Testing a Hybrid Hardware Quantum Multi-Agent System Architecture that Utilizes the Quantum Speed Advantage for Interactive Computer Music

Alexis Kirke

Interdisciplinary Centre for Computer Music Research, University of Plymouth, Drake Circus, Plymouth, PL4 8AA, UK

alexis.kirke@plymouth.ac.uk

Word Count 13920

Testing a Hybrid Quantum Multi-Agent System Architecture in Hardware to Utilize the Quantum Speed Advantage in Interactive Computer Music

This paper introduces MIq (Multi-Agent Interactive qgMuse), which builds on the single agent quantum music research system qgMuse. MIq is the first attempt at a real-time interactive quantum computer music algorithm that utilizes the quantum advantage, and at implementing a musical quantum multi-agent system. MIq is a specific implementation of qHMAS, a Quantum Hybrid Multi-agent System architecture and is implemented on 5 and 14 qubit quantum hardware. qHMAS consists of a group of classical agents and a group of quantum agents connected by one or more classical / quantum hybrid agents that enable communication between the two. Communication between the hybrid agent and the quantum agents, and amongst the quantum agents, is done using quantum teleportation. Previous interactive or real-time music algorithms running on quantum computers have been mappings of classical computing algorithms onto a quantum architecture, with no advantage obtained. However MIq uses quantum teleportation so it can implement the quadratic speed-up of Grover's algorithm - a method from the field of Quantum Computing - for real-time rule-based music.

Keywords: quantum computing, multi-agent systems, algorithmic composition, interactive music systems

Quantum computers utilize the unique properties of quantum physics that manifest at the smallest levels of reality (one of the largest objects that exhibits quantum properties is made up of 60 carbon atoms (Gerlich et al., 2011)). Objects that have quantum properties are below anything we can really experience as human beings. Because the everyday reality we grew up with does not mirror what happens at the smallest levels, we find descriptions of the quantum-level of physical reality "spooky". This leads to much hype in the world of quantum computing about entanglement and superposition - properties that are mathematically simple - but do not mirror our concepts of visual experience.

This "spooky" hype in quantum computing also spills over into its applications. Work has

been done using quantum computers to generate visuals and sound / music. These have been described as quantum music or quantum art, but mostly they are either: (i) visualisations or sonifications of quantum computer processes; or (ii) mappings of classical algorithms onto a quantum computer. These can have a significant educational and artistic interest - but they are not really quantum computer music - in the sense that they are not using the underlying hardware in a uniquely useful way, or to do something new. They are producing work about quantum versions of classical algorithms that have no theoretical or practical speed up over the classical version.

A well-known early computer music tune was the IBM 704 singing Daisy Bell in 1961 (Scandalis et al., 2013). It showed that classical computers could - in hardware - creating a simple musical performance. Similar demos have been done showing that quantum computers could create a musical performance. For example, the first real-time interactive quantum computer musical performance was probably (Kirke, 2016) in which a mezzo-soprano's live singing was sent to a quantum computer and used to generate harmonies and sounds to accompany her. However, the algorithm used - as detailed in (Kirke et al., 2017) - does not provide any advantage over the classical version of the algorithm. The quantum advantage is defined as the potential for a quantum computer to solve problems faster (Riste et al., 2017). There has been minimal work in computer arts / music that can be theoretically proven to utilize the quantum advantage, one example being (Kirke, 2019). One purpose of this paper is to encourage more computer musicians to do true quantum computer music research.

In this paper an approach to building a quantum multi-agent system will be presented. A specific example will be implemented for interactive multi-agent computer music and tested. It builds on previous research into a single agent quantum music system implemented on hardware (Kirke, 2019). This example will first be implemented in simulation, then on a subunit basis on an ibmqx4 quantum computer, and in full on the 14 qubit IBM Q 14 Melbourne. Results will be reported.

Related Work

Multi-agent systems (MAS) are an extremely large field of research, dating back to the founding of Distributed AI in the late 1970s (Weiss, 1999). The precise definition of multi-agent systems has been approached in different ways. For example Wooldridge (2004) focuses on agents that are situated in an environment, with each agent autonomous, and with explicit goal-directed behaviour – i.e. not purely reactive. However there are a number of multi-agent systems where the agents have a central controller (Murray-Rust and Smaill 2005; Baxter et al. 2007), and there are a number of researchers who also define reactive-based systems as multi-agent.

There has been work in the theory of quantum multi-agent systems - e.g. (Klusch 2003) - or quantum -inspired multi-agent systems - e.g. (Badawy et al. 2013). (Alvarez-Rodriguez et al. 2018) is one of the only systems that has implemented a multi-agent system outside of simulation. In their approach, entanglement spreads through generations of agents. Geneological networks enable quantum information features to be inherited. Their work is couched more in artificial life rather than multi-agent systems. qHMAS (described later) is a first attempt at quantum multi-agent system research implemented on quantum hardware. The multi-agent approach for real-time interactive musical composition and performance has been a common approach, as it captures the ensemble nature of much music. This paper will now look at a number of related previous systems (an overview of the field of musical agents - real-time and non-real-time - and a typology for musical agents has been published by Tatar and Pasquier (2019)).

Blackwell and Bentley (2002) and Blackwell (2007) discuss improvisation from experience and from analysis – arguing that musicians respond to each other in improvisation using local-based rules – just as insect swarms do. Their system is a live algorithm which allows it to improvise interactively with human improvisers. Swarms move in 7 dimensions where the dimensions represent features such as: note loudness, time between notes, pitch,

time duration of note, etc. The behaviour of the human improviser will place “attractors” in the space, which the swarms will tend to congregate towards.

Clair et al. (2008) utilize an artificial ant colony to generate music. Ants wander through graphs in multiple dimensions, where vertices trigger pitches and durations. As the ants wander the graphs, they leave an artificial pheromone trail, thus increasing the probability of other ants following that path. This leads to rhythmic and pitch cycles which create repeating sequences that can be musical patterns. The user can play the ants as an instrument by either influencing adding notes they play to the graphs, or by “becoming an ant” themselves and creating pheromone paths around the graphs. Miranda (2003) creates a musical multi-agent system for the purposes of modelling music cognition. They are provided with a physical model of a vocal tract, which synthesises singing-like tones, and a hearing mechanism, which deduces pitch sequences from audio signals. The agents sing simple 3 note tunes to each other, and adapt each other's tunes and as a result, a shared repertoire emerges.

MMAS (Wulfhost et al., 2003) is designed to have agents play along with each other in real time producing compatible harmonies and rhythms. One or more of the agents can optionally be a live human performer (via MIDI control). Agents have a beat detection sensor and beat inference system to enable them to synchronize their playing. They also attempt to detect the current harmonic context to help them select their next note(s), based on a static transition table constructed from 50 popular songs. The agent’s musical knowledge is encoded with fuzzy logic rules such as “if accelerando is molto then dynamic is forte”. Although the agents in MMAS are mostly autonomous they also use a common “blackboard”, shared between themselves. The blackboard could be viewed as a form of additional service agent. Such functionally separate service agents have been common from the start of MAS (Botti et al., 1995) up to the current day (Szymanski et al., 2018), and are used in the system introduced in this paper later.

Linson et al. (2015) describe an agent-based system Odessa for collaborative free improvisation. It utilizes Rodney Brooks's (1999) Subsumption architecture in which

competing behaviours in a system are organised into interactive layers. The three layers in Odessa are designed to: (i) generate new musical output, (ii) respond to musical input, and (iii) to ignore user musical input and continue to generate music independently. Hutchings and McCormack (2019) introduce a system for real-time jazz improvisation where agents are focused on melody or harmony. The melody agent implements a rule-based system for transforming a library of pre-provided melodies. The harmony agent is based on a recurrent LTSM neural network. The neural network is trained using 2986 jazz standards, extracting their chord progressions.

The interactive multi-agent system presented in this paper consists of 5 agents: a classical computing agent that uses rule-based music composition, a human agent that plays in real-time, a service agent that is a specialized quantum/classical hybrid agent, and two quantum agents that listen to the other two agents through the service agent and try to react in real-time. The quantum and hybrid agents utilize three key elements: superposition, Grover's algorithm, and quantum teleportation. The first two of these are covered in the previous single-agent computer music paper (Kirke, 2019) on which this paper builds. The third - teleportation - will now be introduced briefly, in the context of quantum communication.

Quantum Communication

As has been mentioned, one of the main reasons that quantum technologies have received so much attention recently is the promised speed-ups from quantum algorithms. However, there is one other key reason for interest in quantum technologies: quantum communication. Quantum mechanics (QM) can model groups of subatomic particles as a single object called a state vector. The axioms of QM also separate out the idea of a particles state from its measurement. In classical mechanics, if the spin of an electron is defined as +1, then when you measure its spin it will be +1. So the electron has a spin of +1, and its measurement must therefore be +1. In QM, the state vector of a particle represents all that can be known about it, and it is usually some form of probability distribution. So the particle does not have a spin

value until it is measured. State and measurement are different concepts - in a sense the state is "bigger" than the measurement. In fact it can be shown that a quantum state - unlike a measurement - cannot in general be represented by a classical description (e.g. a bit string) no matter how many trillions of bits can be stored or transmitted (Gruska and Imai, 2001).

These properties of QM lead to fascinating possibilities in communications. In particular - it is possible to generate a pair of particles that are represented by a single state vector. Then taking these particles and separating them by say 100 miles *without observing their states* creates a quantum link between the person with one of the particles, and the person with the other. This is the basis of the quantum communication algorithms such as teleportation (Bennett et al., 1993) and superdense coding (Bennett and Wiesner, 1992).

Quantum teleportation is needed when we wish to perform quantum operations on the same data in different locations. It is this performing of operations on that data which is a major motivation for teleportation. If the operations performed at the quantum receiving end do not provide a quantum advantage, then there is not a clear case for the quantum teleportation in a computation system. There are 4 areas that are expected to provide a quantum computing speed-up advantage: Grover's algorithm, Shor's algorithm, Sparse linear equation solving, and physics/chemistry simulation. Of these, Grover's algorithm has a potential quantum advantage in music (Kirke, 2019). Sparse linear equation solving could clearly have applications in any music or sound synthesis which involves linear equations. Shor's algorithm is based around rapid period finding using a quantum Fourier Transform - and it is less clear how very fast period finding has applications in music. Though one possibility might be in the analysis of large databases of waveforms. Another could be in the generation of non-repeating tunes. (However, the quantum advantage only comes into play when such repeat cycles are extremely long.)

There are enough potential applications of quantum computing algorithms to music to justify examining the application of this key process in quantum communication: quantum teleportation. If you are generating quantum computer audio/music data in one geographical

location, and are part of a quantum computer cloud or internet (Kimble, 2008), then how do you transmit the quantum data to a location to be - for example - processed or combined with other data? It may appear that a binary representation can be sent through fibre optics using electromagnetic pulses. However, as has been mentioned, a general quantum state cannot be represented by a binary string. The actual state has to be somehow relocated. This is done by teleportation. In this paper, one focus will be on the feasibility of teleporting superposed musical data in a quantum system between agents.

Quantum Teleportation

Quantum Teleportation enables a qubit state to be transmitted any distance provided that the sender - traditionally called Alice - and a receiver - traditionally called Bob - share an entangled pair. Vitally, Alice must also transmit a number between 0 and 3 (i.e. two classical bits) to Bob using more traditional means (e.g. fibre optics or RF). But once Bob has these bits, he can perform an instantaneous transfer from Alice of her qubit, and Alice's qubit is destroyed at her end. Hence the name teleportation rather than copy or move. Copies cannot be done in quantum computers, by the No Cloning Theorem (Wooters and Zurek, 1982).

Importantly, given that in general a quantum state cannot be described by a string of classical bits, quantum teleportation provides a way of using classical bits and entanglement to move/transmit a quantum state precisely. The catch is, of course, that Alice must provide Bob with half of a stable entangled pair or vice versa. This is no mean physical feat, but is possible and researchers are improving their ability to do this over time. For example a photon's polarization has been teleported 870 miles to another photon (Ren et al., 2017).

The fact that the quantum state itself is teleported, and not just an *observation* of the quantum state, is potentially incredibly powerful, as it allows quantum computation and communications to continue at Bob's end. A circuit version of the teleportation algorithm is shown in Figure 1. Circuits are an alternative to symbolic algebra for representing quantum computing operations. Introductions to the notation are available (O'Donnell, 2015), but a

brief overview will be given here. As in Boolean diagrams, circuits process from left to right, with inputs at the left and outputs at the right. Figure 1 includes three very common circuit elements. The H box is a Hadamard gate – often used to create a superposition. The circular element that looks like a classical exclusive-or, does in fact perform that function in a quantum circuit. However it also creates entanglement between the two joined qubit lines. The boxes that look like semicircular measurement dials are actually measurement operators. They collapse the quantum information to classical, performing an observation.

