A Quantum Algorithm to Efficiently Sample from Interfering Binary Trees

Davide Provasoli,[*] Benjamin Nachman,[†] and Christian Bauer[‡]

Physics Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

Wibe A. de Jong[§]

Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

Quantum computers provide an opportunity to efficiently sample from probability distributions that include non-trivial interference effects between amplitudes. 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. The corresponding naive Markov Chain algorithm does not produce the correct probability distribution and an explicit classical calculation of the full distribution requires exponentially many operations. However, the problem can be reduced to a system of two qubits with repeated measurements, shedding light on a quantum-inspired efficient classical algorithm.

I. INTRODUCTION

Quantum algorithms are promising for various industrial and scientific applications because of their capacity to explore exponentially many states with a polynomial 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 2^N 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 $\mathbb{Z}$.

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 naive 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, an equivalent quantum circuit involving only two qubits can be obtained if we use repeated measurements, and this shed light on a quantum inspired classical algorithm that is indeed an efficient MCMC.

This paper is organized as follows. Section II introduces the quantum tree and illustrates how naive 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. An efficient quantum-inspired classical MCMC is introduced in Sec. VI. The paper ends with conclusions and future outlook in Sec. VII.

---

[*] davideprovasoli@lbl.gov
[†] bpnachman@lbl.gov
[‡] cwbauer@lbl.gov
[§] WAdeJong@lbl.gov

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 $A_L(n)$ when going left and $A_R(n)$ 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 $\text{Pr(path)} \propto |A_{\text{leaf}}|^2$. One can efficiency sample from this distribution in linear time classically using a MCMC algorithm: at each step, move left or right with a probability given by $|A_{L/R}(n)|^2$.

Now, consider the following 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 $A_h^{s_1,s_2}(n)$ for $h \in \{L, R\}$ and $s_i \in \{\uparrow, \downarrow\}$, 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

$$A_{s_0,s_N} = \sum_{\substack{\vec{s}' \in \{\downarrow,\uparrow\}^N \\ s_0'=s_0, s_N'=s_N}} \prod_{n=1}^{N} A_{\lambda_n}^{s'_{n-1},s'_n}(n). \qquad (1)$$

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 but the work presented here is a step toward an inherently quantum parton shower algorithm.
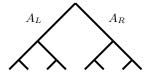

FIG. 1: The rightmost nodes of the above binary tree (leaves) uniquely correspond to trajectories in $\{L, R\}^N$ where L represents going left and R represents going right at a given node. As a generative model, trajectories are sampled according to the square of the quantum amplitude of the path through the tree.

The quantum tree including the full interference effects caused by cross-terms in the sum over all spin histories for a given leaf cannot be implemented in a naive MCMC that extends the one from Fig. 1 where each possibility
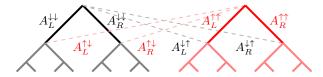

FIG. 2: The same setup as in Fig. 1, except that now there is a spin state associated with every depth in the tree. This can be represented by two trees: one for spin down (left) and one for spin up (right). The system can move between trees, but only the final tree (spin) and leaf are observable. The eight possible amplitudes for a given step are indicated with $A_{L/R}^{s_1,s_2}$, where s_1 is the initial spin and s_2 is the final spin.

was sampled at each step. One method for correctly sampling from the distribution of leafs 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 do not flip the spin, move right and do not flip the spin.

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 $|\downarrow\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|\uparrow\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, then one can compute the full probability distribution of the final spin $|f\rangle$ by matrix multiplication:

$$|f\rangle = \prod_{n=1}^{N} \Delta(n) |i\rangle, \Delta(n) = \begin{pmatrix} A_R^{\downarrow\downarrow}(n) & A_R^{\uparrow\downarrow}(n) \\ A_R^{\downarrow\uparrow}(n) & A_R^{\uparrow\uparrow}(n) \end{pmatrix}. \qquad (2)$$

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:

$$|f\rangle = \prod_{n=k+1}^{N} \Delta(n) \times A(k) \times \prod_{n=1}^{k-1} \Delta(n) |i\rangle, \qquad (3)$$

where

$$A(k) = \begin{pmatrix} A_L^{\downarrow\downarrow}(k) & A_L^{\uparrow\downarrow}(k) \\ A_L^{\downarrow\uparrow}(k) & A_L^{\uparrow\uparrow}(k) \end{pmatrix}. \qquad (4)$$

Equation 3 is also inefficient when considering all $1 \leq k \leq N$, because many products can be reused from one k to another. 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 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 naive classical approach.

