Application of Grover's Algorithm on the ibmqx4 Quantum Computer to Rule-based Algorithmic Music Composition

Alexis Kirke

Interdisciplinary Centre for Computer Music Research, University of Plymouth, UK,
alexis.kirke@plymouth.ac.uk

Abstract: Previous research on quantum computing / mechanics and the arts has usually been in simulation. The small amount of work done in hardware or with actual physical systems has not utilized any of the advantages of quantum computation: the main advantage being the potential speed increase of quantum algorithms. This paper introduces a way of utilizing Grover's algorithm – which has been shown to provide a quadratic speed-up over its classical equivalent – in algorithmic rule-based music composition. The system introduced – qgMuse – is simple but scalable. It lays some groundwork for new ways of addressing a significant problem in computer music research: unstructured random search for desired music features. Example melodies are composed using qgMuse using the ibmqx4 quantum hardware, and the paper concludes with discussion on how such an approach can grow with the improvement of quantum computer hardware and software.

1. Introduction

Why are quantum computers (QC) attracting so much government and private funding? The answer is speed. Shor's algorithm (Shor 2006) is exponentially faster at breaking public key encryption than the fastest non-quantum algorithm. This is seen as a serious potential security threat (Perlner and Cooper 2009). Grover's algorithm (Grover 2001) is quadratically faster than the best classical algorithm at performing an unstructured-database search or function inversion.

Another feature of QC is its probabilistic nature. The non-deterministic nature of QC has an interesting conceptual implication for algorithmic artists. Artistic algorithms have utilized pseudo-random algorithms since the first computer arts were created, right up to some of the most recent creations. This is because randomness helps to prevent the algorithm producing overly repetitive output. Many computer artists prefer to use complexity algorithms rather than randomness, for example cellular automata (Kirke and Miranda 2007). However, at the heart of many of these systems is a pseudo-random choice still. The same parameters will create the same result. So the parameters of the complex algorithm are sometimes pseudo-randomized.

QC is not pseudo-random. The most prevalent interpretation of QC amongst researchers is that it is non-deterministic and has randomness at its heart. A quantum algorithm for which there is a desired deterministic result needs to be run multiple times to get a statistically significant final output. The final output is some averaging of all the intermediate outputs. Such a form of computation provides a new way of thinking about computer arts. Rather than trying to create complexity and randomness from determinism - as in classical computing, QC requires determinism and complexity to be built from randomness. The implications of this reversal of thinking for the arts are hard to imagine at this stage. These questions can only be answered by starting to apply basic quantum algorithms to the arts.

As already mentioned, two main potentially useful algorithms – Shor and Grover - have been identified, but neither has been implemented in hardware in a way that utilizes their quantum speed-up. The purpose of this paper is to develop and test a computer music algorithm qgMuse, that is implementable on a hardware quantum computer, and that does utilize QC as a solution – i.e. it aims to use QC to do things in a way a non-QC could not do.

The reality is that no quantum computer exists that can deal with useful versions of the quantum algorithms mentioned above. For example, the most powerful QC algorithm - Shor's Algorithm for factoring into primes - has been used on a hardware computer recently to factor 15 into 3 and 5 (Monz et al. 2016). This is a factorization that is trivial can clearly be done by hand. The only QCs which claim to be ready for commercial work are the quantum annealers mentioned earlier – made by D-Wave (Kirke and Miranda 2018). These quantum annealers cannot run Shor's or Grover's algorithm – the future "killer apps" of QC.

Faced with this, a computer music researcher may be tempted to simply drop all investigation into gate-based quantum computers - those that can run Shor and Grover - and focus only on quantum annealers, until more powerful and stable gate-based quantum computers are available. This view seems ignorant of musical history. Computer music began its "research" with simple bleeped tunes on early mainframes, and developed in parallel with the development of computing. This paper argues for a similar approach for gate-based quantum computer music. Much previous research in gate-based QC has been done in simulation or theory. There are a couple of exceptions - for example (Kirke 2018) - but none attempted to utilize the quantum algorithms known to give definite and large quantum speed-ups. Work needs to be done on actual quantum computers in these early stages, to ensure that quantum computer music keeps pace with advances in quantum computing, and also to see exactly what is feasible on a quantum computer in computer music terms. Furthermore by reporting such work, it will help to develop a knowledgebase within the arts community.

2. Related Quantum Music Work

Previous designs for performances and music involving quantum mechanical processes have either been metaphorical, based on simulations (online or offline), not utilized the quantum speed-up, or - in the case of actual real-time physics performances - not been directly concerned with quantum effects. It is important to take a moment to define what is meant in this paper by "utilizing the quantum speed-up". There are no algorithms on gate-based quantum computers available that perform tasks faster than a classical computer. The killer quantum computer algorithms have been proved to be faster only in theory. The speed increases, however, are so vast that this has led to the large amount of money being poured into quantum technology research. Furthermore, the implementations of the algorithms on current gate-based quantum computers are at a level where the problems they solve are trivial - for example searching a database of sixteen 1-bit entries, or showing that 15 can be written as 3 times 5. However these implementations are theoretically scalable. Thus when this paper refers to a system "utilizing the quantum speed-up", it means: (a) that the system is based on a quantum algorithm

that has been proved to be theoretically much faster than its classical counterpart, and (b) that the system is in theory scalable so that even if it is a trivial example, it could eventually incorporate examples that will run faster than their classical counterpart.

In terms of offline simulations, one of relevance to this paper is the web page Listen to the Quantum Computer Music (Weimer 2014). Two pieces of music are playable online through MIDI simulations. Each is a sonification of the two key quantum computation algorithms: Shor's and Grover's. The offline sonification of quantum mechanics equations have also been investigated in (Sturm 2000; Sturm 2001) and (O'Flaherty 2009), with the third sonifying LHC data from CERN to create a musical signature for the (at-the-time) undiscovered Higgs Boson. Another paper defines what it calls Quantum Music (Putz and Svozil 2014) in simple theory form, though once again this is by analogy to the equations of quantum mechanics, rather than directly concerned with quantum computing. It examines what one might call the "trivial" representation in quantum music. Each note in a melody is a superposition of all possible notes. It has not been implemented on a hardware QC. It would need to be mapped into the QC realm and then into the hardware QC realm. After that, working with, say, two melodies that are superpositions of 8 notes each – for example entangling them - would require significant circuit complexity. Current hardware quantum computers would not be able to cope with them, purely from a stability point of view (as will be seen later). Most importantly, even if it could be implemented, the presented formalism does not utilize the quantum speed-up.

Certain equations of quantum mechanics have also been used to synthesize new sounds in simulation (Cadiz and Ramos 2014). The orchestral piece "Music of the Quantum" (Coleman 2003) was written as an outreach tool for a physics research group, and has been performed multiple times. The melody is carried between violin and accordion. The aim of this was as a metaphor for the wave particle duality of quantum mechanics, using two contrasting instruments.