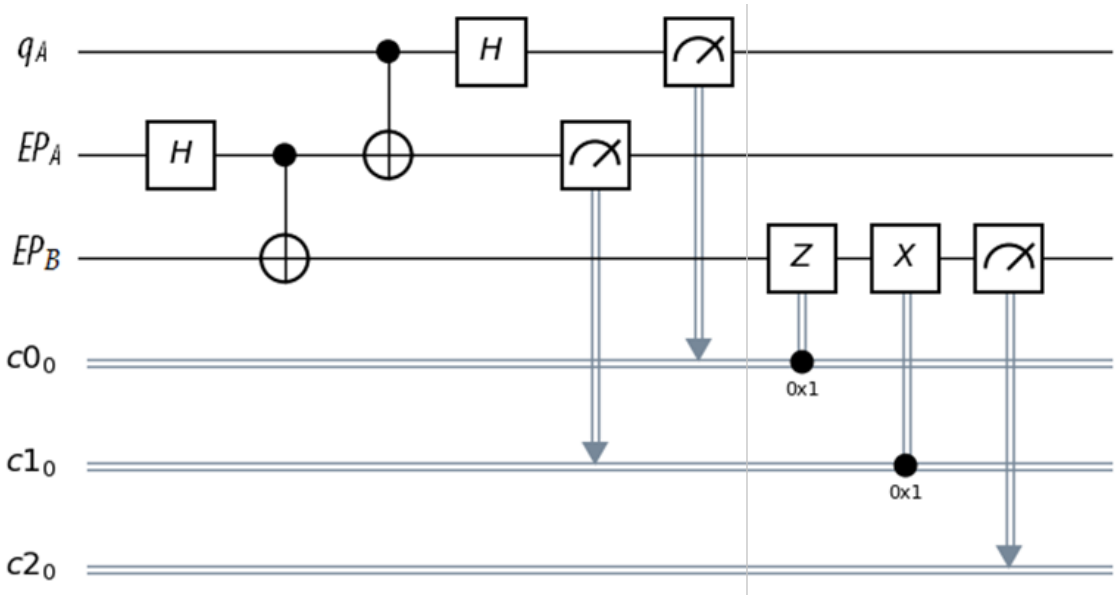

Figure 1. Circuit diagram of quantum teleportation

In a physically separated multi-agent system, the circuit would be divided across two quantum computers, one on either side of the vertical grey line. In reality there is currently no facility to share an entangled pair between two online quantum computers. However it is common to test classical multi-agent systems by implementing agents as different processes on the same computer. Similarly, in this paper, the entangled pair is two entangled qubits within a single quantum computer. Such an approach has identical physical properties to a

version shared across two computers. In Figure 1, EP_A is Alice's half of the entangled pair, EP_B is Bob's part, q_A is the qubit Alice wishes to teleport, and ($c0_0$,$c1_0$) is the pair of bits (giving a number between 0 and 3) that Alice sends by classical methods at the speed of light to Bob. $c2_0$ is the result of observing the teleported qubit EP_B. This helps to clarify the broader algorithm, but can be confusing as qubit sets *A* and *B* appear physically close in circuit notation, but this is an illusion. The fact they are physically separate qubits means they require teleportation to move the state from one qubit to another, no matter what the distance. (The missing technology is not the teleportation algorithm, but the ability to share an entangled pair between two online quantum computers.)

The circuit in Figure 1 is divided into sender and receiver by the grey line. The three operations following the grey line (Z, X and Measure) are hybrid classical quantum conditional operations, acting on Bob's entangled qubit EP_B to turn it into q_A. If the bits sent by Alice - ($c0_0$,$c1_0$) - are (0,0) then neither the Z (quantum phase inversion gate) nor the X (quantum not gate) operation are performed. If the bits are (1, 0) then just the Z is performed, if (0, 1) only the X is performed, and if they are (1,1) then both quantum gates act sequentially on the entangled qubit. The IBM quantum computers used in this paper do not yet have the ability to implement hybrid classical quantum conditional operations (except in simulation). However there is an equivalent circuit that avoids the use of such operators. The final circuit that is actually implemented in the hardware quantum computer to test teleportation is in Figure 2. Once again the grey line shows the division between the sender (q_0 and q_1) and the receiver (q_2).

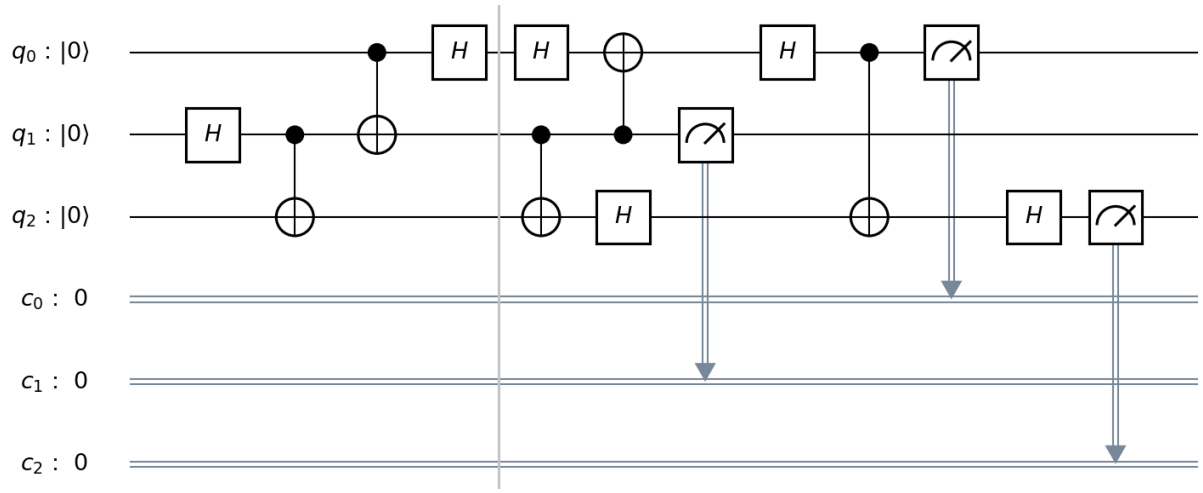

Figure 2. Circuit diagram to implement quantum teleportation in hardware

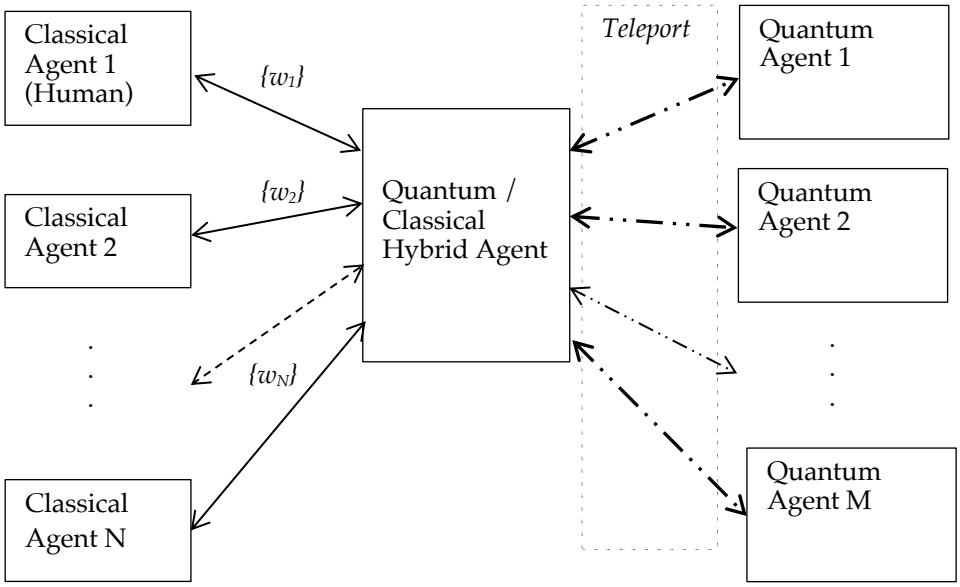

Figure 3: qHMAS - Quantum Hybrid Multi-agent Architecture

Quantum / Classical Hybrid Multi-Agent Systems

One possible architecture for hybrid quantum/classical multi-agent systems is introduced in Figure 3: qHMAS - Quantum Hybrid Multi-agent System architecture. The enabling aspect

of qHMAS is a hybrid quantum/classical agent. Because it is a hybrid agent that uses two types of information - quantum and classical – it is not a true blackboard agent (i.e. one that can be read from by all agents in the system), but a service agent. Also it needs to be both quantum and classical itself. The agent takes the outputs of the classical agents and can quantum superpose them based on sets of weights $\{w_i\}$. Each of the M quantum agents can have a different set of weights - i.e. it can receive a different weighted superposition from each of the N classical agents. The precise way these weights are defined will become clearer in the specific example given in the next section. However it should be made clear that a single weight can be used between two agents - i.e. classical agents can send non-superposed data to quantum agents, as well as superposed data. The hybrid agent next teleports the desired superposition to the relevant quantum agent. That agent then performs processing that utilizes some form of quantum advantage, and can then be observed by the user. Note that in general, the quantum agents can teleport their outputs to each other, though these lines are not shown. This is obviously not the only possible structure for a generalised hybrid multi-agent system, but it contains a number of relevant features for investigation.

As mentioned, a key limitation in quantum multi-agent systems is the No Cloning theorem. This is one of the reasons for having a hybrid service agent. To send data based on the same qubits to two quantum agents for example, the service agent will have to prepare the quantum data *twice.* This multiple preparation of data is called Identical Preparation in quantum mechanics (Hatsopoulos and Gyftopoulos, 1976). The Identical Preparation is all done by the hybrid agent rather than the classical agents, thus allowing more flexibility in the functionality of the classical agents (for example, if could be a live human performer with a MIDI keyboard).

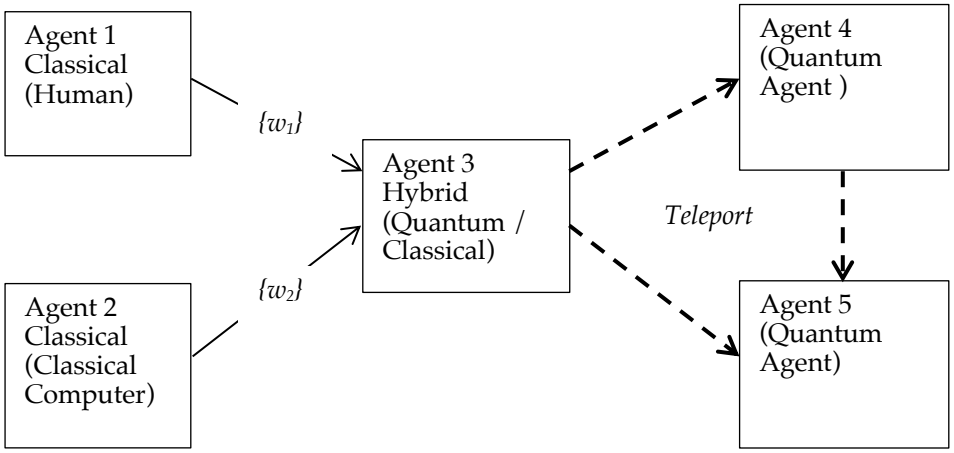

Figure 4: MIq overview, showing the multi-agent system

MIq: Multi-Agent Interactive qgMuse

MIq (Multi-Agent Interactive qgMuse) is an instantiation of the qHMAS architecture above, with N = 2 and M = 2: i.e. two classical performers (a human and a classical computer algorithm) and two quantum performers (two quantum agents).

Before going into detail about the quantum hybrid system MIq, a brief overview of the classical version of the problem is given:

a. There are two performers P1 and P2 who play along with each other (in MIq P2 is a computer music system not a human).
b. There are two rule-based musical artificial intelligences AI1 and AI2
c. (WLOG we will focus on AI1). AI1 takes as input a probabilistic performance PP12 defined as a weighted sum of the performances by P1 and P2 such that: at any time step there will be an X% chance of the music features of A being sampled, and a (1-X)% chance of the music features of B being sampled. X is a parameter defined by the user.
d. AI1 is pre-assigned M Boolean logical rules whose variables are bits from the binary representation of P1/P2.

e. During each performance time step:
    i. The feature distribution from PP12 is sampled to give an actual note N3.
    ii. One of the AI1 rules is selected and solved using an SAT solver, to get a solution for some of the bits in N3.
    iii. The relevant bits of the binary representation of the note N3 are replaced with those from the SAT solution in (ii).
f. This note is then played via a synthesizer.

The precise nature of this classical problem will become clearer as the quantum version is described later. However, elements (a) to (f) above will help to clarify the quantum exposition as well.

The quantum structure is outlined in Figure 4. qgMuse refers to the subsystem used within each of the quantum agents, a simplified version of which was tested on an ibmqx4 (Kirke, 2019). MIq is an example of how a human performer agent, a classical computer agent and quantum computer agents can perform together using quantum communications in real-time, using an algorithm inspired by the quantum speed-up. The agents utilise a simple classical communications protocol called TBE (Tiny Bit Encoding) for communicating with the quantum blackboard agent.

Table 1: Quantum Conversion

Semitone Interval	**TBE**	**Qubits**	**Note length (beats)**	**TBE**	**Qubits**
-8	0000	$\|0\rangle, \|0\rangle, \|0\rangle, \|0\rangle$	4	00	$\|0\rangle, \|0\rangle$
-7	0001	$\|0\rangle, \|0\rangle, \|0\rangle, \|1\rangle$	0.5	01	$\|0\rangle, \|1\rangle$
-6	0010	$\|0\rangle, \|0\rangle, \|1\rangle, \|0\rangle$	2	10	$\|1\rangle, \|0\rangle$
-5	0011	$\|0\rangle, \|0\rangle, \|1\rangle, \|1\rangle$	1	11	$\|1\rangle, \|1\rangle$

-4	0100	$\|0\rangle, \|1\rangle, \|0\rangle, \|0\rangle$	
-3	0101	$\|0\rangle, \|1\rangle, \|0\rangle, \|1\rangle$	
-2	0110	$\|0\rangle, \|1\rangle, \|1\rangle, \|0\rangle$	
-1	0111	$\|0\rangle, \|1\rangle, \|1\rangle, \|1\rangle$	
0	1000	$\|1\rangle, \|0\rangle, \|0\rangle, \|0\rangle$	
1	1001	$\|1\rangle, \|0\rangle, \|0\rangle, \|1\rangle$	
2	1010	$\|1\rangle, \|0\rangle, \|1\rangle, \|0\rangle$	
3	1011	$\|1\rangle, \|0\rangle, \|1\rangle, \|1\rangle$	
4	1100	$\|1\rangle, \|1\rangle, \|0\rangle, \|0\rangle$	
5	1101	$\|1\rangle, \|1\rangle, \|0\rangle, \|1\rangle$	
6	1110	$\|1\rangle, \|1\rangle, \|1\rangle, \|0\rangle$	
7	1111	$\|1\rangle, \|1\rangle, \|1\rangle, \|1\rangle$	

TBE and the Hybrid Agent

TBE needs to be a compact representation to be utilized by the hybrid agent, as the hybrid agent works with qubits encoding bits. The number of qubits available in current online quantum computers for the hybrid agent is most commonly 5, 14 or up to around 60 or 70. But the 60 to 70 qubit machines have limited availability. The focus here will be on 14 qubit machines. Thus TBE needs to encode and combine melodies using significantly less than 14 bits, as some of the 14 qubits are required for processing and so cannot be used for musical data encoding. Thus TBE encodes a melody note using 6 bits, leading to 6 qubits only, which translates into 6 superpositions.

