Quantune: An Automatic Music Generation Using Quantum Computing

Md Shahidur Rahaman, Agm Islam

August 10, 2022

1 Project Description

As quantum computing has recently advanced rapidly, researchers are trying to take advantage of the principles and algorithms they can utilize in the music industry. As a result, the computational approach to dynamic music creation has become an integral aspect of the music business. With the emerging need to establish a music composition model, we developed a model using the Basak-Miranda algorithm [5], which we found the standard convention for generating music. This section will describe the basic functionalities of quantum computing for our project, the model we used for the implementation, and an in-depth analysis of our result.

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 [2]. Hiller et al. [3] 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[1]. 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. [4] 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[6], 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.

- **The 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. $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0|$
- **Hadamard Gate:** One of the essential gates in quantum computing that shifts the Bloch sphere's poles and produces a superposition.
- **The 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.
- **The 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:

The random walk algorithm drove our initial focus on quantum computing. For each movement possibility from a graph vertex, the discrete version of this method employs several qubits incorporating classic random walk with Markov Chains [8]. In addition, the series of twelve-tone notes are provided for music composition, which is then run by a quantum die for one-dimensional implementation of the quantum walk. For example, we can find the following playing note using the Markov chain probability distribution (See Table 1). If we get the calculated value of 0, then the simulation step chooses the left note otherwise picks the right one. Finally, we can implement the quantum die using a single qubit and a Hadamard gate. This algorithm works very well in a 1-D random walk. However, for complex rule implementation in sequence, this will not work. Therefore, we utilize the latest approach for the advanced quantum walk algorithm, the Basak-Miranda algorithm [5]. This approach constructs a rule matrix for the target states using the Markov Chain representation for the sequence rule.
For note $C\#$, suppose we have two equilibrium probability states in the Markov chain implementation. Now, we develop a matrix $\varkappa$ in equation (1). Now for the same scenario, it will proceed to Rule 3 in the matrix, which is the third row. We can extract the one corresponding to the targeted states represented using the qubits from the rule. In the following diagram 1, we get the overall implementation of that algorithm. Considering the pitch we have is $C\#$ we took earlier. The two qubits we get from here are $|0\rangle_2$ and $|8\rangle_2$ for the

	E	F	G	C#	F#	D#	G#	D	B	C	A	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

states E and B. From the matrix, if the simulation's next pitch taken is E, then the winner we got is $|0\rangle_4$. We can quickly get the state using the histogram, and now from that E state, we will follow the same procedure to finish the input pitch simulation.

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

1.3 Implementation for Tune generation:

Our implementation implies a five-step procedure with proper justification of the algorithm discussed earlier.

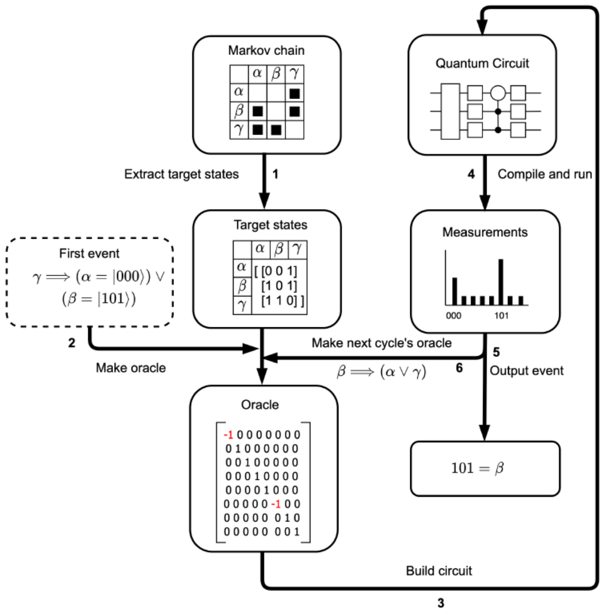
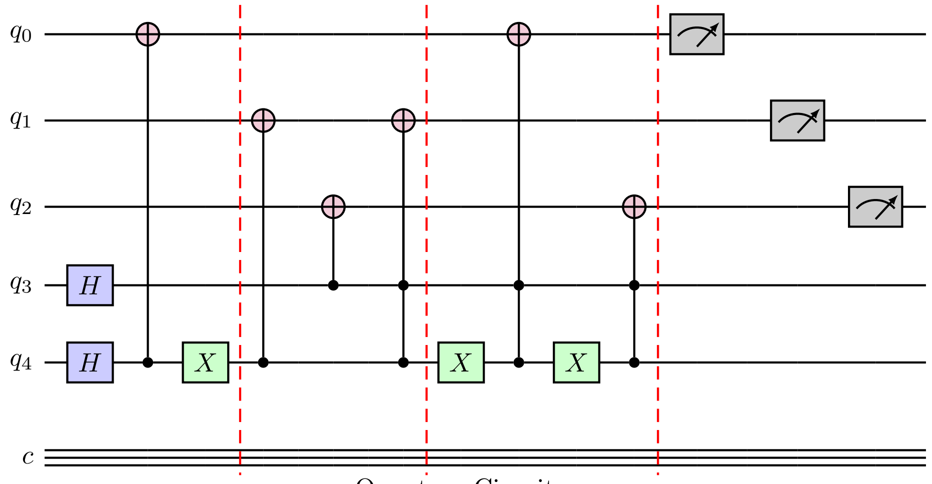