The most impressive quantum simulation performance has been Danceroom Spectroscopy (Glowacki 2012) in which quantum molecular models generate live visuals. Dancers are tracked by camera and their movements treated as the movement of active particles in the real-time molecular model. Thus the dancers act as a mathematically accurate force field on the particles, and these results are seen in large scale animations around the dancers.

There have been performances and music that use real-world quantum-related data. However most of these have been done offline (not using physics occurring during the performance). These include the piece Background Count: a pre-recorded electroacoustic composition that incorporates historical Geiger counter data into its creation (Brody 1997). Another sonification of real physics data, but done offline, was the LHChamber Music project (Hetherton 2014). It was instrumented for a harp, a guitar, two violins, a keyboard, a clarinet and a flute. Different instruments played data from different experiments. Flute and guitar were CMS, Clarinet and Violin I were ATLAS, Violin II was LHCb, Piano was ALICE, and harp was CCC.

The first real-time use of subatomic physical processes for a public performance was Cloud Chamber (Kirke et al. 2011). In Cloud Chamber physical cosmic rays are made visible in real-time,

and some of them are tracked by visual recognition and turned in to sound. A violin plays along with this, and in some versions of the performance, the violin loudness level triggered a continuous proportional electric voltage that changed the subatomic particle tracks, and thus the sounds (creating a form of duet). Cloud Chamber was followed a few years later by a CERN-created system which worked directly, without the need to use a camera. Called the Cosmic Piano, it detects cosmic rays using metal plates and turns them into sound (Culpan 2015). However it had no feedback loop from the acoustic instrument to the cosmic ray tracks, unlike Cloud Chamber.

The previous two discussed performances were live, and the data was not quantum as such. It was quantum-related in that the cosmic rays and cloud chambers are subatomic quantum processes. But the performances do not incorporate actual controlled quantum dynamics or computation in their music. (Kirke et al. 2015) created sound with quantum computing but was primarily about connecting other forms of unconventional computing (PMAP) to a quantum computer. It was designed for use with an online photonic quantum computer, however for technical reasons the computer was taken offline, and so the final results were generated using the online simulator. The paper included the use of the system to compose an orchestral piece of music that musified a photon-based quantum gate called a CNOT, approaching maximum entanglement.

The first use of controlled quantum dynamics in hardware quantum computation to make music was the algorithm qHarmony (Kirke and Miranda 2017). It was implemented on an adiabatic quantum computer and also utilized in real-time in a live music performance with a mezzo-soprano (Kirke 2016). The first use of gate-based quantum computer hardware to make music was the algorithm GATEMEL (Kirke 2018; Kirke 2018b). This algorithm does not utilize the quantum speed-up but only the non-deterministic nature of QC. A second algorithm tested on a gate-based hardware QC that utilizes the non-determinism is Quantum Music Composer (Weaver 2018), which is a step on from GATEMEL in that it implements a Markov chain and harmonies, but does not utilize the quantum speed-up.

3. Rule-based Algorithmic Music Composition

The quantum algorithm introduced in this paper is utilized to support rule-based algorithmic composition. The use of rule-based or knowledge-based methods for algorithmic composition have been common for many years (Papadopoulos and Wiggins 1999). In such an approach, the composer/user predefines a set of rules that can generate or constrain musical features. One of the first algorithmic compositions, the Illiac Suite (Hiller and Isaacson 1957) involved randomly generating notes and then dropping notes which did not fit the rules of certain composition styles – for example textbook counterpoint for the second movement. Since then rule-based systems have developed where the rules can be used to partially or fully generate the actual music features.

Rules can be applied in a bottom-up or top-down approach. For the bottom-up approach the rules are the generative engine themselves. For example, let $r(t)$ be a function that generates a pseudo-random non-negative integer at time t. Then a rule to generate an even numbered musical feature at time t would be $F(t) = 2r(t)$. Or

rules to generate two pitches $p(t)$ and $p^*(t)$ with 5 semi-tones difference would be:

$$p(t) = r(t) \ mod \ 12 \quad (1)$$
$$p^*(t) = p(t) + 5 \quad (2)$$

Now consider the top-down approach. In these cases a feature is generated - usually pseudo-randomly - and then checked against the rule. For example, to implement the rule $F(t) = 2r(t)$ as a top-down rule, a random number R can be generated. Then it can be factored to examine if there exists an integer n such that $R = 2n$ to fulfill the rule. Or for an intervallic example, if two pitches $p(t)$ and $p^*(t)$ are randomly generated, it can then be checked if they fulfil the rule:

$$|p(t) - p^*(t)| = 5 \quad (3)$$

This paper is inspired by the top-down rule approach. Such rules can be highly contextual as well – for example the allowed pitch distance between adjacent or coincident notes could be constrained based on the previous 5 distances looking back in time. Rules can cross reference each other – the allowed adjacent pitch distance could be limited by the allowed coincident distance. The compositional *style* in such a top-down rule-based system usually comes not only from the individual rules, but how they are logically combined. For example, suppose two sub-rules are defined for randomly generated pitches:

$$|p(t) - p(t-1)| > 0 \quad (4)$$
$$p(t) - p(t)^* > 1 \quad (5)$$

Two of the possible methods of combining these sub-rules to make a rule could be:

$$(|p(t) - p(t-1)| > 0) \cdot (p(t) - p(t)^* > 1) = 1 \quad (6)$$

$$(|p(t) - p(t-1)| > 0) \oplus (p(t) - p(t)^* > 1) = 1 \quad (7)$$

The first version ANDs the sub-rules - it only gives value 1 if both sub-rules are satisfied. The second version XORs the sub-rules - it only gives value 1 if only one of the sub-rules is satisfied. These are clearly going to give significantly different musical outcomes. It is this Boolean approach to rule specification that is usually taken in top-down rule-based systems. A significant part of the generative process in such systems is solving equations such as (6) and (7) to check if the features satisfy the equations. A simple generate-and-check approach is generally considered naive. When the number of sub-rules and their combinations grows - as is required for musically interesting and relevant systems - the generate-and-test approach becomes too slow.

For example, the groundbreaking CHORAL system (Ebcioglu 1988) used 350 rules in a Boolean-type form. These rules are designed to capture the style of J. S. Bach for four-part harmonies. Once these levels of complexity are reached, simple generate-and-test is unfeasible, and methods such as constraint programming and backtracking are used (Anders 2018) to speed up the search for musical solutions. At this point the rules become labeled as constraints. The musical problem in general is in fact so complex, that musical constraint programming has at times helped to drive research in general constraint programming. However, there is a key assumption behind the use of constraint programming in rule-based algorithmic composition that makes quantum computing relevant. This assumption is the speed of unstructured random search. Boolean equations (6) and (7) are extremely simple, and can be

solved by eye. In general a Boolean equation of sub-rules sr_i could be more complex, such as:

$$\begin{aligned}&\left((sr_1 \cdot sr_2 \cdot sr_3{}') \oplus (sr_2{}' + sr_4) \cdot sr_3\right)\\ &+\left((sr_1 \cdot sr_5{}' \cdot sr_6) \oplus (sr_6 + sr_8)\right) +\\ &\left((sr_9 \cdot sr_{10}{}' \cdot sr_{11}) \oplus (sr_{10} + sr_{12})\right)\\ &= 1 \qquad (8)\end{aligned}$$

(Note that the apostrophe ' is a logical NOT operation). To find the truth values of the sr_i by eye is not simple. Another option for solving equation (8) is unstructured random search. In this case all possible values of the sr_i are tried (0 or 1) until the solutions are found. This takes, in the worst case, 2^{12} = 4096 iterations. Easily do-able on a fairly old desktop in a trivial amount of time. But as the number of sub-rules grows, the number of average iterations needed to find a solution grows rapidly. If there are 30 sub-rules, then the number of iterations required to find the solution is, on average, $(2^{30})/2$ = 536,870,912. If each calculation of an iteration takes a 10th of a millisecond, that would be about 16 hours to generate one time-step of the music features in the rule-based system. As already mentioned, such a naive approach is not usually used.

