Quantum Walks with Memory on Groups

Michael Batty and Michael McGettrick

May 31, 2010

Abstract

We investigate the analogue of a recently studied quantum walk with memory in the context of a finite cyclic group. We show that something or other happens, and that it is very interesting (!!)

1 Introduction

McGettrick [3] has studied the one dimensional quantum Hadamard walk with memory given by the equations

$$\begin{aligned} |n-1,n,0\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|n,n-1,0\rangle + |n,n+1,1\rangle\right) \\ |n-1,n,1\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|n,n-1,0\rangle - |n,n+1,1\rangle\right) \end{aligned}$$

in the group $\mathbb{Z}$ and showed that the asyptotic probability distribution is somewhat different from the case without memory. Specifically, that localization occurs, which means that if the walk starts at 0 then for all $n \geqslant 0$, the probability that the walk is at the origin after n steps is at least 0.5. This result was verified and improved upon in [2] where the authors showed that the exact probability is $2-\sqrt{2} = 0.58578\ldots$.

Let G be a group with a specified generating set A. Then

- A (classical) random walk over G with respect to A is the assignation to $A^{\pm 1}$ of a probability distribution $p : A \mapsto [0,1]$, so that we have $g \mapsto ga^{\pm 1}$ with probability $p(a^{\pm 1})$. Thus the walk takes place in the Cayley graph of G with respect to A and if it is at any vertex g of the Cayley graph, it progresses along the edge labelled $a^{\pm 1}$ with probability $p(a^{\pm 1})$.

- Corresponding to the generating set A we take a *quantum coin space* spanned by A. A quantum walk without memory over G then takes place in the state space $\mathbb{C}G \otimes \mathbb{C}A$. In this case there is one G-register, and a base state is written as $|g,a\rangle$ where $g \in G$ and $a \in A$. Following [1] in the cyclic case, if A is finite, then define the *quantum Hadamard walk without memory* on G to be given for all $g \in G$ and $a \in A$ by

$$\begin{aligned} |g,a^{-1}\rangle &\mapsto \frac{1}{\sqrt{2}}(|ga^{-1},a^{-1}\rangle + |ga,a\rangle) \\ |g,a\rangle &\mapsto \frac{1}{\sqrt{2}}(|ga^{-1},a^{-1}\rangle - |ga,a\rangle). \end{aligned}$$

  Thus it is the composition of $\oplus_{i=1}^{|A|} W$ with the shift operator

$$|g,a^{\pm 1}\rangle \mapsto |ga^{\pm 1},a^{\pm 1}\rangle,$$

  Where W is the Walsh-Hadamard transformation.

MB: One could also potentially take the Fourier transform over any finite group with cardinality $2|A|$. Come to think of it, I am not sure what I have written down even *is* a Fourier transform. If the product involved was a tensor product rather than a direct product then it would be the Fourier transform over $\mathbb{Z}_2^n$. Maybe it is interesting to consider *any* unitary transformation of the coin space, like considering arbitrary a, b, c and d in the two-state case

It is known in the case $G = \mathbb{Z}_n$ for odd n that, irrespective of initial state, and for a Hadamard coin, that the asymptotic probabilities are uniform, i.e $1/n$ for each of the n points [1].

MMcG: What about even n (do we always get something periodic)? What about other non-Hadamard coins? What happens in Z_n when we add in memory?

- A quantum walk with memory over G is a certain unitary vector space automorphism of $\mathbb{C}(K \times A)$ where K is a certain subset of $G \times G$. In this case there are two G-registers and a base state is written as $|g_1, g_2, a^{\pm 1}\rangle$ where $g_1, g_2 \in G$ and $a \in A$. The walk corresponding to case (c) in [3] is given by

$$|ga^{-1}, g, a^{\pm 1}\rangle \mapsto \frac{1}{\sqrt{2}}\left(|g, ga^{-1}, a^{-1}\rangle \mp |g, ga, a^{+1}\rangle\right)$$
$$|ga, g, a^{\pm 1}\rangle \mapsto \frac{1}{\sqrt{2}}\left(|g, ga, a^{-1}\rangle \mp |g, ga^{-1}, a^{+1}\rangle\right)$$

