Universal quantum computation using the discrete-time quantum walk

Neil B. Lovett,[*] Sally Cooper, Matthew Everitt, Matthew Trevers, and Viv Kendon[†]
School of Physics and Astronomy, University of Leeds, Leeds LS2 9JT, United Kingdom

(Received 19 October 2009; revised manuscript received 2 March 2010; published 30 April 2010)

A proof that continuous-time quantum walks are universal for quantum computation, using unweighted graphs of low degree, has recently been presented by A. M. Childs [Phys. Rev. Lett. **102**, 180501 (2009)]. We present a version based instead on the discrete-time quantum walk. We show that the discrete-time quantum walk is able to implement the same universal gate set and thus both discrete and continuous-time quantum walks are computational primitives. Additionally, we give a set of components on which the discrete-time quantum walk provides perfect state transfer.

DOI: 10.1103/PhysRevA.81.042330 PACS number(s): 03.67.Ac, 05.40.Fb

I. INTRODUCTION

Quantum computers offer the promise of fundamentally faster processing based on quantum mechanical properties. Although a physical device of a useful size is still to be built, many quantum algorithms have already been discovered. The most important of these are the algorithms introduced by Shor [1] and Grover [2], which can factor integers and search an unsorted database, respectively, significantly faster than the best known classical algorithms [1,2].

Quantum walks were initially introduced in both continuous [3] and discrete [4] time, in direct analogy with their classical counterparts, and have since been studied extensively [5]. In the same way that classical random walks are used in computer science for algorithm design, many quantum algorithms have been developed based on quantum walks, with varying speed-ups over the best known classical algorithms for the same problem [6]. These solve the problems using two different approaches: hitting times and searching. In hitting-time problems we start from a specific vertex and want to get to another as quickly as possible. These problems have yielded the largest speed-up, including exponential speed-ups over the classical case [7–9]. Searching for an entry in an unsorted database is a classically time-consuming problem taking on average a time of $O(N)$ to search a set of N entries. Grover's algorithm [2] improves on this to $O(\sqrt{N})$ by using a technique known as amplitude amplification. The same speed-up can be obtained using a quantum walk method on various structures [10–12].

In [13], Childs extends the original results of Feynman [14] to show that a continuous-time quantum walk, on an unweighted graph of bounded degree, is universal for quantum computation. Childs [13] gives an explicit construction that converts a standard gate model computation into a graph, on which a continuous-time quantum walk executes an algorithm by traversing the graph. In this article, we show the equivalent construction of a universal gate set using the discrete-time quantum walk in place of the continuous one. This confirms that both the continuous and the discrete-time quantum walks can be regarded as computational primitives. The construction requires an exponentially large graph for the size of the input as we require 2^n wires for an n-qubit input. The quantum walk takes place on this N-vertex graph just as the continuous-time walk does in the construction by Childs [13]. It is already known that a quantum walk on an N-vertex graph can be simulated efficiently by a universal quantum computer using poly($\log_2 N$) gates, provided there is a simple rule for computing the neighbors of any vertex [7]. Thus, by performing the quantum walk on a quantum computer, the binary encoding brings the resources required back to the expected level.

Our construction for the universal gate set in discrete time is similar to [13] but has maximum degree, d, of eight at any vertex as opposed to three in the continuous case. The continuous-time walk can easily be propagated in one direction with no reflection at the vertices. The discrete-time walk is not so straightforward; it can only be propagated in one direction by using a specific coin corresponding to the σ_x operation. Using this coin restricts the graph to vertices of degree two, providing no way to construct higher-degree structures. Thus, we must use a double-edged wire to accomplish directional propagation. This solution has its roots in the connection between the continuous and the discrete-time walks. Strauch [15] has shown that, as we take the continuous limit of the discrete-time walk on the line, we actually get two copies of the continuous-time walk propagating in opposite directions. Childs [16] later showed a direct correspondence between the discrete and continuous-time quantum walks on arbitrary graphs. In the same work, Childs shows how a discrete-time walk can be used, at its limit of small eigenvalues, to approximate the continuous-time walk. He uses this "lazy" quantum walk approach to allow the discrete-time walk to propagate in the same way as the continuous. This same approach could be used in this case to allow the computation to be performed on the same structures defined in [13]. However, this would require the discrete-time walk to approach the limit at which it is doing very little at each time step. This would then increase the overhead required to allow completely deterministic computation.

