Quantune: An Automatic Music Generation Using Quantum Computing

Md Shahidur Rahaman, Agm Islam

August 10, 2022

1 Project Description

Researchers are attempting to employ the algorithms and principles we may utilize in the music industry to benefit from the recent rapid advancements in quantum computing. Consequently, the music business adopted the computational way of providing interactive music. In response to the rising requirement to construct a model for music production, we conceived a model based on the Basak-Miranda algorithm [6] we reckoned to be the definitive method for creating music.

Over the years, there has been an enormous amount of study on music creation utilizing computer logic and simulation. Geoff Hill first produced a pitch identification tool for playback [3]. Hiller et al. [4] developed a computer named ILLIAC, which can comprise a string quartet, considered the pioneered works of algorithmic music computation. In 2009, Bretan et al. provided a deep neural network technique for tune automation which eventually generates music [2]. The researchers recently shifted their interest to quantum simulation. Because in classical computers, the note, which is the unit of a tune, can be only one state at a time, whereas in quantum computing using superposition, a note can be at multiple states that speed up the simulation. Kirke et al. [5] first experimented the sound and music by applying the quantum process, and Miranda later generated using the musical sequencer. However, we found significantly less research in music generation using the quantum process.

1.1 Basic Quantum Functionalities

For our implementation, we have used qiskit Library [7], an open-source software development kit for developing quantum circuits. The following terminology we need is the qubit, a 2-D quantum system regarded as the rudimentary information carrier in modern quantum computers. We can define qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. The quantum register is where we can perform

the calculations and manipulations of the n number of a qubit. We now then introduce the gates we have used in our implementation.

- **X-Gate:** We consider the first gate, the Pauli-X, which is the quantum counterpart of the NOT gate for classical computers concerning the standard basis, which determines the z-axis on the Bloch sphere [1].
- **Hadamard Gate:** One of the essential gates in quantum computing that shifts the Bloch sphere's poles and produces a superposition.
- **CX Gate:** The CX Gate has a substantial effect on multi-qubit operations because it allows one gate to serve as a control and another as a target.
- **CCX Gate:** This gate is called the Toffoli gate, incorporates a three-qubit operation, and is considered the reversible arrangement of AND gate.

As we are dealing with five-qubits, we need to compute the merged states of our system. We calculate the tensor product($\otimes$) of two or more independent qubits to construct combinational states.

1.2 Algorithm Details:

Our first interest in quantum computing was motivated by the random walk technique. The discrete interpretation of this procedure utilizes several qubits that incorporate a traditional random walk with Markov Chains [9] for each potential movement direction from a graph vertex. In addition, we can offer a set of twelve-tone notes for musical creation. A quantum-die then uses this music to accomplish the quantum walk in one dimension. For instance, we may operate the Markov chain probability distribution to discover the playing note shown in table 1. The simulation phase selects the left note if the computed value is zero; otherwise, it determines the right note. A single qubit and a Hadamard gate will allow us to create the quantum die. A 1-D random walk responds nicely to this technique. This method won't perform, though, when applying complicated rules sequentially. As a result, we use the Basak-Miranda algorithm, the most recent method for the enhanced quantum walk algorithm [6]. The algorithm brought the procedure of generating a matrix considering the sequence rule we obtained from the targeted state in the Markov probability distribution table.
Assume that the Markov chain implementation for note $C\#$ produced two states with equal probabilities. Now we generate a matrix called $\varkappa$ in equation (1). Using rule 3(third row in the matrix), the simulation will then proceed. We can choose the one from the rule that matches the intended states represented by the qubits. We depicted the overall implementation of the method in figure 1. Taking into account the pitch, which we already took, the two qubits we obtain are for the states A and E, respectively, $|0\rangle_2$ and $|8\rangle_2$. Considering the next pitch in the simulation, we may infer from the matrix

