Quantum Information and Computation, Vol. 10, No. 5&6 (2010) 0420–0434

EFFICIENT CIRCUITS FOR QUANTUM WALKS

CHEN-FU CHIANG[a]
School of Electric Engineering and Computer Science, University of Central Florida
4000 Central Florida Boulevard, Orlando, Florida 32816, USA

DANIEL NAGAJ[b]
Research Center for Quantum Information, Institute of Physics, Slovak Academy of Sciences
Dúbravská cesta 9, 84215, Bratislava, Slovakia

PAWEL WOCJAN[c]
School of Electric Engineering and Computer Science, University of Central Florida
4000 Central Florida Boulevard, Orlando, Florida, 32816, USA

Received April 18, 2009
Revised January 5, 2010

We present an efficient general method for realizing a quantum walk operator corresponding to an arbitrary sparse classical random walk. Our approach is based on Grover and Rudolph's method for preparing coherent versions of efficiently integrable probability distributions [1]. This method is intended for use in quantum walk algorithms with polynomial speedups, whose complexity is usually measured in terms of how many times we have to apply a step of a quantum walk [2], compared to the number of necessary classical Markov chain steps. We consider a finer notion of complexity including the number of elementary gates it takes to implement each step of the quantum walk with some desired accuracy. The difference in complexity for various implementation approaches is that our method scales linearly in the sparsity parameter and poly-logarithmically with the inverse of the desired precision. The best previously known general methods either scale quadratically in the sparsity parameter, or polynomially in the inverse precision. Our approach is especially relevant for implementing quantum walks corresponding to classical random walks like those used in the classical algorithms for approximating permanents [3, 4] and sampling from binary contingency tables [5]. In those algorithms, the sparsity parameter grows with the problem size, while maintaining high precision is required.

Keywords: Quantum Walks, Quantum Speedup, Circuit, Markov Chains
Communicated by: R Jozsa & M Mosca

1 Introduction

For many tasks, such as simulated annealing [6, 7], computing the volume of convex bodies [8] and approximating the permanent of a matrix [3, 4] (see references in [9] for more), the best approaches known today are randomized algorithms based on Markov chains (random walks) and sampling. A Markov chain on a state space $\mathcal{E}$ is described by a stochastic matrix

[a]email: cchiang@eecs.ucf.edu
[b]email: daniel.nagaj@savba.sk
[c]email: wocjan@eecs.ucf.edu

$P = (p_{xy})_{x,y\in\mathcal{E}}$. 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. [10]), then there is a unique probability distribution $\pi = (\pi_x)_{x\in\mathcal{E}}$ 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 [2], 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 system with two registers $\mathcal{H} = \mathcal{H}_L \otimes \mathcal{H}_R$. The *quantum update rule*, defined in [11], is any unitary that acts as

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

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

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

is the span of all vectors we get from acting with U on $\left|x\right\rangle_L\left|0\right\rangle_R$ for all $x \in \mathcal{E}$, 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

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

Szegedy defined a step of the quantum walk as

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

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

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

where π is the stationary distribution of P, is an eigenvector of W with eigenvalue 1. Szegedy [2] proved[d] 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.

[d] Nagaj et al. give a simpler way to prove this relationship using Jordan's lemma in [21].

Many of the recent quantum walk algorithms for searching [11, 12, 13, 14], evaluating formulas and span programs [15, 16, 17], quantum simulated annealing [18], quantum sampling [19, 20] and approximating partition functions based on classical Markov chains [9] can be viewed in Szegedy's generalized quantum walk model. 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 with low-degree underlying graphs, 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. The standard polynomial speed-ups of these quantum algorithms are viewed in terms of how many times we have to apply the quantum walk operator versus the number of times we have to apply one step of the classical random walk (Markov chain). However, a finer notion of complexity including the number of elementary gates it takes to implement each step of the quantum walk is needed here. Our work addresses the question whether it is possible to apply the steps of these quantum walk-based algorithms efficiently enough so as not to destroy the polynomial speedups.