However, suppose one can create a search algorithm that is quadratically faster at unstructured random search – i.e. rather than taking *N* iterations it took $\sqrt{N}$ searches. For a system that is quadratically faster, the 30 sub-rule example above would take under an hour. If there were 40 sub-rules, a quadratic speed up would be relatively more dramatic. Such a fast unstructured random search could have a significant effect on top-down rule-based composition, given that an underlying assumption of much of much previous research is that unstructured random search and generate-and-test is too slow. Such a quadratically faster search does theoretically exist and can be implemented, currently for a much smaller number of sub-rules, but in a scalable way. The purpose of this paper is to introduce a small sub-rule system in a scalable way, and implement it on hardware.

The fast search requires that *the Boolean function can be implemented on a quantum computer*. It is the already-mentioned Grover's algorithm (Grover 2001). The fact that Grover's algorithm is implemented on a quantum computer will lead to it being both generative and soft - which will be seen to be useful features of the system introduced later in this paper. Each time Grover is run and collapsed, it generates a suggested solution to the Boolean function, with those that satisfy the rule having the highest probability of being generated. Suppose there are, say, 1,000 correct solutions to a combined rule, and 1 billion incorrect solutions. Then Grover will - most of the time - randomly select one of the correction solutions, but with a lower probability it may output one of the incorrect values. This type of feature is desirable in soft rules, and such soft rules are sometimes desirable in computer music systems (Anders 2018).

4. Grover's Algorithm

At the heart of Grover's algorithm is a representation of the Boolean function to be inverted – called the Oracle. Usually a quantum form of the function to be solved. In the case of rule-based composition, it represents the rule to be solved. The first step is to define the rule inputs as qubits. So if the rule has three binary inputs, then the classical version could be written:

$$r(i_0, i_1, i_2) = 1 \qquad (9)$$

The quantum version is written:

$$R|i_0 i_1 i_2\rangle \text{ or } R|q_0 q_1 q_2\rangle \quad (10)$$

since the inputs will have to be represented as qubits and the rule as a quantum operator *R*. In the algorithm, the inputs [i_0, i_1, i_2] are initialized to 0-valued qubits, giving

$$|q_0 q_1 q_2\rangle = |000\rangle \quad (11)$$

Next in the Grover algorithm, the qubits are put into a superposition of all their possible values of [i_0, i_1, i_2] (these are 000, 001, 010, 011, 100, 101, 110, 111). This is done using three "parallel" *H* gates. Applying them across the 3 qubit state in (11) (and ignoring the scalar multipliers for brevity) will give:

$$A = H^{\otimes 3}|000\rangle = |000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle \quad (12)$$

The notation $H^{\otimes 3}$ represents the fact the *H* is acting on a three qubit state. Next the Oracle operation *R* is performed on this superposition *A*. The Oracle must be represented as a unitary matrix that multiplies the superposition:

$$B = R(A) = R(|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + + |110\rangle + |111\rangle) \quad (13)$$

$$= R|000\rangle + R|001\rangle + R|010\rangle + R|011\rangle + R|100\rangle + R|101\rangle + R|110\rangle + R|111\rangle \quad (14)$$

The Oracle has one key function: it must flip the phase of the part of the superposition *A* in equation (12) that represents the correct solution to the rule - i.e. for which $r(i_0, i_1, i_2) = 1$. This can be done by calculating a quantum version of *r* in such a way that the state for which *r* is satisfied has a negative phase. Suppose the correct solution to $r(i_0, i_1, i_2) = 1$ is 0,1,0. Then applying *R* in equation (48) should give:

$$B = |000\rangle + |001\rangle - |\mathbf{010}\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle \quad (15)$$

Next comes the key part. A quantum function is applied, that moves the superposition of (i_0, i_1, i_2) towards a value that, if measured, now has a higher probability of giving $r(i_0, i_1, i_2) = 1$ (the desired result). The operator that does this is called the Grover diffusion operator, and is (in the 3 qubit case):

$$R_0 = H^{\otimes 3}\,(2|000\rangle\langle 000| - I)H^{\otimes 3} \quad (16)$$

I is the identity matrix. Applying the R_0 operator to the superposition *B* in equation (14) has one key effect. It inverts the weightings of *B* around their mean. The mean of the weightings will have been reduced by making the solution states have negative weightings, so inverting all weightings around the mean will increase the size of the negative weighted items. These are the items which are solutions to equation (9). The combination $R_0 R(A)$ can also be thought of as a rotation of the superposition in a complex vector space in such a way as to move the superposition vector *A* of equation (12) towards the solution. These operations amplify the probability of observing the (i_0, i_1, i_2) which satisfy equation (9). This amplification is very rapid. After less than three applications of Grover (in the 3 input rule case), if the quantum state of the Grover output is observed, then the most likely result will be the values (i_0, i_1, i_2) that give r(i_0, i_1, i_2) = 1.

The fact that quantum algorithms are

probabilistic raises another issue. After less than 3 iterations of a 3 qubit input Grover's algorithm, although the most likely observation will be values of (i_0, i_1, i_2) that obey (9), there is a non-zero probability of measuring the wrong values. This means that if *r* is meant to be a "hard" rule, a quantum algorithm needs to be sampled multiple times in order to get the result required. However observing the output of a quantum algorithm leads to a collapse of the superposition. Thus the algorithm needs to be run again to get another sample output. This can - of course - cancel out the quantum speed up for hard rules. For example, using only 3 inputs as above, the quantum algorithm might have to be run 20 times on the ibmqx4 to give confidence in the result. Then in real terms 40 iterations have been performed, rather than the up-to-8 iterations with the classical version**.** However this disadvantage quickly disappears as the number of inputs increases, given the speed-up factors demonstrated earlier. Also, if a soft rule is desired, then the user may be happy to select the first output given.

5. Implementing Rules on a Hardware QC

Grover's algorithm was introduced in the context of larger-scale rule based systems such as CHORAL, that use constraint programming or advanced Boolean solving tools to deal with the larger numbers of rules and inputs. Using a Grover of this scale would be too complex to provide an introductory example or implement on hardware. Hence a small-scale Grover is utilized in this paper to build the foundations of scalable approach.