We begin by describing the discrete-time quantum walk briefly in Sec. II, then move on to show structures on which the discrete-time quantum walk will allow perfect state transfer in Sec. III. These structures allow us to construct the elements we need to perform computation. Section IV shows the universal gate set we choose and how we implement these using the discrete-time walk. Section V describes how we can link these

[*]pynbl@leeds.ac.uk
[†]V.Kendon@leeds.ac.uk

gates and structures together to form any quantum circuit and elaborates on how this is efficient, despite the size of the graph being exponential in the number of gates required. Finally, in Sec. VI we discuss our findings and the differences with the continuous-time construction of [13].

II. DISCRETE-TIME QUANTUM WALK

Consider a classical random walk on a line in which a walker starts at a specific position and, depending on the outcome of a coin toss, moves either left or right. The outcome after many runs is a binomial distribution about the starting position with a spread (quantified by the standard deviation) of $\sqrt{t}$, where t is the number of time steps. A discrete-time quantum walk is the direct analog of the classical walk with the walker replaced by a quantum particle carrying a two-state quantum system for the coin. The coin toss is effected by a unitary operator. Although this is now deterministic, if we were to measure the coin we would get a random output as in the classical case. We start the quantum walker at the origin and act upon it with a unitary operator for the coin toss, followed by a conditional shift operation (to obtain the movement of the walker) at each time step. We write the basis states of the walker as an ordered pair, $|x,c\rangle$, denoting the position of the walker, x, and the state of the coin, that is, heads ($c = 1$) or tails ($c = 0$). The simplest unitary operator is the Hadamard operator, H, which acts on the state as

$$\begin{aligned} H|x,0\rangle &= \frac{1}{\sqrt{2}}(|x,0\rangle + |x,1\rangle), \\ H|x,1\rangle &= \frac{1}{\sqrt{2}}(|x,0\rangle - |x,1\rangle). \end{aligned} \tag{1}$$

The shift operation, S, acts thus:

$$\begin{aligned} S|x,0\rangle &= |x-1,0\rangle, \\ S|x,1\rangle &= |x+1,1\rangle. \end{aligned} \tag{2}$$

The first three time steps starting at the origin are as follows:

$$\begin{aligned} &(SH)^3|0,0\rangle \\ &= (SH)^2 S\frac{1}{\sqrt{2}}(|0,0\rangle + |0,1\rangle) \\ &= (SH)^2 \frac{1}{\sqrt{2}}(|{-1},0\rangle + |1,1\rangle) \\ &= (SH)S\frac{1}{2}(|{-1},0\rangle + |-1,1\rangle + |1,0\rangle - |1,1\rangle) \\ &= SH\frac{1}{2}(|{-2},0\rangle + |0,1\rangle + |0,0\rangle - |2,1\rangle) \\ &= S\frac{1}{\sqrt{8}}(|{-2},0\rangle + |{-2},1\rangle + |0,0\rangle - |0,1\rangle \\ &\quad +|0,0\rangle + |0,1\rangle - |2,0\rangle + |2,1\rangle) \\ &= \frac{1}{\sqrt{8}}(|{-3},0\rangle + |{-1},1\rangle + 2|{-1},0\rangle - |1,0\rangle + |3,1\rangle). \end{aligned} \tag{3}$$

As the walk progresses, quantum interference occurs whenever there is more than one possible path of t steps to the position. This can be both constructive and destructive, as shown in Eq. (3), which causes some probabilities to be amplified or decreased at each time step. The walk on the line has been solved analytically [17,18], where it was first remarked that the quantum walk spreads quadratically faster than the classical one.

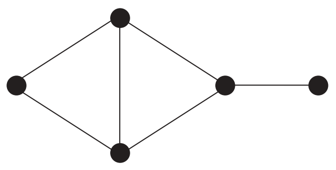
FIG. 1. A general graph with five vertices and six edges. It is undirected and has vertices of varying degree.

The choice of operator at each vertex can greatly affect the dynamics of the walk and its propagation across the structure. A bias can be introduced [19] for $d = 2$; this is done using a generalization of the Hadamard operator,

$$H_{\text{bias}} = \begin{pmatrix} \sqrt{\delta} & \sqrt{1-\delta} \\ \sqrt{1-\delta} & -\sqrt{\delta} \end{pmatrix}, \tag{4}$$

where δ is the bias in the coin. Setting this to $\delta = \frac{1}{2}$ returns the standard Hadamard operator [Eq. (1)]. Similarly, the choice of the walker's initial state is also important, unlike in the classical random walk. A good review of these effects can be found in [20].

