Evolving Instrument Based on Symbiont-Host Metaphor: a Commensal Computation

Jonas Abernot[1], Guillaume Beslon[1], Simon Hickinbotham[2], Sergio Peignier[1], and Christophe Rigotti[1]

[1] Univ Lyon, INSA-Lyon, CNRS, INRIA, LIRIS, UMR5205,
20 av. A. Einstein, F-69621, Villeurbanne, France,
jonas.abernot@ergodicite.lautre.net
guillaume.beslon,sergio.peignier@inria.fr
christophe.rigotti@insa-lyon.fr
[2] University of York, Heslington, York YO10 5GH, UK,
simon.hickinbotham@york.ac.uk

Abstract. We present a suite of tools under development for music creation and performance which combines an evolving music generation algorithm with an evolving motion-sensitive interface. The system can be used as a dance-driven musical instrument that allows an immediate physical interaction with music, because the evolutionary algorithm generates *live code*, that evolves during the performance. The classification of the physical moves required to control the music also evolve, leading to a performance both in sound and in motion. To reach this goal we introduce *commensal computation*, a way of organising artificially evolving system inspired by symbiotic relationships observed in biology. As a consequence the system is split in two *organisms*: a *host* which generates music from input moves; and a *symbiont* which simplifies the representation of the input moves for the host. We describe how it allows evolution, and sketch the artistic possibilities.

Keywords: evolutionary computation, digital musical instrument, motion sensor

1 Introduction

This work is part of the EvoEvo European project (www.evoevo.eu). The project aims to develop new evolutionary approaches in computer science and to produce algorithms based on the latest understanding of molecular and evolutionary biology. Its ultimate goal is to address open-ended problems, where the specifications are either unknown or too complicated to express, and to produce software able to operate in unpredictable, varying conditions.

In this contribution, we propose to build a perpetually evolving musical instrument as a sandpit for experiments in open-ended evolution. This will allow us to both test and develop our ideas within a real-world open environment and to explore the possible complex interactions between an end-user and this kind

of software. We present a suite of tools under development for music creation and performance, which combines an evolving music generation algorithm with an evolving motion-sensitive interface that can be used as a dance-driven musical instrument. As described generically in (Miranda & Wanderley, 2006), digital musical instruments combine a control, i.e. a way to capture human actions, and a music generator. We want this instrument to evolve not only with respect to the produced music, but also to evolve the controls themselves. This would allow the performer to introduce new moves while playing, leading to a performance both in sound and in motion.

The main challenge we want to address is the trade-off between the pace of innovation and the stability of the system, both in terms of control from the performer and of musical aesthetics. To this end, we propose a system that makes it possible for the motion to influence the music and for the music to influence the motion - in other words a *symbiotic* relationship, regulated by bio-inspired evolution and allowing flexibility for improvisation.

Symbiotic relationships are well known in nature and are continually adapting to new situations. They not only have a way of evolving to find solutions in a given environment, but also include a way of keeping enough flexibility to respond appropriately to changes in the environment. A well-documented example of this is the evolution of organelles in Eukaryotes, where two species of bacteria (Sagan, 1967) live together so closely that one organism eventually lives inside the other. The open-endedness observed in natural systems is analogous to the manner in which performers improvise, adjusting their own actions in response to the need of the performance and the actions of the other performers and musicians. Inspired by these ideas, we imagine to evolve virtually a musical instrument formed from two components. A *symbiont* which consumes dance moves and produces simplified moves, and a *host*, which generates live coded music based on these.

In the first part of this article we develop this idea and introduce *commensal computation*, a generalisation of this symbiosis-inspired way of organising genetically inspired programs. Then, we present our solutions for both systems. We present each system, and how the solutions we propose help us pursuing our will of an evolving musical instrument.

2 Commensal Computation

When designing an interacting self-organised system such as the one we envision here, one of the central problems is how to develop a system that is autonomous enough to entertain or even surprise its user and, in the same way, that is rational enough to serve the goals it has been built for (here to play music). This tension is at the heart of what is called “living technologies”: if autonomy is one of the core properties of life, how can a technology be simultaneously “alive” and “controllable”? As said above, how can our music generation tool simultaneously evolve and play the music the artist wants to play?