III.   A QUANTUM SOLUTION

A.   Rotating to a new basis

Let us write the evolution in Eq. (1) in terms of the following two unitary transformations, pertaining to one step starting on the $|\downarrow\rangle$ tree and one step starting on the $|\uparrow\rangle$ tree:

$$
\begin{aligned}
|\downarrow\rangle &\to A_L^{\downarrow\downarrow}\,|L\rangle\,|\downarrow\rangle + A_L^{\downarrow\uparrow}\,|L\rangle\,|\uparrow\rangle + A_R^{\downarrow\downarrow}\,|R\rangle\,|\downarrow\rangle + A_R^{\downarrow\uparrow}\,|R\rangle\,|\uparrow\rangle \\
|\uparrow\rangle &\to A_L^{\uparrow\uparrow}\,|L\rangle\,|\uparrow\rangle + A_L^{\uparrow\downarrow}\,|L\rangle\,|\downarrow\rangle + A_R^{\uparrow\uparrow}\,|R\rangle\,|\uparrow\rangle + A_R^{\uparrow\downarrow}\,|R\rangle\,|\downarrow\rangle \,,
\end{aligned}
\tag{5}
$$

where the amplitudes must satisfy the unitarity conditions:

$$
\begin{aligned}
A_L^{\downarrow\downarrow\,2} + A_L^{\downarrow\uparrow\,2} + A_R^{\downarrow\downarrow\,2} + A_R^{\downarrow\uparrow\,2} &= 1 \\
A_L^{\uparrow\uparrow\,2} + A_L^{\uparrow\downarrow\,2} + A_R^{\uparrow\uparrow\,2} + A_R^{\uparrow\downarrow\,2} &= 1 \,.
\end{aligned}
\tag{6}
$$

This evolution will produce interference terms, since we can reach the same state in more than one way, and as previously mentioned it cannot be implemented with a simple naive MCMC. We would like to rotate to a new basis

$$
\begin{aligned}
|\downarrow'\rangle &= \cos\lambda\,|\downarrow\rangle - \sin\lambda\,|\uparrow\rangle \\
|\uparrow'\rangle &= \sin\lambda\,|\downarrow\rangle + \cos\lambda\,|\uparrow\rangle \,,
\end{aligned}
\tag{7}
$$

such that one evolution step looks like

$$
\begin{aligned}
|\downarrow'\rangle &\to \widetilde{A}_L^{\downarrow\downarrow}\,|L\rangle\,|\downarrow'\rangle + \widetilde{A}_R^{\downarrow\downarrow}\,|R\rangle\,|\downarrow'\rangle \\
|\uparrow'\rangle &\to \widetilde{A}_L^{\uparrow\uparrow}\,|L\rangle\,|\uparrow'\rangle + \widetilde{A}_R^{\uparrow\uparrow}\,|R\rangle\,|\uparrow'\rangle \,,
\end{aligned}
\tag{8}
$$

with unitarity conditions

$$
\begin{aligned}
(\widetilde{A}_L^{\downarrow\downarrow})^2 + (\widetilde{A}_R^{\downarrow\downarrow})^2 &= 1 \\
(\widetilde{A}_L^{\uparrow\uparrow})^2 + (\widetilde{A}_R^{\uparrow\uparrow})^2 &= 1 \,.
\end{aligned}
\tag{9}
$$

In the new basis the two trees decouple and the evolution becomes simple, meaning our quantum states evolve at each step by going either right or left, but they can no longer go in between trees. The original system had six degrees of freedom (8 amplitudes and two unitary conditions given in Eq. (6)) while the new system has only three degrees of freedom (one from λ, four amplitudes and two unitary conditions given in Eq. (9)). This

means that this basis switch is only possible for a subset of cases for the original problem. Since these cases admit a simple quantum algorithm, we focus on these and leave the general case for future studies.

