




## Enhanced connectivity of quantum hardware with digital-analog control

Asier Galicia <sup>1</sup>, Borja Ramon <sup>1</sup>, Enrique Solano,<sup>1,2,3,4</sup> and Mikel Sanz <sup>1,2,3,\*</sup>

<sup>1</sup>*Department of Physical Chemistry, University of the Basque Country, Apartado 644, 48080 Bilbao, Spain*

<sup>2</sup>*IQM, Nymphenburgerstrasse 86, 80636 Munich, Germany*

<sup>3</sup>*IKERBASQUE, Basque Foundation for Science, Maria Diaz de Haro 3, 48013 Bilbao, Spain*

<sup>4</sup>*International Center of Quantum Artificial Intelligence for Science and Technology (QuArtist) and Department of Physics, Shanghai University, 200444 Shanghai, China*



(Received 23 December 2019; accepted 23 June 2020; published 20 July 2020)

Quantum computers based on superconducting circuits are experiencing rapid development, with the aim to outperform classical computers in certain useful tasks in the near future. However, the currently available chip fabrication technologies limit the capability of gathering a large number of high-quality qubits in a single superconducting chip, a requirement for implementing quantum error correction. Furthermore, achieving high connectivity in a chip poses a formidable technological challenge. Here, we propose a hybrid digital-analog quantum algorithm that enhances the physical connectivity among qubits coupled by an arbitrary inhomogeneous nearest-neighbor Ising Hamiltonian and generates an arbitrary all-to-all Ising Hamiltonian only by employing single-qubit rotations. Additionally, we optimize the proposed algorithm in the number of analog blocks and in the time required for the simulation. These results take advantage of the natural evolution of the system by combining the flexibility of digital steps with the robustness of analog quantum computing, allowing us to improve the connectivity of the hardware and the efficiency of quantum algorithms.

DOI: [10.1103/PhysRevResearch.2.033103](https://doi.org/10.1103/PhysRevResearch.2.033103)

### I. INTRODUCTION

Quantum computation has emerged in recent years as a promising technology which aims at solving problems such as the factorization of a composite number [1], studying quantum field theories [2,3], simulating quantum chemistry [4,5] and fluid dynamics [6], and simulating complex systems [7–10] more efficiently than classical computation. Another reason for the increasing interest in the field of quantum computation is due to Grover's algorithm [11], which shows quadratic speedups when compared against classical search algorithms. This technology can be implemented in different quantum platforms, such as trapped ions, photonics, or superconducting qubits, among others. In this last platform, strong efforts have been performed to show an example in which a quantum processor outperforms any classical computer, a milestone recently achieved by Google [12]. This flourishing technology still presents a considerable number of challenges to be solved, such as increasing the number of high-quality qubits in a single quantum processor or achieving interactions among all qubits. These problems characterize the so-called noise intermediate-scale quantum (NISQ) devices [13], quantum chips comprising 50–100 qubits, which are still affected by significant noise.

With the objective of optimizing the currently available resources for quantum computation and, ultimately, implementing a useful quantum computer in the NISQ era, an alternative paradigm of quantum computing, called digital-analog quantum computation (DAQC) [14–16], has recently been proposed. The DAQC paradigm aims at improving the fidelity of the algorithm by exploiting the robustness of the natural dynamics provided by the quantum platform together with the flexibility given by the fast implementation of single qubit rotation (SQR).

Recently, a constructive method to achieve universal quantum computation employing an all-to-all (ATA) Ising Hamiltonian as the analog resource has been proposed in Ref. [15]. Afterward, the implementation of relevant quantum algorithms within the DAQC paradigm has also been addressed with the digital-analog quantum Fourier transform (QFT) [17] and the quantum approximate optimization algorithm (QAOA) [18]. In these references, the authors numerically compared the performance of the DAQC paradigm against the purely digital one under sensible noise sources, showing the former gives remarkably better results when the system scales up with the number of qubits.

The intrinsic connectivity among qubits in a quantum platform is not necessarily well described by an ATA homogeneous Hamiltonian. In fact, a realistic quantum chip is expected to present lower connectivity focused on approximately nearest-neighbor (NN) interactions, since ATA connections require a prohibitively increasing amount of wiring among qubits.

In this paper, we design an algorithm that optimally simulates an arbitrary ATA Ising Hamiltonian employing as a resource a given inhomogeneous NN Ising model and SQR.

\*mikel.sanz@ehu.es

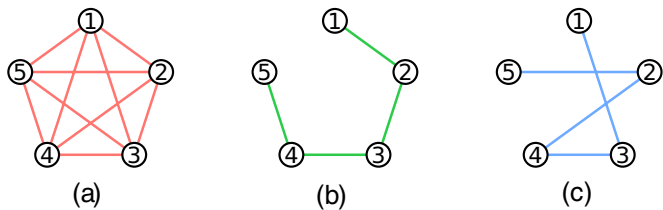


FIG. 1. Examples of Ising Hamiltonians as their graph representation. Panel (a) shows a complete  $K_5$  graph, which represents an all-to-all Hamiltonian  $H_{ATA}(g)$  for five qubits. Panel (b) shows a Hamiltonian path which represents a nearest-neighbor Hamiltonian  $H_{NN}(g)$  for five qubits. Panel (c) shows another Hamiltonian path with the vertex permutation  $P = [1, 3, 4, 2, 5]$ .

This is achieved employing  $O(5L^2)$  analog blocks, i.e., NN evolutions, with  $L$  being the number of qubits of the chip. Even though the particular dynamics considered as a resource is the ZZ Ising Hamiltonian, the proposed algorithm could be extended to other dynamics, such as the  $XX + YY$  Ising Hamiltonian. The simulation of the Hamiltonian is optimal in the dependence of the number of analog blocks and the simulation time required for these analog blocks.

## II. GRAPH REPRESENTATION OF AN ISING HAMILTONIAN

The Ising Hamiltonian for  $L$  qubits can be interpreted as a weighted graph of  $L$  vertices, where the weight of the edge connecting the vertex  $i$  to the vertex  $j$  is  $g_{ij}$ . If two vertices  $i$  and  $j$  are not connected,  $g_{ij} = 0$ .

In this representation, an ATA Ising Hamiltonian of  $L$  qubits becomes a complete graph  $K_L$ , i.e., a graph with edges among every possible vertex without repetition. On the other hand, the NN Ising Hamiltonian is represented as a Hamiltonian path, that is, a path visiting all the possible vertices only once.

It is noteworthy to mention that an arbitrary Hamiltonian path is represented by a permutation of all the vertices in the graph. To recover a Hamiltonian path from a given vertex permutation, it suffices to connect with an edge the vertices that are adjacent in the permutation. An example of a complete graph, together with two Hamiltonian paths, is represented in Fig. 1, where we also show the vertex permutation of the Hamiltonian paths.

Notice that we are currently dealing with ZZ interactions, and thus different Hamiltonian path evolutions commute. This means that the final evolution will be the sum of all the Hamiltonian paths weighted by their respective evolution time. This last statement is summarized in the equation

$$\prod_i e^{it_i \text{HP}_i(g_j)} = e^{i \sum_i t_i \text{HP}_i(g_j)}, \quad (1)$$

where  $\text{HP}_i(g_j)$  is a Hamiltonian that describes a ZZ interaction which has a graph representation of a Hamiltonian path with weights  $g_j$ . This is understood in the graph representation as having as final graph the sum of all the weighted Hamiltonian paths.

Our first task is then to split the complete graph, which represents the ATA Hamiltonian, into a set of Hamiltonian

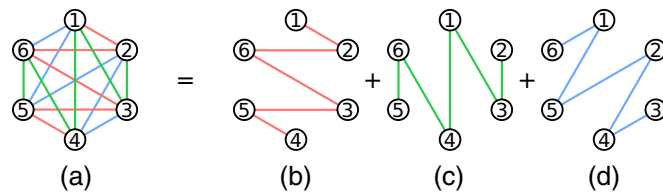


FIG. 2. An example of how to fill exactly a complete graph of six vertices using Hamiltonian paths. Panel (a) represents the complete graph we are trying to obtain. Panels (b), (c), and (d) correspond to the Hamiltonian paths we use. Panel (b) is built by starting in the first node, going forward to the next node, then two nodes backward, three forward, etc. The other two Hamiltonian paths are obtained by rotating the first one. From this construction technique, we obtain Hamiltonian paths defined by the permutations of Eq. (2). The permutations that define panels (b), (c), and (d) are respectively  $P_6^1 = [1, 2, 6, 3, 5, 4]$ ,  $P_6^2 = [2, 3, 1, 4, 6, 5]$ , and  $P_6^3 = [3, 4, 2, 5, 1, 6]$ .

paths that will be later simulated using our resource (the NN Hamiltonian). This will allow us to efficiently decompose the ATA evolution in terms of Hamiltonian paths.