We propose here a bio-inspired approach to this problem in the form of what we call *commensal computation*. In biology a commensal (from the Latin *cum mensa* – at the same table) interaction is a form of mutualism between two organisms where the association is not detrimental but not obviously beneficial to the partners (Hooper & Gordon, 2001). The idea of commensal computation is based on one of the main functions of the gut microbiota: nutrient processing. Gut microbes degrade ingested substances that would otherwise be non-digestible or even harmful to the gut (Hooper, Midtvedt, & Gordon, 2002). This role enables the organism to uptake nutrients originating from a wider variety of sources than would otherwise be the case. In other words, gut microbes preprocess the complex flow of nutrients and transfer the results to their host organism, helping it to regulate its feeding and to extract specific nutrients. Importantly, while doing so, the microbiota lives its own life, and changes and evolves according to its environment, i.e., according to what its host eats. In addition, the nutrient processing by the microbiota enables the host organism to gain resources it uses to survive. In other words, the commensal association of the microbiota and the host contains a part of autonomy (the microbes) and a part of control (the host).

We propose to organise our computational system following the manner in which host and microbiota are engaged in a mutualist association. In commensal computation, the complex data (here generated by the sensors – see below) are preprocessed by a virtual microbiota that transforms them in "digestible" data that the processing system can use. However, such an architecture differs from the classical preprocessing-processing structure in that the preprocessing is here performed by an evolving community of virtual bacteria that uptake data, transform them in recognisable objects (symbols, clusters, classes...) and feed them to the main processing system.

When applied to our music generation system based on motion sensors, this architecture results in a host fed by motion-data and producing music, and a bacterial community that preprocess the motion-data helping the host to interpret the moves. Both "organisms" thus eat at the same table (the motion) and co-evolve. The music produced by the host depends on the command-objects produced by the virtual bacteria. The motion fed to the bacteria depends on the movements the user makes in reaction to the music they hear.

3 System Overview

The architecture of our project is described in figure 1: The performer's moves are captured via a set of wireless sensors that combine accelerometers, gyroscopes and magnetometers (giving a hint about absolute orientation), built by the HIKOB company (`www.hikob.com`). The sensors are worn by the performer. At the moment, we can deploy up to 8 sensors with 9 variables each but the server-client architecture of the software managing the sensors allows us to deploy even more sensors, or to set them up on several performers. Moreover, the commensal architecture we propose also allows the use of heterogeneous sensors,

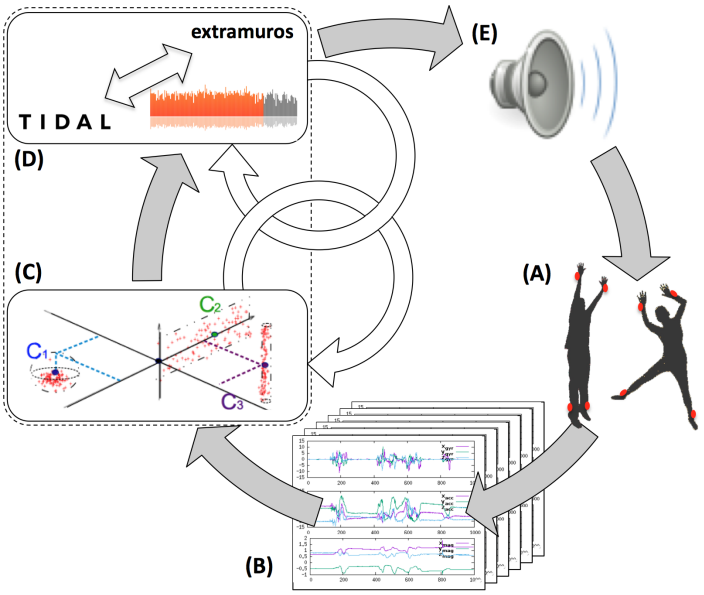

Figure 1 – Architecture of the instrument. (A) Data produced by the performer(s) are captured by multiple sensors, producing a high dimension temporal signal (B). The signal is then processed by a symbiont (C) that performs subspace-clustering on the input data. Each move is associated to a cluster, and the identifiers of these clusters are then used as symbols to feed the host (D). Thanks to this dimension reduction the host is then able to process them to produce music (E), which provide a feed-back to the performer(s).

for instance cameras tracking motion from outside. The data processing and music generation is then based on three bio-inspired metaphors:

Resource cycling In the system the data represent a food resource that is cyclically processed from the initial source (the performer) to the music generating system and back to the performer (grey arrows on figure 1).

