

Research article

Quantum walk processes in quantum devices

Anandu Kalleri Madhu^a, Alexey A. Melnikov^{b,*}, Leonid E. Fedichkin^{b,c},
Alexander P. Alodjants^{d,e}, Ray-Kuang Lee^{a,f,g,h}^a Department of Physics, National Tsing Hua University, Hsinchu 30013, Taiwan^b Valiev Institute of Physics and Technology, Russian Academy of Sciences, 117218 Moscow, Russia^c Department of Theoretical Physics, Moscow Institute of Physics and Technology, 141701 Dolgoprudny, Russia^d ITMO University, 197101 St. Petersburg, Russia^e Quantum Light Engineering Laboratory, Institute of Natural and Exact Sciences, South Ural State University (SUSU), 454080 Chelyabinsk, Russia^f Institute of Photonics Technologies, National Tsing Hua University, Hsinchu 30013, Taiwan^g Physics Division, National Center for Theoretical Sciences, Taipei 10617, Taiwan^h Center for Quantum Technology, Hsinchu 30013, Taiwan

ARTICLE INFO

Dataset link: <https://github.com/q-ml/quantum-walks-quantum-devices>

ABSTRACT

Simulation and programming of current quantum computers as Noisy Intermediate-Scale Quantum (NISQ) devices represent a hot topic at the border of current physical and information sciences. The quantum walk process represents a basic subroutine in many quantum algorithms and plays an important role in studying physical phenomena. Simulating quantum walk processes is computationally challenging for classical processors. With an increasing improvement in qubits fidelity and qubits number in a single register, there is a potential to improve quantum walks simulations substantially. However, efficient ways to simulate quantum walks in qubit registers still have to be explored. Here, we explore the relationship between quantum walk on graphs and quantum circuits. Firstly, we discuss ways to obtain graphs provided quantum circuit. We then explore techniques to represent quantum walk on a graph as a quantum circuit. Specifically, we study hypercube graphs and arbitrary graphs. Our approach to studying the relationship between graphs and quantum circuits paves way for the efficient implementation of quantum walks algorithms on quantum computers.

1. Introduction

Random walks on graphs naturally appear in different physical processes [1–4] and computational subroutines [5–9]. In order to study these physical processes and implement algorithms, there is a need for efficient ways to simulate random walks [10,11]. Quantum walks, quantum analogues of (classical) random walks [12–18], naturally appear in physical processes when studied at a quantum level. Quantum interference, which is at the heart of quantum walks [19–24], potentially enables to accelerate energy transfer in Fenna-Matthews-Olson complexes [25–27] and quantum photonic circuits [28,29]. Understanding quantum walk advantage in particle transfer requires efficient simulation techniques and graph classification algorithms [30–32]. Simulating quantum walks is computationally a #P-hard problem, which makes classical simulators inefficient for the task [33,34]. In this regard, efficiently simulating quantum walks in quantum devices, especially in quantum computers available now represents an important

* Corresponding author.

E-mail address: melnikov@phystech.edu (A.A. Melnikov).

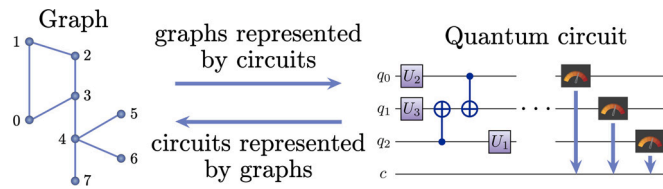


Fig. 1. A schematic relation between graphs and quantum circuits.

task for understanding quantum transport advantages [35,36], and for quantum algorithms implementation [37] as quantum walks represent a universal model of quantum computation [38,39].

However, existing quantum computers are NISQ devices, and possess limited capabilities due to a small number of qubits [40]. In this sense, the simulation of quantum walks on large-scale graphs with such devices represents an important and non-trivial task.

In this work, we study the practical possibilities to run Continuous-Time Quantum Walk (CTQW) processes in NISQ devices. The paper is mainly structured into two main parts as shown in Fig. 1. In the first section, we discuss about how to obtain graphs given a quantum circuit. We talk about two different kinds of graphs: i) hypercubes and ii) arbitrary graphs. In the second section, we discuss how to obtain a quantum circuit given a graph. We talk about how to simulate the CTQW on a graph using a quantum circuit. In particular, we discuss simulating hypercube graphs and arbitrary graphs. As an example of simulating an arbitrary graph, we simulate a 4 node “paw” graph on a quantum computer. We also discuss the technique of Hamiltonian simulation for simulating arbitrary unitary state evolution in a quantum computer.

2. Simulating quantum walks on graphs

Single particle CTQW on graphs is studied in this paper. A quantum particle can be located in one of the d positions on a graph with d vertices, or in a superposition of these positions. A quantum state of this particle can be thought of as a state of a d -level system as shown in Eq. (1):

$$|\psi_d(t)\rangle = \sum_{i=0}^{d-1} \alpha_i(t) |i\rangle \tag{1}$$

where $|\alpha_i(t)|^2$ being the detection probability in vertex i at a time t and verifying in Eq. (2):

$$\sum_{i=0}^{d-1} |\alpha_i(t)|^2 = 1 \tag{2}$$

The evolution of this quantum state is governed by the Hamiltonian with nearest-neighbour hopping terms given by Eq. (3):

$$\mathcal{H}_A = \hbar\Omega \sum_{i,j=0}^{d-1} A_{ij} |i\rangle \langle j| = \hbar\Omega A, \tag{3}$$

where A is an adjacency matrix of a graph on which the quantum walk is performed, A_{ij} are the elements of this matrix and Ω is the hopping frequency.

The unitary quantum state evolution is hence a solution to the Schrödinger equation, which is given by

$$|\psi_d(t)\rangle = e^{-i\Omega t A} |\psi_d(0)\rangle. \tag{4}$$

Note that A is not necessarily symmetric – the weights A_{ij} are complex parameters and can, in general, lead to a chiral quantum walk on a weighted graph [41,42].

Exponentiation of the $\Omega t A$ matrix is computationally challenging for large d . In the case of using a quantum computer, however, it is known that some unitary matrices can be efficiently implemented in time logarithmic in d . However, as the form of a Hermitian matrix A is arbitrary, we cannot efficiently implement any quantum walk on a quantum computer. In the general case, the matrix A is described by $(d^2 - d)/2$ independent complex-valued variables, which means in the worst case, one would need to apply $O(d^2)$ operations making exponential speedup impossible. In this paper, we explore the set of matrices A , for which efficient implementation is possible, and demonstrate this implementation on the IBM Q quantum device.

