A Quantum Algorithm to Efficiently Sample from Interfering Binary Trees

Davide Provasoli,1,* Benjamin Nachman,1,† Wibe A. de Jong,2,‡ and Christian W Bauer1,§
1Physics Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
2Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

Quantum computers provide an opportunity to sample from probability distributions that include non-trivial interference effects between a large number of amplitudes of binary trees. Using a simple process wherein all possible state histories can be specified by a binary tree, we construct an explicit quantum algorithm that runs in polynomial time to sample from the process once. An interesting feature of these binary trees is that they are not unitary, but can still be sampled on a quantum computer.

I. INTRODUCTION

Quantum algorithms are promising for various industrial and scientific applications because of their capacity to explore exponentially many states with a linear number of quantum bits. One of the most well-studied classes of quantum algorithms is the quantum walk [1]. Like the classical random walk, the quantum variants have found widespread use for enhancing a variety of quantum calculations and simulations [2, 3]. While quantum walks are fundamentally different from classical random walks, there are limits in which the quantum algorithm approaches the classical one [4].

A useful feature of a classical random walk is that it can be efficiently simulated using a Markov Chain Monte Carlo (MCMC) because subsequent motion depends only on the current position and not the prior history. This MC property is at the core of some algorithms that simulate many-body physical systems where the generative process is approximately local. For such physical systems that also have important quantum properties, the speed from the MCMC is traded off against the accuracy of an inherently quantum simulation. One such physical system is the parton shower in high energy physics [5], where a quark or a gluon radiates a shower of nearly collinear quarks and gluons. Genuine quantum effects can be approximated as corrections to the MCMC [6], but cannot be directly implemented efficiently in a classical MCMC approach.

Consider the following quantum tree: at every step, a spin 1/2 particle can move one unit left or one unit right. After N steps, this system forms a binary tree with 2N paths. In contrast to a traditional quantum walk, we assume that the path is observable, so moving left and then right is not the same as moving right and then left. For this reason, there is a 1-1 correspondence between the leaves of the tree and the path taken, and the space of measurement outcomes is more naturally \( \{L, R\}^N \) than Z.

When the quantum amplitude for moving left is independent of the spin or if the spin changes deterministically with time, this tree can be efficiently and accurately simulated with a classical MCMC. However, when either of these conditions are violated, a classical MCMC fails to produce the correct probability distribution over final states. While quantum walks with time/space dependence have been studied in the literature [4, 7–10] and there are some similarities to quantum algorithms for decision trees [11], our quantum tree requires a new approach.

In order to efficiently sample from the quantum tree, we introduce a new quantum algorithm that achieves an exponential speedup over an efficient classical calculation of the full final state probability distribution. In addition, we provide an explicit quantum circuit which implements the algorithm and demonstrate its performance on a quantum computer simulation. Interestingly, the final result we extract violates unitarity, even though it can be implemented on a quantum computer. This is qualitatively different from approaches to use non-unitary evolution in quantum walks [12].

This paper is organized as follows. Section II introduces the quantum tree and illustrates how classical algorithms cannot efficiently sample from its probability distribution. A solution to this problem is introduced in Sec. III using a quantum algorithm. An explicit implementation of the quantum circuit is described in Sec. IV and numerical results are presented in Sec. V. The paper ends with conclusions and future outlook in Sec. VI.

II. A CLASSICAL CHALLENGE

Consider a tree like the one shown in Fig. 1, where the quantum amplitude of a node n is given by \( A_L(n) = \sin(\theta_1(n)) \) when going left and \( A_R(n) = \cos(\theta_1(n)) \) when going right. The amplitude for reaching a given leaf is the product over the nodes from its history \( \lambda \in \{L, R\}^N \):

\[
A_{\text{leaf}} = \prod_{n=1}^N A_{\lambda_n}(n).
\]

The probability of paths through the tree (uniquely specified by a leaf) are distributed according to \( \Pr(\text{path}) \propto |A_{\text{leaf}}|^2 \). One can efficiently sample from this distribution in linear time classically using a MCMC algorithm: at each step, move left or right with a probability given by \( |A_{L/R}(n)|^2 \).
Consider the following minimal change to the tree: there is a spin state associated with each depth. Only the spin at the leaf is observable and the amplitudes $A_L$ and $A_R$ depend on the state of the spin. Now, there are many possible paths that correspond to reaching a single leaf. One way to visualize this is illustrated in Fig. 2. There are two copies of the tree, one for spin up and one for spin down. At each step, the system can move between trees or stay on the same tree and then move left or right. The observable final state is the leaf location and the final tree (spin). The amplitudes for going left and right are now spin-dependent. At a given step, the eight possible amplitudes are $A^s_{h} (n)$ for $h \in \{L, R\}$ and $s_i \in \{↑, ↓\}$, where $s_1$ is the initial spin and $s_2$ is the final spin. Since only the final spin is observable, the amplitude to transition from spin $s_0$ to $s_N$ is given by