5.1 qgMuse

qgMuse is a scalable hybrid hardware classical/quantum algorithm implemented using an IBM quantum computer. qgMuse implements two rules, partly inspired by the Narmour Implication-Realisation (I-R) model (Narmour 1992). It is designed to demonstrate a scalable quantum model of composition that will be able to utilize practical speedups from Grover's algorithm once they become available. It is done in the spirit of Monz et al. (2016) which used Shor's quantum algorithm to factor 15 into 5 and 3 on quantum hardware - as a vital and scalable link in quantum computer music research. The fact Shor's algorithm can be used in a simplified form on current QC hardware, is a major step towards using it in situations where the quantum speed-up becomes relevant. Similarly with qgMuse: although it is implemented in a way that doesn't require a quantum speed-up, it lays the framework for a future computer music system based around the same algorithm, that does.

The Narmour I-R model is a proposed approach utilizing soft rules for the behavior of sequences of musical notes. One of the I-R "rules", and one of its assumptions will be interpreted into soft Boolean rules. Firstly the I-R "Registral direction" rule, which states that large pitch intervals tend to precede a pitch direction change, whereas small intervals tend to be continued in the same direction. A large interval will be defined here as spanning 4 white piano notes or more. So the pitch jump c -> f is large, whereas c -> e is small. Or f -> b is large, whereas f -> a is small. qgMuse only uses white notes here as it can only have a limited number of rules for practical quantum implementation. Hence the tonality is restricted at the outset, rather than

implementing it on the QC hardware as a rule. The large interval flag $LI(t)$ will be defined as 1 if the number of white notes of the current pitch to the previous pitch at $t-1$, including both end pitches, is 4 or more, and $LI(t)$ is 0 otherwise. Define the direction change flag $DC(t)$ as 1 if the interval at t is in the opposite direction to the interval at $t-1$, and $DC(t)$ is 0 otherwise. Then consider the XOR-based rule:

$$(LI(t-1) \oplus DC(t))' = 1 \qquad (17)$$

This is satisfied if $LI(t-1)$ and $DC(t)$ are equal, i.e. if a large interval is followed by a direction change, or if a small interval is followed by a no direction change. So if this is implemented as a fuzzy or soft rule, it is compatible with the "Registral Direction" tendencies. The following rule will also be softly implemented:

$$LI(t-1)' \cdot LI(t)' = 1 \qquad (18)$$

which says that all intervals are small (non-large) intervals following small (non-large) intervals. Such a rule would not be useful in a hard-rule generative music system, but is useable in a soft-rule system to make small intervals dominate over large ones - an assumption in Narmour I-R approach. The two rules between them use 3 variables, which requires a 3-input Grover to solve equation (53), made up from the two sub-rules above:

$$\begin{aligned}&\big(LI(t-1) \oplus \quad DC(t)\big)'.\\ &\quad LI(t-1)'.LI(t)' = 1 \qquad (19)\end{aligned}$$

The quantum part of qgMuse is shown in Figure 2. Figure 2's circuit solves the rule $(LI(t-1) \oplus DC(t))'.LI(t-1)'.$ $LI(t)' = 1$ using Grover's algorithm. The oracle is marked up with three rectangles (1), (2) and (3). Everything after the markups is part of the Diffusion operation, and then measurement of the outputs.

$LI(t), DC(t),$ and $LI(t-1)$ are represented by q_0, q_1 and q_2 respectively. The qubit "inputs" are all initialized to $|0\rangle$. Rectangle (1) uses a CNOT gate to take the XOR of q_2 and q_1, represented on q_1. It then uses an X gate to take the NOT of q_1. Thus q_1 now represents $(LI(t-1) \oplus DC(t))'$. The X gates on q_0 and q_2 perform a NOT on q_0 and q_2 to represent $LI(t-1)'$ and $LI(t)'$ respectively. Rectangle (2) implements a control-control-Z (*ccz*) gate, specifically $ccz(q_2, q_1, q_0)$. This can be seen by multiplying out the matrices of the operations, but will not be demonstrated here for space reasons.

Given that q_1 represents $(LI(t-1) \oplus DC(t))'$, q_2 represents $LI(t-1)'$ and q_0 represents $LI(t)'$ then - by the definition of $ccz(|q_0q_1q_2\rangle)$ - $|q_0q_1q_2\rangle$ has its phase flipped if $q_0 = 1$ and $q_1 = 1$ and $q_2 = 1$. In other words, if $(LI(t-1) \oplus DC(t))' = 1$ and $LI(t-1)' = 1$ and $LI(t)' = 1$. Thus, as required, the Oracle represents equation (19). Rectangle (3) simply reverses the operations performed in Rectangle (1) before going through the Diffusion operations, which need q_0 and q_1 to have the same values that had before rectangle (1). The Diffusion operations represent R_0 in equation (16). When this circuit has its output observed, it should provide, with the highest probability, solutions to equation (19).

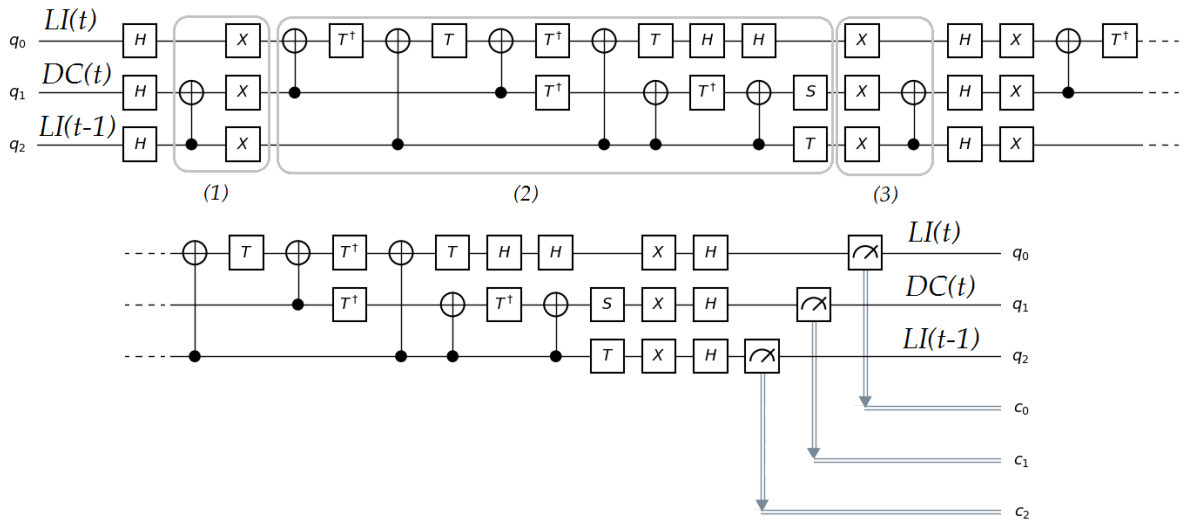

Figure 2: Quantum circuit implementing the rule $(LI(t-1) \oplus DC(t))'.LI(t-1)'.LI(t)' = 1$ on the ibmqx4

The non-quantum part of the qgMuse works as follows. It starts on a base note – middle C will be used - at time step $t = 0$. Then it runs the quantum circuit to invert equation (19) - to get the allowed values of $LI(t-1)$, $DC(t)$ and $LI(t)$. Backtracking is not utilized in qgMuse, so if the allowed $LI(t-1)$ from the QC is different to the actual $LI(t-1)$ from the previous time step, then the whole generation process is skipped and new candidates are requested from the QC. In other words if the previous white note interval was large but the quantum algorithm returns a solution involving the previous interval being small - or vice versa - then qgMuse calls the quantum part of the algorithm again. If the rules were hard rules, then the returned solution would not change on this second call. But for the non-deterministic Grover, the solutions can change.