3. Obtaining graphs given quantum circuits

In this section, we discuss how to obtain graphs given a quantum circuit. A quantum circuit corresponds to a unitary state evolution of a Hamiltonian. This Hamiltonian represents the CTQW of a particle on a graph. The adjacency matrix of this graph is obtained from the underlying unitary represented by the quantum circuit. Therefore, every quantum circuit corresponds to a graph on which CTQW are implemented. In this section, we briefly discuss obtaining two families of graphs: i) hypercubes; ii) arbitrary graphs.

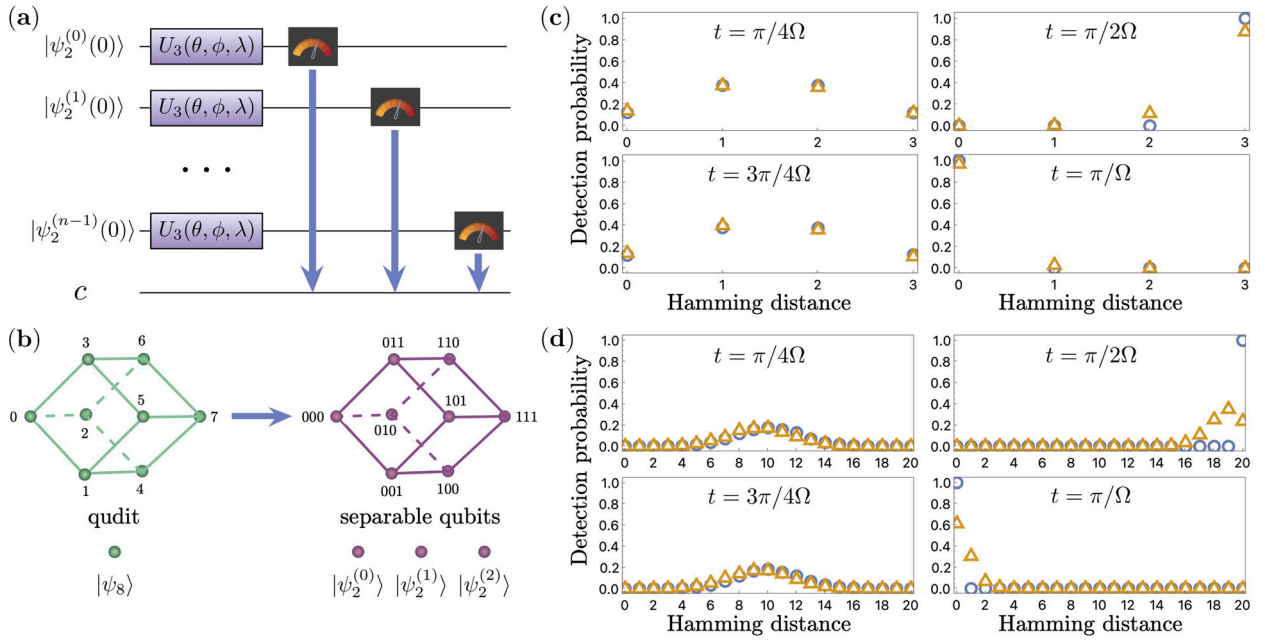


Fig. 2. (a) Quantum circuit for a CTQW implemented on n separable qubits. (b) Two different ways to encode a state of a walking particle on a hypercube graph: one qudit (left), and n separable qubits (right). Quantum and classical simulation of CTQW on (c) a 3-dimensional hypercube and (d) a 20-dimensional hypercube. Each plot shows a probability distribution of particle position on a graph at different times, corresponding to quantum circuit parameters $\phi = -\pi/2, \lambda = \pi/2$ and $\theta = 2\Omega t = \pi/2, \pi, 3\pi/2, 2\pi$. The blue circles and orange triangles correspond to execution on the classical simulator and the quantum IBM Q device, respectively. Hamming distance indicates the distance at which the particle propagated starting from the initial state labeled as “00...0”.

3.1. Quantum walks on hypercubes

Hypercube graphs represent a starting point towards studying graphs with a large number of vertices connected non-trivially. To simulate quantum walks on hypercubes, one needs to establish a mapping. Firstly, an encoding of the particle’s position in the space of qubits needs to be defined. Secondly, a sequence of quantum gates must be specified, which corresponds to a unitary defined by an adjacency matrix. Here we consider two possibilities of mappings, although there could be more, which correspond to an exponential reduction of the number of qubits compared to the graph vertices. The first possibility is based on using separable qubits, whereas the second possibility requires entangled qubits. In both cases the state $|\psi_d(t)\rangle$ is simulated on a classical simulator and IBM Q quantum devices.

The first mapping is demonstrated in Fig. 2(b). Instead of describing a quantum particle’s position as a state of a single d -level quantum system, we can think of the position as a state of an $n = \log_2 d$ qubits system. These two different ways of encoding a quantum walker state are shown in Fig. 2(b) for cube graphs. Both cube graphs in Fig. 2(b) have the same adjacency matrix A^{hc} . The two graphs’ difference is only in the vertex labeling: decimal (left cube, from 0 to 7) and binary (right cube, from 000 to 111) encoding choices. The binary encoding choice replaces a d -level quantum system, qudit, with multiple 2-level systems, qubits.

In addition to the quantum walkers state mapping, we also specify the mapping for unitary operations defined by the graph edges. The binary labelling is shown in Fig. 2(b) on the right helps us to see that two vertices are connected by a bit-flip operation. Moreover, each i -th bit-flip corresponds to a walk on the i -th axis. Therefore, a walk on a hypercube is a sequence of independent bit-flip operations. In other words, CTQW on an arbitrary n -dimensional hypercube can be decomposed into CTQW on n independent line graphs. The same can be observed by decomposing the A^{hc} matrix as in Eq. (5):

$$A^{hc} = \sum_{i=0}^{n-1} I_2^{\otimes i} \otimes A_i^{line} \otimes I_2^{\otimes n-1-i}. \tag{5}$$

Moreover, as a direct consequence of the above identity, the unitary matrix that is a function of A is equal to

$$U(A^{hc}) = e^{-i\Omega t A^{hc}} = \left(e^{-i\Omega t A^{line}} \right)^{\otimes n}, \tag{6}$$

where $A^{line} = |0\rangle\langle 1| + |1\rangle\langle 0|$ represents an adjacency matrix of a two-vertex line graph.

