Eur. Phys. J. D (2024)78:29
https://doi.org/10.1140/epjd/s10053-023-00795-2

THE EUROPEAN PHYSICAL JOURNAL D

Regular Article - Quantum Information

Cycle discrete-time quantum walks on a noisy quantum computer

Vivek Wadhia[1,a], Nicholas Chancellor[1,3,b], and Viv Kendon[1,2,c]

[1] Department of Physics, Durham University, Durham DH1 3LE, United Kingdom
[2] Department of Physics, University of Strathclyde, Glasgow G4 0NG, United Kingdom
[3] School of Computing, Newcastle University, Newcastle NE4 5TG, United Kingdom

Received 14 August 2023 / Accepted 23 December 2023

Abstract. The rapid development of quantum computing has led to increasing interest in quantum algorithms for a variety of different applications. Quantum walks have also experienced a surge in interest due to their potential use in quantum algorithms. Using the qiskit software package, we test how accurately the current generation of quantum computers provided by IBM can simulate a cycle discrete-time quantum walk. Implementing an 8-node, 8-step walk and a simpler 4-node, 4-step discrete-time quantum walk on an IBM quantum device known as `ibmq_quito`, the results for each step of the respective walks are presented. A custom noise model is developed in order to estimate that noise levels in the `ibmq_santiago` quantum device would need to be reduced by at least 94% in order to execute a 16-node, 16-step cycle discrete-time quantum walk to a reasonable level of fidelity.

1 Introduction

In recent years, quantum computing [1,2] has come to the forefront of developments in physics, with the promise of eventually being able to perform certain computations more efficiently than on a classical computer [3–5]. This has motivated the creation of a number of well-known quantum algorithms over the years in order to utilise this computational speed up, such as Shor's algorithm [6] and Grover's algorithm [7].

Along with the development of quantum computing, quantum walks [8,9] have also gained increasing interest. In large part, this is due to their application to quantum algorithms [10–13]. Quantum walks come in two main types, continuous-time and discrete-time. In this work, we focus on discrete-time quantum walks (DTQW), in particular a DTQW on a cycle graph [14], because it has a convenient encoding when executed on a digital quantum computer. However, current hardware has significant imperfections. Our aim in this work is to determine how much IBM processors need to improve to run DTQWs on cycle graphs.

This paper is structured as follows. In Sect. 2, an implementation of a cycle discrete-time quantum walk is presented using a binary encoding. In Sect. 3, we present the results of executing this algorithm for eight steps of an 8-node quantum walk, and then for four steps of a 4-node quantum walk, on an IBM quantum device known as `ibmq_quito`, as well as on a local, noiseless simulation using a quantum device known as `qasm_simulator`. The results for both devices are compared using a fidelity measure for probability distributions. In Sect. 4, the same algorithm is extended to 16 steps of a 16-node DTQW and run on `ibmq_santiago`, whilst also being run on a noisy version of the simulator, for varying noise levels, with the fidelity being calculated for every step of the walk. Results comparing the fidelity of the `ibmq_santiago` device with simulated noisy walks are presented. We conclude in Sect. 5.

2 Background

2.1 Quantum gates

In analogy with classical logic gates, quantum computations can be constructed from quantum logic gates. In this work, we use the following quantum gates: X gate acts in the same way a classical logical NOT gate does, $|0\rangle \rightarrow |1\rangle$ and $|1\rangle \rightarrow |0\rangle$. The Hadamard ($H$) gate creates a superposition state, $|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. A CNOT (Controlled-Not) gate flips the state of the target qubit if the state of the control qubit is $|1\rangle$. Similarly, a Toffoli ($C3$-X) gate flips the state of the target qubit if both of the control qubits are in the state $|1\rangle$. C4-NOT ($C4$-X) and

[a] e-mail: vivek.wadhia@outlook.com
[b] e-mail: Nick.Chancellor@newcastle.ac.uk (corresponding author)
[c] e-mail: viv.kendon@strath.ac.uk

Published online: 25 March 2024

C5-NOT ($C5\text{-}X$) gates flip the state of the target qubit if all 3 or 4 of the respective control qubits are in the state $|1\rangle$. An SX ($\sqrt{X}$) gate acts $|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$ and $|1\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$, where i is the imaginary unit ($i^2 = -1$). The identity gate (ID) leaves any state it acts on unchanged. RZ performs a rotation of the qubit state around the z-axis by a given angle.

For all DTQW quantum circuits used, the H gate acts as the Hadamard coin operator. X, C-NOT, Toffoli, C4-NOT and C5-NOT gates all act to change the value of the binary string (quantum register) that denotes the position of the walker.

2.2 Quantum walk on a graph

A graph $g(v, e)$ is defined as a set of nodes v and edges e that connect pairs of nodes. Of particular interest for this work are cycle graphs, denoted by C_n where n is the number of nodes. For example, Fig. 1 is a C_8 graph with the nodes labelled as used in the quantum walk algorithms.