Partitioning a complete graph into a set of Hamiltonian paths resembles the Hamiltonian decomposition problem, which is about partitioning a complete graph into a set of Hamiltonian cycles. This last problem was solved by Walecki [19,20] in 1890, who used a construction in which one Hamiltonian cycle is rotated to get all the cycles that compose the complete graph. Using a similar decomposition schematized for six qubits in Fig. 2, we can decompose the complete graph into a set of disjoint Hamiltonian paths. These Hamiltonian paths are characterized by the vertex permutation

$$P_L^k(j) = \begin{cases} (k - 1 + \frac{j}{2}) \bmod L + 1, & \text{if } j \text{ even,} \\ (k - 1 - \frac{j-1}{2}) \bmod L + 1, & \text{if } j \text{ odd,} \end{cases} \quad (2)$$

where  $P_L^k \in S_L$ ,  $S_L$  is the symmetric group for  $L$  elements,  $L$  is the number of qubits, and  $k \in \mathbf{Z}$  such that  $1 \leq k \leq L/2$ .  $j$  represents the  $j$ th position of the vertex permutation. Note that  $k$  is just a label for the Hamiltonian path.

It is also noteworthy to mention that this partition is only valid for an even number of qubits. When dealing with an odd number of qubits, and hence, an odd number of vertex in the graph representation, we use the notion of single perfect matching. This involves using the same Hamiltonian path construction of Eq. (2), but allowing one Hamiltonian path to overlap with the rest. This will pose no problem in our later construction, since we will be able to selectively turn off the desired interactions.

In Fig. 2, we show an example for six qubits where we also depicted the corresponding Hamiltonian paths. Hence, the problem now consists on efficiently simulating these Hamiltonian paths to obtain the ATA evolution.

## III. SWAPPING GATES

The next step is to obtain each of the Hamiltonian path connections using a NN Hamiltonian as a resource. For that, we will change the connections using a SWAP-like gate  $U$  that

performs the operations

$$\begin{aligned} U(\sigma_z \otimes I)U^\dagger &= I \otimes \sigma_z, \\ U(I \otimes \sigma_z)U^\dagger &= \sigma_z \otimes I. \end{aligned} \quad (3)$$

The gate that changes the action of an arbitrary operator in a qubit to another qubit is the SWAP gate, defined as

$$U_{\text{SWAP}} = e^{i\frac{\pi}{4}(X^1X^2+Y^1Y^2+Z^1Z^2)}, \quad (4)$$

where the superindices 1 and 2 refer to the qubit on which the gate acts. However, as we only need to change  $Z$  gates, we have some degrees of freedom available. More precisely, the most general unitary gate that fulfils Eq. (3) is

$$\begin{aligned} U(\alpha, \beta, \gamma) &= R_z^1 \left[ \pi \left( \frac{\gamma - \alpha}{2} + \frac{1}{2} + \beta \right) \right] \\ &\quad \times e^{i\frac{\pi}{4}(X^1X^2+Y^1Y^2+(\gamma+\alpha)Z^1Z^2)} \\ &\quad \times R_z^1 \left[ \pi \left( \frac{\gamma - \alpha}{2} - \frac{1}{2} - \beta \right) \right], \end{aligned} \quad (5)$$

where  $R_z^1[\theta] = e^{i\sigma_z^1 \frac{\theta}{2}}$  and  $\alpha, \beta, \gamma \in \mathbb{R}$ .

Since we will later build this gate using inhomogeneous Ising Hamiltonians from the DAQC perspective, we will set  $\alpha = \gamma = 0$  and  $\beta = -\frac{1}{2}$ , obtaining the so-called iSWAP gate. The choice of parameters will minimize the amount of single-qubit gates and analog blocks required, since

$$U_{\text{iSWAP}} = e^{i\frac{\pi}{4}(X^1X^2+Y^1Y^2)}. \quad (6)$$

We will focus now on obtaining a sequence of iSWAP gates acting on adjacent qubits that efficiently transforms a system with NN connections into a system with the desired Hamiltonian path connections. The reason to restrict ourselves to adjacent iSWAP gates is that we want to decompose them using NN Ising Hamiltonians as a resource.

For the sake of clarity, we will use  $U_{ij}$  to represent the iSWAP gate between qubits  $i$  and  $j$ . We will first show how this operation transforms a system with NN couplings. The result will be a system represented by a Hamiltonian path in its graph representation.

If we sandwich  $\sigma_z^k \sigma_z^l$ , a gate that applies the  $Z$  gate to the qubits  $k$  and  $l$ , with the gate  $U_{ij}$ , we obtain

$$U_{ij} \sigma_z^k \sigma_z^l U_{ij}^\dagger = \sigma_z^{\tau_{ij}(k)} \sigma_z^{\tau_{ij}(l)}, \quad (7)$$

where we have defined the function  $\tau_{ij}$  as a permutation of the indices  $i$  and  $j$ , that is, a transposition. More precisely, if  $k \neq i, j$ ,  $\tau_{ij}(k) = k$ , otherwise,  $\tau_{ij}(i) = j$  and  $\tau_{ij}(j) = i$ . Basically, the  $U_{ij}$  gate changes  $\sigma_z$  gates acting on qubit  $i$  to act on qubit  $j$ .

Because of the unitarity of the iSWAP operation, its action on a NN Hamiltonian evolution can be written as  $U_{ij} e^{itH(g)_{\text{NN}}} U_{ij}^\dagger = e^{itU_{ij}H(g)_{\text{NN}}U_{ij}^\dagger}$ . Therefore, the final Hamiltonian results in

$$\begin{aligned} U_{ij} H_{\text{NN}}(g) U_{ij}^\dagger &= \sum_{k=1}^{L-1} g U_{ij} \sigma_z^k \sigma_z^{k+1} U_{ij}^\dagger \\ &= \sum_{k=1}^{L-1} g \sigma_z^{\tau_{ij}(k)} \sigma_z^{\tau_{ij}(k+1)}. \end{aligned} \quad (8)$$

The initial vertex permutation defining the system's NN coupling was  $P = [1, 2, 3, \dots, L]$ . After the iSWAP operation, the permutation that defines our system is  $P' = [\tau_{ij}(1), \tau_{ij}(2), \tau_{ij}(3), \dots, \tau_{ij}(L)]$ . This approach is straightforwardly generalized to a system with arbitrary connections. This means that, after sandwiching with iSWAP gates a system with certain connections, it interchanges the ones that had the two qubits being affected by the iSWAP gate. In our case, since we will be dealing with systems represented by a Hamiltonian path, the iSWAP operation reduces to a transposition in the corresponding vertex permutation.

As the set of transpositions that our NN resource can implement,  $\{\tau_{i+1}\}$  for  $i \in [1, L-1]$ , is a generator of the symmetric group  $S_L$ , we can obtain any desired permutation using the correct transpositions. This means that we can simulate the evolution of any system with couplings represented by a Hamiltonian path using as resource the system with NN couplings. For example, the Ising Hamiltonian that represents Fig. 1(c), which we will call  $H_1$ , is just obtained using the following transformations  $H_1(g) = U_{34} U_{23} H_{\text{NN}}(g) U_{23}^\dagger U_{34}^\dagger$ .

