A Model of Inter-musician Communication for Artificial Musical Intelligence

Oscar Puerto & David Thue
Center for Analysis and Design of Intelligent Agents
School of Computer Science
Reykjavik University
Menntavegur 1, 101 Reykjavik, Iceland
oscar15@ru.is, davidthue@ru.is

Abstract

Artificial Musical Intelligence is a subject that spans a broad array of disciplines related to human cognition, social interaction, cultural understanding, and music generation. Although significant progress has been made on particular areas within this subject, the combination of these areas remains largely unexplored. In this paper, we propose an architecture that facilitates the integration of prior work on Artificial Intelligence and music, with a focus on enabling computational creativity. Specifically, our architecture represents the verbal and non-verbal communication used by human musicians using a novel multi-agent interaction model, inspired by the interactions that a jazz quartet exhibits when it performs. In addition to supporting direct communication between autonomous musicians, our architecture presents a useful step toward integrating the different subareas of Artificial Musical Intelligence.

Introduction

Artificial Musical Intelligence is a broad area of research that uses Artificial Intelligence (AI) techniques to build autonomous, interactive, musical systems (Collins 2006). Hiller (1959) was one of the pioneers of combining AI and music, building an application that generated musical compositions based on rule systems and Markov Chains. Later, Cope's (1992) "Experiments in Music Intelligence" used symbolic AI techniques such as grammars to generate musical compositions. Readers interested in the history of AI and music are encouraged to read Miranda's (2013) survey. We believe that the intersection of AI and music is an ideal context for the study of computational creativity.

Computational Creativity is the use of autonomous systems to generate and apply new ideas that would be considered creative in different disciplines of social life, including art, sciences, and engineering (Besold et al. 2015). In this paper, we focus particularly on the creativity that is inherent in collaborative, improvised, musical performance, and we adopt Roads's (1985) assertion that both composition and performance can be usefully tackled using AI techniques. Research and practical efforts that pursue these objectives are commonly discussed under the title "Musical Metacreation" (MuMe) (Eigenfeldt and Bown 2012), which can be well thought of as a subfield of Computational Creativity.

Musical Metacreation

The study of MuMe techniques has been approached from several different perspectives, which we roughly organize into three subareas: Algorithmic Composition, Live Algorithms, and Musical Multi-agent Systems.

Algorithmic Composition is a subarea of MuMe that seeks to automate different aspects of musical composition, including orchestration, score editing, and sound synthesis (Fernández and Vico 2013). For example, STELLA (Taube 1993) is an automated music representation system that can be used to edit musical scores.

Live Algorithms seeks to build autonomous systems that can perform in a collaborative musical setting, sharing the same privileges, roles, and abilities as a human performer. For example, IXI LANG (Magnusson 2011b) is an interpreted programming language that produces musical events in response to instructions typed in real-time – a practice known as "coding live music" (Magnusson 2011a) or simply *live coding*. Autonomy is a central concern when creating such systems, meaning that the interaction between the musician and the system must be strictly collaborative; neither should control the other. Elements of Algorithmic Composition, live electronics (e.g., IMAGINARY LANDSCAPE (Cage 1960)), and free improvisation are often combined to satisfy this constraint (Blackwell 2009). While Algorithmic Composition aims to model different elements of the composition task, Live Algorithms seeks to model the creative abilities that can be observed when human musicians perform.

Musical Multi-agent Systems is a subarea of MuMe that seeks to model the composition and performance of music as a task that requires collaboration between multiple agents. The general concepts of multi-agent systems can been applied to MuMe in two ways: multiple agents can be used to represent a single autonomous musician (e.g., a composer and a performer) (Murray-Rust, Smaill, and Edwards 2006), or the behaviour of multiple autonomous musicians can be represented as a single multi-agent system (e.g., a string quartet) (Wulfhorst, Nakayama, and Vicari 2003; Carôt, Krämer, and Schuller 2006; Thomaz and Queiroz 2009). Ideas related to computer networking are often used in this context, with communication protocols being defined and used to deliver messages between interacting agents.

Although many useful advances have been made in each of these three subareas, methods and architectures for com-

bining such advances remain largely unexplored. Recently, Bown, Carey, and Eigenfeldt (2015) developed "Musebot Ensemble" – a platform for agent-based music creation in which musical agents are developed individually and designed to accomplish a specific role in a larger ensemble. In their paper, the authors asserted that the agents in the ensemble (called "Musebots") must be able to communicate musical ideas through a set of predefined messages, toward supporting collective performances that are analogous to those of human musical ensembles.