In order to find the correct rotation angle λ to implement Eq. (7), we must solve:

$$
\begin{aligned}
|\downarrow'\rangle \to {}&\cos\lambda(A_L^{\downarrow\downarrow}\,|L\rangle\,|\downarrow\rangle + A_L^{\downarrow\uparrow}\,|L\rangle\,|\uparrow\rangle + A_R^{\downarrow\downarrow}\,|R\rangle\,|\downarrow\rangle \\
&+ A_R^{\downarrow\uparrow}\,|R\rangle\,|\uparrow\rangle) - \sin\lambda(A_L^{\uparrow\uparrow}\,|L\rangle\,|\uparrow\rangle + A_L^{\uparrow\downarrow}\,|L\rangle\,|\downarrow\rangle \\
&+ A_R^{\uparrow\uparrow}\,|R\rangle\,|\uparrow\rangle + A_R^{\uparrow\downarrow}\,|R\rangle\,|\downarrow\rangle) \\
={}&|L\rangle\Big[\cos\lambda A_L^{\downarrow\downarrow}\,|\downarrow\rangle + \cos\lambda A_L^{\downarrow\uparrow}\,|\uparrow\rangle - \sin\lambda A_L^{\uparrow\uparrow}\,|\uparrow\rangle \\
&- \sin\lambda A_L^{\uparrow\downarrow}\,|\downarrow\rangle\Big] + |R\rangle\Big[\cos\lambda A_R^{\downarrow\downarrow}\,|\downarrow\rangle \\
&+ \cos\lambda A_R^{\downarrow\uparrow}\,|\uparrow\rangle - \sin\lambda A_R^{\uparrow\uparrow}\,|\uparrow\rangle - \sin\lambda A_R^{\uparrow\downarrow}\,|\downarrow\rangle\Big] \,.
\end{aligned}
\tag{10}
$$

Focusing on the term proportional to $|L\rangle$, from Eq. (10) we have

$$
\begin{aligned}
\widetilde{A}_L^{\downarrow\downarrow}\,|\downarrow'\rangle &= \widetilde{A}_L^{\downarrow\downarrow}\big(\cos\lambda\,|\downarrow\rangle - \sin\lambda\,|\uparrow\rangle\big) \\
&= \cos\lambda A_L^{\downarrow\downarrow}\,|\downarrow\rangle + \cos\lambda A_L^{\downarrow\uparrow}\,|\uparrow\rangle - \sin\lambda A_L^{\uparrow\uparrow}\,|\uparrow\rangle \\
&\quad - \sin\lambda A_L^{\uparrow\downarrow}\,|\downarrow\rangle \,,
\end{aligned}
\tag{11}
$$

so that we get the following two equations:

$$
\begin{aligned}
\cos\lambda\widetilde{A}_L^{\downarrow\downarrow} &= \cos\lambda A_L^{\downarrow\downarrow} - \sin\lambda A_L^{\uparrow\downarrow} \\
-\sin\lambda\widetilde{A}_L^{\downarrow\downarrow} &= \cos\lambda A_L^{\downarrow\uparrow} - \sin\lambda A_L^{\uparrow\uparrow}
\end{aligned}
\tag{12}
$$

If we multiply the top equation by $\sin\lambda$, the bottom equation by $\cos\lambda$ and add them we get

$$
\cos\lambda\sin\lambda(A_L^{\downarrow\downarrow} - A_L^{\uparrow\uparrow}) + \cos\lambda^2(A_L^{\uparrow\downarrow} + A_L^{\downarrow\uparrow}) - A_L^{\uparrow\downarrow} = 0 \,.
\tag{13}
$$

