Circuit Implementation of Discrete-Time Quantum Walks via the Shunt Decomposition Method

Allan Wing-Bocanegra

Tecnologico de Monterrey, Escuela de Ingenieria y Ciencias
Ave. Eugenio Garza Sada 2501, Monterrey, 64849, N.L., Mexico.
A00832476@tec.mx

Salvador E. Venegas-Andraca

Tecnologico de Monterrey, Escuela de Ingenieria y Ciencias
Ave. Eugenio Garza Sada 2501, Monterrey, 64849, N.L., Mexico.
svenegas@tec.mx

April 5, 2023

Abstract

Several models have been proposed to build evolution operators to perform quantum walks in a theoretical way, although when wanting to map the resulting evolution operators into quantum circuits to run them in quantum computers, it is often the case that the mapping process is in fact complicated. Nevertheless, when the adjacency matrix of a graph can be decomposed into a sum of permutation matrices, we can always build a shift operator for a quantum walk that has a block diagonal matrix representation. In this paper, we analyze the mapping process of block diagonal operators into quantum circuit form, and apply this method to obtain quantum circuits that generate quantum walks on the most common topologies found in the literature: the straight line, the cyclic graph, the hypercube and the complete graph. The obtained circuits are then executed on quantum processors of the type Falcon r5.11L and Falcon r4T (two of each type) through IBM Quantum Composer platform and on the Qiskit Aer simulator, performing three steps for each topology. The resulting distributions were compared against analytical distributions, using the statistical distance ℓ_1 as a performance metric. Regarding experimental executions, we obtained short ℓ_1 distances in the cases of quantum circuits with a low amount of multi-control gates, being the quantum processors of the type Falcon r4T the ones that provided more accurate results.

1 Introduction

A random walk is a process describing the dynamics an agent follows by taking random steps on the vertices of a connected graph, i.e. it is a stochastic process formed by successive summation of independent, identically distributed random variables [1]. This mathematical model has been widely used in many areas of science such as in chemistry to study the configurations of polymer-based materials [2], in biology to study the movement and dispersal of animals and microorganisms [3], in computer science to perform image cosegmentation [4] and

in economics to study the behaviour of stock markets [5], just to name a few examples and bring to light the versatility of random walks as a mathematical tool.

Quantum walks constitute a promising universal model of quantum computation originally inspired in random walks, that also studies the dynamics of a walker moving on the vertices of a graph, but in this case the walker can be in a quantum superposition of states and, thus, take multiple directions and have different phases at each step. This brings special properties such as quadratically wider spread of probability distribution [6] and quadratically faster hitting time [7], which are key features to obtain a speed-up over classical algorithms. Quantum walks can be either discrete [8] or continuous [9] in time. Within the field of discrete-time quantum walks we can find coined [10] or coinless [11, 12] models. In this paper, we focus on the Coined Discrete-Time Quantum Walk model [13], which is usually just called Discrete-Time Quantum Walk (DTQW) in the literature since it was the first model of its type.

The Discrete-Time Quantum Walk model is an active research field that has already proved to be fruitful as it has been used to solve or attempt to solve problems such as image and public-key encryption [14, 15], the search of elements in an unstructured database [16, 17], the design and training of neural networks [18, 19], and the quantization of the PageRank algorithm [20, 21, 22], among others, providing a speed-up when compared with random-walk approaches for the same problems. Nevertheless, up to date, quantum walk-based algorithms have mostly been tested theoretically or through simulations [16, 18], and not experimentally due to the fact that quantum computers of the current Noisy Intermediate-Scale Quantum (NISQ) era [23], generate much noise and decoherence, and thus are not suitable to calculate efficiently the probability distributions of quantum walks except for very simple cases, such as few steps of a quantum walk on a line of few nodes as presented by K. N. Cassemiro *et al.* [24] and M. A. Broome *et al.* [25] on specific quantum processors for this task, and by A. Shakeel [26] and K. Georgopoulos [27] on general-purpose quantum computers. Efficient experimental implementations of quantum walks on different topologies have been done using other models of quantum walks, i.e. staggered quantum walks [28], topological quantum walks [29] and continuous-time quantum walks [30, 31, 32], being one key factor, according to [28], the fact that coinless (e.g. staggered, topological or continuous) quantum walks do not need an extra register for the coin state, and in NISQ computers additional qubits induce more noise to the system, thus reducing fidelity in experimental runs.

As a step forward to efficiently run DTQW-based algorithms on quantum computers, in this work we study the implementation of DTQW on IBM quantum computers and simulators, for the most common topologies found in the literature, i.e. the line, cycle, hypercube, and complete graphs. We will analyze how to theoretically build the circuits based on the shunt decomposition method, how to optimize these circuits using different techniques, how to implement them on IBM quantum computers and simulators, and, finally, as the decompositions are not unique, we will compare their performance in order to choose the decomposition that behaves best and is therefore suitable to be scaled for a larger number of qubits.

The structure of this paper is the following: In section 2 we introduce the theoretical basis of DTQW and the shunt decomposition method, the method we use to build evolution operators, as block diagonal unitary operators. We then show, in section 3, the general way in which block diagonal unitary operators can be mapped into quantum circuit representation in order to be implemented in quantum computers. In section 4 we apply the method described in section 3 for the n-line, the n-cycle, the n-hypercube, and the 2^n-complete graph. We also provide a synthesis for some circuits based on a matrix analysis approach of their evolution operators, and on different techniques to reduce networks of multi-control gates. In section 5 we implement the quantum circuits derived in section 4 on IBM quantum computers and Qiskit simulators. In this section we also provide the experimental distributions of the quantum states after measurement, and compare them with

the theoretical ones. Finally, in section 6, we present the conclusions

2   Theory of Discrete-Time Quantum Walks

A Discrete-Time Quantum Walk, considers a walker that moves between the vertices of a graph $\mathcal{G}(V, A)$ with adjacency matrix $\mathcal{A}$, where V and A are the vertex and arc sets, respectively. A DTQW is defined by three elements: the quantum state of a walker, the evolution operator of the system and a set of measurement operators. The state of the walker at time t is given by the composite quantum state presented in Eq. (1):

$$|\psi(t)\rangle = \sum_{i,j} a_{ij}|c_i\rangle \otimes |v_j\rangle \tag{1}$$

where $|v_j\rangle \in \mathcal{H}_P$ is a state associated to vertex $v_j \in \mathbb{Z}$, and $|c_k\rangle \in \mathcal{H}_C$, with $c_k \in \mathbb{Z}$, is a state associated to a set of arcs $(v_j, v_i) \in A$. The relation of relation coin states and arcs of $\mathcal{G}$ is explained in the next paragraphs (see Eq. (6)). $\mathcal{H}_P$ and $\mathcal{H}_C$ are complex Hilbert spaces of size n' and m', and are called position and coin spaces, respectively. Consider $|v_j\rangle$ and $|c_k\rangle$ to be represented by the corresponding canonical basis. The evolution operator U is given by the product of the shift and coin operators, i.e.

$$U = S(C \otimes I_{n'}) \tag{2}$$

One step of the walker consists in the application of U to $|\psi(t)\rangle$, the walker standing on vertex $v_j \in V$ first gets in a superposition of coin states by the action of $C \otimes I_{n'}$, and then moves toward all the adjacent vertices at the same time, by the action of S. The state of the system after t steps is given by $|\psi(t)\rangle = U^t|\psi_0\rangle$, where $|\psi_0\rangle$ is the initial state of the walker.

Finally, we define the measurement operators of the system as

$$M_j = I_{m'} \otimes |v_j\rangle\langle v_j| \tag{3}$$

and the probability to find a walker on vertex v_j after t steps is given by

$$P(|v_j\rangle) = \langle\psi(t)|M_j^\dagger M_j|\psi(t)\rangle \tag{4}$$

Different methods to construct evolution operators to perform walks on different topologies have been proposed. Particularly, the case in which the shift operator of a DTQW is a block diagonal matrix, whose block elements are permutation matrices that allows us to easily manipulate S. Godsil and Zhan [33] named this method *shunt decomposition*, and studied its application to perform DTQWs on regular graphs. Montanaro [34] went further and proved the method to work for strongly connected digraphs, i.e. digraphs in which every pair of vertices are connected by a path, although it also works for graphs with strongly connected subgraphs that do not necessarily connect between them, as will be presented in a particular case later in this paper.

Now we explain the construction of the evolution operator. Let $\mathcal{G}$ be m'-regular graph, with adjacency matrix $\mathcal{A}(\mathcal{G})$. Suppose $\mathcal{A}^\mathsf{T}$ can be decomposed as the sum of m' permutation matrices, $\mathcal{P}_i^\mathsf{T} \in \mathbb{R}^{n' \times n'}$, i.e.

$$\mathcal{A}^\mathsf{T} = \mathcal{P}_0^\mathsf{T} + \mathcal{P}_1^\mathsf{T} + \cdots + \mathcal{P}_{m'-1}^\mathsf{T} \tag{5}$$

We call each permutation matrix a *shunt*. Next, we associate each shunt with a coin basis state, and define the shift operator of the system as expressed in Eq. (6).

$$S = \sum_{i=0}^{m'-1} |c_i\rangle\langle c_i| \otimes \mathcal{P}_i^\mathsf{T} \tag{6}$$

Thus, a walker with coin state $|c_i\rangle$ will only be able to move through arcs associated with shunt $\mathcal{P}_i^\mathsf{T}$. Notice that we use matrices $\mathcal{P}_i^\mathsf{T}$ to construct S, since otherwise the quantum walker would move in the opposite direction to the arcs of $\mathcal{G}$.

In explicit matrix notation, the shift operator takes the following form

$$S = \begin{pmatrix} \mathcal{P}_0^\mathsf{T} & 0 & \dots & 0 \\ 0 & \mathcal{P}_1^\mathsf{T} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathcal{P}_{m'-1}^\mathsf{T} \end{pmatrix} \tag{7}$$

Because S is a block diagonal matrix with unitary matrices as entries, S is unitary too, and contains the same information of the connections between nodes than the original adjacency matrix.

The evolution operator can be completed by the choice of any coin operator C of dimension $m' \times m'$. Nevertheless, in this work we will focus on the study of the circuit implementation of shift operators, and thus we decided to always use Grover and Hadamard operators for this task.

Finally, the evolution operator is applied on a bipartite state $|\psi(t)\rangle \in H_C \otimes H_P$, with $dim(H_C) = m'$ and $dim(H_P) = n'$, and the measurement operators act as defined in Eqs. (3) and (4).