If the generation continues (as opposed to requesting new candidates from the QC) then a second set of checks are done. If the returned solution for $LI(t)$ is 1 (i.e. a large interval is required) then the generated white note interval will be limited to a size of 8 (including both notes in the interval), otherwise it will be limited to a size of 3. If the returned solution for $DC(t)$ is 1 (i.e. a direction change is required) then a white note interval is generated between -8 and 8 (for large interval allowed) or between -3 and 3 (for large interval not allowed). Otherwise the generated interval will be in the same direction as the previous generated interval. The limiting of large intervals to white note size 8 is arbitrary, allowing for octave jumps at the most.

It can be seen that this implementation of the Grover approach in qgMuse is not generate-and-test. It would be more accurately described as solve-and-generate. The future quantum increase in speed would come from solving the Boolean equation. The process of turning the solution into pitches is implemented classically. This requires thoughtful designing of the sub-rules. The easy case is when the returned solution *cannot* be satisfied by the music generated thus far - in which case the Grover can be called again. Suppose a solution is found, and it involves returned values from the QC a large number of variables for which

Grover is relevant - say 30 variables. Then depending on the nature of these variables, a classical rule-based search may need to be run, to find solutions that give the allowed values for the variables. This could involve constraint satisfaction techniques. Thus the qgMuse approach does not replace constraint-based approaches, it provides a possible method for redesigning such approaches to speed up the parts that could involve generate-and-search sub-processes. For example, suppose the qgMuse quantum element returns 1, 1 and 0 for $LI(t), LI(t-1)$ and $DC(t)$. This could be viewed as setting up another rule:

$$(LI(t) = 1) \cdot (LI(t-1) = 1) \cdot (DC(t) = 0) = 1 \quad (20)$$

or

$$LI(t) \cdot LI(t-1) \cdot DC(t)' = 1 \quad (21)$$

This is simpler to solve than the full combined rule defined earlier as:

$$(LI(t-1) \oplus DC(t))'. LI(t-1)'. LI(t)' = 1 \quad (22)$$

It is so simple that it is easily implementable programmatically, as done in this version of qgMuse. Also - as has already mentioned - the quantum approach provides "softness" for free. It is worth mentioning at this point that the softness is not as controllable as that available in some soft rule-based systems. A classical soft rule-based system could use defined probability distributions to ensure that the softness is controlled. Whereas the softness produced by Grover is a function of the quantum indeterminacy and errors in quantum hardware - a less controllable combination. But in terms of speed it is "instant" - it does not require probability distribution or pseudorandom algorithms - the quantum mechanics provides it instantly.

6. Results

Two runs of qgMuse of eight 4/4 bars each on the ibmqx4 quantum computer are shown in Figures 3 and 4. Looking at the melodies it is not clear how well the rules are being followed. To give further insight, Figure 5 shows a plot of the output of the Grover algorithm in Figure 2 sampled 4096 times. IBM's Qiskit API converts the qubit observations to hexadecimal. So in Figure 5, 0x0 is 000 binary - i.e. all qubits are observed as 0. Similarly 0x7 is 111 binary - i.e. all qubits observed as 1, and 0x5 is binary 101 - i.e. q_1 is observed as 0, but q_0 and q_2 are observed as 1. The solution of equation (19) can be seen by eye to be: all variables are 0. In the quantum world, this means that the most likely result of a measurement is all variables as 0. Figure 5 shows that the ibmqx4 returns all 0 approximately 52% of the time. The other 48% of the time, the incorrect values will be returned. It was initially desired that for soft rules, correct values be returned only most of the time. However a likelihood of 52% for a correct value, and 48% for a random one, is probably lower than desired for most soft rules. However, running the Grover 4096 times on the IBM online simulator gives Figure 6. This shows that the correct results are returned approximated 77% of the time, with a roughly equal spread across the incorrect results - totalling up to around 23%. This demonstrates that of the approximately 48% of incorrect results in the ibmqx4, around half of them are due to quantum hardware errors, rather than designed quantum effects. 23% error in soft rules seems a more reasonable figure - and is

approximately what the ibmqx4 would give if it was a perfectly engineered quantum computer (i.e. more like the simulator).