A discrete-time quantum walk on a cycle graph requires a quantum walker that moves on the nodes of a graph, accompanied by a coin that is "flipped" to determine the direction to move in [14,15]. The full quantum system (of the walk) is a combination of the position state and the coin state, and the basis states are written $|x, c\rangle$ where x labels the node and $c \in \{0, 1\}$ the coin state. The general state of a DTQW at a time t is

$$|\psi(t)\rangle = \sum_{x,c} \alpha_{x,c}(t)|x, c\rangle, \tag{1}$$

where $\alpha_{x,c}(t) \in \mathbb{C}$ and normalised as $\sum_{x,c} |\alpha_{x,c}(t)|^2 = 1$. Conventionally, the walker is initialised at the origin, i.e. $x = 0$ at $t = 0$, and the coin is in some chosen initial state. The coin is "flipped" by applying a unitary operator, known as a coin operator. For this work, we use

$$\hat{\mathbf{C}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{2}$$

which is also known as a Hadamard coin, since the matrix is the same as a Hadamard quantum gate. A conditional shift operator then acts to move the quantum walker to adjacent nodes of the graph. This requires a mapping between coin states and the nodes at the ends of the edges to be specified [16]

$$\zeta : \mathbb{Z}_{|v|} \times \mathbb{Z}_d \rightarrow \mathbb{Z}_{|v|} \times \mathbb{Z}_d : (x, c) \mapsto \zeta(x, c) = (y, k). \tag{3}$$

$\mathbb{Z}_n$ is the additive group of integers $\{0, \ldots, n-1\}$ modulo n, d is the degree of the node (the number of edges connecting to it), x and y label each end of edge (x, y) and k is the updated coin state. The shift then acts

$$\hat{\mathbf{S}}|x, c\rangle = |y, k\rangle. \tag{4}$$

In other words, moving the walker and coin along the edge (x, y), with x being the starting node, y being the end node and updating the coin state, according to the mapping in equation (3). The application of the coin operator followed by the shift operator, at each unit time step, is sometimes denoted by the unitary operator $\hat{\mathbf{T}}$, i.e.

$$\hat{\mathbf{T}} = \hat{\mathbf{S}}\hat{\mathbf{C}}. \tag{5}$$

Therefore, a DTQW of t time steps is achieved by applying $\hat{\mathbf{T}}$, t times,

$$|\psi(t)\rangle = \hat{\mathbf{T}}^t|\psi(0)\rangle. \tag{6}$$

For this work, the quantum register is always initialised in the all zero state. That is $|0000\rangle$, $|000\rangle$ and $|00000\rangle$ for the 8-node, 4-node and 16-node DTQWs, respectively.

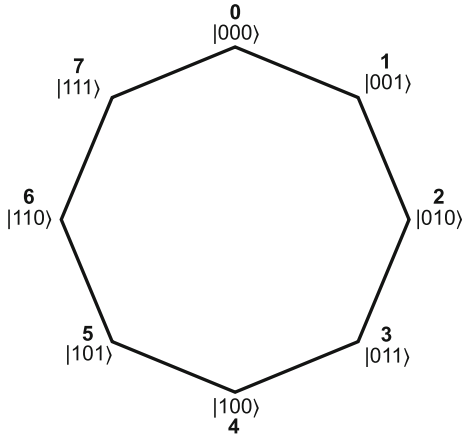

Fig. 1 8-node cycle graph. The binary string (i.e. the state of the quantum register) gives the site number shown in decimal (bold)

2.3 Algorithm

Using the work of Kemp et al. [17], who in turn build on the work of Douglas and Wang [18], it is possible to build a quantum circuit that performs a quantum walk on a cycle graph. For our purposes, we consider the example presented in [17], namely, that of a 4-qubit circuit that corresponds to an 8-node cycle graph. The cycle graph is depicted in Fig. 1. The nodes of the graph correspond to the possible positions of the walker.

The circuit for this 8-node walk is shown in Fig. 2. The quantum register consists of four qubits, three of which are used to represent the position of the walker as a binary label, whilst the final qubit is used as the coin state. Essentially, the circuit is built of two large sets of gates, an INCREMENT step gate and a DECREMENT step gate, acting as the shift operator, which themselves consist of CNOT and X gates. These INCREMENT and DECREMENT gates are repeated depending on the number of steps required. The coin operator used is a Hadamard gate applied before each INCREMENT step. One step involves H, INCREMENT, DECREMENT; two steps involve H, INCRE-

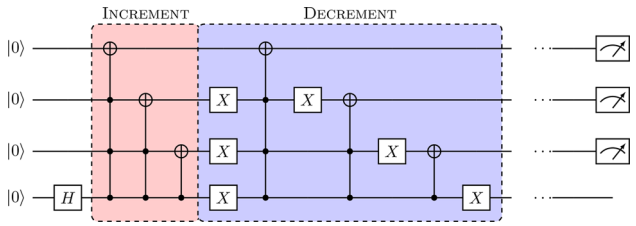

Fig. 2 Quantum circuit of [17]. Figure depicts one step of the walk. H, INCREMENT and DECREMENT gate sets are repeated for the desired number of steps. Quantum gates as they appear (from left to right) are: Hadamard, $C4$-X, Toffoli, CNOT, X, etc., ending with measurement. Bottom most qubit (as pictured in the diagram) acts as the coin, the rest of the qubits are used to determine the position label (see Fig. 1)

MENT, DECREMENT, H, INCREMENT, DECREMENT, and so forth.

2.4 Fidelity

In order to make a quantitative comparison between results from two different processors, such as `ibmq_quito` and the `qasm_simulator`, we now introduce the measure of fidelity to be used. The Hellinger fidelity [19] is defined as

$$\mathcal{F}_{\mathcal{H}} = (1 - \mathcal{H}^2)^2 \tag{7}$$

where $\mathcal{H}$ is the Hellinger distance.

$$\mathcal{H} = \left[\int \left(\sqrt{p(z)} - \sqrt{q(z)} \right)^2 \mathrm{d}z \right]^{\frac{1}{2}} \tag{8}$$

where $p(z)$ and $q(z)$ are two probability distributions [20]. Essentially, the Hellinger fidelity is a means of comparing how similar two classical probability distributions are. The probability distributions we compare are the results of the executed circuit which are expressed as counts, calculated from the number of times each position is measured as the outcome in a set of repeated runs of the circuit. The fidelity takes values in the range 0 to 1, with a value of 1 indicating a perfect correlation between distributions, and a value of 0 indicating a perfect anti-correlation between distributions. A value of 0.5 indicates no correlation, the distributions have a random overlap. This means we expect a fidelity of around 0.5 when we compare completely noisy states to the perfect outcome of a noise-free simulation. Noisy states tend to a value of 0.5 because this indicates maximum randomness in the results. A fidelity of less than 0.5 indicates the results of the computation are incorrect and, on average, anti-correlated. A fidelity of 0.8 or higher shows a clear correlation between the distributions and therefore indicates the results of the computation are more often correct than incorrect over repeated runs.