While the Musebot Ensemble platform offers a basis for integrating various MuMe techniques, its model of how agents communicate can be improved. Specifically, it lacks support for direct communication between agents, choosing instead to allow only a special, centralized "Conductor" agent to communicate directly with each Musebot. The result is that some agents can be left unaware of decisions that are made by other agents, reducing their ability to perform well (Eigenfeldt, Bown, and Carey 2015). Furthermore, direct communication between agents is essential to certain kinds of music, where the need for real-time coordination is inherent to the musical style (e.g., small-group jazz (Bastien and Hostager 1988)).

In this paper, we propose an architecture for Musical Metacreation that offers two contributions. First, it extends the Musebot Ensemble platform with a model of direct communication between autonomous musicians. Second, it does so in a way that facilitates integrating recent advances in the MuMe subareas of Algorithmic Composition, Live Algorithms, and Musical Multi-agent Systems.

The remainder of this paper is organized as follows. We begin with a brief formulation of our challenge and follow with an overview of related work, covering each of MuMe's subareas in turn. We then present our architecture in two parts; we describe its overall structure and each of its component parts, and then explain how the parts interact with one another. We conclude by discussing our contributions and offering some suggestions for future work.

Problem Formulation

The study of human cognition and how it can be modelled has contributed to several improvements in our daily lives, such as "smart systems" that use AI techniques to ease the experiences of their users. Inspired by this perspective, we view the study and emulation of human musical abilities as an important avenue to explore in the pursuit of autonomous musical systems. As we described in the Introduction, one set of abilities that merits emulation are those that enable and support direct communication between musicians, spanning both verbal and non-verbal modes. Specifically, one's abilities to negotiate, synchronize, compose, and perform with others are essential in the context of collaborative musical improvisation (Walker 1997). Furthermore, from their case-study observation of a jazz quartet's peformance, Bastien and Hostager (1988) concluded that such musicians engage in direct verbal and non-verbal communication across three distinct modes: instruction, cooperation, and collaboration.

In this work, we seek to extend the Musebot Ensemble platform in a way that facilitates direct communication between autonomous musicians, while at the same time supporting and demonstrating the integration of different techniques from the three subareas of MuMe.

Related Work

The community of researchers studying MuMe has grown over the years, with projects like the Musebot Ensemble platform and research networks like Live Algorithms for Music (LAM) seeking to encourage interest and integration in the context of musical creativity. As a result, numerous papers have been published in the MuMe subareas of Algorithmic Composition, Live Algorithms, and Musical Multi-agent Systems, and we consider several of them in the subsections that follow. We will conclude our review of related research by summarizing recent efforts to facilitate and promote integration across MuMe's subareas.

Algorithmic Composition

Algorithmic Composition (AC) is an area of research that has contributed to several technological advances in the music industry, as many tools have been created to help musicians automate their composition tasks. In this section, we focus only on algorithms for composition that involve the application of AI, and particularly on those that do not require any human intervention.

Generative grammars and Markov Models are some of the first methods of AI that were used in AC (Rader 1974; Roads and Wieneke 1979; Rueda, Assayag, and Dubnov 2006; Keller and Morrison 2007; Morris, Simon, and Basu 2008). Abdallah and Gold (2014) offer a detailed explanation of grammar based models and Markov models, as well as a comparison between them. Researchers were also interested in evolutionary methods, in which a subset of solutions are generated from an initial set and then evaluated using a fitness function to measure their quality (Coello et al. 2007). Another method that is widely implemented in AC is Artificial Neural Networks, in which interconnected processing units are typically used to accomplish a pattern recognition task (Yegnanarayana 2009). For example, Goldman et al. (1996) developed a hybrid system based on the communication and cooperation of agents. These agents applied heuristic rules to solve problems relevant to polyphonic music composition, in real time. The melodies produced by the system are generated by neural networks that predict the expected value of each subsequent note in the melody. Although Goldman et al. successfully modelled some aspects of inter-agent communication (such as agreeing on which notes to play together), other important aspects were not included in the model (e.g., cueing transitions or negotiating over necessary tasks). Furthermore, their system focused primarily on modelling the cognitive processes of a human musician and their individual capacity to undertake different musical tasks (e.g., analyzing possible combination of notes and performing them at the same time). Nishijima and Watanabe (1993) used ANNs to learn musical styles. An overview and taxonomy of methods in AC is provided in Fernández and Vico's (2013) survey.

Despite the capacity of AC to automate certain aspects of music composition, it remains challenging to represent features that are exclusively in the domain of communication between agents, such as the ability to share musical ideas with another musician.

Live Algorithms

Much research on the topic of Live Algorithms has focused on the challenges of sound analysis and beat tracking, toward allowing autonomous musicians to synchronize their performance and effectively take turns with human musicians. An example of such work is ANTESCOFO (Cont 2008), an anticipatory score-following system based on the collaboration between two agents (an audio agent and a tempo agent). These agents allow the system to synchronize accurately with its musical partner. An interesting feature of this system is its capability to predict changes to the structure of the music and follow those changes precisely in real time, providing an atmosphere of accompaniment with its partner. Similarly to ANTESCOFO, GENJAM is capable of performing alongside a human musician. GENJAM (Biles 2002) is a jazz improvisation system that evolves musical ideas trained by a human mentor while playing them interactively with a human performer. Biles stressed that one of the most valuable features of his system is its ability to "trade fours or eights" – a part of jazz performance where soloists take turns improvising and exchanging musical ideas in a way that mimics human verbal conversation. Blackwell (2003) suggested that the interaction between musicians in a musical ensemble could be represented by the self-organization components of swarms. Alternatively, Harrald (2007) discussed interactive improvisation in musical ensembles using the game-theoretic concept of the Prisoner's Dilemma. Finally, members of the research network "Live Algorithms for Music" have provided an extensive description about of Algorithms, where they classified the different attributes that a live algorithm must have (Blackwell, Bown, and Young 2012).

While we believe that the study of Live Algorithms is essential to the development of Artificial Musical Intelligence, it has thus far only addressed the challenges of building autonomous systems that can perform together with humans. In contrast, our work seeks to understand and address the challenge of having multiple autonomous systems perform together, without any reliance on human participation.

Musical Multi-agent Systems

While the common practice of collaborative music performance is represented by musicians playing together in the same physical space, Carôt, Krämer, and Schuller (2006) provided a different perspective. In their paper, they discussed the challenges of *network music performance*, where musicians perform collaboratively from separate physical places, using the Internet as a communication channel. The notion of a network music performance provides a compelling abstraction for the study of Musical Multi-agent Systems, since interaction over a network requires a formalization of communication protocols that provide rules to govern the exchanges of messages between agents. For example, Wulfhorst, Nakayama, and Vicari (2003) described a way to implement multi-agent musical interaction while also describing the representation of cognitive musical agents. Recently, various protocols have been designed by researchers in Multi-agent Systems. Murray-Rust, Smaill, and Edwards (2006) described an example of a protocol based on speech acts, where two agents (a musician agent and a composer agent) use a formal set of musical acts to establish an accurate understanding between them. INMAMUSYS (Delgado, Fajardo, and Molina-Solana 2009) is a multi-agent system that aims to create music in response to a user-specified profile of emotional content. The system composes a piece of music that attempts to meet the given emotional profile (e.g., users can request a "happy" song).

While several efforts to create autonomous music systems have used a multi-agent approach, the majority of them have modelled *each autonomous musician* as a multi-agent system, while saying little about how a group of such musicians should coordinate or interact. Furthermore, the integration of a multi-agent system framework in the context of the Musebot Ensemble platform (i.e. representing a Musebot as a multi-agent system) remains unexplored, and we believe that such work could benefit the interaction between agents in the Musebot Ensemble platform.

Efforts Toward Integration

Workshops on MuMe are held annually in conjunction with the International Conference on Computational Creativity (ICCC), toward inspiring collaboration and integration between artistic and technological approaches. The Musebot Ensemble platform arose as a result of this effort, with the particular goal of promoting both collaboration and the evaluation of work done in the field (Bown, Carey, and Eigenfeldt 2015). Eigenfeldt, Bown, and Carey (2015) presented the first application of this platform, including specifications for their Musebots, the architecture of the Musebot Ensemble platform, and a discussion of its benefits and limitations. They further asserted that the platform does not take precedents from a human band, but in our view, the use of such precedents holds great potential for advancing our knowledge of musically creative systems (as we argued in the Introduction). Finally, Thomaz and Queiroz (2009) developed a framework that aims to integrate several ideas from previous work (including pulse detection, instrument simulation, and automatic accompaniment) in the context of Musical Multi-agent Systems. While this framework offers a convenient layer of supporting functionality (eg., synchronization), it does not support the kinds of direct communication between agents that we pursue in this work.

General Architecture

Our goal is to extend the Musebot Ensemble platform in a way that supports direct communication between Musebots while simultaneously offering a clear avenue for integrating recent advancements in MuMe's subareas. We have chosen to use the concept of a jazz quartet as a case study for this work, since jazz is a genre of music that routinely requires real-time coordination and improvisation from its players.

It thus serves as a suitable and convenient proving ground for the techniques of Algorithmic Composition, Live Algorithms, and Musical Multi-agent Systems. Furthermore, jazz performance (among humans) has been studied from the perspective of social science due to its inherent social interactivity, providing us with a solid point of reference when considering whether and how autonomous musicians can be made to play jazz. We will base our discussion on the work of Bastien and Hostager (1988), who presented a study of how four jazz musicians could coordinate their musical ideas without the benefit of rehearsal and without the use of sheet music. In their study, the authors found that the musicians communicated through a variety of different means, including visual, verbal, and non-verbal cues. We aim to model such communications between autonomous musical agents.

To extend the Musebot Ensemble platform in a way that supports direct communication between Musebots, we propose a two-level agent architecture in which, unlike previous work, each Musebot is itself comprised of multiple interacting agents. Figure 1 shows a graphical representation of our architecture, using part of Bastien and Hostager's jazz quartet as an example (a saxophonist and a pianist).

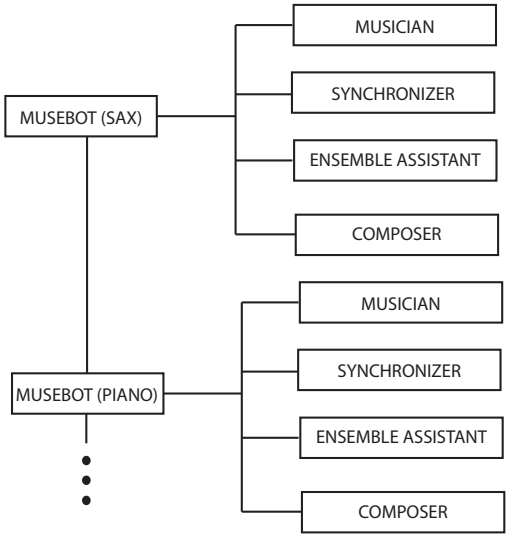

Figure 1: An example of our Musebot agent architecture. Each Musebot represents an autonomous musician and is a multi-agent system composed of four agents: a musician, a synchronizer, an ensemble assistant, and a composer.

At the top level (on the left of the figure), each Musebot represents a single autonomous musician such as a saxophonist or a pianist. At the lower level (on the right of the figure), each Musebot is made up of four different agents (musician, synchronizer, ensemble assistant, and composer) which are designed to distribute the various tasks that a Musebot should perform.

Agents at the lower level share a common goal: to help the Musebot perform appropriately as part of the Musebot Ensemble, including communicating and interacting effectively with other Musebots. To achieve this goal, a number of actions are executed by the agents based on the role that this Musebot has been given within the musical ensemble. For example, common roles in a jazz quartet are "leader/soloist" and "accompanist", and their actions could include the leader requesting an accompanist to play an introduction for the next song. We describe each of the agents separately in the subsections that follow, and then explain how they interact thereafter.

We modeled our agents using finite state machines, similarly to Barbuceanu and Fox's (1995) prior work in the context of industrial manufacturing. Barbuceanu and Fox modeled the communication between intelligent agents along a supply chain, using a framework based on a coordination language to represent different levels of coordination. This language allowed them to model a variety of interactive conversations using a variety of different finite state machines.

Musician Agent

In our architecture, the Musician agent is the primary component of a Musebot. It is responsible for carrying out the Musebot's role in the ensemble (e.g., soloist or accompanist), and in doing so, it interacts with the rest of the agents in the architecture. The behaviour of this agent is represented by the finite state machine shown in Figure 2. After registering with a service that tracks the set of musicians in the current ensemble, the Musebot that is designated the leader (e.g., by an external user) will retrieve the structure of the song, which is set prior to the performance by an external user. Then, the leader will share the structure with the rest of the musicians. Next, it will negotiate the introduction to the song by requesting for another Musebot to agree to play an introduction. In case every Musebot refuses or fails to play the introduction, the leader will continue trying to find someone that wants to cooperate. From there, it will remain ready to improvise a solo whenever it feels appropriate. Finally, it will either request an ending to the song (similarly to how it requested an introduction) or it will pass the leadership to another musician, supporting the new soloist from then on as an accompanist. For Musebots that are initially designated as accompanists, registration is followed by receiving the structure of the song from the leader and either accepting or rejecting the leader's request of play the introduction. From then on, it will play an accompaniment to the song (following the given structure) while waiting for a request to either end the song or become the next soloist.

Synchronizer Agent

One of the functions of the Synchronizer agent is to store information about events that happen during the progression of a song. For instance, when a Musebot's musician agent is ready to perform an introduction, it will inform its synchronizer of the time at which it started to play, along with the expected duration of the introduction. This information will then be stored by the synchronizer and kept available for sharing with the rest of the agents through an interaction protocol. This mechanism allows any agents that ignore this information to request and calculate the time at which the introduction will be finished, toward knowing when they must

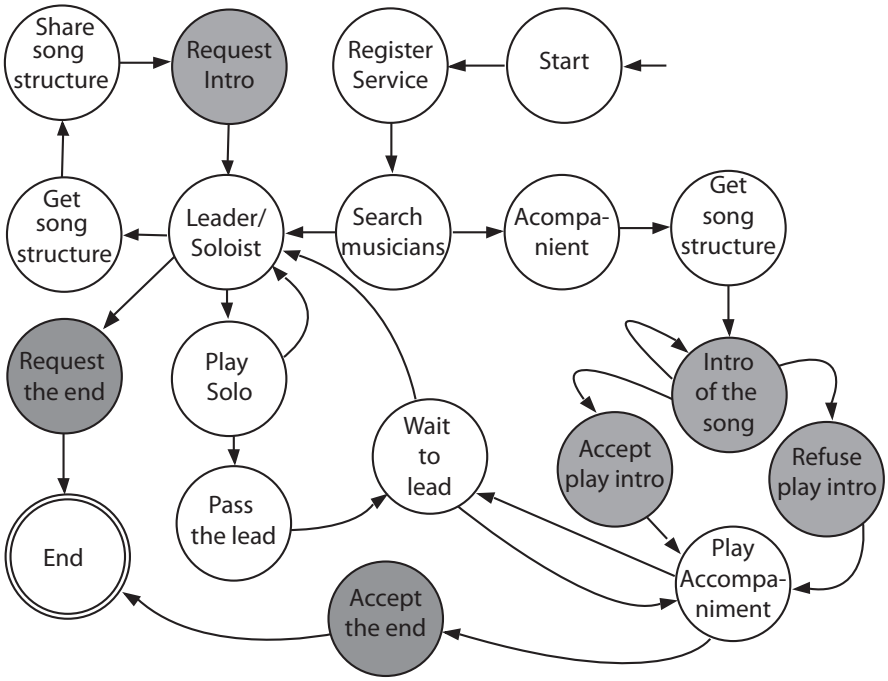

Figure 2: The Musician agent's finite state machine. Shaded states are involved in the example from Figure 3.

play their part of the song. This agent also provides a point of integration for recent advances in Live Algorithms.

Ensemble Assistant Agent

This agent serves as an intermediary between the Musebot Conductor (which is required by the Musebot Ensemble platform) and each Musebot. The Musebot Conductor provides a way for external users to control the ensemble (e.g., varying tempo, volume, or which Musebots are involved) (Eigenfeldt, Bown, and Carey 2015).

Composer Agent

The goal of this agent is to compose melodies, chord progressions, and/or solos at run-time. Each composition will depend on the role of the agent's "parent" Musebot and the instrument that it is playing (e.g., a string bassist would be less likely to play chords than a pianist). Coordinated with the help of the synchronizer, our implementation constructs each composition using JMusic, a Java library that encodes music as a symbolic representation analogous to CPN (Common Practice Notation) and plays it using the JAVA MIDI soundbank. This agent provides a point of integration for recent advances in Algorithmic Composition.

Interaction Between Agents

The interaction between agents in our architecture is managed by two different mechanisms: the Musebot Ensemble platform and a variety of interaction protocols. The interaction required by the Musebot Ensemble platform is defined by a specific set of messages (Eigenfeldt, Bown, and Carey 2015), which are exchanged between the Musebot Conductor and the Musebots. The messages are human-readable and classified into categories. For example, the message "/mc/time" is broadcasted by the Musebot Conductor to every Musebot in the ensemble, conveying the tempo of the composition for use in synchronizing the agents. Similarly, "agent/kill" is a message that indicates that the particular agent receiving this message should stop performing. In our architecture, this interaction mechanism is handled by the Ensemble Assistant agent; its principal task is to interpret these messages and transmit them to the multi-agent system.

Interaction Protocols

We developed our Musebots using JADE (Java Agent Development Framework) an agent-oriented framework designed in compliance with FIPA specifications. FIPA (Foundation for Intelligent Physical Agents) is an organization that provides an agent communication language along with standards for a number of interaction protocols (FIPA 2002). These interaction protocols are represented as sequences of messages (based on speech acts) for handling different actions between agents such as agreements, negotiation, and others (Bellifemine, Poggi, and Rimassa 1999). An example of a FIPA interaction protocol that we have implemented is shown in Figure 3.

We used the FIPA Contract Net Interaction Protocol to model part of the interaction between musicians that Bastien and Hostager (1988) described in their work. Specifically, prior to performing a song, the jazz quartet took some time to discuss which musician should play an introduction to the song. The Contract Net Interaction Protocol provides all of

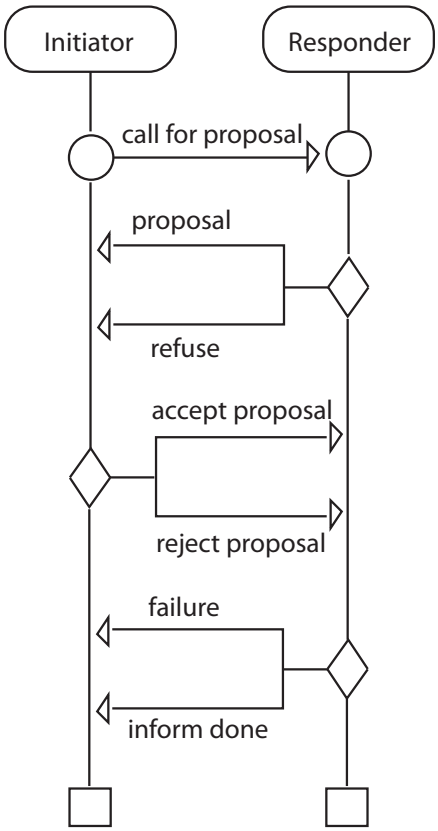

Figure 3: One of the agent interaction protocols that we implemented to support communication in our agent architecture (the FIPA Contract Net Interaction Protocol).

the necessary elements to represent this negotiation. The Musebot designated as the leader becomes the initiator of the conversation while the rest of the Musebots become responders. The initiator will send a call for proposal to the responders, proposing that one of them should play an introduction. Each responder will then reply with a proposal to play the introduction or a refusal to play it. The initiator will evaluate the received proposals, accepting one of them and rejecting the others. Once the proposal is accepted by the initiator, the responder will compose and play the introduction to the song and inform the initiator that the action was successfully completed. In case of failure by the responder, the initiator will repeat the conversation until the introduction gets played. While this example describes an interaction between agents at the top level of the architecture (e.g., a saxophonist interacting with a bassist, a drummer, and a pianist) other conversations are carried out internally by the agents at the lower level. For instance, there is a constant communication between the musician agent and the synchronizer agent each time a piece of the composition is planned to be played. Furthermore, the components of the different Musebots can also communicate one to another. One case where this occurs is between the synchronizer agents, which collectively share their information to ensure that every Musebot's musician agent will be able to coordinate timings with the others. For example, the pianist might need to know when a section of the chorus (played by all musicians) will be finished, so that it can be ready to play a solo at that time.

Discussion

We have presented an architecture for Musical Metacreation (MuMe) that pursues the goal of integration across MuMe's subareas and extends the capabilities of the Musebot Ensemble platform. At the time of writing, our implementation of this architecture is well underway, with completion expected within the next two months.

Compared to previous approaches, our architecture offers certain benefits. By extending the Musebot Ensemble platform rather than attempting to replace it, it supports some cross-compatibility with different implementations of the platform. For example, given an ensemble made up of Musebots defined using our architecture, adding an arbitrary other Musebot into the ensemble (i.e., one which does not implement our architecture) should result in as viable a performance as the Musebot Ensemble platform allows. However, since the new Musebot's abilities to communicate with the others would be effectively reduced (because it lacks our architecture), the resulting ensemble performance might be impaired. Testing these hypotheses with a variety of different Musebots remains as future work.

The second benefit offered by our architecture is its ability to represent direct communication between Musebots using standardized protocols (e.g., the FIPA Contract Net Interaction Protocol). Eigenfeldt, Bown, and Carey (2015) claimed that there is no need for conversations between the agents in the Musebot Ensemble, since everything can be handled by passing messages through the Conductor. However, having conversations based on interactive protocols allows Musebots to negotiate, coordinate, and plan autonomously in a peer-to-peer fashion, which is a closer representation of how human musicians perform.

The third benefit of our architecture is that the multi-agent design of each Musebot offers convenient points of integration for recent advances in both Live Algorithms (in the Synchronizer Agent) and Algorithmic Composition (in the Composer Agent). In addition to the precedence offered by prior work, our choice to model each Musebot as a multi-agent system has some support from the field of Neuroscience. Specifically, Zatorre, Chen, and Penhune (2007) measured the activity of different areas of the brain during music performance, finding relationships between motor function and auditory interaction. While we do not claim that the specific agents in our architecture represent an optimal design or the true operation of the human brain, we have found that utility can be gained from having a multi-agent representation for each autonomous musician.

Finally, a fourth benefit of our architecture is that it allows Musebots to mimic two modes of communication that are used commonly when (human) musicians perform, Non-verbal cooperative modes (e.g., visual and musical cues) are mimicked when our Musebots attempt to pass the lead to another Musebot, and non-democratic instructive modes (e.g.,

sharing information about the next song) are mimicked when our Musebots share the structure of the song with the other Musebots in the ensemble.

Conclusions and Future Work

There is still much to accomplish in the pursuit of Artificial Musical Intelligence and the goals of Musical Metacreation. We view our architecture as a step toward this direction, since it offers both an extension to existing, related work and a convenient basis for integrating other recent advances. Future development of the architecture will involve implementing techniques from Algorithmic Composition in the Composer Agent and from Live Algorithms in the Synchronizer Agent, as well as an analysis of the communicative behaviours in the architecture. We plan to study these behaviours by analyzing detailed transcripts of the simulations performed by our Musebot Ensembles, to identify the musical and communicative events that happen during the progression of the song and compare them to similar analyses of human musical performances. It would also be interesting to apply our model of communication in a jazz quartet to different genres of music, both with and without additional Musebots that do not implement our architecture. Finally, we hope that our integrated, communication-focused approach will encourage and support further collaborative work in Musical Metacreation.

References

Abdallah, S. A., and Gold, N. E. 2014. Comparing models of symbolic music using probabilistic grammars and probabilistic programming. In *Proceedings of the 40th International Computer Music Conference (ICMC—SMC—2014)*, 1524–1531.

Barbuceanu, M., and Fox, M. S. 1995. Cool: A language for describing coordination in multi agent systems. In *ICMAS*, 17–24.

Bastien, D. T., and Hostager, T. J. 1988. Jazz as a process of organizational innovation. *Communication Research* 15(5):582–602.

Bellifemine, F.; Poggi, A.; and Rimassa, G. 1999. JADE–a FIPA-compliant agent framework. In *Proceedings of PAAM*, volume 99, 33. London.

Besold, T. R.; Schorlemmer, M.; Smaill, A.; et al. 2015. *Computational creativity research: towards creative machines*. Springer.

Biles, J. A. 2002. Genjam: Evolution of a jazz improviser. *Creative evolutionary systems* 168:2.

Blackwell, T.; Bown, O.; and Young, M. 2012. Live algorithms: towards autonomous computer improvisers. In *Computers and Creativity*. Springer. 147–174.

Blackwell, T. 2003. Swarm music: improvised music with multi-swarms. In *In Proceedings of the AISB '03 Symposium on Artificial Intelligence and Creativity in Arts and Science*, 41–49. AISB.

Blackwell, T. 2009. Live algorithms. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Bown, O.; Carey, B.; and Eigenfeldt, A. 2015. Manifesto for a musebot ensemble: A platform for live interactive performance between multiple autonomous musical agents. In *Proceedings of the International Symposium of Electronic Art*.

Cage, J. 1960. *Imaginary landscape, no. 1: for records of constant and variable frequency, large Chinese cymbal and string piano*. Number 1. Henmar Press.

Carôt, A.; Krämer, U.; and Schuller, G. 2006. Network music performance (nmp) in narrow band networks. In *Audio Engineering Society Convention 120*. Audio Engineering Society.

Coello, C. A. C.; Lamont, G. B.; Van Veldhuizen, D. A.; et al. 2007. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer.

Collins, N. M. 2006. *Towards autonomous agents for live computer music: Realtime machine listening and interactive music systems*. Ph.D. Dissertation, University of Cambridge.

Cont, A. 2008. Antescofo: Anticipatory synchronization and control of interactive parameters in computer music. In *International Computer Music Conference (ICMC)*, 33–40.

Cope, D. 1992. Computer modeling of musical intelligence in emi. *Computer Music Journal* 16(2):69–83.

Delgado, M.; Fajardo, W.; and Molina-Solana, M. 2009. Inmamusys: Intelligent multiagent music system. *Expert Systems with Applications* 36(3):4574–4580.

Eigenfeldt, A., and Bown, O., eds. 2012. *Proceedings of the 1st International Workshop on Musical Metacreation (MUME 2012)*. AAAI Press.

Eigenfeldt, A.; Bown, O.; and Carey, B. 2015. Collaborative composition with creative systems: Reflections on the first musebot ensemble. In *Proceedings of the Sixth International Conference on Computational Creativity*, 134. ICCC.

Fernández, J. D., and Vico, F. 2013. Ai methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research* 48:513–582.

FIPA. 2002. FIPA ACL message structure specification. *Foundation for Intelligent Physical Agents, http://www. fipa. org/specs/fipa00061/SC00061G. html (30.6. 2004)*.

Goldman, C. V.; Gang, D.; Rosenschein, J. S.; and Lehmann, D. 1996. Netneg: A hybrid interactive architecture for composing polyphonic music in real time. In *Proceedings of the International Computer Music Conference*, 133–140. Michigan Publishing.

Harrald, L. 2007. Collaborative music making with live algorithms. In *Australasian Computer Music Conference*, 59–65. Australasian Computer Music Association.

Hiller, L. A. 1959. Computer music. *Scientific American* 201:109–121.

Keller, R. M., and Morrison, D. R. 2007. A grammatical approach to automatic improvisation. In *Proceedings of the Fourth Sound and Music Conference*. SMC.

Magnusson, T. 2011a. Algorithms as scores: Coding live music. *Leonardo Music Journal* 21:19–23.

Magnusson, T. 2011b. ixi lang: a supercollider parasite for live coding. In *Proceedings of International Computer Music Conference*, 503–506. University of Huddersfield.

Miranda, E. R. 2013. *Readings in music and artificial intelligence*, volume 20. Routledge.

Morris, D.; Simon, I.; and Basu, S. 2008. Exposing parameters of a trained dynamic model for interactive music creation. In *AAAI*, 784–791.

Murray-Rust, D.; Smaill, A.; and Edwards, M. 2006. Mama: An architecture for interactive musical agents. *Frontiers in Artificial Intelligence and Applications* 141:36.

Nishijima, M., and Watanabe, K. 1993. Interactive music composer based on neural networks. *Fujitsu scientific and technical journal* 29(2):189–192.

Rader, G. M. 1974. A method for composing simple traditional music by computer. *Communications of the ACM* 17(11):631–638.

Roads, C., and Wieneke, P. 1979. Grammars as representations for music. *Computer Music Journal* 48–55.

Roads, C. 1985. Research in music and artificial intelligence. *ACM Computing Surveys (CSUR)* 17(2):163–190.

Rueda, C.; Assayag, G.; and Dubnov, S. 2006. A concurrent constraints factor oracle model for music improvisation. In *XXXII Conferencia Latinoamericana de Informática CLEI 2006*, 1–1.

Seddon, F. A. 2005. Modes of communication during jazz improvisation. *British Journal of Music Education* 22(01):47–61.

Taube, H. 1993. Stella: Persistent score representation and score editing in common music. *Computer Music Journal* 17(4):38–50.

Thomaz, L. F., and Queiroz, M. 2009. A framework for musical multiagent systems. *Proceedings of the SMC* 1(2):10.

Walker, W. F. 1997. A computer participant in musical improvisation. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, 123–130. ACM.

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*, 584–591. ACM.

Yegnanarayana, B. 2009. *Artificial neural networks*. PHI Learning Pvt. Ltd.

Zatorre, R. J.; Chen, J. L.; and Penhune, V. B. 2007. When the brain plays music: auditory–motor interactions in music perception and production. *Nature reviews neuroscience* 8(7):547–558.