For universal computation we need a quantum walk on a more complex graph. In graph theory, a general graph, G, is an ordered pair consisting of a set of vertices, V, and a set of edges, E, which link the vertices. The number of edges incident on a vertex is the degree of the vertex. Figure 1 shows a small general graph which has vertices of varying degree. It is also undirected, meaning that the edges allow movement in both directions. A different operator (coin) is needed at vertices with $d > 2$ in order to act on the entire state space [21,22].

The Grover coin can be extended to any degree at a vertex,

$$G^{(d)} = \begin{bmatrix} \frac{2}{d} & \cdots & \frac{2}{d} \\ \vdots & \ddots & \vdots \\ \frac{2}{d} & \cdots & \frac{2}{d} \end{bmatrix} - I_d, \tag{5}$$

where d is the degree of the vertex and I_d is the identity matrix of the same dimension.

Coins of degree four will be needed at most of the vertices in our computational graphs. The Grover coin in four dimensions [Eq. (5)] reduces to

$$G^{(4)} = \frac{1}{2}\begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix}. \tag{6}$$

Using these higher-dimensional coins can cause the walker to be reflected back upon itself with some probability. Figure 2 shows this reflection for a vertex of degree $d = 4$. In quantum-walk search and other quantum-walk algorithms this can be useful to provide interference. However, here we need to ensure that the walker moves in one direction only, from left to right, so it most resembles the circuit model of quantum

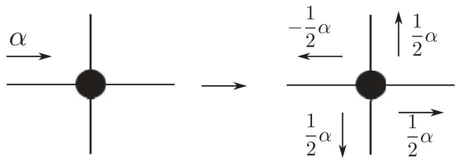

FIG. 2. An example of a portion of the walker reflecting back upon itself. This is a single degree-four vertex with the Grover coin [Eq. (5)] operating on the incoming amplitude α.

computation. We show how we accomplish this forward-only propagation in Sec. IV.

III. PERFECT STATE TRANSFER

As a preliminary to our quantum computation scheme, we discuss structures on which perfect state transfer can be achieved using the discrete-time quantum walk. Perfect state transfer has been investigated in the context of spin chains by Landahl *et al.* [23,24]. The propagation of the state through spin systems follows the same dynamics as a continuous-time quantum walk. Perfect state transfer can occur on chains of length 2 or 3, hypercubes of any size, and chains with different coupling strengths engineered to optimize state transfer.

The closely related properties of instantaneous mixing and periodic cycles have been studied in detail for quantum walks. For the continuous-time quantum walk, instantaneous mixing has been investigated by Tamon *et al.* [25–28]. They showed in [25] that this is achieved on cycles of two, three, and four vertices only. For the discrete-time walk, slightly larger cycles show exact periodic behavior. Travaglione and Milburn [29] showed that a cycle of four vertices has a periodicity of eight time steps, after which the entire state returns to the starting position. Tregenna *et al.* [20] showed that more periodic cycles exist, cycles of two, three, four, five, six, eight, and ten were shown numerically to be periodic by varying both the bias and the phase in the coin. Perfect state transfer occurs at half the periodic cycle for even cycles, where we obtain the entire state at the opposite point of the cycle, as shown in Fig. 3.

For our case of using the walk for computation, we require the walk to travel perfectly in a single direction. On the structures mentioned, the quantum walk travels around the

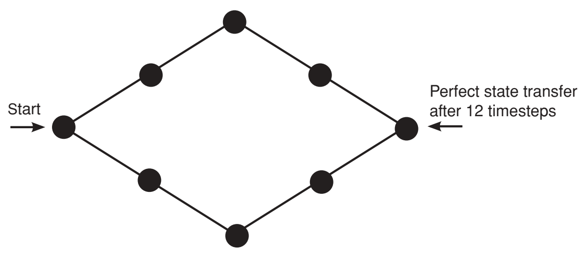

FIG. 3. Cycle of eight vertices which gives perfect state transfer from the initial vertex to the opposite vertex after half of the period, 12 time steps. The entire state returns to the initial vertex in a full period, 24 time steps.

cycle in both directions and interferes to produce perfect state transfer. Using a completely biased coin,

$$H_{\text{max. bias}} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{7}$$

we can make the state transfer perfectly around the cycle in a single direction. However, if we then try to attach another structure to the cycle, this periodicity is broken in both cases. The Grover coin [Eq. (5)] can be used to overcome part of this problem at vertices with an equal number of input edges as output edges. For any vertex of even degree, it will transfer the entire state from all the input edges to all the output edges provided the inputs are all equal in both amplitude and phase. These results led us to the designs that work for universal computation.