We next implement the CTQW on n -dimensional hypercubes in qubit registers of IBM Q. Because of the unitary operations simplified form, obtained in Eq. (6), we implement n identical CTQW on a line. Each quantum walk on the line is simulated by a single-qubit unitary that evolves the qubit state $|\psi_2^{(i)}\rangle$ shown in Fig. 2(a). For the simulation, we use the U_3 gate given by Eq. (7) in Qiskit to evolve the CTQW during the time $t = \theta/2\Omega$:

$$U_3(\theta, \phi, \lambda) = \begin{bmatrix} \cos \frac{\theta}{2} & -e^{i\lambda} \sin \frac{\theta}{2} \\ e^{i\phi} \sin \frac{\theta}{2} & e^{i(\lambda+\phi)} \cos \frac{\theta}{2} \end{bmatrix}. \tag{7}$$

A combined CTQW circuit, which is composed of the U_3 gates is depicted in Fig. 2(a). The initial position of the simulated particle is defined by the initial quantum state $|\psi_{2^n}(0)\rangle = |\psi_2^{(1)}(0)\rangle \otimes \dots \otimes |\psi_2^{(n)}(0)\rangle$. Because of the symmetry of hypercube graphs, without the loss of generality, the initial vertex will always be “00...0” and $|\psi_{2^n}(0)\rangle = |00\dots 0\rangle$. Simulating the evolution of $|\psi_{2^n}(t)\rangle$ with the specified initial condition, we observe the detection probabilities $p_k = |\langle k|\psi_{2^n}(t)\rangle|^2$ by repeating the circuit execution on the IBM Q quantum device.

As a result of the simulations, we obtained the particle’s probability to be in different vertices of the hypercubes in different time steps. The results for graphs with $n = 3$ and $n = 20$ are given in Fig. 2(c) and Fig. 2(d), respectively. The results demonstrate that starting from the “00...0” initial node, the quantum particle moves along the graph reaching the furthest node “11...1” with almost unit probability at time $t = \pi/2\Omega$. The same holds for both $n = 3$ and $n = 20$. Indeed, for any dimension n , the time for a perfect particle transfer is constant $t = \pi/2\Omega$. This is expected since we evolved the walk through separable qubits independently. In the results shown in Fig. 2(c)-(d), classical simulated results are shown as blue circles and quantum simulated results are shown as orange triangles. We observe that for times $t = \pi/4\Omega$ and $t = 3\pi/4\Omega$ the quantum simulated probability distribution matches the classical simulated probability distribution with an error in probability below 0.05 for all data points. In cases of $t = \pi/2\Omega$ and $t = \pi/\Omega$, however, the errors in quantum simulation go up to 0.15 for $n = 3$, and up to 0.8 for $n = 20$. In addition, the errors are counter-intuitively lower for $t = \pi/\Omega$ compared to $t = \pi/2\Omega$, which is explained by the underlying quantum system symmetry. However, in the case of separable qubits, the quantum simulation does not provide any more advantage than classical simulation since we haven’t used any entangled qubits for the simulation.

3.2. Quantum walks on arbitrary graphs

Not all unitary operations are efficiently implementable on a quantum computer, hence not all graphs A with d vertices can be implemented with $O(\log d)$ qubits in time $O(\text{poly}(\log d))$. Nonetheless, as we demonstrated in the previous section, for the hypercube graphs it is possible. By studying CTQW on hypercubes, and providing different implementations of the same process, we conclude that some implementations are more feasible than others. To see what other graphs have a feasible and efficient implementation on a quantum device, we propose the following.

To find out which graphs can be efficiently implemented on a quantum computer, we explore different quantum circuits that are efficiently implementable on the IBM Q quantum computer. Each of these quantum circuits represent some specific unitary operation U , which has a corresponding adjacency matrix $A(U)$. Because of the form of evolution $U = e^{-i\Omega t A}$ in Eq. (4), $A(U) = i \log(U)/\Omega t$. The logarithm of a unitary matrix is not uniquely defined, however [43]. First, adding a term $2\pi k I$, $k \in \mathbb{Z}$ with I being the identity matrix does not introduce a change in matrix A . Second, adding a global phase $c \in \mathbb{C}$ to a unitary $U' = cU$ does not change the particle’s quantum walk dynamics. The freedom in phase leads to the possibility to add a term φI , $\varphi \in \mathbb{R}$ to any adjacency matrix without affecting the quantum walk simulation. Third, one adds a complete graph with a factor $2\pi k A_{\text{complete}}/n\Omega t$, $k \in \mathbb{Z}$, without introducing a change in U . This is a consequence of A_{complete}/n being an idempotent matrix with $n -$ number of qubits. This condition is only valid if A_{complete}/n commutes with U . Finally, combining all three possibilities to modify A together, we obtain

$$A = \frac{i}{\Omega t} \log(U) + \varphi I + \frac{2\pi k}{n\Omega t} A_{\text{complete}}, \tag{8}$$

with free parameters $k \in \mathbb{Z}$, and $\varphi \in \mathbb{R}$. In addition to the derived Eq. (8), multiplying A by an arbitrary factor $b \in \mathbb{R}$ leads to an effective rescaling of the transition frequency $\Omega' = \Omega/b$, which broadens the set of available A even more.

