340	01234
$\pi(s)$	α	α^2	α^3	α^4	$\alpha^5 = e$
$s = s_0 \cdots s_4$	04321	10432	21043	32104	43210
$\pi(s)$	β	$\alpha\beta$	$\alpha^2\beta$	$\alpha^3\beta$	$\alpha^4\beta$

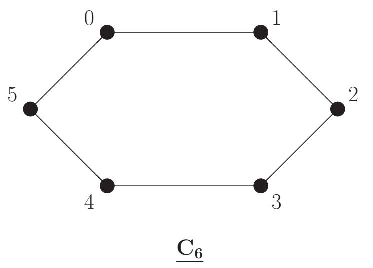

FIG. 14. Cycle graph C_6.

is the second generator relation in Eqs. (27). Finally, using Table II, direct calculation as in Eqs. (35) and (36) shows that $\alpha\beta = \beta\alpha^4$, which establishes the final generator relation in Eqs. (27). We see that α and β generate Aut(C_5) and satisfy the generator relations for D_5 and so generate a ten element group isomorphic to D_5. In summary, we have shown that the GI AQA found all ten graph automorphisms of C_5, and that the group formed from these automorphisms is isomorphic to the dihedral group D_5, which is the correct automorphism group for C_5.

$N = 6$. The cycle graph C_6 appears in Fig. 14. It has six vertices and six edges, degree sequence {2,2,2,2,2,2}, and adjacency matrix

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{38}$$

Numerical simulation of the GI AQA applied to the GI instance with $G = C_6$ and $G' = G$ found a vanishing final ground-state energy $E_{gs} = 0$. The GI AQA thus correctly identifies C_6 as being isomorphic to itself. The simulation also found that the final ground-state subspace is 12-fold degenerate. Table III lists the integer strings $s = s_0 \cdots s_5$ that result from the 12 CBS $|s_b\rangle$ that span the final ground-state subspace (see Secs. II B and III). Each integer string s fixes the bottom row of a permutation $\pi(s)$ [see Eq. (26)], which

TABLE III. Automorphism group Aut(C_6) of the cycle graph C_6 as found by the GI AQA. The first and third rows list the integer strings $s = s_0 \cdots s_5$ determined by the labels of the 12 CBS $|s_b\rangle$ that span the final ground-state subspace (see text). Each string s determines a graph automorphism $\pi(s)$ via Eq. (26). The second and fourth rows associate each integer string s in the first and third rows with a graph automorphism $\pi(s)$. They also identify the two strings that give rise to the graph automorphisms α and β that generate Aut(C_6), and write each graph automorphism $\pi(s)$ as a product of an appropriate power of α and β. Note that e is the identity automorphism, and the product notation assumes the rightmost factor acts first.

$s = s_0 \cdots s_5$	501234	450123	345012	234501	123450	012345
$\pi(s)$	α	α^2	α^3	α^4	α^5	$\alpha^6 = e$
$s = s_0 \cdots s_5$	105432	210543	321054	432105	543210	054321
$\pi(s)$	β	$\alpha\beta$	$\alpha^2\beta$	$\alpha^3\beta$	$\alpha^4\beta$	$\alpha^5\beta$

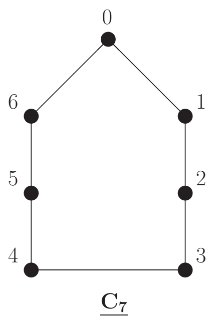

FIG. 15. Cycle graph C_7.

is a graph automorphism of C_6. The demonstration of this is identical to the demonstration given for C_4 and so will not be repeated here. Just as for Aut(C_4), the graph automorphisms in Table III are all the elements of Aut(C_6), which is seen to have order 12. Note that this is the same as the order of the dihedral group D_6.

We now show that Aut(C_6) is isomorphic to the dihedral group D_6 by showing that the graph automorphisms $\alpha = \pi(501234)$ and $\beta = \pi(105432)$ generate Aut(C_6), and satisfy the generator relations [Eqs. (27)] for D_6. The second and fourth rows of Table III establish that α and β are the generators of Aut(C_6) as they show that each element of Aut(C_6) is a product of an appropriate power of α and β, and that all possible products of powers of α and β appear in these two rows. Following the discussion for C_4, it is a simple matter to show that α corresponds to a 60° clockwise rotation of C_6. Thus six applications of α rotates C_6 by 360°, which leaves it invariant. Thus $\alpha^6 = e$, which is the first of the generator relations in Eqs. (27). Similarly, β can be shown to correspond to a reflection of C_6 about a vertical axis that bisects C_6. Thus two applications of β leave C_6 invariant. Thus $\beta^2 = e$, which is the second generator relation in Eqs. (27). Finally, using Table III, direct calculation as in Eqs. (35) and (36) shows that $\alpha\beta = \beta\alpha^5$, which establishes the final generator relation in Eqs. (27). We see that α and β generate Aut(C_6) and satisfy the generator relations for D_6 and so generate a 12 element group isomorphic to D_6. In summary, we have shown that the GI AQA found all 12 graph automorphisms of C_6, and that the group formed from these automorphisms is isomorphic to the dihedral group D_6, which is the correct automorphism group for C_6.

$N = 7$. The cycle graph C_7 appears in Fig. 15. It has seven vertices and seven edges, degree sequence {2,2,2,2,2,2,2}, and adjacency matrix

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{39}$$

Numerical simulation of the GI AQA applied to the GI instance with $G = C_7$ and $G' = G$ found a vanishing final

TABLE IV. Automorphism group Aut(C_7) of the cycle graph C_7 as found by the GI AQA. The odd rows list the integer strings $s = s_0 \cdots s_6$ determined by the labels of the 14 CBS $|s_b\rangle$ that span the final ground-state subspace (see text). Each string s determines a graph automorphism $\pi(s)$ via Eq. (26). Each even row associates each integer string s in the odd row preceding it with a graph automorphism $\pi(s)$. They also identify the two strings that give rise to the graph automorphisms α and β that generate Aut(C_7), and write each graph automorphism $\pi(s)$ as a product of an appropriate power of α and β. Note that e is the identity automorphism, and the product notation assumes the rightmost factor acts first.

$s = s_0 \cdots s_6$	6012345	5601234	4560123	3456012
$\pi(s)$	α	α^2	α^3	α^4
$s = s_0 \cdots s_6$	2345601	1234560	0123456	
$\pi(s)$	α^5	α^6	$\alpha^7 = e$	
$s = s_0 \cdots s_6$	0654321	1065432	2106543	3210654
$\pi(s)$	β	$\alpha\beta$	$\alpha^2\beta$	$\alpha^3\beta$
$s = s_0 \cdots s_6$	4321065	5432106	6543210	
$\pi(s)$	$\alpha^4\beta$	$\alpha^5\beta$	$\alpha^6\beta$	

ground-state energy $E_{gs} = 0$. The GI AQA thus correctly identifies C_7 as being isomorphic to itself. The simulation also found that the final ground-state subspace is 14-fold degenerate. Table IV lists the integer strings $s = s_0 \cdots s_6$ that result from the 14 CBS $|s_b\rangle$ that span the final ground-state subspace (see Secs. II B and III). Each integer string s fixes the bottom row of a permutation $\pi(s)$ [see Eq. (26)], which is a graph automorphism of C_7. The demonstration of this is identical to the demonstration given for C_4 and so will not be repeated here. Just as for Aut(C_4), the graph automorphisms in Table IV are all the elements of Aut(C_7), which is seen to have order 14. Note that this is the same as the order of the dihedral group D_7.