3 Mapping of DTQWs to the Quantum Circuit Model

Let $\mathcal{G}(V, A)$ be a graph with connected components of order $n' = 2^n$, whose adjacency matrix can be decomposed into the sum of $m' = 2^m$ permutation matrices and whose vertices are labeled by integers. Associated to the vertices of the graph $\mathcal{G}$, there exists a set of 2^n canonical basis vectors $\{|v_j\rangle : v_j \in \mathbb{Z}\}$ that span H_P, and associated to the arcs connected to vertex v_i there is a set of 2^m canonical basis vectors $\{|c_i\rangle : c_i \in \mathbb{Z}\}$ that span H_C.

In order to map the basis states of H_c and H_p into a composite state of qubits, we must map v_j and c_k into bitstring notation. Thus, we define the function $f_V : \mathbb{Z} \to B$, where B is the set of all bitstrings, and consider the basis states of a qubit to be given by

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{8}$$

In this way, we are able to do the following map

$$|f_V(v_i)\rangle = |q_n \dots q_1 q_0\rangle = |q_n\rangle \otimes \dots \otimes |q_1\rangle \otimes |q_0\rangle \tag{9}$$

where $q_i \in \{0, 1\}$.

This means that every basis vector $|v_i\rangle$ can be rewritten as a composite state given by the tensor product of n qubits. The same mapping is done for the coin space, allowing us to express the state vector of a quantum walker, $|\psi\rangle = |c_i\rangle \otimes |v_i\rangle$, in bitstring notation.

This mapping can be represented in an $(n+m)$-qubit quantum circuit, where the first n qubits form a register $|q_{n-1} \dots q_1 q_0\rangle$ that corresponds to the position states, and the last m qubits form a register $|q_{n+m-1} \dots q_{n+1} q_n\rangle$

(a)

(b)

Figure 1 (a) Example of a multi-control-$\mathcal{P}_i^\mathsf{T}$ gate, along with its matrix representation. The pattern of white and black dots can be interpreted as binary code, and defines the position of $\mathcal{P}_i^\mathsf{T}$ in the matrix representation of $C_{3,7}(\mathcal{P}_i^\mathsf{T})$, which follows the form of Eq. (7). (b) Circuit implementation of a block diagonal shift operator. Notice the controls and the target of each gate span the coin and position registers, respectively, which holds whenever mapping a block diagonal operator to circuit form

that corresponds to the coin states. Note that the leftmost state is associated with the less significant bit in bitstring representation.

To completely map a DTQW into a quantum circuit, we need to express the evolution operator, $U = S(C \otimes I_P)$, as a set of quantum gates. A general method can be followed for graphs whose adjacency matrices can be decomposed in 2^m permutation matrices $\mathcal{P}_i^\mathsf{T}$ of size 2^n, as in Eq. (5), which will be used as block diagonal elements of S. Given that all matrices $\mathcal{P}_i^\mathsf{T}$ are unitary, there exists a quantum gate representation for all of them. Out of the quantum gate $\mathcal{P}_i^\mathsf{T}$, we can create a controlled gate, $C_{m,j}(\mathcal{P}_i^\mathsf{T})$, which uses all the qubits of the coin register as control qubits, as shown in Fig. 1(a). We call the black and the white dots in this figure *controls*. The pattern that they create can be interpreted as binary code, where the black controls represent a 1, the white controls represent a 0 and the less significant bit of the bitstring is the uppermost control in the position register. The second subindex $j = 0, 1, \ldots, 2^m - 1$ in $C_{m,j}(\mathcal{P}_i^\mathsf{T})$ represents the value of the bitstring formed by the control qubits of the gate, while m represents the number of control qubits. Notice that we reserve this notation for fully control gates which use the qubits at the bottom of the target gate as control qubits. If the target gate is controlled using the top qubits we use the notation $C^{r,k}(U)$. In the case the gate is fully controlled using both top and bottom qubits, we use the notation $C_{s,l}^{r,k}(U)$.

The result of controlling $\mathcal{P}_i^\mathsf{T}$ is a 2^{n+m} block diagonal matrix, where all the diagonal elements of the matrix are the $2^n \times 2^n$ identity, I_{2^n}, except for the jth element, which will be the matrix $\mathcal{P}_i^\mathsf{T}$. This is given by Eq. (10):

$$C_{m,j}(\mathcal{P}_i^\mathsf{T}) = I_{2^{jn}} \oplus \mathcal{P}_i^\mathsf{T} \oplus I_{2^{(2^m-j-1)n}} \tag{10}$$

where $i, j = 0, 1, \ldots, 2^m - 1$ and we define I_0 as a zero-dimensional matrix.. Notice that i and j do not have to coincide, which just means that the matrices $\mathcal{P}_i^\mathsf{T}$ can be shuffled in the diagonal indistinguishably.

This way, we can find 2^m block diagonal gates $C_{m,j}(\mathcal{P}_i^\mathsf{T})$ in which the matrices $\mathcal{P}_i^\mathsf{T}$ are placed in a different

position j relative to each other, and the rest of the elements are I_{2^n}. When these controlled gates are placed next to each other in a quantum circuit, as in Fig. 1(b), the associated matrix representation is given by

$$S = \prod_{j=0}^{2^m-1} C_{m,j}(\mathcal{P}_i^{\mathsf{T}}) \tag{11}$$

a block diagonal operator whose elements are the additive decomposition of the adjacency matrix associated to graph $\mathcal{G}$, i.e. we obtain Eq. (7).

To complete the mapping of the evolution operator we can add a Hadamard coin by simply placing a single-qubit Hadamard gate to all the qubits of the coin register to the left of the circuit for the shift operator.

4 Implementation of Shift Operators for DTQWs on Common Topologies

In this section, we will analyze the shunt decomposition of the adjacency matrices of the most common graphs studied in the field of quantum walks: the line with n vertices, the n-cycle graph, the 2^n-hypercube and the 2^n-complete graph with self-loops, as well as their quantum circuit implementation.

4.1 DTQW on the n-cycle graph

The adjacency matrix of a 2^n-cycle graph can be deduced following the pattern presented in [35], as shown in Eq. (12).

$$\mathcal{A}_{cycle} = \begin{pmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 0 \end{pmatrix} \tag{12}$$

In [36], Li *et al.* present general sequences of multi-control not gates that conform n-qubit increment and decrement gates, as shown in Fig. (2). The transpose of the matrix forms of these gates are given in Eqs. (13) and (14).

$$\mathcal{A}_{inc}^{\mathsf{T}} = \begin{pmatrix} 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 0 \end{pmatrix} \tag{13}$$

Figure 2 (a) n-qubit increment gate. (b) $n-$qubit decrement gate

$$\mathcal{A}_{dec}^{\mathsf{T}} = \begin{pmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \end{pmatrix} \tag{14}$$

Thus, the transpose adjacency matrix of a 2^n-cycle graph can be written as a linear combination of the n-qubit transpose increment and decrement operators

$$\mathcal{A}_{cycle}^{\mathsf{T}} = \mathcal{A}_{inc}^{\mathsf{T}} + \mathcal{A}_{dec}^{\mathsf{T}} \tag{15}$$

where both $\mathcal{A}_{inc}^{\mathsf{T}}$ and $\mathcal{A}_{dec}^{\mathsf{T}}$ are unitary.

According to Eq. (7), once we have the additive decomposition of an adjacency matrix associated to a graph $\mathcal{G}$ into unitary matrices, we can construct a shift operator for a DTQW as shown in Eq. (16).

$$S = \begin{pmatrix} \mathcal{A}_{inc}^{\mathsf{T}} & 0 \\ 0 & \mathcal{A}_{dec}^{\mathsf{T}} \end{pmatrix} \tag{16}$$

Moreover, according to Eq. (11) we can build a sequence of controlled gates such that when applied subsequently, they give out Eq. (16). The sequence of gates $C_{1,0}(\mathcal{A}_{inc}^{\mathsf{T}})$ and $C_{1,1}(\mathcal{A}_{dec}^{\mathsf{T}})$ is presented in Fig. (3). This is the common implementation of a circuit for a cycle graph, although if a further matrix approach is made to Eq. (16) the number of gates can be reduced in half. Firstly, notice that

$$\mathcal{A}_{dec} = \mathcal{A}_{inc}^{\mathsf{T}} \tag{17}$$

Now we define the following $2^n \times 2^n$ matrix

$$J = \begin{pmatrix} 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \end{pmatrix} \tag{18}$$

When J is applied simultaneously to the right and to the left of a circulant matrix, i.e. a matrix of the form presented in Eq. (19), the result is the transpose of the original matrix [26, 37].

$$C = \begin{pmatrix} c_0 & c_{n-1} & \dots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & \dots & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{n-2} & \vdots & \ddots & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \dots & c_1 & c_0 \end{pmatrix} \tag{19}$$

Given that $\mathcal{A}_{inc}^{\mathsf{T}}$ lies on this category, $\mathcal{A}_{dec}^{\mathsf{T}}$ can be written alternatively as

$$\mathcal{A}_{dec}^{\mathsf{T}} = J\mathcal{A}_{inc}^{\mathsf{T}}J \tag{20}$$

so that (16) becomes

$$S = \begin{pmatrix} \mathcal{A}_{inc}^{\mathsf{T}} & 0 \\ 0 & J\mathcal{A}_{inc}^{\mathsf{T}}J \end{pmatrix} \tag{21}$$

This matrix can be decomposed in the following way

Figure 3 Shift operator for a 2^n-cycle graph composed of an $n-$qubit-controlled increment gate and an $n-$qubit-controlled decrement gate

(a)

(b)

Figure 4 Both (a) and (b) display a quantum circuit for the S operator of a DTQW on a 2^n-cycle graph, according to Eq. (22). The gates at the sides of the increment circuit are different implementations of $C_{1,1}(J)$ (Eq. (24)). The equivalence between both circuit forms of $C_{1,1}(J)$ has been used in [38] and proven in [39]

$$S = \begin{pmatrix} I_{2^n} & 0 \\ 0 & J \end{pmatrix} \begin{pmatrix} \mathcal{A}_{inc}^{\mathsf{T}} & 0 \\ 0 & \mathcal{A}_{inc}^{\mathsf{T}} \end{pmatrix} \begin{pmatrix} I_{2^n} & 0 \\ 0 & J \end{pmatrix} \tag{22}$$

where

$$I_2 \otimes \mathcal{A}_{inc}^{\mathsf{T}} = \begin{pmatrix} \mathcal{A}_{inc}^{\mathsf{T}} & 0 \\ 0 & \mathcal{A}_{inc}^{\mathsf{T}} \end{pmatrix} \tag{23}$$

and

$$C_{1,1}(J) = \begin{pmatrix} I_{2^n} & 0 \\ 0 & J \end{pmatrix} \tag{24}$$

$C_{1,1}(J)$ can be obtained by controlling with a black control a network of σ_x gates applied to all qubits of the position register, given that $\sigma_x^{\otimes n}$ generates J, or by implementing CNOT gates in cascade configuration, as shown in Fig. (4), which displays two reduced quantum circuit implementations of the shift operator presented in Eq. (16). The equivalence between the two circuit forms of $C_{1,1}(J)$ was used in [38] to decompose general multi-control gates and proven in [39] for low dimensional instances through a recursion relation which can be used to obtain higher-dimensional instances.