The derived Eq. (8) helps us to obtain a variety of adjacency matrices of potential interest given the unitary transformation. Our implementation of the unitary transformations, in turn, is adjusted to the quantum device’s connectivity. From the quantum device’s connectivity limitations, it is possible to simulate which $U(A)$ are easily implementable with k -depth quantum circuits. For this, we automated sampling of random circuits that implement perfect transport from the state “0...00” to the state “1...11” with a circuit depth up to 5. We obtained these circuits to find feasible U , and from it, feasible graphs $A(U)$. In the case of 4 qubits, we obtained a graph shown in Fig. 3(b), by randomly implementing U_3 and CNOT gates. The corresponding quantum circuit is given in Fig. 3(a). Given that the initial state is “0000”, the circuit’s transformation is equivalent to unitary in Eq. (6). However, for the general initial particle’s position, i.e., general $|\psi_{16}(0)\rangle$, the transformation is different from the hypercubes cases. This difference can be observed from the disconnected graph shown in Fig. 3(b) compared to a fully connected graph of a 4-dimensional cube in Fig. 4(b). Quantum and classical simulation results in Fig. 3(c) demonstrate the chiral nature of the graph. Indeed, starting from $|\psi_{16}(0)\rangle = |0000\rangle$, particle ends in $|1111\rangle$, however starting from $|\psi_{16}(0)\rangle = |1111\rangle$ brings the particle to the “0110” vertex. The simulations performed for the initial state $|\psi_{16}(0)\rangle = |1010\rangle$ show that the particle is bounded to the subspace of the “1010” and “0010” vertices. Note that the fidelities of all the quantum simulations in Fig. 3 are above 0.75.

4. Obtaining quantum circuits given graphs

In the section above, we have discussed how to obtain graphs given quantum circuits. In this section, we discuss obtaining quantum circuits corresponding to CTQW of a particle on a graph. The adjacency matrix A of a graph corresponds to the Hamiltonian of the CTQW of a particle on that graph. Therefore, to represent a graph as a quantum circuit, we simulate the unitary state evolution

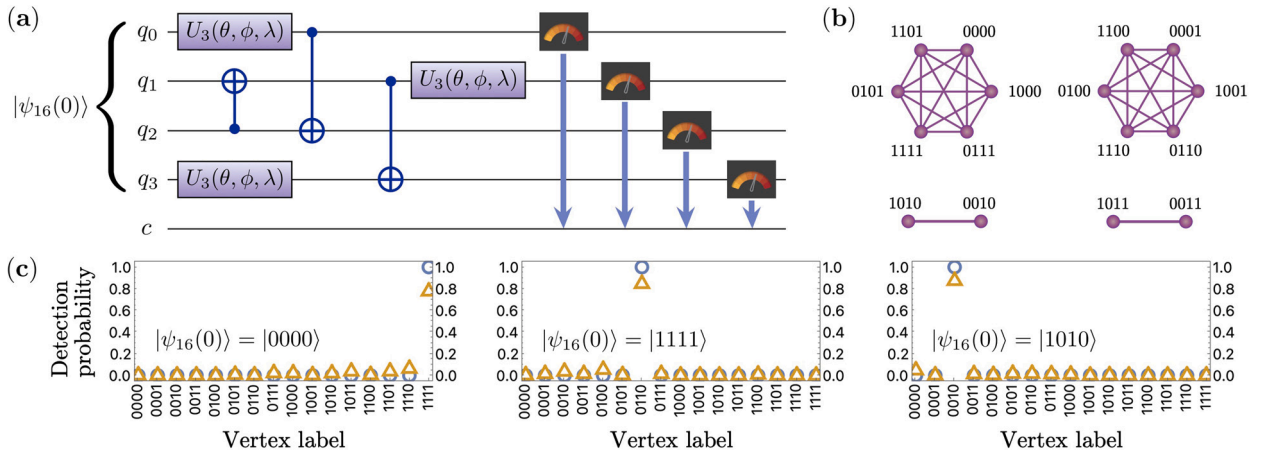


Fig. 3. (a) A 4-qubit quantum circuit consisting of $U_3(\theta = \pi, \phi = -\pi/2, \lambda = \pi/2)$ and CNOT gates with a circuit depth of 5. (b) A weighted graph generated from a 4-qubit quantum circuit with U_3 and CNOT gates on which we evolved a CTQW starting from the node “0000” and ending up in “1111” with probability one. Weights are not shown. (c) Quantum and classical simulation results for a quantum walk on the graph (b). Each plot shows a probability distribution of particles position on a graph for different initial conditions: $|\psi_{16}(0)\rangle = |0000\rangle$, $|1111\rangle$, and $|1010\rangle$. The blue circles and orange triangles correspond to execution on the simulator and the IBM Q quantum register, respectively. Vertex labels correspond to vertices in (b).