We now show that Aut(C_7) is isomorphic to the dihedral group D_7 by showing that the graph automorphisms $\alpha = \pi(6012345)$ and $\beta = \pi(0654321)$ generate Aut(C_7), and satisfy the generator relations [Eqs. (27)] for D_7. The second and fourth rows of Table IV establish that α and β are the generators of Aut(C_7) as they show that each element of Aut(C_7) is a product of an appropriate power of α and β, and that all possible products of powers of α and β appear in these two rows. Following the discussion for C_4, it is a simple matter to show that α corresponds to a $2\pi/7$ radian clockwise rotation of C_7. Thus seven applications of α rotates C_7 by 360°, which leaves it invariant. Thus $\alpha^7 = e$, which is the first of the generator relations in Eqs. (27). Similarly, β can be shown to correspond to a reflection of C_7 about a vertical axis that passes through vertex 0 in Fig. 15. Thus two applications of β leave C_7 invariant. Thus $\beta^2 = e$, which is the second generator relation in Eqs. (27). Finally, using Table IV, direct calculation as in Eqs. (35) and (36) shows that $\alpha\beta = \beta\alpha^6$, which establishes the final generator relation in Eqs. (27). We see that α and β generate Aut(C_7) and satisfy the generator relations for D_7 and so generate a 14 element group isomorphic to D_7. In summary, we have shown that the GI AQA found all 14 graph automorphisms of C_7, and that the group formed from these automorphisms is isomorphic to the dihedral group D_7, which is the correct automorphism group for C_7.

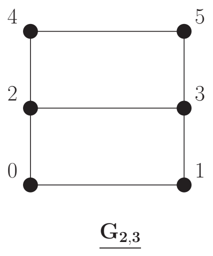

FIG. 16. Grid graph $G_{2,3}$.

2. Grid graph $G_{2,3}$

The grid graph $G_{2,3}$ appears in Fig. 16. It has six vertices and seven edges, degree sequence {3,3,2,2,2,2}, and adjacency matrix

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}. \tag{40}$$

$G_{2,3}$ has two reflection symmetries. The first is reflection about the horizontal line passing through vertices 2 and 3, and the second is reflection about a vertical line that bisects $G_{2,3}$. Let α and β denote the permutation of vertices that these reflections produce. It follows from their definition that $\alpha = \pi(452301)$ and $\beta = \pi(103254)$. They are the generators of the automorphism group of $G_{2,3}$: Aut($G_{2,3}$) = $\langle \alpha,\beta \rangle$. As reflections, they satisfy $\alpha^2 = \beta^2 = e$. Applying these reflections successively gives $\alpha\beta = \pi(543210)$, which is a fourth symmetry of $G_{2,3}$. Applying these reflections in the reverse order, it is easy to verify that $\alpha\beta = \beta\alpha$. Thus Aut($G_{2,3}$) is a four element Abelian group generated by the reflections α and β. Let us now compare this with the results found by the GI AQA.

Numerical simulation of the GI AQA applied to the GI instance with $G = G_{2,3}$ and $G' = G$ found a vanishing final ground-state energy $E_{gs} = 0$. The GI AQA thus correctly identifies $G_{2,3}$ as being isomorphic to itself. The simulation also found that the final ground-state subspace is fourfold degenerate. Table V lists the integer strings $s = s_0 \cdots s_5$ that result from the four CBS $|s_b\rangle$ that span the final ground-state subspace (see Secs. II B and III). Each integer string s fixes the bottom row of a permutation $\pi(s)$ [see Eq. (26)], which is a graph automorphism of $G_{2,3}$. The demonstration of this is identical to the demonstration given for C_4 and so will not be repeated here. Notice that the GI AQA found the graph automorphisms $\pi(452301)$ and $\pi(103254)$ which implement the reflection symmetries of $G_{2,3}$ described above. As they implement reflections, it follows that $\alpha^2 = \beta^2 = e$. The GI AQA also found the graph automorphism $\pi(543210)$, which is the composite symmetry $\alpha\beta$ described above. Using Table V,

TABLE V. Automorphism group Aut($G_{2,3}$) of the grid graph $G_{2,3}$ as found by the GI AQA. The first row lists the integer strings $s = s_0 \cdots s_5$ determined by the labels of the four CBS $|s_b\rangle$ that span the final ground-state subspace (see text). Each string s determines a graph automorphism $\pi(s)$ via Eq. (26). The second row associates each integer string s in the first row with a graph automorphism $\pi(s)$. They also identify the two strings that give rise to the graph automorphisms α and β that generate Aut($G_{2,3}$), and write each graph automorphism $\pi(s)$ as a product of an appropriate power of α and β. Note that e is the identity automorphism, and the product notation assumes the rightmost factor acts first.

$s = s_0 \cdots s_5$	452301	103254	012345	543210
$\pi(s)$	α	β	$\alpha^2 = \beta^2 = e$	$\alpha\beta$

direct calculation as in Eqs. (35) and (36) shows that $\alpha\beta = \beta\alpha$. Thus the GI AQA correctly found the two generators α and β of Aut($G_{2,3}$), correctly determined all four elements of Aut($G_{2,3}$), and correctly determined that Aut($G_{2,3}$) is an Abelian group. In summary, the GI AQA correctly determined the automorphism group of $G_{2,3}$.

3. *Wheel graph W_7*

The wheel graph W_7 appears in Fig. 17. It has seven vertices and 12 edges, degree sequence {6,3,3,3,3,3,3}, and adjacency matrix

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}. \qquad (41)$$

The automorphism group Aut(W_7) is isomorphic to the dihedral group D_6 which, as we have seen, has 12 elements and is generated by two graph automorphisms α and β that satisfy the generator relations in Eqs. (27) with $n = 6$ and respectively, rotate W_7 clockwise about vertex 6 by 60°, and reflect W_7 about a vertical axis through vertex 6. The graph automorphisms of W_7 thus fix vertex 6. Let us now compare this with the results found by the GI AQA.

Numerical simulation of the GI AQA applied to the GI instance with $G = W_7$ and $G' = G$ found a vanishing final

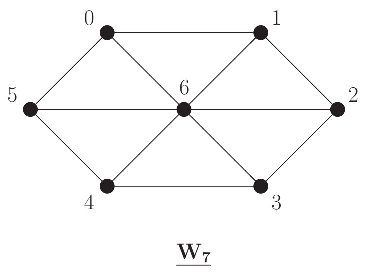

FIG. 17. Wheel graph W_7.

TABLE VI. Automorphism group Aut(W_7) of the wheel graph W_7 as found by the GI AQA. The odd rows list the integer strings $s = s_0 \cdots s_6$ determined by the labels of the 12 CBS $|s_b\rangle$ that span the final ground-state subspace (see text). Each string s determines a graph automorphism $\pi(s)$ via Eq. (26). Each even row associates each integer string s in the odd row preceding it with a graph automorphism $\pi(s)$. They also identify the two strings that give rise to the graph automorphisms α and β that generate Aut(W_7), and write each graph automorphism $\pi(s)$ as a product of an appropriate power of α and β. Note that e is the identity automorphism, and the product notation assumes the rightmost factor acts first.