In Section 2, we review the recent alternative approaches to the implementation of U, such as those relying on efficient simulation of sparse Hamiltonians [22]. We find that they either scale quadratically in the sparsity parameter d, or polynomially in $\frac{1}{\epsilon}$, where ϵ is the allowed error in the implementation of U. When there is only a small number of neighbors connected to each state x, or we do not need to use many steps of the quantum walk so that we can tolerate more implementation error, one could use these methods. However, the subtle algorithms like [9] require many precise uses of U which couple many (a number growing with the system size) neighboring states. In Appendix 1 we show a particular example (a first step towards a possible future quantum version of the classical algorithm for approximating the permanent [3, 4]), where the alternative approaches to U destroy the polynomial speedup of the quantum algorithm. This is why we developed our new method, scaling linearly in the sparsity parameter d and polynomially in $\log \frac{1}{\epsilon}$.

Our general approach to the implementation of quantum walks based on sparse classical Markov chains is based on Grover and Rudolph's method of preparing states corresponding to efficiently integrable probability distributions [1]. In our case, the quantum samples we need to prepare correspond to probability distributions that are supported on at most d states of $\mathcal{E}$, which implies that they are efficiently integrable. Thus, we can use the method [1] to obtain an efficient circuit for the quantum update. The basic trick underlying Grover and Rudolph's method, preparing superpositions by subsequent rotations, was first proposed by Zalka [23]. Note that Childs [24], investigating the relationship between continuous-time [25] and discrete-time [26] quantum walks, also proposed to use [1], also for some quantum walks with non-sparse underlying graphs.

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 $\mathcal{E}$, with $|\mathcal{E}| = 2^m$, with a transition matrix $P = (p_{xy})_{x,y\in\mathcal{E}}$. Assume that

- there are at most d possible transitions from each state (P is sparse),
- the transition probabilities p_{xy} are given with t-bit precision, with $t = \Omega\left(\log \frac{1}{\epsilon} + \log d\right)$,
- 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:

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

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

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

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

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

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

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

for all $x \in \mathcal{E}$, 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 our discussion in Section 4. In Appendix 1, we give an example where our approach is better than the alternative methods, and finally, we present the remaining details for the quantum update circuit, its required resources, and its implementation in Appendix B.

2 Alternative Ways of Implementing the Quantum Update

Before we give our efficient method, we review the alternative approaches in more detail. 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

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

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. Moreover, the number of gates used in each U scales at least linearly with d and m.

The second approach is to apply novel general techniques for implementing arbitrary row-and-column-sparse unitaries, due to Childs [27] and Jordan and Wocjan [28]. 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 $\frac{1}{\epsilon}$ (and linearly in d and m).

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 $x \in \mathcal{E}$ 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

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

The transformations M_b can be implemented using $O(d^2)$ elementary gates based on the decomposition of unitaries into a product of two-level matrices [29]. The special case $d = 2$ is worked out in the paper by Aharonov and Ta-Shma [30]. The reflection $\mathrm{Ref}_{\mathcal{A}} = 2\Pi_{\mathcal{A}} - \mathbb{I}$ then has the form

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

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 *quadratically* with d (and linearly in m and $\log \frac{1}{\epsilon}$), 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 $\mathrm{poly}(\log d)$ factors), linearly in $m = \log |\mathcal{E}|$ and *polynomially* in $\log \frac{1}{\epsilon}$.

3 Overview of the Quantum Algorithm

Our efficient circuit for the Quantum Update Rule

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

works in the following way:

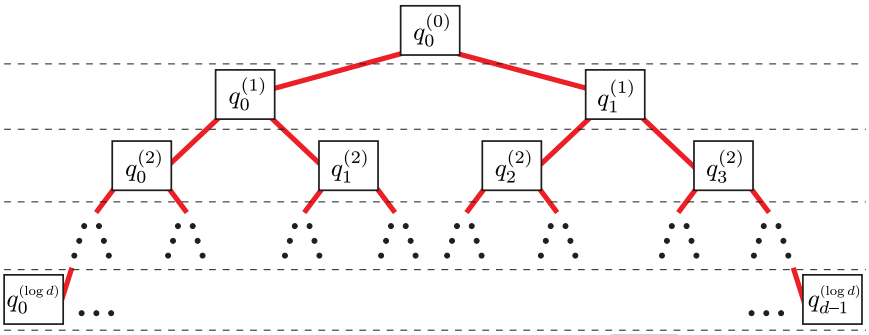

Fig. 1. The scheme for preparing the superposition $\sum_{i=0}^{d-1} \sqrt{q_i^{(\log d)}}\,|i\rangle$ in $\log d$ rounds.

