An Efficient Circuit for the Quantum Walk Update Rule

Chen-Fu Chiang* Daniel Nagaj† Pawel Wocjan‡

March 23, 2009

Abstract

We show how to efficiently implement quantum update rules corresponding to arbitrary sparse classical walks (Markov chains). These rules are required to realize quantum walks as defined by Szegedy [8]. Our efficient construction settles the often-raised objection against this core element of quantum walk based algorithms. The key component we use is Grover and Rudolph's method for preparing coherent versions of probability distributions that are obtained by discretizing efficiently integrable probability densities [15].

1 Introduction

For many tasks, such as simulated annealing [1, 2], computing the volume of convex bodies [3] and approximating the permanent of a matrix [4, 5] (see references in [6] for more), the best approaches known today are randomized algorithms based on Markov chains (random walks) and sampling. A Markov chain on the state space Ω is described by a stochastic matrix $P = (p_{xy})_{x,y\in\Omega}$. Its entry p_{xy} is equal to the probability of making a transition from state x to state y in the next step. If the Markov chain P is ergodic (see e.g. [7]), then there is a unique probability distribution $\pi = (\pi_x)_{x\in\Omega}$ such that $\pi P = \pi$. This probability distribution is referred to as the stationary distribution. Moreover, we always approach π from any initial probability distribution, after applying P infinitely many times. For simplicity, we assume that the Markov chain is reversible, meaning that the condition $\pi_x p_{xy} = \pi_y p_{yx}$ is fulfilled for all distinct x and y. The largest eigenvalue of the matrix P is $\lambda_0 = 1$. The corresponding eigenvector is equal to the stationary distribution π. How fast a given Markov chain approaches π is governed by the second eigenvalue λ_1 of P (which is strictly less than 1), or viewed alternatively, by the eigenvalue gap $\delta = 1 - \lambda_1$ of the matrix P. This determines the performance of random walk based algorithms whose goal is to sample from the stationary distribution π.

In [8], Szegedy defined a *quantum walk* as a quantum analogue of a classical Markov chain. Each step of the quantum walk needs to be unitary, and it is convenient to define it on a quantum

*School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA. Email: cchiang@eecs.ucf.edu

†Research Center for Quantum Information, Institute of Physics, Slovak Academy of Sciences, Dúbravská cesta 9, 84215 Bratislava, Slovakia, and Quniverse, Líščie údolie 116, 841 04, Bratislava, Slovakia. Email: daniel.nagaj@savba.sk

‡School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA. Email: wocjan@eecs.ucf.edu

system with two registers $\mathcal{H} = \mathcal{H}_L \otimes \mathcal{H}_R$. The *quantum update rule* is any unitary that acts as

$$U |x\rangle_L |0\rangle_R = |x\rangle_L \sum_y \sqrt{p_{xy}} |y\rangle_R \tag{1}$$

on inputs of the form $|x\rangle_L |0\rangle_R$ for all $x \in \Omega$. (Its action on inputs $|x\rangle_L |y \neq 0\rangle_R$ can be chosen arbitrarily.) Using such U, we define two subspaces of $\mathcal{H}$. First,

$$\mathcal{A} = \text{span}\{U |x\rangle_L |0\rangle_R\} \tag{2}$$

is the span of all vectors we get from acting with U on $|x\rangle_L |0\rangle_R$ for all $x \in \Omega$, and second, the subspace $\mathcal{B} = S\mathcal{A}$ is the subspace we get by swapping the two registers of $\mathcal{A}$. Using the quantum update, we can implement a reflection about the subspace $\mathcal{A}$ as

$$\text{Ref}_{\mathcal{A}} = U \left(2 |0\rangle \langle 0|_R - \mathbb{I}\right) U^\dagger. \tag{3}$$

Szegedy defined a step of the quantum walk as

$$W = \text{Ref}_{\mathcal{B}} \cdot \text{Ref}_{\mathcal{A}}, \tag{4}$$

a composition of the two reflections about $\mathcal{A}$ and $\mathcal{B}$. This operation is unitary, and the state

$$|\psi_\pi\rangle = \sum_x \sum_y \sqrt{\pi_{xy}} |x\rangle_1 |y\rangle_2 , \tag{5}$$

where π is the stationary distribution of P, is an eigenvector of W with eigenvalue 1. Szegedy [8] proved[1] that when we parametrize the eigenvalues of W as $e^{i\pi\theta_i}$, the second smallest phase θ_1 (after $\theta_0 = 0$) is related to the second largest eigenvalue λ_1 of P as $|\theta_1| > \sqrt{1 - \lambda_1}$. This can be viewed as a square-root relationship $\Delta > \sqrt{\delta}$ between the phase gap $\Delta = |\theta_1 - \theta_0|$ of the unitary operator W and the spectral gap $\delta = |\lambda_0 - \lambda_1|$ of P. This relationship is at the heart of the quantum speedups of quantum walk based algorithms over their classical counterparts.