MMcG: In [2] the quantum coin space is twice as large, and this incorporates memory. That is, the coin space for a cyclic group has four states. But it is not simply the tensor product of two sets of Walsh-Hadamard transformations; there is entanglement

In a classical random walk on the group $\mathbb{Z}$ the probability that the random walk is at any given point tends to zero as the number of steps tends to infinity.
In this paper we consider quantum Hadamard walks over a finite cyclic group $\mathbb{Z}_k$ with a standard generating set. There are various possibilities for a quantum Hadamard walk without memory on the group $\mathbb{Z}$. Here are three of them: This is achieved by replacing $|k\rangle$ in the computations by $|k \mod n\rangle$, so that there are k possible states for a group register, resulting in a $2k$-dimensional state space in the case without memory, and a $4k$-dimensional state space in the case with memory. There are various possibilities. Here are three of them:

1. The equivalent of the walk in [3], a "right-flip"

$$\begin{aligned} |n, R\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|n-1, R\rangle + |n+1, L\rangle\right) \\ |n, L\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|n-1, R\rangle - |n+1, L\rangle\right) \end{aligned}$$

2. The equivalent of the walk in [4], a "left-flip"

$$\begin{aligned} |n, R\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|n-1, R\rangle + |n+1, L\rangle\right) \\ |n, L\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|n+1, R\rangle - |n-1, L\rangle\right) \end{aligned}$$

3. The *left-reflecting walk* which it seems (by its behaviour) is the one in [1].

$$\begin{aligned} |n, R\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|n-1, R\rangle + |n+1, L\rangle\right) \\ |n, L\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|n+1, L\rangle - |n-1, R\rangle\right) \end{aligned}$$

2 The Right-Flip Walk

2.1 The Special Case $\mathbb{Z}_2$.

Here a group register may contain $|n\rangle$ or $|n+1\rangle$ only.

2.1.1 Without Memory

In $\mathbb{Z}_2$ without memory we have a 4-dimensional state space and

$$\begin{aligned} |n,0\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|n+1,0\rangle|n+1,1\rangle\right) \\ &\mapsto \frac{1}{2}\left(|n,0\rangle+|n,1\rangle+|n,0\rangle-|n,1\rangle\right) \\ &= |n,0\rangle \end{aligned}$$

so that the walk is periodic with period 2. This is not surprising as the operator W_2 is easily seen to decompose as a tensor product $X \otimes H$ where H is the Walsh-Hadamard transformation and X is a Pauli X-gate (cyclic permutation matrix of order 2), both of which have period 2.

2.1.2 With Memory

In $\mathbb{Z}_2$ with memory we have a 4-dimensional state space (states such as $|0,0,i\rangle$ and $|1,1,i\rangle$ are not allowed)

$$\begin{aligned} |n+1,n,0\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|n,n+1,0\rangle+|n,n+1,1\rangle\right) \\ &\mapsto \frac{1}{2}\left(|n+1,n,0\rangle+|n+1,n,1\rangle+|n+1,n,0\rangle-|n+1,n,1\rangle\right) \\ &= |n+1,n,0\rangle \end{aligned}$$

so that the walk is also periodic with period 2.

2.2 The Special Case $\mathbb{Z}_3$.

Here a group register may contain $|n\rangle$, $|n+1\rangle$ or $|n+2\rangle$.

2.2.1 Without memory

The state space is 6-dimensional (in $\mathbb{Z}_n$ in general it has dimension $2n$) and

$$\begin{aligned} |n,0\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|n+2,0\rangle+|n+1,1\rangle\right) \\ &\mapsto \frac{1}{2}\left(|n+1,0\rangle+|n,1\rangle+|n,0\rangle-|n+2,1\rangle\right) \\ &\mapsto \frac{1}{2\sqrt{2}}\left(|n,0\rangle+|n+2,1\rangle+2|n+2,0\rangle-|n+1,0\rangle+|n,1\rangle\right) \\ &\mapsto \frac{1}{4}\left(|n,1\rangle-|n,0\rangle+3|n+1,0\rangle+2|n+2,0\rangle-|n+2,1\rangle\right) \end{aligned}$$