$$A_{s_0, s_N} = \sum_{s_0' \in \{↑, ↓\}^N} N \prod_{n=1}^{N} A^{s_0'_1 \cdots s_0'_{n-1}, s_0'_{n}}_{s_1} (n). \quad (1)$$

For convenience, the 8 amplitudes for a given step are parameterized as

$$A_L^{↑} (n) = \cos(\theta^L_F (n)) \sin(\theta_1 (n))$$
$$A_L^{↓} (n) = \sin(\theta^L_F (n)) \sin(\theta_3 (n))$$
$$A_R^{↑} (n) = \cos(\theta^R_F (n)) \sin(\theta_2 (n))$$
$$A_R^{↓} (n) = \sin(\theta^R_F (n)) \sin(\theta_4 (n))$$
$$A_L^{↑} (n) = \cos(\theta^L_F (n)) \cos(\theta_1 (n))$$
$$A_L^{↓} (n) = \sin(\theta^L_F (n)) \cos(\theta_3 (n))$$
$$A_R^{↑} (n) = \cos(\theta^R_F (n)) \cos(\theta_2 (n))$$
$$A_R^{↓} (n) = \sin(\theta^R_F (n)) \cos(\theta_4 (n))$$

While there may be multiple applications of this quantum tree, one motivation is the parton shower in quantum chromodynamics (QCD) where quarks or gluons radiate gluons (going left in the tree) at decreasing angles (deeper $n$). The connection with QCD is not exact (in part because of unitarity violation - see Sec. III D), but the work presented here is an important step toward an inherently quantum parton shower algorithm.

The quantum tree including the full interference effects caused by cross-terms in the sum over all spin histories for a given leaf can not be implemented in a simple MCMC. One method for correctly sampling from the distribution of leaves and final spins is to sum over all paths to compute the probabilities for each state. For a tree of depth $N$, the calculation of the total amplitude would naively scale as $4^N$ since there are 4 possibilities at every node: move left and flip the spin, move right and flip the spin, move left and don’t flip the spin, move right and don’t flip the spin. For simplicity, we assume that the amplitude to move left with a spin flip is zero ($\theta_3 = \theta_4 = 0$) - in that case, the naive scaling is $3^N$. In practice, a more efficient scheme to calculate the full probability distribution is possible by reusing calculations. We further simplify the problem by setting $\theta^L_F = \theta^R_F = \theta_F$ and labeling $\theta_1 = \theta_3, \theta_2 = \theta_4$.

One way to efficiently calculate the probability distribution is to represent the problem as a set of matrix multiplications. To see this, consider the leaf corresponding to never taking the left branch. The probability for the two possible states (spin up or spin down) requires summing over all possible spin trajectories. If the initial spin is $|i\rangle$ for $|↓\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|↑\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, then one can compute the full probability distribution of the final spin $|f\rangle$ by matrix multiplication:

$$|f\rangle = \prod_{n=1}^{N} \Delta (n) |i\rangle, \Delta (n) = \left( \begin{array}{cc} A_L^{↑} (n) & A_R^{↑} (n) \\ A_L^{↓} (n) & A_R^{↓} (n) \end{array} \right). \quad (3)$$

Therefore, the amplitude for the right-move only case can be computed with $\mathcal{O}(N)$ multiplications. The same logic applies to the calculation of the amplitude for exactly one left branch at step $k$:

$$|f\rangle = \prod_{n=k+1}^{N} \Delta (n) \times A(k) \times \prod_{n=1}^{k-1} \Delta (n) |i\rangle, \quad (4)$$

where

$$A(k) = \left( \begin{array}{cc} A_L^{↑} (k) & 0 \\ 0 & A_L^{↓} (k) \end{array} \right). \quad (5)$$
Equation 4 is also inefficient when considering all $1 \leq k \leq N$, because many products can be reused from one $k$ value. However, even with the maximal amount of reuse, there must be at least one matrix multiplication per $k$ value. By the same logic, there must be at least one matrix multiplication for every fixed number of left branchings. There are a total of $2^N$ leaves and therefore the minimum number of matrix multiplications scales exponentially with $N$. Particular re-use schemes can be deployed to show that the scaling is $2^N$ (instead of the naive $3^N$) and to calculate the coefficient of the exponential scaling. In the next section will show that there exists a quantum algorithm that can distribute events from this probability distribution, where a single event can be generated in polynomial time. This therefore provides an exponential speedup over the classical approach.