Table 1 Qiskit module versions

Module	Version
`qiskit-terra`	`0.16.4`
`qiskit-aer`	`0.7.6`
`qiskit-ignis`	`0.5.2`
`qiskit-ibmq-provider`	`0.12.1`
`qiskit-aqua`	`0.8.2`

2.5 Backends and transpilation

A *backend* refers to either a simulation of a quantum device or a real quantum device provided by IBM. There are three backends used in this work. The first is `qasm_simulator`, a local, noiseless, error-less, classical simulation of an arbitrary 5-qubit quantum device. The second is `ibmq_quito`, a real 5-qubit quantum device, and the third is `ibmq_santiago`, also a real 5-qubit quantum device. Quantum circuits are executed by first being programmed using the qiskit software package, and then being submitted online (for the case of real quantum devices) to join a queue of all jobs to be executed on that particular backend. Once execution is completed, results are displayed through the qiskit software frontend.

Transpilation [21] is the process of converting quantum circuits, consisting of a range of standard gates into the native gates used by IBM quantum devices. There are only certain gates that can be executed on IBM quantum hardware. The gates are: {CNOT, ID, RZ, SX, X}. These form a universal set of gates, so all operations can be implemented using only these gates. The transpilation process occurs automatically when the user submits the circuit to be executed to the backend. Transpilation does not occur in the `qasm_simulator` as it is not a real quantum device, and can simulate all gates directly.

2.6 Software versions used

Design and execution of circuits was done using `qiskit_v0.24.0` running on `Python 3.8.5`. `Spyder 4.1.5` is used as the IDE to write code along with `IPython 7.19.0`. Table 1 shows the version for each qiskit module.

3 Runs and simulations

The rest of this paper examines DTQWs in the following order of dimensionality. Beginning with 8-node, followed by 4-node, and finally in Sect. 4 a 16-node. The reason for this ordering is to determine the impact of noise on the fidelity of results from both smaller and larger systems, as compared to the 8-node system already established in literature.