2.2.2 With memory

The state space is 12-dimensional (in $\mathbb{Z}_n$ in general it has dimension $4n$) and

$$\begin{aligned} |n+2,n,0\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|n,n+2,0\rangle+|n,n+1,1\rangle\right) \\ &\mapsto \frac{1}{2}\left(|n+2,n,0\rangle+|n+2,n+1,1\rangle+|n+1,n,0\rangle-|n+1,n+2,1\rangle\right) \\ &\mapsto \frac{1}{2\sqrt{2}}\left(|n,n+2,0\rangle+|n,n+1,1\rangle+|n+1,n+2,0\rangle-|n+1,n,1\rangle+|n,n+1,0\rangle\right. \\ &\quad \left.+|n,n+2,1\rangle-|n+2,n+1,0\rangle+|n+2,n,1\rangle\right) \\ &\mapsto \frac{1}{\sqrt{13}}\left(|n+2,n,0\rangle+2|n+1,n,0\rangle+|n+2,n+1,0\rangle+|n+2,n,1\rangle-|n,n+1,0\rangle\right. \\ &\quad \left.+|n,n+2,1\rangle+|n+2,n,0\rangle-|n+1,n+2,0\rangle-|n+1,n,1\rangle+|n,n+2,0\rangle\right) \end{aligned}$$

3 The Left-Flip Walk

4 Left-Reflecting Walk

4.1 Without Memory

Consider the *left-reflecting* walk without memory given by

$$\begin{aligned} |n,R\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|n-1,R\rangle + |n+1,L\rangle\right) \\ |n,L\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|n+1,L\rangle - |n-1,R\rangle\right) \end{aligned}$$

implemented as follows:

```
public static SparseVector<String> cycleWalk(SparseVector<String> state,
                int mod){
        SparseVector<String> result = new SparseVector<string>();
        foreach (string s in state.Keys){
                int firstReg = Int32.Parse(s.Substring(0, s.Length - 1));
                SparseVector<String> vectorToAdd = new SparseVector<string>();
                if (s[s.Length - 1] == 'R'){
                        vectorToAdd[(shiftRightWithMod(firstReg,mod)).ToString()
                                + "R"] = 1 / Math.Sqrt(2) * state[s];
                        vectorToAdd[(shiftLeftWithMod(firstReg,mod)).ToString()
                                + "L"] = 1 / Math.Sqrt(2) * state[s];
                }
                if (s[s.Length - 1] == 'L'){
                        vectorToAdd[(shiftLeftWithMod(firstReg,mod)).ToString()
                                + "R"] = 1 / Math.Sqrt(2) * state[s];
                        vectorToAdd[(shiftRightWithMod(firstReg,mod)).ToString()
                                + "L"] = -1 / Math.Sqrt(2) * state[s];
                }
        result = (SparseVector<String>)result.plus(vectorToAdd);
        }
        return result;
}
```

Listing 1: C# code fragment for the left reflecting walk

For the cyclic group k the possibilities $k = 1$ up to $k = 16$ were tried. For all odd orders, *mixing* occurs, that is, the probability distribution of the location of the walk tends to a uniform distribution. For the even orders mixing did not occur. The probability distribution was periodic and the following was obtained (omitting normalization).

order	period	sequence
2	2	$\|0\rangle, \|1\rangle, \|0\rangle, \|1\rangle, \ldots$
4	4	$\|0\rangle, \|1\rangle + \|3\rangle, \|2\rangle, \|1\rangle + \|3\rangle, \|0\rangle$
6	2	$\|0\rangle + \|2\rangle + \|4\rangle, \|1\rangle + \|3\rangle + \|5\rangle, \ldots$
8	4	$\|0\rangle + \|4\rangle, \|1\rangle + \|3\rangle + \|5\rangle + \|7\rangle, \|2\rangle + \|6\rangle, \|1\rangle + \|3\rangle + \|5\rangle + \|7\rangle, \ldots$
10	2	$\|0\rangle + \|2\rangle + \cdots + \|8\rangle, \|1\rangle + \|3\rangle + \cdots + \|9\rangle, \ldots$
12	4	$\|0\rangle + \|4\rangle + \|8\rangle, \|1\rangle + \|3\rangle + \cdots + \|11\rangle, \|2\rangle + \|6\rangle + \|10\rangle, \|1\rangle + \|3\rangle + \cdots + \|11\rangle, \ldots$
14	2	$\|0\rangle + \|2\rangle + \cdots + \|12\rangle, \|1\rangle + \|3\rangle + \cdots + \|13\rangle, \ldots$
16	4	$\|0\rangle + \|4\rangle + \cdots + \|12\rangle, \|1\rangle + \|3\rangle + \cdots + \|15\rangle, \|2\rangle + \|6\rangle + \cdots + \|14\rangle,$ $\|1\rangle + \|3\rangle + \cdots + \|15\rangle, \ldots$