The encoding mapping used by the hybrid agent is shown in Table 1. A number of simplifcations have been utilised to fit within the 6-bit limitation. Four bits are set aside for pitch and two for timing. Pitch melodic interval sizes are limited to between -8 and 7 semitones. Another simplification is that there are only 4 possible quantised note lengths: 0.5, 1, 2 and 4 beats. Finally, the time encoding is limited to a description of duration, with no onset data: notes are assumed to happen one after the other with no time gaps. The semitone interval encoding is standard binary encoding, though it has been re-centered around the

decimal value 8. This allows negative and positive intervals to be represented by increasing positive numbers within the range of the 4 bits set aside for pitch. The timing encoding is designed so that the least significant bit of the timing bits is a 1 for the shorter intervals - acting as a form of "shorter intervals flag". (This could be useful in bit-based quantum processing done in the quantum agents.)

Consider an example. Without loss of generality it will be assumed that all tunes have a reference starting point of C4 (middle C on the piano). Let a beat be defined as 1 second. Suppose the Human performer (Agent 1 in Figure 2) plays C4 for 0.25 seconds then G3 for 0.25 seconds. Then the note list is [C4, G3]. Given the reference point of C4, the melodic intervals are [0, -5]. The beat lengths are [0.25, 0.25]. Thus the TBE encoding, using Table 1, would be [1000, 0011] for pitch and [11, 11] for rhythm respectively. Converting these to qubits - in the format (interval ; timing) – gives, using the Dirac bracket notation for qubits:

$$\text{Human}(0) = (|1\rangle, |0\rangle, |0\rangle, |0\rangle \; ; \; |1\rangle, |1\rangle) \quad (1)$$

$$\text{Human}(1) = (|0\rangle, |0\rangle, |1\rangle, |1\rangle \; ; \; |1\rangle, |1\rangle) \quad (2)$$

Now suppose the classical computing music algorithm (Agent 2 in Figure 4) is - at the same time the human Agent 1 is playing - algorithmically generating its first two notes: G3 for 0.5 beats and F3 for 0.125 beats. Then this gives melodic intervals of [-5, -1] and durations of [0.5, 0.5]. The quantum TBE encoding would be, by Table 1:

$$\text{Classical\_Computer}\ (0) = (|0\rangle, |0\rangle, |1\rangle, |1\rangle \; ; \; |0\rangle, |1\rangle) \quad (3)$$

$$\text{Classical\_Computer}\ (1) = (|0\rangle, |1\rangle, |1\rangle, |1\rangle \; ; \; |0\rangle, |1\rangle) \quad (4)$$

These conversions are done by the hybrid agent (Agent 3 in Figure 4). After the conversion to TBE, the elements are buffered by the hybrid agent in note arrival order. This is done without synchronising time stamps - Agent 3 will consider the TBE note encoded in equation

(1) to be happen at the same time as the note encoded in equation (3), and the TBE in equation (2) to be simultaneous with (4). Next the hybrid agent generates a superposition of the TBE qubits across agents 1 and 2. This is done using a general unitary rotation gate and a Pauli X gate, based on Table 2.

So using the example given above in (1)-(4), the superposition would work as follows. The process starts with the most significant bit of the 4 pitch qubits in (1) and (3). It iterates rightwards towards the least significant bit and then moves on to the timing qubits going from most significant bit to least significant bit. So first the qubit for the most significant bit of the pitch for Agent 1 is superposed with the qubit for the most significant bit of the pitch in Agent 2 using Table 2. These are combined using the weights defined for each classical agent in the weight set $\{w_j\}$. The hybrid agent 3 uses Table 2 to combine TBE equations (1) and (3) qubit by qubit (moving from left to right) to give a superposition of (for sending to Agent 4):

$$(w_{41}|1\rangle + w_{42}|0\rangle, |0\rangle, w_{41}|0\rangle + w_{42}|1\rangle, w_{41}|0\rangle + w_{42}|1\rangle, w_{41}|1\rangle + w_{42}|0\rangle, |1\rangle) \quad (5)$$

Table 2: Tune Superposition

Agent 1 qubit (human)	**Agent 2 qubit (classical computer agent)**	**Superposition Operation**	**Resulting Qubit**
$\|0\rangle$	$\|0\rangle$	$Id\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$U3(tan^{-1}\frac{w_2}{w_1}, 0,0)\|0\rangle$	$w_1\|0\rangle + w_2\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$X(U3\left(tan^{-1}\frac{w_2}{w_1}, 0,0\right)\|0\rangle)$	$w_2\|0\rangle + w_1\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$X\|0\rangle$	$\|1\rangle$

As mentioned, each quantum agent *i* can receive a different set of weightings relating to a single classical agent. For example, quantum Agent 5 could receive a superposed TBE melody that is 0.5 of the human (Agent 1) and 0.5 of the classical computer (Agent 2); whilst quantum agent 4 receives a superposed TBE melody that is 0.75 of the human and 0.25 of the classical computer. The sum square of each pair of, in this example weights w_{41} and w_{42}, and weights w_{51} and w_{52}, must be 1 by the axioms of quantum mechanics. The superposed melody is then transmitted by the hybrid agent to the quantum agents. As already discussed, to transmit a quantum state requires the use of quantum teleportation.

To utilise teleporation, for each quantum agent *j*, the hybrid agent and the quantum agent *j* must share an entangled pair - represented by q_1 and q_2 respectively in Figure 2 (EP_A and EP_B in Figure 1). For example, this could be two photons that are generated using parametric downconversion (Rubin et al. 1994). In Figure 1, q_0 (q_A in Figure 2) is a TBE superposition generated by the hybrid agent using Tables 1 and 2. The output of the q_2/EP_B line will be the hybrid agent 3's qubit q_0/q_A from column 4 in Table 2.

The hybrid agent performs two Identical Preparations of the appropriately weighted superposition and teleports to each quantum agent. Each quantum agent will then have a qubit weighted superposition of the TBEs for the two tunes from agent 1 and 2's music. A key aspect of MIq is that the agents now apply quantum processes that have an advantage over their classical counterpart. The quantum agents generate music using two processes. The first is a soft rule-based quantum process that is faster than its classical counterpart. The second is to provide the results of the soft rule-based processing combined with the superposition, for classical observation. The rule-based process will replace some of the qubits in each TBE note by elements that define a "musical style" or “signature” for the specific quantum agent. The classical observation will collapse all the qubit superpositions that were transmitted to the quantum agent, based on the probabilities defined by the weightings in the superposition. Thus if the weightings were 0.5 and 0.5, then half the time the observed note will be influenced by Agent 1 and half the time by Agent 2. If they were

0.75 and 0.25 then three quarters of the time the observed note will be influenced by Agent 1 and a quarter of the time by Agent 2. Once all 6 TBE qubits in a TBE have been observed, they can be converted back to MIDI and played in realtime by a MIDI synthesizer or sampler. So the weightings define the relative musical influence of classical Agents 1 and 2 on quantum Agents 4 and 5.

Quantum Advantage

The rule solving process used by the quantum agents is called Grover's algorithm (Grover, 1996). The main motivation for applying quantum computing and communication in MIq is to lay the foundations for the increased speed of Grover's algorithm. In real-time duets between human and computer performers, speed is of the essence in the computer processing. Grover's is used here because it solves a rule with N logical variables using random unstructured search in at most of $\sqrt{2^N}$ time. For example, take a musical rule with 20 logical variables. Variables could represent states such as: the previous two pitches played by the classical computer are the same, the note length played by the human performer is less than a certain value, the pitch played by the human is consistent with the key mode of previous pitches played, etc. If this 20 variable rule is solved using random unstructured search on a classical computer, it requires up to 2^N = 1,048,576 iterations. If each search iteration took only 10 microseconds, the whole would take almost 10 seconds, which is too slow for real-time interactive computer improvisation. One efficient way around this is to precompute all possible results and store them in 16GB of RAM before a performance. However, if a second musical rule is added, with say 9 logical variables, then the number of iterations for brute-force solutions exponentially expands to requiring up to 8TB of precomputed RAM. Using Grover's algorithm, solving the two musical rules, totaling 29 logical variables, would take at most the square root of the 2^{29} number of iterations = 23170. Thus at 4 microseconds per operation, the time taken would be 0.25 seconds, which is much more practical for real-time algorithmic improvisation; as well as being more practical than

computing 8TB of RAM for each new rule added. Another advantage of the Grover approach over the precomputing approach, is that the rule structure can be more dynamic. The rule set can be changed in unlimited ways during a performance because the rules are solved dynamically rather than precomputed.

The reality is that no quantum computer exists that can deal with useful (fast) versions of Grover's algorithm. For example, the most powerful QC algorithm - Shor's Algorithm for factoring into primes - has been used on a hardware computer recently to factor 21 into 3 and 7 (Martín-López, E., 2012). This is a factorization that is trivial can clearly be done by hand. Faced with this, a computer arts researcher may be tempted to simply drop all investigation into gate-based quantum computers - those that can run Shor and Grover, until more powerful and stable gate-based quantum computers are available. However it should be born in mind that although QCs can only run factorizations of 21, the very existence of Shor's algorithm has begun to change some military organizations' encryption strategies (SAFEcrypto, 2015). The same potential exists for quantum computer music and Grover's algorithm. 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. Work needs to be done on actual quantum computers in these early stages, to ensure that computer arts keeps pace with advances in quantum computing, and also to test exactly what is feasible on a quantum computer in musical terms. So although the music generated by the research reported here is simple, the paper's contribution lies not in the quality of that music but in the foundations that the present work lays for future musical work as quantum computing matures.

(Kirke, 2019) goes into detail of how Grover's algorithm could combine or relate to the cutting-edge methods currently available in rule-based classical computer music (Anders, 2018). One element driving these cutting-edge systems is the assumption of the inefficient speed of unstructured random search (Ebcioglu, 1988). This issue is reminiscent of a similar

assumption made early in the evolution of chess computers. In Shannon's seminal 1950s paper on computer chess (Shannon, 1950), it was stated that computer chess machines would not be able to perform an exhaustive ("brute-force") search of future game possibilities due to such a search taking too long. Thus a number of heuristics and alternative methods were developed and used in chess-playing algorithms. However as classical computation speeds increased beyond what Shannon could have imagined, brute-force searches played a much greater role in chess computer strategies than Shannon originally envisioned (Larson, 2018). Modern chess computers still use heuristics, but brute-force is a key part of their strategies as well. Analogously, if quantum computers can be built that can run huge unstructured searches accurately and efficiently, it is entirely feasible that the speeds attained by quantum search algorithms will reduce the need for, or augment, some of the approximations, algorithms and heuristics used for solving in rule-based composition systems. This can lead to combinations of the fast quantum "brute" search with the best classical rule-based solvers.

The main contribution of this paper over (Kirke, 2019) is the recognition that when moving to multi-agent quantum music systems using quantum processing such as Grover, teleportation protocols need to be developed and tested. A secondary contribution is that MIq can run interactively with a human agent and classical agent, allowing live performances.

Note, that Grover is not the only possible algorithm that could be run in the quantum agents. As already mentioned, one other candidate is the HHL algorithm (Harrow et al., 2009), also known as the quantum algorithm for linear systems of equations. Suppose an algorithmic music system can be partially represented by a system of sparse linear equations with a low condition number. Given that the output of that part of the system is some function of the solution vector for the linear equations, then this can be solved exponentially faster on a quantum computer than on a classical computer.

Agent 2: The Classical Computing Agent

Having given an overview of TBE and the hybrid agent, all agents in MIq, including the

hybrid agent, will now be described in more detail – starting with the classical agent. Because the focus of this paper is on the innovation in quantum multi-agent systems and their use in interactive computer music, the algorithm in the classical computing agent (agent 2 in Figure 4) is of little interest as an innovation. Hence a simple algorithm is used here.

Agent 2 has a user-defined repertoire of 10 chords - each of 3 to 4 notes, within the key of C-minor. Like the TBE system it has a rhythm palette of 4 possible beat lengths: 0.5, 1, 2 and 4. The 10 chords are used to create - by hand - 5 pre-defined chord progressions of 3-5 chords which are stored as templates in Agent 2. A new tune generated by Agent 2 is derived by first randomly selecting one of these five chord sequences. Then the algorithm moves through each chord in the defined sequence in order in the template. For each chord, the algorithm randomly selects a constituent pitch from that chord. Then it chooses a "passing note" pitch of a random number of semitones above or below that chord note (maximum 3 semitones). Then it selects the next chord, and repeats this process, and so on. Thus if there are P chords in the generated sequence, 2P pitches will be chosen in total. This process is then repeated a fixed number of times defined by the user, say Q. The result set of 2PQ pitches are used to create the pitches of the classical computing Agent 2's output. The beat lengths are picked uniformly randomly from the beat palette. To create the final melody, notes are selected in sequence from 2PQ notes left to right until the sum of beats is integer and at least R beats long (where R is some user-defined integer >= 2). These notes are then the output melody, sent to the hybrid agent. Note that because of this length R limitation, it may be that not all 2PQ pitches are used.

