Quantum Random Walk: Computationally Efficient Approach

by José Luis Gómez-Muñoz
http://homepage.cem.itesm.mx/lgomez/quantum/
jose.luis.gomez@itesm.mx

Introduction

This is a more efficient implementation of the quantum random walk. The price is that is less "elegant" than the naive implementation, in the sense that some operators are not defined the way you do it in a handwritten quantum calculation.

The quantum random walker (Y. Aharonov, L. Davidovich, and N. Zagury "Quantum random walks" Phys. Rev. A 48, 1687 - 1690 (1993)) is made of a "coin" and a "walker", each one with its own state-space, which will make a composite system with the coin and the walker in entanglement. A unitary operator will be defined to "flip the coin", while another unitary operator will be defined to "move the walker" based on coin's result. The second operator produces entanglement between coin and walker.

Load the Package

First load the Quantum`Notation` package. Write:
Needs["Quantum`Notation`"];
then press at the same time the keys SHIFT-ENTER to evaluate. *Mathematica* will load the package:

```
Needs["Quantum`Notation`"]
```

```
Quantum`Notation` Version 2.2.0. (July 2010)
A Mathematica package for Quantum calculations in Dirac bra-ket notation
by José Luis Gómez-Muñoz

Execute SetQuantumAliases[] in order to use
  the keyboard to enter quantum objects in Dirac's notation
SetQuantumAliases[] must be executed again in each new
  notebook that is created, only one time per notebook.
```

In order to use the keyboard to enter quantum objects write:
SetQuantumAliases[ ];
then press at the same time the keys SHIFT-ENTER to evaluate. The semicolon prevents *Mathematica* from printing the help message. Remember that SetQuantumAliases[ ] must be evaluated again in each new notebook:

```
SetQuantumAliases[];
```

Quantum Random Walk using fast *Mathematica* commands

The coin C can have only two states, 0 and 1 (head and tail), while the walker P can be in any (discrete) position from -steps to steps. The coin is inially in state 0 and the walker is initially at the origin 0:

```
| w[0]〉 = | 0ĉ〉⊗ | 0p̂〉
```

```
| 0ĉ, 0p̂〉
```

Here we define the "fliping coin" operator, which in this case is a Hadamard operator, but it could be any unitary operator that acts only in the coin C. The coin can have only two states, 0 and 1 (head and tail). Notice the last sign is negative:

$$h = \frac{1}{\sqrt{2}} \left(|0_{\hat{c}}\rangle \cdot \langle 0_{\hat{c}}| + |1_{\hat{c}}\rangle \cdot \langle 0_{\hat{c}}| + |0_{\hat{c}}\rangle \cdot \langle 1_{\hat{c}}| - |1_{\hat{c}}\rangle \cdot \langle 1_{\hat{c}}| \right)$$

$$\frac{|0_{\hat{c}}\rangle \cdot \langle 0_{\hat{c}}| + |1_{\hat{c}}\rangle \cdot \langle 0_{\hat{c}}| + |0_{\hat{c}}\rangle \cdot \langle 1_{\hat{c}}| - |1_{\hat{c}}\rangle \cdot \langle 1_{\hat{c}}|}{\sqrt{2}}$$

This is one of the "effcient" parts of this implementation: After "flipping the coin" the walker is moved using the *Mathematica* ReplaceAll[] command. This evaluation does not produce any output, however the evolution of the system is calculted and stored in | w[0]〉, | w[1]〉,..., | w[20]〉

```
steps = 20;
Do[
  | w[k]〉 =
  Expand[ReplaceAll[h · | w[k - 1]〉,
     { | 0ĉ, j_p̂〉 :→ | 0ĉ, (j - 1)p̂〉, | 1ĉ, j_p̂〉 :→ | 1ĉ, (j + 1)p̂〉 }]], {k, 1, steps, 1}
]
```

Each state of the composite system coin-walker is stored. For example this is the state at the second step:

```
| w[2]〉
```

$$\frac{1}{2} |0_{\hat{c}}, (-2)_{\hat{p}}\rangle + \frac{1}{2} |0_{\hat{c}}, 0_{\hat{p}}\rangle + \frac{1}{2} |1_{\hat{c}}, 0_{\hat{p}}\rangle - \frac{1}{2} |1_{\hat{c}}, 2_{\hat{p}}\rangle$$

And this is the state at the fifth step:

```
| w[5]〉
```

$$\frac{|0_{\hat{c}}, (-5)_{\hat{p}}\rangle}{4\sqrt{2}} + \frac{|0_{\hat{c}}, (-3)_{\hat{p}}\rangle}{\sqrt{2}} - \frac{|0_{\hat{c}}, 3_{\hat{p}}\rangle}{4\sqrt{2}} + \frac{|1_{\hat{c}}, (-3)_{\hat{p}}\rangle}{4\sqrt{2}} + \frac{|1_{\hat{c}}, (-1)_{\hat{p}}\rangle}{2\sqrt{2}} - \frac{|1_{\hat{c}}, 1_{\hat{p}}\rangle}{2\sqrt{2}} + \frac{|1_{\hat{c}}, 3_{\hat{p}}\rangle}{2\sqrt{2}} + \frac{|1_{\hat{c}}, 5_{\hat{p}}\rangle}{4\sqrt{2}}$$