The circuit presented in Fig. 4(a) was proposed by [26]. These circuits are suitable to efficiently simulate DTQWs on cycle graphs of size 2^n only.

We now present a further approach, based on the matrix analysis of the shift operator, in order to implement quantum circuits capable of simulating a DTQW on a general n-cycle graph.

First consider the matrix associated to an n-quibit increment gate (Eq. (13)). In [40], Li *et al.* define a general method to decompose a square unitary matrix of order k into a product of no more than $k(k-1)/2$ two-level unitary matrices of the same size. A special case of this decomposition is the one in which the matrices $\mathcal{T}_i$ have size $2^n \times 2^n$ and take the following form

$$\mathcal{T}_i = I_i \oplus \sigma_x \oplus I_{2^n-i-2}; \quad i = 0, 1, ..., 2^n - 2 \tag{25}$$

or in explicit matrix notation

$$\mathcal{T}_i = \begin{pmatrix} I_i & 0 & 0 \\ 0 & \sigma_x & 0 \\ 0 & 0 & I_{2^n-i-2} \end{pmatrix}, \tag{26}$$

where we define I_0 as a zero-dimensional matrix.

In simple terms, $\mathcal{T}_i$ is an identity matrix in which a σ_x matrix was replaced in the main diagonal, and the subindex i in Eq. (26) represents the number of ones above σ_x. The action of $\mathcal{T}_i$ on any matrix is to exchange the rows i and $i+1$. In group theory this operation is known as *adjacent transposition* [41], thus we call $\mathcal{T}_i$ the adjacent transposition operator.

The increment and decrement operators (Eqs. (13) and (14)) can be decomposed as the product of all matrices $\mathcal{T}_i$ in descending and ascending order, respectively, i.e.

$$\mathcal{A}_{inc}^\mathsf{T} = \mathcal{T}_0 \mathcal{T}_1 \cdots \mathcal{T}_{2^n-2} \tag{27}$$

$$\mathcal{A}_{dec}^\mathsf{T} = \mathcal{T}_{2^n-2} \cdots \mathcal{T}_1 \mathcal{T}_0 \tag{28}$$

which can be though of as applying the sequence of $\mathcal{T}_i$ matrices to the identity I_{2^n} in order to switch its rows.

Now, if instead of multiplying the whole sequence of matrices $\mathcal{T}_i$ to obtain the n-qubit transpose increment operator, we truncate the product up to the first k matrices, i.e., we obtain the operator of Eq. (29).

$$(\mathcal{A}_{inc}^k)^\mathsf{T} = \mathcal{T}_0 \mathcal{T}_1 \ldots \mathcal{T}_{k-2} \tag{29}$$

or explicitly

$$(\mathcal{A}_{inc}^k)^\mathsf{T} = (\mathcal{A}_{inc}')^\mathsf{T} \oplus I_{2^n-k} = \begin{pmatrix} (\mathcal{A}_{inc}')^\mathsf{T} & 0 \\ 0 & I_{2^n-k} \end{pmatrix} \tag{30}$$

where $(\mathcal{A}_{inc}')^\mathsf{T}$ is the $k \times k$ transpose increment operator, i.e. $(\mathcal{A}_{inc}')^\mathsf{T}$ is a $k \times k$ version of $\mathcal{A}_{inc}^\mathsf{T}$ in Eq. (13), which is always $2^n \times 2^n$, but with $k \le 2^n$. Thus, although $(\mathcal{A}_{inc}^k)^\mathsf{T}$ is a $2^n \times 2^n$ operator, when applied to a register of n qubits, it will only have effect on the first k states, leaving the rest intact.

The same can be done for the transpose decrement operator, i.e. we define $(\mathcal{A}_{dec}^k)^\mathsf{T} = \mathcal{T}_{k-2} \cdots \mathcal{T}_1 \mathcal{T}_0$. Explicitly

$$(\mathcal{A}_{dec}^k)^\mathsf{T} = (\mathcal{A}_{dec}')^\mathsf{T} \oplus I_{2^n-k} = \begin{pmatrix} (\mathcal{A}_{dec}')^\mathsf{T} & 0 \\ 0 & I_{2^n-k} \end{pmatrix} \tag{31}$$

Figure 5 (a) Displays the sequence to create gates $\mathcal{T}_i$ whose index follow the relation $i = 2j$. (b) Displays the sequence to create gates $\mathcal{T}_i$ whose index follow the relation $i = 4j + 1$. Both sequences are given by all the combinations of multi-controlled not or swap gates

The circuit implementation of all adjacent transposition operators $\mathcal{T}_i$ is not trivial, given that the quantum gates corresponding to operators with consecutive indices do not follow the same structure. We present three quantum gate structures that generate transposition operators $\mathcal{T}_i$ for which i belongs to one of the sequences $(2j)_{j=0}^{j=2^{n-1}}$, $(4j+1)_{j=0}^{j=2^{n-2}}$, and $(4j+3)_{j=0}^{j=2^{n-2}-1}$. The union of all the sequences generate all natural numbers from 0 to $2^n - 2$, thus the combination of transposition operators corresponding to different sequences generate every $\mathcal{T}_i$ that follows Eq. (25). Figures 5(a) and 5(b) show the general pattern of transposition operators associated to the first and second sequences, respectively. Each gate uses all the qubits of the position register for its construction. The transposition operators associated to the third sequence must follow a more complex process for their construction that will be described next.

Firstly, the transposition gates associated to $(4j+3)_{j=0}^{j=2^{n-2}-1}$ will only be used in the case where the position register consists of three or more qubits, otherwise the gates associated to the first two sequences alone will suffice, given that for n qubits we can have a maximum of $2^n - 1$ transposition gates. Now, to construct transposition gates for position registers of more than two qubits, we will make use of multi-control SWAP gate whose controls are placed in the upper qubits, and we will denote it by $C^{r,i}(SWAP)$, where we remark that r and i refer to the number of controls and the value of the binary string formed by the controls, respectively, as described in section 3.

The $C^{r,i}(SWAP)$ gate will allow us to define a set of basis gates which will serve to generate different transposition gates $\mathcal{T}_i$ through a recurrence relation. Each basis gate, $\mathcal{B}_{\mathcal{T}}^r$, is composed of a $C^{r,0}(SWAP)$ gate at the core, surrounded to the left and to the right by CNOT gates in descending and ascending echelon-like form, respectively, as can be seen in the sequence presented in Fig. 6(a). The superindex in $\mathcal{B}_{\mathcal{T}}^r$ stands for the number of controls in the core $C^{r,0}(SWAP)$ gate. For $n \geq 3$ qubits, there will be $n - 2$ basis gates following the pattern of Fig. 6(a)

Once we have constructed all basis gates $\mathcal{B}_{\mathcal{T}}^r$ for $n \geq 3$ qubits, the set of transposition gates $\mathcal{T}_i$ associated to $(4j+3)_{j=0}^{j=2^{n-2}-1}$ can be obtained by a recursive relation. To explain how this recursive relation works, we refer to Fig. 6(b) to explain the first instances of the relation and then provide a general rule.

The upper circuit in Fig. 6(b) takes place in a 3-qubit register, which means there is place for one transposition gate, thus the transposition gate $\mathcal{T}_3$ coincides with the basis gate $\mathcal{B}_{\mathcal{T}}^1$. The middle circuit in this figure

(a)

(b)

Figure 6 (a) Sequence of basis gates $\mathcal{B}_{\mathcal{T}}^r$ to generate the set of transposition gates $\mathcal{T}_i$ associated to $(4j+3)_{j=0}^{j=2^{n-2}-1}$ as the number of qubits increeaces. (b) Sequence of circuits that contain all the transposition gates $\mathcal{T}_i$ associated to $(4j+3)_{j=0}^{j=2^{n-2}-1}$ for $n=3,4,5$. This sequence allows us to see that as we increase the number of qubits, the transposition gates for n qubits can be constructed using the set of gates for $n-1$ qubits, i.e. if $\mathcal{S}_{\mathcal{T}}^n$ is the set of transposition gates associated to $(4j+3)_{j=0}^{j=2^{n-2}-1}$ for n qubits, then the recurrence relation $\mathcal{S}_{\mathcal{T}}^n = C_{1,0}(\mathcal{S}_{\mathcal{T}}^{n-1})\mathcal{B}_{\mathcal{T}}^{n-2}C_{1,1}(\mathcal{S}_{\mathcal{T}}^{n-1})$ complies

takes place in a 4-qubit register, thus there is place for 3 transposition gates. The center gate, $\mathcal{T}_7$, coincides with the basis gate $\mathcal{B}_{\mathcal{T}}^2$, and the gates to left and right of $\mathcal{T}_7$ are built by taking the 3-qubit $\mathcal{T}_3$ gate and controlling the core $C^{1,0}(SWAP)$ gate with a bottom white and a black control, respectively. For the lower circuit, we follow a similar rationale. We place the basis gate $\mathcal{B}_{\mathcal{T}}^3$ at the center of the sequence, then take the sequence of 4-qubit $\mathcal{T}_i$ gates built in the previous circuit and place them to the left and right of $\mathcal{B}_{\mathcal{T}}^3$ but now with an extra bottom white and black control, respectively, to the core $C^{r,i}(SWAP)$ gates associated to 3-qubit $\mathcal{T}_i$ gates. In general, the sequence of transposition gates associated to $(4j+3)_{j=0}^{j=2^{n-2}-1}$ for a register of n qubits, is obtained by taking the $\mathcal{B}_{\mathcal{T}}^{n-2}$ basis gate and use it as the center gate of the sequence, then taking the sequence of $\mathcal{T}_i$ gates for a register of $n-1$ qubits, which we will call $\mathcal{S}_{\mathcal{T}}^{n-1}$ for simplicity, and finally adding a controlled version of $\mathcal{S}_{\mathcal{T}}^{n-1}$ to the left and right of $\mathcal{B}_{\mathcal{T}}^{n-2}$, using bottom white and black controls, respectively. In short notation, $\mathcal{S}_{\mathcal{T}}^n = C_{1,0}(\mathcal{S}_{\mathcal{T}}^{n-1})\mathcal{B}_{\mathcal{T}}^{n-2}C_{1,1}(\mathcal{S}_{\mathcal{T}}^{n-1})$. The index of each n-qubit $\mathcal{T}_i$ gate is assigned following $(4j+3)_{j=0}^{j=2^{n-2}-1}$ from left to right.