IV. UNIVERSAL GATE SET

We now show how we construct a universal gate set with the discrete-time quantum walk. Although the gate set we implement is the same as in [13], the structures used to propagate the discrete-time walk are different. The gate set used is the standard universal set comprising the controlled-not (CNOT) gate,

$$C_{\text{NOT}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{8}$$

the single-qubit Hadamard,

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{9}$$

and phase-shift gates (we implement the specific phase shift known as the $\frac{\pi}{8}$ gate),

$$P\left(\frac{\pi}{8}\right) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}. \tag{10}$$

These gates create a universal set that can implement any quantum computation [30].

In order to represent quantum states, Childs defines his computational basis states as quantum wires. The other gates required for universality are then attached to wires and used to connect them together. The computation is represented as a quantum walk on these wires and structures, where the computation flows from input to output (left to right in our diagrams). Note that this encoding is not meant to be implemented directly. The wires represent computational basis states rather than qubits; thus, the model does not represent a physical architecture. Instead, the underlying graph structure created would be used to help "program" a quantum computer. We first show how to construct a simple wire along which the quantum walk will propagate naturally in one direction. We use two edges per wire to ensure that no reflection occurs at a vertex. We distribute the walker across the two edges, which then recombine at the next vertex. As the split is equal, the Grover coin in effect moves both halves to the output edges

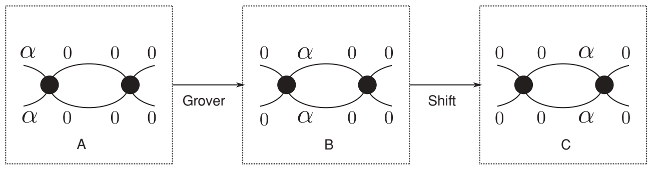

FIG. 4. Grover and shift operation acting on a vertex of degree $d = 4$. Section A shows the initial state, B shows the state after the Grover coin is applied, and C is after the shift operation.

of the vertex. Figure 4 shows this operation and the "shift" to the next vertex in explicit steps. The Grover diffusion coin [Eq. (5)] is used at each vertex of degree $d = 4$. The initial and final vertices are in effect degree four if we include other edges attached to either end. Figure 5 shows the basic wire we use. The computation would start with the amplitude at the initial vertex spread equally across the pair of edges in a wire. For example, the state $\alpha|0\rangle + \beta|1\rangle$, where $|\alpha|^2 + |\beta|^2 = 1$, would be split thus,

$$|\psi_{\text{initial}}\rangle = \frac{1}{\sqrt{2}}\left[\alpha|0\rangle_a + \alpha|0\rangle_b + \beta|1\rangle_a + \beta|1\rangle_b\right], \quad (11)$$

where the subscript a refers to the top line of the wire and subscript b is the bottom line. The walk propagates left to right on the wire deterministically, in this case reaching the incoming edges of the final vertex in four time steps. These wires form the basic connections in the computation.

The simplest gate to construct is the CNOT. It is trivial to implement by just exchanging the wires of the second qubit. The CNOT gate is shown in Fig. 6 and shows how the second qubit is flipped but the first qubit is untouched.

The phase gate [Eq. (10)] requires the addition of a relative phase to one wire or computational basis state in relation to the other. To accomplish this, but still have only one coin operator for each vertex of the same degree, we modify the basic wire and add a phase factor, $e^{i\phi}$, to it,

$$G_{\phi}^{(4)} = e^{i\phi}G^{(4)}. \quad (12)$$

Thus, as the walk propagates along a basic wire, it now picks up a phase of $e^{i\phi}$ each time it passes through a vertex of degree $d = 4$. For the wires shown in Fig. 5 the walker would pick up a phase of $e^{5i\phi}$ as it reaches the final vertex. The phase added here is arbitrary and can be set to any value so long as it is set to the same value for all vertices of degree $d = 4$. As we are looking to implement a $\pi/8$ gate, we set it as follows: $\phi = -\pi/4$. In order to add a relative phase of $\pi/4$ between the $|0\rangle$ and $|1\rangle$ wires, we insert the structure in Fig. 7 into the graph at the required point. In this structure there are also

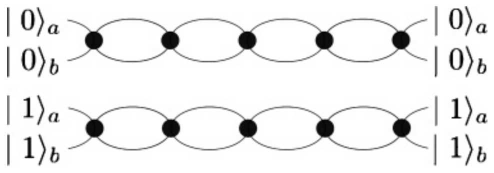

FIG. 5. Basic wire used to propagate the quantum walk from left to right only. At a vertex of degree $d = 4$, the Grover diffusion coin is used. The initial state is split across the pair of edges in the wire [Eq. (11)].