For the sake of simplicity, we will define a sequence of transpositions to be

$$S_{i \rightarrow j} = \begin{cases} \tau_{i+1} \tau_{i+2} \tau_{i+3} \dots \tau_{j-1} j & \text{if } i < j \\ \mathbb{I} & \text{if } i \geq j \end{cases} \quad (9)$$

The permutations defined in Eq. (2) can be composed in terms of two groups of sequences,

$$\begin{aligned} G_1(k) &= S_{1 \rightarrow 2}^{2k-2} S_{2 \rightarrow 3}^{2k-3} \dots \\ &\quad \times S_{1 \rightarrow 2k-4}^4 S_{2 \rightarrow 2k-3}^3 S_{1 \rightarrow 2k-2}^2 S_{2 \rightarrow 2k-1}^1, \end{aligned} \quad (10)$$

$$\begin{aligned} G_2(k, L) &= S_{L-1 \rightarrow L}^{L-2k-1} S_{L-2 \rightarrow L-1}^{L-2k-2} \dots \\ &\quad \times S_{2k+4 \rightarrow L-1}^4 S_{2k+3 \rightarrow L}^3 S_{2k+2 \rightarrow L-1}^2 S_{2k+1 \rightarrow L}^1, \end{aligned} \quad (11)$$

where  $k$  and  $L$  refer to the permutation  $P_L^k$  we are building and the sequences have been labeled to make clear the order of application. That is,  $P_L^k = G_1(k)G_2(k, L)$ . In Appendix A, we prove that these are indeed the groups of sequences needed to obtain the desired permutations of Eq. (2). Note that both groups of sequences commute between them.

Let us now show an example for the case of six qubits. In order to obtain the permutation  $P_6^3$  from Fig. 3, we need to apply the transpositions defined in Eq. (11), which are  $G_1(3)$  and  $G_2(3, 6)$ . However, in the particular case of  $G_2(3, 6)$ , since  $2 \times k + 1 = 7$  and  $L = 6$ ,  $G_2(3, 6) = \mathbb{I}$ . The sequences applied are  $G_1(3) = S_{1 \rightarrow 2}^4 S_{2 \rightarrow 3}^3 S_{1 \rightarrow 4}^2 S_{2 \rightarrow 5}^1$ . The circuit that implements this set of transpositions using iSWAP gates is shown in Fig. 3, where each column of iSWAP gates represents one sequence of transpositions.

To sum up, in this section we have shown an algorithm that, using adjacent iSWAP gates and the NN Ising Hamiltonian as resource, is able to simulate the evolution of an ATA Hamiltonian for the case of an even number of qubits. Let us now simplify the circuit so that it requires the smallest possible amount of iSWAP gates.

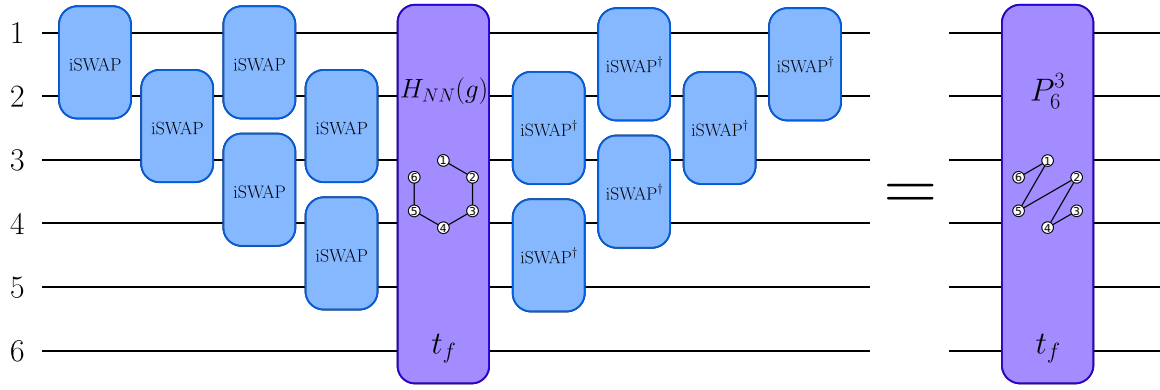


FIG. 3. Circuit that simulates the evolution of a Hamiltonian path with vertex permutation  $P_6^3$ . Each of the surrounding columns of iSWAP gates is related with one of the sequences belonging to the groups defined in Eq. (11).

#### IV. SIMPLIFICATION OF THE CIRCUIT

In this section, we will show an optimized version of the previously discussed circuit. Besides, we will also give a decomposition of the circuit in terms of ZZ gates, which will be useful later.

Since we now need to combine the set of Hamiltonian paths  $\{P_L^k\}$  to obtain the desired ATA evolution, it is possible to simplify the total number of iSWAP ( $U$ ) gates needed. The total circuit is described by the set of gates,

$$\begin{aligned}
 F(k, L) &= \prod_{i=2, i \text{ even}}^{2k+1} U_{i,i+1} \prod_{i=2k+1, i \text{ even}}^{L-1} U_{i,i+1}^\dagger \\
 &\times \prod_{i=1, i \text{ odd}}^{2k} U_{i,i+1} \prod_{i=2k+1, i \text{ odd}}^{L-1} U_{i,i+1}^\dagger, \\
 &\times k \in \left[1, \frac{L}{2} - 1\right], F(0, L) \\
 &= \prod_{j=\frac{L}{2}}^{j=1} \left[ \prod_{i=2j-1, i \text{ odd}}^{L-1} U_{i,i+1} \prod_{i=2j-2, i \text{ even}}^{L-2} U_{i,i+1} \right], \\
 F\left(\frac{L}{2}, L\right) &= \prod_{j=1}^{j=\frac{L}{2}} \left[ \prod_{i=1, i \text{ odd}}^{L-2j} U_{i,i+1}^\dagger \prod_{i=2, i \text{ even}}^{L+1-2j} U_{i,i+1}^\dagger \right]. \quad (12)
 \end{aligned}$$

The final ATA evolution will be described as

$$e^{it_f H_{ATA}} = \left[ \prod_{k=1}^{\frac{L}{2}} F(k, L) e^{it_f H_{NN}} \right] F(0, L). \quad (13)$$

In Appendix C, we show the demonstration leading to this simplified circuit. In Fig. 4, we show an example of the simplification for the case of six qubits.

Using Eq. (5), we can decompose the gates  $F(k, L)$  into a set of SQR and  $ZZ_{i,j}(\phi = g_j t_f)$  gates acting on adjacent qubits. Grouping parallel  $ZZ_{ij}$  gates naturally leads to a circuit which can be implemented using the DAQC protocol, as explained in the following section. An example for the case of six qubits is shown in Fig. 5.

We now take into account that two consecutive parallel sets of iSWAP gates can be decomposed into three analog blocks

plus some SQR (as will be proven in the next section). Since the total number of  $F(k, L)$  for  $k \in [1, \frac{L}{2} - 1]$  gates is  $\frac{L}{2} - 1$  and each one requires two parallel sets of iSWAP gates, we require a total of  $\frac{3L}{2} - 3$  analog blocks to generate these gates. We also need at most  $3(\frac{L}{2} - 2)$  analog gates to implement the  $F(0, L)$  set of gates and  $3(\frac{L}{2} - 1)$  analog gates to implement  $F(\frac{L}{2}, L)$ . Moreover, we need  $\frac{L}{2}$  analog blocks more evolving during a time  $t_f$ . The total amount of inhomogeneous analog blocks needed is then  $5L - 12$ .

In the following section, given a NN inhomogeneous Hamiltonian with fixed couplings, we will derive an algorithm to simulate the evolution of an arbitrary NN inhomogeneous Hamiltonian efficiently, both in time and in the number of analog blocks. This is the last step before obtaining an algorithm to simulate an arbitrary inhomogeneous ATA Hamiltonian.

#### V. SIMULATING AN ARBITRARY INHOMOGENEOUS HAMILTONIAN

In this section, we will show an algorithm to simulate the evolution of an arbitrary inhomogeneous NN Hamiltonian under the paradigm of DAQC, employing a similar algorithm to the one shown in Ref. [15]. In this case, our resource will be a fixed inhomogeneous NN Hamiltonian. The algorithm we use has the following three advantages over the one shown in Ref. [15]: (i) It works for an arbitrary number of qubits, (ii) it requires the minimum amount of analog blocks, and (iii) it optimizes the time required for the simulation.

We will first need to notice that it is possible to selectively change the sign of any desired combination of couplings. This is done by surrounding some of the qubits with X gates. We represent the action of the X gates by colouring the corresponding qubits in the graph representation (see Fig. 6). In order to change the sign of the desired combination of couplings, it suffices to color differently the qubits connected to the desired couplings. In Appendix B, we prove that this can be done for a NN chain with an arbitrary length. In Fig. 6, we change the sign of all the couplings in Fig. 6(a) and the sign of just one coupling in Fig. 6(b).