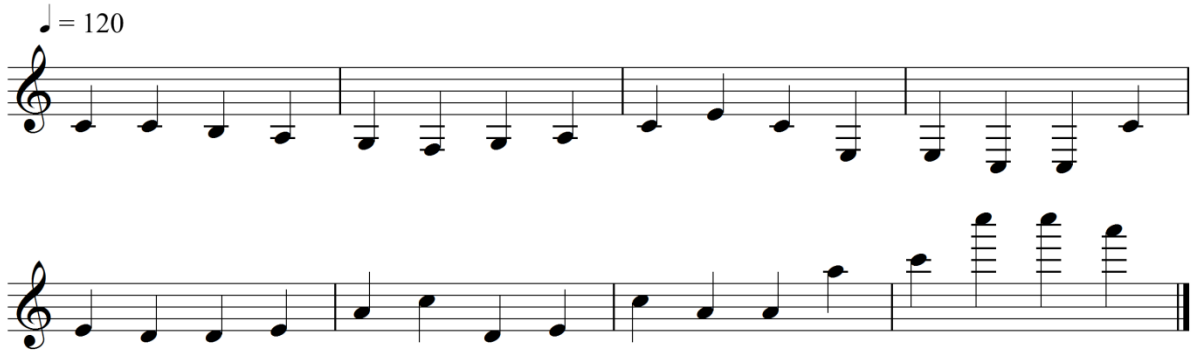

Figure 3: 8 bars of qgMuse running on the IBM quantum hardware - Melody A

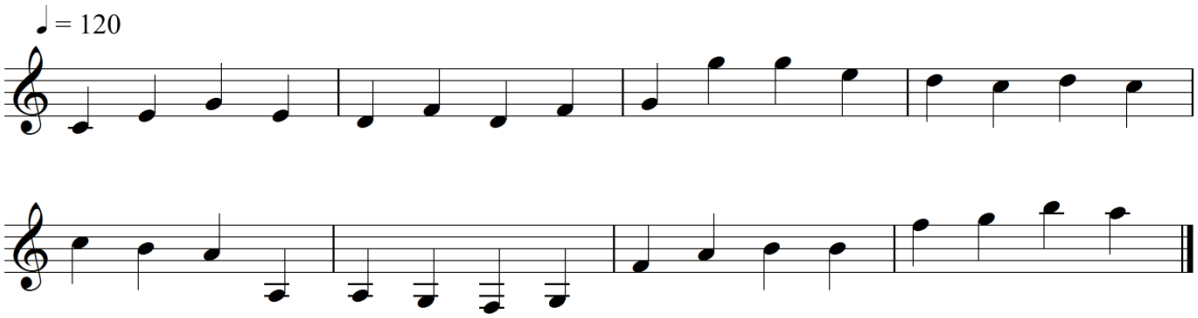

Figure 4: 8 bars of qgMuse running on the IBM quantum hardware - Melody B

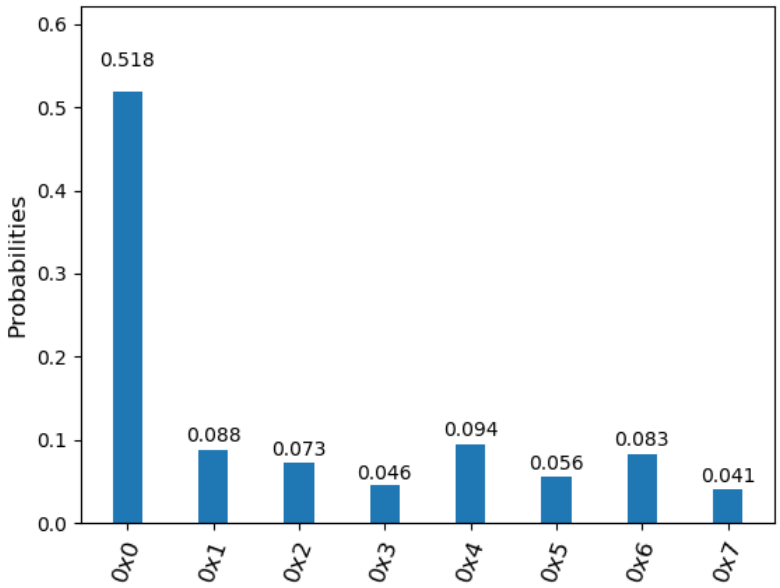

Figure 5: Grover rule outputs shown for the IBM quantum hardware ibmqx4

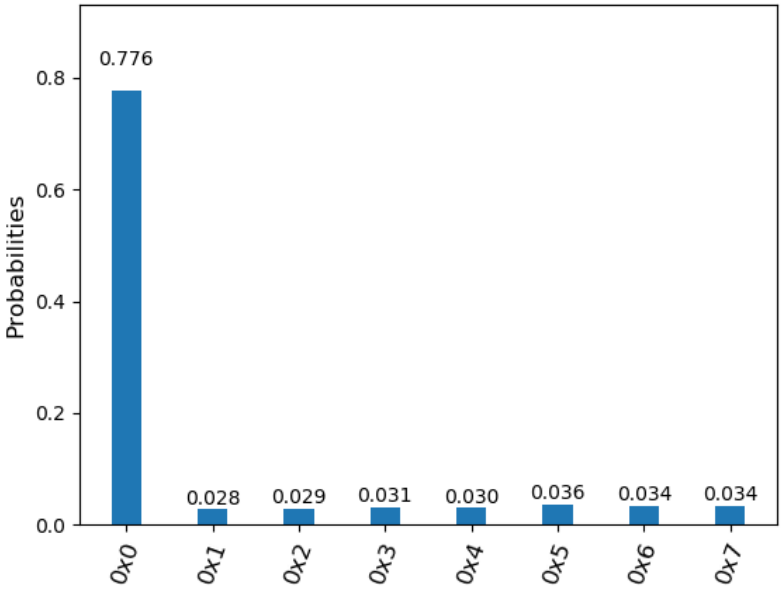

Figure 6: Grover rule outputs shown for the IBM online simulator

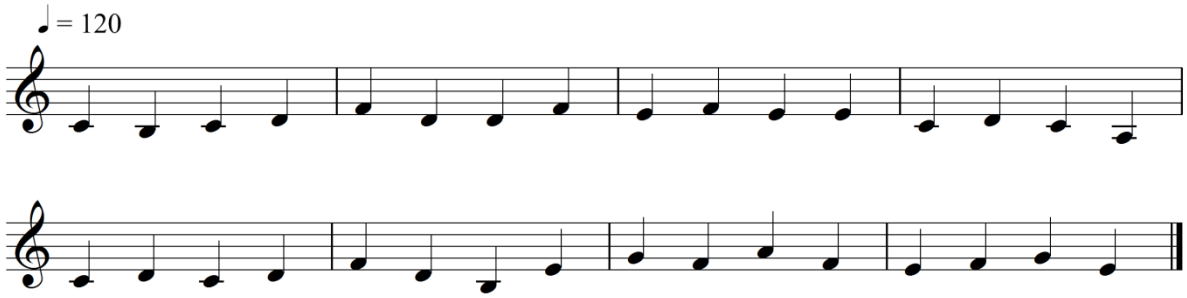

Figure 7: 8 bars of qgMuse running on the IBM quantum simulator - Melody C

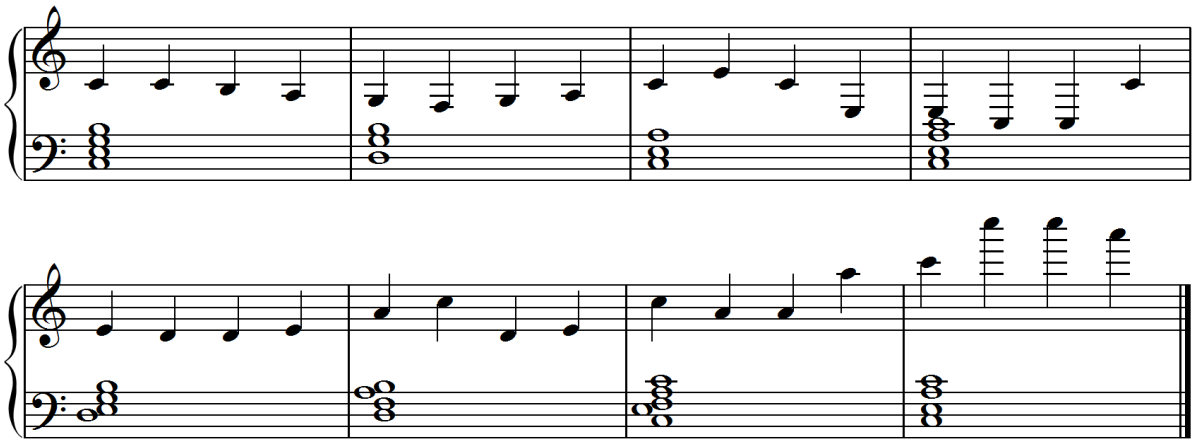

Figure 8: The melody in Figure 6 - Melody A, harmonised by an independent harmony algorithm qHarmony on a DWave 2X

Thus given that Figures 3 and 4 should only exhibit 52% correct behaviour, it is not feasible to use them to evaluate the statistical results. Generating larger and larger tunes to increase statistical clarity is pointless - Figure 5 (together with the deterministic behaviour of the classical computer code part of qgMuse) fully characterises qgMuse, and Figures 3 and 4 simply prove the feasibility of a musical output. However, the Figure 7 score of the online quantum simulator running qgMuse does allow some judgement by eye of its musical output (in terms of rule-following). The vast majority of intervals in Figure 7 are non-large / small intervals - showing one sub-rule (equation (18)) is having an impact in a way that was not so clear in the hardware results. However - as would be expected - the sub-rule of direction changes coinciding with a previous large interval (equation (17)) is not obviously visible.