Szegedy's generalized quantum walk model is used in many of the recent quantum walk algorithms for searching [10], quantum simulated annealing [11], quantum sampling [12, 13] and approximating partition functions based on classical Markov chains [6]. For all these algorithms, an essential step in implementing the quantum walk W is the ability to implement the quantum update rule (1). For the basic search-like and combinatorial algorithms, an efficient implementation of the corresponding quantum walks is straightforward. However, for complicated transition schemes coming from Markov chains like those for simulated annealing or for approximating partition functions of the Potts model, the situation is not so clear-cut. Recently, we have often heard objections claiming that in those cases it could be hard to efficiently apply U, thus destroying the speedups achieved by the quantum walk-based algorithms.

The common answer about the implementation of U relies on the methods for efficient simulation of sparse Hamiltonians [14]. When there is only a small number of neighbors connected to each x, one might try these methods. However, the subtle quantum walk algorithms require U to work with high precision. A careful analysis of this simulation approach reveals that to sufficiently suppress the errors introduced in the simulation process, the simulation methods can indeed become too costly and destroy the originally claimed quantum speedups.

[1] Nagaj et al. give a simpler way to prove this relationship using Jordan's lemma in [9].

We have an answer to this problem. The key component of our quantum update is based on Grover and Rudolph's method of preparing states corresponding to efficiently integrable probability distributions [15]. In our case, the quantum samples we need to prepare correspond to probability distributions that are supported on at most d states of Ω, which implies that they are efficiently integrable. Thus, we can use the method [15] to obtain an efficient circuit for the quantum update. Note that Childs [16], investigating the relationship between continuous-time [17] and discrete-time [18] quantum walks has also recently utilized this approach [15] for a different task – simulating sparse Hamiltonians on a star-like graph. Also, the basic trick underlying Grover and Rudolph's method, preparing superpositions by subsequent rotations, was first proposed by Zalka [19].

This is our main result about the quantum update rule U, the essential ingredient in the implementation of the quantum walk defined as the quantum analogue of the original Markov chain:

Theorem 1 (An Efficient Quantum Update Rule)**.** *Consider a reversible Markov chain on the state space Ω, with $|\Omega| = 2^m$, with a transition matrix $P = (p_{xy})_{x,y\in\Omega}$. Assume that*

1. *there are at most d possible transitions from each state (P is sparse),*
2. *the transition probabilities p_{xy} are given with t-bit precision, with $t = \Omega\left(\log\frac{1}{\epsilon} + \log d\right)$,*
3. *we have access to a reversible circuit returning the list of (at most d) neighbors of the state x (according to P), which can be turned into an efficient quantum circuit N:*

$$N\left|x\right\rangle\left|0\right\rangle\cdots\left|0\right\rangle = \left|x\right\rangle\left|y_0^x\right\rangle\cdots\left|y_{d-1}^x\right\rangle, \tag{6}$$

4. *we have access to a reversible circuit which can be turned into an efficient quantum circuit T acting as*

$$T\left|x\right\rangle\left|0\right\rangle\cdots\left|0\right\rangle = \left|x\right\rangle|p_{xy_0^x}\rangle\cdots|p_{xy_{d-1}^x}\rangle. \tag{7}$$

Then there exists an efficient quantum circuit $\tilde{U}$ simulating the quantum update rule

$$U\left|x\right\rangle\left|0\right\rangle = \left|x\right\rangle\sum_y \sqrt{p_{xy}}\left|y\right\rangle, \tag{8}$$

where the sum over y is over the neighbors of x, and p_{xy} are the elements of P, with precision

$$\left\|\left(U-\tilde{U}\right)\left|x\right\rangle\otimes\left|0\right\rangle\right\| \leq \epsilon \tag{9}$$

for all $x \in \Omega$, with required resources scaling linearly in m, polynomially in $\log\frac{1}{\epsilon}$ and linearly in d (with an additional poly$(\log d)$ factor).

The paper is organized as follows. In Section 2, we describe the alternative approaches one could take to implement the quantum update and discuss their efficiency. In Section 3 we present our algorithm based on Grover & Rudolph's state preparation method. We conclude in Section 4 and present the remaining details for the quantum update circuit, its required resources, and its implementation in Appendix A.

2 Alternative Ways of Implementing the Quantum Update

Before we give our efficient method, we review the alternative approaches in more detail, discussing their shortcomings. We know of three other ways how one could think of implementing the quantum update. The first two are based on techniques for simulating Hamiltonian time evolutions, while the third uses a novel technique for implementing combinatorially block-diagonal unitaries.