We will now decompose a  $H_{NN}(g'_j)$  evolution during a time  $t_f$  into a set of  $H_{NN}(g_j)$  evolutions that have been evolving

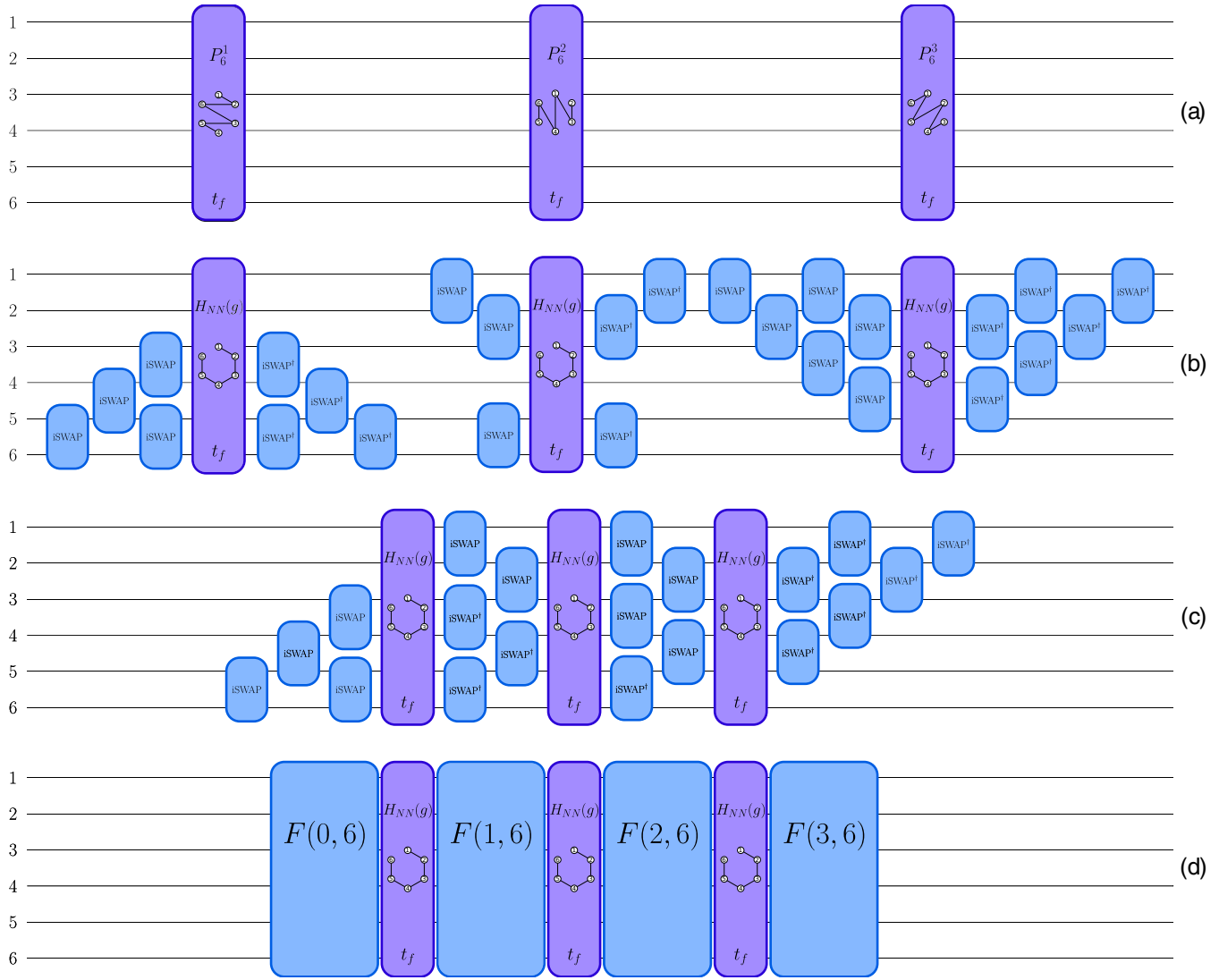


FIG. 4. Simplified circuit simulating an ATA Hamiltonian evolution. Panel (a) shows the three Hamiltonian paths that form the ATA evolution. Panel (b) is obtained by introducing the  $i$ SWAP gates that transform a NN Hamiltonian in the desired Hamiltonian paths. This gates are described by the group of transpositions defined in Eqs. (10) and (11). The set of transpositions obtained with this equation is translated into a set of  $i$ SWAP/ $i$ SWAP $^\dagger$  gates surrounding the NN Hamiltonian as discussed in Sec. III. In panel (c), we simplify the circuit by making use of  $i$ SWAP  $i$ SWAP $^\dagger = \mathbb{I}$ . The resulting gates are the set of  $i$ SWAP gates defined in Eq. (12), which lead to the circuit depicted in panel (d).

during a time  $t_n$  each:

$$\begin{aligned}
 t_f H_{NN}(g'_j) &= t_f \sum_{j=1}^{L-1} g'_j \sigma_z^j \sigma_z^{j+1} \\
 &= \sum_{j=1}^{L-1} \sum_{n=1}^{L-1} t_n g_j (-1)^{f_n(j)+f_n(j+1)} \sigma_z^j \sigma_z^{j+1} \\
 &= \sum_{j=1}^{L-1} \sum_{n=1}^{L-1} t_n \left( \prod_{k=1}^{L-1} X_k^{f_n(k)} \right) g_j \sigma_z^j \sigma_z^{j+1} \left( \prod_{k=1}^{L-1} X_k^{f_n(k)} \right) \\
 &= \sum_{n=1}^{L-1} t_n \left( \prod_{k=1}^{L-1} X_k^{f_n(k)} \right) H_{NN}(g_j) \left( \prod_{k=1}^{L-1} X_k^{f_n(k)} \right), \quad (14)
 \end{aligned}$$

where  $X_i^j$  is an  $X$  gate applied on the  $i$ th qubit  $j$  times and  $f_n(k)$  is a binary function that determines whether an  $X$  gate

is being applied in the  $k$ th qubit during the  $n$ th analog block, yet to be determined. In the first step, we make use of

$$b_j = \frac{g'_j}{g_j} = \sum_{n=1}^{L-1} M_{nj} \frac{t_n}{t_f}, \quad (15)$$

where  $M_{nj} = (-1)^{f_n(j)+f_n(j+1)}$ . This defines the following system of linear equations,

$$\mathbf{b} = M \frac{\mathbf{t}}{t_f}. \quad (16)$$

The matrix  $M$  has only  $\pm 1$  entries. The interpretation of  $M_{nj} = -1$  is that during the  $n$ th analog block, the  $j$ th coupling changes the sign. From now on, we will only focus on the  $M$  matrix instead of the  $f_n(j)$  functions.