Now if we repeat the same process with the transformation of $|\uparrow'\rangle$ and once again we focus on the terms proportional to $|L\rangle$ we obtain

$$
\begin{aligned}
\sin\lambda\widetilde{A}_L^{\uparrow\uparrow} &= \sin\lambda A_L^{\downarrow\downarrow} + \cos\lambda A_L^{\downarrow\uparrow} \\
\cos\lambda\widetilde{A}_L^{\uparrow\uparrow} &= \sin\lambda A_L^{\downarrow\uparrow} + \cos\lambda A_L^{\uparrow\uparrow}
\end{aligned}
\tag{14}
$$

and

$$
\cos\lambda\sin\lambda(A_L^{\downarrow\downarrow} - A_L^{\uparrow\uparrow}) + \cos\lambda^2(A_L^{\uparrow\downarrow} + A_L^{\downarrow\uparrow}) - A_L^{\downarrow\uparrow} = 0 \,.
\tag{15}
$$

Then Eq. (13) and Eq. (15) imply

$$
A_L^{\downarrow\uparrow} = A_L^{\uparrow\downarrow} \equiv A_L
\tag{16}
$$

and they become

$$
\cos\lambda\sin\lambda(A_L^{\downarrow\downarrow} - A_L^{\uparrow\uparrow}) + \cos 2\lambda A_L = 0 \,.
\tag{17}
$$

We can now solve for λ in terms of A_L, $A_L^{\uparrow\uparrow}$ and $A_L^{\downarrow\downarrow}$, which are free parameters we will specify in the unrotated

basis, and use the result to solve for $\widetilde{A}_L^{\downarrow\downarrow}$ and $\widetilde{A}_L^{\uparrow\uparrow}$ in Eqs. 12 and 14 . When we do so we get

$$\widetilde{A}_L^{\downarrow\downarrow} = A_L^{\downarrow\downarrow}$$
$$- \frac{\sqrt{4A_L^2(A_L^{\downarrow\downarrow} - A_L^{\uparrow\uparrow})^2 + (A_L^{\downarrow\downarrow} - A_L^{\uparrow\uparrow})^4} + (A_L^{\downarrow\downarrow} - A_L^{\uparrow\uparrow})^2}{2(A_L^{\downarrow\downarrow} - A_L^{\uparrow\uparrow})}$$

$$(18)$$

$$\widetilde{A}_L^{\uparrow\uparrow} = A_L^{\downarrow\downarrow}$$
$$+ \frac{2A_L^2(A_L^{\downarrow\downarrow} - A_L^{\uparrow\uparrow})}{\sqrt{4A_L^2(A_L^{\downarrow\downarrow} - A_L^{\uparrow\uparrow})^2 + (A_L^{\downarrow\downarrow} - A_L^{\uparrow\uparrow})^4} + (A_L^{\downarrow\downarrow} - A_L^{\uparrow\uparrow})^2} \, .$$

$$(19)$$

We can then find $\widetilde{A}_R^{\uparrow\uparrow}$ and $\widetilde{A}_R^{\downarrow\downarrow}$ from unitarity conditions in 9. Of course we could have performed the same derivation focusing on terms proportional to $|R\rangle$ instead, in which case, instead of eq. 17, we would have found

$$A_R^{\downarrow\uparrow} = A_R^{\uparrow\downarrow} \equiv A_R \qquad (20)$$

and

$$\cos\lambda \sin\lambda (A_R^{\downarrow\downarrow} - A_R^{\uparrow\uparrow}) + \cos 2\lambda A_R = 0 \, . \qquad (21)$$

B. Tree evolution as an efficient quantum algorithm

We now introduce a quantum algorithm which can solve the system introduced in the previous section in polynomial time. The algorithm implements the change of basis discussed above, it evolves the system in the decoupled basis and then rotates back to the original basis, creating interferences between all the possible paths which lead to the same final leaf and spin.

To illustrate the algorithm 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

$$|\Psi_{n,N}\rangle = |s\rangle\, |\lambda_1\lambda_2\ldots\lambda_n\ldots\lambda_N\rangle$$
$$\equiv |\psi_{n,N}\rangle \, , \qquad (22)$$

where n denotes how many steps have occured 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 determines the node reached after n steps.