(i) 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.
(ii) Using the list of probabilities, prepare the superposition

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

in an extra 'superposition' register S.
(iii) Using the list of neighbors, put $\sum_{i=0}^{d-1} \sqrt{p_{xy_i^x}}\,|y_i^x\rangle_R\,|i\rangle_S$ in the registers R and S.
(iv) 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 B. 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 $\{p_{xy_i^x}, 0 \le i \le d-1\}$ 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 [1]. 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 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 $0 \leq a_i^{(j)} \leq 1$. 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 Appendix B 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

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

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:

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

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

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

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

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

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* efficient quantum circuits for implementing *arbitrary* sparse quantum walks has not been considered in detail in the literature so far. We were interested in an efficient implementation of a step of a quantum walk and finding one with a favorable scaling of the number of required operations with d (the sparsity parameter) and the accuracy parameter $\frac{1}{\epsilon}$. Its intended use are algorithms based on quantum walks with polynomial speedups over their classical Markov Chain counterparts.

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

It has been known that quantum walks could be implemented using techniques for simulating Hamiltonian time evolutions. However, the complexity would grow *polynomially* in $\frac{1}{\epsilon}$ if we were to rely on simulating Hamiltonian dynamics (see Section 2). This would be fatal for quantum algorithms such as the one for estimating partition functions in [9] or future algorithms for approximating the permanent, losing the polynomial quantum speed-ups over their classical counterparts. An alternative for implementing quantum walks whose running complexity scales logarithmically in $\frac{1}{\epsilon}$ exists. It relies on the implementation of combinatorially block-diagonal unitaries. However, its running time grows *quadratically* in d (see Section 2). When the sparsity of the walk d grows with the system size n, this brings an extra factor of n to the complexity of the algorithms, destroying or decreasing its polynomial speedup. This is true e.g. for the example given in Appendix 1. Therefore, our approach to the quantum update is again more suitable for this task.

Acknowledgments

We would like to thank an anonymous referee for many insightful comments and questions. We also thank Rolando Somma, Sergio Boixo, Stephen Jordan and Robert R. Tucci for helpful discussions. 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 OP CE QUTE ITMS NFP 26240120009, by the Slovak Research and Development Agency under the contract No. APVV LPP-0430-09, and by the VEGA grant QWAEN 2/0092/09. Part of this work was done while D. N. was visiting University of Central Florida.

References

1. L. Grover, T. Rudolph (2002), *Creating Superpositions that Correspond to Efficiently Integrable Probability Distributions*, arXiv:quant-ph/0208112.
2. M. Szegedy (2004), *Quantum Speed-up of Markov Chain Based Algorithms*, Proc. of 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 32–41.
3. M. Jerrum, A. Sinclair, and E. Vigoda (2004), *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.
4. I. Bezáková, D. Štefankovič, V. Vazirani and E. Vigoda (2008), *Accelerating Simulated Annealing for the Permanent and Combinatorial Counting Problems*, SIAM J. Comput., vol. 37, No. 5, pp. 1429–1454.
5. I. Bezáková, A. Sinclair, D. Štefankovič and E. Vigoda (2006), *Negative Examples for Sequential Importance Sampling of Binary Contingency Tables*, Proc. of Algorithms – ESA 2006, Lecture Notes in Computer Science, Springer Berlin/Heidelberg, vol. 4168, pp. 0302-9743.
6. S. Kirkpatrick, C. Gelatt and M. Vecchi (1983), *Optimization by Simulated Annealing*, Science, New Series, vol. 220, No.4598, pp. 671–680.

[e] Of course, much more efficient approaches exist for specific walks (e.g. those on regular, constant-degree graphs).