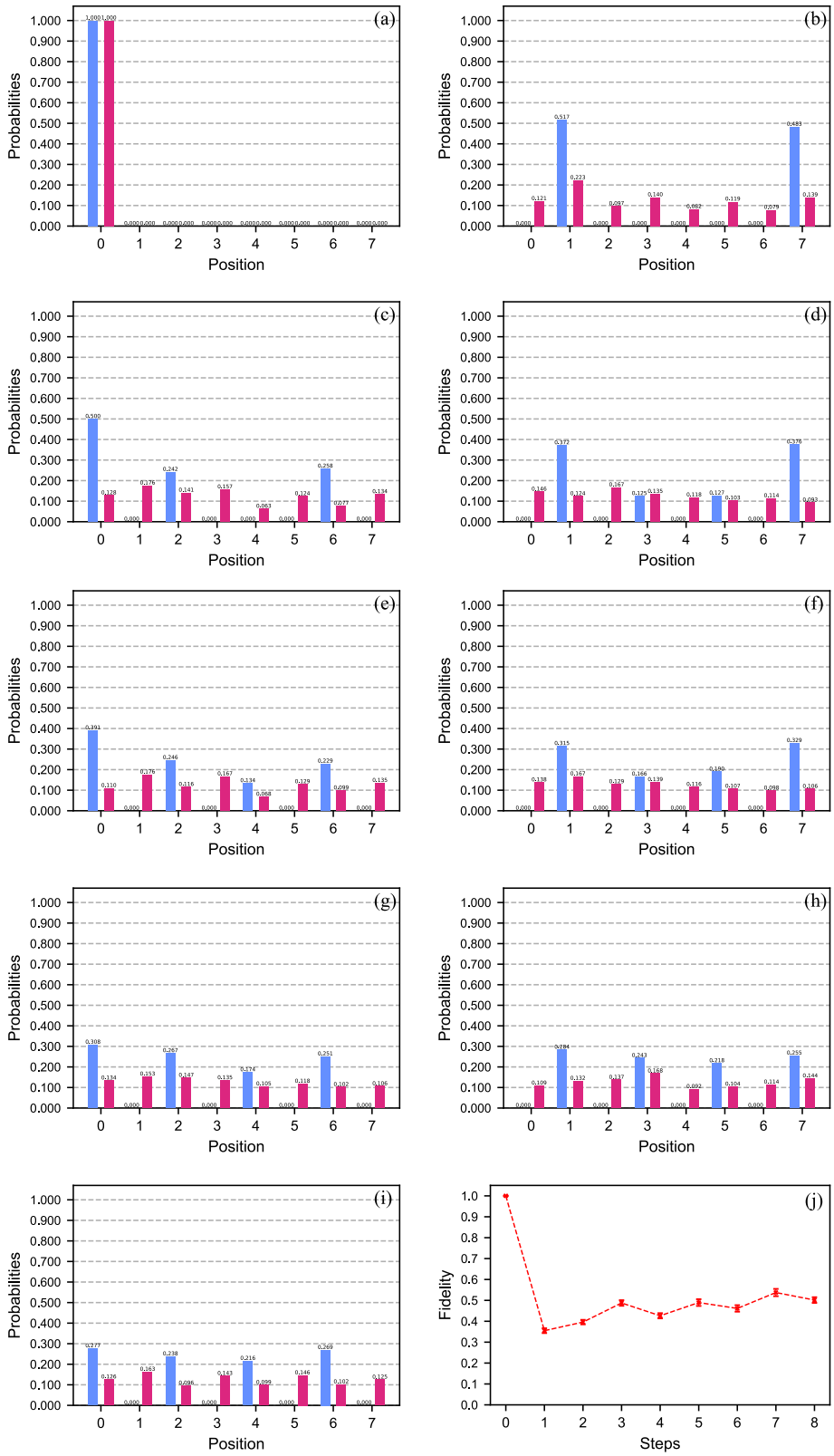

Fig. 3 Discrete-time quantum walk for an 8-node graph. Panels **a**–**i** show steps 0 to 8 of the walk, respectively. Bars represent the probability of finding the walker at a particular site. The blue bars represent the result of the simulator i.e. the expected result and red bars the real device. Panel **j** shows the Hellinger fidelity (7) between the real device and the simulator for each step of the walk (dotted line is to guide the eye). Error bars on panel **j** represent standard error

3.1 Eight node quantum walk

Using the algorithm and circuit presented in Sect. 2.3 and Fig. 2, respectively, an eight-step DTQW on a cycle graph consisting of eight nodes is executed on `ibmq_quito` and the `qasm_simulator`, with the counts for each measured location of the quantum walk recorded for each step of the walk. To do this, the walk needs to be repeated for each number of steps, i.e. the walk is executed for a specific number of steps and the register is measured. Then the walk is restarted, the desired number of steps are executed, and the register is again measured. This process is repeated to obtain results for a all the diferent steps in a walk. To compare the performance of the `ibmq_quito` versus the `qasm_simulator`, the fidelity defined in Sect. 2.4 is calculated from the probability distributions derived from the repeated runs. Since the simulator is regarded as providing the ideal (expected) result, free of the effect of any noise or errors, the fidelity quantifies the performance of the real `ibmq_quito` hardware. The results are presented in Fig. 3.

As can be seen in Fig. 3, the `ibmq_quito` device performs poorly. A basic expectation would be that the real device follows the first few steps of the walk with a reasonable level of fidelity. However, panel **j** of Fig. 3 shows that the performance of the device does not achieve this, with an average fidelity of 0.45 from step one onwards. Furthermore, a fidelity below 0.5 indicates an anti-correlation between the expected results and the real device. This could be due to bit flip error(s) occurring early in the computation, causing the value of the quantum register to become anti-correlated compared to the expected value. However, as the number of steps increases, the fidelity tends to 0.5, indicating the dominance of noise for longer computations.

3.2 Four-node quantum walk

Let us now consider a simpler version of the algorithm presented in Sect. 2.3, one that consists of only four nodes instead of eight, as shown in Fig. 4.

The circuit now needs just three qubits in total, qubits one and two are used to record the position and qubit three is used for the coin. This quantum walk simulation circuit is shown in Fig. 5. Specifically, the pattern the circuit follows is: H acting on coin, STEP, H acting on coin, STEP and so on for further steps of the quantum walk. The circuit only consists of one large gate, a STEP gate that incorporates the functionality of the INCREMENT and DECREMENT gates of [17].

This algorithm was executed on `ibmq_quito` for four steps of the walk, with the counts measured at each step and the fidelity with the `qasm_simulator` calculated. The results are presented in Fig. 6.

As can be seen in Fig. 6, `ibm_quito` performs the 4-node quantum walk with far higher fidelity than the 8-node. Although the fidelity of the results do decrease slightly during the initial steps, all steps are maintained at a fidelity of higher than 0.8, showing a good correlation between simulator and the real device. In other words, the correct computations are being performed, on average. This is in clear contrast to the results presented in Fig. 3.

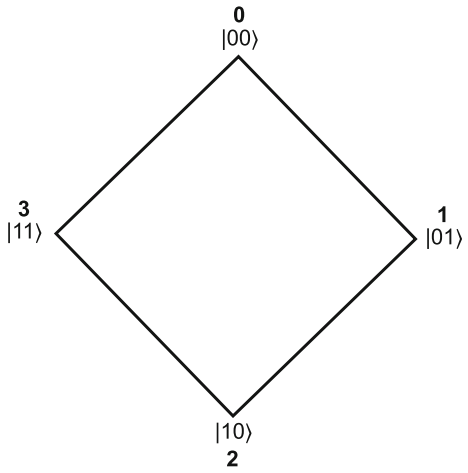

Fig. 4 4-node quantum walk graph. As in Fig. 1 nodes represent the possible positions of the walker and the binary string corresponds to the label number in decimal (bold)

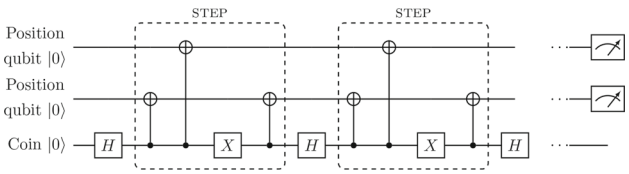

Fig. 5 Circuit for 4-node quantum walk. Decomposition of STEP gates are enclosed by the dashed lines. Coin operation is given by a single Hadamard gate acting on qubit three. Two complete steps of the quantum walk are shown, these gates are repeated for the desired number of steps

3.3 Analysis of 8- and 4- node walks

Let us now consider these results, and why the 4-node quantum walk performs similarly to the simulator whilst the 8-node counterpart does not. On inspection of their respective quantum circuits (shown in Figs. 5, 2, respectively), both appear to be relatively simple. However, focusing on the INCREMENT and DECREMENT gates of the 8-node circuit, they contain Toffoli and C4-NOT ($C4$-X) gates. Implementation of these gates proves to be a non-trivial task. The qiskit software has a method of implementing Toffoli gates by reducing them into a set of other gates. In total, this decomposition consists of 18 gates: six CNOT gates, 10 RZ gates and 2 SX gates for each Toffoli. This is already a large increase in the total number of gates required to perform a single Toffoli operation. For the case of a C4-NOT or generally any Cn-NOT gate of higher order (e.g. $C5$-X, $C6$-X, etc.), there currently exists no optimised way of implementing these gates in qiskit. A C4-NOT is transpiled into a 34-gate circuit: 14 CNOT gates, 18 RZ and two SX. A single