To explain what these different qubits encode, recall that at each step the spin can either flip or not flip meaning we can go form one tree to the other or we can stay on the same tree, and the path can either go right or 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. While we keep track whether the path went right or left at each step and we measure this information at the end of the evolution by measuring all the λ_i qubits, we do not keep track of whether the the spin flipped or not in a particular step, which is why we can reach the same node with different spin histories.

The quantum circuit which implements the evolution is shown in Fig. 3. The $|\lambda_i\rangle$ qubits are initialized in the $|0\rangle$ state while the spin qubit, on the other hand, can be initialized in any superposition of $|0\rangle$ and $|1\rangle$. The R gate is responsible for rotating into the diagonalized basis, it is given by the 2×2 real unitary matrix

$$R = \begin{pmatrix} \cos\lambda & -\sin\lambda \\ \sin\lambda & \cos\lambda \end{pmatrix} , \qquad (23)$$

while $R^\dagger$ rotates back to the original basis at the end of the evolution before we perform a measurement. The $U_{\uparrow/\downarrow}^i$ gates are also single qubit operations represented by 2×2 real unitary matrices (we drop the step index for simplicity), which in quantum computing are referred to as R_Y rotations:

$$U_\downarrow = \begin{pmatrix} \cos\theta_\downarrow & -\sin\theta_\downarrow \\ \sin\theta_\downarrow & \cos\theta_\downarrow \end{pmatrix} = \begin{pmatrix} \widetilde{A}_L^{\downarrow\downarrow} & -\widetilde{A}_R^{\downarrow\downarrow} \\ \widetilde{A}_R^{\downarrow\downarrow} & \widetilde{A}_L^{\downarrow\downarrow} \end{pmatrix}$$

$$U_\uparrow = \begin{pmatrix} \cos\theta_\uparrow & -\sin\theta_\uparrow \\ \sin\theta_\uparrow & \cos\theta_\uparrow \end{pmatrix} = \begin{pmatrix} \widetilde{A}_L^{\uparrow\uparrow} & -\widetilde{A}_R^{\uparrow\uparrow} \\ \widetilde{A}_R^{\uparrow\uparrow} & \widetilde{A}_L^{\uparrow\uparrow} \end{pmatrix} , \qquad (24)$$

where we define the basis states on which these matrices act on as $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.
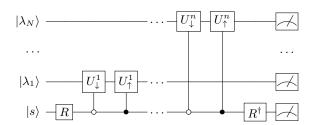

FIG. 3: Complete quantum circuit which implements full tree evolution

In general the probabilities of the path to go left or right (as well as λ) could depend on the step, meaning the matrices $U_\downarrow$ and $U_\uparrow$ are different at each step. If λ is different, then $R^\dagger(\lambda)R(\lambda')$ operations would need to be inserted between each step. At the end of the circuit evolution, we measure all of the qubits and we record the output. This way we sampled the distribution of final states and generated one event. This corresponds, in our tree notation, to reaching a final tree leaf with definite spin.

IV. IMPLEMENTATION ON A QUANTUM COMPUTER

In order to run a full quantum simulation of our circuit, and implement it on a currently available test bed, it is necessary to decompose it in terms of single qubit gates and CNOT gates only We have to break down two controlled-R_Y operations, one controlled on $|0\rangle$ and one controlled on $|1\rangle$. We use Fig. 4 to relate a unitary transformation controlled on $|0\rangle$ to one controlled on $|1\rangle$, where X is the standard CNOT gate. To break down the

FIG. 4: Decomposition of a single controlled gate.

controlled-R_Y gate, one uses the decomposition shown in Fig. 5, with
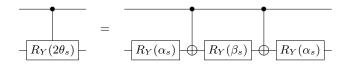

FIG. 5: The decomposition of the controlled R_Y gate.

$$\alpha_s = \frac{\theta_s}{2} \quad \beta_s = -\theta_s \,. \tag{25}$$

Combining the above results our quantum circuit can be decomposed in terms of $2 + 12N$ standard qubit gates (single qubit gates and CNOT gates), showing that the number of gates scales linearly with the number of steps.