Agents 4 and 5: The Quantum Agents

The quantum agents 4 and 5 (Figure 4) receive a superposition (a “weighted combination”) of the tunes from hybrid Agent 3, created from combining the human Agent 1 and the classical computing Agent 2. For example, the ordered list from equation (5) is sent element by element by the hybrid Agent 3 to the quantum agents. These are stored in a 5-element

quantum memory (Wang et al., 2019). So at the end of the quantum buffering, Agents 4 and 5 have the whole note data vector, both pitch and timing as shown in (6).

$$(|p_0\rangle, |p_1\rangle, |p_2\rangle, |p_3\rangle\ ;\ |t_0\rangle, |t_1\rangle) \tag{6}$$

However, a quantum agent does not necessarily make use of all of the TBE qubits it receives. The quantum agents are rule-based agents and can generate and replace / transform some of the TBE qubits themselves. In MIq, agents 4 and 5 each solve logical rules to replace certain elements of the TBE received. This allows them to place their "signature" or “style” upon each note of the tune. So for example, they may have logical rules that define how the first bit of the rhythm TBE and the first bit of the pitch TBE of a note can be related. In MIq, quantum agents have a list of S rules. The rule list is stepped through as each TBE qubit comes in, it is solved, and used to replace the relevant qubits. In the testing in this paper, S = 2. So Agent 4 and Agent 5 each have two predefined rules which they use during each qubit interaction. Agent 4's rule list is shown in (7) and (8), and Agent 5's in (9) and (10). Because each has only two rules, the rules will simply be switched between for alternate TBE teleports from hybrid agent 3. In a more complex system, there could be many rules which become active and inactive during different arcs of the performance, or when triggered by the performer.

$$\left((p_3 \oplus p_1) \oplus t_0'\right) \cdot (p_3 \oplus p_1) \cdot p_3' = 1 \tag{7}$$

$$(p_3 \oplus p_1) \cdot p_3 \cdot t_0 = 1 \tag{8}$$

$$(p_3 \oplus p_2) \cdot p_2 \cdot t_1' = 1 \tag{9}$$

$$\left((p_3 \oplus p_2') \oplus t_1\right) \cdot (p_3 \oplus p_2') \cdot p_3' = 1 \tag{10}$$

So rules (7) and (8) for Agent 4 constrain the relationship between bits p_3, p_1 and t_0 in ordered list (6). Suppose Agent 4 receives the TBE list in equation (5). It will solve one of

rules (7) or (8) to get three values, and convert equation (5) to include p_3, p_1 and t_0:

$$(w_{41}|1\rangle + w_{42}|0\rangle, |p_1\rangle, w_{41}|0\rangle + w_{42}|1\rangle, |p_3\rangle, |t_0\rangle, |1\rangle) \quad (11)$$

This process is represented for Agent 4 in Figure 5.

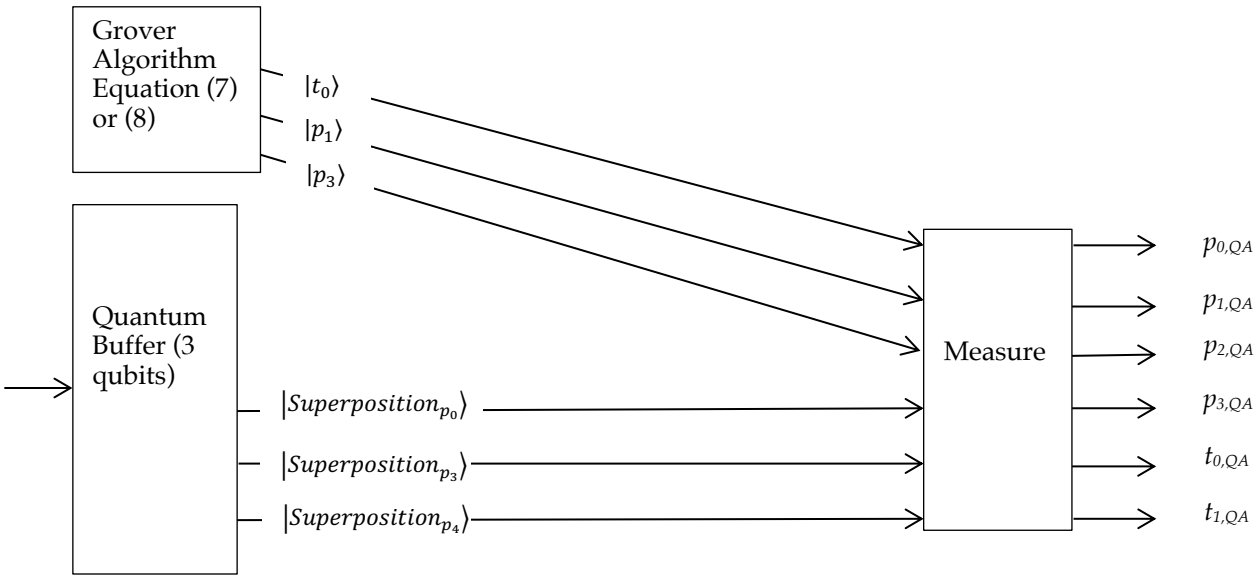

Figure 5. Block diagram of the Quantum Agent 4

Equation (11) is an opportunity to give further insight into the computation being done to create the music. Earlier it was stated how the Grover could replace or transform the *music* from Agents 1 and 2. However it can only *replace* qubits, not transform them. This is because Grover is not an input/output algorithm as such. It merely returns solutions to the embedded logical equation. The final output of Agents 4 and 5 is a transformation of the music from Agents 1 and 2 using Agent 4/5's Grovers and a weighted superposition. But music features from Agents 1 and 2 are not directly input into the Grovers. However, a qubit transformation approach, where a quantum algorithm in Agents 4/5 (involving a Grover)

takes inputs from the actual music features of Agents 1 and 2 could be envisioned. One approach is to use quantum logic gates in Agents 4/5 that take the output of the Grover, and use it to transform the superposition from Agents 1 and 2. For example a quantum XOR (also called a CNOT) or a quantum AND (also called a Toffoli gate). The outputs of these gates would then be a logical transformation of the qubits coming from Agents 1 and 2, by the qubit coming from a Grover solution. This would be relatively easy to implement with more available qubits.

These four examples are not particularly interesting music rules, but examples of how the quantum advantage can be used to solve rules faster than a classical agent during interactive algorithmic composition. Having a list of rules enables them to be a dynamic part of the performance. So for example, the user could switch on and off certain rules in Agents 4 and 5, add new rules, or adjust rules - all while the performance is happening. This dynamic ability is key, both as a way of providing live flexibility, but also because – without it –rules could simply be pre-solved and stored in memory (as discussed in the 16GB example earlier). In a more advanced system, the rules would be more musical, and the switching protocol could be much more intelligent and controllable. Solving each of rules (7)-(10) by random search would require up to 8 iterations to cover the possible values of the pitch and timing variables. Solving by Grover requires at most floor($\sqrt{8}$), i.e. at most 2 iterations (Grover 1996). Obviously in this case (7)-(10) are small enough to be solved by eye - e.g. Equation (7) is $(p_1, p_3, t_0) = (1,0,1)$ and Equation (7) is $(p_1, p_3, t_0) = (0,1,1)$. But as the complexity of the quantum agents' rules grow, so the system requires a larger number of iterations in classical search and also becomes unsolvable by eye. The Grover advantage increases quadratically, but is used here as a proof-of-concept in hardware.

The Grover architecture used to solve (7) to (10) is the same as that used in (Kirke, 2019), but with four new oracles, representing the four rules. These are shown in Figure 6 - each subfigure implements (7), (8), (9) and (10) respectively. Each Oracle has in common a control-control-Z gate, that implements the AND between the sub-rules in such a way as to

flip the phase of the correct solution. An explanation of the Oracle structure is also available in (Kirke, 2019). All lines in Figure 6 have a $|0\rangle$ qubit input by default. Each line focuses on one TBE bit, and the relevant TBE bit is shown to the right of each Oracle.

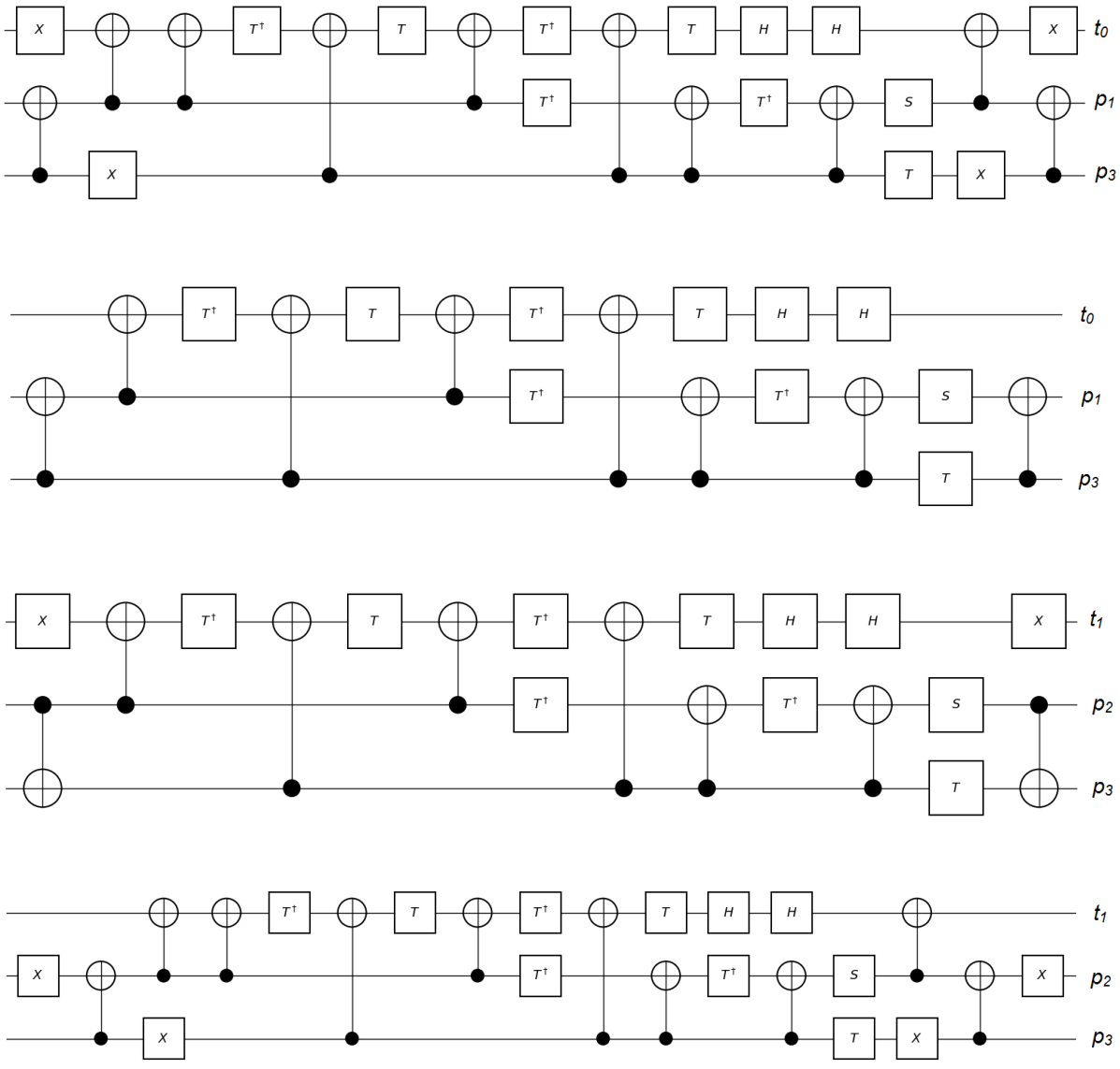

Figure 6. The four Oracles for equations (7)-(10). Each oracles has a 0-qubit input and the relevant TBE bits are marked for each line.

To enable a multi-agent system to be implemented in current online quantum hardware, the number of qubits used and teleported needs to be minimised, due to the limits in available qubits. In this particular multi-agent system implementation - the quantum agents replace 3 of the 6 teleported qubits with their own generated value (e.g. Agent 4 generates its own values for p_1, p_3, t_0). Therefore only three of the six qubits need to be teleported by the hybrid agent. The teleportation circuit in Figure 2 requires 3 qubit lines to teleport a single qubit of information. Figure 5 shows that equations (7)-(10) can each be solved using a 3 qubit line Grover. Thus the total required for a quantum agent to generate its 6 qubit TBE is: 3 qubit lines for each of the 3 teleported qubits plus 3 qubit lines for the selected Grover, totalling 12 qubit lines. This can be implemented on - for example - IBM's Melbourne 14 qubit quantum computer. Note that this neglects to implement a true quantum memory / buffer. In the experiments conducted to test MIq, the quantum circuits are sent to the Melbourne Q 14 quantum computer via the Python Qiskit SDK. So the quantum memory can be managed in the Python code, i.e. classically.

Results

Results of the system will be broken down into multiple parts to aid with clarity. First the classical algorithmic agent 2 output will be demonstrated, then the way in which the hybrid agent 3 combines the algorithmic agent's output with the human's live playing, both in quantum simulation and in hardware. The results of teleportation from hybrid agent 3 to the quantum agent 4 will be demonstrated, followed by a demonstration of the results of the Grover rules - once again in quantum simulation and hardware. Finally some example outputs will be shown from the complete system.