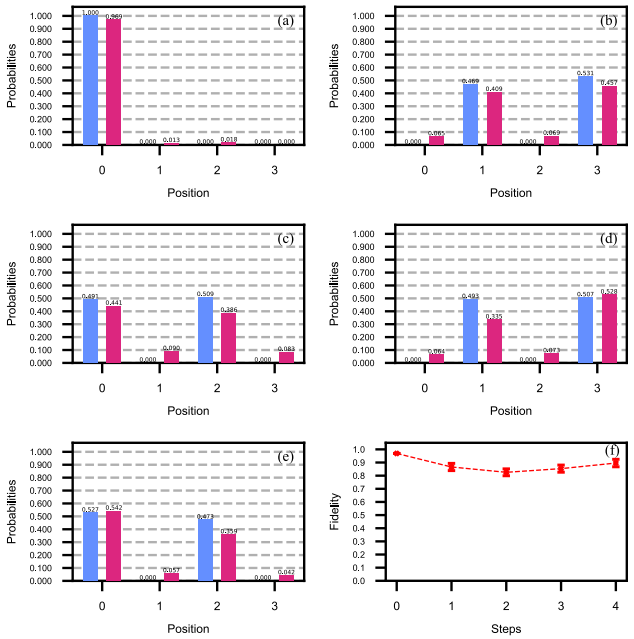

Fig. 6 DTQW for a 4-node graph. Panels **a**–**e** show steps 0 to 4 of the walk, respectively. Bars represent the probability of finding the walker at a certain site denoted by a bit string. The blue bars represent the result of the simulator and red bars the result of the real device. Panel **f** shows the fidelity, comparing the two probability distributions, of the walk for each step between the real device and simulator. Error bars represent standard error

C4-NOT gate thus needs a large number of gates that form a fairly large circuit by themselves, and the implementation from [17] contains two Toffolis and two C4-NOT gates for a single step of the quantum walk. This explains why the quantum walk on `ibm_quito` appears as just noise. The number of gates being implemented, even for a single step, has enough noise associated with each gate to degrade the results to the point where it is unrecognisable. In comparison, the 4-node circuit does not contain any Toffoli or C4-NOT gates, so the only gates to transpile are H and X, which explains why the results achieve a higher fidelity. This circuit, when transpiled is much closer to Fig. 5, and contains a modest number of gates, so noise degradation does not ruin the result. It is worth mentioning that the transpiling process does not solely consist of transforming the circuit into gates, but also decides which qubits to use in the device, determined by the hardware's topology, among other processes [21]. Factors such as the aforementioned topology constraints can have a negative impact on the fidelity of a computation by, for example, increasing the total number of gates in a circuit, leading to more noise.

4 Custom noise models

A feature of the qiskit software package is the ability to create and use noise models. IBM offers a noise model of each available backend that, in principle, models the noise levels of that device, on that day, as accurately as qiskit will allow. However, it is also possible for the user to create their own custom noise models by varying, and including/not including, parameters associated with noise in superconducting quantum computers. Noise models are created using the `qasm_simulator` as a base template, noise parameters are then introduced on top of this template. The main parameter in the custom noise models comes from the depolarising error channel, and is modelled by [22]

$$E(\rho) = (1-\lambda)\rho + \lambda \mathrm{Tr}[\rho]\frac{\mathbb{1}}{2^n} \tag{9}$$

with $0 \leq \lambda \leq 4^n/(4^n-1)$, where λ is the depolarising error parameter, n is the number of qubits, $\mathbb{1}$ is the identity matrix (normalised by dividing by 2^n) and ρ is the density matrix of the state.

For depolarising noise, the quantum correlations are removed over time, and this will turn a quantum walk into a classical random walk. The limiting distribution for a classical random walk on any regular graph (such as the cycle) is a uniform distribution, independent of the initial state. However, we are simulating noise on the quantum computer, which is not the same as a model of a noisy quantum walk. Nonetheless, removing quantum correlations will still have the effect over long times of making the output of the computation a random distribution over the possible outcomes, independent of the initial state.

We can use the customised noise model to estimate how much noise levels in the IBM quantum device `ibmq_santiago` would need to be reduced in order to achieve a consistent, high-fidelity result for a discrete-time quantum walk. To investigate this, consider the DTQW on a cycle graph, using the same paradigm as the walks in the previous section. However, now the quantum register is extended to a total of five qubits, this being the maximum number available to us. Four qubits are used for the binary encoding of the position of the walker and the fifth qubit stores the coin state. Accordingly, the number of sites on the graph, for this quantum walk, can now increase to a total of 16, as shown in Fig. 7. The corresponding quantum circuit for this walk is shown in Fig. 8.

The parameters that make-up a noise model are: single-qubit gate errors, two-qubit gate errors, three-qubit gate errors, and multiple (i.e. four and five for this algorithm) qubit gate errors. The single-qubit gate errors correspond to the single qubit unitary operators in the circuit. In the case of the 16-node DTQW that we use, this includes H and X gates. The two, three and multiple qubit errors correspond to CNOT, Toffoli and $C4\text{-}X/C5\text{-}X$ gates, respectively. In practice, these errors represent depolarising error rates associated with each of the gates, and are numbers specified by the user.

In order to create a custom noise model that is able to accurately model a real device, we begin by an initial guess of the error rates associated with each of the aforementioned gate types, single-qubit, two-qubit, etc.