It has been observed in the past that without some independent harmonic context, algorithmic melodies are difficult to evaluate (Papadopoulos and Wiggins 1999). Figure 8 shows the run of 8 bars output from Figure 3, this time with an independently generated harmonic context. The chords are generated using qHarmony - an algorithm for generating simple white note harmonies without any temporal context. It takes as input one or two white notes, and outputs a white note chord to play with the notes. In this case two notes - the first and third note of each bar - are sent to qHarmony to select the chord for the whole bar. It does not use context of the previous or following chords. It is interesting to note that qHarmony is also run here on quantum hardware - it was in fact the first quantum hardware music algorithm - but on a very different system to the ibmqx4. It is a D-Wave 2X adiabatic quantum computer, referred to in the introduction to this paper as a quantum annealer. The algorithm is beyond the scope of this paper, but more information about qHarmony and adiabatic quantum computing can be found here (Kirke and Miranda 2017). qHarmony has previously been utilized in both live performance (Kirke 2016) and offline testing. There is one slight artistic license taken with the harmonies in Figure 8. When qHarmony returns a harmonisation - it actually returns a selection of harmonisation solutions, together with their energy level for each. In theory, the lower the energy level, the better the solution. Bars 7 and 8 both returned the same solution chords at the lowest energy level. To create a small sense of movement between the penultimate bar and the final bar, the second lowest energy (energy -54) solution for Bar 7 was selected instead of the lowest energy solution (energy -56).

The tunes sound reasonably presentable, including the harmonized version. Some of this is due to the fact of the implicit rule of qgMuse - using only piano white notes. However a high level of musical quality is not expected from the limited sub-rule-set size feasible on current hardware QCs.

The results of qgMuse above support the idea of the non-deterministic nature of quantum computers being a potential "feature rather that a bug" - in terms of implementing soft rules. Furthermore, Figures 3 to 7 are consistent with the idea that the rules are being followed the majority of the time. Given the usage of a quantum computer with a similar level of hardware error (in the final outputs), but with a much larger number of qubits, then qgMuse can be scaled in a simple way. The number of sub-rules can be increased. This can be done with conjunction or disjunction - i.e. by ANDing the rules (as

described earlier) or ORing them, or a combination (a quantum OR can be built using *x* gates and *ccx* gates). Then each Boolean input can be assigned to a qubit.

Note that the proviso above about the level of error in the final outputs is key. Just increasing the number of qubits and the size of the Boolean functions, at the current level of gate error, will create an essentially random output. The input / output error refers to this: given a certain input - and given the resulting output - how far away are the outputs from the outputs of a perfect simulator? To achieve the same level of error as seen in Figure 5 for 100 gates is beyond current quantum technology. However if the gate errors can be kept small enough, so that perhaps 500 gates can be combined with the same level of input / output error as seen in Figure 5, then more interesting musical problems become possible using Grover. Given this level of error, *exactly* the same methods can be used as used in this paper, the main challenge becoming - how to implement Oracles for the various musical rules.

Of course, at the current level of simplicity of qgMuse required by quantum technology, the whole rule-based process in qgMuse process could have been done trivially. Rather than running the Grover, the results of the randomly generated notes and of calculating *LI* and *DC* could have been inserted into the classical Boolean equations. If they failed to fulfil those equations, then 75% of the time (say, to allow for softness) the proposed interval could have been ignored and a new one generated. This approach works fine until the number of equations and inputs grows large. Then various heuristics and optimisation methodologies need to be introduced. Gradually as the number of rules and inputs grows, the result becomes very slow to generate on a classical system. At this point the problem moves into the domain where a Grover-assisted algorithm using the ideas of qgMuse, on a sufficiently powerful future quantum computer, could have an advantage.

The reality is that a future quantum computer music top-down rule-based system will probably be a hybrid between constraint programming and quantum computer unstructured search. The quantum part will be the Grover methods used in this paper, but implemented in more error-tolerant ways and utilizing quantum error reduction algorithms (Singh et al. 2018). But whatever the precise balance and approach used, the basic structure of a Grover solving of a Boolean rule will remain the same, as described in this paper. It can in fact be proved that Grover's algorithm is optimal for unstructured search (Zalka 1997).

7. Conclusions

In evaluating the design and results of qgMuse, context is key. For example, qgMuse is slower than a fully classical version. The quantum advantage of Grover is superseded by the simplicity of the rules, and the small number of inputs. A classical version will be faster, as accessing the small number of quantum computers sometimes involves a queue time, and obviously running the simulator is slower than the classical equivalent. Similarly, from a musical point of view, there are only two simplistic rules being used, that are applied to a musical context of one previous note and a small number of music features.

Thus the evaluation context is as follows: (a) Shor's algorithm is currently only factoring numbers of the order of 15; (b) qgMuse is scalable as a concept; and (c) qgMuse has a natural soft non-deterministic nature. Examining the second point, which is overwhelmingly the

most important: suppose there is a conjunction of 150 Boolean musical sub-rules with 30 inputs, similar to that discussed earlier. Finding solutions to that conjunction by classical unstructured search would be unfeasible, and may never become feasible given the limitations of Moore's law (Waldrop 2016). However quantum computing is moving forward at a significant rate in 3 key areas: number of qubits, stability of processing, and error reduction methods. When the 16 qubit IBM Melbourne was released, it came rapidly on the heels of the 5 qubit ibmqx4. Other companies claim to have QCs with greater than 60 qubits, but many are not publically available to use yet. The instability of hardware quantum computers is not a new issue, and there have been many years of research (Lidar and Brun 2013) into how to reduce the susceptibility of quantum computing to errors such as those encountered in this paper, including on the ibmqx4 itself (Singh et al. 2018). In the next couple of years, someone may publish an error tolerant version of the 3-input Grover for the IBM q 16 Melbourne. It is not unfeasible that in 4 years the IBM q 16 Melbourne will be superseded by a much more stable q 30, with versions of the Grover algorithm that allow input and processing juggling on the machine leading to the use of perhaps 10-20 inputs. There are researchers who say that the demonstrating of quantum supremacy over classical computers in hardware for certain algorithms is only a matter of years away (Harrow and Montanaro 2017). Though it is not clear yet how this might relate to timescales for a similar demonstration with Grover's algorithm.