V. NUMERICAL RESULTS

This section shows some numerical results for simulations of the quantum algorithm and how it compares with a naive classical MCMC implementation. The quantum circuit is implemented with Qiskit [12]. 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. For these illustrations, the state always starts as spin down.

A naive 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 $\lambda \neq 0$.

We run our simulations with $N = 20$, with constant values of $\cos^2(\theta_\uparrow) = 0.5$ and $\cos^2(\theta_\downarrow) = 0.8$. Figure 6 shows histograms of the two observables for different values of λ while Fig. 7 shows how the expectation values for the two observables scales with λ.

As expected, the expectation values are the same for the naive MCMC and for the quantum algorithm when $\lambda = 0$, but differ as interference effects are introduced. We have verified that the results from the quantum algorithm agree with the 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 $\lambda \to \pi/2$, in which case the spin flips at each step in a deterministic way and thus there are no interference effects.
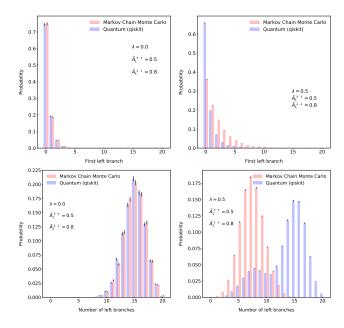

FIG. 6: Top: histograms of the first depth in which the tree goes left. Bottom: the number of left branches from the entire tree. Left: $\lambda = 0$ (no interference effects). Right: $\lambda = 0.5$. Error bars correspond to Poission uncertainties from the finite simulation.

VI. A QUANTUM-INSPIRED CLASSICAL ALGORITHM

At each step the $U_\downarrow$ and $U_\uparrow$ gates are conditionally applied to a new qubit, but after that the qubit is left alone until the final measurement at the end of the evolution. Therefore, at each step one could measure the qubit on which the $U_{\downarrow/\uparrow}$ gates act on, store the result in a classical register, reset it to the initial $|0\rangle$ state and reuse it for the next step. Using this method of repeated measurements and resetting the measured qubits one can rewrite the circuit in terms of just two qubits as shown in Fig-
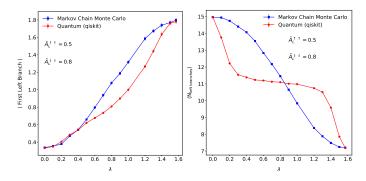
FIG. 7: Left: the expectation value of the depth of the first left branch as a function of λ. Right: the expectation value for the number of total left branches as a function of λ. Error bars correspond to Poission uncertainties from the finite simulation.

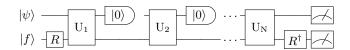ure 8. At each step one records the measurement on the

FIG. 8: A quantum algorithm that implements the circuit shown in Figure 3; each U_i corresponds to the two controlled operations in the earlier circuit.

second qubit, and at the very end the first qubit is measured. The combination of these measurements makes up one event. Note that because this circuit can be implemented using just 2 qubits, one can in fact find an efficient quantum inspired classical algorithm.

We show how this classical algorithm works by considering the kth step. Because the second qubit is reset to $|0\rangle$ after each step, the state at the beginning of the step (before U_k is applied) will always be of the form

$$|\psi_k\rangle = \begin{pmatrix} a_1^{(k)} \\ 0 \\ a_3^{(k)} \\ 0 \end{pmatrix}. \qquad (26)$$

After applying the U_k operation through matrix multiplication one finds

$$|\psi_k\rangle \rightarrow U_k |\psi_k\rangle = \begin{pmatrix} b_1^{(k)} \\ b_2^{(k)} \\ b_3^{(k)} \\ b_4^{(k)} \end{pmatrix}, \qquad (27)$$

where the $b_i^{(k)}$ are determined from the $a_1^{(k)}$ and $a_3^{(k)}$ through multiplication with the matrix U_k.