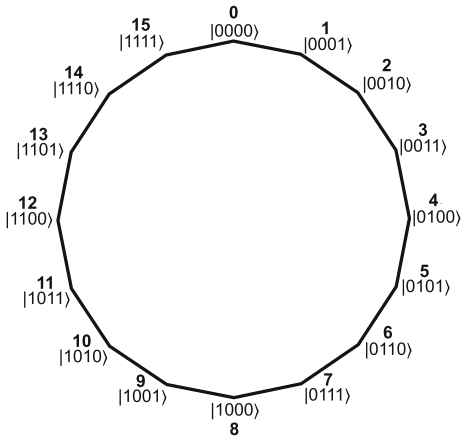

Fig. 7 Diagram of the 16-node cycle graph used in the DTQW. Nodes represent the possible positions of the walker and the binary string corresponds to the node label in decimal (bold)

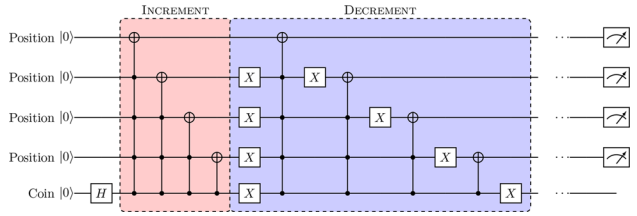

Fig. 8 Quantum circuit for the 16-node DTQW using 5 qubits. Figure depicts one step of the walk. Bottom most qubit (as pictured) acts as the coin and the rest of the qubits are used to represent the position of the walker

Table 2 Error rate per gate used in the custom noise model to emulate `ibmq_santiago`

Type of gate	Error
Single-qubit	0.005
Two-qubit	0.02
Three-qubit	0.04
Multiple-qubit	0.6

Multiple-qubit refers to four or more qubit gates

We then examine the outputted results of the 16-node, 16-step cycle DTQW, run on this custom noise model, and compare it to the results outputted by the real device. By examining how close the Hellinger fidelity of the two distributions (from the custom noise model and real device) are, the values for the gate errors can be refined. By refining the gate errors until a Hellinger fidelity of 1.0, or as close to that value as possible, is achieved, a custom noise model of a real device is able to be constructed. The final values for the gate errors, which are the probability of error per gate, are shown in Table 2 below.

The values shown in Table 2 are referred to in this text as *full noise strength*, as they refer to the maximum amount of noise modelled, i.e. the amount of noise that most closely resembles the real device.

As mentioned previously, IBM also provides a noise model for each of their backends. This noise model has the error rates pre-determined by automated analysis of each of the devices every so often, e.g. every 24 h. Due to the susceptibility of transmon qubits to environmental noise, these error rates vary over time. Hence, this method of modelling noise is a means of keeping track of that variation, and provides the most accurate error values. The other advantage is that the user does not need to define each individual error rate to match the current noise levels, this is done automatically. In addition to depolarising errors, the IBM noise model also contains: readout errors, errors associated with the measurement of qubits, and a thermal relaxation error that essentially consists of the T_1 and T_2 relaxation times [23]. It is important to note that the depolarising and thermal relaxation errors only apply to single- and two-qubit gates, so the noise model needs to be applied to circuits with three-qubit and higher gates transpiled into one- and two-qubit gates. In this paper, the noise model provided by IBM is referred to as the *IBM noise model*.

In order to investigate how noise affects the performance of this generation of IBM quantum computers, the DTQW for 16 nodes is executed, first on the base `qasm_simulator` with no noise, and second using the custom noise model, for 16 steps of the walk in single step increments. The noise level is then reduced in gradual decrements, until there are no depolarising errors in the custom noise model. The same DTQW is then executed on the IBM noise model of `ibmq_santiago` and the real `ibmq_santiago`. The fidelity for each step of the quantum walk is recorded up to a maximum of 16 steps. The fidelity for each backend: custom noise model, IBM noise model and `ibmq_santiago`, is found by comparing the probability distributions of each step of the walk with the corresponding step from the `qasm_simulator`, which can be thought of as the ideal or expected result, as it contains no noise.

Figure 9 shows the fidelity for each step of the quantum walk and for decreasing noise levels. In Fig. 9, 100% refers to 100% of the full noise strength, i.e. the values in Table 2; 90% refers to 90% of each error in Table 2, etc. At 0% of full noise strength, the errors all take a value of 0. However, it is important to note that this is not exactly the same (although extremely close) as the base `qasm_simulator`. The distinction is that at 0% noise strength the noise model still takes into account the small errors introduced by the transpilation process (see Sect. 2.5 and [21] for explanation).

As can be seen from Fig. 9, for all noise strengths except 0%, the performance is poor. Noise strengths 60–100% all begin with a fidelity below 0.5, in other words they cannot even produce an accurate result for one step of the quantum walk (recall that one step consists of the sequence of single and multi-qubit gates shown in Fig. 8). Similarly, noise strengths 30–50% show some improvement but still not enough to be considered accurate. Significant improvements in the fidelity only begin to be seen when at 10% and 20% of

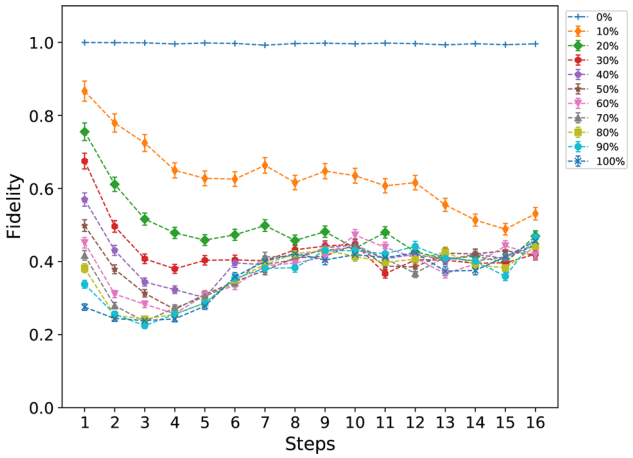

Fig. 9 Fidelity at each step of a 16-node, 16-step DTQW, for varying noise strengths of the custom noise model. Error bars represent standard error