The first method is to directly realize the reflection $\mathrm{Ref}_{\mathcal{A}}$ as $\exp(-i\Pi_{\mathcal{A}}\tau)$ for time $\tau = \frac{\pi}{2}$, where the projector $\Pi_{\mathcal{A}}$ onto the subspace $\mathcal{A}$ turns out to be a sparse Hamiltonian. Observe that the projector

$$\Pi_{\mathcal{A}} = \sum_{x\in\Omega} |x\rangle\langle x| \otimes \sum_{y,y'\in\Omega} \sqrt{p_{xy}}\sqrt{p_{xy'}}|y\rangle\langle y'|$$

is a sparse Hamiltonian provided that P is sparse. Thus, we can approximately implement the reflection $\mathrm{Ref}_{\mathcal{A}}$ by simulating the time evolution according to $H = \Pi_{\mathcal{A}}$ for the time $\tau = \frac{\pi}{2}$. The same methods apply to the reflection $\mathrm{Ref}_{\mathcal{B}}$, so we can approximately implement the quantum walk $W(P)$, which is a product of these two reflections. The requirements of this method scale *polynomially* in $\frac{1}{\epsilon}$, where ϵ is the desired accuracy of the unitary quantum update.

The second approach is to apply novel general techniques for implementing arbitrary row-and-column-sparse unitaries, due to Childs [20] and Jordan and Wocjan [21]. Similarly to the first method, it relies on simulating a sparse Hamiltonian for a particular time. However, the complexity of this method again scales *polynomially* in $\frac{1}{\epsilon}$.

The third alternative is to utilize techniques for implementing combinatorially block-diagonal unitary matrices. A (unitary) matrix M is called combinatorially block-diagonal if there exists a permutation matrix P (i.e., a unitary matrix with entries 0 and 1) such that

$$PMP^{-1} = \bigoplus_{b=1}^{B} M_b$$

and the sizes of the blocks M_b are bounded from above by some small d. The method works as follows: each $x \in \Omega$ can be represented by the pair $\{b(x), p(x)\}$, where $b(x)$ denotes the block number of x and $p(x)$ denotes the position of x inside the block $b(x)$. The unitary M can then be realized by

1. the basis change $|x\rangle \mapsto |b(x)\rangle \otimes |p(x)\rangle$,
2. the controlled operation $\sum_{b=1}^{B} |b\rangle\langle b| \otimes M_b$, and
3. the basis change $|b(x)\rangle \otimes |p(x)\rangle \mapsto |x\rangle$.

The transformations M_b can be implemented using $O(4^s)$ elementary gates based on the decomposition of unitaries into a product of two-level matrices [22]. The special case $d = 2$ is worked out in the paper by Aharonov and Ta-Shma [23]. The reflection $\mathrm{Ref}_{\mathcal{A}} = 2\Pi_{\mathcal{A}} - \mathbb{I}$ then has the form

$$\mathrm{Ref}_{\mathcal{A}} = \sum_{x\in\Omega} |x\rangle\langle x| \otimes \left(\sum_{y,y'\in\Omega} \sqrt{p_{xy}}\sqrt{p_{xy'}}|y\rangle\langle y'| - \delta_{y,y'} \right) ,$$

where $\delta_{y,y'} = 1$ for $y = y'$ and 0 otherwise. Viewed in this form, we see that $\mathrm{Ref}_{\mathcal{A}}$ is a combinatorially block-diagonal unitary matrix, with a block decomposition with respect to the 'macro' coordinate

x. Inside each 'macro' block labeled by x, we obtain a 'micro' block of size d corresponding to all y with $p_{xy} > 0$ and many 'micro' blocks of size 1 corresponding to all y with $p_{xy} = 0$ after a simple permutation of the rows and columns. The disadvantage of this way of implementing quantum walks is that its complexity scales *exponentially* with d, the maximum number of neighbors for each state x.

In the next Section, we show how to implement the quantum update rule by a circuit with the number of operations scaling *linearly* with the sparsity parameter d (with additional poly($\log d$) factors) and *polynomially* in $\log \frac{1}{\epsilon}$.

3 Overview of the Quantum Algorithm

Our efficient circuit for the Quantum Update Rule

$$U \left|x\right\rangle_L \left|0\right\rangle_R = \left|x\right\rangle_L \sum_{i=0}^{d-1} \sqrt{p_{xy_i^x}} \left|y_i^x\right\rangle_R \tag{10}$$

works in the following way:

1. Looking at x in the 'left' register, put a list of its (at most d) neighbors y_i^x into an extra register and the corresponding transition probabilities $p_{xy_i^x}$ into another extra register.

2. Using the list of probabilities, prepare the superposition

$$\sum_{i=0}^{d-1} \sqrt{p_{xy_i^x}} \left|i\right\rangle_S \tag{11}$$

   in an extra 'superposition' register S.

3. Using the list of neighbors, put $\sum_{i=0}^{d-1} \sqrt{p_{xy_i^x}} \left|y_i^x\right\rangle_R \left|i\right\rangle_S$ in the registers R and S.

4. Clean up the S register using the list of neighbors of x and uncompute the transition probability list and the neighbor list.

We already assumed we can implement Step 1 of this algorithm efficiently. The second, crucial step is described in Section 3.1. Additional details for steps 3 and 4 are spelled out in Appendix A. Finally, the cleanup step 4 is possible because of the unitarity of step 1.