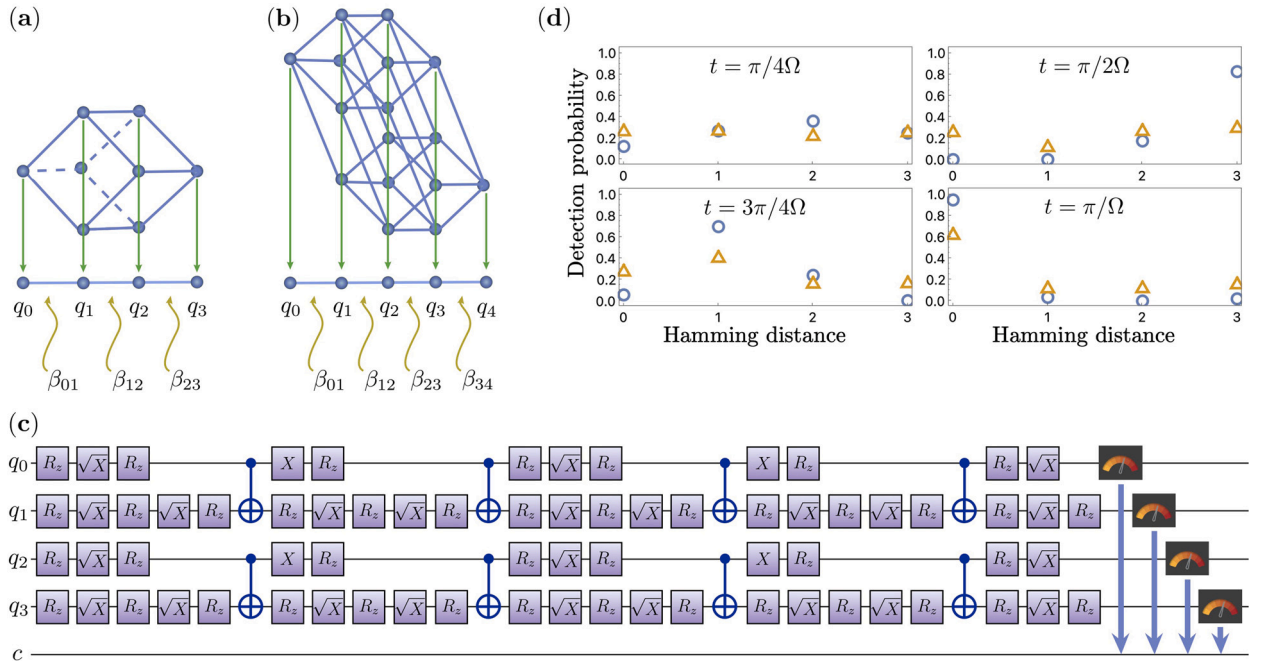


Fig. 4. (a)-(b) Mapping an n -dimensional hypercube graph to a line graph. Examples for $n = 3$ (a) and $n = 4$ (b) are shown. Weights β are defined in Eq. (11). (c) Quantum circuit for a CTQW implemented for $t = \pi/2\Omega$ on four entangled qubits. (d) Quantum and classical simulation results for implementing a CTQW on a weighted line with 4 vertices. Each plot shows a probability distribution of particles position on a graph at different times: $t = \pi/4\Omega, \pi/2\Omega, 3\pi/4\Omega$, and π/Ω . The blue circles and orange triangles correspond to execution on the simulator and the IBM Q quantum register, respectively. Hamming distance indicates the distance at which particle propagated starting from the initial state corresponding to an excited state of the q_0 qubit.

of the Hamiltonian corresponding to the adjacency matrix of the graph. In this section, we discuss how to simulate: i) hypercube graphs; ii) arbitrary graphs on a quantum computer using quantum circuits.

4.1. Quantum walks on hypercubes

A different mapping can be used to simulate a CTQW on a hypercube. In this mapping, one exploits the fact that there is an equal probability of detecting a particle in vertices with the same Hamming distance. By taking all the symmetries into account, one obtains a weighted line graph. The mapping procedure is shown in Fig. 4 for $n = 3$ (a) and $n = 4$ (b). To construct the Hamiltonian that governs the time evolution of the quantum walk in the mapped space, we use XY coupling terms, where:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{9}$$

and

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \tag{10}$$

are the Pauli matrices as represented in Eq. (9) and Eq. (10) respectively. The nodes of the weighted line graph are implemented as qubits and are expressed as X and Y terms in our Hamiltonian. The weights of the edges of the graph are specified by the strength of the couplings between qubits. In our Hamiltonian, we represent them as coefficients of the XY coupling terms $\beta_{i,i+1} = \sqrt{(1+i)(n-i)}$, where n is the dimension of the hypercube and the indices i represent qubits or nodes of the graph. The Hamiltonian that governs the quantum walk in the mapped space can be defined for an arbitrary n :

$$H^{\text{hc} \rightarrow \text{line}} = \frac{1}{2} \sum_{i=0}^{n-1} \beta_{i,i+1} (X_i X_{i+1} + Y_i Y_{i+1}). \tag{11}$$

Implementing a quantum walk in this mapping has an advantage of the exponential reduction of the number of qubits, as there are only $(n + 1)$ qubits needed for n -dimensional hypercube CTQW implementation. This is similar to the mapping with separable qubits considered before, which required n qubits.

For the simulation of CTQW on a hypercube with this mapping, we have to implement the above Hamiltonian in Eq. (11) on the IBM quantum computer. Here, we simulate a hypercube of dimension $n = 3$ mapped to a weighted line graph. The corresponding Hamiltonian for the weighted line graph can be obtained from Eq. (11) for $n = 3$ as shown in Eq. (12),

$$H^{\text{hc} \rightarrow \text{line}} = \frac{\sqrt{3}}{2} (X_0 X_1 + Y_0 Y_1) + (X_1 X_2 + Y_1 Y_2) + \frac{\sqrt{3}}{2} (X_2 X_3 + Y_2 Y_3). \tag{12}$$

The 16×16 adjacency matrix of the line graph corresponds to this Hamiltonian. Since the terms of the Hamiltonian do not commute, we use Trotter decomposition to simulate the Hamiltonian. Trotter decomposition can be used to accurately simulate the Hamiltonian’s unitary time evolution by breaking it up into a series of short time-steps as shown in Eq. (13)

$$\exp \left[-i \sum_{j=1}^m H_j t \right] = \prod_{j=1}^m \exp [-i H_j t] + O(m^2 t^2), \tag{13}$$

where m is the number of time-steps.

Noting that Trotter decomposition is currently widely used in the framework of Quantum Approximate Optimization Algorithm (QAOA), which uses both quantum and classical computer resources, see, e.g., [44,45]. In practice, if the number of variational parameters is large enough, QAOA can solve the MaxCut problem with high enough accuracy.

Ideally, a larger number of time-steps in the decomposition leads to more accurate results. But in our case, we have to keep in mind the depth of the circuit, as a larger circuit depth can lead to an accumulation of errors. In our Trotter decomposition, we keep the number of time-steps at $m = 6$ to have a smaller circuit depth without compromising the accuracy of the results. In general, as the dimension of the hypercube increases, the circuit depth increases by a factor of two gates. The corresponding quantum circuit obtained for the CTQW simulation is given in Fig. 4(c). The circuit consists of R_z and \sqrt{X} gate. The R_z is defined in Eq. (14) as,

$$R_z(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}. \tag{14}$$

The \sqrt{X} gate is defined in Eq. (15) as,

$$\sqrt{X} = \begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix}. \tag{15}$$

In this mapping, the nodes of the line are encoded using one-hot encoding. In this encoding, the states are represented by bit strings, which consist of “1” at nodes where the particle can be found and zeros elsewhere. Therefore, the total Hilbert space of the CTQW gets reduced to these states. The described encoding helps in error-correcting all the other states those are not valid in the one-hot encoding, which we obtain while implementing the CTQW on IBM Q devices.