7. V. Černý (1985), *A thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm*, Journal of Optimization Theory and Applications, vol. 45 , pp. 41–51.
8. L. Lovász and S. Vempala (2006), *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.
9. P. Wocjan, C. Chiang, D. Nagaj and A. Abeyesinghe (2009), *A Quantum Algorithm for Approximating Partition Functions*, Phys. Rev. A 80, 022340.
10. Ch. M. Grinstead, J. L. Snell (1997), *Introduction to Probability*, American Mathematical Society.
11. F. Magniez, A. Nayak, J. Roland, and M. Santha (2007), *Search via Quantum Walk*, Proc. of the 39th Annual ACM Symposium on Theory of Computing, pp. 575–584.
12. A. Ambainis (2004), *Search via Quantum Walk*, ACM SIGACT News, Vol. 35 (2), 22.
13. A. Ambainis (2004), *Quantum Walk Algorithm for Element Distinctness*, Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 22–31.
14. F. Magniez, M. Santha, M. Szegedy (2005), *Quantum algorithms for the triangle problem*, Proc. of the 16th ACM-SIAM symposium on Discrete algorithms, 1109.
15. B. W. Reichardt, R. Špalek (2008), *Span-program-based quantum algorithm for evaluating formulas*, Proceedings of the 40th STOC, 103.
16. A. Ambainis, A. Childs, B. Reichardt, R. Špalek and S. Zhang (2007), *Any AND-OR Formula of Size N Can Be Evaluated in Time $N^{1/2+o(1)}$ on A Quantum Computer*, Proc of 48th FOCS, pp. 363-372.
17. E. Farhi, S. Gutmann and S. Gutmann (2008), *A Quantum Algorithm for the Hamiltonian NAND Tree*, Theory of Computing, vol. 4, pp. 169–190.
18. R. Somma, S. Boixo, H. Barnum, E. Knill (2008), *Quantum Simulations of Classical Annealing Processes*, Phys. Rev. Lett. vol. 101, pp. 130504.
19. P. Richter (2007), *Quantum Speed-Up of Classical Mixing Processes*, Physical Review A, vol. 76, 042306.
20. P. Wocjan and A. Abeyesinghe (2008), *Speed-up via Quantum Sampling*, Phys. Rev. A, vol. 78, pp. 042336.
21. D. Nagaj, P. Wocjan, Y. Zhang (2009), *Fast QMA Amplification*, QIC vol. 9 no. 11&12 pp. 1053–1068.
22. D. Berry, G. Ahokas, R. Cleve and B. Sanders (2007), *Efficient Quantum Algorithms for Simulating Sparse Hamiltonians*, Communications in Mathematical Physics, vol. 270, pp. 359–371.
23. C. Zalka (1998), *Efficient Simulation of Quantum Systems by Quantum Computers*, Proc. Roy. Soc. Lond. A 454, pp. 313–322.
24. A. Childs (2009), *On the Relationship between Continuous- and Discrete-time Quantum Walk*, Communications in Mathematical Physics.
25. E. Farhi, S. Gutmann (1998), *Quantum Computation and Decision Trees*, Phys. Rev. A, 58:915-928.
26. J. Kempe (2003), *Quantum random walk algorithms*, Contemp. Phys., 44, 302-327.
27. A. Childs (March 2009), *Personal Communication.*
28. S. Jordan, P. Wocjan (2009), *Efficient Quantum Circuits for Arbitrary Sparse Unitaries*, arXiv:0904.2211.
29. M. Reck, A. Zeilinger, H. Bernstein, P. Bertani (1994), *Experimental Realization of Any Discrete Unitary Operator*, Phys. Rev. Lett. 73, 58.
30. D. Aharonov and A. Ta-Shma (2003), *Adiabatic Quantum State Generation and Statistical Zero Knowledge*, Proc. of the 35th annual ACM symposium on Theory, pp. 20–29.