3.1 Preparing Superpositions à la Grover and Rudolph

The main difficulty is the efficient preparation of (11). We start with a list of transition probabilities $\{p_{xy_i^x}, 0 \leq i \leq d-1\}$ with the normalization property $\sum_{i=0}^{d-1} p_{xy_i^x} = 1$. Our approach is an application of the powerful general procedure of [15]. The idea is to build the superposition up in $\log d$ rounds of doubling the number of terms in the superposition (see Figure 1). Each round involves one of the qubits in the register S, to which we apply a rotation depending on the state of qubits which we have already touched.

For simplicity, let us first assume all points x have exactly d neighbors and that all transition probabilities $p_{xy_i^x}$ are nonzero, and deal with the general case in Section 3.2. To clean up the

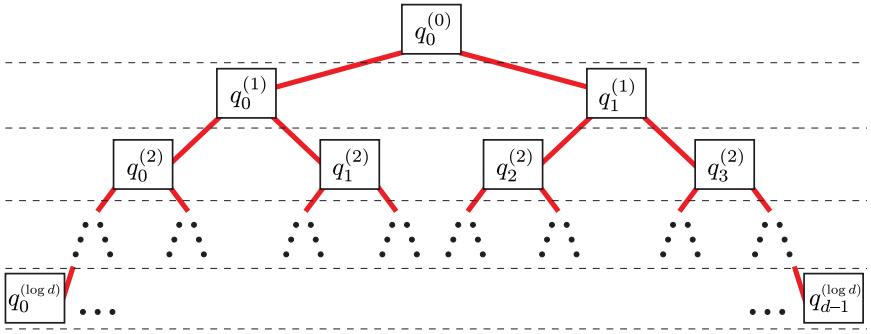

Figure 1: The scheme for preparing the superposition $\sum_{i=0}^{d-1} \sqrt{q_i^{(\log d)}}\,|i\rangle$ in $\log d$ rounds.

notation, denote $q_i = p_{xy_i^x}$. Working up from the last row in Figure 1 where $q_i^{(\log d)} = q_i$, we first compute the $d-1$ numbers $q_i^{(k)}$ for $i = 0, \ldots, 2^k - 1$ and $k = 0, \ldots, (\log d) - 1$ from

$$q_i^{(k-1)} = q_{2i}^{(k)} + q_{2i+1}^{(k)}. \tag{12}$$

The transition probabilities sum to 1, so we end with $q_0^{(0)} = 1$ at the top.

Our goal is to prepare $|\psi_{\log d}\rangle = \sum_{i=0}^{d-1} \sqrt{q_i}\,|i\rangle$. We start with $\log d$ qubits in the state

$$|\psi_0\rangle = |0\rangle_1 |0\rangle_2 \cdots |0\rangle_{\log d}\,. \tag{13}$$

In the first round we prepare

$$|\psi_1\rangle = \left(\sqrt{q_0^{(1)}}\,|0\rangle_1 + \sqrt{q_1^{(1)}}\,|1\rangle_1\right) |0\rangle_2 \cdots |0\rangle_{\log d} \tag{14}$$

by applying a rotation to the first qubit. A rotation

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{15}$$

by $\theta_0^{(1)} = \cos^{-1}\sqrt{q_0^{(1)}}$ does this job. In the second round, we apply a rotation to the second qubit. However, the amount of rotation now has to depend on the state of the first qubit. When the first qubit is $|0\rangle$, we apply a rotation by

$$\theta_0^{(2)} = \cos^{-1}\sqrt{\frac{q_0^{(2)}}{q_0^{(1)}}}, \tag{16}$$

Analogously, when the first qubit is $|1\rangle$, we choose

$$\theta_1^{(2)} = \cos^{-1}\sqrt{\frac{q_2^{(2)}}{q_1^{(1)}}}. \tag{17}$$

Observe that the second round turns (14) into

$$|\psi_2\rangle = \left(\sqrt{q_0^{(2)}}\,|00\rangle_{1,2} + \sqrt{q_1^{(2)}}\,|01\rangle_{1,2} + \sqrt{q_2^{(2)}}\,|10\rangle_{1,2} + \sqrt{q_3^{(2)}}\,|11\rangle_{1,2}\right)|0\rangle_3 \cdots |0\rangle_{\log d}\,. \tag{18}$$

Let us generalize this procedure. Before the j-th round, the qubits j and higher are still in the state $|0\rangle$, while the first $j-1$ qubits tell us where in the tree (see Figure 1) we are. In round j, we thus need to rotate the j-th qubit by

$$\theta_i^{(j)} = \cos^{-1}\sqrt{\frac{q_{2i}^{(j)}}{q_i^{(j-1)}}}, \tag{19}$$

depending on the state $|i\rangle$ which is encoded in binary in the first $j-1$ qubits of the 'superposition' register S.

Applying $\log d$ rounds of this procedure results in preparing the desired superposition (11), with the states $|i\rangle$ encoded in binary in the $\log d$ qubits.