For $n = 3$ hypercube, which is shown in Fig. 4(a), we can compare this evolution of the CTQW mapped to a line with the CTQW implemented on individual qubits as shown in Fig. 2(c). We observe that the walk’s evolution on each time step is similar for both implementations if we correct the experimental errors occurring during the implementation. The quantum and classical simulation results are shown in Fig. 4(d). Compared to the simulation results in Fig. 2(c), the probabilities mismatch has two origins: errors because of the Trotterization procedure, and error because of the larger depth of the experimental quantum circuit. In both Fig. 2

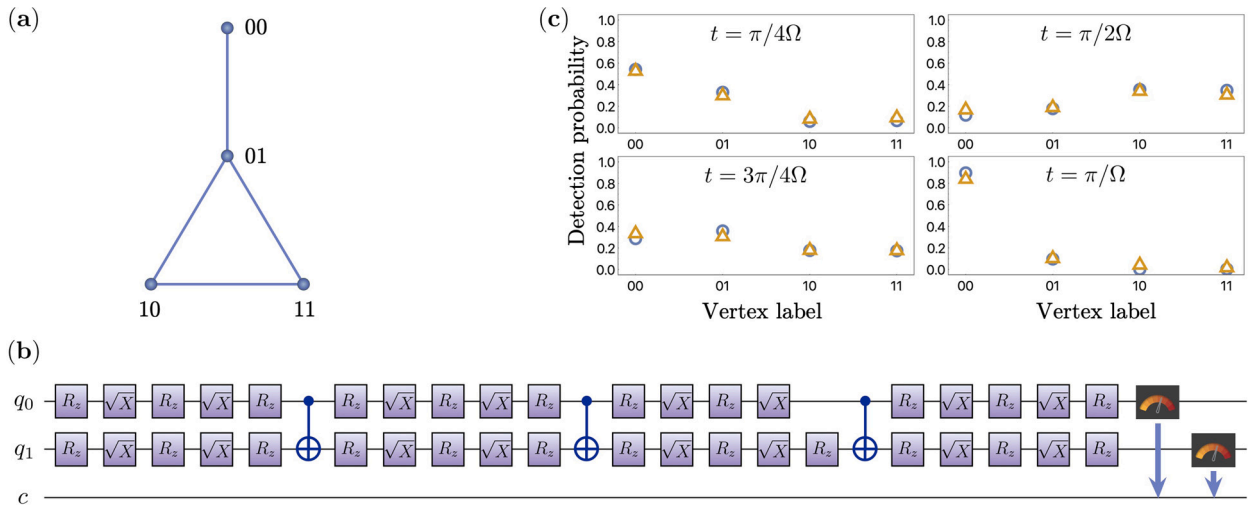


Fig. 5. (a) A ‘paw’ graph with 4 nodes labelled in order is shown (b) Quantum circuit for a CTQW implemented for $t = \pi/2\Omega$ on the ‘paw’ graph. (c) Quantum and classical simulation results for implementing a CTQW ‘paw’ graph with 4 vertices. Each plot shows a probability distribution of particles position on a graph at different times: $t = \pi/4\Omega$, $\pi/2\Omega$, $3\pi/4\Omega$, and π/Ω . The blue circles and orange triangles correspond to execution on the simulator and the IBM Q quantum register, respectively. Vertex labels indicate the labelling of different vertices of the graph.

and Fig. 4, the transport between opposite hypercube vertices should be noticed. In the case of the implementation shown in Fig. 4, also corresponds to transport in quantum spin networks [46].

4.2. Simulating arbitrary graphs using quantum circuits

In this section, we discuss how an arbitrary graph can be mapped to a quantum circuit. We know that the adjacency matrix of a graph corresponds to the Hamiltonian which governs the state evolution of the quantum particle on the graph. Therefore, in order to simulate a graph on a quantum computer, we need to find an efficient way to decompose the unitary matrix which corresponds to the state evolution of the Hamiltonian into Pauli matrices. Hamiltonian simulation method finds an efficient decomposition of the unitary state evolution of a Hamiltonian into a product of Pauli terms which in turn, can be simulated in a quantum computer by using gates which are native to the architecture. We can use the Hamiltonian simulation technique to simulate a graph on a quantum computer by mapping the graph to a quantum circuit.

For example, we simulated a ‘paw’ graph with 4 nodes as shown in Fig. 5. Also, shown is the quantum circuit obtained from Hamiltonian simulation using Qiskit implemented on the IBM quantum computer. The feasibility of implementing an arbitrary graph on a NISQ device can be evaluated by estimating the complexity of the Hamiltonian simulation method used. The query complexity of the Hamiltonian simulation directly depends on the size or complexity of the graph. We will discuss more about Hamiltonian simulation methods and their complexity in the next section.

4.3. Hamiltonian simulation

Hamiltonian simulation methods in quantum information science address the problem of efficient simulation of quantum systems. The goal of the algorithm is to find an approximation to a unitary matrix U such that, $\|U - e^{-i\Omega t A}\| \leq \epsilon$ where ϵ is the maximum simulation error and $\|\cdot\|$ is the spectral norm. There are different algorithms or techniques with varying complexity used for simulating the unitary state evolution of a Hamiltonian in a quantum computer. They can be mainly divided into divide and conquer algorithms and quantum walk algorithms. We can use Hamiltonian simulation techniques to simulate an arbitrary graph on a quantum computer. The complexity of the algorithm determines how efficiently we can simulate graphs on a quantum computer. Therefore, larger complex graphs are more challenging to be simulated on quantum computers. Thus creating the need to improve the performance of current Hamiltonian simulation algorithms more important. Some of the most prominent algorithms for Hamiltonian simulation along with their query complexities are given in Table 1.

5. Conclusion

We demonstrate the feasibility of NISQ device for implementation of CTQW. Quantum walks are implemented utilizing classical and quantum simulation, where quantum simulations are performed on currently available quantum computers of IBM Q. All quantum walks are implemented with the number of qubits that scale logarithmically with the graph size.

The presented results consist of two parts. First, a mapping between circuits and graphs is shown. A method to obtain the CTQW on a graph corresponding to a quantum circuit is discussed. The quantum circuit is simulated on the state-of-the-art IBM quantum computer to obtain the graphs. Hypercube graphs and arbitrary graphs are obtained. In the second part, a method to map graphs to

Table 1

Algorithms for Hamiltonian simulation along with their query complexity where k is the sparsity, t is the time parameter and ϵ is the error parameter. Here QW refers to Quantum walks and QSP refers to Quantum Signal Processing respectively.