In order to apply the generalization of the increment and decrement operators to k qubits to a DTQW, we will proceed similarly as in the case of a 2^n-cycle graph. That is, we build controlled gates out of $(\mathcal{A}_{inc}^k)^\intercal$ and $(\mathcal{A}_{dec}^k)^\intercal$, so that we compute the operators presented in Eqs. (32) and (33):

$$C_{1,0}((A_{inc}^k)^\intercal) = \begin{pmatrix} (\mathcal{A}_{inc}')^\intercal & 0 & 0 & 0 \\ 0 & I_{2^n-k} & 0 & 0 \\ 0 & 0 & I_k & 0 \\ 0 & 0 & 0 & I_{2^n-k} \end{pmatrix} \tag{32}$$

$$C_{1,1}((A_{dec}^k)^\intercal) = \begin{pmatrix} I_k & 0 & 0 & 0 \\ 0 & I_{2^n-k} & 0 & 0 \\ 0 & 0 & (\mathcal{A}_{dec}^k)^\intercal & 0 \\ 0 & 0 & 0 & I_{2^n-k,} \end{pmatrix} \tag{33}$$

which act on quantum states $|\psi\rangle$ with coin state $|0\rangle$ and $|1\rangle$, respectively. Thus the shift operator of a DTQW on a k-cycle graph is given by $S = C_{1,0}((\mathcal{A}_{inc}^k)^\intercal)C_{1,1}((\mathcal{A}_{dec}^k)^\intercal)$, i.e.

$$S = \begin{pmatrix} (\mathcal{A}_{inc}')^\intercal & 0 & 0 & 0 \\ 0 & I_{2^n-k} & 0 & 0 \\ 0 & 0 & (\mathcal{A}_{dec}')^\intercal & 0 \\ 0 & 0 & 0 & I_{2^n-k} \end{pmatrix} \tag{34}$$

As an example, consider the 5-cycle graph (see. Fig. 7(a)), with adjacency matrix which can be decomposed as a sum of 5×5 increment and decrement operators.

$$(\mathcal{A}_{5c})^\intercal = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{35}$$

In order to be implementable as a quantum circuit, an operator must have size $2^n \times 2^n$, thus we augment both 5×5 increment and decrement operators according to equations (30) and (31), and we obtain

$$(\mathcal{A}_{inc}^5)^\intercal = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{36}$$

$$(\mathcal{A}_{dec}^5)^{\intercal} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{37}$$

This procedure induces three extra vertices to the graph, as shown in Fig. 7(b).

Next, according to Eq. (34) we write the shift operator for this system as

$$S_{5C} = \begin{pmatrix} (\mathcal{A}_{inc}^5)^{\intercal} & 0 \\ 0 & (\mathcal{A}_{dec}^5)^{\intercal} \end{pmatrix} \tag{38}$$

The quantum circuit associated to this operator is shown in Fig. 7(c).

(a)   (b)

(c)

Figure 7 (a) 5-cycle graph. (b) 8-vertex graph which contains a 5-cycle graph as a subgraph and three isolated nodes. (c) Circuit implementation of the evolution operator of a DTQW on (b), according to Eq. (34). To perform a DTQW on the 5-cycle using circuit (c) initialize the position state of the walker on any state $|v\rangle$ for which $v = 0, \ldots, 4$

The position register must be initialized in a composite state whose bitstring representation corresponds to the label (an integer number) of any vertex of the subgraph that conforms a 5-cycle graph. If the composite state is initialized in any of the other vertices the walker will remain on that vertex every time the evolution

operator is applied. The coin operator can be initialized either with the state $|0\rangle$, $|1\rangle$ or a superposition of both states.

The action of the shift operator S_{5C} on the quantum states of the position register associated to the labels of the vertices is presented in Table(1). From this example, notice that the minimum number of qubits to implement a DTQW on a k-cycle graph in the circuit model is $n+1$, where n is the minimum value such that $2^n > k$.

Table 1 Action of the shift operator for a DTQW on a 5-cycle (Fig. 7) on the states $|0\rangle|q_1q_2q_3\rangle$ and $|1\rangle|q_1q_2q_3\rangle$. The first and second pair of columns represent the transition of position states with coin states $|0\rangle$ and $|1\rangle$, respectively

coin $|0\rangle$		coin $|1\rangle$	
base 2	base 10	base 2	base 10
$|000\rangle \rightarrow |001\rangle$	$|0\rangle \rightarrow |1\rangle$	$|000\rangle \rightarrow |100\rangle$	$|0\rangle \rightarrow |4\rangle$
$|001\rangle \rightarrow |010\rangle$	$|1\rangle \rightarrow |2\rangle$	$|100\rangle \rightarrow |011\rangle$	$|4\rangle \rightarrow |3\rangle$
$|010\rangle \rightarrow |011\rangle$	$|2\rangle \rightarrow |3\rangle$	$|011\rangle \rightarrow |010\rangle$	$|3\rangle \rightarrow |2\rangle$
$|011\rangle \rightarrow |100\rangle$	$|3\rangle \rightarrow |4\rangle$	$|010\rangle \rightarrow |001\rangle$	$|2\rangle \rightarrow |1\rangle$
$|100\rangle \rightarrow |000\rangle$	$|4\rangle \rightarrow |0\rangle$	$|001\rangle \rightarrow |000\rangle$	$|1\rangle \rightarrow |0\rangle$
$|101\rangle \leftrightarrow |101\rangle$	$|5\rangle \leftrightarrow |5\rangle$	$|101\rangle \leftrightarrow |101\rangle$	$|5\rangle \leftrightarrow |5\rangle$
$|110\rangle \leftrightarrow |110\rangle$	$|6\rangle \leftrightarrow |6\rangle$	$|110\rangle \leftrightarrow |110\rangle$	$|6\rangle \leftrightarrow |6\rangle$
$|111\rangle \leftrightarrow |111\rangle$	$|7\rangle \leftrightarrow |7\rangle$	$|111\rangle \leftrightarrow |111\rangle$	$|7\rangle \leftrightarrow |7\rangle$

Notice that the set of adjacent transposition operators $\mathcal{T}_i$ has applications way beyond the cycle graph, given that it is a well known results in group theory that a general permutation is obtained by the product of transpositions [41] – where a transposition is the permutation of two elements leaving the rest intact – and any transposition can be obtained by the product of adjacent transpositions [42]. Thus, as we have provided the circuit form of all adjacent transpositions of rows in $2^n \times 2^n$ matrices, then any $2^n \times 2^n$ permutation matrix (or shunt in this work) can be generated by the set of operators $\mathcal{T}_i$. Combining this result with Eq. (5), we can generate the quantum circuit of associated to any transposed adjacency matrix.

4.2 DTQW on the line

Consider a graph constituted by n vertices each one connected only to the previous and next ones, except for the vertices at the endings, which are only connected to the previous node. We refer to this mathematical object as the n-node line graph, or simply the n-line. Fig. 8 displays a $7 - line$ as an example. The adjacency matrix of an n-line is an $n \times n$ matrix presented in Eq. (39).

Figure 8 7-line graph

Figure 9 Steps to transform a cycle graph into a line graph according to Table 2. (a) 8-cycle graph with labels in ascending order. (b) 8-cycle graph with transformed labels to binary. (c) 8-cycle graph with labels interpreted as two's complement (d) Rearranged 8-cycle graph into a straight line

$$\mathcal{A}_{line} = \begin{pmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 0 \end{pmatrix} \tag{39}$$

Notice that its form is quite similar to the adjacency matrix of a cycle graph, with the only difference that it only has elements in the diagonals above and below the main one, and not in any other entry. In order to implement a DTQW on an n-line, we can take advantage of this fact, and of what we derived in the previous subsection.

We consider a 2^n-cycle graph with nodes labeled with consecutive integers in ascending order starting from zero. Next we map the integer representation of each label to binary. After that, we consider the obtained bitstrings to be in two's complement notation[1], so that if we go back to integer representation, the upper half of the labels now take the values from -1 to -2^n. The proposed transformation for the numbers 0 to 8 is shown in Table 2. Finally, we rearrange the nodes of the graph, in ascending order in a straight line, and consider only the subgraph formed by the vertices with labels in the interval $[-n', n']$, where $n' = 2^{n-1} - 1$. Notice that n' is always odd, thus there is always a central node in the line graphs studied in this work. An example of the process of transformation from an 8-cycle graph to a 7-line is shown in Fig. 9.

In view of the transformation previously proposed, a DTQW on a line can be understood as a DTQW on a 2^n-cycle graph whose labels have been mapped using two's complement and which holds the restriction that

---

[1] Two's complement is a way to represent both positive and negative numbers in bitstring notation. Strings whose leftmost bit $b_n = 0$, correspond to positive numbers. When $b_n = 1$, we have a negative number. The rest of the bits, $N = b_{n-1} \dots b_2 b_1$, represent the value of the number and b_n only indicates the symbol. When $b_n = 0$, the right substring, N, has the same relation with decimal numbers as in binary notation. However, when $b_n = 1$, the relation of N with decimal numbers is *inverted* with respect to binary notation. E.g., for a 3-bit string, let $b_n = 1$, then for N equals 00, 01, 10 and 11 the associated numbers are 4, 3, 2 and 1, respectively, in such a way that 100, 101, 110 and 111 are associated to -4, -3, -2 and -1, respectively.

Table 2 Transformation of labels of an $8 - cycle$ graph. Visual sequence given in Fig. 9

Decimal		Binary		Two's complement		Decimal
0	$\rightarrow$	000	$\rightarrow$	000	$\rightarrow$	0
1	$\rightarrow$	001	$\rightarrow$	001	$\rightarrow$	1
2	$\rightarrow$	010	$\rightarrow$	010	$\rightarrow$	2
3	$\rightarrow$	011	$\rightarrow$	011	$\rightarrow$	3
4	$\rightarrow$	100	$\rightarrow$	100	$\rightarrow$	-4
5	$\rightarrow$	101	$\rightarrow$	101	$\rightarrow$	-3
6	$\rightarrow$	110	$\rightarrow$	110	$\rightarrow$	-2
7	$\rightarrow$	111	$\rightarrow$	111	$\rightarrow$	-1

the evolution operator can be applied a maximum number of times $k \leq n' - n_0$, where $n' = 2^{n-1} - 1$ and n_0 is the initial node of the walk. If it is applied a greater number of times than k, the walker will move in a cycle.