III. A QUANTUM SOLUTION

A. Tree evolution as an efficient quantum algorithm

The system introduced in the previous section can be solved in polynomial time using a quantum algorithm that is described below. Once again we consider a tree of the kind illustrated in Fig. 2 with $N$ total nodes and a spin degree of freedom. The state which is evolved in our quantum circuit is given by

$$|\Psi_{n,N}\rangle = |a\rangle |s\rangle |\lambda_1 \lambda_2 \ldots \lambda_n \ldots \lambda_N\rangle$$

where $n$ denotes how many steps have occurred, and the combination of $|s\rangle$ and $|\lambda_1 \lambda_2 \ldots \lambda_n \ldots \lambda_N\rangle$ is abbreviated by $|\psi_{n,N}\rangle$, which we later will refer to as the physical state after step $n$, since it determines the node reached after $n$ steps.

To explain what these different qubits encode, recall that at each step three things can happen: the spin flips and the path goes right, the spin does not flip and the path goes right or the path goes left. At the end of the evolution, if we measure $|\lambda_i\rangle$ in the $|0\rangle$ state it denotes that the path went right at node $i$, while if we measure it in the $|1\rangle$ state, it denotes the path went left at node $i$. For the ket $|s\rangle$, $|0\rangle$ represents spin down and $|1\rangle$ represents spin up. In other words, these qubits uniquely identify a particular node in the two trees.

However, at each step, in the general case there are two ways to get to each node [see the 8 amplitudes in Eq. (2)], and after the simplifications discussed in the last section there are two ways to get to a right node. The ancillary qubit $|a\rangle$ holds the information whether a given node was reached by flipping trees or not in the latest step. Since the observable does not distinguish between the different spin flip histories, one can interfere the relevant amplitudes after each step. This allows to extract the required information contained in the ancillary qubit after each step, such that the same qubit can be used repeatedly. This is discussed in more detail later.

The evolution at a single step $i$ is shown in Fig. 3 (defining its shorthand notation as module $B_i$). Since

$$|\lambda_{n-1}\rangle$$

|...

|\lambda_0\rangle

\[U_{A,\downarrow}\] \[U_{A,\uparrow}\] $B_i$

|$a\rangle$

\[\mathbb{I}\]

FIG. 3. The quantum circuit at a single step. The $\lambda_i$ encode the path through the tree and $s$ is the spin, which is only observable at the end of the tree. $B_i$ represents the block surrounded by the dashed box.

|$a\rangle$ holds the information about the flipping of the spin, it controls which of the $U_A$ operations are applied. By applying the Hadamard gate at the end of the step, the amplitudes with and without a flip are added, creating the interference as given in Eq. (1).

The initial state of the system consists of the ancillary qubit $|a\rangle$ and all of the $|\lambda_i\rangle$ qubits being in the $|0\rangle$ state. The spin qubit, on the other hand, can be initialized in any superposition of $|0\rangle$ and $|1\rangle$.

The $U$ gates of Fig. 3 are single qubit operations represented by $2 \times 2$ unitary matrices with real entries, where we define the basis states on which these matrices act on as $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. The unitary transformation $U_F$ describes the amplitude for the spin to flip and is given by

$$U_F = \begin{pmatrix} \cos(\theta_F) & -\sin(\theta_F) \\ \sin(\theta_F) & \cos(\theta_F) \end{pmatrix}.$$  

The $U_{A,s}$ gates describe the amplitudes for the path to go left or right, and because these amplitudes depend on the spin of the system there is a $U_{A,\downarrow}$, which acts conditionally on the spin being $|0\rangle$, and a $U_{A,\uparrow}$, which acts conditionally on the spin being $|1\rangle$. Their matrix representation (with $s \in \{\downarrow, \uparrow\}$) is

$$U_{A,s} = \begin{pmatrix} \cos(\theta_s) & -\sin(\theta_s) \\ \sin(\theta_s) & \cos(\theta_s) \end{pmatrix}.$$  