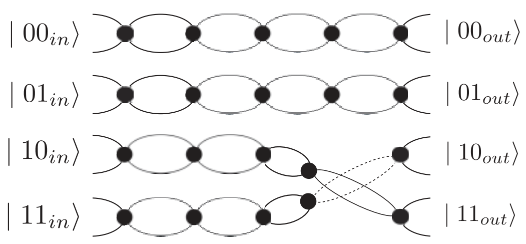

FIG. 6. Structure used to implement a CNOT gate. In this case the first qubit is the control and the second is the target. The target qubit's wires are interchanged and the control qubit is left unaltered. The dotted lines represent wires passing underneath the solid lines—there is no interaction between these wires.

vertices of degree $d = 2$, at which we use the Grover coin at its limit of degree $d = 2$,

$$G^{(2)} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \quad (13)$$

We add no phase to Eq. (13) so as the walker passes through these vertices no phase is picked up. In Fig. 7, the walker propagates along the $|1\rangle$ wire and picks up a phase of $e^{-i\pi}$ in four time steps as it only passes through four vertices of degree four. However, the $|0\rangle$ wire picks up a phase of $e^{-5i\frac{\pi}{4}}$ in the same number of time steps as all its vertices are of degree four. Relative to the $|0\rangle$ wire, the $|1\rangle$ wire will pick up a phase of $e^{i\frac{\pi}{4}}$. Therefore, using the structure described here we obtain the operation in Eq. (10).

The last gate in the universal set is the Hadamard gate. This requires an interaction between the two computational basis states. The structure we use to perform this operation is shown in Fig. 8. This looks complex in relation to the other gates we have shown so we break it up to explain it more clearly. Sections A and C of the structure are each two phase gates giving a relative phase of i to the $|1\rangle$ wire before and after the main section of the gate (B). Section B of the structure combines the two inputs from the $|0\rangle$ and $|1\rangle$ wires and then splits this across the outputs equally. The structure here is similar to the basis-changing gate in [13]. In order to obtain

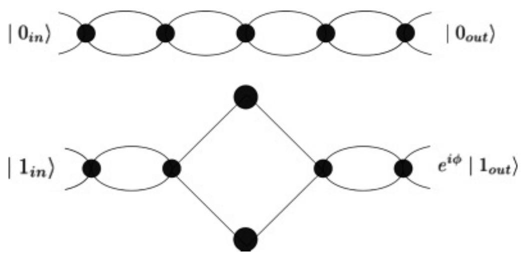

FIG. 7. Phase-gate structure. The $d = 2$ Grover coin [Eq. (13)] is used at the vertices of degree $d = 2$. The $|1\rangle$ wire will pick up a phase of $e^{i\phi}$ relative to the $|0\rangle$ wire. In our construction we actually obtain the operation corresponding to a phase of $e^{i\frac{\pi}{4}}$ as we set $\phi = -\pi/4$.

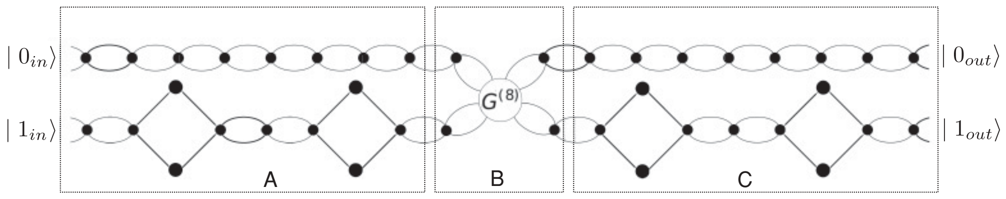

FIG. 8. Hadamard gate structure. Sections A and C add a relative phase of i to the $|1\rangle$ wire. The structure adds a global phase of $3\pi/4$ to the wires.