The general pattern seems to be for $k = 2(2n+1)$ we have a period 2 sequence

$$\sum |2m+1\rangle, \sum |2m\rangle, \ldots$$

and for $k = 4n$ we have a period 4 sequence

$$\sum |2m+1\rangle, \sum |4m\rangle, \sum |2m+1\rangle, \sum |2(2m+1)\rangle, \ldots$$

where m is an integer (in the right range).

5 Amplitudes

5.1 Hadamard Quantum Walk on $\mathbb{Z}$

We start from the known explict formula for amplitudes, given in various references ([5, Eqn. 5.27, page 74], [6, Equation II.15, page 7], [4, Equation (2), Lemma 4]). Considering a t-step walk, we want to know the amplitude at point n with $|n| \leq t$.

$$\Psi_R(n,t) = \begin{cases} 0 & \text{if } n+t \text{ is odd} \\ \frac{1}{\sqrt{2^t}} & \text{if } n = t \\ \frac{1}{\sqrt{2^t}} \sum_k \binom{(t-n)/2-1}{k-1}\binom{(t+n)/2}{k}(-1)^{(t-n)/2-k} & \text{if } n+t \text{ is even and } n \neq t \end{cases}$$

(The R above as a subscript refers to "right moving": Analogous formulae are present for L ("left moving"), that we won't write for now.) (We assume from now on in this section that $t+n$ is even...) The references listed do not make explicit the summation limits in k, so here we give them explicitly:

$$\Psi_R(n,t) = \frac{1}{\sqrt{2^t}} \sum_{k=1}^{(t-|n|)/2} \binom{(t-n)/2-1}{k-1}\binom{(t+n)/2}{k}(-1)^{(t-n)/2-k} \tag{1}$$

5.2 Hadamard Quantum Walk on $\mathbb{Z}_p$

We must distinguish the cases for p even or odd: For p even, the amplitudes will still be zero for $n+t$ odd, simply because ($n \mod p$) has the same parity as n. But for p odd, once t is appreciably larger than n, none of the amplitudes will be zero.
Set $n = x + rp$ where $r \in \mathbb{Z}$ and $0 \leq x \leq p-1$. Then we write Eqn. 1 as

$$\Psi_R(x,t,p) = \frac{1}{\sqrt{2^t}} \sum_r \sum_{k=1}^{(t-|x+rp|)/2} \binom{(t-(x+rp))/2-1}{k-1}\binom{(t+x+rp)/2}{k}(-1)^{(t-(x+rp))/2-k} \tag{2}$$

We break this rather long expression by writing

$$\Psi_R(x,t,p) = \sum_r \sum_{k=1}^{(t-|x+rp|)/2} f(x,t,p,r,k) \tag{3}$$

where

$$f(x,t,p,r,k) = \frac{1}{\sqrt{2^t}} \binom{(t-(x+rp))/2-1}{k-1}\binom{(t+x+rp)/2}{k}(-1)^{(t-(x+rp))/2-k} \tag{4}$$

1. Even p

   - <u>odd $x+t$:</u> $\Psi_R(x,t,p) = 0$

- even $x+t$: Since p is even, contributions to point x come from

$$\ldots x-2p, x-p, x, x+p, x+2p, \ldots \tag{5}$$

Simple investigation gives the upper and lower limits on r to be

$$r^+ = 1 + \lfloor (t-x-1)/p \rfloor \tag{6}$$
$$r^- = -\lfloor (t+x-1)/p \rfloor. \tag{7}$$

We then add in separately the contribution (if it exists) from $n=t$ (this will exist iff $x = t \bmod p$) to get

$$\Psi_R(x,t,p) = \frac{\delta_{t \bmod p,\ x}}{\sqrt{2^t}} + \sum_{r=r^-}^{r^+} \sum_{k=1}^{(t-|x+rp|)/2} f(x,t,p,r,k) \tag{8}$$

where δ is the standard Kronecker delta function, $\delta_{x,y} = 1$ iff $x = y$, otherwise zero.