$s = s_0 \cdots s_6$	5012346	4501236	3450126
$\pi(s)$	α	α^2	α^3
$s = s_0 \cdots s_6$	2345016	1234506	0123456
$\pi(s)$	α^4	α^5	$\alpha^6 = e$
$s = s_0 \cdots s_6$	1054326	2105436	3210546
$\pi(s)$	β	$\alpha\beta$	$\alpha^2\beta$
$s = s_0 \cdots s_6$	4321056	5432106	0543216
$\pi(s)$	$\alpha^3\beta$	$\alpha^4\beta$	$\alpha^5\beta$

ground-state energy $E_{gs} = 0$. The GI AQA thus correctly identifies W_7 as being isomorphic to itself. The simulation also found that the final ground-state subspace is 12-fold degenerate. Table VI lists the integer strings $s = s_0 \cdots s_6$ that result from the 12 CBS $|s_b\rangle$ that span the final ground-state subspace (see Secs. II B and III). Each integer string s fixes the bottom row of a permutation $\pi(s)$ [see Eq. (26)], which is a graph automorphism of W_7. The demonstration of this is identical to the demonstration given for C_4 and so will not be repeated here. Just as for Aut(C_4), the graph automorphisms in Table VI are all the elements of Aut(W_7), which is seen to have order 12. Note that this is the same as the order of the dihedral group D_6.

We now show that Aut(W_7) is isomorphic to the dihedral group D_6 by showing that the graph automorphisms $\alpha = \pi(5012346)$ and $\beta = \pi(1054326)$ generate Aut(W_7), and satisfy the generator relations [Eqs. (27)] for D_6. The second and fourth rows of Table VI establish that α and β are the generators of Aut(W_7) as they show that each element of Aut(W_7) is a product of an appropriate power of α and β, and that all possible products of powers of α and β appear in these two rows. Following the discussion for C_4, it is a simple matter to show that α corresponds to a 60° clockwise rotation about vertex 6 of W_7. Thus six applications of α rotates W_7 by 360°, which leaves it invariant. Thus $\alpha^6 = e$, which is the first of the generator relations in Eqs. (27). Similarly, β can be shown to correspond to a reflection of W_7 about a vertical axis passing through vertex 6 of W_7. Thus two applications of β leave W_7 invariant. Thus $\beta^2 = e$, which is the second generator relation in Eqs. (27). Finally, using Table VI, direct calculation as in Eqs. (35) and (36) shows that $\alpha\beta = \beta\alpha^5$, which establishes the final generator relation in Eqs. (27). We see that α and β generate Aut(W_7) and satisfy the generator relations for D_6 and so generate a 12 element group isomorphic to D_6. In summary, we have shown that the GI AQA found all 12 graph automorphisms of W_7, and that the group formed from these automorphisms is isomorphic to the dihedral group D_6, which is the correct automorphism group for W_7.

V. EXPERIMENTAL IMPLEMENTATION

In this section we express the GI problem Hamiltonian H_P in a form more suitable for experimental implementation. We saw in Sec. II C that the eigenvalues of H_P are given by the cost function $C(s)$, which is reproduced here for convenience:

$$C(s) = C_1(s) + C_2(s) + C_3(s), \tag{42}$$

with

$$C_1(s) = \sum_{i=0}^{N-1} \sum_{\alpha=N}^{M} \delta_{s_i,\alpha}, \tag{43}$$

$$C_2(s) = \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} \delta_{s_i,s_j}, \tag{44}$$

$$C_3(s) = \|\sigma(s)A\sigma^T(s) - A'\|_i. \tag{45}$$

Here $s = s_0 \cdots s_{N-1}$ is the integer string derived from the binary string $s_b = (s_0 \cdots s_{U-1}) \cdots (s_{(N-1)U} \cdots s_{NU-1})$ via Eqs. (6) and (7) with $U = \lceil \log_2 N \rceil$. The matrix $\sigma(s)$ was defined in Eq. (8) as

$$\sigma_{i,j}(s) = \begin{cases} 0, & \text{if} \quad s_j > N-1 \\ \delta_{i,s_j}, & \text{if} \quad 0 \leqslant s_j \leqslant N-1. \end{cases} \tag{46}$$

Note that $\sigma_{i,j}(s)$ can be written more compactly as

$$\sigma_{i,j}(s) = \delta_{i,s_j} \prod_{\alpha=N}^{M} (1 - \delta_{s_j,\alpha}). \tag{47}$$

We see from Eqs. (42)–(45) and (47) that the s dependence of $C(s)$ enters through the Kronecker deltas. This type of s dependence is not well suited for experimental implementation and so our task is to find a more convenient form for the Kronecker delta.

We begin with $\delta_{a,b}$ in the case where $a,b \in \{0,1\}$. Here we write

$$\delta_{a,b} = (a+b-1)^2 = \begin{cases} 0 & (a \neq b) \\ 1 & (a = b), \end{cases} \tag{48}$$

which can be checked by inserting values for a and b. Now consider $\delta_{s,k}$ when s and k are U-bit integers. The binary decompositions of s and k are

$$s = \sum_{i=0}^{U-1} s_i \, (2)^i \,, \tag{49}$$

$$k = \sum_{i=0}^{U-1} k_i \, (2)^i \,. \tag{50}$$

For s and k to be equal, all their corresponding bits must be equal. Thus we can write

$$\begin{aligned} \delta_{s,k} &= \prod_{i=0}^{U-1} \delta_{s_i,k_i} \\ &= \prod_{i=0}^{U-1} (s_i + k_i - 1)^2 = \begin{cases} 1 & (\text{all } s_i = k_i) \\ 0 & (\text{some } s_i \neq k_i). \end{cases} \end{aligned} \tag{51}$$

Equation (51) allows each Kronecker delta appearing in $C(s)$ to be converted to a polynomial in the components of the integer string s. From Eqs. (43) and (44) we see that $C_1(s)$ and $C_2(s)$ are each $2U$ local. For $C_3(s)$ we must write $\sigma(s)A\sigma^T(s)$ in a form that makes the Kronecker deltas explicit. Using Eq. (47), we have

$$\begin{aligned} \sigma(s)A\sigma^T(s) &= \sum_{i,j=0}^{N-1} \sigma_{li}(s) A_{ij} \sigma_{mj}(s) \\ &= \sum_{i,j=0}^{N-1} \left\{ \delta_{l,s_i} \prod_{\alpha=N}^{M} (1 - \delta_{s_i,\alpha}) \right\} A_{ij} \\ &\quad \times \left\{ \delta_{m,s_j} \prod_{\beta=N}^{M} (1 - \delta_{s_j,\beta}) \right\}. \end{aligned} \tag{52}$$

Inserting Eq. (51) into Eq. (52), we see that $\sigma(s)A\sigma^T(s) - A'$ is $4U(M-N+1)$ local. If we use the L_1 norm (L_2 norm) in Eq. (45), then we see that $C_3(s)$ is $4U(M-N+1)$ local [$8U(M-N+1)$ local].