Algorithms	Complexity
Product formulas [47,48]	$O(d^3 t (dt/\epsilon)^{1/2k})$
Phase estimation on Quantum Walks (QW) [49,50]	$O(dt/\sqrt{\epsilon})$
Fractional queries [51] or Truncated Taylor series [52]	$O(d^2 t \frac{\log(d^2 t/\epsilon)}{\log \log(d^2 t/\epsilon)})$
Linear combination of QW [53]	$O(dt \frac{\log(dt/\epsilon)}{\log \log(dt/\epsilon)})$
Quantum Signal Processing [54]	$O(dt + \log(1/\epsilon))$
Qubitization/ QSP [55]	$O(d^2 t + \log(1/\epsilon))$

the circuit is studied. Hypercubes and arbitrary graphs are simulated on a quantum computer using quantum circuits. Both classical and quantum simulation results are obtained and compared. The technique of Hamiltonian simulation is discussed and various algorithms for Hamiltonian simulation are listed with their complexities.

With our work, we hence established an analogy between quantum circuits and graphs that allowed us to tackle computationally challenging simulation problems. This result paves the way towards the practical realization of quantum advantage in quantum walk simulation for algorithm development.

CRedit authorship contribution statement

Anandu Kalleri Madhu and Alexey Melnikov: Conceived and designed the experiments.

Anandu Kalleri Madhu and Alexander Alodjants: Contributed analysis tools and data.

Anandu Kalleri Madhu: Performed the experiments.

Anandu Kalleri Madhu, Alexey Melnikov, Leonid Fedichkin, Alexander Alodjants, Ray-Kuang Lee: Analyzed and interpreted the data; Wrote the paper.

Funding statement

Leonid Fedichkin was supported by Ministry of Science and Higher Education of the Russian Federation for Valiev Institute of Physics and Technology of RAS [Program No. FFNN-2022-0016].

Alexander Alodjants was supported by Ministry of Science and Higher Education of the Russian Federation and South Ural State University [agreement No. 075-15-2022-1116].

Ray-Kuang Lee was supported by Ministry of Science and Technology of Taiwan [No. 109-2112-M-007-019-MY3, 109-2627-M-008-001, 110-2123-M-007-002].

Declaration of competing interest

The authors declare no competing interests.

Data availability

Data associated with this study has been deposited at <https://github.com/q-ml/quantum-walks-quantum-devices>.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.heliyon.2023.e13416>.

References

- [1] M. Kac, Random walk and the theory of Brownian motion, *Am. Math. Mon.* 54 (1947) 369.
- [2] F. Bartumeus, M.G.E. da Luz, G.M. Viswanathan, J. Catalan, Animal search strategies: a quantitative random-walk analysis, *Ecology* 86 (2005) 3078.
- [3] D. Brockmann, L. Hufnagel, T. Geisel, The scaling laws of human travel, *Nature* 439 (2006) 462.
- [4] E.A. Codling, M.J. Plank, S. Benhamou, Random walk models in biology, *J. R. Soc. Interface* 5 (2008) 813.
- [5] R. Motwani, P. Raghavan, *Randomized Algorithms*, Cambridge University Press, New York, USA, 1995, Chap. 6.
- [6] F. Wang, D.P. Landau, Efficient, multiple-range random walk algorithm to calculate the density of states, *Phys. Rev. Lett.* 86 (2001) 2050.
- [7] T. Sottinen, Fractional Brownian motion, random walks and binary market models, *Finance Stoch.* 5 (2001) 343.
- [8] K.K. Sabelfeld, N.A. Simonov, *Random Walks on Boundary for Solving PDEs*, Walter de Gruyter, 2013.