	A	F	C	G#	D#	F#	D#	G	E	C	E	A#
E	0.25							0.25	0.25	0.25		
D#			0.25		0.25			0.25		0.25		
C#	0.5								0.5			
G							0.33	0.33		0.33		
D					0.33		0.33				0.33	
F				0.33		0.33		0.33				
C			0.25				0.25	0.25				0.25
F#		.50						.50				
A				0.25	0.25		0.25			0.25		
G#				0.25			0.25			0.25	0.25	
A#	0.25				0.25			0.25		0.25		

Table 1: Markov Chain Sequence Rules

that the winner is $|3\rangle_4$. The histogram allows us to rapidly determine the state, and we will then use the same steps to complete the input tuning computation.

$$\varkappa = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{1}$$

1.3 Implementation for Tune generation:

The algorithm stated before is adequately justified in five steps by our implementation.

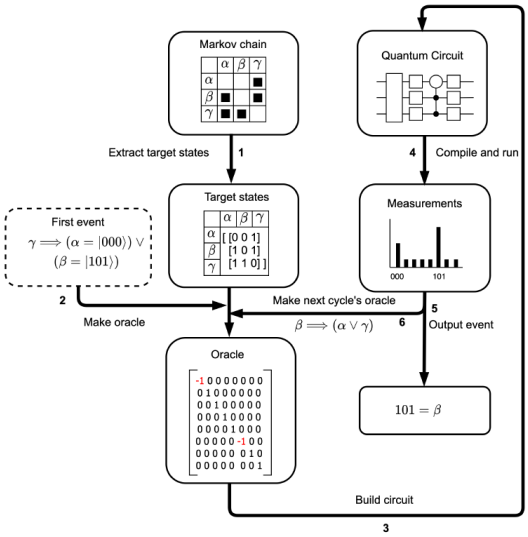
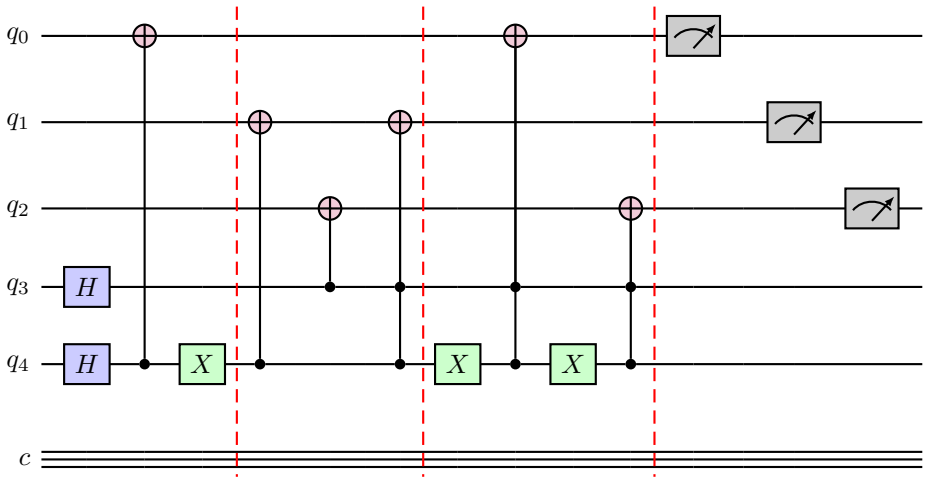


Figure 1: Steps in Basak-Miranda Implementation [6]



Implemented Quantum Circuit

- **Circuit Creation:** We use a different approach called qwalk to create the circuit in the first stage. We first specify the qubits, conventional bits, and quantum registers within that approach. Then, we consider the three conventional bits and the five qubits. Finally, we employ the Hadamard(H), Pauli X-gate(X), Controller-Not(CX), and Controlled-Controlled-Not methods using these (CCX). We honor from the gates that we created a multi-qubit procedure for our model.