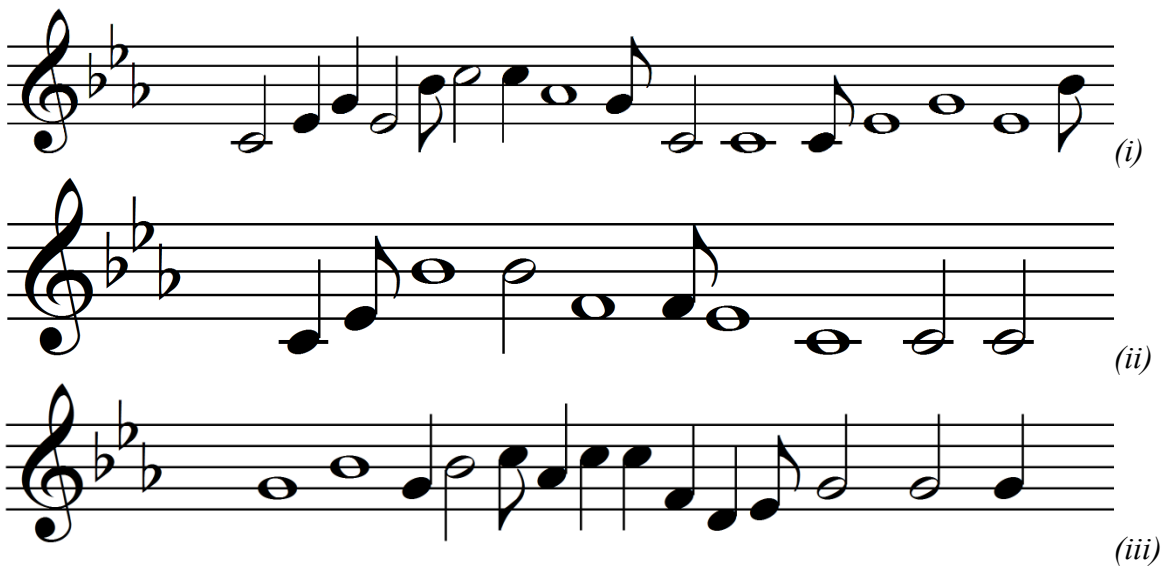

Figure 7: Three Example outputs (i), (ii) and (iii) from the Classical Computing Agent

Classical Algorithmic Agent

Figure 7 shows three example outputs - labelled (i), (ii) and (iii) - for the Classical Agent (agent 2). As already mentioned, they are based on a number of pre-defined sequences made up of common chords (such as Cm, Eb, F, Bb and so forth). As expected, the three resulting tunes fulfill the main requirement: they use only the 4 timing durations (4, 2, 1 and 0.5 beats), all intervals are within the interval limit of TBE, and the tunes are "tuneful" from a western point of view (if limited in novelty and variation). Note that all MIDI output systems used in this paper “pitch quantize” any tunes to C minor harmonic.

Hybrid Agent

The first process undertaken by the hybrid agent 3 is to convert to TBE. Below is the output of the hybrid agent's subsystem for converting to TBE for tune (i) in Figure 7. The first row is [pitch interval, duration] representation of tune 7(i) followed by the TBE stream:

[0,2] [3,1] [4,1] [-4,2] ...

00000101110**101101**111101…

As already seen in Table 1, pitch is divided into 4 bit chunks, and timing into two bit chunks - giving total chunk lengths of 6 bits. The 6 bits relate to Table 1 moving from left to right. So writing a TBE chunk as $p_0p_1p_2p_3t_0t_1$ means that p_0 is the most significant bit in Table 1 for pitch and p_3 the least significant bit, and t_0 is the most significant bit of timing in Table 1. Intervals are measured in semitones. The first interval is always defined as 0.

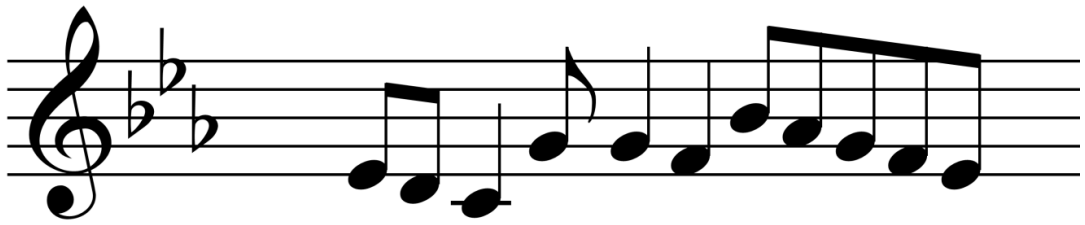

Figure 8: Output from Agent 1 - the human agent - a live quantized MIDI keyboard.

This TBE stream for Agent 2 is now combined with human agent 1 and converted to quantum data. To examine more closely the results of the quantum encoding step, consider Agent 1, the human agent - which is basically a MIDI keyboard that is live time quantized to the timing palette of the system. An example melody played live by the agent (the author) is shown in Figure 8.

The TBE stream for Figure 8 generated live by the hybrid agent 3 is:

000010101111110011010010111101101010100011010000011101000110100000100001101100110000010000111101

In the real-time system, Agent 1 (the user) is able to play on the MIDI keyboard in chunks of notes (11 in the case of Figure 8). In parallel, Agent 2 (the classical computer) generates its note set using the algorithm already described. Then these two sets of notes are sent as MIDI to the hybrid agent 3. It is this agent that converts the two note sets to TBE, and then uses

Table 2 to generate the superpositions.

Superposition Test

For the classical agent 2 and human agent 1 examples already shown, i.e. the outputs in Figures 7 (i) and 8 - the resulting teleported superpositions from hybrid Agent 3 cannot be directly plotted. This is because quantum states cannot be observed without being destroyed. However it can be checked that the statistics of the observed and collapsed state match those implied by the quantum state built and designed by Agent 3. The weightings described in Table 2 will lead to predictable statistical results. For example, if the TBE bits from Agent 1 and 2 are equally weighted in the superposition, each should have a 50/50 chance of being observed. Whereas if Agent 1's TBE is weight is 0.8 and Agent 2's is 0.2, then 80% of the time Agent 1's TBE should be observed.

A quantum circuit was built in Agent 3 that superposes the inputs from Agents 1 and 2 as defined in Table 2, and then teleports the results using Figure 2. The superposition and teleportation results are first tested using the IBM simulator, with the results in Table 3. Two sets of weightings (human, classical computer) were implemented (0.5, 0.5) and (0.8, 0.2). These are superposed and teleported with the human Agent 1 using the tune in Figure 8, and classical Agent 2 using the tune in Figure 7 (i). The process for examining the results is to only keep observations where Agents 1 and 2 had *different* values for their TBE, otherwise it is impossible to tell which of Agent 1 or 2's TBE bit was observed. This process is called measuring the "match count" here. The results of the IBM offline simulator are as expected, giving statistical observations close to the weightings values, as seen in Table 3.

Table 3: Simulator Results for Teleport and Superposition

Platform	**Agent 1/2 Superposition Weights pre-teleport**	**Agent 1 mean match count**	**Agent 2 mean match count**	**Agent 1 / 2 match %**
Simulator (10 runs)	0.5 / 0.5	8	9	0.47 / 0.53
Simulator (10 runs)	0.8 / 0.2	12.9	4.1	0.76/ 0.24

Table 4: Effect of superposition and teleporting of the p_0 qubit on quantum hardware ibmqx4.

Platform	**Agent 1/2 Superposition Weights pre-teleport**	**Mean prob. of observed teleported superposed p_0 being the p_0 of Agent 1/2**
ibmqx4 (4096 runs)	0.5 / 0.5	0.5 / 0.5
ibmqx4 (4096 runs)	0.8 / 0.2	0.77 / 0.23

To examine how this compares with running in quantum hardware, the approach is tested for the p_0 qubit being put into superposition and teleported. Exactly the same circuitry is used for all other p and t qubits, so it should be sufficient to test the process by testing one qubit in hybrid agent 3. And by using a single qubit it is possible to make the ibmqx4 run the test 4096 times for each p_0 in the two superposed and teleported tunes. Table 4 shows the results. It can be seen that the results for the 0.5 / 0.5 weighting are more reliable than the results for the 0.8 / 0.2 weighting. However given the hardware errors in current quantum computers (Finke, 2018), the results are quite reasonable. Once again, the match count approach is used. For the 0.5 / 0.5 case, p_0 is observed after teleportation as being the same as Agent 1's p_0 50%

of the time and the same as Agent 2's p_0 50% of the time. For the 0.8 / 0.2 weighting, it is Agent 1's p_0 77% of the time and the same as Agent 2's p_0 23% of the time.

To examine the effects of superposition on the whole tune, it is possible to use the IBM offline simulator and switch off the Grover algorithms in - for example - quantum agent 4. This will cause the entire output of agent 4 to simply be an observation of the superpositions teleported to its quantum memory. This is easy to do in the simulator, but there are not enough qubits to do it in the available quantum hardware. The simulator results implicitly assume two points: that the process of setting up the superposition is error-free, and that the teleportation is perfect. As has already been seen, in a physical quantum system, this will not be so. For example the ibmqx4 and the IBM Melbourne have hardware imperfections which mean that: (i) the superpositions prepared by the hybrid agent will not be exact, and (ii) the teleportations between agents will lose quantum information.

Teleportation Test

We will now look in detail at how the tunes in Figure 7 (i) and Figure 8, generated by the classical computer agent 1 and human agent 2, are being sent to hybrid agent 3. The resulting TBEs have already been (partially) tabulated in the previous subsection. These will now be superposed and teleported (firstly using the IBM offline simulator) by Agent 3 and teleported to Agent 4 and observed. The results of three repeated teleportations of these two superposed tunes - with equal (0.5, 0.5) superpositions - as observed from Agent 4, are shown in Figure 9. Although the same two tunes are used as inputs from Agents 1 and 2, the resulting TBE observed from Agent 4 will be non-deterministic due to its quantum preparation.

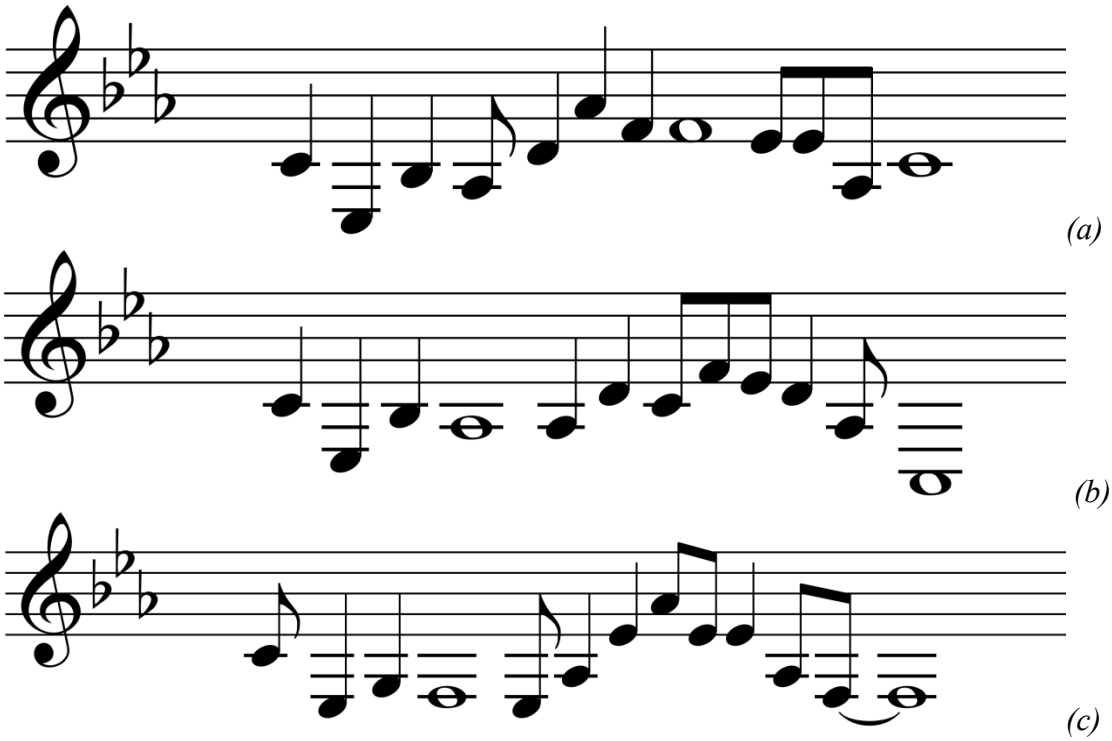

Figure 9. Three Examples (a), (b) and (c) of equal (0.5, 0.5) superpositions of Figure 7(ii) and Figure 8 teleported to Agent 4 and observed.

In Figure 9, the pitches are the same for (a) and (b) for the first three notes. The timings are the same for notes two to four. This is reflected in quantum Agent 4's collapsed TBE output for Figure 9 (a) and (b) shown below. Above it are the TBE encodings of the Agent 1's and Agent 2's tunes before they are superposed and teleported by hybrid Agent 3:

Agent 1 0000**10**10111111001101001011110110101010001101000001110100011010000..
Agent 2 0000**01**01110101101111110110001101101111010101100101110101100101100 1
Agent 4a 0000**11**11111101101111010111011011011100011011000011101000101110001
Agent 4b 0000**11**11111101101101110011101101101111010101100101110100101100000 1

Consider the highlighted elements of the TBE encodings. Although Agents 1 and 2 have different timings for their notes - 10 (2 beats) and 01 (0.5 beats) respectively - the teleported superposition stream contains both possible timings in a single qubit. This is why Agent 4 can be observed to have timing 11 (1 beat). This same process occurs through the whole

observation of Agent 4's qubits, which were then converted to intervals and durations and into the music in Figure 9. This can be seen even more clearly - once again using the offline simulator - if human Agent 1 is given a weighting of 0.8 and classical Agent 2 of weighting 0.2. The resulting TBE outputs of observing quantum Agent 4 are:

Agent 1 00001010111111001101001011110110101010001101000001110100011010000..
Agent 2 00000101110101101111110110001101101111010101100101110101100101 1001
Agent 4a 00001011110101001101001011110111101010001101000001110100011010 0000
Agent 4b 00001010111101001101001010110110101010000101000001110100011010 0001

Unsurprisingly the top row of TBE for Agent 1 is very similar to the bottom two rows - observations of Agent 4 - because the superposition is now weighted much more towards human agent 1. However, they are not identical. At times Agent 2's music effects Agent 4's output as well. Figure 10 shows the resulting tunes from the bottom two rows of the TBE tabulations above (Agent 4a and 4b). These tunes tend towards lower notes because, as can be seen in Figure 8, most of the intervals played by the human are downward intervals - the 80% contributor to Tunes (a) and (b).