This implies that instead of finding an additive decomposition for the adjacency matrix shown in Eq. (39) and getting a new shift operator out of it, we can consider the shift operator already presented in Eq. (16), and for the circuit implementation, we consider the circuit presented in Fig. 4. The minimum number of qubits needed to implement a walk on a line of k steps is obtained by solving Eq. (40) for n and then finding $\lceil n \rceil + 1$, where the extra unit is due to the coin register.

$$k = 2^{n-1} - n_0 - 1 \tag{40}$$

4.3 DTQW on the hypercube

The hypercube graph, $Q_n(V, E)$, is an n-regular graph, built from the set of vertices and edges of the n-dimensional hypercube, also called $n-$cube. The graphical representation of Q_n for n = 1, 2, 3, 4 is shown in Fig. (10).

1-cube

2-cube

3-cube

4-cube

Figure 10 Visual representation of the n-cube for $n = 1, 2, 3, 4$

In order to construct the general graph Q_n, let S_n be the set of all n-bitstings, and let $V = \{v_i : v_i \in S_n\}$

Figure 11 (a) Quantum gate representation of the matrices that conform the shunt decomposition of the adjacency matrix of a 4-cube. Quantum circuit associated to the shift operator of a DTQW on a (b) 4-cube and (c) 3-cube with self-loops

be the set of bitstrings that are labels for the vertices of the n-cube. Two such bitstrings v_i and v_j correspond to adjacent vertices if and only if they differ only in one bit, that is, their Hamming distance is equal to one. Adjacent vertices are connected by an edge $E_i \in E$, such that $E = \{\{v_i, v_j\}|v_i, v_j \in S_n;\ v_i \neq v_j\}$ is the set of edges of the graph. Each vertex v_j is connected to n other vertices by n edges E_i. Notice according to section 2 a DTQW takes place on a directed graph, thus, to comply with this requirement, we can replace edges E_i by a set of parallel arcs pointing in opposite directions.

The usual definition of the shift operator for a DTQW on an n-cube is given in Eq. (41).

$$S = \sum_{i=0}^{m-1} \sum_{j=0}^{2^m-1} |c_i, v_j \oplus e_i\rangle\langle c_i, v_j| \tag{41}$$

or

$$S = \sum_{i=0}^{m-1} \sum_{j=0}^{2^m-1} |c_i\rangle\langle c_i| \otimes |v_j \oplus e_i\rangle\langle v_j|$$

where e_i is the n-dimensional bitstrings with all bits zero except for the ith bit, which is 1, c_i and v_j are m and n-dimensional bitstring, respectively, and the symbol $\oplus$ represents the binary sum or bit-wise xor. Notice that in this system the dimension of the coin spacace, m, is related to the dimension of the position space, n, as $n = 2^m$.

The factor $|c_i\rangle\langle c_i|$ restricts S to be a block diagonal matrix, and thus the block diagonal elements, $\sum_{j=0}^{2^m-1} |v_j \oplus e_i\rangle\langle v_j|$ (for fixed i), are the shunt decomposition of the adjacency matrix of a 2^m-cube.

The adjacency matrix of an n-cube can be obtained following the next recursive pattern presented in [43]

$$\mathcal{A}_{n+1} = \begin{pmatrix} \mathcal{A}_n & I_n \\ I_n & \mathcal{A}_n \end{pmatrix};\ \ n > 0;\ \ \mathcal{A}_0 = (0) \tag{42}$$

For any dimension $n = 2^m$, $\mathcal{A}_n$ can be decomposed as the sum of unitary matrices, $\mathcal{A}_{2^m} = \sum_{i=0}^{n-1} \mathcal{P}_i^{\mathsf{T}}$, where each $\mathcal{P}_i^{\mathsf{T}}$ is written in terms of the Pauli matrix σ_x and the 2×2 identity only

$$\mathcal{P}_i^{\mathsf{T}} = I_2^{\otimes i} \otimes \sigma_x \otimes I_2^{\otimes(n-i-1)}; \quad i = 0, 2, \ldots, n-1 \tag{43}$$

where we define $I_2^{\otimes 0} = (1)$.

For the case in which v_j is the bitstring whose binary value corresponds to decimal value of the index j, then the following equation holds for some bitstring e_i

$$\mathcal{P}_i^{\mathsf{T}} = I_2^{\otimes i} \otimes \sigma_x \otimes I_2^{\otimes(n-i-1)} = \sum_{j=0}^{2^m-1} |v_j \oplus e_i\rangle\langle v_j| \tag{44}$$

This can be corroborated by explicit matrix analysis of both expressions.

Therefore, according to Eq. (7), the matrix representation of the shift operator is given by

$$S = \bigoplus_{i=0}^{n-1} I_2^{\otimes i} \otimes \sigma_x \otimes I_2^{\otimes(n-i-1)} \tag{45}$$

In order to get the quantum circuit associated with S, we use the fact that a quantum gate can be built for every unitary matrix $\mathcal{P}_i^{\mathsf{T}}$ by setting a register of n qubits, and placing a σ_x gate in the ith qubit leaving the rest of the register empty, as in the example shown in Fig. 11(a). Next, we control all gates $\mathcal{P}_i^{\mathsf{T}}$ using all the qubits of the coin register with a different pattern of black and white controls for each $\mathcal{P}_i^{\mathsf{T}}$, in order to obtain the quantum gates $C_{m,j}(\mathcal{P}_i^{\mathsf{T}})$, as explained in section 3. The combination of gates $C_{m,j}(\mathcal{P}_i^{\mathsf{T}})$ is equal to the shift operator of the system, and form a pattern as in the example shown in Fig. 11(b).

Furthermore, we can use the circuit of a 2^m-cube to obtain a circuit for any cube of dimension lower than 2^m, with the restriction that the number of dimension we decrease from the original cube, will be the number of self-loops added to all the vertices of the new structure. To do this, simply remove the first k CNOTs with the largest range; the resulting circuit will be that of a $(2^m - k)$-cube with k self-loops. For instance, if we want to obtain a circuit for a 3-cube with self-loops, then we modify the circuit of a 4-cube to obtain the circuit shown in Fig. 11(c).

The general pattern of the quantum circuit representation of S was proposed by Douglas and Wang in [44], being a key element to our paper to construct an explicit link between the shunt decomposition method and the quantum circuit representation of S. In order to simplify the circuit, notice that negated black controls are represented by white controls, and we negate a black control by placing σ_x gates at both sides of it. Current quantum computers do not allow the direct implementation of white controls, thus if we are to build a quantum circuit that consists of a sequence of multi-control gates, we want that when we replace all white controls with black controls sorrounded by σ_x gates to the left and to the right, as many σ_x gates as possible cancel out. This can be achieved by initially designing the circuit in such a way that as many white controls are next to each other in the sequence of multi-control gates. For this purpose, we propose the use of gray code, which a bitstring representation of numbers such that for two consecutive numbers in decimal notation the corresponding bitstrings differ in only one bit. That is, this sequence maximizes the resemblance of consecutive bitstrings, and if we associate 0's and 1's with white and black controls, respectively, when using this sequence to generate all $C_{m,j}(\mathcal{P}_i^{\mathsf{T}})$ gates that constitute the shift operator of a 2^n-cube, we can cancel out a wide number of redundant σ_x gates. Although, for gray code to be efficient, we must start the sequence with the string of all ones, or to put it in another way, the leftmost (or rightmost, it is indistinct) gate must be controlled with only black controls, and then single-bit-flipping sequence can be used to control the following gates. This is exemplified in Fig. 12.

(a)   (b)

(c)

Figure 12 (a) Circuit implementation of the shift operator of a DTQW on a 8-cube proposed in [44], where the controls follow gray code sequence. (b) Current quantum computers only admit black controls, thus white controls must replace by black controls surrounded by two σ_x gates. (c) Redundant CNOTs are cancelled out, which could represent an advantage if the circuit is to be implemented in noisy quantum computers

4.4   DTQW on the 2^m-complete graph with self-loops

An n-complete graph with self-loops, $\mathcal{K}_n$ is a set of n vertices such that the vertex v_i is connected with all other vertices v_j, including itself. An example of this type of graph is given in Fig. (13).

Figure 13 Graph $\mathcal{K}_8$

The adjacency matrix of an $n-$node complete graph with self-loops at every vertex is a matrix whose entries

are all ones (Eq. (46)). Here we propose a method to decompose the adjacency matrix of the $\mathcal{K}_{2^m}$ graph, where m is the dimension of the coin register.

$$
\mathcal{A}_{\mathcal{K}_n} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 & 1 & 1 \\ 1 & 1 & 1 & \dots & 1 & 1 & 1 \\ 1 & 1 & 1 & \dots & 1 & 1 & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & \dots & 1 & 1 & 1 \\ 1 & 1 & 1 & \dots & 1 & 1 & 1 \\ 1 & 1 & 1 & \dots & 1 & 1 & 1 \end{pmatrix} \tag{46}
$$

The decomposition can be done by means of the tensor products of σ_x and I_2 operators. For this purpose, we appeal again to binary code: consider the σ_x operator to be a "1", and I_2 operator to be a "0". Now consider the tensor product of a number n_1 of operators σ_x and a number n_2 of operators I_2 ($n_1 + n_2 = m$), in such a way that their sequence is interpreted as binary code. Every sequence of tensor products defines a matrix $\mathcal{P}_i^\mathsf{T}$, such that the index i corresponds to the number the sequence represents. The sequence of operators for the graph $\mathcal{K}_8$ is shown in the set of Eqs. (47).

$$
\mathcal{P}_0^\mathsf{T} = I_2 \otimes I_2 \otimes I_2 \tag{47a}
$$
$$
\mathcal{P}_1^\mathsf{T} = I_2 \otimes I_2 \otimes \sigma_x \tag{47b}
$$
$$
\mathcal{P}_2^\mathsf{T} = I_2 \otimes \sigma_x \otimes I_2 \tag{47c}
$$
$$
\mathcal{P}_3^\mathsf{T} = I_2 \otimes \sigma_x \otimes \sigma_x \tag{47d}
$$
$$
\mathcal{P}_4^\mathsf{T} = \sigma_x \otimes I_2 \otimes I_2 \tag{47e}
$$
$$
\mathcal{P}_5^\mathsf{T} = \sigma_x \otimes I_2 \otimes \sigma_x \tag{47f}
$$
$$
\mathcal{P}_6^\mathsf{T} = \sigma_x \otimes \sigma_x \otimes I_2 \tag{47g}
$$
$$
\mathcal{P}_7^\mathsf{T} = \sigma_x \otimes \sigma_x \otimes \sigma_x \tag{47h}
$$

Then

$$
\mathcal{A}_{\mathcal{K}_8} = \sum_{i=0}^{7} \mathcal{P}_i^\mathsf{T} \tag{48}
$$

Fig. 14(a) shows the sequence of quantum gates that corresponds to every operator $\mathcal{P}_i^\mathsf{T}$. Similar to the previous sections, we have to control these gates and then place them in sequence in order to obtain a quantum circuit for a shift operator that performs a DTQW on the graph $\mathcal{K}_8$. In this case, the quantum circuit can be simplified if the index j of the controlled gate, $C_{m,j}(\mathcal{P}_i^\mathsf{T})$, is equal to the index of the operator $\mathcal{P}_i^\mathsf{T}$. This means that every sequence of σ_x and I_2 gates will be controlled by a sequence of white and black dots that represent the same number in binary, as shown in Fig. 14(b).