3.2 A nonuniform case

In Section 3.1, we assumed each x had exactly d neighbors it could transition to. To deal with having fewer neighbors (and zero transition probabilities), we only need to add an extra 'flag' register F_i for each of the d neighbors y_i^x in the neighbor list. This 'flag' will be 0 if the transition probability $p_{xy_i^x}$ is zero. Conditioning the operations in steps 2-4 of our algorithm (see Section 3) on these 'flag' registers will deal with the nonuniform case as well.

3.3 Precision requirements

We assumed that each of the probabilities $p_{xy_i^x}$ was given with t-bit precision. Our goal was to produce a quantum sample (11) whose amplitudes would be precise to t bits as well. Let us investigate how much precision we need in our circuit to achieve this.

For any x, the imperfections in $q_i^{\log d} = p_{xy_i^x}$ (see Section 3.1) come from the $\log d$ rotations by imperfectly calculated angles θ. The argument of the inverse cosine in (19)

$$a_i^{(j)} = \sqrt{\frac{q_{2i}^{(j)}}{q_i^{(j-1)}}} \tag{20}$$

obeys $0 \le a_i^{(j)} \le 1$. The errors in the rotations are the largest for $a_i^{(j)}$ close to 0 or 1 (i.e. when the θ's are close to $\frac{\pi}{2}$ or 0). To get a better handle on these errors, we introduce extra flag qubits signaling $a_i^{(j)} = 0$ or $a_i^{(j)} = 1$ (see the Appendix for details). In these two special cases, the rotation by θ becomes an identity or a simple bit flip. On the other hand, because the q's are given with t bits, for a's bounded away from 0 and 1, we have

$$\sqrt{\frac{2^{-t}}{1}} \le a \le \sqrt{\frac{1-2^{-t}}{1}}. \tag{21}$$

We choose to use an n-bit precision circuit for computing the a's, guaranteeing that $|\tilde{a} - a| \leq 2^{-n}$. Using the Taylor expansion, we bound the errors on the angles θ:

$$|\tilde{\theta} - \theta| = |\cos^{-1}\tilde{a} - \cos^{-1} a| = \left|(\tilde{a} - a)\frac{d\cos^{-1} a}{da} + \ldots\right| \leq c_1 \frac{2^{-n}}{\sqrt{1-a^2}} \leq c_1 2^{-n+\frac{t}{2}}, \qquad (22)$$

because a is bounded away from 1 as (21).

Each amplitude in (11) comes from multiplying out $\log d$ terms of the form $\cos\theta_i^j$ or $\sin\theta_i^j$. For our range of θ's, the error in each sine or cosine is upper bounded by

$$|\sin\tilde{\theta} - \sin\theta| \leq |\tilde{\theta} - \theta|, \qquad |\cos\tilde{\theta} - \cos\theta| \leq |\tilde{\theta} - \theta|. \qquad (23)$$

Therefore, the final error in each final amplitude is upper bounded by

$$\Delta_i = \left|\sqrt{\tilde{q}_i} - \sqrt{q_i}\right| \leq c_1 (\log d) 2^{-n+\frac{t}{2}}. \qquad (24)$$

Note that the factor $\log d$ is small. Therefore, to ensure t-bit precision for the final amplitudes, it is enough to work with $n = \frac{3}{2}t + \Omega(1)$ bits of precision during the computation of the θ's. We conclude that our circuit can be implemented efficiently and keep the required precision.

4 Conclusion

The problem of constructing *explicit* quantum circuits for implementing *arbitrary* quantum walks has not been considered in detail in the literature so far. It has been known that quantum walks could be implemented using techniques for simulating Hamiltonian time evolutions. However, the complexity would grow *polynomially* in $1/\epsilon$ if we were to rely on simulating Hamiltonian dynamics (see Section 2). This would be fatal for the quantum algorithm for estimating partition functions in [6] since the quantum speed-up over its classical counterparts would be lost. An alternative for implementing quantum walks whose running complexity scales logarithmically in $1/\epsilon$ exists, relying on the implementation of combinatorially block-diagonal unitaries. Its disadvantage is its running time, growing *exponentially* in d (see Section 2).

We showed how to efficiently implement Szegedy's quantum walks $W(P)$ that are derived from arbitrary classical sparse walks $P = (p_{xy})_{x,y\in\Omega}$. We constructed a quantum circuit $\tilde{U}$ that approximately implements the quantum update rule (8) with circuit complexity scaling *linearly* (with additional logarithmic factors) in d, the degree of sparseness of P, *linearly* in $m = \log|\Omega|$ and *polynomially* in $\log\frac{1}{\epsilon}$, where ϵ denotes the desired approximation accuracy (9).