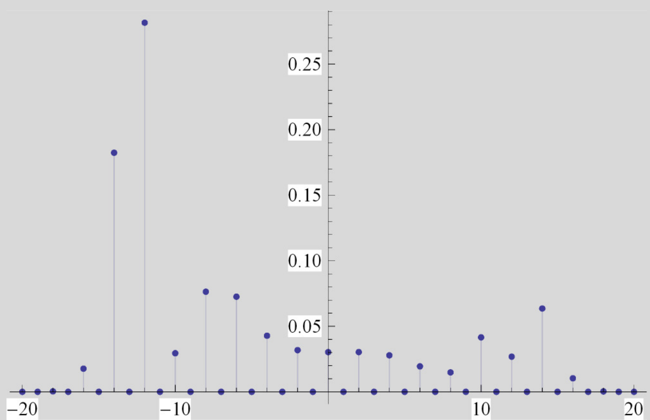
Here we calculate the probabilities for each walker position. This is another "efficient" part of this implementation:

```
prob = Table[
  {j,
   ketList = Cases[ | w[steps]⟩, x_. * | y_ĉ, j_p̂⟩];
   braList = Cases[⟨w[steps] |, x_. * ⟨y_ĉ, j_p̂ |];
   productList = MapThread[Function[{b, k}, b · k], {braList, ketList}];
   Apply[Plus, productList]
  },
  {j, -steps, steps}
 ]
```

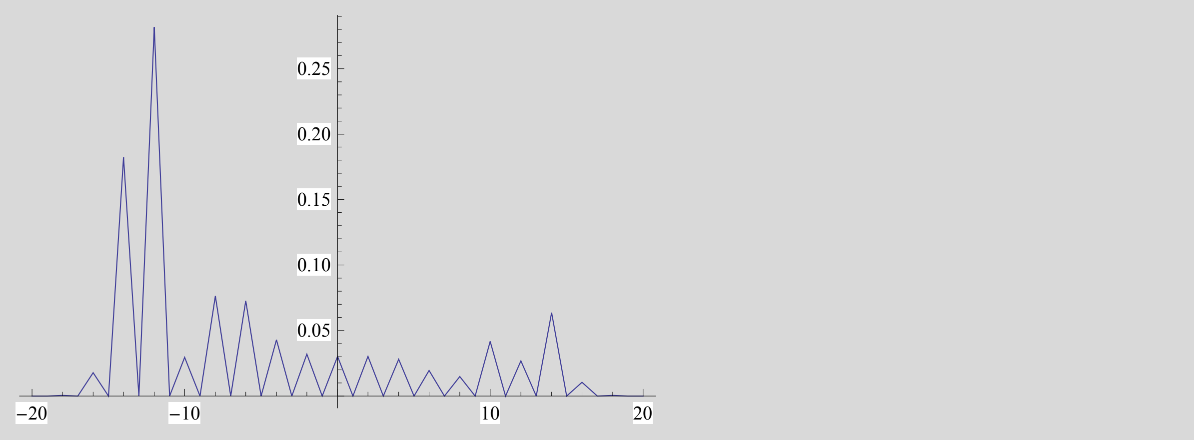
$$\left\{\left\{-20, \frac{1}{1048576}\right\}, \{-19, 0\}, \left\{-18, \frac{181}{524288}\right\}, \{-17, 0\}, \left\{-16, \frac{9257}{524288}\right\}, \{-15, 0\}, \left\{-14, \frac{95617}{524288}\right\}, \{-13, 0\}, \left\{-12, \frac{295265}{1048576}\right\}, \{-11, 0\}, \left\{-10, \frac{965}{32768}\right\}, \{-9, 0\}, \left\{-8, \frac{2501}{32768}\right\}, \{-7, 0\}, \left\{-6, \frac{2377}{32768}\right\}, \{-5, 0\}, \left\{-4, \frac{11221}{262144}\right\}, \{-3, 0\}, \left\{-2, \frac{4165}{131072}\right\}, \{-1, 0\}, \left\{0, \frac{3969}{131072}\right\}, \{1, 0\}, \left\{2, \frac{3969}{131072}\right\}, \{3, 0\}, \left\{4, \frac{7301}{262144}\right\}, \{5, 0\}, \left\{6, \frac{637}{32768}\right\}, \{7, 0\}, \left\{8, \frac{485}{32768}\right\}, \{9, 0\}, \left\{10, \frac{1361}{32768}\right\}, \{11, 0\}, \left\{12, \frac{28097}{1048576}\right\}, \{13, 0\}, \left\{14, \frac{33317}{524288}\right\}, \{15, 0\}, \left\{16, \frac{5417}{524288}\right\}, \{17, 0\}, \left\{18, \frac{145}{524288}\right\}, \{19, 0\}, \left\{20, \frac{1}{1048576}\right\}\right\}$$

Here is a plot of the probabilities for each position of the walker.

```
ListPlot[prob, PlotRange → All, Filling → Axis]
```



```
ListPlot[prob, PlotRange → All, Joined → True]
```



by José Luis Gómez-Muñoz
http://homepage.cem.itesm.mx/lgomez/quantum/
jose.luis.gomez@itesm.mx