The addition of the identity gate to the circuits is just for illustrative purposes, it can be neglected. Although, even if we do so, some of the gates $C_{m,j}(\mathcal{P}_i^\mathsf{T})$ are still multi-target multi-control. This type of gate can be decomposed into a set of single-target multi-control gates [45], as shown in the example of Fig. 15(a). Then, multi-controlled gates that have the same target qubit and whose sequence of black and white dots differs only in one element can be simplified as shown in the example of Fig. 15(b). This property has been derived in different works [46, 47, 48] using different methods.

Figure 14 (a) Gate sequence corresponding to the permutation matrices presented in the set of Eqs. (47). (b) Gate sequence corresponding to the shift operator of a DTQW on $\mathcal{K}_8$ (Fig. (13)). Notice that the sequence of identity and CNOT gates is the same as the sequence of black and white dots for each gate

When both properties are applied to the whole network of multi-control CNOT gates shown in Fig. (15), everything can be reduced to only three gates, as shown in Fig. 16(a). The sequence of controlled gates can be continued (see Fig. 16(b)) to obtain a shift operator for a DTQW on a complete graph with 2^m vertices. Notice that we can use the same procedure replacing the gate σ_x by an arbitrary multi-qubit U gate to obtain efficient implementation of operators shift for different graphs.

In terms of outer products, the shift operator for this system can be defined in a similar way than the usual definition of the shift operator for a DTQW on the hypercube Eq. (41), with the only difference that in this case, the direction bitstrings e_i are all bitstrings of size m, in such a way that we associate one bitstring to each edge connected to the vertex where the quantum walker stands on. This provides 2^m directions for the walker to move at each vertex. The analysis of the matrix form of S resulting from this idea conducted us to find the decomposition presented in the set of Eqs. (47), which can be extended to any dimension 2^m following the same procedure.

Douglas and Wang proposed in [44] a circuit which can be generalized to perform a DTQW on a graph $\mathcal{K}_{2^n}$, using n SWAP gates, arranged as presented in Fig. 17. The circuit was the result of mapping a different shift operator to perform a DTQW on the same topology. The shift operator used has the following action on the state of a walker $S|c_i, v_j\rangle = |v_j, c_i\rangle$, thus it seems logical to use SWAP operators for its mapping to a quantum

Figure 15 Examples of the (a) Decomposition of a multi-control multi-target gate. (b) Synthesis of two multi-control single-target gates which differ by only one control color for the same control qubit

Figure 16 (a) Synthesized circuit after applying the decompositions shown in Fig. 15 to the shift operator of a DTQW on $\mathcal{K}_8$ (Fig. 14(b)). (b) Shift operator composed of n CNOT gates to perform a DTQW on $\mathcal{K}_{2^n}$

Figure 17 (a) Sequence of n SWAP gates corresponding to the shift operator of a DTQW on $\mathcal{K}_{2^n}$, proposed in [44]. (b) Decomposition of a SWAP gate into CNOT gates

circuit. Each SWAP gate can be implemented using three CNOT gates, therefore the shift operator proposed by Douglas and Wang can be implemented using $3n$ CNOT gates, while the circuit we propose for a DTQW on the same topology, needs only n CNOT gates, as can be seen from Fig. 16(b), which proves it to be more efficiently implementable in terms of CNOT gates.

5   Simulation and Experimental Implementation of DTQWs

In this section we implement specific cases of the quantum circuits associated to the shift operators of the topologies studied in the previous sections, using both IBM's quantum computers publicly available through IBM Quantum [49] and Qiskit Aer simulator. The topologies that we will focus on in this section are the graph $\mathcal{K}_4$, the 8-line, the 3-cube with self-loops and the 5-cycle graph, presented in this order in the coming subsections. For the first two graphs experimental results will be presented, while for the last two graphs we limit ourselves to present Qiskit simulations. The quantum processors we used to perform the experimental computations are: ibmq_manila, ibmq_bogota, ibmq_quito, and ibmq_lima, with quantum volume (QV) 32, 32, 16, and 8, respectively. All of these quantum processors have 5 qubits, connected in T-like (quito and lima) and line-like (manila and bogota) topologies as shown in Fig. (18). Due to the low amount of available qubits we restrict our DTQWs to take place in low-dimensional cases of the graphs studied. We implemented the quantum circuits directly in the Quantum Composer.

We utilized the statistical distance between distributions

$$\ell_1(P, Q) = \frac{1}{2} \sum_{\forall i} |P(i) - Q(i)| \tag{49}$$

as a metric to evaluate the performance of each quantum circuit compared to the theoretical distributions, where the shorter the distance, the higher the resemblance of an experimental distribution to the theoretical one. To have a visual representation of the performance of the implemented DTQWs, we present a summary

Figure 18 Connectivity map of the quantum processors used in this work to perform DTQWs. (a) Processor type: Falcon r5.11L. ibmq_manila and ibmq_bogota follow this connectivity. (b) Processor type: Falcon r4T. ibmq_quito and ibmq_lima follow this connectivity. See [50] for more information on the processors

grid of the obtained probability distributions in the following sections, that include the theoretical distributions and the experimental distributions with best metrics for each model and each step, regardless of the quantum processor used. In all cases, i.e. simulated, experimental and theoretical DTQWs, the initial state of the walker was the state $\psi = |0\rangle_C \otimes |0\rangle_P$, for all the models. The *theoretical* distributions were calculated analytically, using the corresponding shift and coin operators of each graph, obtained as described in the previous sections.

A repository with all the circuits resulting from this work in Qiskit format, is available in [51].

5.1 DTQW on the graph $\mathcal{K}_4$

To perform a DTQW on a graph $\mathcal{K}_{2^m}$ we need $2m$ qubits, thus, with the available 5-qubit quantum computers we can only run circuits for the graphs $\mathcal{K}_2$ and $\mathcal{K}_4$, although we will focus on the latter. We compare the performance of both models for the graph $\mathcal{K}_4$ shown in Figs. 16 and 17. The metrics of the experimental distributions are shown in Tables 3 and 4, where the former corresponds to the model introduced in this paper ($CNOT$ model), while the latter corresponds to the model introduced by Douglas and Wang ($SWAP$ model) [44]. The depth and size of the original single-step circuits is 2 and 4, respectively, for both models. However, to be able to run in IBM's quantum computers, these circuits must undergo a transpilation process to adapt to the topology (see Fig. 18) of the quantum computers and be optimized for noise reduction. Furthermore, this process involves a change of basis to the set $CNOT$, $\sqrt{x}$ and R_z, which is the native set of gates in IBM's quantum computers. Thus, in the case of the one-step transpiled circuits, the size of the circuit with best metric is 11 and 14, for the $CNOT$ model, while the depth and size of the circuit with best metric for the $SWAP$ model is 15 and 18, respectively.

Table 3 ℓ_1 distance between experimental and theoretical distributions for the graph $\mathcal{K}_4$ using the $CNOT$ model. ibmq_quito has the best overall performance.

Device/Steps	1 step	2 steps	3 steps	Avg
ibmq_manila	0.057	0.114	**0.297**	0.156
ibmq_bogota	0.114	0.091	0.352	0.185
ibmq_quito	0.048	**0.024**	0.319	**0.130**
ibmq_lima	**0.040**	0.049	0.384	0.158

Although the topology is the same for both models, the dynamics of the walker changes depending on the shift operator. The most visible case arises when we compare the third step of both models, while for the

Table 4 ℓ_1 distance between experimental and theoretical distributions for the graph $\mathcal{K}_4$ using the $SWAP$ model. ibmq_manila has the best overall performance.

Device/Steps	1 step	2 steps	3 steps	Avg
ibmq_manila	0.051	0.130	**0.182**	**0.121**
ibmq_bogota	0.130	0.144	0.227	0.167
ibmq_quito	0.112	0.101	0.341	0.185
ibmq_lima	**0.048**	**0.054**	0.355	0.152

$CNOT$ model we obtain the state $|0011\rangle$ after measurement, for the $SWAP$ model we obtain the state $|0000\rangle$ (see Fig. 19). Additionally, the the walker needs 8 applications of the evolution operator to complete a cycle on the graph using the former model, while with the latter the walker needs only 4 applications. We define the cycle of a DTQW as the number of steps T such that $U^T = I$, and as a consequence the quantum state of a walker $|\psi(t)\rangle$ returns to its initial state. Further simulations using Qiskit, show that the number of steps to complete a cycle in the graph $\mathcal{K}_{2^m}$ remains constant for $m = 1, 2, 4, 8, 16$, which suggests that these might be constant values for any dimension 2^m.

Figure 19 Comparison between theoretical and best experimental probability distributions of a DTQW on a $\mathcal{K}_4$ graph of 1, 2 and 3 steps. (a) Distribution using the $CNOT$ model and (b) Distribution using the $SWAP$ model. The processor used to obtain each distribution is indicated on top of them. The metrics of each distribution can be found in Tables 3 and 4. These distributions present the best metrics of all the models studied in this work, which is attributed to the simplicity of the circuit in both cases and to the fact that one qubit less was needed to perform the DTQWs in comparison to the other models. Notice that the measured state after a DTQW of three steps is different in both models. Although we perform a DTQW on the same topology, the fact that the $CNOT$ and $SWAP$ models have diagonal and non-diagonal shift operators, respectively, changes the dynamics of the walker

5.2 DTQW on the 8-line

In the case of the 8-line, two models were considered, for which four qubits were needed. The first model is presented in Fig. 4(a) and we label it LRA (Long-Range Arrangement); the second model is presented in Fig. 4(b) and we label it NNA (Nearest-Neighbors Arrangement as proposed in [38]). Comparing these models we want to study if the arrangement of CNOT gates to the sides of the increment operator has an effect on the

circuit performance. Specially because NNA is optimal to run in a line-like quantum computer (see Fig. 18(a)), as it adjusts perfectly to this topology, while LRA must be transpiled into a completely different sequence of CNOT gates, which increases substantially the total number of gates. This fact can be illustrated from the example presented in Fig. 20. Regarding T-like configuration, in principle there is no preferred model.

The ℓ_1 distances between experimental and theoretical distributions are summarized in Tables 5 and 6, where the first table corresponds to the LRA model and the second one to the NNA model.