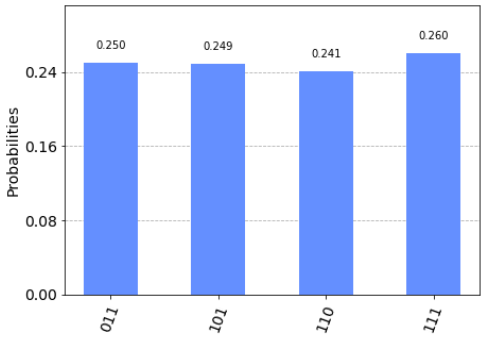


Figure 2: Histogram of note generation

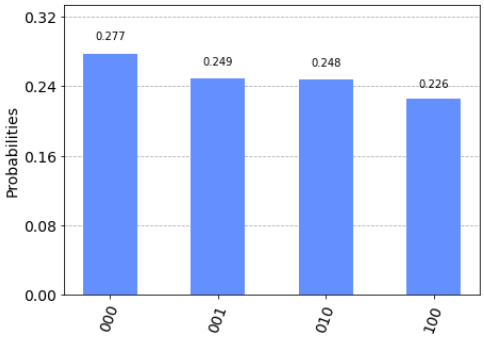


Figure 3: Histogram of rhythm generation

- **Transpilation:** We infer the development of the transpilation mechanism to repurpose our input circuit to mimic the structural design of a quantum device is the second stage of our approach.
- **Note Generation:** Our circuit is repeated 100 times in the third phase to produce the note. Then, in 25 rounds, the histogram tracks the pitches that the simulation chose. We depicted the 100^{th} note production probability using a histogram in figure 2.
- **Rhythm Generation:** The process of creating rhythms using the earlier-created technique comes next. Then, we execute the circuit method one hundred times to discover the beats. Finally, we construct the histogram in figure 3 for this process to determine the state using the pertinent probability.
- **Audio Generation:** We utilize the MIDIUti [8] to generate the audio mp3 file from the rhythm and tune we got from the earlier states. We undertook the duration, pitch, loudness, and channel. The note is generated and added to each cycle. Finally, we created the mp3 audio file so that we could execute it. Our model's code is available on our GitHub repository:

1.4 Web Application:

We developed an interactive web application to play the generated music to achieve our objective and motivation, and it is now open to the public. We have utilized the frontend with React.Js and ran the code we constructed for the backend to play the music in the music gallery. The publicly hosted URL link: https://qtune.thepixel.men/

1.5 Conclusion & Future Works

With the emerging demand in the field of music, we acknowledge this implementation will open a new door for developers and quantum computing enthusiasts. We thoroughly investigated our performance with proper justification with two different methodologies incorporating quantum superposition to pitch generation and the Basak-Miranda algorithm to rhythm generation. Our web application plays the music from the backend code we generated from the quantum principles and functionalities. For our future work, we would like to try human-voice integration, which may result in automation in song creation without human interaction.

References

[1] Michel Boyer, Rotem Liss, and Tal Mor. Geometry of entanglement in the bloch sphere. *Physical Review A*, 95(3):032308, 2017.

[2] Mason Bretan, Gil Weinberg, and Larry Heck. A unit selection methodology for music generation using deep neural networks. *arXiv preprint arXiv:1612.03789*, 2016.

[3] Paul Doornbusch. Computer sound synthesis in 1951: The music of csirac. *Computer Music Journal*, 28(1):10–25, 2004.

[4] Lejaren Hiller and Leonard Maxwell Isaacson. *Illiac suite, for string quartet*, volume 30. New Music Edition, 1957.

[5] Alexis Kirke and Eduardo R Miranda. Experiments in sound and music quantum computing. In *Guide to unconventional computing for music*, pages 121–157. Springer, 2017.

[6] Eduardo R Miranda and Suchitra T Bask. Quantum computer music: Foundations and initial experiments. *arXiv preprint arXiv:2110.12408*, 2021.

[7] Pierriccardo Olivieri, Mehrnoosh Askarpour, and Elisabetta Di Nitto. Experimental implementation of discrete time quantum walk with the ibm qiskit library. In *2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE)*, pages 33–38. IEEE, 2021.

[8] Noah Studach. Robot composer framework, 2018.

[9] Xin-She Yang, TO Ting, and Mehmet Karamanoglu. Random walks, lévy flights, markov chains and metaheuristic optimization. In *Future information communication technology and applications*, pages 1055–1064. Springer, 2013.