So it has been shown so far, how TBE bits can be superposed and teleported in hardware, resulting in the expected statistical distributions. This was done in simulation because testing the superposition/teleportation system for all 6 qubits cannot be done in hardware when the Grover is switched off, as too many teleportation qubits are required. It will be seen below, that once the Grover is switched on, a system test can be run in hardware - since the full system requires fewer qubits than the system without the Grover (due to reduced teleportation overheads).

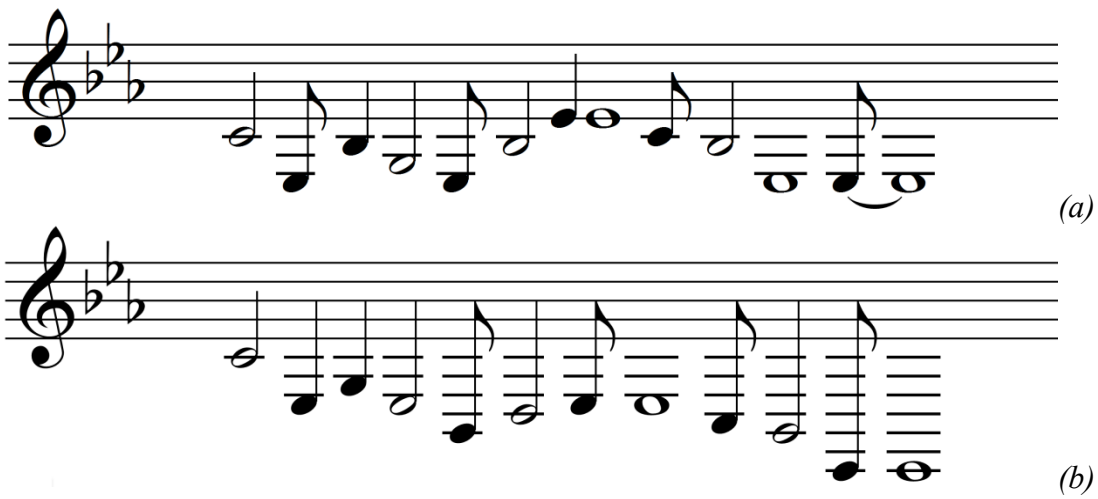

Figure 10. Two Examples (a) and (b) of 0.8/0.2 superpositions teleported to Agent 4 and observed.

Grover Test

The above method of combination of two tunes by superposition is conceptually interesting, the resulting musical effects are not unique to QC. They could be achieved classically using a statistical computer algorithm. As has been mentioned, there is no advantage to simply implementing statistical algorithms on a quantum computer per se. The advantage of the quantum approach is that quantum agents can then utilize the quantum speed-up. In this case, Grover's algorithm is used in Agents 4 and 5 to solve a potentially dynamic rule set - in this example rules (7) to (10).

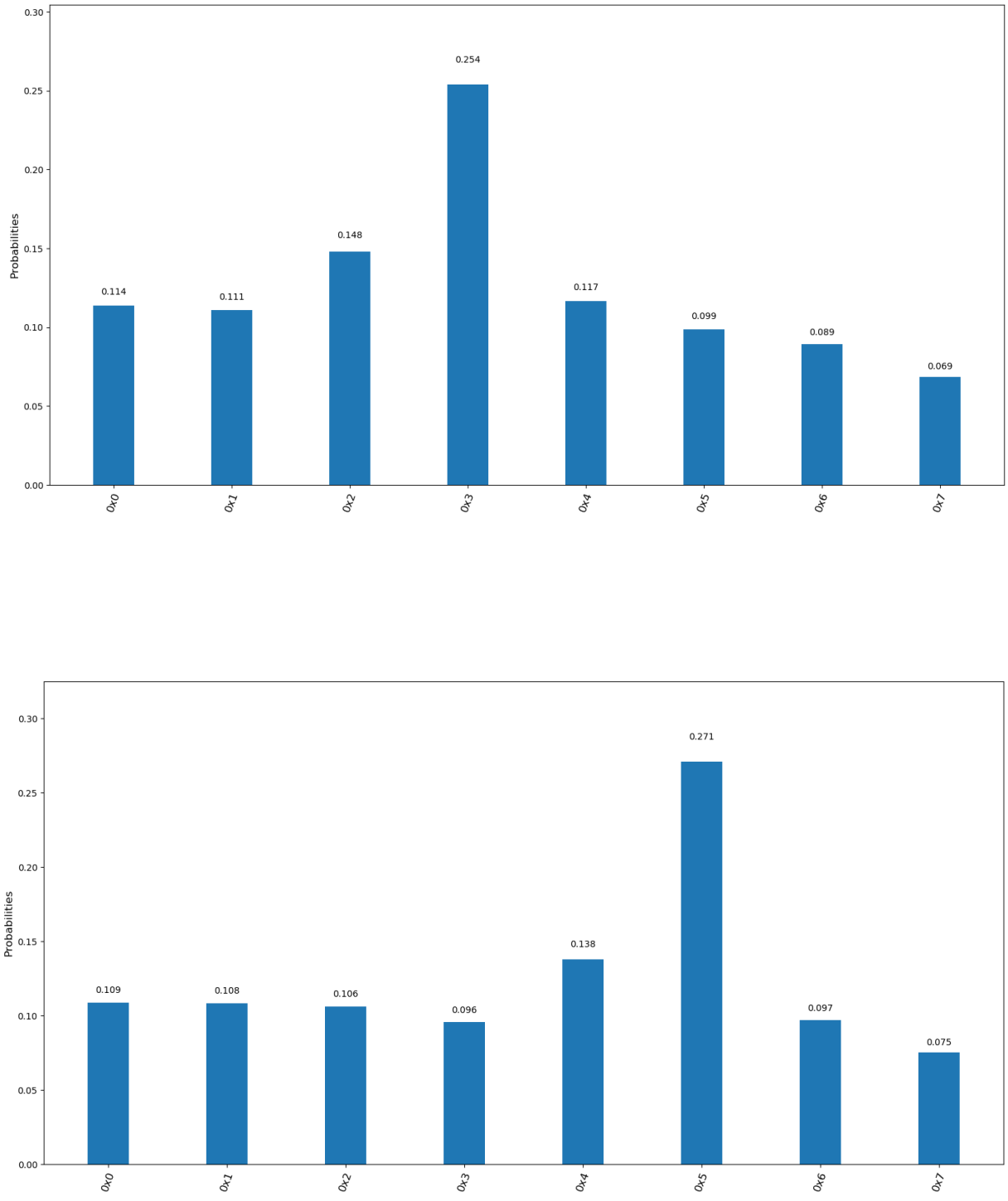

Figure 11. The outputs of the top two Oracles in Figure 6 – solving equations (7) and (8).

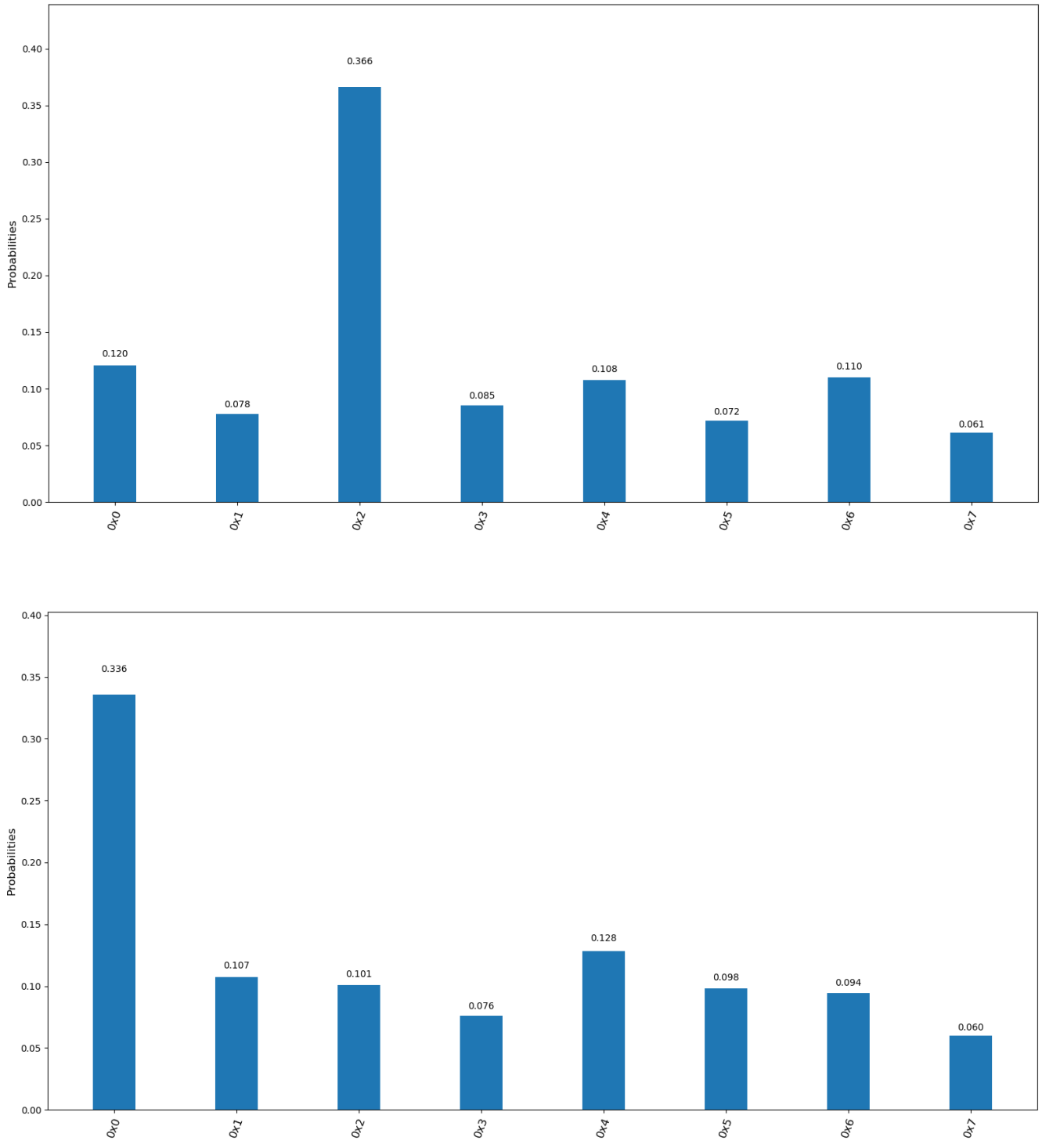

Figure 12. The outputs of the bottom two Oracles in Figure 6 - solving equations (9) & (10).

To test the Grover element of MIq, the Oracles defined for (7) to (10) will first be tested on quantum hardware to see if they give the correct solutions. Figures 11 and 12 show the results of running the Grovers for agents 4 and 5 respectively on the ibmqx4 hardware. Figure 11 shows the output for 4096 runs to solve equations (7) in the top sub-figure, and (8) in the lower sub-figure. Figure 12 shows 4096 runs for (9) and (10). The ibmqx4 gives outputs in reverse binary converted into hexadecimal. So the four most likely outputs for solutions to (7), (8), (9) and (10) are: 0x3, 0x5, 0x2 and 0x0. Converted to binary and then reversed gives: 110, 101, 010 and 000. The first digit of each binary number represents the top line of the related Oracle in Figure 5, and the third digit of each binary number represents the bottom line of the related Oracle in Figure 5. Solving by eye equations (7) to (10) and tabulating the results in the same format gives: 110, 101, 010 and 000 as required. The full circuit for each Grover includes only one iteration of the Grover operations. As opposed to the up-to 8 iterations required for an unstructured search by a classical computer.

As already mentioned, the effect of Agent 4's Grover - if perfect - would be to alternately set $p_1, p_3, t_0 = 1,0,0$ and then $p_1, p_3, t_0 = 0,0,1$. This would create alternating TBE chunks of the form $p_0 1 p_2 00 t_1$ and $p_0 0 p_2 01 t_1$. So for the first case (by Table 1) this forces one note to have a longer length (00 or 10) with an interval of size 4 or -4; and the next note to have a short length (01 or 11) and an interval of size -8, -6, 0, or 6. This leads Agent 4 to be stylistically biased towards medium to larger melodic intervals, particularly in the negative direction; and to alternating between longer and shorter length notes.

Note that the Grover algorithm is non-deterministic. So if used in a single-shot way (as is done in MIq) then it will only exhibit a tendency to give the correct answer. It will not always give the correct answer. In fact looking at Figures 11 and 12, in current hardware there will be many instances of the Grover giving the "incorrect" answer from a logical point of view. This incorrect answer - leads to soft-rules. Such soft-rules are often desirable in rule-based music algorithms (Anders, 2018). Even in the case where one wishes to use the Grovers in MIq more deterministically, they could be run multiple times and an average

taken. For small numbers of inputs/rules this could cancel out the quantum speed-up. However, as the number of rules and inputs increases, the speed-up will become essentially quadratic again, despite a linear increase in the number of repeated samples.

Full System Implementation on Hardware

Two versions of the full system were implemented, one for hardware testing, and one for a live performance. The issue with system implementation is that the number of qubits required - 12 - means that the IBM Q 14 Melbourne must be used rather than the ibmqx4. The Q 14 has higher two input gate error rates - specifically CNOT - which is used frequently - particularly in the Grovers (see Figures 11 and 12). Thus it is expected that a system test will be unable to independently verify the expected outputs of MIq found in the ibmqx4. However, implementing the full system is a major step in demonstrating the feasibility of building a hardware circuit to implement MIq. Given the successful unit testing of all of the elements of MIq, and the feasibility of implementing it on hardware, despite the hardware imperfections - it provides a strong proof-of-concept for a scalable use of the quantum speed-up and quantum communications in quantum music systems and multiagent systems.

It is worth taking a moment to highlight the value of scalability. All quantum algorithms (for example Shor's algorithm, HHL and Grover's algorithm) currently run more slowly than their classical counterparts. 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 utilizes 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 can eventually incorporate examples that will run much faster than their classical counterparts.