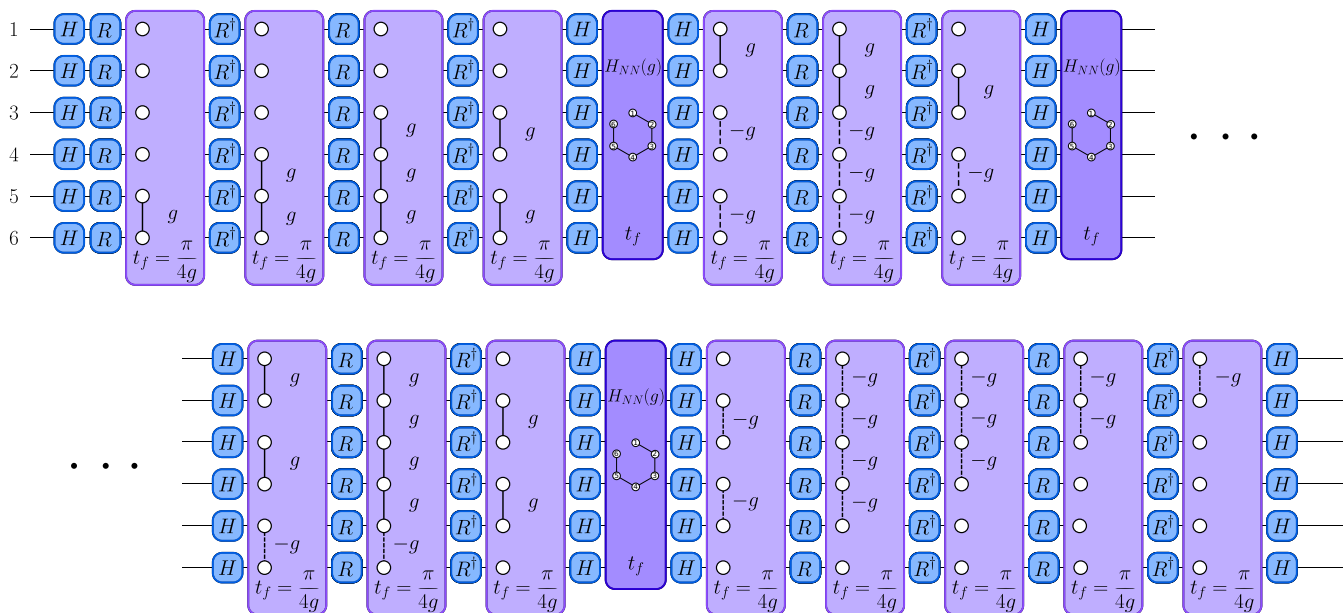


FIG. 5. This circuit is obtained by replacing each iSWAP in Fig. 4 by the expression shown in Eq. (6), where each of the exponentials have been multiplied by the appropriate single-qubit gates to transform them into ZZ interactions. Then, these interactions are gathered and expressed as evolutions of NN Ising Hamiltonians, characterized by the different couplings  $g$  and their evolution time  $t_f$ . Each of these inhomogeneous Hamiltonian evolutions can be implemented using the DAQC circuit discussed in Sec. V. The SQR employed in this circuit are the Hadamard gate ( $H$ ) and the gate  $R$ , defined as  $R = HSH$ , where  $S$  is the phase gate.

We will now assume without loss of generality that the following conditions hold  $\forall j$ :

$$b_j \geq b_{j+1}, \tag{17}$$

$$|b_1| \geq |b_j|, \tag{18}$$

$$b_j > 0. \tag{19}$$

Without loss of generality,  $b_j$  can always be relabeled and have their signs changed to hold these inequalities. Under these conditions, we propose the following  $M$  matrix:

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 & & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & \ddots & 1 & 1 & 1 & 1 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ -1 & -1 & -1 & -1 & \ddots & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & \ddots & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & \dots & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & \dots & -1 & -1 & -1 & 1 \end{bmatrix}. \tag{20}$$

After inverting this matrix (see Appendix D), Eq. (16) leads to the time intervals

$$\frac{t_k}{t_f} = \frac{b_k - b_{k+1}}{2}, \tag{21}$$

$$\frac{t_{L-1}}{t_f} = \frac{b_1 + b_{L-1}}{2}, \tag{22}$$

with  $k \in [1, L - 2]$ . Recall that  $t_k = 0$  means that we do not need the  $k$ th analog block for the simulation.

Let us now prove that these solutions for  $t_n$  imply the minimum amount of analog blocks and that the time required for the simulation is minimal. As, through Eqs. (21) and (22), we are mapping the set of times  $\{t_n\}_{n=1}^{L-1}$  to the values  $\{b_j\}_{j=1}^{L-1}$ , we need at least as much different  $t_n$  as  $b_j$ . Indeed, suppose that  $b_j = b_{j'}$ . We can relabel them such that  $j' = j + 1$ . Then, from Eq. (21), we get that  $t_j = 0$ , so the total number of analog blocks is reduced by 1. Another particularly relevant case is when  $b_j = 0$  for  $k$  different values, for which the number of analog blocks needed is reduced by  $k - 1$ .

The time required to simulate the desired  $H_{NN}(g'_j)$  is defined as  $t_{\text{sim}} = \sum_{n=1}^{L-1} |t_n|$ . Note that we do not take into account the time required to implement the  $X$  gates since we are supposing that they are ideal digital blocks, instantaneous. However, we believe that the circuit will still be minimum in time as long as we can parallelize the application of these gates. In Appendix E, we prove that,

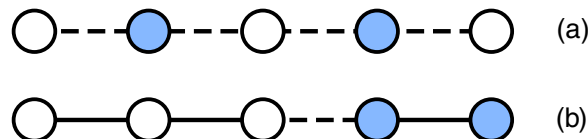


FIG. 6. Colored graphs representing the surrounding of  $X$  gates in the evolution of a NN system for five qubits. A colored node corresponds to a qubit surrounded by  $X$  gates. The dotted lines represent the change of sign in the coupling. Panel (a) represents the inversion of all couplings, which is the same as inverting the time evolution of the system. Panel (b) represents the inversion of only one of the couplings.