The number of terms in $C(s)$, and thus in H_P, follows straightforwardly from Eqs. (43)–(45). Starting with $C_1(s)$, there is a term for each value of i and α that appears in the sum. There are thus $T_1 = N(M-N+1)$ terms in $C_1(s)$. From Eq. (5), $M = 2^U - 1$, where $U = \lceil \log_2 N \rceil$ and N is the number of vertices in each graph appearing in the GI instance. It follows from the definition of U that $M < 2N$ and so $T_1 < N(N+1)$. Recall from Sec. III that the number of qubits L_N needed for an N-vertex GI instance is $L_N = N\lceil \log_2 N \rceil$. Thus $T_1 < (L_N/\lceil \log_2 N \rceil)(L_N/\lceil \log_2 N \rceil + 1) < CL_N^2$, where C is an appropriate constant. Thus $T_1(L_N) = O(L_N^2)$. A similar analysis shows that the number of terms in $C_2(s)$ is $T_2(L_N) = O(L_N^2)$. Finally, $C_3(s)$ is the L_i norm of an $N \times N$ matrix. For the L_1 norm used in our simulations, the number of terms in $C_3(s)$ is $T_3 = N^2$. Following the above analysis, this gives $T_3(L_N) = O(L_N^2)$. Putting everything together gives that the total number of terms associated with $C(s)$, and so also H_P, is $T(L_N) = T_1(L_N) + T_2(L_N) + T_3(L_N) = O(L_N^2)$. The initial Hamiltonian H_i contains one term for each qubit [see Eq. (15)] and so the number of terms in H_i is L_N. Thus the total number of terms in the full time-dependent Hamiltonian $H(t)$ is $O(L_N^2)$ and so scales quadratically with the number of qubits L_N.

Clearly, the degree of difficulty associated with experimentally implementing a quantum algorithm depends strongly on the architecture of the hardware on which it is to be run. The simplest situation would be an architecture which allows an arbitrary number of qubits to be simultaneously coupled, independently of where they were located on the processor. Unfortunately, such a hardware architecture does not presently exist. For hardware platforms designed to run adiabatic quantum optimization algorithms, the D-Wave hardware is furthest along [29]. Each qubit on the D-Wave processor couples to at most six neighboring qubits, and only two-qubit Ising coupling interactions are possible. The initial Hamiltonian H_i [see Eq. (15)] is easily programed onto the hardware. However, a problem Hamiltonian H_P which is not of Ising form is more challenging, requiring an embedding procedure that (i) reduces all k-local interactions with $k \geqslant 3$

to 2-local form; and (ii) two-qubit coupling interactions that match the hardware's Chimera coupling graph. A procedure for carrying out this reduction based on Ref. [15] is described in Appendix B. Alternative approaches appear in Refs. [30] and [31].

VI. SUBGRAPH ISOMORPHISM PROBLEM

An instance of the SGI problem consists of an N-vertex graph G and an n-vertex graph H with $n \leqslant N$. The question to be answered is whether G contains an n-vertex subgraph that is isomorphic to H. The SGI problem is known to be NP-complete [32] and is believed to be more difficult to solve than the GI problem. Here we show how an instance of SGI can be converted into an instance of a COP whose cost function has a minimum value that vanishes when G contains a subgraph isomorphic to H, and is greater than zero otherwise. The SGI cost function will be seen to be a natural generalization of the GI cost function given in Eqs. (9)–(12). The SGI COP can then be solved using adiabatic quantum evolution as was done for the GI problem.

As with the GI problem, we would like to determine whether there exists an isomorphism π of G that produces a new graph $\pi(G)$ that contains H as a subgraph. Just as with the GI problem, we consider linear maps $\sigma(s)$ [see Eqs. (6)–(8)] that transform the adjacency matrix A of G to $\tilde{A}(s) = \sigma(s) A \sigma^T(s)$. We then search $\pi(G)$ to determine whether there is a subset of n vertices that yields a subgraph that is equal to H.

To begin the process of converting an SGI instance into an instance of a COP, let (i) α label all the $\binom{N}{n}$ ways of choosing n vertices from the N vertices in $\pi(G)$; and (ii) $|i\rangle$ ($|\alpha_i\rangle$) be an n-component (N-component) vector whose ith (α_ith) component is 1, and all other components are 0. Thus α labels the choice $(\alpha_0, \ldots, \alpha_{n-1})$ of n vertices out of the N vertices of $\pi(G)$. We now show that an $n \times N$ matrix P_α can be used to form an $n \times n$ matrix $\mathcal{A}_\alpha(s)$ whose matrix elements are the matrix elements of $\tilde{A}(s)$ associated with the n vertices appearing in α. To that purpose, define

$$P_\alpha = \sum_{i=0}^{n-1} |i\rangle\langle\alpha_i|, \tag{53}$$

$$\mathcal{A}_\alpha(s) = P_\alpha \tilde{A}(s) P_\alpha^T, \tag{54}$$

where

$$\tilde{A}(s) = \sum_{l,m=0}^{N-1} \tilde{A}_{l,m}(s)|l\rangle\langle m|. \tag{55}$$

It follows from these definitions that

$$\begin{aligned}\mathcal{A}_\alpha(s) &= \sum_{i,=0}^{n-1} |i\rangle\langle\alpha_i| \sum_{l,m=0}^{N-1} \tilde{A}_{l,m}(s)|l\rangle\langle m| \sum_{j=0}^{n-1} |\alpha_j\rangle\langle j| \\ &= \sum_{i,j=0}^{n-1} \tilde{A}_{\alpha_i,\alpha_j}(s)|i\rangle\langle j|. \end{aligned}\tag{56}$$

Thus the matrix elements $(\mathcal{A}_\alpha)_{i,j}(s)$ are precisely the matrix elements $\tilde{A}_{\alpha_i,\alpha_j}(s)$ associated with all possible pairs of vertices drawn from $(\alpha_0, \ldots, \alpha_{n-1})$. The matrix $\mathcal{A}_\alpha(s)$ is thus the adjacency matrix for the subgraph g_α composed of the vertices appearing in α, along with all the edges in $\pi(G)$ that join them. With $\mathcal{A}_\alpha(s)$, we can, as in the GI problem, test whether g_α is equal to H by checking whether $||\mathcal{A}_\alpha(s) - A'||_i$ vanishes or not. Here A' is the adjacency matrix of the graph H, and $||O||_i$ is the L_i norm of O.

We can now define a cost function whose minimum value vanishes if and only if G contains a subgraph g that is isomorphic to H. Because the transformation $\sigma(s)$ must be a permutation matrix when g is isomorphic to H, we again introduce the penalty functions $C_1(s)$ and $C_2(s)$ used in the GI COP to penalize those integer strings s that produce a $\sigma(s)$ that is not a permutation matrix:

$$C_1(s) = \sum_{i=0}^{N-1} \sum_{\alpha=N}^{M} \delta_{s_i,\alpha}, \tag{57}$$

$$C_2(s) = \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} \delta_{s_i,s_j}. \tag{58}$$

The final penalty function $C_3(s)$ for the SGI problem generalizes the one used in the GI problem. It is defined to be

$$C_3(s) = \prod_{\alpha=1}^{\binom{N}{n}} ||\mathcal{A}_\alpha(s) - A'||_i, \tag{59}$$

where the product is over all $\binom{N}{n}$ ways of choosing n vertices out of N vertices. Note that $C_3(s)$ vanishes if and only if G contains an n-vertex subgraph isomorphic to H. This follows since, if G contains an n-vertex subgraph g isomorphic to H, there exists a permutation $\pi(s)$ of G that has an n-vertex subgraph that is equal to H. Thus, for this s, there is a choice α of n vertices that gives a subgraph for which $\mathcal{A}_\alpha(s) - A' = 0$. It follows from Eq. (59) that $C_3(s) = 0$. On the other hand, if $C_3(s) = 0$, it follows that at least one of the factors on the RHS of Eq. (59) vanishes. Thus there is a choice α of n vertices for which $||\mathcal{A}_\alpha(s) - A'||_i = 0$. Thus $\mathcal{A}_\alpha(s) = A'$, and so G has a subgraph isomorphic to H.