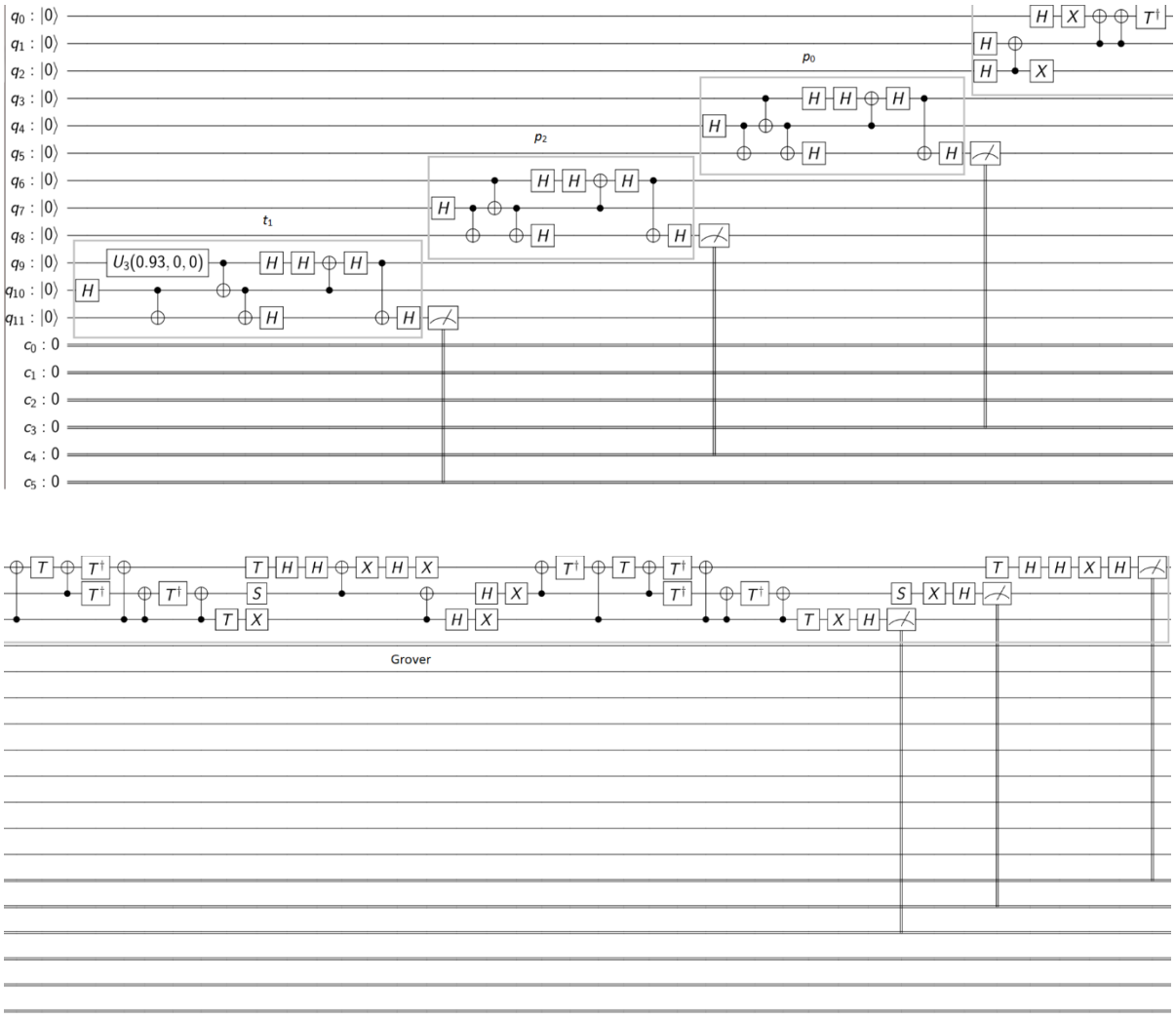

Figure 13. A full quantum circuit in the IBM Q 14 that implements Agent 4 and the quantum part of agent 2, for the first two TBE chunks from Agents 1 and 2 in the earlier examples.

Hardware Testing

Having tested the single qubit superposition and teleportation on hardware, and the Grover oracles on hardware, the system is implemented as a whole on the IBM Q14 Melbourne. A full circuit implementing a quantum agent (agent 4 or 5) and the quantum part of Agent 3 is shown in Figure 13. The circuit is split into two parts with lines going from left to right, and the top subfigure lines connected to the corresponding lines of the bottom subfigure. The first

three groups of components in the top subfigure, boxed and moving diagonally up, are the three TBE superposition being generated and teleported (in Agent 4's case these will be p_1, p_3 and t_0.) The $U_3(0.93, 0, 0)$ on the far left of the top subfigure, and acting on q_9, is the unitary rotation operation that implements the bit superposition of Table 2. To the right of the 3 teleports and spilling into the bottom subfigure is the Grover algorithm for Agent 4 - Equation (6) – whose Oracle is shown at the top of Figure 5. **(**Note that this circuit structure has to be partially rebuilt and executed for each new TBE chunk, so as to implement the new superpositions from Agent 3, and the alternating Grover oracles for Agents 4 and 5.)

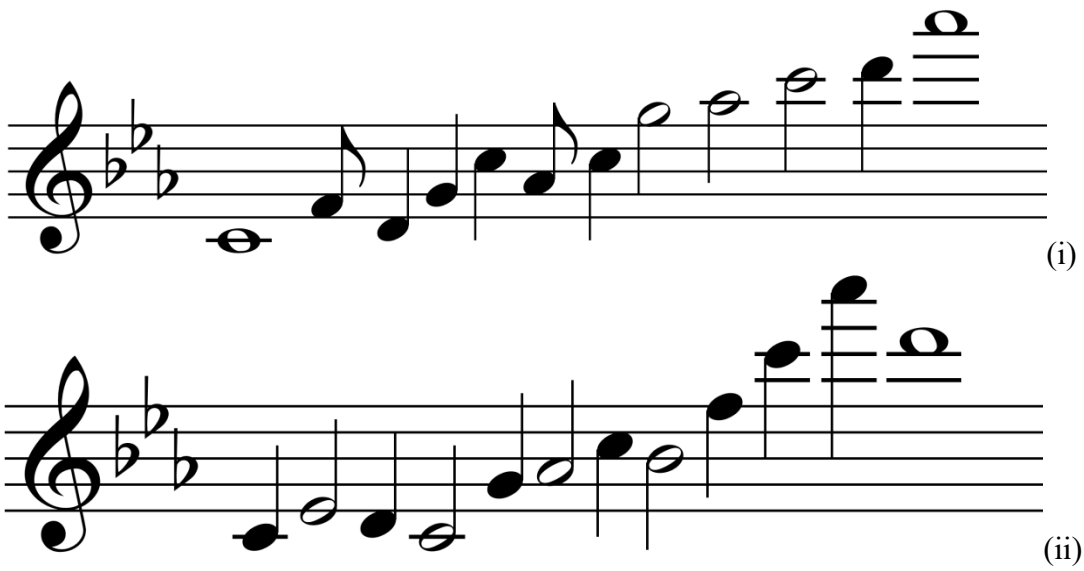

Figure 14. Example Outputs of IBM Q 14 Melbourne for Agent 4 with weights 0.8 / 0.2

Figure 14 shows two example outputs for Agent 4 on the IBM Q 14 Melbourne with weights 0.8 and 0.2 - using the same musical inputs as used in the unit testing earlier. The note pitches have a tendency to move upwards, which is in contradiction to Figure 10 – which implied that the 0.8 / 0.2 should have a lower pitch tendency. Agent 5 at 0.8/0.2 in Figure 15 shows a similar pattern of upward movement, more so than Agent 4. Based on Table 1, the upward tendency implies that p_0 is tending to be a 1 due to quantum hardware imperfections.

(A selection of the hardware musical output examples from this section can be heard a non-live electronic music recording available online (Kirke, 2019b) at 16m37s.)

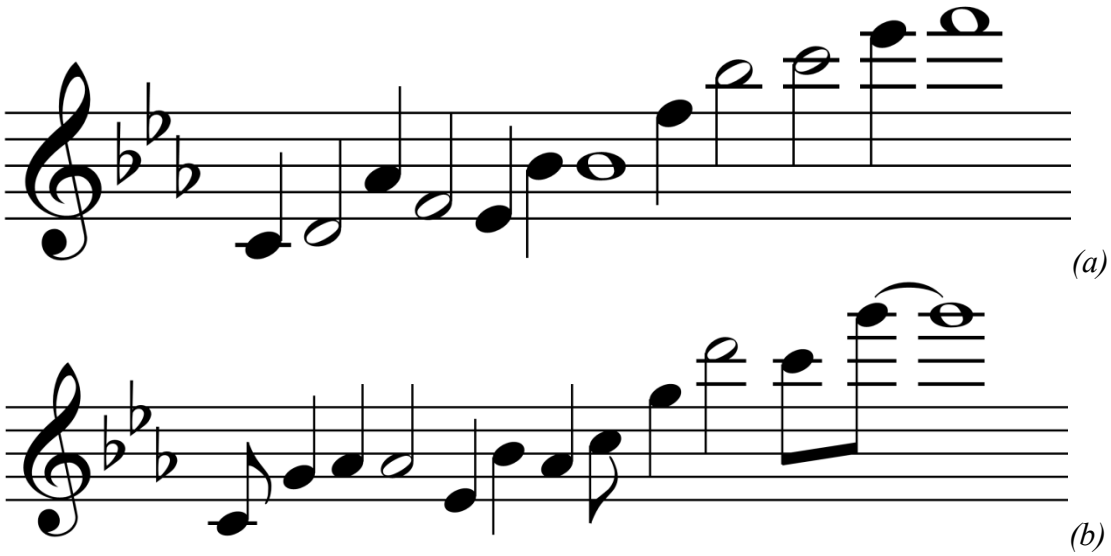

Figure 15. Example Outputs of IBM Q 14 Melbourne for Agent 5 with weights 0.8 / 0.2

Live Performance Hardware Implementation

In May 2019, a live performance - also streamed live over the internet - was done using MIq and the IBM Melbourne Q 14 (Kirke, 2019c). Agent 1 was a human performer playing a MIDI keyboard, Agent 2 was the classical computing agent using the same parameters as used in the examples earlier in the paper.

The agent cycle was as follows. First the core Python thread executes Agent 1 (i.e. it takes input from the MIDI keyboard of a small number of notes). These notes are heard by the audience in real time as they are played, because they are re-routed by OSC to a piano sound-generator. Then classical Agent 2 plays (generating *N* notes using the same settings as in the examples). A parallel thread is then triggered (called here Thread CC) to send the classical computer's generated tune to a MIDI sound unit, and a thread (called here Thread Q) is triggered to execute Agent 3 (the hybrid agent) and Agent 4 (the quantum agent). Thread Q

generates a quantum circuit based on creating an equal superposition between Agents 1 and 2 - i.e. (0.5, 0.5) - teleporting the result, and then overlaying a Grover solution on the result. This circuit is sent to the IBM Q 14, executed and observed, and the results converted to MIDI and sent to a sound generator via OSC. Note that Python Thread's Q and CC are allowed to spawn in multiple versions, but Thread Q is not allowed to send more than one circuit to the IBM Q 14 at any one time.

Discussion

There are four perspectives from which to evaluate the outputs of quantum Agents 4 and 5 such as those in Figures 14 and 15: (i) Is the algorithm correctly implemented in theory? (ii) How musically useful is the algorithm? (iii) How useful is qHMAS as a quantum multi-agent approach? (iv) What can be said about the feasibility of quantum hardware implementation of MIq and qHMAS?

Is the algorithm correctly implemented in theory?

The first test performed on the algorithm was whether the superpositions, once they had been teleported, matched the expected statistical distribution. On the simulator (Table 3) it was found that the superposition (0.5, 0.5) led to an approximately 50/50 influence between Agents 1 and 2 being teleported; and the superposition (0.8, 0.2) led to an approximately 80/20 influence from Agent 1 and 2 respectively, and that this was teleported. Without loss of generality, this was tested in the one-qubit case on a hardware quantum computer (Table 4) - the ibmqx4 - because the circuitry was exactly the same for all superposed and teleported bits. The results for the two superposition levels were similarly successful across 4096 executions.

The Grover algorithms running in Agents 4 and 5 were independently tested on the ibmqx4 hardware. The results were shown in Figures 11 and 12. They were more "fuzzy" (error-prone) than the results for the superposition/teleport test. The correct solution did

indeed have the highest probability, but there was a greater probability of getting the wrong solution. The rest of the hybrid algorithm is fairly trivial - involving converting MIDI into TBE and vice-versa, and so required simple mostly unreported technical testing.

How musically useful is the algorithm?

Because of the small number of bits available, the focus of usefulness in this research is not on the algorithm's musicality, but on its speed. Having said that, Tune (i) in Figure 14 is fairly pleasant to listen to. Tune (ii) less so because of its rhythmical regularity. But neither sound like random walks. Another weakness of the underlying classical computing elements is the fact that the hybrid agent is buffering in such a way that notes of different lengths from different agents are still paired together. For live performance this is not a dynamic approach. The dynamic approach would be that if Agent 1 played two quarter notes while Agent 2 played a half note, then a real-time decision system could be used. For example - split agent 1's half note into two quarter notes and make two superpositions.

The tunes produced by the Melbourne Q 14 for Agents 4 and 5 are not consistently pleasant. However, this paper does not claim to objectively evaluate the music. The paper claims that this is an example of a multi-agent music algorithm which can, as quantum computing and communications research and hardware develops, lead to a huge increase in processing speeds. This is musically useful in a real-time system for reasons explained earlier. This will give a potential speed up in reaction time in a real-time music system from almost 10 seconds (in the example given earlier), which is too slow for interactive computer improvisation, to 0.01 seconds. To enable this to occur when multiple separate musical agents are involved, requiring the use of teleportation to move states around the multi-agent system. This approach is indeed the case in MIq, where a different set of Grover's are performed by Agents 4 and 5. Conversely, it is pointless using teleportation, unless some fast quantum operation is going to be performed locally. The arguments for the desirability of using multi-agent systems in music, particularly in real-time music creation, are well-known,

and detailed in the multi-agent musical systems referenced earlier. If it is desired to implement quantum agents running advantageous algorithms, then communication is needed that can deal with quantum information (Kimble, 2008).