the desired operation on this structure, we have designed a coin for vertices of degree $d = 8$:

$$G^{(8)} = \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & i & i & -1 \\ 0 & 0 & 0 & 0 & i & 1 & -1 & i \\ 0 & 0 & 0 & 0 & i & -1 & 1 & i \\ 0 & 0 & 0 & 0 & -1 & i & i & 1 \\ i & -1 & 1 & i & 0 & 0 & 0 & 0 \\ -1 & i & i & 1 & 0 & 0 & 0 & 0 \\ 1 & i & i & -1 & 0 & 0 & 0 & 0 \\ i & 1 & -1 & i & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{14}$$

This operator combines the complex Hadamard operator,

$$H_i = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & i \\ i & 1 \end{bmatrix}, \tag{15}$$

and the σ_x,

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{16}$$

in a tensor product form,

$$G^{(8)} = (H_i \otimes H_i) \otimes \sigma_x, \tag{17}$$

with the top two and bottom two rows of the $H_i \otimes H_i$ matrix rearranged. This rearrangement ensures the outputs come out in the same order as the input states. This gate adds a global phase of $3\pi/4$. The phase gates at the start and end of the central section give us the relative phase of -1 required for the Hadamard operation. We note here that the choice of where to place the phases in this construction is arbitrary. The same result can be achieved by using the degree-four Grover coin [Eq. (6)] with no phase at vertices of degree four and the degree-two Grover coin [Eq. (13)], with a phase of $\pi/4$ at vertices of degree two. However, by placing the phases on the Grover coin in the fashion we described previously [Eqs. (12) and (13)] means that the global phase added by the Hadamard gate [Eq. (14)] corresponds to the phase added by a wire of the same length.

V. CONSTRUCTING QUANTUM CIRCUITS

Thus far, each gate we have described only acts on one or two qubits. However, nontrivial quantum computers involve many qubits. We now describe how to link these wires and structures together to form larger circuits. Figure 10 shows the underlying graph structure of the circuit in Fig. 9. The graph structure is obtained by connecting together wires and structures so that the walk flows from left to right. For this reason, we designed our wire and structures with both input and output vertices of degree four, thus making it simple to link them together. The initial state of the computation is set on all or a subset of the vertices on the left-hand side of the graph, with the amplitude at each vertex split across the incoming edges. This initial state can be thought of as the first column of vertices in the graph structure in superposition, with each subsequent column of vertices representing a further time step. For example, in Fig. 10 this column of vertices is the set prior to the Hadamard structures. The walker is propagated across the graph structure, from left to right deterministically, for the required number of time steps. We therefore do not require the addition of momentum filters or separators as in the continuous-time case. Our structures all propagate the walker at the same speed, meaning output from the wires will be synchronized throughout the computation. Finally, the walker picks up a global phase of $-\pi/4$ per vertex that is not part of a gate that changes the phase, so all the wires also stay synchronized in phase. Thus, we know with certainty that, after the required number of time steps, the walker will have a distribution over just the output vertices on the right-hand side of the graph. Once the computation has been completed, we measure the output vertices. We will find the walker at just one of these vertices, representing the output of the computation.

The graph structure in Fig. 10 is clearly larger in size than its equivalent representation in the circuit model (Fig. 9). In fact, for a general n-qubit computation the equivalent graph will have 2^n wires, one for each combination of computational basis states. Similarly, we require more gate structures than in the circuit model. Single-qubit structures are repeated 2^{n-1} times and for the CNOT gate we need 2^{n-2} structures. As an example, we can see that the phase gate acting on qubit 3 in Fig. 9 is repeated four times in the underlying graph structure of Fig. 10, one for each combination of wires involving qubit 3. Although this seems as though we would lose any form of quantum speed-up due to the exponential number of gates required in

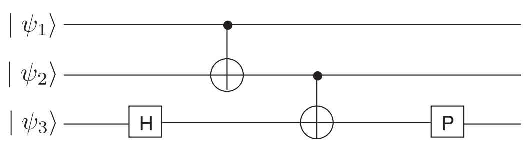

FIG. 9. Quantum circuit on three qubits. A Hadamard operation is performed on qubit 3 followed by two CNOT gates. Finally, a phase gate is applied on qubit 3. The underlying graph of this structure is shown in Fig. 10.