From these tables we can see that the performance of LRA model is better, which we attribute to the depth and size of the original and transpiled circuits. The depth and size of the original one-step circuit for the LRA model are 10 and 10, while the depth and size for its transpiled version with best metric are 35 and 44, respectively. In the case of the original one-step NNA model, the depth and size are 13 and 14, while the depth and size of its transpiled version with best metric are 43 and 52, respectively. In Fig. 21 we present the theoretical and experimental distributions for the first three steps on this topology. Notice that regardless of the lower amount of CNOT gates in the NNA model when using a line-like quantum processor (manila and bogota) we still obtain worse results, which suggests that the increment operator has a worse performance in these type of quantum processors.

(a)

(b)

Figure 20 Comparison between the original and transpiled circuit implementations of gate $C_{1,1}(J)$ using (a) NNA and (b) LRA. The quantum circuits were transpiled to the connectivity map of ibmq_manila, which has a line-like form (see 18(a)). Notice from (a) that NNA remains the same after the transpilation process, which indicates it is the optimal arrangement for this type of line-like topologies

Figure 21 Comparison between theoretical and best experimental probability distributions of a DTQW on an 8-line of 1, 2 and 3 steps. The first row displays the theoretical distributions for each step, while the second and third rows display experimental distributions with the best metrics for each for each step of the LRA and NNA models, respectively. The best metrics are the values in bold in tables 5 and 6. The processor used to obtain each distribution is indicated on top of them. Notice that in all cases, the experimental distributions resemble the theoretical ones, which indicates the circuits were implemented efficiently, hence we have a good performance. This fact is also supported by the low statistical distances for these models reported in the aforementioned tables

Table 5 ℓ_1 distance between experimental and theoretical distributions for the 8-line using the NNA model. Shorter distances indicate the best experimental performance for each step. ibmq_quito has the best overall performance.

Device/Steps	1 step	2 steps	3 steps	Avg
ibmq_manila	0.548	0.534	0.655	0.579
ibmq_bogota	0.457	0.597	**0.431**	0.495
ibmq_quito	**0.402**	**0.369**	0.455	**0.409**
ibmq_lima	0.548	0.534	0.655	0.579

Table 6 ℓ_1 distance between experimental and theoretical distributions for the 8-line using the LRA model. Shorter distances indicate the best experimental performance for each step. ibmq_lima has the best overall performance.

Device/Steps	1 step	2 steps	3 steps	Avg
ibmq_manila	0.474	0.438	0.428	0.446
ibmq_bogota	0.457	0.345	0.590	0.464
ibmq_quito	0.402	0.386	**0.400**	0.396
ibmq_lima	**0.339**	**0.337**	0.456	**0.377**

Attempts to efficiently run the quantum circuit for a DTQW on a line of 16 nodes were done, although the probability distributions in this case were poor given that as we increase the dimensionality of the graph, the number and size of multi-control gates is consequently incremented too, and current NISQ computers have problems maintaining a good fidelity when running circuits with a large number of multi-control gates, as was the case when running 4-qubit instances of Grover's algorithm in [52]. This problem was also present when running the circuits for the DTQW on the cycle and hypercube graphs even for low-dimensional instances, thus, in the next sections we decided to replace experimental executions on IBM quantum computers for Qiskit simulations using the Aer simulator in order to prove the functionality of our model.

5.3   DTQW on the 5-cycle graph

In section 4.1 we presented the quantum circuit of the shift operator for a walk on a 5-cycle graph (Fig. 7(c)), based on CNOT, $C_{m,j}^{n,i}(X)$ and $C_{m,j}^{n,i}(SWAP)$ gates. Qiskit allows us to easily implement the $C_{m,j}^{n,i}(X)$ gate using the *mcx* method. To take advantage of this method, we can use the fact that the $C_{m,j}^{n,i}(SWAP)$ gate, can be decomposed into a set of two CNOT and one $C_{m,j}^{n,i}(X)$ gates, as shown in Fig. 22.

Fig. 23 shows the comparison between Qiskit-simulated and theoretical distributions, for the first, second and third steps of a DTQW on the 5-cycle, for which we obtained statistical distances of 0.029, 0.0215 and 0.011, respectively.

Figure 22 Example of the decomposition of a multi-control swap gate in terms of CNOT gates. The left-hand side circuit is a particular case of a multi-cotrol Fredkin gate, which was efficiently decomposed in [53, 54], resulting in the right-hand side circuit

Figure 23 Theoretical and Qiskit-simulated probability distributions of a DTQW of 1, 2 and 3 steps on a 5-cycle graph

5.4 DTQW on the 3-cube with self-loops

To perform a DTQW on a 3-cube with self-loops, we use the circuit proposed in Fig. 11(c) for the shift operator, while for the coin operator we use the Grover operator, given that its behavior has been widely studied for this topology in [55, 56, 16, 57].

Fig. 24 shows the comparison between Qiskit-simulated and theoretical distributions, for the first, second and third steps of a DTQW on the 3-cube, for which we obtained statistical distances of 0.021, 0.025 and 0.0195, respectively.

Figure 24 Comparison between theoretical and Qiskit-simulated probability distributions of a DTQW on a 3-cube with self-loops of 1, 2 and 3 steps. The first row displays the theoretical distributions for each step, while the second row displays Qiskit-simulated distributions.

6   Conclusion

In this paper we have introduced a general method to map evolution operators of DTQWs into quantum circuits, focusing on the case in which the shift operator is a unitary block diagonal matrix, given that this structure allows a direct circuit implementation using fully controlled quantum gates.

We have used our method to build circuits for graphs commonly presented in the literature of DTQWs, like line, cycle, hypercube and complete graphs. We have shown that these circuits – some of them previously introduced in the literature – can be derived from our general method. Furthermore, studying the block diagonal form of the shift operators, we extended number of dimensions in which a DTQW can take place in the case of the hypercube and the cycle graph. For the hypercube, we explained how to modify the circuit proposed by Douglas and Wang [44] for a 2^n-hypercube, to perform a DTQW on a hypercube of any dimension by modifying the number of self-loops in the graph. Additionally for this topology, we provided a way to reduce redundant NOT gates by forcing the pattern created by white and black controls of adjacent multi-control gates to follow two's complement sequence. In the case of the cycle graph, we provided a sequence of gates, which we called adjacent transposition operators, $\mathcal{T}_i$, that allow us the build the shift operator for a cycle graph of any size. Moreover, the adjacent transposition operators might serve as building blocks to construct the quantum circuit associated to any possible permutation matrix, however, due to the fact that all of them are composed of fully controlled gates, they are not suitable to be implemented efficiently in NISQ computers, due to fidelity issues. Finally, we also used the fact that the decomposition of multi-control quantum gates has been widely studied in order to propose efficient methods to synthesize the circuits, being the most notable case the one for the DTQW on a 2^n-complete graph, which was reduced from an exponential to a linear number of gates.

In order to prove the reliability of the shunt decomposition method, in section 5 we ran punctual cases of the quantum circuits studied section 4, using both IBM's public quantum computers and Qiskit Aer simulator. The distributions obtained from experimental executions and simulations were compared against analytical results using the ℓ_1 distance as metric. The topologies we considered were the $\mathcal{K}_4$, 8-line, 5-cycle and 3-cube

graphs. However, we were able to obtain short ℓ_1 distances only for the first two topologies, given that the quantum circuits associated to them were the simplest ones. In the case of the last two topologies, poor results were obtained due to the fact that the circuits for these topologies demand a large number of multi-control not gates, which cannot yet be efficiently run on the quantum computers used. Therefore, we decided to use Qiskit Aer simulations in these cases, obtaining short ℓ_1 distances in both of them.

Looking to determine whether performance varies from one model to another, in the case of the complete graph, we compared a circuit obtained using the shunt decomposition method ($CNOT$ model) with a circuit whose shift operator is not block diagonal ($SWAP$ model) [44]. Both circuits in original form have the same complexity, and even when transpiled, the $SWAP$ transpiled circuit had only a few more quantum gates than the $CNOT$ transpiled circuit (see [51]), which resulted in very short ℓ_1 distances for both models, as presented in tables 3 and 4. However, we did find out a difference between the models, i.e. in the third step, using the $SWAP$ model, there was 100% probability to find the walker in the state associated to node 0, while for the same step, but using the $CNOT$ model, there was 100% probability for the walker to be in the state associated to node 1 (see Fig. 19). This provides evidence, that different shift operators for the same topology might lead to different walker dynamics.

Another important aspect to mention is that ibmq_quito and ibmq_lima were the quantum computers that gave out the largest number of best metrics for specific steps in different models executed experimentally, being ibmq_quito the one that had the best average performance in 2 out of 4 models studied. We attribute this to the connectivity of these quantum computers, which is T-like (Fig. 18), rather than other factors like quantum volume which is 16 and 8 for ibmq_quito and ibmq_lima, respectively, and 32 for both bogota and manila.

Finally, we conclude that the method we developed to map block diagonal unitary operators into quantum circuits can be applied to execute DTQW whose shift operator is constructed based on the shunt decomposition of the adjacency matrix on which the DTQW takes place. Although, the method has experimental limitations in the sense that, in general, it requires a large amount of multi-control gates which induce errors in the execution, we believe it is still be relevant given that is it perfectly suitable to study quantum circuits in an analytical manner and through simulations.