2. Odd p

- odd t: Contributions to $\Psi_R(x,t,p)$ come from every **second** value of r. If x is odd, contributions come from $r \in \{\cdots -2, 0, 2, \ldots\}$. If x is even, contributions come from $r \in \{\cdots -3, -1, 1, 3, \ldots\}$.
- even t: Contributions to $\Psi_R(x,t,p)$ come from every **second** value of r. If x is even, contributions come from $r \in \{\cdots -2, 0, 2, \ldots\}$. If x is odd, contributions come from $r \in \{\cdots -3, -1, 1, 3, \ldots\}$.

In either case, we have

$$\Psi_R(x,t,p) = \frac{\delta_{t \bmod p,\ x}}{\sqrt{2^t}} + \sum_{r \in S} \sum_{k=1}^{(t-|x+rp|)/2} f(x,t,p,r,k) \tag{9}$$

where

$$S = \{v \in \mathbb{Z} | r^- \le v \le r^+, \text{ and } (t-x-v) \bmod 2 = 0\} \tag{10}$$

5.3 Some Examples on $\mathbb{Z}_p$

Consider the transitions

$$|n,0\rangle \mapsto \frac{1}{\sqrt{2}}(|n+1,0\rangle + |n-1,1\rangle)$$
$$|n,1\rangle \mapsto \frac{1}{\sqrt{2}}(|n+1,0\rangle - |n-1,1\rangle)$$

so that 0 means "move right" while 1 means "move left", and we get the phase of -1 for 2 successive left moves. Starting at $|0,0\rangle$ we get

$$\begin{aligned}
|0,0\rangle &\mapsto \frac{1}{\sqrt{2}}(|1,0\rangle + |-1,1\rangle) \\
&\mapsto \frac{1}{2}(|2,0\rangle + |0,1\rangle + |0,0\rangle - |-2,1\rangle) \\
&\mapsto \frac{1}{2\sqrt{2}}(|3,0\rangle - |-1,0\rangle + 2|1,0\rangle - |1,1\rangle + |-3,1\rangle) \\
&\mapsto \frac{1}{4}(|4,0\rangle + 3|2,0\rangle + |2,1\rangle - |0,0\rangle + |0,1\rangle + |-2,0\rangle - |-2,1\rangle - |-4,1\rangle) \\
&\mapsto \frac{1}{4\sqrt{2}}(|5,0\rangle + 4|3,0\rangle + |3,1\rangle + 2|1,1\rangle - 2|-1,1\rangle - |-3,0\rangle + 2|-3,1\rangle + |-5,1\rangle)
\end{aligned}$$

where we color in red the right movers for ease of legibility. Considering only the right movers, the un-normalized amplitudes are

t	$\Psi_R(n,t)$
0	1
1	0, 1
2	0, 1, 1
3	0, -1, 2, 1
4	0, 1, -1, 3, 1
5	0, -1, 0, 0, 4, 1

where the $t+1$ values in each row correspond to n taking on the values $-t, -t+2, \ldots, t-2, t$.

5.3.1 $\mathbb{Z}_3$

Considering the 5-step example we are following, so $t = 5$ and $p = 3$, the un-normalized right moving amplitudes at the 3 possible positions are

At $x = 0$: $4 - 1 = 3$

At $x = 1$: $0 + 0 = 0$

At $x = 2$: $1 + 0 = 1$

Equation (9) becomes

$$\Psi_R(x,5,3) = \frac{\delta_{2,\,x}}{\sqrt{2^5}} + \sum_{r\in S} \sum_{k=1}^{(5-|x+3r|)/2} f(x,5,3,r,k) \tag{11}$$

with $r^- = -\lfloor (4+x)/3 \rfloor$ and $r^+ = 1 + \lfloor (4-x)/3 \rfloor$. (Again in what follows we ignore $\sqrt{2}$ factors.)