$\uparrow$ The possible step dependence of $\theta_F$ and $\theta_s$ and other quantities from Sec. II is suppressed for compactness.
Now that we have defined the circuit block $B_i$ for one step, we can construct the circuit for the full evolution for $N$ steps. This is shown schematically in Fig. 4. The last operation on the ancillary qubit, labeled with a $|0\rangle$, corresponds to measuring the ancillary qubit and proceeding to the next step only if we measure $|0\rangle$. If we measure instead the ancillary qubit to be in the $|1\rangle$ state, the circuit evolution is interrupted and we start all over again. As we will see in an explicit example later, accepting only the states where the ancillary qubit is in the $|0\rangle$ state selects the correct interferences between physical states, where all the amplitudes are positive. The state with the ancillary qubit in the state $|1\rangle$ would give the linear superposition of the amplitudes with a negative sign, unlike Eq. (1). Now that the ancillary qubit has been measured, it can be reused for the next step, which explains the previous assertion that only a single ancillary qubit is necessary.

![Circuit Diagram](image.png)

**FIG. 4.** The full quantum circuit written in terms of the single steps $B_i$. The last operation on the ancillary qubit, labeled with a $|0\rangle$, corresponds to measuring the ancillary qubit and proceeding to the next step only if we measure $|0\rangle$.

In general the probability of the spin to flip and the probabilities of the path to go left or right depend on the evolution variable, meaning the matrices $U_F$, $U_A\downarrow$ and $U_A\uparrow$ are different at each step. At the end of the circuit evolution, we measure the physical state $|\psi_{N,N}\rangle$ (we have already measured the ancillary qubit) and we record the output. This way we sampled the distribution of physical final states and generated one event. This corresponds, in our tree notation, to reaching a final tree leaf with definite spin.

### B. Circuit Evolution

We explicitly compute the circuit evolution for two steps. For simplicity, we start the spin qubit in the $|0\rangle$ state, such that the initial state is

$$|\psi_{0,2}\rangle = |0\rangle |0\rangle |00\rangle .$$

(9)

After $B_1$ is applied the state is evolved to

$$\frac{1}{\sqrt{2}} \left[ \cos(\theta_F) \cos(\theta_{\downarrow}) (|0\rangle + |1\rangle) |0\rangle |00\rangle + \cos(\theta_F) \sin(\theta_{\downarrow}) (|0\rangle + |1\rangle) |0\rangle |10\rangle + \sin(\theta_F) (|0\rangle - |1\rangle) |1\rangle |00\rangle \right].$$

(10)

The negative sign in the last line of the above equations shows that $|a\rangle = |1\rangle$ encodes the difference instead of the sum of amplitudes. Performing the conditional measurement on the ancillary qubit we find

$$|\Psi_{1,2}\rangle = \frac{1}{\sqrt{2}} \left[ \cos(\theta_F) \cos(\theta_{\downarrow}) |0\rangle |0\rangle |00\rangle + \cos(\theta_F) \sin(\theta_{\downarrow}) |0\rangle |0\rangle |10\rangle + \sin(\theta_F) |0\rangle |1\rangle |00\rangle \right].$$

(11)

Applying the second circuit block ($B_2$ plus the conditional measurement) we obtain

$$|\psi_{2,2}\rangle = \frac{1}{2} \left[ (\cos^2(\theta_{\downarrow}) + \sin^2(\theta_F) + \sin^2(\theta_F)) |0\rangle |0\rangle |00\rangle + \ldots \right],$$

(12)

where the amplitudes for the remaining leaves do not have multiple terms as is the case with the displayed leaf amplitude. Physically, this corresponds to the fact that there are two ways to reach the leaf $|0\rangle |00\rangle$: always going right and either never swapping trees or swapping twice.

If we go to a higher number of steps or if we start the fermion in a superposition of the $|0\rangle$ and $|1\rangle$ states, the number of interferences grows very quickly. To compute the probability of measuring an eigenstate we square the appropriate amplitude and we multiply it by a factor of $2^N$ (which equals 4 in this case). The latter is necessary because the factor of $\frac{1}{\sqrt{2^N}}$ in front of the final state is not physical, but is the result of applying $N$ Hadamard gates and selecting to keep only the states we want with the conditional measurement we apply to the ancillary qubit at each step.