Appendix A: Applications

A.1 Approximating the Permanent: Where Sparsity and Accuracy Matter

In this Appendix we present a particular example of a quantum algorithm with a polyno-

mial speedup over its classical counterpart, requiring our efficient approach to implementing quantum walks. The example is a rather naïve quantization of the classical algorithm for approximating the permanent of a matrix

$$per(A) = \sum_{\sigma} \prod_{i=1}^{n} a_{i,\sigma(i)}, \tag{A.1}$$

where σ runs all over the permutations of $[1, \ldots, n]$. For a 0/1 matrix A, the permanent of A is exactly the number of perfect matchings in the bipartite graph with bipartite adjacency matrix A. A classical FPRAS (fully polynomial randomized approximation scheme) [4] for this task involves taking $O^*\left(n^7\right)$ steps of a Markov chain (here O^* means up to logarithmic factors). It produces an approximation to the permanent within $[(1-\eta)\, per(A), (1+\eta)\, per(A)]$ by using

1. $\ell = O^*(n)$ stages of simulated annealing,
2. at each stage, generating $S = O^*\left(n^2\right)$ samples from a particular Markov chain,
3. $T = O^*\left(n^4\right)$ Markov chain invocations to generate a sample from its approximate steady state.

The failure probability of each stage is set to $\hat{\eta} = o\left(1/m^4\right)$ so that $\eta = \ell\hat{\eta}$ is small. Hence, the total complexity (number of Markov chain steps used) is $\ell ST = O^*\left(n^7\right)$.

The sparsity parameter d of the Markov chains involved scales with the problem size m. Therefore, the dependence of the implementation of the corresponding quantum walk on d becomes significant. Furthermore, because of the many stages of simulated annealing and sampling, the error ϵ in implementation of each quantum walk operator needs to smaller than one over the number of quantum walk steps involved.

The simplest quantized algorithm uses a quantum walk instead of the Markov Chain, and requires $O^*\left(n^5\right)$ steps of a quantum walk, as the mixing of the quantum walk requires only $\sqrt{T} = O^*\left(n^2\right)$ steps. However, it is important to choose an efficient circuit to implement each step of the quantum walk. A bad choice could destroy the speedup.

Let us compare what happens when this algorithms utilizes the different methods for quantum walk implementation as subroutines, counting the number of required elementary gates. Note that in this counting, all of the methods (classical and quantum) we will mention share a common factor m (the log of the state space size). However, the scaling in d (the sparsity parameter) and $\frac{1}{\epsilon}$ (precision) is what distinguishes them.

Let us look at the alternative approaches given in Section 2, and show that the small n^2 polynomial speedup is lost. The first two of these approaches scale with $\frac{1}{\epsilon}$. This brings an extra $\frac{1}{\epsilon} \propto \sqrt{T} \propto n^2$ factor to the complexity of the algorithm, destroying the speedup. The third alternative uses $O^*\left(d^2\right)$ elementary gates, adding an extra factor of $d^2 = n^2$, again destroying the speedup. On the other hand, our method uses only $O^*\left(d\right) = n$ gates (the scaling coming from precision requirements only adds logarithmic factors), and we thus retain some of the quantum advantage.

This example was just an illustration of a scenario, where our efficient implementation of a quantum walk (see Section 3) is necessary. However, we see its use in a future much better quantum algorithm for approximating the permanent, using not only quantum walks, but also quantizing the sampling/counting subroutine as in [9].

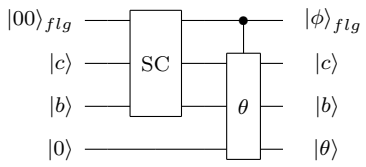

Fig. B.1. The Determine Angle Circuit DAC.

Appendix B: 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 $\mathcal{E}$, with $|\mathcal{E}| = 2^m$. 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 $x \in \mathcal{E}$ 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 $|x\rangle \in \mathbb{C}^{2m}$.

B.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 $\mathbb{C}^{|\mathcal{E}|}$ and produces a list of neighbors of x as

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

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

$$T\,|x\rangle_L\,|0\rangle^{\otimes d} = |x\rangle_L \otimes |p_{xy_0^x}\rangle \cdots |p_{xy_{d-1}^x}\rangle. \tag{B.2}$$

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 [1] of preparing quantum samples for such probability distributions.