The cost function for the SGI problem is now defined to be

$$C(s) = C_1(s) + C_2(s) + C_3(s), \tag{60}$$

where $C_1(s)$, $C_2(s)$, and $C_3(s)$ are defined in Eqs. (57)–(59). This gives rise to the following COP:

Subgraph isomorphism COP. Given an N-vertex graph G and an n-vertex graph H with $n \leqslant N$, and the associated cost function $C(s)$ defined in Eq. (60), find an integer string s_* that minimizes $C(s)$.

By construction (i) $C(s_*) = 0$ if and only if G contains a subgraph isomorphic to H, and $\sigma(s_*)$ is the permutation matrix that transforms G into a graph $\pi(G)$ that has H as a subgraph; and (ii) $C(s_*) > 0$ otherwise.

As with the GI COP, the SGI COP can be solved using adiabatic quantum evolution. The quantum algorithm for SGI begins by preparing the $L = N\lceil \log_2 N \rceil$ qubit register in the ground state of H_i [see Eq. (15)] and then driving the qubit register dynamics using the time-dependent Hamiltonian $H(t) = (1 - t/T)H_i + (t/T)H_P$. Here the problem Hamiltonian H_P is defined to be diagonal in the computational basis $|s_b\rangle$ and to have associated eigenvalues $C(s)$, where s is found from s_b according to Eqs. (6) and (7). At the end of the evolution the qubits are measured in the computational basis. The outcome is the bit string s_b^* so that the final state of the register is $|s_b^*\rangle$ and its energy is $C(s^*)$, where s^* is the integer string derived from s_b^*. In the adiabatic limit, $C(s^*)$ will be the ground-state energy, and if $C(s^*) = 0$ (> 0) the algorithm concludes that G contains (does not contain) a subgraph isomorphic to H. In the case where G does contain a subgraph isomorphic to H, the algorithm also returns the permutation $\pi_* = \pi(s^*)$ that converts G to the graph $\pi_*(G)$ that contains H as a subgraph. Note that any real application of AQO will only be approximately adiabatic. Thus the probability that the final energy $C(s^*)$ will be the ground-state energy will be $1-\epsilon$. In this case the SGI quantum algorithm must be run $k \sim O(\ln(1-\delta)/\ln\epsilon)$ times so that, with probability $\delta > 1-\epsilon$, at least one of the measurements will return the ground-state energy. We can make δ arbitrarily close to 1 by choosing k sufficiently large.

VII. SUMMARY

In this paper we have presented a quantum algorithm that solves arbitrary instances of the graph isomorphism problem and which provides an approach for finding the automorphism group of a graph. We numerically simulated the algorithm's quantum dynamics and showed that it correctly (i) distinguished nonisomorphic graphs, (ii) recognized isomorphic graphs, and (iii) determined the automorphism group of a given graph. We also discussed the quantum algorithm's experimental implementation, and showed how it can be generalized to give a quantum algorithm that solves arbitrary instances of the NP-complete subgraph isomorphism problem. As explained in Appendix A, the minimum energy gap for an adiabatic quantum algorithm largely determines the algorithm's computational complexity. Determining this gap in the limit of large problem size is currently an important open problem in adiabatic quantum computing (see Sec. IV). It is thus not possible to determine the computational complexity of adiabatic quantum algorithms in general, nor, consequently, of the specific adiabatic quantum algorithms presented in this paper. Adiabatic quantum computing has been shown to be equivalent to the circuit model of quantum computing [12–14], and so development of adiabatic quantum algorithms continues to be of great interest.

ACKNOWLEDGMENTS

We thank W. G. Macready and D. Dahl for many interesting discussions, and F. G. thanks T. Howell III for continued support.

APPENDIX A: QUANTUM ADIABATIC THEOREM

The question of how the state of a physical system changes when the system's environment undergoes a slow variation is an old one. For quantum systems, the answer is contained in the quantum adiabatic theorem which was proved by Born and Fock not long after the birth of quantum mechanics [33]. Subsequent work relaxed a number of the assumptions underlying the original proof [34–43], thereby widening the theorem's range of validity. As the physical setting for the quantum adiabatic theorem occurs often, it forms the foundation for many important applications in atomic, molecular, and chemical physics. Recently, it has been used as the basis for a novel alternative approach to quantum computing known as adiabatic quantum computing [21,22]. In this appendix we provide a brief review of the quantum adiabatic theorem (Appendix A 1) and then describe how it is used in adiabatic quantum computing (Appendix A 2).

1. Quantum adiabatic theorem

Consider a quantum system coupled to an environment that changes slowly over a time T. The dynamical evolution of its state $|\psi(t)\rangle$ is determined by the Schrödinger equation, which will be driven by a slowly varying Hamiltonian $H(t)$. The system's Hilbert space is assumed to be finite dimensional with dimension d. Thus at each instant t there will be d instantaneous energy eigenstates $|E_k(t)\rangle$ satisfying

$$H(t)|E_k(t)\rangle = E_k(t)|E_k(t)\rangle, \tag{A1}$$

with $E_0(t) \leqslant E_1(t) \leqslant \cdots \leqslant E_{d-1}(t)$.

Now suppose that we initially prepare the quantum system in the instantaneous energy eigenstate $|E_k(0)\rangle$ of the initial Hamiltonian $H(0)$. The quantum adiabatic theorem states that, in the limit $T \to \infty$, the final state $|\psi(T)\rangle$ will be (to within a phase factor) the instantaneous energy eigenstate $|E_k(T)\rangle$ of the final Hamiltonian $H(T)$. Note that in our discussions of adiabatic quantum computing, the initial state will always be the ground state of $H(0)$: $|\psi(0)\rangle = |E_0(0)\rangle$.

For reasons that will become clear below, the energy gap $\Delta(t) = E_1(t) - E_0(t)$ separating the two lowest instantaneous energy levels proves to be extremely important in adiabatic quantum computing. Although it is possible to prove the quantum adiabatic theorem for systems with a vanishing energy gap [41], the rate of convergence to the adiabatic limit can be arbitrarily slow. On the other hand, when the gap $\Delta(t)$ is nonvanishing for $0 \leqslant t \leqslant T$, it is possible to estimate how large T must be for the dynamics to be effectively adiabatic. Starting from the observation that for adiabatic dynamics there will be negligible probability to find the quantum system at $t = T$ in an energy level other than the ground state, a straightforward analysis [19,20] leads to the following adiabaticity constraint:

$$T \gg \frac{\hbar M}{\Delta^2}, \tag{A2}$$

where

$$M = \max_{0\leqslant s\leqslant 1}\left|\langle E_1(s)|\frac{d\tilde{H}(s)}{ds}|E_0(s)\rangle\right|, \tag{A3}$$

$$\Delta = \min_{0\leqslant s\leqslant 1}[E_1(s) - E_0(s)], \tag{A4}$$

$s = t/T$, and $\tilde{H}(s) = H(sT)$. The quantity Δ is the minimum energy gap arising during the adiabatic evolution.