### C. Quantum Complexity

We now want to show that the above quantum circuit can generate one event in polynomial time, meaning the number of standard quantum gates employed grows polynomially with the number of steps. Each step in the circuit consist of a constant number of gates. To determine the complexity of the quantum circuit we have to find how many times, on average, we must run a circuit block to generate one event. If we are simulating $N$ steps we must run $N$ circuit blocks and after each block we must measure the ancillary qubit to be in the $|0\rangle$ state. For a sequence of $N$ measurements on one qubit there are $2^N$ possible outcomes, meaning we would have to run $\sim 2^N$ circuit block on average to obtain one event. However,
because we stop as soon as we measure the ancillary qubit to be in the $|1\rangle$ state, the number of possible outcomes is much smaller. We can have the case where the first measurement gives $|0\rangle$ and then we stop, the case where the first two measurements give $|0\rangle$ and then we stop, and so forth until the case where all $N$ measurements give $|0\rangle$ and we record the event. This corresponds to a total of

$$1 + 2 + 3 + \ldots + N = \frac{1}{2}N(N + 1) \simeq N^2 \quad (13)$$

possible outcomes. This means the number of times we must run a circuit block on average to obtain one event grows as $N^2$. Because each circuit block is made up of a fixed number of quantum gates we have shown that the complexity of the quantum algorithm grows polynomially, with leading term proportional to $N^2$. Of course, to obtain a given observable with accuracy $\epsilon$ requires the number of events to scale with $1/\epsilon^2$, and the scaling of the quantum algorithm to compute such an observable is therefore of order $N^2/\epsilon^2$.

D. Unitarity

An interesting feature of the problem we are considering is the non-unitarity of the total cross section, i.e. the sum of the all the probabilities to obtain the possible physical final states is not equal to one. This is not in disagreement with the fact that all transformations in the quantum circuit are unitary, since the physical states are only those for which the ancillary qubit was in the state $|0\rangle$ and the final probability is multiplied by a factor of $2^N$. Thus, the violation of unitarity is simply due to the fact that the total probability for the ancillary qubits to be in the state $|0\rangle$ $N$ times is not equal to $2^{-N}$, which is the result of the quantum interference present in this scenario. Because the amplitudes of the physical states are always positive, the interferences always happen with a plus sign, explaining why we always find the total probability to be larger than one.

IV. EXPLICIT IMPLEMENTATION ON A QUANTUM COMPUTER

In order to run a full and realistic quantum simulation of our quantum circuit, and eventually implement it on a test bed, it is necessary to decompose it in terms of single qubit gates and CNOT gates only. In this section we use well-known results for the decomposition of quantum gates that can be found for example in Section 4 of Ref.[13]. The current form of one circuit block was presented in Fig. 3. The CNOT gate and the Hadamard gate do not need to be simplified any further. $U_F$ can be written as $R_Y(2\theta_F)$, a standard single qubit rotation gate. In current quantum computer prototypes such single qubit rotation gates can be implemented by approximating them through a series expansion.

We are left with two $C^2(U_{A,s})$ operations, which will require more work to decompose. We use the decomposition in Fig. 5 to relate a unitary transformation controlled on $|0\rangle$ to one controlled on $|1\rangle$, where $X$ is the standard CNOT gate. The double controlled gate can then be decomposed as shown in Fig. 6. Finally, to break down the controlled-$R_Y$ gate, one uses the decomposition shown in Fig. 7, where

$$\alpha_s = \frac{\theta_s}{2} \quad \beta_s = -\theta_s. \quad (14)$$

Combining the above results our quantum circuit can be rewritten only in terms of 24 standard single qubit gates and 17 CNOT gates, yielding the 41 gate circuit shown in Fig. 8. A major challenge in implementing this circuit on a quantum computer is the lack of support to perform sequential measurements on the same qubit, using the measurement output as a condition to perform further operations. However, we were able to simulate our quantum circuit using IBM’s Qiskit Python API [14]. Our work might provide some motivations to find ways to make such repeated measurements feasible and efficient on quantum computing hardware.

V. RESULTS

This section shows some numerical results for simulations of the quantum algorithm and how it compares
VI. CONCLUSIONS

In this work, we have introduced a system similar to the quantum walk which smoothly interpolates between a binary tree, amenable to classical MCMC approaches, and interfering trees with non-trivial quantum phenomenology. When non-trivial interference effects are

with a classical MCMC implementation. The quantum circuit is implemented with Qiskit [14]. To compute the distributions of various observables, the algorithm is run many times and each measured outcome (leaf and final spin) is recorded. With these ‘events’, it is possible to then compute the distribution of any observable. For illustration, two observables are considered: the number of times the system moved left and the first time the system moved left. As in the calculation from Sec. III, the state always starts as spin down.