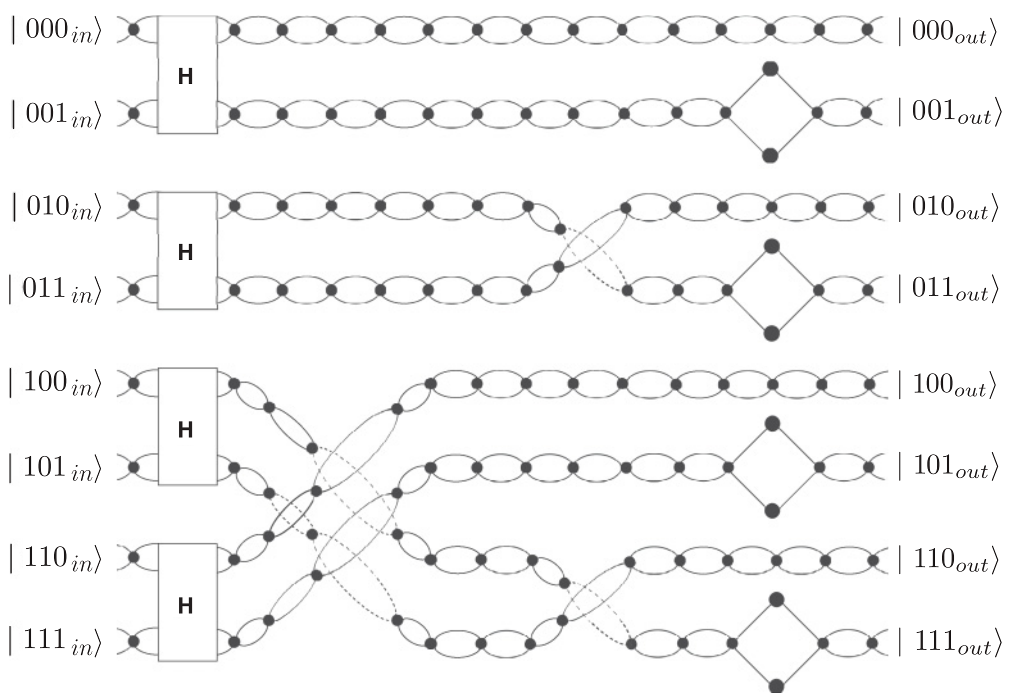

FIG. 10. Graph representing the quantum circuit in Fig. 9. The Hadamard structure, H, is the same as in Fig. 8. The dotted lines represent wires passing underneath the solid lines—there is no interaction between these wires.

the underlying graph, this is not the case. Consider simulating a classical random walk on a classical computer, the N-vertex graph is represented in $\log_2 N$ bits of memory with each vertex having a unique binary number as a label. In a similar fashion, if we simulate a quantum walk on a quantum computer, the N-vertex graph can be represented by $\log_2 N$ qubits. Therefore, if we encode our graph using qubits, we can describe the 2^n wires in just n qubits. By manipulation of a single qubit, we can affect all combinations of wires associated with that qubit. As the state moves across the graph, the adjacent vertices must be established. In complex graphs the description of the graph and its connections is often exponential in size and an oracle must be used to store it [7]. The graphs produced here are of bounded degree and have a regularity stemming from the repetition of gate structures on combinations of wires involving a specific qubit. Due to the labeling of the wires, we know where to place each structure based on one bit in the label; thus, we can efficiently describe the graph. For example, consider the second CNOT gate in Fig. 9, which operates on qubit 3 with qubit 2 as control. We can see from Fig. 10 that it is easy to identify where the CNOT structures should be placed. The labeling of the wires shows that the middle bit determines which combinations of wires relate to the second qubit having a value of 1. Similarly, we can also identify which wire it should link to by the last bit in the labeling scheme: It needs to be flipped relative to the original wire; that is, $|010\rangle$ links to $|011\rangle$.

VI. DISCUSSION

In this article we have described an alternative to the construction in [13] using the discrete-time quantum walk. This shows the discrete-time quantum walk is universal; therefore, any quantum algorithm can be reformulated as a discrete-time quantum walk algorithm. It also confirms that the discrete and continuous-time walks are both computational primitives and thus computationally equivalent. This equivalence is dependent on the number of steps in both cases being of the same order. Our gate constructs require twice the number of edges compared to the continuous-time case but the same number of wires. Our phase gate requires an additional time step in relation to the continuous-time phase-gate construct. The number of time steps required for a computation is also the same as the continuous-time case but with a small overhead depending on the number of phase gates required.