Figure 1: Steps in Basak-Miranda Implementation [5]



Quantum Circuit

- **Circuit Creation:** In the first step, we undertake a separate method qwalk for the circuit creation. Inside that method, initially, we declare the qubits, classical bits, and quantum registers. We consider the five qubits and three classical bits. Using these, we utilize the Hadamard(H), Pauli X-gate(X), Controller-Not(CX), Controlled-Controlled-Not(CCX).

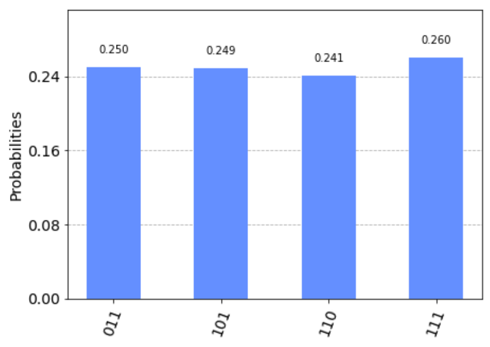


Figure 2: Histogram of Note generation

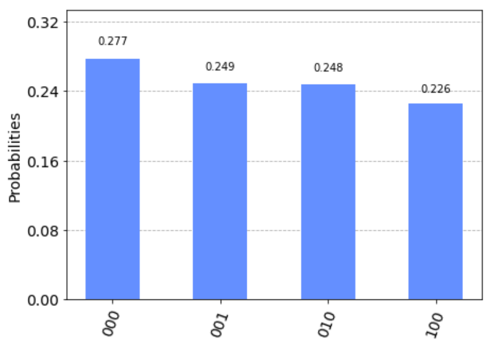


Figure 3: Histogram of Rhythm generation

We can observe we implemented a multi-qubit operation for our model from the gates.

- **Transpilation:** The second step of our implementation is developing the transpilation method to reuse our input circuit for approximating the architectural configuration of a quantum device.
- **Note Generation:** In the third step, we generate the note by running our circuit 100 times. For every 25 iterations, we induce the histogram to observe the pitches the simulation selected. Figure 2 depicted the histogram with the probabilities of the 100^{th} note generation.
- **Rhythm Generation:** The next step is the process of rhythm creation using the method we created earlier, we run the circuit 100 times to find out the rhythms, and for this operation, we generate the histogram with the corresponding probability for the state selection in figure 3.
- **Audio File Generation:** Finally, we use the built-in python module MIDIUtil [7] to create the audio file from the rhythms we got from the earlier step. The following action takes the pitch, duration, channel, and volume. Each iteration generates the note and adds the notes. Finally, generate the mp3 audio file, which we can run.

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: http://quantune.thepixel.men/

References

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

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

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

[4] 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.

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

[6] 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.

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

[8] 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.