A classical MCMC is constructed by sampling from the squared amplitudes at each step. This classical simulation does not contain any interference effects and is therefore expected to produce the incorrect probability distributions for a generic observable when \( \theta_F \neq 0 \).

We run our simulations with \( N = 4 \), with \( \cos^2(\theta_F) \) for each step taken as \( (0.2, 0.3, 0.4, 0.5) \) and \( (0.4, 0.5, 0.6, 0.7) \) for \( \cos^2(\theta_1) \). Figure 9 shows results for the number of left branches, while Fig. 10 shows results for the step the first left branch occurred. In both cases, the histograms in the left plot show the probability distributions when \( \theta_F = 0 \) (both for the quantum algorithm and a classical MCMC), while the right plot shows how the expectation value of the observables depends on \( \cos^2(\theta_F) \). As expected, the expectation values are the same for the MCMC and for the quantum algorithm when \( \theta_F = 0 \), but differ as interference effects are introduced. We have verified that the results from the quantum algorithm agree with the analytical calculation of the full probability distribution using the exponentially scaling method introduced in Sec. II. The difference between the MCMC and the quantum algorithm also goes to zero as \( \cos \theta_F \to 0 \), in which case the spin flips at each step in a deterministic way and thus there are no interference effects.
introduced, a classical calculation of all possible outcomes scales exponentially with the depth of the tree. We have introduced a quantum algorithm that uses an innovative remeasuring technique to sample from the interfering trees with polynomial scaling with the depth of the tree. In addition to constructing an explicit quantum circuit to implement the algorithm, some numerical results were presented with a simulated quantum computer.

Given the wide-ranging applicability of classical random walks and quantum walks to aiding complex algorithms, it is likely that the algorithm presented here will be a useful addition to the quantum toolkit. The application of the interfering trees algorithm and its variations to empowering MCMC algorithms of physical systems could empower many body simulations where quantum effects were previously ignored. More complex simulations and calculations will also be possible as quantum software and hardware continue to improve.

ACKNOWLEDGMENTS

This work is supported by the DOE under contract DE-AC02-05CH11231. In particular, support comes from Quantum Information Science Enabled Discovery (QuantISED) for High Energy Physics (KA2401032).

[1] Y. Aharonov, L. Davidovich, and N. Zagury, “Quantum random walks,” Phys. Rev. A 48, 1687–1690 (1993).
[2] Salvador Elías Venegas-Andraca, “Quantum walks: a comprehensive review,” Quantum Information Processing 11, 1015–1106 (2012).
[3] J Kempe, “Quantum random walks: An introductory overview,” Contemporary Physics 44, 307–327 (2003), https://doi.org/10.1080/00107151031000110776.
[4] Miquel Montero, “Classical-like behavior in quantum walks with inhomogeneous, time-dependent coin operators,” Phys. Rev. A 93, 062316 (2016).
[5] C. Patrignani et al. (Particle Data Group), “Review of Particle Physics, Monte Carlo Event Generators,” Chin. Phys. C40, 100001 (2016).
[6] Zoltan Nagy and Davison E. Soper, “A parton shower based on factorization of the quantum density matrix,” JHEP 06, 097 (2014), arXiv:1401.6364 [hep-ph].
[7] A P Flitney, D Abbott, and N F Johnson, “Quantum walks with history dependence,” Journal of Physics A: Mathematical and General 37, 7581 (2004).
[8] M. C. Bañuls, C. Navarrete, A. Pérez, Eugenio Roldán, and J. C. Soriano, “Quantum walk with a time-dependent coin,” Phys. Rev. A 73, 062304 (2006).
[9] P. Xue, R. Zhang, H. Qin, X. Zhan, Z. H. Bian, J. Li, and Barry C. Sanders, “Experimental quantum-walk revival with a time-dependent coin,” Phys. Rev. Lett. 114, 140502 (2015).
[10] S Panahiyan and S Fritzschke, “Controlling quantum random walk with a step-dependent coin,” New Journal of Physics 20, 083028 (2018).
[11] Edward Farhi and Sam Gutmann, “Quantum computation and decision trees,” Phys. Rev. A 58, 915–928 (1998).
[12] VIV KENDON, “Decoherence in quantum walks - a review,” Mathematical Structures in Computer Science 17, 1169–1220 (2007).
[13] Michael A. Nielsen and Isaac L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition, 10th ed. (Cambridge University Press, New York, NY, USA, 2011).
[14] IBM Research, “Qiskit, an open-source computing framework.” (2018).