The quantum adiabatic theorem provides a mechanism for traversing a path $|\psi(t)\rangle$ through Hilbert space that begins at a given state $|\psi_i\rangle$ and ends at a desired final state $|\psi_f\rangle$. To see this, let H_i and H_f be local Hermitian operators whose ground states are $|\psi_i\rangle$ and $|\psi_f\rangle$, respectively. An Hermitian operator is local if it couples at most k particles, with k finite. Suppose that we can apply a time-dependent Hamiltonian $H(t)$ over a time interval $0 \leqslant t \leqslant T$, with $H(0) = H_i$, and $H(T) = H_f$. Suppose further that we prepare our quantum system in the ground state $|\psi_i\rangle$ of H_i, and then apply $H(t)$ to it. In the limit $T \rightarrow \infty$, the quantum adiabatic theorem guarantees that the system at time T will be in the desired state $|\psi_f\rangle$. The outcome is thus a continuous path from $|\psi_i\rangle$ to $|\psi_f\rangle$.

2. Adiabatic quantum computing

To connect this discussion to quantum computing, imagine that there is a computational problem we would like to solve, and that we are able to construct a local Hamiltonian H_f whose ground state $|\psi_f\rangle$ encodes the solution to our problem. Often, the computational basis states can be chosen to be the eigenstates of H_P. Our GI problem Hamiltonian H_P is an example of such a Hamiltonian (see Sec. III). Let H_i be a local Hamiltonian operator whose ground state $|\psi_i\rangle$ is easy to prepare [e.g., see Eq. (15)]. In adiabatic quantum computing, the procedure presented in the preceding paragraph is applied with $|\psi(0)\rangle = |\psi_i\rangle$ and $H(t)$ tracing out a path from H_i to H_f (in the space of Hermitian operators). Originally, Ref. [21] chose $H(t)$ to linearly interpolate from H_i to H_f:

$$H(t) = (1 - t/T)\, H_i + (t/T)\, H_f. \tag{A5}$$

Writing $s = t/T$ and $\tilde{H}(s) = H(sT)$ gives

$$\tilde{H}(s) = (1 - s)\, H_i + s H_f. \tag{A6}$$

More general interpolation schemes are possible: $H(t) = A(t)H_i + B(t)H_f$, where we require $A(0) = 1$ [$B(0) = 0$] and $A(T) = 0$ [$B(T) = 1$]. See, for example, Refs. [15,25,27]. By choosing T sufficiently large, the final state $|\psi(T)\rangle$ can be brought arbitrarily close to $|\psi_f\rangle$. An appropriate measurement then yields $|\psi_f\rangle$ with probability close to 1, and thus yields the desired solution to our computational problem. Adiabatic quantum computing thus finds the solution to a computational problem by homing in on the ground state $|\psi_f\rangle$ of H_f which encodes the solution. The homing mechanism is provided by the quantum adiabatic theorem. It has been shown that adiabatic quantum computing has the same computational power as the circuit model for quantum computing [12–14]. It thus provides an important alternative approach to quantum computing that is especially well suited to problems that reduce to quantum state generation.

We are now in a position to state the protocol for the adiabatic quantum evolution (AQE) algorithm [21]:

(1) Prepare an n-qubit quantum register in the ground state $|\psi_i\rangle$ of H_i.

(2) At $t = 0$, apply $H(t)$ to the quantum register for a time T.

(3) At time $t = T$, measure the qubits in the computational basis.

Because the computational basis states are typically the eigenstates of H_f, the final measurement leaves the qubits in an eigenstate of H_f. In the adiabatic limit $T \rightarrow \infty$, the final measurement leaves the qubits in the ground state of H_f (which encodes the solution we are trying to find) with probability $P_{\text{success}} \rightarrow 1$.

Now for large, but finite T, the Schrödinger dynamics is approximately adiabatic. Thus, with probability $1 - \epsilon$, the measurement returns the problem solution. In this case the AQE algorithm must be run more than once. Suppose we run it κ times. The probability that we do *not* get the problem solution in *any* of the κ runs is ϵ^{κ}. We can make this probability take an arbitrarily small value $\tilde{\epsilon}$ by choosing $\kappa \sim O(\log(1/\tilde{\epsilon}))$. Thus with probability arbitrarily close to 1, one of the κ measurement results will yield the problem solution.

The adiabaticity constraint [see Eq. (A2)] specifies a lower bound which the runtime T must exceed if the Schrödinger dynamics is to be effectively adiabatic. In all applications of interest to date, the matrix element M appearing in this constraint scales polynomially with problem size N. So long as this is true, Eq. (A2) indicates that the scaling behavior of the runtime $T(N)$ is determined by the scaling behavior of the minimum gap $\Delta(N)$. Now, if at $t = 0$ the quantum register is prepared in the ground state of the initial Hamiltonian $H(0)$, and its dynamics is effectively adiabatic, its state at later times will be effectively restricted to the subspace spanned by the instantaneous ground and first excited states. Standard arguments [44] indicate that, in the absence of symmetry, these two energy levels will typically not cross. Thus the minimum gap $\Delta(N)$ will typically not vanish, and Eq. (A2) indicates that an effectively adiabatic dynamics can be obtained with finite $T(N)$. An algorithm, classical or quantum, is said to efficiently (inefficiently) solve a computational problem if its runtime scales polynomially (superpolynomially) with problem size. Thus, if $\Delta(N)$ scales inverse polynomially (superpolynomially) with N, then $T(N)$ will scale polynomially (superpolynomially) with N, and the AQE algorithm will be an efficient (inefficient) algorithm. We see that the scaling behavior of the minimum gap $\Delta(N)$ largely controls the computational complexity of the AQE algorithm.

APPENDIX B: EMBEDDING PROCEDURE FOR D-WAVE HARDWARE

This appendix briefly describes the embedding procedure used in Ref. [15] to program a non-Ising problem Hamiltonian H_P onto a D-Wave One processor. The processor architecture is shown in Fig. 18. This procedure also applies to a D-Wave Two processor.

As discussed in Sec. III, the GI algorithm presented in this paper constructs H_P to (i) be diagonal in the computational basis $\{|a_0 \cdots a_{L-1}\rangle : a_i = 0,1\}$; and (ii) have eigenvalues $C(a)$, where $a = a_0 \cdots a_{L-1}$, L is the number of qubits, and $C(a)$ is given by Eqs. (42)–(45) with $s \rightarrow a$. The cost function $C(a)$ is not yet ready for experimental implementation for two reasons. First, there are k-qubit interactions with $k > 2$ which cannot be implemented as the processor can only couple pairs of qubits; and second, two-qubit couplings may not correspond to available couplings on the processor (see Fig. 18). Procedures for removing each of these obstacles are presented, respectively, in Appendices B 1 and B 2. We summarize these procedures in Appendix B 3. To keep the discussion concrete, we examine a single k-qubit coupling term $A = a_1 \cdots a_k$ with $a_i = 0,1$. The resulting procedure must then be applied to each term in $C(a)$.