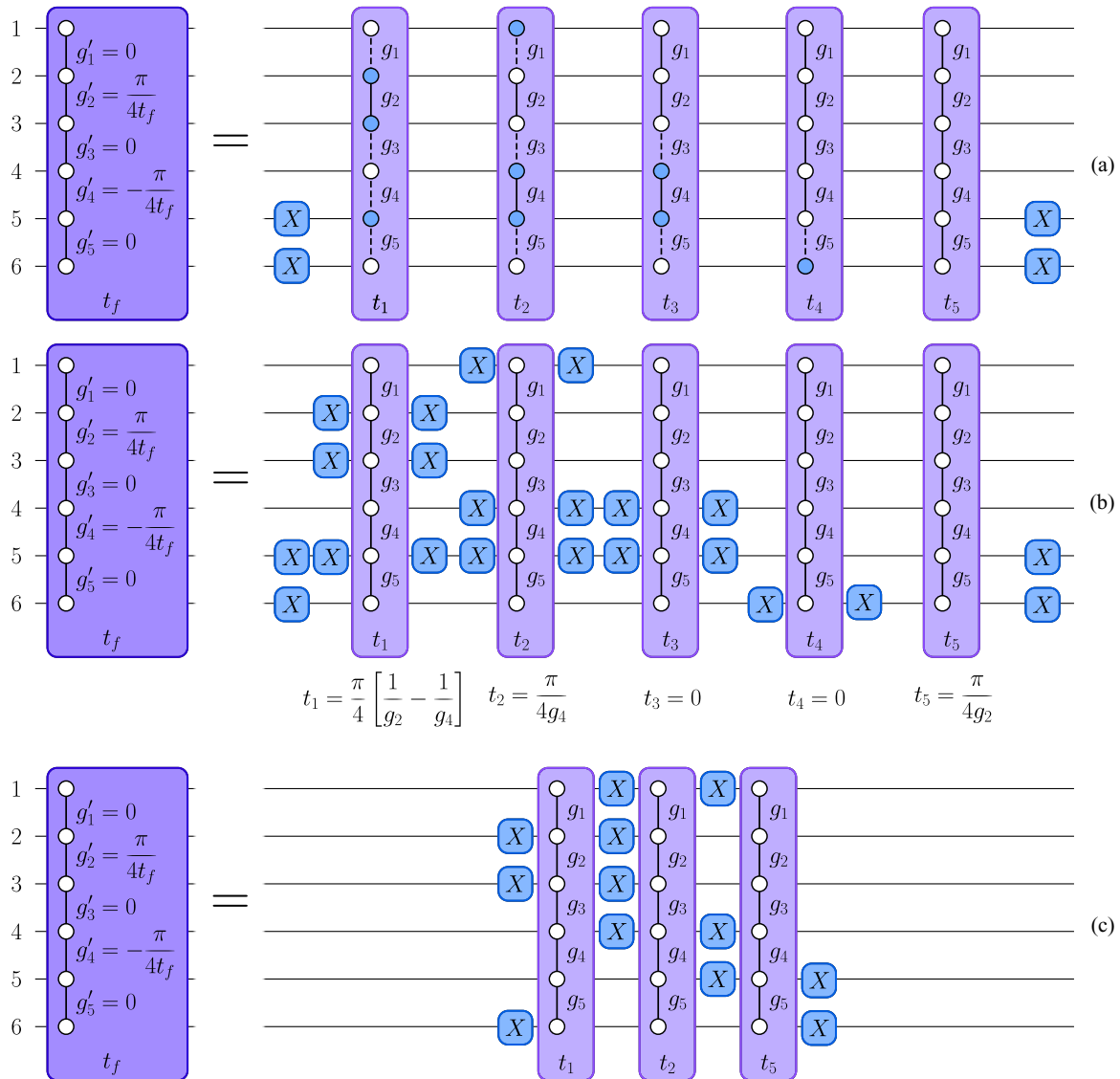


FIG. 7. Implementation of the circuit discussed in Sec. V. The digital block implemented is used in the circuit of Fig. 5 to generate an iSWAP gate between qubits 2 and 3, in parallel with an iSWAP<sup>†</sup> gate between qubits 4 and 5. The analog blocks shown in the right-hand side of panel (a) represent the evolution of a NN system, where the sign of some of the couplings have been inverted according to Eq. (20). The different evolution times are determined by Eqs. (21) and (22). In order to meet the constraints imposed by Eqs. (17)–(19), the coefficients  $b_1, b_2, b_3, b_4,$  and  $b_5$  are equal to  $\frac{g'_2}{g_2}, \frac{g'_4}{g_4}, \frac{g'_1}{g_1}, \frac{g'_3}{g_3},$  and  $\frac{g'_5}{g_5}$ , respectively (we made the assumption that  $g_4 > g_2$ ). Since  $b_2 < 0$ , we need to change  $g_4$  of sign in all the analog blocks, which is achieved by applying the X gates of panel (a). The dashed lines in the analog block represent a coupling changed of sign. These dashed lines connects two qubits with different color, whereas a solid line connects two qubits with the same color. The change of sign of the  $i$ th coupling ( $b_i$ ) is given by the  $i$ th column of the matrix defined in Eq. (20). For example, the first column of the matrix shows that all the signs of  $b_i$  belonging to the first block must be inverted except from  $b_1$ , which is related to  $g_2$ . Consequently, all the signs of the couplings must be inverted, except from  $g_2$ . In panel (b), we sandwiched each analog block with X gates in the qubits that were colored, representing in that way the same evolution of panel (a). We also show the time evolution corresponding to each analog block. Lastly, in panel (c) we simplified the previous circuit both by eliminating all the analog blocks with zero time evolution, and all unnecessary X gates, taking into account that  $XX = \mathbb{I}$ .

under the constrains of Eqs. (17)–(19),  $\min(t_{\text{sim}}) \equiv t_{\text{min}} = |b_1|t_f$ . We also prove in Appendix E that our circuit requires a time  $t_{\text{min}}$  to perform the simulation of an arbitrary inhomogenous Hamiltonian, which is the minimum time possible.

As an example, in Fig. 7 we represent the implementation of one of the analog blocks shown in Fig. 5, required for a set of iSWAP gates. More precisely, the depicted block is necessary

for an iSWAP gate between the qubits 2 and 3 and an iSWAP<sup>†</sup> gate between the qubits 4 and 5. It is noteworthy to mention that we require at least  $L - 1$  analog blocks to simulate the evolution of an arbitrary inhomogeneous NN Hamiltonian.

Until now, we have shown an algorithm that simulates the evolution of an homogeneous ATA Hamiltonian using as resource an inhomogeneous NN Hamiltonian. Furthermore, it is attained with  $O(5L^2)$  analog blocks, since we need

$O(5L)$  inhomogeneous analog blocks, each of which can be simulated using  $O(L)$  homogeneous analog resources.

It is straightforward to modify the circuit in order to simulate an inhomogeneous ATA Hamiltonian with negligible impact on the performance. It suffices to change the analog blocks of Fig. 4 for the necessary inhomogeneous NN Ising Hamiltonian.

In order to implement this circuit for an odd number of qubits,  $L$ , we can use the same set of Hamiltonian paths,  $P_L^k$ , of Eq. (2). In this case,  $k \in [1, \frac{L+1}{2}]$  and, in order to obtain an homogeneous ATA Hamiltonian, we need to set to zero some of the couplings used for the Hamiltonian path evolution representing  $P_L^{\frac{L+1}{2}}$ . It should be noted that the number of analog blocks will still be  $O(5L^2)$ . Even though we do not discuss here how to obtain the iSWAP gates for this case, techniques similar to those discussed in Appendix A can be employed.

## VI. CONCLUSIONS

We have shown that, within the DAQC paradigm, naturally arising evolutions can be utilized to simulate any inhomogeneous ATA Ising Hamiltonian along with SQR. In particular, we have designed an algorithm based on a NN Ising Hamiltonian with  $O(5L^2)$  analog blocks, where  $L$  is the number of qubits in the chip. For this, we also discussed both a digital approach that simulates an ATA system while having NN-like connections and an algorithm that simulates under the DAQC paradigm the evolution of an inhomogeneous Hamiltonian. This last algorithm has been proven to be efficient in the number of analog blocks and in the simulation time required, as long as we treat SQR as ideal gates. This protocol can be extended to platforms described by different Hamiltonians, such as the  $XX + YY$  NN Ising Hamiltonian.

## ACKNOWLEDGMENTS

The authors acknowledge support from Spanish Government PGC2018-095113-B-I00 (MCIU/AEI/FEDER, UE) and Basque Government IT986-16. The authors also acknowledge support from the projects QMiCS (820505) and OpenSuperQ (820363) of the EU Flagship on Quantum Technologies, as well as from the EU FET Open project Quomorphic (828826). This material is also based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advance Scientific Computing Research (ASCR), under field work Proposal No. ERKJ333.

## APPENDIX A: GROUP DEMONSTRATION

In this Appendix, we prove that the combination of sequences  $G_1(k)$  and  $G_2(k, L)$ , defined in Eq. (11), decompose  $P_L^k$  into a set of adjacent transposition, that is,  $P_L^k = G_1(k)G_2(k, L)$ . For that, we will first briefly discuss how to state the problem in terms of the matrix representation of the permutation group [21]. For the sake of clarity, we will denote by  $T_k$  the matrix representation of a transposition  $\tau_k$ .

The problem we are solving can be stated as obtaining a finite set of transpositions  $\{T_k\}_{k=1}^M$  that transforms the vector  $\mathbf{b}$  with components  $b_i$  into the vector  $\mathbf{b}'$  with components  $b_{P_L^k(i)}$ .

That is,  $\prod_{k=1}^M T_k \mathbf{b} = \mathbf{b}'$ . However, since  $T^{-1} = T$ , this set of transpositions will hold that  $\prod_{k=M}^1 T_k \mathbf{b}' = \mathbf{b}$ . The decomposition of  $P_L^k$  is then  $P_L^k = \tau^1 \circ \tau^2 \circ \dots \circ \tau^k$ , where we use  $b \circ a = b(a)$  to denote the order of the operations.

The only restriction we will impose in the available transpositions is that they need to be adjacent. Denoting by  $T(i, j)$  the transposition  $\tau_{i,j}$  that transposes the elements  $i$  and  $j$ , we note that  $T(i, j)\mathbf{b}$  transposes the elements of  $\mathbf{b}$  in the positions  $i$  and  $j$ . Hence, we can use algorithms, such as the Bubble Sort algorithm or its parallelized version, to directly obtain an optimized set of transpositions  $T^k$  that shorts a given vector  $\mathbf{b}'$ . Indeed, the set of transpositions  $G_1$  and  $G_2$  have been obtained using those techniques, though we considered it necessary to give a closed formula, which we now prove to be correct.

We will now define  $c_{i,l}$  as the operation that fulfills

$$c_{i,l}(P_L^k(j)) = \begin{cases} P_L^k(j) & \text{if } j \neq i, l \\ P_L^k(i) & \text{if } j = l \\ P_L^k(l) & \text{if } j = i. \end{cases} \quad (\text{A1})$$

That is, it changes the position of the entry  $i$  with the entry  $l$ . We will define  $g_1$  and  $g_2$  as  $G_1$  and  $G_2$  in Eqs. (10) and (11) but with all the  $\tau_{ij}$  operations replaced by  $c_{ij}$ . From the representation we see that replacing all the transpositions,  $\tau_{ij}$  for  $c_{ij}$ , makes the new group of operations  $g_1$  and  $g_2$  to fulfill the equation

$$g_1 \circ g_2 \circ P_L^k = 1, \quad (\text{A2})$$

where 1 stands for the identity permutation. This again resembles to shorting and array defined by  $P_L^k$  where the operations  $c_{ij}$  are the changes in positions made to that array. We continue by realizing that

$$P_L^k(j) = \begin{cases} P_{2k}^k(j) & \text{if } j \leq 2k, \\ P_{L-2k}^{L-2k}(j-2k) + 2k & \text{if } j > 2k, \end{cases} \quad (\text{A3})$$

that is, we obtain two commuting permutations. This is why two commuting groups of sequences,  $G_1(k)$  and  $G_2(k, L)$ , arise. Note that the second permutation, which can be regarded as  $P_{L'}^{L'}$  with  $L' = L - 2k$ , is obtained from

$$G_2(L') = S_{L'-1 \rightarrow L'}^{L'-1} \circ S_{L'-2 \rightarrow L'-1}^{L'-2} \circ \dots \times \dots \circ S_{4 \rightarrow L'-1}^4 \circ S_{3 \rightarrow L'}^3 \circ S_{2 \rightarrow L'-1}^2 \circ S_{1 \rightarrow L'}^1. \quad (\text{A4})$$

$G_2(L')$  differs from  $G_2(k, L)$  by a factor of  $2k$  that appears in all the sequences. This extra factor in  $G_2(k, L)$  comes from Eq. (A3), where it appears adding to the permutation  $P_{L-2k}^{L-2k}$ . It has the effect of changing the action of all transpositions from  $\tau_{ij}$  to  $\tau_{i+2k, j+2k}$ .