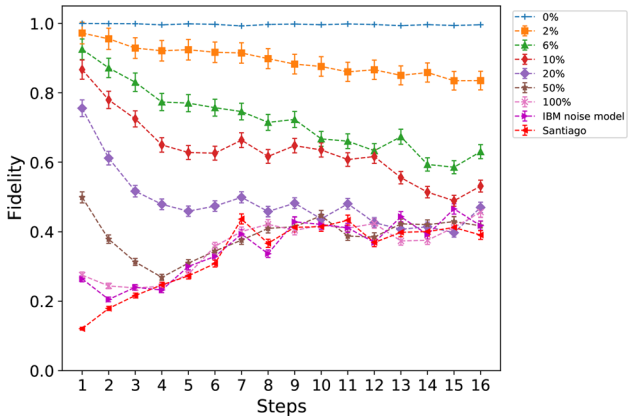

Fig. 10 Fidelity at each step of a 16-node, 16-step DTQW, for varying noise strengths of the custom noise model as well as the IBM noise model and `ibm_santiago`. Error bars represent standard error

full noise strength. However, the fidelity soon declines in the case of 20% noise strength, and after just 2 steps the results of the walk are indiscernible from random outputs. In the case of 10% of full noise strength, the performance is markedly better, with the fidelity being consistently higher than all other noise strengths for every step of the walk. Here, approximately 13 steps of the walk are achieved at a reasonable standard with a final fidelity around 0.6, which is roughly the limit before the results of the walk start to become indistinguishable from random outputs i.e. noise. The 0% noise strength achieves a constant fidelity of almost exactly 1.0, with minor fluctuations occurring due to the transpilation errors. This confirms that 0% noise strength and the `qasm_simulator` are very similar. In all other cases for the custom noise model, it can be seen that the fidelity tends to an equilibrium value of approximately 0.5 as the number of steps tends to 16, showing the effect of noise overcoming the results of the walk. Figure 10 shows the fidelity for the same DTQW, for selected noise strengths, and includes the noise model of `ibmq_santiago` as well as `ibmq_santiago` itself.

From Fig. 10 a direct comparison between the noise models and `ibmq_santiago` can be made. Overall, the worst performing is the real device, starting with a low fidelity that gradually increases but never exceeds 0.5. This indicates a slight anti-correlation with the expected results, particularly for the first few steps. However, beyond roughly 6 steps these tend to random outputs (noise). The performance of the IBM noise model at 100% strength is very similar to `ibmq_santiago`. For the IBM noise model, this would be expected but the custom noise model also follows the real device very closely, overlapping at 4, 9 and 10 steps. Although overall the performance of the IBM noise model is slightly closer to the real device, it is interesting to see this near correspondence. This indicates that depolarising errors are the dominant contribution to noise, since it is possible to so closely model (as shown in Figs. 9, 10) the real device without the addition of other errors. The addition of 2% and 6% noise strength also shows the non-linear improvement in performance once the level of noise is below 10% of the initial parameters. For 6% noise strength, the entire walk of 16 steps is achievable with a reasonable fidelity, only dropping slightly towards 16 steps, even at this point the results of the walk are distinguishable from any random outputs (noise). The 2% noise strength performs excellently, and by far has the highest overall fidelity (aside from 0% of course), never dropping below 0.8 fidelity. All steps of the walk are considered accurate and extrapolation of Fig. 10 would seem to suggest that many more additional (probably around 8) steps could be achieved with high or good fidelity results for 2% noise strength. We therefore conclude that in order for a DTQW on a 16-node cycle graph to be executed with consistent, high-fidelity results, for each step, the noise levels within `ibmq_santiago` would need to be reduced by approximately 94%.

The explanation for the poor fidelity for the 8-node cycle presented in in Sect. 3 is now clear. The poor fidelity of `ibmq_santiago`, the IBM noise model and most of the custom noise models, is due to the transpilation process expanding the number of native gates, and with it, the total noise.

5 Conclusion

We have carried out two different size cycle discrete-time quantum walks on an IBM quantum computer, `ibmq_quito`, showing that the device is unable to produce high-fidelity results for an 8-node, 8-step walk. However, it is able to produce reasonably high-fidelity results for a 4-node, 4-step walk. We have established the reason for this discrepancy in results between the two walks is due to the transpilation process. Specifically, the 4-node walk uses fewer qubits and fewer gates, as well as using only CNOT gates, whilst the

8-node walk requires *C3-X* (Toffoli) and *C4-X* gates, that are transpiled into many more native gates. Therefore, less noise occurs in the execution of the 4-node walk, and it is able to achieve a much higher fidelity. Section 4 then established, using custom noise models, a method of approximating by how much the noise in `ibmq_santiago` would need to be reduced in order to execute a 16-node, 16-step cycle discrete-time quantum walk. Inspection of Figs. 9 and 10, revealed that a decrease in noise levels of approximately 94% would be sufficient to achieve this task.

Although our conclusions show that this generation of IBM quantum computers have a long way to go in terms of reduction of noise levels, for even a modest-sized DTQW, it is encouraging that a smaller DTQW is currently viable.

Quantum walks are by no means the only algorithm that can be used to benchmark performance of quantum computers. For example, randomised benchmarking [24,25] has been used to characterise noise in IBM quantum computers. Furthermore, combinatorial optimisation problems have also been used to measure performance [26]. However, fundamentally, all quantum algorithms implemented on IBM superconducting quantum computers use quantum gates. Specifically, we tested the gates in Sect. 2.5. Therefore, the conclusions regarding transpilation and noise associated with each gate, highlighting in particular Toffoli, *C4-X* and *C5-X* gates, hold true for any algorithm that uses those gates, implemented on these processors.