5 Acknowledgments

We would like to thank Rolando Somma, Sergio Boixo and Robert R. Tucci for starting the debate about the quantum update rules, and Stephen Jordan for helpful discussions on the alternative methods for implementing them. C. C. and P. W. gratefully acknowledge the support by NSF grants CCF-0726771 and CCF-0746600. D. N. gratefully acknowledges support by European Project QAP 2004-IST-FETPI-15848 and by the Slovak Research and Development Agency under the contract No. APVV-0673-07. Part of this work was done while D. N. was visiting University of Central Florida.

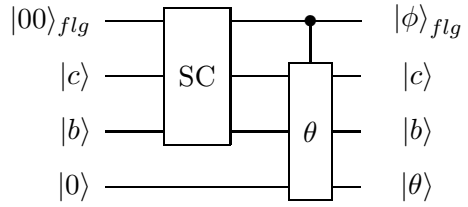

Figure 2: The Determine Angle Circuit DAC.

A Additional Details for the Efficient Quantum Update Circuit

In this Appendix, we spell out additional details for our Quantum Update circuit as well as draw the circuit out for a $d = 4$.

The state space of the classical Markov chain P is Ω, with $|\Omega| = 2^m$. The entries of P are p_{xy}, the transition probabilities from state x to state y. We assume that P is sparse, i.e. that for each $x \in \Omega$ there are at most d neighbors y_i^x such that $p_{xy_i^x} > 0$, and their number is small, i.e. $d \ll 2^m$. Since d is a constant, we can assume without loss of generality that $d = 2^r$. We want to implement the quantum (8), where $|x\rangle \in \mathbb{C}^{2m}$.

A.1 Preparation

Classically, our knowledge of P can be encoded into efficient reversible circuits outputting the neighbors and transition probabilities for the point x. We will use quantum versions N and T of these circuits, with the following properties. The neighbor circuit N acts on d copies of $\mathbb{C}^{|\Omega|}$ and produces a list of neighbors of x as

$$N\,|x\rangle_L\,|0\rangle^{\otimes d} = |x\rangle_L \otimes |y_0^x\rangle \cdots \left|y_{d-1}^x\right\rangle. \tag{25}$$

All the transition probabilities $p_{xy_i^x}$ are given with t-bit precision. The transition probability circuit T acts on a register holding a state $|x\rangle$ and d copies of $\left(\mathbb{C}^2\right)^{\otimes t}$, producing a list of transition probabilities for neighbors of $|x\rangle$ as

$$T\,|x\rangle_L\,|0\rangle^{\otimes d} = |x\rangle_L \otimes |p_{xy_0^x}\rangle \cdots |p_{xy_{d-1}^x}\rangle. \tag{26}$$

To simplify the notation, let us label $q_i = p_{xy_i^x}$. We now prepare all the terms $q_i^{(k)}$, filling the tree in Figure (1). Starting from $q_i^{(\log d)} = q_i$, we use an adding circuit (ADD) doing the operation $q_i^{(k-1)} = q_{2i}^{(k)} + q_{2i+1}^{(k)}$. The probability distribution $\{q_i\}$ is efficiently integrable, so filling the tree of $q_i^{(k)}$ is easy, and we can use Grover and Rudolph's method [15] of preparing quantum samples for such probability distributions.

A.2 Determining the rotation angles

After the preparation described in the previous Section, we need to compute the appropriate rotation angles $\tilde{\theta}_i^{(k)}$ for Grover and Rudolph's method. For this, we use the Determine Angle Circuit

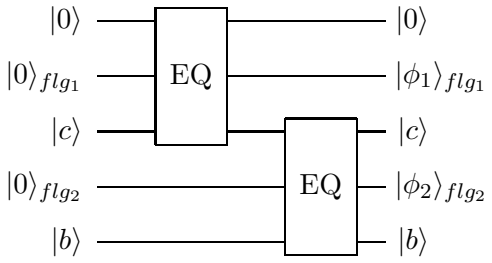

Figure 3: The circuit SC handling special cases.

(DAC). This circuit produces

$$\theta_i^{(k)} = \cos^{-1}\sqrt{\frac{q_{2i}^{(k)}}{q_i^{(k-1)}}}, \tag{27}$$

while also handling the special cases $q_{2i}^{(k)} = q_i^{(k-1)}$ and $q_i^{(k-1)} = 0$. For simplicity, let us label $b = q_i^{(k-1)}$, $c = q_{2i}^{(k)}$. The DAC circuit first checks the special cases, and then, conditioned on the state of the two two flag qubits, computes (27). We draw it in Figure 2, with the special case-analysing circuit SC given in Figure A.2. Here EQ is a subroutine testing whether two qubits (in computational basis states) are the same. The first EQ tests the states $|0\rangle$ and $|c\rangle$, while the second EQ runs the test on $|c\rangle$ and $|b\rangle$. We have the following four scenarios depending on the flag qubits

00	the circuit θ computes normally ,	
$01, 11$	the circuit θ does nothing (keeps angle $= 0$, as $b = c$) ,	(28)
10	the circuit θ outputs $\theta = \pi/2$, as $c = 0$.	