The ibmqx4 has one qubit gate and two qubit gate fidelity of 99.7% and 95.8% (Finke 2018). These are measures of the accuracy of the single input and two input gates. Two input gates such as *cx* are vital, but are much harder to implement than single input gates such as *x*. The 16 qubit Melbourne has 99.7% and 92.8% for these values - more error prone than the ibmqx4 (in fact the experiments in this paper were first run on the Melbourne, but then discarded due to impractical error rates). Google have built a 72-qubit quantum computer and report accuracies of 99.9% and 99.4%. Recently IonQ announced a 79 qubit machine and report rates of 99.97% and 99.3% (IonQ 2018). While all of this is happening, it is expected that the research community will develop a strong quantum subset of research looking at how to implement Boolean solving problems on quantum computers. IBM have released a machine learning kit for their computers, but they have not tested it running Grover on Boolean equations on hardware.

The Grover approach taken in qgMuse is fairly simplistic and linear. CHORAL and its progeny utilize many ingenious methods to speed up the search for solutions. Quantum-enhanced versions of these methods are sure to emerge. The computer music community can utilize such advantages.

The computer music world can also contribute to the quantum computing research world. This paper remains one of the few in existence that attempts to give a mid-level insight into Grover's algorithm. Most papers on Grover's algorithm are either entirely non-expert, non-practical or too complex for the average programmer. Computer music can provide an environment for utilizing quantum algorithms that - because of music's *relatively* lightly structured nature - will help non-quantum physicists and non-mathematicians to gain greater practical insight. Machine learning / statistical analysis for language processing or molecular pattern recognition are highly

constricted problems leading to extremely technical papers, whose results are meaningless to most non-experts. Music and sound allow for freer output structures - that can still produce a sense of meaning and pattern, with relatively light-touch algorithms. Hence a quantum computer musician may write an paper explaining their work and its results in a far more accessible way than a quantum computational chemist.

8. Acknowledgements

Thanks to Yassine Hamoudi of the Research Institute on the Foundations of Computer Science at Université Paris-Diderot for his optimization and error-checking of the Grover part of qgMuse.

REFERENCES

Anders, T. (2018) Compositions created with constraint programming. The Oxford Handbook of Algorithmic Music, p.133.

Brody, J. (1997) Background Count for percussion and 2 channel electroacoustic.

Cadiz, R. and Ramos, J. (2014) Sound Synthesis of a Gaussian Quantum Particle in an Infinite Square Well, Computer Music Journal, Vol. 38, No. 4, Pages 53-67, MIT Press.

Coleman, J. (2003) Music of the Quantum. Columbia University, New York, 2003.

Culpan, D. (2015) CERN's 'Cosmic Piano' uses particle data to make music, Wired, 8 Sept 2015. https://www.wired.co.uk/paper/cern-cosmic-piano

Ebcioğlu, K. (1988) An expert system for harmonizing four-part chorales. Computer Music Journal, 12(3), pp.43-51.

Finke, D (2018) https://quantumcomputingreport.com/scorecards/qubit-quality/ "Qubit Quality", Last Accessed Feb 2019

Glowacki, D., Tew, P., Mitchell, T., McIntosh-Smith, S. (2012) danceroom Spectroscopy: Interactive quantum molecular dynamics accelerated on GPU architectures using OpenCL In The fourth UK Many-Core developer conference (UKMAC 2012), Bristol, UK.

Grover, L. (2001) From Schrödinger's equation to quantum search algorithm. In: American Journal of Physics, volume 69(7): 769-777.

Harrow, A.W. and Montanaro, A. (2017) Quantum computational supremacy. Nature, 549(7671), p.203.

Hetherton, S. (2014) CERN scientists perform their data, 3 OCTOBER, 2014 | By Sophie Hetherton, CERN

Hiller Jr, L.A. and Isaacson, L.M. (1957) Musical composition with a high speed digital computer. In Audio Engineering Society Convention 9. Audio Engineering Society.

Kirke, A. and Miranda, E.R. (2007) Evaluating mappings for cellular automata music. In Proceedings of ECAL Workshop on Music and Artificial Life.

Kirke, A., Miranda, E., Chiaramonte, A. Troisi, A., J. Matthias, J., N. Fry, N., C. McCabe, C. (2013) Cloud Chamber: A Performance with Real Time Two-Way Interaction Between Subatomic Particles and Violinist. In: Leonardo Journal, volume 46(1).

Kirke, A. (2016) Superposition Symphony, Port Eliot Festival 29 July 2016 https://www.youtube.com/watch?v=-S5hU4oMWag

Kirke, A. and Miranda, E.R. (2017) Experiments in Sound and Music Quantum Computing. In Guide to Unconventional

Computing for Music (pp. 121-157). Springer, Cham.

Kirke, A. (2018) Programming Gate-based Hardware Quantum Computers for Music, Quantum Music and Beyond, Belgrade, Serbia, March 2018.

Kirke, A. (2018b) Programming Gate-based Hardware Quantum Computers for Music, Muzikologija, 24:21-37.

Monz, T., Nigg, D., Martinez, E.A., Brandl, M.F., Schindler, P., Rines, R., Wang, S.X., Chuang, I.L. and Blatt, R. (2016) Realization of a scalable Shor algorithm. Science, 351(6277), pp.1068-1070.

Narmour, E. (1992) The analysis and cognition of melodic complexity: The implication-realization model. University of Chicago Press.

O' Flaherty, E. (2009) LHCsound: Sonification of the ATLAS data output. STFC Small Awards Scheme.

Papadopoulos, G. and Wiggins, G. (1999) April. AI methods for algorithmic composition: A survey, a critical view and future prospects. In AISB Symposium on Musical Creativity (Vol. 124, pp. 110-117). Edinburgh, UK.

Perlner, R.A. and Cooper, D.A. (2009) April. Quantum resistant public key cryptography: a survey. In Proceedings of the 8th Symposium on Identity and Trust on the Internet (pp. 85-93). ACM.

Putz, V., Svozil, K. (2015) Quantum Music, Soft Computing, 1-5.

Shor, P. (2006). Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. In SIAM Journal of Computing, volume 26(5):1484–1509.

Singh, R.K., Panda, B., Behera, B.K. and Panigrahi, P.K. (2018). Demonstration of a general fault-tolerant quantum error detection code for (2n+ 1)-qubit entangled state on IBM 16-qubit quantum computer. arXiv preprint arXiv:1807.02883.

Sturm, B. (2000) Sonification of Particle Systems via de Broglie's Hypothesis. In Proceedings of the 2000 International Conference on Auditory Display. Atlanta, Georgia.

Sturm, B., Composing for an Ensemble of Atoms: The Metamorphosis of Scientific Experiment into Music. In Organised Sound, volume 6(2):131-145, 2001.

Waldrop, M.M. (2016) The chips are down for Moore's law. Nature News, 530(7589), p.144.

Weaver, J. (2018) https://github.com/JavaFXpert/quantum-toy-piano-ibmq, Last Accessed Feb 2019.

Weimer, H. (2010) Listen to Quantum Computer Music. In Quantenblog, available from http://www.quantenblog.net/physics/quantum-computer-music, last accessed Feb 2019.

Zalka, C. (1999) Grover's quantum searching algorithm is optimal. Physical Review A, 60(4), p.2746.