Acknowledgements VK and NC were partially funded by UKRI EPSRC Grant No. EP/T026715/1 and EP/T026715/2, and by the UK Quantum Technology Hub in Computing and Simulation (grant EP/T001062/1). We acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

Author contributions

VK and NC proposed initial study. VK proposed use of fidelity. VW designed 4-node DTQW circuit and custom noise models. VW performed data collection. All authors performed analysis. First draft of the manuscript was written by VW, VK and NC contributed to subsequent amendments. All authors read and approved the final manuscript.

Data availability statement This manuscript has associated data in a data repository. [Authors' comment: Data sets and code used in this manuscript available at the Durham University Collection: https://doi.org/10.15128/r15x21tf47n.]

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

1. R.P. Feynman, Simulating physics with computers. (1982). https://doi.org/10.1007/BF02650179
2. D. Deutsch, Quantum theory, the Church–Turing principle and the universal quantum computer. Proc. R. Soc. Lond. **400**, 97 (1985)
3. D. Deutsch, R. Jozsa, Rapid solution of problems by quantum computation. Proc. Math. Phys. Sci. **439**(1907), 553–558 (1992)
4. E. Bernstein, U. Vazirani, Quantum complexity theory. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*. STOC '93 (Association for Computing Machinery, New York, NY, USA, 1993), pp. 11–20 . https://doi.org/10.1145/167088.167097
5. D.R. Simon, On the power of quantum computation. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 116–123 (1994). https://doi.org/10.1109/SFCS.1994.365701
6. P.W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997). https://doi.org/10.1137/s0097539795293172
7. L.K. Grover, A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '96 (Association for Computing Machinery, New York, NY, USA, 1996) pp. 212–219. https://doi.org/10.1145/237814.237866
8. S. Gudder, Quantum probability and operational statistics. Found. Phys. **20**(5), 499–527 (1990). https://doi.org/10.1007/BF01883237
9. Y. Aharonov, L. Davidovich, N. Zagury, Quantum random walks. Phys. Rev. A **48**, 1687–1690 (1993). https://doi.org/10.1103/PhysRevA.48.1687
10. E. Farhi, S. Gutmann, Quantum computation and decision trees. Phys. Rev. A **58**(2), 915–928 (1998). https://doi.org/10.1103/physreva.58.915
11. D. Aharonov, A. Ambainis, J. Kempe, U. Vazirani, Quantum walks on graphs. In *Conference Proceedings of the Annual ACM Symposium on Theory of Computing* (2001). https://doi.org/10.1145/380752.380758
12. A. Ambainis, E. Bach, A. Nayak, A. Vishwanath, J. Watrous, One-dimensional quantum walks. In *Conference Proceedings of the Annual ACM Symposium on*

Theory of Computing (2001). https://doi.org/10.1145/380752.380757
13. N. Shenvi, J. Kempe, K.B. Whaley, Quantum random-walk search algorithm. Phys. Rev. A **67**, 052307 (2003). https://doi.org/10.1103/PhysRevA.67.052307
14. S.E. Venegas-Andraca, Quantum walks: a comprehensive review. Quantum Inf. Process. **11**(5), 1015–1106 (2012). https://doi.org/10.1007/s11128-012-0432-5
15. V. Kendon, C. Tamon, Perfect state transfer in quantum walks on graphs. J. Comput. Theor. Nanosci. **8**, 422 (2010). https://doi.org/10.1166/jctn.2011.1706
16. V. Kendon, Quantum walks on general graphs. Int. J. Quantum Inf. **04**(05), 791–805 (2006). https://doi.org/10.1142/s0219749906002195
17. J. Kemp, S. Nishio, R. Satoh, D. Vogt-Lee, T. Bassan, https://github.com/qiskit-community/qiskit-community-tutorials/blob/master/terra/qis_adv/quantum_walk.ipynb. Accessed 25 Apr 2021
18. B.L. Douglas, J.B. Wang, Efficient quantum circuit implementation of quantum walks. Phys. Rev. A **79**, 052335 (2009). https://doi.org/10.1103/PhysRevA.79.052335
19. https://qiskit.org/documentation/stubs/qiskit.quantum_info.hellinger_fidelity.html. Accessed 25 June 2021
20. L. Wasserman, *Lecture Notes* 27, pp. 36–705 (2020). https://www.stat.cmu.edu/~larry/=stat705/Lecture27.pdf
21. https://qiskit.org/documentation/apidoc/transpiler.html. Acessed 25 Apr 2021
22. https://qiskit.org/documentation/stubs/qiskit_aer.noise.depolarizing_error.html. Accessed 28 Aug 2021
23. https://qiskit.org/documentation/stubs/qiskit.providers.aer.noise.NoiseModel.html#qiskit.providers.aer.noise.NoiseModel.from_backend. Accessed 24 Aug 2021
24. E. Magesan, J.M. Gambetta, J. Emerson, Scalable and robust randomized benchmarking of quantum processes. Phys. Rev. Lett. **106**(18), 180504 (2011). https://doi.org/10.1103/physrevlett.106.180504
25. E. Magesan, J.M. Gambetta, J. Emerson, Characterizing quantum gates via randomized benchmarking. Phys. Rev. A **85**(4), 042311 (2012). https://doi.org/10.1103/physreva.85.042311
26. M.T. Khumalo, H.A. Chieza, K. Prag, M. Woolway, An investigation of IBM quantum computing device performance on combinatorial optimisation problems. Neural Comput. Appl. (2022). https://doi.org/10.1007/s00521-022-07438-4