Another difference in the two constructions is the degree of the graphs produced. In the continuous-time case the maximum degree of any vertex in the graph is three. In the discrete-time case we use vertices of higher degree to ensure directional propagation. In most of the structures, this is a doubling of the degree at a vertex, as shown in the case of the basic wire and the phase-gate structure. The Hadamard structure we propose here, however, does not follow this doubling. It would seem reasonable, from the equivalent degree-three structure in the work by Childs, that it may be possible to decompose our Hadamard structure into one with degree-six vertices. The doubling of the degree at vertices would then correspond directly to the continuous case.

ACKNOWLEDGMENTS

We thank Andrew Childs for helpful comments on a draft of the manuscript. N.L. is funded by the UK Engineering and Physical Sciences Research Council. V.K. is funded by a Royal Society University Research Fellowship. S.C. and M.T. were funded by Nuffield Foundation Science Undergraduate Research Bursaries. M.E. was funded by the University of Leeds.

[1] P. W. Shor, SIAM J. Comput. **26**, 1484 (1997).
[2] L. K. Grover, in *Proceedings of the 28th Annual ACM STOC, 1996* (ACM, New York, 1996), p. 212.
[3] E. Farhi and S. Gutmann, Phys. Rev. A **58**, 915 (1998).
[4] Y. Aharonov, L. Davidovich, and N. Zagury, Phys. Rev. A **48**, 1687 (1993).
[5] J. Kempe, Contemp. Phys. **44**, 307 (2003).
[6] A. Ambainis, Int. J. Quantum Inf. **1**(4), 507 (2003).
[7] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. Spielman, in *Proceedings of the 35th ACM Symposium on Theory of Computing* (Association for Computing Machinery, New York, 2003), p. 59.
[8] J. Kempe, in *Proceedings of the RANDOM'03 Lecture Notes in Computer Science* (Springer, 2003), p. 354.
[9] A. M. Childs, L. J. Schulman, and U. V. Vazirani, in *Proceedings of the 48th IEEE Symposium on Foundations of Computer Science* (IEEE, Washington, DC, 2007), p. 395.
[10] N. Shenvi, J. Kempe, and K. B. Whaley, Phys. Rev. A **67**, 052307 (2003).
[11] A. M. Childs and J. Goldstone, Phys. Rev. A **70**, 022314 (2004).
[12] A. M. Childs and J. Goldstone, Phys. Rev. A **70**, 042312 (2004).
[13] A. M. Childs, Phys. Rev. Lett. **102**, 180501 (2009).
[14] R. P. Feynman, Opt. News **11**, 11 (1985).
[15] F. W. Strauch, Phys. Rev. A **74**, 030301(R) (2006).
[16] A. M. Childs, Commun. Math. Phys. **294**, 581 (2010).
[17] A. Ambainis, E. Bach, A. Nayak, A. Vishwanath, and J. Watrous, in *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing* (ACM, New York, 2001), p. 60.
[18] A. Nayak and A. Vishwanath, e-print arXiv:quant-ph/0010117.
[19] E. Bach, S. Coppersmith, M. Paz-Goldschen, R. Joynt, and J. Watrous, J. Comput. Syst. Sci. **69**(4), 562 (2004).
[20] B. Tregenna, W. Flanagan, R. Maile, and V. Kendon, New J. Phys. **5**, 83 (2003).
[21] T. D. Mackay, S. D. Bartlett, L. T. Stephenson, and B. C. Sanders, J. Phys. A **35**, 2745 (2002).
[22] V. Kendon, Int. J. Quantum Inf. **4**(5), 791 (2006).
[23] M. Christandl, N. Datta, A. Ekert, and A. J. Landahl, Phys. Rev. Lett. **92**, 187902 (2004).
[24] M. Christandl, N. Datta, T. C. Dorlas, A. Ekert, A. Kay, and A. J. Landahl, Phys. Rev. A **71**, 032312 (2005).
[25] A. Ahmadi, R. Belk, C. Tamon, and C. Wendler, Quantum Inf. Comp. **3**, 611 (2003).
[26] L. Fedichkin, D. Solenov, and C. Tamon, Quantum Inf. Comp. **6**(3), 263 (2006).
[27] W. Carlson, A. Ford, E. Harris, J. Rosen, C. Tamon, and K. Wrobel, Quantum Inf. Comp. **7**(8), 738 (2007).
[28] A. Best, M. Kliegl, S. Mead-Gluchacki, and C. Tamon, Int. J. Quantum. Inf. **6**(6), 1135 (2008).
[29] B. C. Travaglione and G. J. Milburn, Phys. Rev. A **65**, 032310 (2002).
[30] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, Phys. Rev. A **52**, 3457 (1995).