We will only prove how to short the permutation that fulfills  $g_1(k) \circ P_{2k}^k = 1$  because proving that  $g_2'(L') \circ P_{L'}^{L'} = 1$  requires the same steps. We will prove this by induction.

Since  $g_1(1)$  is the identity operation and  $P_2^1$  is the identity permutation, it is clear that  $g_1(1) \circ P_2^1 = 1$ . We now suppose that  $g_1(k-1) \circ P_{2k-2}^{k-1} = 1$  and we prove that, with this condition,  $g_1(k) \circ P_{2k}^k = 1$ . Since  $g_1(k) \circ P_{2k}^k = g_1(k-1) \circ s_{1 \rightarrow 2k-2} \circ s_{2 \rightarrow 2k-1} \circ P_{2k}^k$ , it suffices to prove that  $s_{1 \rightarrow 2k-2} \circ s_{2 \rightarrow 2k-1} \circ P_{2k}^k = P_{2k-2}^{k-1}$ , where  $s_{i \rightarrow j}$  is defined in Eq. (9) but with all the  $\tau_{ij}$  operations replaced by  $c_{ij}$  operations. Notice that  $s_{2 \rightarrow 2k-1}$  has the effect of changing the position of all



the entries in  $P_L^k$  except for the last and the first one. All the numbers in an odd position change to their left position and all the number in a even position change to their right position. Hence, the permutation obtained from  $\pi_1 = s_{2 \rightarrow 2k-1} \circ P_{2k}^k$  results in

$$\begin{aligned} \pi_1(j) &= \begin{cases} P_{2k}^k(j+1) & \text{if } j \text{ even} \\ P_{2k}^k(j-1) & \text{if } j \text{ odd} \\ P_{2k}^k(1) & \text{if } j = 1 \\ P_{2k}^k(2k) & \text{if } j = 2k \end{cases} \\ &= \begin{cases} P_{2k}^k(j+1) & \text{if } j \text{ even} \\ P_{2k}^k(j-1) & \text{if } j \text{ odd} \\ 2k & \text{if } j = 2k \end{cases}, \end{aligned}$$

where we used  $P_{2k}^k(0) = P_{2k}^k(1)$ .

We now compute the operation  $\pi_2 = s_{1 \rightarrow 2k-2} \circ \pi_1$ , which changes the position of all the entries except from the last two. Hence,

$$\begin{aligned} \pi_2(j) &= \begin{cases} \pi_1(j-1) & \text{if } j \text{ even} \\ \pi_1(j+1) & \text{if } j \text{ odd} \\ \pi_1(2k-1) & \text{if } j = 2k-1 \\ \pi_1(2k) & \text{if } j = 2k \end{cases} \\ &= \begin{cases} P_{2k}^k(j-2) & \text{if } j \text{ even} \\ P_{2k}^k(j+2) & \text{if } j \text{ odd} \\ 2k-1 & \text{if } j = 2k-1 \\ 2k & \text{if } j = 2k \end{cases} \\ &= \begin{cases} P_{2k-2}^{k-1}(j) & \text{if } j < 2k-1 \\ 2k-1 & \text{if } j = 2k-1 \\ 2k & \text{if } j = 2k \end{cases}. \end{aligned}$$

Since  $g_1(k-1)$  does not affect to the positions  $2k$  and  $2k-1$ ,

$$\begin{aligned} &g_1(k-1) \circ \pi_1(j) \\ &= \begin{cases} g_1(k-1) \circ P_{2k-2}^{k-1}(j) & \text{if } j < 2k-1 \\ 2k-1 & \text{if } j = 2k-1 \\ 2k & \text{if } j = 2k \end{cases} \\ &= 1. \end{aligned}$$

This completes the proof.

## APPENDIX B: COLORED GRAPHS DEMONSTRATION

In order to prove that we can selectively change the sign of a coupling in an arbitrary length NN chain, we will use an induction process. If  $L$  is the number of nodes in a NN chain, then  $k = L - 1$  is the number of couplings.

Inverting the couplings for the  $k = 1$  case is trivial. Supposing that we can selectively change the coupling sign of the  $k = k' - 1$ , we will prove that we can do it for the case  $k = k'$ . The construction relies in the fact that, in order to change the sign of the new coupling, it suffices to change the color of the newly added node. In this case, the color of the new node must be different from its neighbor's color, which can always be achieved in the NN case. If we do not want to change the sign of the  $k'$  coupling, we just need to color the new node as its neighbor.

## APPENDIX C: DEMONSTRATION OF EQ. (12)

In this Appendix, we show how to obtain the gates defined in Eq. (12). For that, we first define various set of gates that will simplify the notation. We define  $\tilde{S}$  to be the set of  $i\text{SWAP}_{ij}^\dagger$  gates that are obtained from substituting  $\tau_{ij} \rightarrow i\text{SWAP}_{ij}^\dagger$  in Eq. (9). At the same time,  $\tilde{G}$  is defined to be the set of  $i\text{SWAP}^\dagger$  gates that are obtained from substituting  $S \rightarrow \tilde{S}$  in Eqs. (10) and (11). Hence, it holds that

$$\text{HP}(P_L^k) = \tilde{G}_1^\dagger \tilde{G}_2^\dagger H_{\text{NN}} \tilde{G}_2 \tilde{G}_1, \quad (\text{C1})$$

where  $H_{\text{NN}}$  is the NN Hamiltonian and  $\text{HP}(P_L^k)$  is the ZZ interaction Hamiltonian described by the Hamiltonian Path  $P_L^k$ .

$F(k, L)$  describes the gates between the NN Hamiltonians that will be used to implement the Hamiltonian paths with vertex permutation  $P_L^k$  and  $P_L^{k-1}$ . Hence,

$$F(k, L) = \begin{cases} \tilde{G}_1^\dagger(k+1) \tilde{G}_2^\dagger(k+1, L) \tilde{G}_1(k) \tilde{G}_2(k, L) & k \in [1, \frac{L}{2} - 1], \\ \tilde{G}_2^\dagger(1, L) & k = 0, \\ \tilde{G}_1(\frac{L}{2}) & k = \frac{L}{2}, \end{cases} \quad (\text{C2})$$

where it is straightforward to prove that  $F(k, L)$  has the form described in Eq. (12) for  $k = 0$  and  $k = \frac{L}{2}$ . For  $k \in [1, \frac{L}{2} - 1]$ , we will prove that the set of  $i\text{SWAP}$  gates can be further simplified to obtain Eq. (12).

We first note that the following equations hold from the definition of  $\tilde{G}$ ,

$$\tilde{G}_1(k+1) = \tilde{G}_1(k) \tilde{S}_{1 \rightarrow 2k} \tilde{S}_{2 \rightarrow 2k+1}, \quad (\text{C3})$$

$$\tilde{G}_2(k, L) = \tilde{G}_2(k+1, L) \tilde{S}_{2k+2 \rightarrow L-1} \tilde{S}_{2k+1 \rightarrow L}. \quad (\text{C4})$$

Hence, it follows that

$$\begin{aligned} &\tilde{G}_1^\dagger(k+1) \tilde{G}_2^\dagger(k+1, L) \tilde{G}_1(k) \tilde{G}_2(k, L) \\ &= \tilde{S}_{2 \rightarrow 2k+1}^\dagger \tilde{S}_{1 \rightarrow 2k}^\dagger \tilde{G}_1^\dagger(k) \tilde{G}_2^\dagger(k+1, L) \tilde{G}_1(k) \tilde{G}_2(k, L) \\ &= \tilde{S}_{2 \rightarrow 2k+1}^\dagger \tilde{S}_{1 \rightarrow 2k}^\dagger \tilde{G}_2^\dagger(k+1, L) \tilde{G}_2(k, L) \\ &= \tilde{S}_{2 \rightarrow 2k+1}^\dagger \tilde{S}_{1 \rightarrow 2k}^\dagger \tilde{G}_2^\dagger(k+1, L) \tilde{G}_2 \\ &\quad \times (k+1, L) \tilde{S}_{2k+2 \rightarrow L-1} \tilde{S}_{2k+1 \rightarrow L} \\ &= \tilde{S}_{2 \rightarrow 2k+1}^\dagger \tilde{S}_{1 \rightarrow 2k}^\dagger \tilde{S}_{2k+2 \rightarrow L-1} \tilde{S}_{2k+1 \rightarrow L}, \end{aligned}$$

where we used that  $[\tilde{G}_1^\dagger(k), \tilde{G}_2^\dagger(k+1, L)] = 0$  and  $\tilde{G}_1^\dagger(k) \tilde{G}_1(k) = \tilde{G}_2^\dagger(k+1, L) \tilde{G}_2(k+1, L) = \mathbb{I}$ .