From this one finds that the probabilities P_0 and P_1 to measure the second qubit as $|0\rangle$ or $|1\rangle$ are given by

$$P_0 = b_1^{(k)2} + b_3^{(k)2}, \qquad P_1 = b_2^{(k)2} + b_4^{(k)2} \qquad (28)$$

The corresponding states after resetting the second qubit to $|0\rangle$ are given by

$$|\psi_k\rangle_0 = \frac{1}{\sqrt{P_0}} \begin{pmatrix} b_1^{(k)} \\ 0 \\ b_3^{(k)} \\ 0 \end{pmatrix}, \qquad |\psi_k\rangle_1 = \frac{1}{\sqrt{P_1}} \begin{pmatrix} b_2^{(k)} \\ 0 \\ b_4^{(k)} \\ 0 \end{pmatrix}. \qquad (29)$$

Both of these states have form

$$|\psi_{k+1}\rangle = \begin{pmatrix} a_1^{(k+1)} \\ 0 \\ a_3^{(k+1)} \\ 0 \end{pmatrix}, \qquad (30)$$

which has exactly the same form of the state we started with, so that this process can be repeated again.

The same result can be obtained classically by the following classical algorithm for generating a single event, which starts from a fermion in the superposition $f = af_1 + \sqrt{1-a^2}f_2$, and where we have defined the matrix $U \equiv U_k$. The event is stored in the classical register c_f holding the type of fermion and $c_\psi[\text{step}]$, which holds whether a left branch happened at the given step. The procedure is described algorithmically in Alg. (1).

Create empty vector for classical register $c_\psi[m]$
Set $a_1 = a$ and $a_3 = \sqrt{1-a^2}$
for step $= 1 \ldots m$ **do**
    Set $b_i = U_{ij}a_j$
    Set $P_0 = (b_1^2 + b_3^2)$ and $P_1 = b_2^2 + b_4^2$
    if rand() $< P_0$ **then**
        $c[\text{step}] = 0$
        $a_1 = b_1/\sqrt{P_0}$ and $a_3 = b_3/\sqrt{P_0}$
    else
        $c[\text{step}] = 1$
        $a_1 = b_2/\sqrt{P_1}$ and $a_3 = b_4/\sqrt{P_1}$
    end if
end for
if rand() $< a_1^2/(a_1^2 + a_3^2)$ **then**
    $c_f = 0$
else
    $c_f = 1$
end if

ALGORITHM 1: Quantum inspired classical algorithm.

VII. CONCLUSIONS

In this work, we have introduced a system similar to the quantum walk which smoothly interpolates between a binary tree, amenable to naive classical MCMC approaches, and interfering trees with non-trivial quantum phenomenology. When non-trivial interference effects are 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. Interestingly, the simple nature of the quantum algorithm inspired a classical approach that is still a Markov chain and thus efficient.

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.

[1] Y. Aharonov, L. Davidovich, and N. Zagury, Phys. Rev. A **48**, 1687 (1993).

[2] S. E. Venegas-Andraca, Quantum Information Processing **11**, 1015 (2012).

[3] J. Kempe, Contemporary Physics **44**, 307 (2003), https://doi.org/10.1080/00107151031000110776.

[4] M. Montero, Phys. Rev. A **93**, 062316 (2016).

[5] C. Patrignani *et al.* (Particle Data Group), Chin. Phys. **C40**, 100001 (2016).

[6] Z. Nagy and D. E. Soper, JHEP **06**, 097 (2014), arXiv:1401.6364 [hep-ph].

[7] A. P. Flitney, D. Abbott, and N. F. Johnson, Journal of Physics A: Mathematical and General **37**, 7581 (2004).

[8] M. C. Bañuls, C. Navarrete, A. Pérez, E. Roldán, and J. C. Soriano, Phys. Rev. A **73**, 062304 (2006).

[9] P. Xue, R. Zhang, H. Qin, X. Zhan, Z. H. Bian, J. Li, and B. C. Sanders, Phys. Rev. Lett. **114**, 140502 (2015).

[10] S. Panahiyan and S. Fritzsche, New Journal of Physics **20**, 083028 (2018).

[11] E. Farhi and S. Gutmann, Phys. Rev. A **58**, 915 (1998).

[12] IBM Research, "Qiskit, an open-source computing framework," (2018).