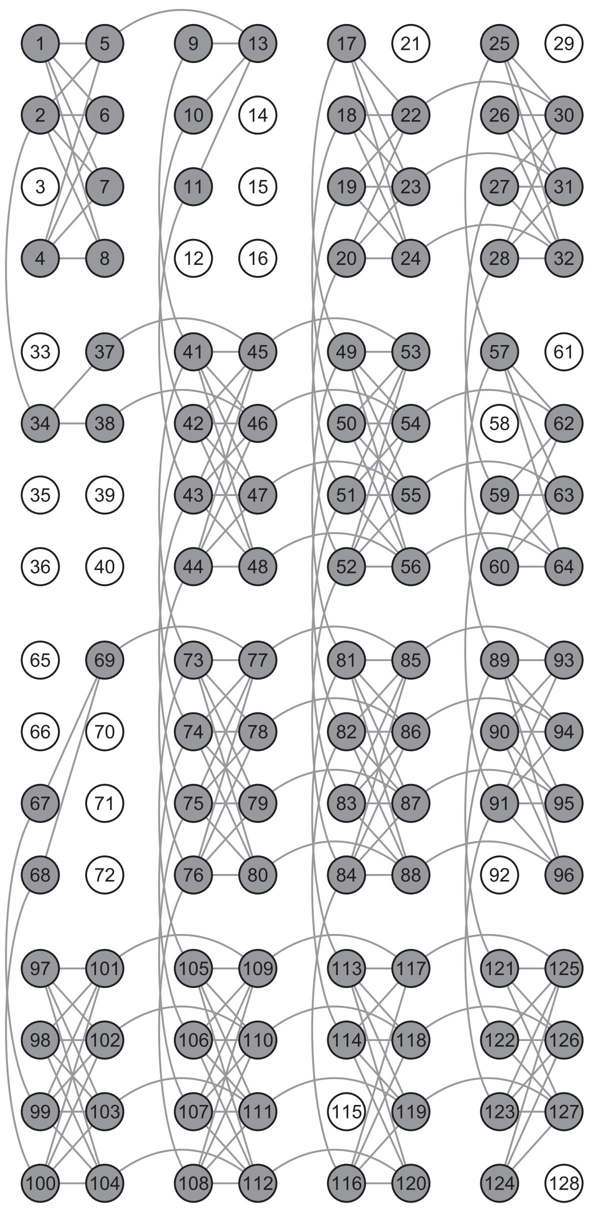

FIG. 18. Layout of qubits and couplers for a D-Wave One processor. The processor architecture is a 4×4 array of unit cells, with each unit cell containing eight qubits. Within a unit cell, each of the four qubits in the left-hand partition (LHP) connects to all four qubits in the right-hand partition (RHP), and vice versa. A qubit in the LHP (RHP) of a unit cell also connects to the corresponding qubit in the LHP (RHP) in the unit cells above and below (to the left and right of) it. Most qubits couple to six neighbors. Qubits are labeled from 1 to 128, and edges between qubits indicate couplers which may take programmable values. Gray qubits indicate usable qubits, while white qubits indicate qubits which, due to fabrication defects, could not be calibrated to operating tolerances and were not used.

1. Reduction to pairwise coupling

Here we describe how to reduce a k-qubit coupling term $A = a_1 \cdots a_k$ with $a_i = 0,1$ to a sum of two-qubit (viz., pairwise) coupling terms. We first show how to reduce a three-qubit coupling term to pairwise coupling terms (Appendix B 1 a), and then use lessons learned to reduce the k-qubit term A to pairwise coupling (Appendix B 1 b).

a. Three-qubit case

We begin by showing how to reduce a three-qubit coupling term $a_1a_2a_3$ to pairwise coupling by introducing (i) an ancillary variable b which takes only two values $\{0,1\}$, and (ii) the penalty function

$$P(a_1,a_2;b) = a_1a_2 - 2(a_1 + a_2)b + 3b. \tag{B1}$$

Notice that $P(a_1,a_2;b) = 0$ (>0) when the input values for a_1, a_2, and b satisfy $b = a_1a_2$ $(b \neq a_1a_2)$. Now consider the quadratic cost function

$$h(b) = ba_3 + \mu P(a_1,a_2;b)$$

for given values of μ, a_1, and a_2. For μ sufficiently large, $h(b)$ is minimized when the value of b satisfies the equality constraint $b^* = a_1a_2$. As noted above, for this optimal value of b, the penalty function $P(a_1,a_2;b^*) = 0$, and so the optimum cost $h(b^*)$ is

$$\begin{aligned} h(b^*) &= b^*a_3 + P(a_1,a_2;b^*) \\ &= a_1a_2a_3, \end{aligned}$$

where $b^* = a_1a_2$ has been used in going from the first to the second line. Thus, for values of b satisfying the equality constraint $b = a_1a_2$, the cost function $h(b)$, which is a sum of two-qubit coupling terms, reproduces the three-qubit coupling term $a_1a_2a_3$. By choosing μ sufficiently large, values of b that do not satisfy the equality constraint can be pushed to large cost (viz., energy), making such b values inaccessible during adiabatic quantum evolution.

b. k-qubit case

To reduce the k-qubit coupling term $A = a_1 \cdots a_k$ to pairwise coupling we (i) introduce ancillary bit variables $b_2, \ldots, b_{k-1}$, and (ii) impose the constraints $b_{k-1} = a_{k-1}a_k$ and $b_j = a_jb_{j+1}$ $(j = 2, \ldots, k-2)$ through the penalty function

$$P(\mathbf{a};\mathbf{b}) = P(a_{k-1},a_k;b_{k-1}) + \sum_{j=2}^{k-2} P(a_j,b_{j+1};b_j),$$

where $\mathbf{a} = (a_1, \ldots, a_k)$, $\mathbf{b} = (b_2, \ldots, b_{k-1})$, and $P(a,b;c)$ is defined in Eq. (B1). The quadratic cost function $C_A(\mathbf{a},\mathbf{b})$ is defined to be

$$C_A(\mathbf{a},\mathbf{b}) = a_1b_2 + \mu\, P(\mathbf{a};\mathbf{b}).$$

We require that the optimal values $(\mathbf{a}^*,\mathbf{b}^*)$ satisfy the $k-1$ imposed constraints so that $P(\mathbf{a}^*,\mathbf{b}^*) = 0$. For optimal values, the cost function evaluates to

$$\begin{aligned} C_A(\mathbf{a}^*,\mathbf{b}^*) &= a_1^*b_2^* + \mu P(\mathbf{a}^*,\mathbf{b}^*) \\ &= a_1^*a_2^*b_3^* \\ &= \vdots \\ &= a_1^* \cdots a_{k-2}^*b_{k-1}^* \\ &= a_1^*a_k^*, \end{aligned}$$

where (in the interest of clarity) we have introduced the constraints one at a time in going from one line to the next. Here μ is a penalty weight whose value is chosen large enough to freeze-out nonoptimal values of **a** and **b** during adiabatic quantum evolution. Thus, for values of **a** and **b** satisfying the $k-1$ equality constraints, and for μ sufficiently large, the cost function $C_A(\mathbf{a},\mathbf{b})$, which is a sum of two-qubit coupling terms, reproduces the k-qubit coupling term $A = a_1 \cdots a_k$ as desired.