We now have that

$$F(k, L) = \begin{cases} \tilde{S}_{2 \rightarrow 2k+1}^\dagger \tilde{S}_{1 \rightarrow 2k}^\dagger \tilde{S}_{2k+2 \rightarrow L-1} \tilde{S}_{2k+1 \rightarrow L} & k \in [1, \frac{L}{2} - 1], \\ \tilde{G}_2^\dagger(1, L) & k = 0, \\ \tilde{G}_1(\frac{L}{2}) & k = \frac{L}{2}, \end{cases} \quad (\text{C5})$$

which when expressed in terms of the iSWAP gates leads to Eq. (12).

#### APPENDIX D: INVERSION OF MATRIX $M$

For the inversion of the matrix  $M$  of Eq. (20), it suffices to make row operations that transforms the  $M$  into the identity matrix while the identity matrix is transformed into  $M^{-1}$ . The required row operations are

$$r_i = \frac{r_i + r_1}{2}, \quad i > 1, \quad (\text{D1})$$

$$r_i = r_i - r_{i+1}, \quad i \in [1, L - 2], \quad (\text{D2})$$

where  $r_i$  refers to the  $i$ th row,  $L$  is the number of qubits, and hence  $L - 1$  is the dimension of  $M$ .

#### APPENDIX E: MINIMUM TIME OF SIMULATION FOR ARBITRARY INHOMOGENEOUS NN HAMILTONIAN

In this Appendix, we will prove that the solutions obtained in Eq. (21) give, under the constraints of Eqs. (17)–

(19), the minimum value possible to  $t_{\text{sim}}$ . This variable is defined as

$$t_{\text{sim}} = \sum_{n=1}^{L-1} |t_n|. \quad (\text{E1})$$

We will first prove that  $\min(t_{\text{sim}}) = |b_1 t_f|$ . For that, recall that Eq. (15) defines the relation between  $b_j$  and  $t_n$ . Computing its absolute value, we obtain

$$\begin{aligned} |b_j| &= \left| \sum_{n=1}^{L-1} M_{nj} \frac{t_n}{t_f} \right| \leq \sum_{n=1}^{L-1} \left| \frac{t_n}{t_f} \right| \\ &\leq \frac{t_{\text{sim}}}{t_f}. \end{aligned}$$

Using the constraints, we know that  $b_1$  is the maximum value of  $b_j \forall j$ . This proves that  $\min(t_{\text{sim}}) = |b_1 t_f|$ .

The solutions obtained in Eq. (21) hold that  $t_k > 0 \forall k$ . We now compute  $\sum_{k=1}^{L-1} |t_k| = \sum_{k=1}^{L-1} t_k = b_1 t_f$ , which proves that this set of solutions makes  $t_{\text{sim}}$  minimum.

Notice that even though the constraints may seem too restrictive,  $b_j$  can always be relabeled or changed its sign in order to hold them.

- 
- [1] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM Rev.* **41**, 303 (1999).
  - [2] E. A. Martinez, C. A. Muschik, P. Schindler, D. Nigg, A. Erhard, M. Heyl, P. Hauke, M. Dalmonte, T. Monz, P. Zoller, and R. Blatt, Real-time dynamics of lattice gauge theories with a few-qubit quantum computer, *Nature (London)* **534**, 516 (2016).
  - [3] N. Klco, E. F. Dumitrescu, A. J. McCaskey, T. D. Morris, R. C. Pooser, M. Sanz, E. Solano, P. Lougovski, and M. J. Savage, Quantum-classical computation of Schwinger model dynamics using quantum computers, *Phys. Rev. A* **98**, 032331 (2018).
  - [4] L. García-Álvarez, U. Las Heras, A. Mezzacapo, M. Sanz, E. Solano, and L. Lamata, Quantum chemistry and charge transport in biomolecules with superconducting circuits, *Sci. Rep.* **6**, 27836 (2016).
  - [5] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, *Nature (London)* **549**, 242 (2017).
  - [6] A. Mezzacapo, M. Sanz, L. Lamata, I. L. Egusquiza, S. Succi, and E. Solano, Quantum simulator for transport phenomena in fluid flows, *Sci. Rep.* **5**, 13153 (2015).
  - [7] B. P. Lanyon, C. Hempel, D. Nigg, M. Müller, R. Gerritsma, F. Zähringer, P. Schindler, J. T. Barreiro, M. Rambach, G. Kirchmair, M. Hennrich, P. Zoller, R. Blatt, and C. F. Roos, Universal digital quantum simulation with trapped ions, *Science* **334**, 57 (2011).
  - [8] D. Ballester, G. Romero, J. J. García-Ripoll, F. Deppe, and E. Solano, Quantum Simulation of the Ultrastrong-Coupling Dynamics in Circuit Quantum Electrodynamics, *Phys. Rev. X* **2**, 021007 (2012).
  - [9] S. Felicetti, M. Sanz, L. Lamata, G. Romero, G. Johansson, P. Delsing, and E. Solano, Dynamical Casimir Effect Entangles Artificial Atoms, *Phys. Rev. Lett.* **113**, 093602 (2014).
  - [10] R. Sweke, M. Sanz, I. Sinayskiy, F. Petruccione, and E. Solano, Digital quantum simulation of many-body non-Markovian dynamics, *Phys. Rev. A* **94**, 022317 (2016).
  - [11] L. K. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC)* (ACM, New York, 1996).
  - [12] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell *et al.*, Quantum supremacy using a programmable superconducting processor, *Nature (London)* **574**, 505 (2019).
  - [13] J. Preskill, Quantum computing in the NISQ era and beyond, *Quantum* **2**, 79 (2018).
  - [14] J. L. Dodd, M. A. Nielsen, M. J. Bremner, and R. T. Thew, Universal quantum computation and simulation using any entangling Hamiltonian and local unitaries, *Phys. Rev. A* **65**, 040301 (2002).
  - [15] A. Parra-Rodríguez, P. Lougovski, L. Lamata, E. Solano, and M. Sanz, Digital-analog quantum computation, *Phys. Rev. A* **101**, 022305 (2020).
  - [16] L. Lamata, A. Parra-Rodríguez, M. Sanz, and E. Solano, Digital-analog quantum simulations with superconducting circuits, *Adv. Phys. X* **3**, 1457981 (2018).
  - [17] A. Martin, L. Lamata, E. Solano, and M. Sanz, Digital-analog quantum algorithm for the quantum Fourier transform, *Phys. Rev. Res.* **2**, 013012 (2020).

- [18] D. Headley, T. Müller, A. Martin, E. Solano, M. Sanz, and F. K. Wilhelm, Approximating the quantum approximate optimisation algorithm, [arXiv:2002.12215](https://arxiv.org/abs/2002.12215).
- [19] R. Glebov, Z. Luria, and B. Sudakov, The number of Hamiltonian decompositions of regular graphs, *Israel J. Math.* **222**, 91 (2017).
- [20] B. Alspach, J. C. Bermond, and D. Sotteau, in *Cycles and Rays*, NATO ASI Series Vol. 301, edited by G. Hahn, G. Sabidussi, and R. E. Woodrow (Springer, Berlin, 1990).
- [21] B. E. Sagan, *The Symmetric Group: Representations, Combinatorial Algorithms, and Symmetric Functions*, Graduate Texts in Mathematics Vol. 203 (Springer Science & Business Media, Berlin, 2013).