Acknowledgments

References

[1] Lawler, G.F., Limic, V.: Random walk: A modern introduction (2010). `https://doi.org/10.1017/CBO9780511750854`

[2] Weiss, G.H.: Random walks and their applications: Widely used as mathematical models, random walks play an important role in several areas of physics, chemistry, and biology. American Scientist **71**(1), 65–71 (1983)

[3] Codling, E.A., Plank, M.J., Benhamou, S.: Random walk models in biology. Journal of The Royal Society Interface **5**(25), 813–834 (2008). `https://doi.org/10.1098/rsif.2008.0014`

[4] Lee, C., Jang, W.-D., Sim, J.-Y., Kim, C.-S.: Multiple random walkers and their application to image cosegmentation. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3837–3845 (2015). `https://doi.org/10.1109/CVPR.2015.7299008`

[5] Abakah, E.J., Alagidede, P., Mensah, L., Ohene-Asare, K.: Non-linear approach to random walk test in selected african countries. International Journal of Managerial Finance **14**(3), 362–376 (2018). `https://doi.org/10.1108/ijmf-10-2017-0235`

[6] Kendon, V.: Quantum walk computation. AIP Conference Proceedings **1633**(1), 177–179 (2014) https://aip.scitation.org/doi/pdf/10.1063/1.4903129. `https://doi.org/10.1063/1.4903129`

[7] Krovi, H., Magniez, F., Ozols, M., Roland, J.: Quantum walks can find a marked element on any graph. Algorithmica **74**(2), 851–907 (2015). `https://doi.org/10.1007/s00453-015-9979-8`

[8] Aharonov, D., Ambainis, A., Kempe, J., Vazirani, U.: Quantum Walks On Graphs (2002)

[9] Farhi, E., Gutmann, S.: Quantum computation and decision trees. Physical Review A **58**(2), 915–928 (1998). `https://doi.org/10.1103/physreva.58.915`

[10] Chandrashekar, C.M.: Discrete-Time Quantum Walk - Dynamics and Applications (2010)

[11] Szegedy, M.: Quantum speed-up of markov chain based algorithms. 45th Annual IEEE Symposium on Foundations of Computer Science (2004). `https://doi.org/10.1109/focs.2004.53`

[12] Portugal, R.: Staggered quantum walks on graphs. Phys. Rev. A **93**, 062335 (2016). `https://doi.org/10.1103/PhysRevA.93.062335`

[13] Venegas-Andraca, S..: Quantum walk: a comprehensive review. Quantum Information Processing **11**(5), 1015–1106 (2012)

[14] Yang, Y.-G., Pan, Q.-X., Sun, S.-J., Xu, P.: Novel image encryption based on quantum walks. Scientific Reports **5**(1) (2015). `https://doi.org/10.1038/srep07784`

[15] Vlachou, C., Rodrigues, J., Mateus, P., Paunković, N., Souto, A.: Quantum walk public-key cryptographic system. International Journal of Quantum Information **13**(07), 1550050 (2015). `https://doi.org/10.1142/s0219749915500501`

[16] Shenvi, N., Kempe, J., Whaley, K.B.: Quantum random-walk search algorithm. Physical Review A **67**(5) (2003). `https://doi.org/10.1103/physreva.67.052307`

[17] Bezerra, G.A., Lugão, P.H., Portugal, R.: Quantum-walk-based search algorithms with multiple marked vertices. Physical Review A **103**(6) (2021). `https://doi.org/10.1103/physreva.103.062202`

[18] Dernbach, S., Mohseni-Kabir, A., Pal, S., Gepner, M., Towsley, D.: Quantum walk neural networks with feature dependent coins. Applied Network Science **4**(1) (2019). `https://doi.org/10.1007/s41109-019-0188-2`

[19] de Souza, L.S., de Carvalho, J.H.A., Ferreira, T.A.E.: Quantum walk to train a classical artificial neural network. In: 2019 8th Brazilian Conference on Intelligent Systems (BRACIS), pp. 836–841 (2019). `https://doi.org/10.1109/BRACIS.2019.00149`

[20] Paparo, G.D., Martin-Delgado, M.A.: Google in a quantum network. Scientific Reports **2**(1) (2012). https://doi.org/10.1038/srep00444

[21] Chawla, P., Mangal, R., Chandrashekar, C.M.: Discrete-time quantum walk algorithm for ranking nodes on a network. Quantum Information Processing **19**(5) (2020). https://doi.org/10.1007/s11128-020-02650-4

[22] Tulsi, A.: Faster quantum-walk algorithm for the two-dimensional spatial search. Phys. Rev. A **78**, 012310 (2008). https://doi.org/10.1103/PhysRevA.78.012310

[23] Preskill, J.: Quantum computing in the nisq era and beyond. Quantum **2**, 79 (2018). https://doi.org/10.22331/q-2018-08-06-79

[24] Schreiber, A., Cassemiro, K.N., Poto ček, V., Gábris, A., Mosley, P.J., Andersson, E., Jex, I., Silberhorn, C.: Photons walking the line: A quantum walk with adjustable coin operations. Phys. Rev. Lett. **104**, 050502 (2010). https://doi.org/10.1103/PhysRevLett.104.050502

[25] Broome, M.A., Fedrizzi, A., Lanyon, B.P., Kassal, I., Aspuru-Guzik, A., White, A.G.: Discrete single-photon quantum walks with tunable decoherence. Phys. Rev. Lett. **104**, 153602 (2010). https://doi.org/10.1103/PhysRevLett.104.153602

[26] Shakeel, A.: Efficient and scalable quantum walk algorithms via the quantum fourier transform. Quantum Information Processing **19**(9) (2020). https://doi.org/10.1007/s11128-020-02834-y

[27] Georgopoulos, K., Emary, C., Zuliani, P.: Comparison of quantum-walk implementations on noisy intermediate-scale quantum computers. Phys. Rev. A **103**, 022408 (2021). https://doi.org/10.1103/PhysRevA.103.022408

[28] Acasiete, F., Agostini, F.P., Moqadam, J.K., Portugal, R.: Implementation of quantum walks on ibm quantum computers. Quantum Information Processing **19**(12) (2020). https://doi.org/10.1007/s11128-020-02938-5

[29] Balu, R., Castillo, D., Siopsis, G.: Physical realization of topological quantum walks on IBM-q and beyond. Quantum Science and Technology **3**(3), 035001 (2018). https://doi.org/10.1088/2058-9565/aab823

[30] Tang, H., Lin, X.-F., Feng, Z., Chen, J.-Y., Gao, J., Sun, K., Wang, C.-Y., Lai, P.-C., Xu, X.-Y., Wang, Y., et al.: Experimental two-dimensional quantum walk on a photonic chip. Science Advances **4**(5) (2018). https://doi.org/10.1126/sciadv.aat3174

[31] Qiang, X., Loke, T., Montanaro, A., Aungskunsiri, K., Zhou, X., O'Brien, J.L., Wang, J.B., Matthews, J.C.: Efficient quantum walk on a quantum processor. Nature Communications **7**(1) (2016). https://doi.org/10.1038/ncomms11511

[32] Jiao, Z.-Q., Gao, J., Zhou, W.-H., Wang, X.-W., Ren, R.-J., Xu, X.-Y., Qiao, L.-F., Wang, Y., Jin, X.-M.: Two-dimensional quantum walks of correlated photons. Optica **8**(9), 1129–1135 (2021). https://doi.org/10.1364/OPTICA.425879

[33] Godsil, C., Zhan, H.: Discrete-time quantum walks and graph structures. Journal of Combinatorial Theory, Series A **167**, 181–212 (2019). https://doi.org/10.1016/j.jcta.2019.05.003

[34] Montanaro, A.: Quantum walks on directed graphs. Quantum Information and Computation **7**(1&2), 93–102 (2007). `https://doi.org/10.26421/qic7.1-2-5`

[35] Carnia, E., Suyudi, M., Aisah, I., Supriatna, A.K.: A review on eigen values of adjacency matrix of graph with cliques. AIP Conference Proceedings (2017). `https://doi.org/10.1063/1.4995116`

[36] Li, X., Yang, G., Torres, C.L., Zheng, D., Wang, K.L.: A class of efficient quantum incrementer gates for quantum circuit synthesis. International Journal of Modern Physics B **28**(01), 1350191 (2013). `https://doi.org/10.1142/s0217979213501919`

[37] Golub, G. Van Loan, C.: Matrix computations, 4th Edition. Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore (2013).

[38] Mottonen, M., Vartiainen, J.J.: Decompositions of general quantum gates. arXiv (2005). `https://doi.org/10.48550/ARXIV.QUANT-PH/0504100`. https://arxiv.org/abs/quant-ph/0504100

[39] Tucci, R.R.: QC Paulinesia. arXiv (2004). `https://doi.org/https://doi.org/10.1007/978-1-4`. https://arxiv.org/abs/quant-ph/0407215

[40] LI, C.-K., Roberts, R., Yin, X.: Decomposition of unitary matrices and quantum gates. International Journal of Quantum Information **11**(01), 1350015 (2013). `https://doi.org/10.1142/s0219749913500159`

[41] Rotman, J.J.: An introduction to the theory of groups, 4th Edition. Springer New York, NY, Baltimore (1994). `https://doi.org/https://doi.org/10.1007/978-1-4612-4176-8`.

[42] Olver, F.W.J., et al.: NIST Digital Library of Mathematical Functions. http://dlmf.nist.gov/, Release 1.0.26 of 2020-03-15. Section 26.13 Permutations: Cycle Notation (2020). http://dlmf.nist.gov/

[43] Florkowski, S.F.: Spectral graph theory of the hypercube. PhD thesis, Naval Postgraduate School (2008)

[44] Douglas, B.L., Wang, J.B.: Efficient quantum circuit implementation of quantum walks. Physical Review A **79**(5) (2009). `https://doi.org/10.1103/physreva.79.052335`

[45] Daraeizadeh, S., Kumar, P.: Efficient implementation of multi-control toffoli gates in linear nearest neighbor arrays. PhD thesis, Wichita State University (2014)

[46] Rahman, M.Z., Rice, J.E.: Templates for positive and negative control toffoli networks. Reversible Computation, 125–136 (2014). `https://doi.org/10.1007/978-3-319-08494-7_10`

[47] Cheng, X., Guan, Z., Wang, W., Zhu, L.: A simplification algorithm for reversible logic network of positive/negative control gates. 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (2012). `https://doi.org/10.1109/fskd.2012.6233837`

[48] Arabzadeh, M., Saeedi, M., Zamani, M.S.: Rule-based optimization of reversible circuits. 2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC) (2010). `https://doi.org/10.1109/aspdac.2010.5419685`

[49] IBM quantum. `https://quantum-computing.ibm.com/`

[50] IBM quantum processor types. `https://quantum-computing.ibm.com/composer/docs/iqx/manage/systems/processors`

[51] Wing, A.: Allanwing-QC/quantum-walks-via-shunt-decomposition-circuits (2022). `https://github.com/allanwing-qc/Quantum-Walks-via-Shunt-Decomposition-Circuits`

[52] Mandviwalla, A., Ohshiro, K., Ji, B.: Implementing Grover's Algorithm on the IBM Quantum Computers. 2018 IEEE International Conference on Big Data (Big Data), 2531-2537 (2018). `https://doi.org/10.1109/BigData.2018.8622457`

[53] Liu, J., Bello, L., Zhou, H.: Relaxed Peephole Optimization: A Novel Compiler Optimization for Quantum Circuits. arXiv (2020). `https://doi.org/10.48550/ARXIV.2012.07711`. https://arxiv.org/abs/2012.07711

[54] Heese, R., Bickert, P., Niederle, A.E.: Representation of binary classification trees with binary features by quantum circuits. Quantum **6**, 676 (2022). `https://doi.org/10.22331/q-2022-03-30-676`

[55] Moore, C., Russell, A.: Quantum Walks on the Hypercube (2001)

[56] Portugal, R.: Quantum walks and search algorithms (2019). `https://doi.org/10.1007/978-1-4614-6336-8`

[57] Makmal, A., Zhu, M., Manzano, D., Tiersch, M., Briegel, H.J.: Quantum walks on embedded hypercubes. Physical Review A **90**(2) (2014). `https://doi.org/10.1103/physreva.90.022314`