The third option corresponds to $c = 0$, when all the probability in the next layer of the tree is concentrated in the right branch. We then simply flip the superposition qubit, using $\theta = \frac{\pi}{2}$.

A.3 Creating superpositions and mapping

After the angle is determined, we apply the corresponding rotation to the appropriate qubit in the superposition register S, as described in Section 3.1. We then uncompute the rotation angle.

Once the final superposition is created in S, we invoke a mapping circuit M. This M acts on the register holding the names of the d neighbors of x, the superposition register, and the output register R. It takes y_j^x, the name of the j-th neighbor of x, and puts it into the output register as

$$M|0\rangle_R \otimes |y_0^x\rangle \otimes \ldots \otimes |y_{d-1}^x\rangle \otimes |j\rangle_S = |y_j^x\rangle_R \otimes |y_0^x\rangle \otimes \ldots \otimes |y_{d-1}^x\rangle \otimes |j\rangle_S. \tag{29}$$

We can do this, because the names of the states in Ω are given as computational basis states. The next step is to uncompute the label j in the last register with a cleaning circuit C as

$$C|y_i^x\rangle_R \otimes |y_0^x\rangle \otimes \ldots \otimes |y_{d-1}^x\rangle \otimes |j\rangle = \begin{cases} |y_i^x\rangle_R \otimes |y_0^x\rangle \otimes \ldots \otimes |y_{d-1}^x\rangle \otimes |j\rangle & \text{if } i \neq j \\ |y_i^x\rangle_R \otimes |y_0^x\rangle \otimes \ldots \otimes |y_{d-1}^x\rangle \otimes |0\rangle & \text{if } i = j. \end{cases} \tag{30}$$

Table 1: Required numbers of qubits

Register Type	Required number of qubits
x (register L)	m
y (register R)	m
y_i^x (neighbor list)	$d \times m$
q_i's (probabilities)	$(2d-2) \times t$
flag qubits	2
θ (rotation angle)	$n = \frac{3t}{2} + \Omega(1)$
ancillae for computing θ	$a_\theta = \mathrm{poly}(n) = \mathrm{poly}(t)$
superposition register S	$r = \log d$

These two steps transferred the superposition from the register S (with $r = \log d$ qubits), into the output register R (which has m qubits). The final step of our procedure is to uncompute (clean up) the lists of neighbors and transition probabilities.

A.4 The required resources

Let us count the number of qubits and operations required for our quantum update rule U based on a d-sparse stochastic transition matrix P. The number of ancillae required is $\Omega(dm+dt)$, where 2^m is the size of the state space and t is the required precision of the transition probabilities. Moreover, the required number of operations scales like $\Omega(d\, r\, m\, a_\theta)$, where $r = \log d$ and a_θ is the number of operations required to compute the angle θ with $n = \Omega(t)$-bit precision. Finally, when we have t-bit precision of the final amplitudes, the precision of the unitary we applied is

$$\left\| \left(U - \tilde{U} \right) |x\rangle \otimes |0\rangle \right\| \leq \epsilon, \tag{31}$$

for any $x \in \Omega$ when $t = \Omega\left(\log d + \log \frac{1}{\epsilon}\right)$. The total number of operations in our circuit thus scales like

$$\Omega\left(md\, \mathrm{poly}\left(\log d\right) + md \left(\log d\right) \mathrm{poly}\left(\log \frac{1}{\epsilon} \right) \right). \tag{32}$$

Besides the registers for the input $|x\rangle_L$ and output $|0\rangle_R$, we need d registers (with m qubits) to hold the names of the neighbors of x, and $2d-2$ registers (with t qubits) to store the transition probabilities q_i. The DAC circuit requires two extra flag qubits and a register with $n = \frac{3t}{2} + \Omega(1)$ qubits to store the angle θ. Computing the angle θ requires a circuit with $poly(n)$ qubits. Finally, the superposition register S holds r qubits. These requirements are summed in Table 1.

To conclude, we draw out the superposition-creating part of the quantum update for $d = 4$ in Figure 4. The first two lines represent the superposition register S, in which we prepare

$$|\varphi\rangle \;=\; \sqrt{q_0^{(2)}}|00\rangle + \sqrt{q_1^{(2)}}|01\rangle + \sqrt{q_2^{(2)}}|10\rangle + \sqrt{q_3^{(2)}}|11\rangle = \sum_{i=0}^{3} \sqrt{q_i}\,|i\rangle\,. \tag{33}$$