- [9] C. Gkantsidis, M. Mihail, A. Saberi, Random walks in peer-to-peer networks: algorithms and evaluation, *Perform. Eval.* 63 (2006) 241.
- [10] D.T. Gillespie, Monte Carlo simulation of random walks with residence time dependent transition probability rates, *J. Comput. Phys.* 28 (1978) 395.
- [11] M.B. Cohen, J. Kelner, J. Peebles, R. Peng, A. Sidford, A. Vladu, Faster algorithms for computing the stationary distribution, simulating random walks, and more, in: 57th Annual Symposium on Foundations of Comp. Sci., 2016, pp. 583–592.
- [12] Y. Aharonov, L. Davidovich, N. Zagury, Quantum random walks, *Phys. Rev. A* 48 (1993) 1687.
- [13] J. Kempe, Quantum random walks: an introductory overview, *Contemp. Phys.* 44 (2003) 307.
- [14] N. Konno, *Quantum Walks in Quantum Potential Theory*, Springer, 2008, pp. 309–452.
- [15] S.E. Venegas-Andraca, Quantum walks: a comprehensive review, *Quantum Inf. Process.* 11 (2012) 1015.
- [16] E. Farhi, S. Gutmann, Quantum computation and decision trees, *Phys. Rev. A* 58 (1998) 915.
- [17] K. Manouchehri, J. Wang, *Physical Implementation of Quantum Walks*, Springer, Berlin, Heidelberg, 2013.
- [18] L. Fedichkin, F. Meshchaninov, Analysis and applications of quantum walks, *J. Math. Sci.* 252 (2020) 104.
- [19] H. Krovi, T.A. Brun, Hitting time for quantum walks on the hypercube, *Phys. Rev. A* 73 (2006) 032341.
- [20] D. Solenov, L. Fedichkin, Continuous-time quantum walks on a cycle graph, *Phys. Rev. A* 73 (2006) 012313.
- [21] L. Fedichkin, D. Solenov, C. Tamon, Mixing and decoherence in continuous-time quantum walks on cycles, *Quantum Inf. Comput.* 6 (2006) 263.
- [22] A.A. Melnikov, L.E. Fedichkin, Quantum walks of interacting fermions on a cycle graph, *Sci. Rep.* 6 (2016) 34226.
- [23] Q.-P. Su, Y. Zhang, L. Yu, J.-Q. Zhou, J.-S. Jin, X.-Q. Xu, S.-J. Xiong, Q. Xu, Z. Sun, K. Chen, et al., Experimental demonstration of quantum walks with initial superposition states, *npj Quantum Inf.* 5 (2019) 1.
- [24] W.-X. Cui, Y. Xing, L. Qi, X. Han, S. Liu, S. Zhang, H.-F. Wang, Quantum walks in periodically kicked circuit qed lattice, *Opt. Express* 28 (2020) 13532.
- [25] G.S. Engel, T.R. Calhoun, E.L. Read, T.-K. Ahn, T. Mančal, Y.-C. Cheng, R.E. Blankenship, G.R. Fleming, Evidence for wavelike energy transfer through quantum coherence in photosynthetic systems, *Nature* 446 (2007) 782.
- [26] M. Mohseni, P. Rebentrost, S. Lloyd, A. Aspuru-Guzik, Environment-assisted quantum walks in photosynthetic energy transfer, *J. Chem. Phys.* 129 (2008) 174106.
- [27] E. Harel, G.S. Engel, Quantum coherence spectroscopy reveals complex dynamics in bacterial light-harvesting complex 2 (LH2), *Proc. Natl. Acad. Sci.* 109 (2012) 706.
- [28] F. Flamini, N. Spagnolo, F. Sciarrino, Photonic quantum information processing: a review, *Rep. Prog. Phys.* 82 (2018) 016001.
- [29] A. Bazhenov, M. Nikitina, A.P. Alodjants, High temperature superradiant phase transition in quantum structures with a complex network interface, *Opt. Lett.* 47 (2022) 3119.
- [30] A.A. Melnikov, L.E. Fedichkin, A. Alodjants, Predicting quantum advantage by quantum walk with convolutional neural networks, *New J. Phys.* 21 (2019) 125002.
- [31] A.A. Melnikov, L.E. Fedichkin, R.-K. Lee, A. Alodjants, Machine learning transfer efficiencies for noisy quantum walks, *Adv. Quantum Technol.* 3 (2020) 1900115.
- [32] A. Kryukov, R. Abramov, L.E. Fedichkin, A. Alodjants, A.A. Melnikov, Supervised graph classification for chiral quantum walks, *Phys. Rev. A* 105 (2022) 022208.
- [33] X. Qiang, T. Loke, A. Montanaro, K. Aungskunsiri, X. Zhou, J.L. O'Brien, J.B. Wang, J.C. Matthews, Efficient quantum walk on a quantum processor, *Nat. Commun.* 7 (2016) 1.
- [34] F. Acasiete, F.P. Agostini, J.K. Moqadam, R. Portugal, Implementation of quantum walks on IBM quantum computers, *Quantum Inf. Process.* 19 (2020) 1.
- [35] N.C. Harris, G.R. Steinbrecher, M. Prabhu, Y. Lahini, J. Mower, D. Bunandar, C. Chen, F.N. Wong, T. Baehr-Jones, M. Hochberg, et al., Quantum transport simulations in a programmable nanophotonic processor, *Nat. Photonics* 11 (2017) 447.
- [36] C. Maier, T. Brydges, P. Jurcevic, N. Trautmann, C. Hempel, B.P. Lanyon, P. Hauke, R. Blatt, C.F. Roos, Environment-assisted quantum transport in a 10-qubit network, *Phys. Rev. Lett.* 122 (2019) 050501.
- [37] C.-C. Chen, S.-Y. Shiau, M.-F. Wu, Y.-R. Wu, Hybrid classical-quantum linear solver using noisy intermediate-scale quantum machines, *Sci. Rep.* 9 (2019) 1.
- [38] A.M. Childs, Universal computation by quantum walk, *Phys. Rev. Lett.* 102 (2009) 180501.
- [39] A.M. Childs, D. Gosset, Z. Webb, Universal computation by multiparticle quantum walk, *Science* 339 (2013) 791.
- [40] J. Preskill, Quantum computing in the NISQ era and beyond, *Quantum* 2 (2018) 79.
- [41] Z. Zimboras, M. Faccin, Z. Kadar, J.D. Whitfield, B.P. Lanyon, J. Biamonte, Quantum transport enhancement by time-reversal symmetry breaking, *Sci. Rep.* 3 (2013) 2361.
- [42] D. Lu, J.D. Biamonte, J. Li, H. Li, T.H. Johnson, V. Bergholm, M. Faccin, Z. Zimborás, R. Laflamme, J. Baugh, S. Lloyd, Chiral quantum walks, *Phys. Rev. A* 93 (2016) 042302.
- [43] T.A. Loring, Computing a logarithm of a unitary matrix with general spectrum, *Numer. Linear Algebra Appl.* 21 (2014) 744.
- [44] E. Farhi, J. Goldstone, S. Gutmann, A quantum approximate optimization algorithm, arXiv:1411.4028, 2014.
- [45] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, M. Lukin, Quantum approximate optimization algorithm: performance, mechanism, and implementation on near-term devices, *Phys. Rev. X* 10 (2020) 021067.
- [46] M. Christandl, N. Datta, A. Ekert, A.J. Landahl, Perfect state transfer in quantum spin networks, *Phys. Rev. Lett.* 92 (2004) 187902.
- [47] D.W. Berry, G. Ahokas, R. Cleve, B.C. Sanders, E-efficient quantum algorithms for simulating sparse Hamiltonians, *Commun. Math. Phys.* 270 (2007) 359.
- [48] A.M. Childs, R. Kothari, Simulating sparse Hamiltonians with star decompositions, in: *Conf. on Quantum Comput., Commun., and Cryptogr.*, Springer, 2010, pp. 94–103.
- [49] A.M. Childs, On the relationship between continuous- and discrete-time quantum walk, *Commun. Math. Phys.* 294 (2010) 581.
- [50] D.W. Berry, A.M. Childs, Black-box Hamiltonian simulation and unitary implementation, preprint, arXiv:0910.4157, 2009.
- [51] D.W. Berry, A.M. Childs, R. Cleve, R. Kothari, R.D. Somma, Exponential Improvement in Precision for Simulating Sparse Hamiltonians, *Forum of Mathematics, Sigma*, vol. 5, Cambridge University Press, 2017.
- [52] D.W. Berry, A.M. Childs, R. Cleve, R. Kothari, R.D. Somma, Simulating Hamiltonian dynamics with a truncated Taylor series, *Phys. Rev. Lett.* 114 (2015) 090502.
- [53] D.W. Berry, A.M. Childs, R. Kothari, Hamiltonian simulation with nearly optimal dependence on all parameters, in: *Proc. - Annu. IEEE Symp. Found. Comput. Sci. FOCS, IEEE*, 2015, pp. 792–809.
- [54] G.H. Low, I.L. Chuang, Optimal Hamiltonian simulation by quantum signal processing, *Phys. Rev. Lett.* 118 (2017) 010501.
- [55] G.H. Low, I.L. Chuang, Hamiltonian simulation by qubitization, *Quantum* 3 (2019) 163.