- **Amplitude at $x = 1$:** $r^- = -1,\ r^+ = 2$, and $S = \{0, 2\}$. So we get

$$\Psi_R(1,5,3) = \sum_{k=1}^{2} f(1,5,3,0,k) = f(1,5,3,0,1) + f(1,5,3,0,2) = -2 + 2 = 0 \tag{12}$$

- **Amplitude at $x = 0$:** $r^- = -1,\ r^+ = 2$, and $S = \{-1, 1\}$. So we get

$$\Psi_R(0,5,3) = f(0,5,3,-1,1) + f(0,5,3,1,1) = -1 + 4 = 3 \tag{13}$$

5.4 Asymptotics

Given the explicit formulae for the amplitudes, we should calculate:

- What is $\lim_{t\to\infty} \Psi_R(x,t,p)$ for odd p?
- What is $\lim_{t\to\infty} \frac{1}{t} \sum_t \Psi_R(x,t,p)$ for odd p? (this should be the uniform distribution talked about in other papers...)
- What is $\lim_{t\to\infty} \frac{1}{p} \sum_{k=0}^{p-1} \Psi_R(x,t+k,p)$ for odd p? (this should be the uniform distribution also - we are summing over the cycle?!)
- The periodicity that seems to occur for even p should be provable from the explicit amplitudes, i.e. that there is some fixed l for which

$$\Psi_R(x,t,p) = \Psi_R(x,t+l,p) \quad \forall x. \tag{14}$$

6 Full Program Listing

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace GroupWalker
{
    class Program
    {
        static int period = 2;
        static string initialState = "0R";

        static void Main(string[] args)
        {
            SparseVector<String> state = new SparseVector<string>();
            state[initialState] = 1;
            Console.WriteLine("Step␣0");
            printFirstRegister(period, state);
            for (int i = 0; i < 2; i++)
            {
                Console.WriteLine("Step␣"+(i+1).ToString());
                state = cycleWalk(state,period);
                printFirstRegister(period,state);

            }
            Console.ReadKey(true);
        }

        public static SparseVector<String> cycleWalk(SparseVector<String> state,
            int mod)
        {
            SparseVector<String> result = new SparseVector<string>();
            foreach (string s in state.Keys)
            {
                int firstReg = Int32.Parse(s.Substring(0, s.Length - 1));
                SparseVector<String> vectorToAdd = new SparseVector<string>();
                if (s[s.Length - 1] == 'R')
                {
                    vectorToAdd[(shiftRightWithMod(firstReg,mod)).ToString()
                      + "R"] = 1 / Math.Sqrt(2) * state[s];
                    vectorToAdd[(shiftLeftWithMod(firstReg,mod)).ToString()
                      + "L"] = 1 / Math.Sqrt(2) * state[s];
                }
                if (s[s.Length - 1] == 'L')
                {
                    vectorToAdd[(shiftLeftWithMod(firstReg,mod)).ToString()
                      + "R"] = 1 / Math.Sqrt(2) * state[s];
                    vectorToAdd[(shiftRightWithMod(firstReg,mod)).ToString()
                      + "L"] = -1 / Math.Sqrt(2) * state[s];
                }
                result = (SparseVector<String>)result.plus(vectorToAdd);
            }
            return result;
        }

        public static string bar(double d)
        {
            int length = Convert.ToInt32(d * 50);
            string result="";
            for (int i = 0; i < length; i++)
                result += "X";
```

```
            return result;
        }

        public static void printState(int mod, SparseVector<String> state)
        {
            for (int i = 0; i < mod; i++)
            {
                if (state.ContainsKey(i.ToString()+"R"))
                    System.Console.WriteLine("|" + i + "R>␣"
                        + bar(state[i.ToString() + "R"]));
            }
            for (int i = 0; i < mod; i++)
            {
                if (state.ContainsKey(i.ToString() + "L"))
                System.Console.WriteLine("|" + i + "L>␣"
                        + bar(state[i.ToString() + "L"]));
            }
        }

        public static void printFirstRegister(int mod, SparseVector<String> state)
        {
            for (int i = 0; i < mod; i++)
            {
                double amplitude = 0;
                if (state.ContainsKey(i.ToString() + "R"))
                    amplitude += Math.Pow(state[i.ToString() + "R"],2);
                if (state.ContainsKey(i.ToString() + "L"))
                    amplitude += Math.Pow(state[i.ToString()+"L"],2);
                amplitude = Math.Sqrt(amplitude);
                System.Console.WriteLine("|" + i + ">␣" + bar(amplitude));
            }
        }

        public static int shiftLeft(int n)
        {
            return n - 1;
        }

        public static int shiftRight(int n)
        {
            return n + 1;
        }

        public static int shiftLeftWithMod(int n, int mod)
        {
            if (n>0)
                return n-1;
            return mod - 1;
        }

        public static int shiftRightWithMod(int n, int mod)
        {
            if (n < mod - 1)
                return n + 1;
            return 0;
        }