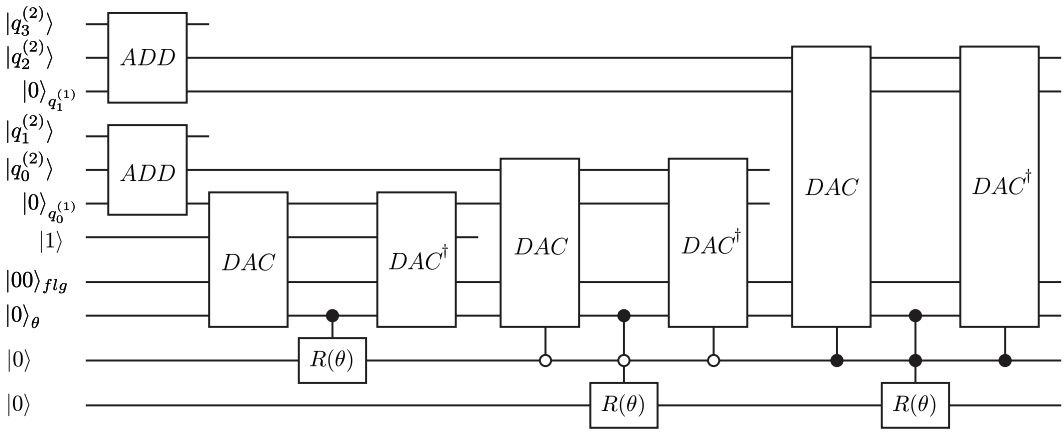

Figure 4: The efficient Quantum Update, creating the superposition (11) for $d = 4$. The bottom two lines represent the 'superposition' register S.

References

[1] S. Kirkpatrick, C. Gelatt and M. Vecchi, *Optimization by Simulated Annealing*, Science, New Series, vol. 220, No.4598, pp. 671–680 (1983).

[2] V. Černý, *A thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm*, Journal of Optimization Theory and Applications, vol. 45 , pp. 41–51 (1985).

[3] L. Lovász and S. Vempala, *Simulated Annealing in Convex Bodies and an $O^*(n^4)$ Volume Algorithm*, Journal of Computer and System Sciences, vol. 72, issue 2, pp. 392–417 (2006).

[4] M. Jerrum, A. Sinclair, and E. Vigoda, *A Polynomial-Time Approximation Algorithm for the Permanent of a Matrix Non-Negative Entries*, Journal of the ACM, vol. 51, issue 4, pp. 671–697 (2004).

[5] I. Bezáková, D. Štefankovič, V. Vazirani and E. Vigoda, *Accelerating Simulated Annealing for the Permanent and Combinatorial Counting Problems*, SIAM J. Comput., vol. 37, No. 5, pp. 1429–1454 (2008).

[6] P. Wocjan, C. Chiang, A. Abeyesinghe and D. Nagaj, *Quantum Speed-up for Approximating Partition Functions*, arXiv:0811.0596 (2008).

[7] Ch. M. Grinstead, J. L. Snell, *Introduction to Probability*, American Mathematical Society, (1997).

[8] M. Szegedy, *Quantum Speed-up of Markov Chain Based Algorithms*, Proc. of 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 32–41 (2004).

[9] D. Nagaj, P. Wocjan, Y. Zhang, *Fast QMA Amplification*, in preparation (2009).

[10] F. Magniez, A. Nayak, J. Roland, and M. Santha, *Search via Quantum Walk*, Proc. of the 39th Annual ACM Symposium on Theory of Computing, pp. 575–584 (2007).

[11] R. Somma, S. Boixo, H. Barnum, E. Knill, *Quantum Simulations of Classical Annealing Processes*, arXiv:0804.1571 (2008).

[12] P. Richter, *Quantum Speed-Up of Classical Mixing Processes*, Physical Review A, vol. 76, 042306 (2007).

[13] P. Wocjan and A. Abeyesinghe, *Speed-up via Quantum Sampling*, Phys. Rev. A, vol. 78, pp. 042336 (2008).

[14] D. Berry, G. Ahokas, R. Cleve and B. Sanders, *Efficient Quantum Algorithms for Simulating Sparse Hamiltonians*, Communications in Mathematical Physics, vol. 270, pp. 359–371 (2007).

[15] L. Grover, T. Rudolph, *Creating Superpositions that Correspond to Efficiently Integrable Probability Distributions*, arXiv:quant-ph/0208112 (2002).

[16] A. Childs, *On the Relationship between Continuous- and Discrete-time Quantum Walk*, arXiv:0810.0312 (2008).

[17] E. Farhi, S. Gutmann, *Quantum computation and decision trees*, Phys. Rev. A, 58:915-928 (1998).

[18] J. Kempe, *Quantum random walk algorithms*, Contemp. Phys., 44, 302-327 (2003).

[19] C. Zalka, *Efficient Simulation of Quantum Systems by Quantum Computers*, Proc. Roy. Soc. Lond. A 454, pp. 313–322 (1998).

[20] A. Childs, *Personal Communication*, March 2009.

[21] S. Jordan, P. Wocjan, *in preparation* (2009).

[22] M. A. Nielsen, I. L. Chuang, *Quantum Information and Computation*, Cambridge University Press, Cambridge, UK (2000).

[23] D. Aharonov and A. Ta-Shma, *Adiabatic Quantum State Generation and Statistical Zero Knowledge*, Proc. of the 35th annual ACM symposium on Theory, pp. 20–29 (2003).