Meta-organism The commensal architecture forms a meta-organism (dashed rectangle on figure 1) made of two levels: a host and a population of symbionts. The host uses the symbols (i.e., the cluster identifiers) produced by the best symbiont (the one building the most compact clusters). The music itself is generated by the interaction between both levels.

Co-evolution The two white arrows on figure 1 symbolise the co-evolution of the two parts of the instrument: Host and symbiont each evolve separately but the data-flow they receive are inter-dependent due to the global data cycling in the system.

4 The Symbiont

The symbiont in our system is the component that is responsible for generating symbols from the input moves fed in by the performer. The various movements made by the dancer are all different from each other. However, the dancer is likely to produce movements that can be related to abstract groups representing more canonical moves. These canonical moves, when identified, can be the symbols destined for the host. So, the system requires an algorithm to perform this identification and to link the movements of the performer to the corresponding canonical moves.

This algorithm must be able to handle the high dimensionality of the input. Thus, at first glance, we might need a dimensionality reduction step as Principal Component Analysis. But useful dimensions might not be the same for each move, as some moves might be focused for instance on a hand, a foot, on a rotation or on an acceleration. Thus, the problem, here, is not of the kind solved by a principal component analysis, it is a subspace clustering problem. Each canonical move can be described in its own subspace, each dimension of the given subspace being related to a particular variable of a given sensor. When a move is captured, it can then be simply compared to the clusters and be associated to the most similar canonical move. The algorithm must adjust the canonical moves and their subspaces over time, and also has to detect and create new canonical moves if the performer uses new ones. We have to chose a *dynamic* subspace clustering algorithm able to handle changing data. More than that, it has to be able to handle data from the open-ended world of moves the performer is going to invent.

Recently, Chameleoclust, a new evolutionary algorithm that tackles the subspace clustering problem was presented in (Peignier, Rigotti, & Beslon, 2015). This algorithm simulates a population of evolving individuals, where these individuals represent candidate solutions to a subspace clustering problem. Each individual is characterized by its genome, that is composed of a list of genes, where these genes are tuples coding for the locations of the cluster centers and for the dimensions used by these clusters. Such an individual will be a symbiont in the commensal architecture, and the cluster centers that are encoded by its genome will be a set of canonical moves. Chameleoclust incorporates different bio-inspired features from the in silico experimental evolution formalisms of (Knibbe, Coulon, Mazet, Fayard, & Beslon, 2007) and (Crombach & Hogeweg, 2007), such as a variable genome length, both functional and non-functional genes, and mutation operators including large chromosomal rearrangements. These mutational operators may modify the genome elements but also the genome structure, and provides the algorithm with a large degree of freedom. This evolvable genome structure allows the algorithm to adapt the number of clusters it produces and also their average dimensionality without specific parameter tuning.

For each of the symbiont simulated by Chameleoclust, there are two processing parts: one is the association of a new move to the nearest canonical move, and the other is the update of the canonical moves themselves. The associa-

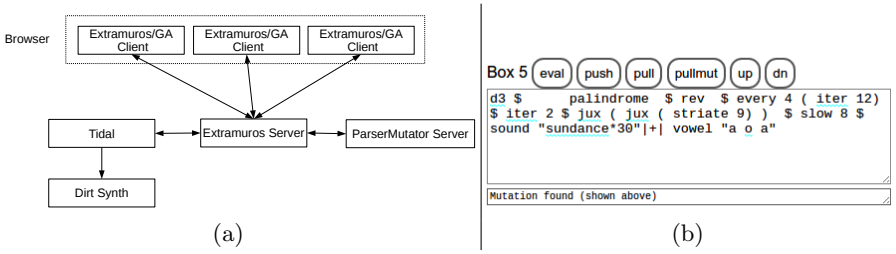

Figure 2 – Overview of the host. Panel (a) shows the relationships between the clients, web server, parsermutator server and the Tidal / Dirt audio synthesis engine. Panel (b) shows a text entry box in the web client, containing the control buttons and an example of Tidal pattern.

tion step is fast, while the update step is a more time-consuming. So, we used a version of the subspace clustering algorithm still inspired on Chameleoclust, but more efficient in order to handle the stream of moves on-the-fly. In this version, individuals are also encoding candidate solutions and evolve to tackle the subspace clustering problem, but the algorithm does no longer rely on a fine-grained genome representation. This version simply keeps track of the cluster center locations and of the number of genes, but not of the detailed gene encoding. In addition, to achieve a more efficient exploration of the space of possible clustering models, this version of the algorithm uses coordinates from data objects themselves (picked in the stream) to build and adjust each coordinate of candidate cluster centers.