How useful is qHMAS as a quantum multi-agent approach?

This has already been touched on above. The main purpose of current quantum advantages is speed. If it is desired to create a distributed multi-agent quantum system utilizing different algorithms such as Grover or HHL, then clearly teleportation is needed for communication of quantum states. qHMAS is specifically designed as a *hybrid* system. In other words some of the agents are classical. This requires some form of interface between the classical and the quantum. In qHMAS this is provided by a hybrid agent (agent 3 in MIq), but in theory every classical agent could be made a hybrid classical / quantum agent and perform its own teleportation. This could a less efficient approach, given the hardware requirements involved in implementing a quantum system - usually requiring a great deal of shielding and multi-stage refrigeration. Localising this hybridity in a single service agent, reduces the amount of hardware required in the hybrid system. It also enables a broader class of classical agents to be brought into the system. For example in MIq, the only requirement is that classical agents can transmit in MIDI format.

Aside from the use of a hybrid agent, the other defining element in qHMAS is the ability to superpose the classical inputs. It should be noted that Figure 1 only provides this as *an option*. It is perfectly feasible for the weight sets in Figure 1, and the ongoing routing to the quantum agents, to enable the outputs of the classical agents to be sent *individually* to the quantum agents (and to each other). In MIq, the weights could be set easily so that only Agent 1's outputs were sent to Agent 4, and only Agent 2's to Agent 5. So qHMAS does provide a general routing architecture for a hybrid classical / quantum MAS. The addition of the superposition option in the hybrid agent has two possible functions: (i) as a way of minimising the expensive communication pipeline between the classical and the quantum;

(ii) as a method of probabilistic merging of data in a quantum form.

In the same way that building a quantum agent is expensive (hence the use of a single hybrid agent), building a quantum communication line is expensive. It requires the delicate sharing of an entangled pair to perform the teleportation. In qHMAS this happens for every qubit, shared to every agent. Thus in multi-agent applications where the superposing of bit values is acceptable or desirable, such superposed teleportation lines can save on resources. In the MIq case, it is done because it is creatively desirable. One could imagine other cases in which it was resource-wise desirable: for example providing communication priorities between two communication lines L1 and L2 - going between say classical agent's 1 and 2 and quantum Agent 4. Assume L1 and L2 are explicit - i.e. losing data doesn't make them unintelligible, just lowers their information content linearly. Suppose these two communication lines have inverse priority. In other words when L1 is higher priority, L2 is lower priority and vice-versa. L1 and L2 can be superposed by a blackboard agent - and during L1 priority times the weight for L1 in the superposition can be increase. Then during L2 priority times, the superposition weight for L2 can be increased.

What can be said about the feasibility of quantum hardware implementation of MIq and qHMAS?

Given the current quantum hardware situation, tests of quantum multi-agent systems are indicative at best. However, it is a reasonable philosophy that one should always run a system on hardware if one can. Running in simulation is also vital for most future-looking research. But it can have the effect of making a researcher look too far into the future. By forcing oneself to at least try to implement on hardware, it produces a creative tension between the possible and potential. The IBM Q 14 Melbourne does not run the MIq implementation of qHMAS well. If MIq is broken down into subunits and run on the 5-qubit ibmqx4, each subunit runs in the expected way. When all units are placed together (3 qubits superposed and teleported, 3 qubits generated by Grover) the results are very fuzzy. The

main reason for this is probably because Teleportation and Grover use a significant number of two input gates (which are error-prone on the Q14). (Kirke, 2019) has shown that two-input gate errors can quickly dominate a Q14 circuit. But the current paper has shown – by including generalisable testing on the ibmqx4 - that once a Q14-equivalent is built with hardware errors comparable to the ibmqx4, then the specific implementation of qHMAS – MIq – will work as expected.

The next two possible pieces of future work for MIq and qHMAS are to implement them either on: (i) a less error-prone 12 qubit machine, or (ii) on a machine with a much larger number of qubits that can be used for error correction. Option (ii) is particularly attractive as the number of qubits available is growing, and as far back as 1995, Peter Shor showed that using 3 qubit lines, a single qubit line could be made less error prone (Shor, 1995). If a sufficient number of qubits were available to provide error correction, and MIq could be implemented for a complex enough Grover (or HHL) music problem set, then far more reliable results could be demonstrated for interactive music generation.

Conclusions

This paper has introduced qHMAS, a Quantum Hybrid Multi-agent System architecture, and implemented it using two types of quantum hardware and a quantum simulator. A qHMAS system consists of a group of classical agents and a group of quantum agents linked by one or more classical / quantum hybrid agents that enable communication between the two types. Communication between the hybrid agent and the quantum agents, and amongst the quantum agents, is done using quantum teleportation. A specific implementation of qHMAS has been introduced - MIq (Multi-Agent Interactive qgMuse). MIq builds on qgMuse (a single agent quantum music system) as a first attempt to design an interactive quantum computer music algorithm that utilizes the quantum advantage, and a quantum musical multi-agent system. Previous real-time music algorithms running on quantum computers have been mappings of classical computing algorithms onto a quantum architecture, with no advantage obtained.

However MIq implements Grover's algorithm for real-time rule-based music to utilize the quadratic speed-up.

References

1. Alvarez-Rodriguez, U., Sanz, M., Lamata, L. and Solano, E., 2018. “Quantum artificial life in an IBM quantum computer.” *Nature Scientific reports*, 8.
2. Anders, T., 2018. Compositions created with constraint programming. The Oxford Handbook of Algorithmic Music, p.133.
3. Baxter, J.W., Horn, G.S. and Leivers, D.P., 2007. Fly-by-agent: Controlling a pool of UAVs via a multi-agent system. In International Conference on Innovative Techniques and Applications of Artificial Intelligence (pp. 219-230). Springer, London.
4. Badawy, R., Yassine, A., Heßler, A., Hirsch, B. and Albayrak, S., 2013. “A novel multi-agent system utilizing quantum-inspired evolution for demand side management in the future smart grid.” *Integrated Computer-Aided Engineering*, 20(2), pp.127-141.
5. Blackwell, T., 2007. Swarming and music. In Evolutionary Computer Music (pp. 194-217). Springer, London.
6. Bennett, C.H., Wiesner, S. 1992. "Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states". *Physical Review Letters*. 69 (20): 2881.
7. Bennett, C.H., Brassard, G., Crépeau, C., Jozsa, R., Peres, A., Wootters, W.K. 1993. "Teleporting an Unknown Quantum State via Dual Classical and Einstein–Podolsky–Rosen Channels". *Phys. Rev. Lett.* 70 (13): 1895–1899
8. Blackwell, T.M. and Bentley, P., 2002, May. Improvised music with swarms. In Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600) (Vol. 2, pp. 1462-1467). IEEE.

9. Botti, V., Barber, F., Crespo, A., Onaindia, E., García-Fornes, A., Ripoll, I., Gallardo, D. and Hernández, L., 1995. “A temporal blackboard for a multi-agent environment.” *Data and knowledge engineering*, 15(3), pp.189-211.
10. Brooks, R.A., 1999. Cambrian intelligence: The early history of the new AI. MIT press.
11. Clair, R., Monmarché, N. and Slimane, M., 2008, December. Interactions between an artificial colony of musical ants and an impaired human composer: towards accessible generative arts. In Proceedings of the 11th Generative Art Conference.
12. Corcoles-gonzalez, A. 2019. IBM Q 5 Tenerife V1.x.x Version Log, https://github.com/Qiskit/ibmq-device-information/blob/master/backends/tenerife/V1/README.md (Last accessed 14 May 2019)
13. Finke, D. 2018. "Qubit Quality" https://quantumcomputingreport.com/scorecards/qubit-quality/ Last Accessed 13 Feb 2019
14. Gerlich, S., Eibenberger, S., Tomandl, M., Nimmrichter, S., Hornberger, K., Fagan, P.J., Tüxen, J., Mayor, M. and Arndt, M., 2011. “Quantum interference of large organic molecules.” *Nature communications*, 2, p.263.
15. Grover, L.K., 1996. A fast quantum mechanical algorithm for database search. arXiv preprint quant-ph/9605043.
16. Gruska, J., Imai, I. Power, 2001. Puzzles and Properties of Entanglement. pp 25–68, appearing in Machines, Computations, and Universality: Third International Conference. edited by Maurice Margenstern, Yurii Rogozhin. (see p 41)
17. Harrow, A., Hassidim, A. and Lloyd, S., 2008. "Quantum algorithm for solving linear systems of equations". *Physical Review Letters*. 103 (15): 150502.
18. Hatsopoulos, G.N. and Gyftopoulos, E.P., 1976. “A unified quantum theory of mechanics and thermodynamics. Part I. Postulates.” *Foundations of Physics*, 6(1),

pp.15-31.

19. Hutchings, P. and McCormack J., 2017. Using Autonomous Agents to Improvise Music Compositions in Real-Time, Computational Intelligence in Music, Sound, Art and Design: 6th International Conference, EvoMUSART 2017, Amsterdam, The Netherlands, April 19–21, 2017, Proceedings (pp.114-127)
20. Kimble, H.J., 2008. "The quantum internet." *Nature*, 453(7198), p.1023.
21. Kirke, A. 2016. "Superposition Symphony", Port Eliot Festival 29 July 2016
22. https://www.youtube.com/watch?v=-S5hU4oMWag
23. 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.
24. Kirke, A. 2019. "Applying Quantum Hardware to non-Scientific Problems: Grover's Algorithm and Rule-based Algorithmic Music Composition", International Journal of Unconventional Computing, Old City Publishing, USA (Accepted), Pre-print arXiv:1902.04237v3 [cs.AI]
25. Kirke, A. 2019b. 7th June 19 [Podcast]. 7 June. Available at https://www.alexiskirke.com/podcast/my-quantum-computer-wrote-a-podcast-5-teleportation-the-musical-quantum-computing-ai-entertainment/ (Accessed: 25 July 2019).
26. Kirke, A. 2019c. 2nd May 19 [Podcast]. 2 May. Available at https://www.alexiskirke.com/podcast/my-quantum-computer-wrote-a-podcast-3-teleporting-from-westeros-livestream-quantum-computing-ai-entertainment/ (Accessed: 25 July 2019).
27. Klusch, M., 2003. Toward quantum computational agents. In International Workshop on Computational Autonomy (pp. 170-186). Springer, Berlin, Heidelberg.
28. Larson, E.J., 2018, "A Brief History of Computer Chess", *The Quad Magazine*, TBS.
29. Linson, A., Dobbyn, C., Lewis, G.E. and Laney, R., 2015. "A subsumption agent for

collaborative free improvisation." *Computer Music Journal*, 39(4), pp.96-115.

30. Martín-López, E., Laing, A., Lawson, T., Alvarez, R., Zhou, X.Q. and O'brien, J.L., 2012. “Experimental realization of Shor's quantum factoring algorithm using qubit recycling.” *Nature Photonics*, 6(11), pp.773.
31. Miranda, E.R., 2003. "On the evolution of music in a society of self-taught digital creatures." *Digital Creativity*, 14(1), pp.29-42.
32. Murray-Rust, D.S. and Smaill, A., 2005. Musical acts and musical agents. In Proceedings of the 5th Open Workshop of MUSICNETWORK.
33. Ren, J.G., Xu, P., Yong, H.L., Zhang, L., Liao, S.K., Yin, J., Liu, W.Y., Cai, W.Q., Yang, M., Li, L. and Yang, K.X., 2017. “Ground-to-satellite quantum teleportation.” *Nature*, 549(7670), p.70.
34. Ristè, D., da Silva, M.P., Ryan, C.A., Cross, A.W., Córcoles, A.D., Smolin, J.A., Gambetta, J.M., Chow, J.M. and Johnson, B.R., 2017. “Demonstration of quantum advantage in machine learning.” *npj Quantum Information*, 3(1), p.16.
35. Scandalis, G.P., Smith, J.O. and Porcaro, N., 2013. “Physical modeling of musical instruments on handheld mobile devices.” *The Journal of the Acoustical Society of America*, 134(5), pp.4243-4243.
36. Shannon, C.E., 1950. XXII. “Programming a computer for playing chess.” The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 41(314), pp.256-275.
37. Shor, P.W., 1995. Scheme for reducing decoherence in quantum computer memory. Physical review A, 52(4), p.R2493.
38. Szymanski, L., Sniezynski, B. and Indurkhya, B., 2018. “A multi-agent blackboard architecture for supporting legal decision-making.” *Computer Science*, 19(4).
39. Tatar, K. and Pasquier, P., 2019. “Musical agents: A typology and state of the art towards Musical Metacreation.” *Journal of New Music Research*, 48(1), pp.56-105.
40. Wang, Y., Li, J., Zhang, S., Su, K., Zhou, Y., Liao, K., Du, S., Yan, H. and Zhu, S.L.,

2019. “Efficient quantum memory for single-photon polarization qubits.” *Nature Photonics*, p.1.

41. Weiss, G. ed., 1999. Multiagent systems: a modern approach to distributed artificial intelligence. MIT press.
42. Wooldridge, M., 2009. An introduction to multiagent systems. John Wiley & Sons.
43. Wooters, W.K. and Zurek, W.K., 1982. “Quantum no-cloning theorem.” *Nature*, 299, p.802.
44. Wulfhorst, R.D., Nakayama, L. and Vicari, R.M., 2003. A multiagent approach for musical interactive systems. In Proceedings of the second international joint conference on Autonomous agents and multiagent systems (pp. 584-591). ACM.