2. Matching required couplings to hardware couplings

A cost function with only pairwise coupling such as results from the procedure described in Appendix B 1 may still not be experimentally realizable on a D-Wave processor as the pairwise couplings arising in $C_A(\mathbf{a},\mathbf{b})$ may not match the two-qubit couplings available on the processor. The primal graph of a quadratic cost function such as $C_A(\mathbf{a},\mathbf{b})$ is the graph whose vertices are the qubit variables, and whose edges indicate pairwise-coupled qubits. An arbitrary primal graph can be embedded into a sufficiently large qubit graph having the structure of Fig. 18. An embedding maps a primal graph vertex to one or more vertices in the qubit graph, where the image vertices form a connected subgraph of the qubit graph. The string of connected qubits are linked together with strong ferromagnetic couplings so that in the lowest energy state, these qubits have identical Bloch vectors. For example, to couple qubits 104 and 75 in Fig. 18 (which are not directly coupled) with coupling strength $\mathcal{J}$, we ferromagnetically couple qubits 104, 112, and 107 using strongly negative $J_{104,112}$ and $J_{107,112}$ values. Qubits 107 and 75 are directly coupled by the processor and so the desired coupling $\mathcal{J}$ is applied to the edge (viz., coupler $J_{107,75}$) connecting qubits 107 and 75: $J_{107,75} = \mathcal{J}$. The ferromagnetic chain thus effects the desired coupling of qubits 104 and 75. This embedding procedure must be carried out for each pair of primal graph vertices joined by an edge whose associated qubits are not directly coupled in the processor architecture.

3. Summary

By combining the procedures described in this appendix it is possible to transform any cost function $C(a)$ into a quadratic cost function with pairwise couplings that match the couplings specified by the processor architecture. The trade-off is the introduction of ancilla qubits that are needed to reduce the coupling interactions to pairwise coupling and to match the two-qubit couplings available on the processor.

[1] J. Köbler, U. Schöning, and J. Torán, *The Graph Isomorphism Problem* (Birkhäuser, Boston, 1993).

[2] S. Arora and B. Barak, *Computational Complexity* (Cambridge University Press, New York, 2009).

[3] R. Jozsa, Comput. Sci. Eng. **3**, 34 (2001).

[4] E. Bernstein and U. Vazarani, in *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, edited by R. Kosaraju, D. Johnson, and A. Aggarwal (ACM, San Diego, 1993), p. 11.

[5] C. Moore, A. Russell, and L. J. Schulman, in *46th Annual IEEE Symposium on Foundations of Computer Science*, edited by E. Tardos (IEEE, Los Alamitos, 2005), p. 479.

[6] V. Gudkov and S. Nussinov, arXiv:cond-mat/0209112v2.

[7] T. Rudolph, arXiv:quant-ph/0206068v1.

[8] S.-Y. Shiau, R. Joynt, and S. N. Coppersmith, Quantum Inf. Comput. **5**, 492 (2005).

[9] J. K. Gamble, M. Friesen, D. Zhou, R. Joynt, and S. N. Coppersmith, Phys. Rev. A **81**, 052313 (2010).

[10] K. Rudinger, J. K. Gamble, M. Wellons, E. Bach, M. Friesen, R. Joynt, and S. N. Coppersmith, Phys. Rev. A **86**, 022334 (2012).

[11] I. Hen and A. P. Young, Phys. Rev. A **86**, 042310 (2012).

[12] D. Aharonov, W. Van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, SIAM J. Comput. **37**, 166 (2007).

[13] J. Kempe, A. Kitaev, and O. Regev, SIAM J. Comput. **35**, 1070 (2006).

[14] R. Oliveira and B. M. Terhal, Quantum Inf. Comput. **8**, 900 (2008).

[15] Z. Bian, F. Chudak, W. G. Macready, L. Clark, and F. Gaitan, Phys. Rev. Lett. **111**, 130505 (2013).

[16] G. Chartrand and P. Zhang, *A First Course in Graph Theory* (Dover, Mineola, 2012).

[17] M. Mosca, in *Encyclopedia of Complexity and System Science*, edited by Robert A. Meyers (Springer, New York, 2009), p. 7088.

[18] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, in *45th Annual IEEE Symposium on Foundations of Computer Science (FOCS'04)*, edited by E. Upfal (IEEE, Los Alamitos, 2004), p. 42.

[19] A. Messiah, *Quantum Mechanics* (Wiley, New York, 1962), Vol. II, p. 750.

[20] L. I. Schiff, *Quantum Mechanics*, 3rd ed. (McGraw-Hill, New York, 1968).

[21] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, J. Phys. A **34**, 643 (2001).

[22] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, Science **292**, 472 (2001).

[23] F. Gaitan, Int. J. Quantum Inf. **04**, 843 (2006).

[24] F. Gaitan, Complexity **14**, 21 (2009).

[25] F. Gaitan and L. Clark, Phys. Rev. Lett. **108**, 010501 (2012).

[26] E. Farhi, J. Goldstone, and S. Gutmann, arXiv:quant-ph/0208135v1.

[27] J. Roland and N. J. Cerf, Phys. Rev. A **65**, 042308 (2002).

[28] D. M. Cvetković, M. Doob, and H. Sachs, *Spectra of Graphs* (Academic, New York, 1980).

[29] M. W. Johnson *et al.*, Nature (London) **473**, 194 (2011).

[30] R. Babbush, B. O'Gorman, and A. Aspuru-Guzik, Ann. Phys. **525**, 877 (2013).

[31] J. D. Whitfield, M. Faccin, and J. D. Biamonte, Eur. Phys. Lett. **99**, 57004 (2012).

[32] M. R. Garey and D. S. Johnson, *Computers and Intractability* (W. H. Freeman and Company, New York, 1979).

[33] M. Born and V. Fock, Z. Phys. **51**, 165 (1928).

[34] T. Kato, Phys. Soc. Jpn. **5**, 435 (1950).

[35] J. E. Avron, R. Seiler, and L. G. Yaffe, Commun. Math. Phys. **110**, 33 (1987); **156**, 649 (1993).

[36] G. Nenciu, J. Phys. A **13**, L15 (1980).

[37] K. O. Friedrichs, *Special Topics in Quantum Theory*, Lecture Notes (Courant Institute of Mathematical Science, New York University 1955); *On the Adiabatic Theorem in Quantum Theory, Part I* (Courant Institute of Mathematical Sciences, New York University, 1956); , *On the Adiabatic Theorem in Quantum Theory, Part II* (Courant Institute of Mathematical Science, New York University, 1956).

[38] G. Hagedorn, Ann. Phys. **196**, 278 (1989).

[39] J. E. Avron, J. S. Howland, and B. Simon, Commun. Math. Phys. **128**, 497 (1990).

[40] J. E. Avron and A. Elgart, Phys. Rev. A **58**, 4300 (1998).

[41] J. E. Avron, and A. Elgart, Commun. Math. Phys. **203**, 445 (1999).

[42] H. Narnhofer, and W. Thirring, Phys. Rev. A **26**, 3646 (1982).

[43] A. Martinez and S. Nakamura, C. R. Acd. Sci. Paris **318**, 1153 (1994).

[44] L. D. Landau, and E. M. Lifshitz, *Quantum Mechanics*, 3rd ed. (Butterworth-Heinemann, New York, 1977).