B.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 (DAC). This circuit produces

$$\theta_i^{(k)} = \cos^{-1}\sqrt{\frac{q_{2i}^{(k)}}{q_i^{(k-1)}}}, \tag{B.3}$$

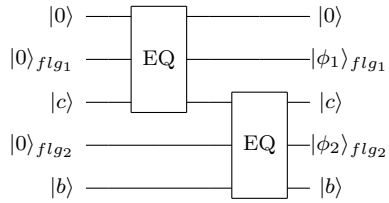

Fig. B.2. The circuit SC handling special cases.

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 (B.3). We draw it in Figure B.1, with the special case-analysing circuit SC given in Figure B.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

$$\begin{array}{ll} 00 & \text{the circuit } \theta \text{ computes normally ,} \\ 01, 11 & \text{the circuit } \theta \text{ does nothing (keeps angle} = 0, \text{ as } b = c) \text{ ,} \\ 10 & \text{the circuit } \theta \text{ outputs } \theta = \pi/2, \text{ as } c = 0. \end{array} \tag{B.4}$$

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 $\theta = \frac{\pi}{2}$.

B.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

$$M|0\rangle_R \otimes |y_0^x\rangle \otimes \ldots \otimes |y_{d-1}^x\rangle \otimes |j\rangle_S = |y_j^x\rangle_R \otimes |y_0^x\rangle \otimes \ldots \otimes |y_{d-1}^x\rangle \otimes |j\rangle_S. \tag{B.5}$$

We can do this, because the names of the states in $\mathcal{E}$ 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

$$C|y_i^x\rangle_R \otimes |y_0^x\rangle \otimes \ldots \otimes |y_{d-1}^x\rangle \otimes |j\rangle = \begin{cases} |y_i^x\rangle_R \otimes |y_0^x\rangle \otimes \ldots \otimes |y_{d-1}^x\rangle \otimes |j\rangle & \text{if } i \neq j \\ |y_i^x\rangle_R \otimes |y_0^x\rangle \otimes \ldots \otimes |y_{d-1}^x\rangle \otimes |0\rangle & \text{if } i = j. \end{cases} \tag{B.6}$$

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.

B.4 The required resources

Table B.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	$r = \log d$

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

$$\left\| \left(U - \tilde{U} \right) |x\rangle \otimes |0\rangle \right\| \leq \epsilon, \tag{B.7}$$

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

$$\Omega\left(md\, \mathrm{poly}\left(\log d\right) + md \left(\log d\right) \mathrm{poly}\left(\log \frac{1}{\epsilon} \right) \right). \tag{B.8}$$

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 B.1.

To conclude, we draw out the superposition-creating part of the quantum update for $d = 4$ in Figure B.3. The first two lines represent the superposition register S, in which we prepare

$$|\varphi\rangle \quad = \quad \sqrt{q_0^{(2)}}|00\rangle + \sqrt{q_1^{(2)}}|01\rangle + \sqrt{q_2^{(2)}}|10\rangle + \sqrt{q_3^{(2)}}|11\rangle = \sum_{i=0}^{3} \sqrt{q_i}\, |i\rangle\,. \tag{B.9}$$

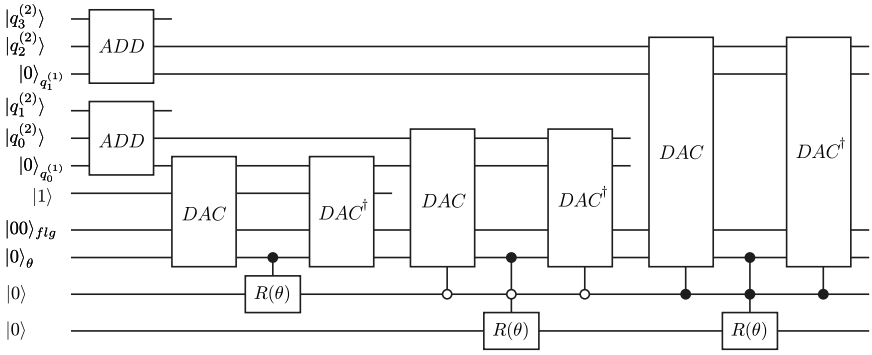

Fig. B.3. The efficient Quantum Update, creating the superposition (11) for $d = 4$. The bottom two lines represent the 'superposition' register S.