    }

}
```

Listing 2: C# code for main GroupWalker class

```
using System;
```

```
using System.Linq;
using System.Collections.Generic;
using System.Text;

namespace GroupWalker
{
    public class SparseVector<T> : Dictionary<T, double>, ISparseVector<T>
    {

        public ISparseVector<T> plus(ISparseVector<T> v2)
        {
            SparseVector<T> result = new SparseVector<T>();
            foreach (T t in this.Keys)
                if (v2.ContainsKey(t))
                {
                    if (this[t] + v2[t] != 0)
                        result[t] = this[t] + v2[t];
                }
                else
                    result[t] = this[t];
            foreach (T t in v2.Keys)
                if (!this.ContainsKey(t))
                    result[t] = v2[t];
            return result;
        }

        public ISparseVector<T> times(double f)
        {
            if (f == 0)
                return new SparseVector<T>();
            else
            {
                SparseVector<T> result = new SparseVector<T>();
                foreach (T t in this.Keys)
                    result[t] = f * this[t];
                return result;
            }
        }

        public ISparseVector<T> minus(ISparseVector<T> v1, ISparseVector<T> v2)
        {
            return v1.plus(v2.times(-1));
        }

        new public double this[T index]
        {
            set
            {
                if (value != 0)
                    base[index] = value;
            }
            get
            {
                return base[index];
            }
        }

        public double dot(ISparseVector<T> v)
        {
            double result = 0;
            foreach (T t in Keys)
                if (v.ContainsKey(t))
                    result += this[t] * v[t];
```

```
            return result;
        }

        public double modulus()
        {
            return Math.Sqrt(this.dot(this));
        }

        public ISparseVector<T> normalize()
        {
            if (this.modulus() == 0)
                return this;
            else
                return this.times(1 / this.modulus());
        }

        public override string ToString()
        {
            string result = "";
            foreach (T t in Keys)
                result += this[t] + "|" + t.ToString() + ">␣";
            return result;
        }

        public double distance(ISparseVector<T> v)
        {
            return 1 - this.normalize().dot(v.normalize());
        }

        public ISparseVector<T> closest(List<ISparseVector<T>> vList)
        {
            double leastDistance = 2;
            SparseVector<T> result = new SparseVector<T>();
            foreach (SparseVector<T> v in vList)
            {
                double currentDistance = distance(v);
                if (currentDistance <= leastDistance)
                {
                    leastDistance = currentDistance;
                    result = v;
                }
            }
            return result;
        }

    }
}
```

Listing 3: C# code for sparse vector class

References

[1] Aharonov, D., Ambainis, A., Kempe, J. and Vazirani, U. *Quantum Walks on Graphs*, In Proceedings of the 33rd ACM Symposium on Theory of Computing (2000) 50-59.

[2] Konno, N. and Machida, T. *Limit Theorems for Quantum Walks with Memory* arXiv:1004.0443v2 [quant-ph] (2010).

[3] McGettrick, M., *One Dimensional Quantum Walks with Memory* Quantum Information and Computation Vol. 10, No. 5&6 (2010) 509-524.

[4] Ambainis, A., Bach, E., Nayak, A., Vishwanath, A. and Watrous, J. *One Dimensional Quantum Walks* Proceedings of the 33rd ACM Symposium on Theory of Computing, pages 3749, 2001.

[5] Venegas-Andraca, S.E., *Quantum Walks for Computer Scientists*, Morgan and Claypool (2008).

[6] Brun, T.A., Carteret, H.A. and Ambainis, A., *Quantum Walks driven by many coins*, arXiv:0210161 [quant-ph] (2002). [Equation II.15, page 7]

Michael Batty,
School of Engineering and Computer Science,
Durham University,
Durham,
UK,
DH1 3LE,
email: michael.batty@dur.ac.uk

Michael McGettrick,
School of Mathematics,
The National University of Ireland,
Galway,
Ireland.
e-mail: