Autopia: An AI Collaborator for Live Coding Music Performances

Norah Lorway
Academy of Music and Theatre Arts,
Falmouth University, UK
norah.lorway@falmouth.ac.uk

Edward J. Powley
Games Academy,
Falmouth University, UK
edward.powley@falmouth.ac.uk

Arthur Wilson
School of Communication,
Royal College of Art, UK
arthur.wilson789@hotmail.com

John A. Speakman
Games Academy,
Falmouth University, UK
john.andrew.speakman@falmouth.ac.uk

Matthew Jarvis
Academy of Music and Theatre Arts,
Falmouth University
mj196235@falmouth.ac.uk

ABSTRACT

Live coding is "the activity of writing (parts of) a program while it runs" (Ward et al., 2004). One significant application of live coding is in algorithmic music, where the performer modifies the code generating the music in a live context. *Utopia*[1] is a software tool for collaborative live coding performances, allowing several performers (each with their own laptop producing its own sound) to communicate and share code during a performance. We have made an AI bot, *Autopia*, which can participate in such performances, communicating with human performers through Utopia. This form of human-AI collaboration allows us to explore the implications of computational creativity from the perspective of live coding.

[1] https://github.com/muellmusik/Utopia

BACKGROUND

LIVE CODING

Live coding is the activity of manipulating, interacting and writing parts of a program whilst it runs (Ward et al., 2004). Whilst live coding can be used in a variety of contexts, it is most commonly used to create improvised computer music and visual art.

The diversity of musical and artistic output achievable with live coding techniques has seen practitioners perform in many different settings, including jazz bars, festivals and algoraves --- an event in which performers use algorithms to create both music and visuals that can be performed in the context of a rave. What began as a niche practice has evolved into an international community of artists, programmers, and researchers. With a rising interest in "creative coding", live coding is well positioned to find more mainstream appeal.

At algoraves, the screen of each performer is publicly projected to create transparency between the performer and the audience. The Temporary Organisation for the Permanence of Live Algorithm Programming (TOPLAP) make it clear how important the publicity of the live coder's screen is in their manifesto draft: "Obscurantism is dangerous. Show us your screens" (TOPLAP, 2010).

A central concern when performing live electronic music is how to present "liveness" to the audience. The public screening of the performer's code at an algorave is often discussed in regards to this dynamic between the performer and audience, where the level of risk involved in the performance is made explicit. However, in the context of the system described in this paper, we are more concerned with the effect that this has on the performer themselves. Any performer at an algorave must be prepared to share their code publicly, which inherently encourages a mindset of collaboration and communal learning with live coders. Additionally, the system we describe here puts the audience in the loop: allowing for a type of real-time audience feedback mediated by technology.

COLLABORATIVE LIVE CODING

Collaborative live coding takes its roots from laptop orchestra/ensemble such as the Princeton Laptop Orchestra (PLOrk), an ensemble of computer based instruments formed at Princeton University (Trueman, 2007). The orchestra is a part of the music research community at the University and is concerned with investigating ways in which the computer can be integrated into conventional music making. PLOrk attempts to radically transform those ideals (Trueman, 2007). Each PLOrk meta instrument consists of a laptop, multi-channel hemispherical speaker and a variety of control devices such as game controllers, sensors amongst others (Trueman, 2007). The orchestra consists of 12-15 students and staff ranging from musicians, computer scientists, engineers and others and uses a combination of wireless networking and video in order to augment the role of the conductor (Trueman, 2007).

UK based live coding ensembles such as the Birmingham Ensemble for Electroacoustic Research (BEER) based at the University of Birmingham

have taken influence from ensembles such as PLOrk, but differ in terms of the way they integrate communication and collaboration within the ensemble. The ensemble was formed in 2011 by Scott Wilson and Norah Lorway (Wilson et al., 2014) and began as an "exploration of the potential of networked music system" for structured improvisation (Wilson et al., 2014). The ensemble works primarily in the SuperCollider (SC) language[2] and the JITLib (Just in Time Library)[3] classes in SC for basic live coding functionality (Wilson et al., 2014). In terms of ensemble communication and coordination, BEER uses Utopia (Wilson et al 2013), a SuperCollider library for the creation of networked music application which builds on the Republic quark[4] and other such networked performance systems in SuperCollider. Networked collaboration in live coding was present from the inception of live coding where multiple machines are clock-synchronized exchanging TCP/IP network messages (Collins et al., 2003). Utopia aims to provide a more modular approach to networked collaboration, featuring enhanced flexibility and security over other existing solutions. It also provides an efficient way to synchronize communication, code and data sharing over a local network. Unlike an ensemble such as PLOrk which uses a human conductor such as in a traditional orchestra, Utopia eliminates the need for this, allowing for a more streamlined shared approach, where performers collectively make musical decisions.

MOTIVATION

COMPUTATIONAL CREATIVITY

Using an AI bot within the context of a networked live coding performance, is an idea that builds on a study undertaken by McLean and Wiggins (2010), regarding live coding towards Computational Creativity.

Computational Creativity can be described as the aim of "endowing machines with creative behaviours" (Pasquier et al., 2017), and systems designed to do so can be put to practical uses from simulating and automating existing human processes (creativity as it is), to discovering novel outcomes (creativity as it could be) (Pasquier et al., 2017), which could be valuable to the "scientific study of creativity" (Wiggins and Forth, 2018). In the context of this proposal, we are concerned with the latter.

The McLean and Wiggins (2010) study highlighted a view among live coding practitioners that the code resulting from their practice contains an element of the programmers style, and that "many feel they are not encoding a particular piece, but how to make pieces in their own particular manner" (McLean and Wiggins, 2010). This is a sentiment that is echoed by Wiggins and Forth (2018) in the following statement:

> "*In a manner akin to the extended-mind theory of consciousness (Clark and Chalmers, 1998), the live coder becomes attuned to thinking with and through the medium of code and musical abstractions, such that the software can be understood as becoming*

[2] https://github.com/supercollider/supercollider
[3] http://doc.sccode.org/Overviews/JITLib.html
[4] https://github.com/supercollider-quarks/Republic

> *part of the live coder's cognition and creativity"* (Wiggins and Forth, 2018).

Through a process of "reflexive interaction" (Wiggins and Forth, 2018), the human performer(s) and artificial agent each influence the actions of the other. Entering into a "complex feedback loop" (Fiebrink and Caramiaux, 2018), the artificial agent becomes an "imperfect mirror" of the human performer(s) (Wiggins and Forth, 2018). We propose that through the analysis of the artificial agent's behaviours, we can extend our understanding of what constitutes "valuable" musical output, while challenging existing dogmatic approaches to live coding practice, and techniques relating to the chosen programming language (SuperCollider), where the formalisation and subsequent manipulation of syntax trees can provide new insight to the language's potential. Finally, it can provide insight into the nature of creativity in general, by analysing emergent behaviour from the bot.

Ultimately, our motivation can be summarised in the following quote:

> *"When the computer becomes a conversation partner, or a boat rocking us in unexpected directions, we may find that the technologies we build become more useful, more musical, more interesting than our original conceptions"* (Fiebrink and Caramiaux, 2018).

GAMIFICATION

There has been work on the use of gamification to facilitate creativity (Kalinauskas, 2014). This generally draws upon the idea of flow (Csikszentmihalyi, 2009) — the idea being that flow is important to creativity, and that including some game-like elements in a creative software or process can help to put users into this flow state. Taken further, this leads to the idea of casual creators (Compton and Mateas, 2015) — creative tools whose interface is designed to promote a "playful, powerful, and pleasurable" user experience (unlike more traditional creative software where "powerful" would take precedence over the other two). Aiming for playfulness in this context can also promote curiosity and experimentation (Nelson et al., 2018).

Gamification has also been studied in the context of collective creativity (Skarzauskiene and Kalinauskas, 2014). There are obvious analogies between collaborating on creative tasks and playing a multiplayer game, and the ideas used in the latter to foster collaboration (or, in some cases, competition) may prove useful in the former. For instance, the Female Interface Research Ensemble (FIRE) based at the University of Birmingham, used Utopia and gamified collaborative approaches in their algorave performance during The New Interfaces for Musical Expression conference in 2014 in London, UK (Lorway et al., 2014). As another example, Nilson (2007) proposes a number of game-like exercises, many of them collaborative and/or competitive, to be used by live coders in a practice context.

We propose taking a gamified collaborative creative environment and adding a "bot" — an AI agent which interacts in the same way as a human would. Bots in multiplayer games are often used as sparring partners for offline practice matches, or to make up the numbers when not enough human players are available for a game, however the fact that the play style of

bots is different to that of humans tends to change the dynamics of the game. We are interested in studying whether the same is true for a collaborative live coding performance — how does the introduction of one or more bot performers change the dynamics of the performance?

THE BOT

In our previous paper on Autopia (Anonymous, 2019a) we proposed a bot that participated in collaborative live coding performances in the same way as a human performer. Such a system would incorporate two components: a chatbot interface to the Utopia chat, and a genetic programming system to generate SuperCollider code. The first of these remains as future work, however we now have a functioning prototype of the second part.

The bot implements the Template-Based Object-Oriented Genetic-Programming algorithm (Anonymous, 2019b) in C#, set to automatically construct SuperCollider code from a series of pre-defined templates. These templates are built using a genetic sequence, which is used to select the initial template, usually a single line of SuperCollider code which has been broken into its constituent parts, as strings. At present the templates were hand-coded into the system and are fixed at runtime, however future work may allow new templates to be extracted from other performers' code (shared over Utopia) whilst the system runs. The variables used in these templates are filled in as values read directly from the genetic algorithm or as variables created at an earlier point in the automatic construction of the code.

This occurs in 3 phases: an initialization phase, which generates a series of initial sine waves, a modification phase which alters those waves and an execution phase which plays the generated sounds. Each of these phases corresponds to its own library of templates. The generated code is then injected into the SuperCollider IDE by simulating keypresses, mimicking the appearance of a human live coder typing the code in (albeit at super-human speed). A simulated press of Shift+Enter then causes the generated code to be executed and produce sound.

Code can be generated in a batch and bred together, representing a generation. A call can be made which takes two agents (genetic sequences which may be used to generate SuperCollider code) and breed them together using a simple genetic crossover algorithm to produce a new, offspring agent. Using this technique, multiple generations of agents may be generated which can be used, with selection, to breed against a fitness function.

AUDIENCE COLLABORATION

Any evolutionary computing approach requires a fitness evaluation function. In the current version of Autopia, the fitness evaluation comes directly from the audience. We set up a web server along with a wi-fi router to which the audience were invited to connect their smartphones. Upon connecting, the audience member is given a simple slider ranging from 0 to 100 and the instruction "Score what you're hearing" (Figure 1). On each generation of the evolutionary algorithm, each individual in the population is played for approximately 10 seconds. At the end of the 10 seconds, the slider values chosen by the audience are averaged and this

value is taken as the fitness of that individual. Individuals ranked highly by the audience are more likely to be selected as parents for the next generation.

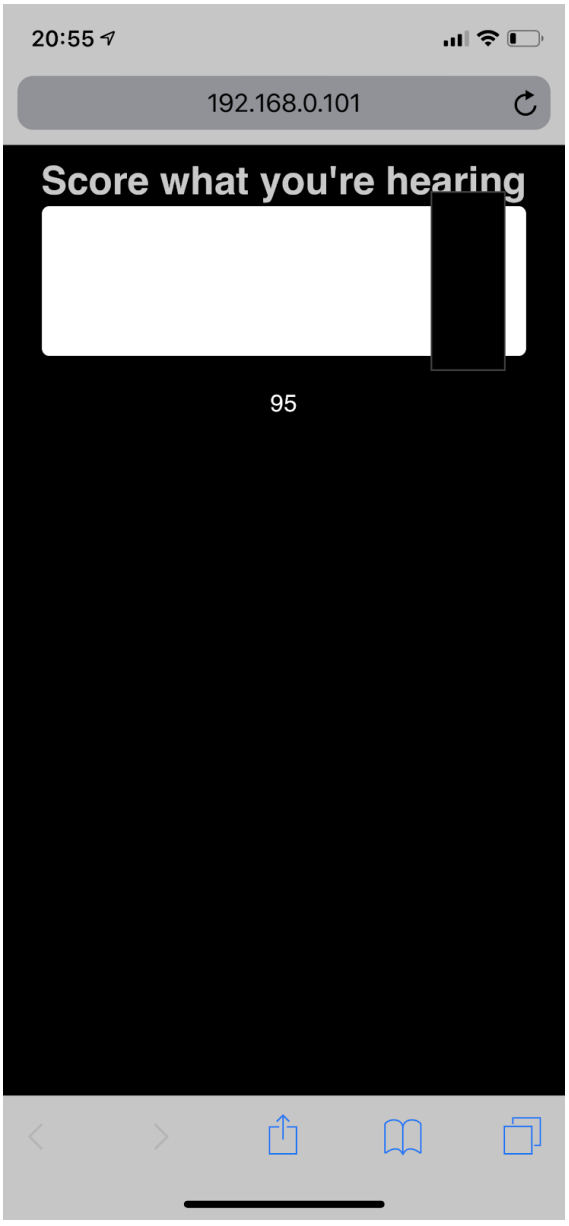

Figure 1. The web-based interface for audience participation.

This voting system introduces an aspect of gamification to the system, with the audience participating as "players". A similar voting-based idea, but amongst performers, was previously tested in Republic. This allows participants to vote each other up and down, giving them feedback on their contributions (and for the bot, explicitly shifting the fitness evaluation towards the preferences of the other performers and the audience).

THE PERFORMANCE

Figure 2. A photograph from the debut performance of Autopia.

In June 2019 we tested Autopia in a performance at the Academy of Music and Theatre Arts, Falmouth University. The performance consisted of Autopia playing alone for around 1 hour with audience participation to shape the evolution of sound, at which point two live coding performers (two of the authors) joined the stage and performed alongside Autopia for around 30 minutes. Throughout the performance the Autopia interface was projected onto a large screen (Figure 2), showing the SuperCollider IDE, an oscilloscope of the output signal, the Utopia interface, and the logging output from the bot's evolutionary algorithm. A video excerpt from the performance is available online.[5]

[5] https://vimeo.com/349044280

FUTURE WORK

As noted above, currently the GP system is based on hand-coded templates (lines of SuperCollider code which have been extracted and marked up with variable placeholders by hand). Whilst the system can already generate a wide variety of sounds, it is limited by the selection of templates coded in. The next step is to allow the system to expand its library of templates as it runs. When other (human) performers execute code and it is shared through Utopia, the GP system will add the code to its own population, to introduce variety to the gene pool and allow Autopia to build upon what the other performers are doing.

The fitness evaluation in the GP system currently comes from audience participation. This does have some limitations, namely that the speed of evolution is limited to the speed at which the population members can be played to the audience, and sometimes (especially early in the evolutionary process) the sounds may be silence, unpleasant noise or otherwise undesirable.

We propose to evaluate the fitness of individuals in the population through a basic machine listening process: individuals will be run through a second instance of SuperCollider, and the system will perform a frequency analysis (i.e. Fourier transform) on the resulting audio output. This will be compared to a frequency analysis of the audio output being produced by the other performers. The more similarity in frequency characteristics between the two, the higher the fitness. As a first step this should at least weed out those population members which produce undesirable results (such as silence or white noise), though clearly the refinement of the fitness measure is a fruitful line of future work. Collins (2006) suggests a number of more sophisticated machine listening approaches which may prove useful, and provides a JavaScript library[6] implementing several of these techniques.

CONCLUSIONS

Using AI in the context of live coding is relatively new and unexplored. The idea of AI collaborators has been well explored in Computational Creativity, including in musical contexts, however the process used by the AI can sometimes be opaque to observers and is almost certainly quite different to the process used by human performers. By combining AI with live coding we hope to overcome this — humans and bots are participating at the same level and in the same way (i.e. by manipulating code) — bringing the human-AI ensemble closer to liveness. This also goes towards achieving the goal, set out by the Birmingham Laptop Ensemble (Booth and Gurevich) in their manifesto, of "integration, collaboration and the blurring of the distinctions between, composer-performer-collaborator in a democratic non-authoritarian ensemble" (BiLE).

The state of flow is clearly desirable in creative activities. The use of gamification can potentially be a powerful way of getting participants into this flow state, as well as the idea of voting borrowed from multiplayer games helping to facilitate the goals described above. The effect of introducing a bot performer on the human performers' flow state

[6] https://github.com/sicklincoln/MMLL

is less easy to predict — our hope is that the bot will act as a "conversation partner" (Fiebrink and Caramiaux, 2018) and thus provide inspiration during a performance.

REFERENCES

Anonymous. Paper on Autopia. Workshop proceedings, 2019a.

Anonymous. Paper on template-based genetic programming. Workshop proceedings, 2019b.

BiLE. BiLE manifesto. https://bilensemble.wordpress.com/manifesto/.

Graham Booth and Michael Gurevich. Proceeding from performance: An ethnograpy of the Birmingham Laptop Ensemble.

Andy Clark and David J. Chalmers. The extended mind. Analysis, 58:7–19, 1998.

Nick Collins. Towards Autonomous Agents for Live Computer Music: Realtime Machine Listening and Interactive Music Systems. PhD thesis, University of Cambridge, 2006.

Nick Collins, Alex McLean, Julian Rohrhuber, and Adrian Ward. Live coding in laptop performance. Org. Sound, 8(3):321–330, December 2003. ISSN 1355-7718. doi: 10. 1017/S135577180300030X. URL http://dx.doi.org/10.1017/S135577180300030X.

Kate Compton and Michael Mateas. Casual creators. In Proceedings of the 6th International Conference on Computational Creativity, pages 228–235, 2015.

Mihaly Csikszentmihalyi. Creativity: Flow and the Psychology of Discovery and Invention. Harper Perennial Modern Classics. HarperCollins e-books, 2009. ISBN 9780061844034. URL https://books.google.co.uk/ books?id=aci_Ea4c6woC.

Rebecca Fiebrink and Baptiste Caramiaux. The machine learning algorithm as creative musical tool. In Roger T. Dean and Alex McLean, editors, The Oxford Handbook of Algorithmic Music. Oxford University Press, 2018. doi: 10.1093/oxfordhb/9780190226992.013.19.

Marius Kalinauskas. Gamification in fostering creativity. Social Technologies, 4:62–75, 10 2014. doi: 10.13165/ ST-14-4-1-05.

John R. Koza. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA, 1992. ISBN 0-262-11170-5.

Norah Lorway, Brenna Cantwell, and Edie Pearce. FIRENGINE: a new interface for gestural interaction in live laptop performances. In Proceedings of New Interfaces for Musical Expression (NIME), 2014.

Alex McLean and Geraint A. Wiggins. Live coding towards computational creativity. In Proceedings of the First International Conference on Computational Creativity, 2010.

Mark J. Nelson, Swen E. Gaudl, Simon Colton, and Sebastian Deterding. Curious users of casual creators. In Proceedings of FDG Workshop:

Curiosity in Games, 2018.

Click Nilson. Live coding practice. In Proceedings of the 7th International Conference on New Interfaces for Musical Expression, NIME '07, pages 112–117, New York, NY, USA, 2007. ACM. doi: 10.1145/1279740.1279760. URL http://doi.acm.org/10.1145/1279740.1279760.

Philippe Pasquier, Arne Eigenfeldt, Oliver Bown, and Shlomo Dubnov. An introduction to musical metacreation. Comput. Entertain., 14(2):2:1–2:14, January 2017. ISSN 1544-3574. doi: 10.1145/2930672.

Aelita Skarzauskiene and Marius Kalinauskas. Fostering collective creativity through gamification. 10 2014.

TOPLAP. Manifestodraft. https://toplap.org/wiki/ManifestoDraft, 2010.

Dan Trueman. Why a laptop orchestra? Org. Sound, 12(2):171–179, August 2007. ISSN 1355-7718. doi: 10.1017/ S135577180700180X. URL http://dx.doi.org/10.1017/ S135577180700180X.

Adrian Ward, Julian Rohrhuber, Fredrik Olofsson, Alex Mclean, Dave Griffiths, Nick Collins, and Amy Alexander. Live algorithm programming and a temporary organisation for its promotion. In Olga Goriunova and Alexei Shulgin, editors, read me, Software Art and Cultures. 2004. ISBN 87988444040.

Geraint A. Wiggins and Jamie Forth. Computational creativity and live algorithms. In Roger T. Dean and Alex McLean, editors, The Oxford Handbook of Algorithmic Music. Oxford University Press, 2018. doi: 10.1093/oxfordhb/9780190226992.013.19.

Scott Wilson, Norah Lorway, Rosalyn Coull, Konstantinos Vasilakos, and Tim Moyers. Free as in BEER: Some explorations into structured improvisation using networked live-coding systems. Computer Music Journal, 38(1):54–64, 2014. doi: 10.1162/COMJ_a_00229.