5 The Host

The host in our system is the component that is responsible for generating music from the input symbols received from the symbiont. For this application, we require a mean of specifying musical input that can contain simple but highly structured commands. The solution we have chosen is Tidal (McLean, 2014), a musical notation system that is used for *live coding*: the use of domain-specific languages (DSLs) to improvise new musical pieces in real time.

The host is based on Extramuros (Ogborn, Tsabary, Jarvis, Cardenas, & McLean, 2015), which facilitates network-based collaboration on live coding performances. The development of the evolutionary component for this system is described in (Hickinbotham & Stepney, 2016). This platform provides a natural bridge between the symbiont and the host, as illustrated in figure 2.

In the Extramuros system, a population of patterns is held in the Document Object Model (DOM) of a client web page. Each performer can maintain his own population of patterns in this way without interfering with other live coders in the performance. Aesthetically pleasing patterns are 'pushed' to the genetic algorithm population, and mutants of these patterns are 'pulled' from it later

on. The fitness of a pattern is increased or decreased according to the way it is used in the performance.

Since the system is designed to operate in a live performance, it is important to let the performer(s) decide when to do this. To facilitate this flexibility, we added five new buttons to each text-entry box, giving a total of six operations:

1. **eval**: Send the pattern to the Tidal interpreter
2. **push**: Send the pattern to the population (via the ParserMutator)
3. **pull**: Pull a pattern from the population
4. **pullmut**: Request a mutation of the current pattern from the ParserMutator
5. **up**: Increase the fitness of the current pattern (if it exists in the population)
6. **dn**: Decrease the fitness of the current pattern; remove it from the population if fitness < 0

We have described how the host of our instrument is able to produce music from a web browser. However, in the current situation the performer is not interacting with it via buttons, she/he is feeding it with moves. These moves, that are likely to be complex, with inherent noise, as described in part 3, are processed by the symbiont before being transmitted in a more abstract description to the host.

6 Experiments

Even though this system is a work in progress, several experiments have been conducted with it or part of it. This section presents two experiments, one with only the symbiont part of the system, linked to a non-evolving music generator, and the other one with a preliminary version of the whole system. The setup of each experiment is exposed, and illustrated with a video of the working system.

In all these experiments, sensors collected the acceleration in three directions, the rotation speed in three directions and the magnetic field in three direction. The measurement frequency was 50 Hz. The tempo was set to 60 beats per minute, and the data were collected over each 1 second interval and aggregated by computing the mean and the variance of each measure. The resulting 18 descriptors over the last second of movement were then fed to the symbiont.

6.1 Symbiont alone

In this experiment, the host part was a static system in the sense that it contained a set of predefined sample sounds. When the symbiont used a new cluster identifier (i.e., a new symbol), then this identifier was simply associated to a new sample sound among the predefined ones. Each time the symbiont outputed a symbol (at each beat), the host simply played the corresponding sample. Sounds could be longer than one beat, and thus could lead to a superposition of different sample sounds. However, if a sample was not finished when the same sample was triggered again, then it was not superimposed to itself, but started over from the

beginning. In this experiment, a dancer wore two sensor units, each associated to a different symbiont, thus two sounds were triggered at each beat. Videos of preliminary working sessions with two professional dancers of the Anou Skan Company are available at (EvoMove-team, 2016a) and (EvoMove-team, 2016b).

6.2 Complete system

This experiment was made during the *Conference on Computer Simulation of Musical Creativity* in June 2016. The proposed video (given as supplementary material) is a short excerpt from this demonstration. It was one of the very early use of the whole system, made by inexperienced performers.

The host still provided sound samples, but contrarily to the experiment reported previously, these sample sounds were not static and were evolved live. Another important difference is that the identification of a move by the symbiont did not trigger one single play of a sound. In this experiment each move was associated to an action that started or stopped an infinite loop playing the sample.

A performer (on the right in the video) interacted with the sample evolution and could see a representation of all available sound samples. He was able to patch them directly, or to express preferences toward some of them, and managed the pace of their evolution. A second performer (on the left in the video) wore a single unit sensor on one of his arms. As in the previous experiments, the symbiont was fed by data from the sensor, and it output cluster identifiers. Each cluster identifier was associated by the host to one of the evolving sound sample, and to one command on this sample: a start or a stop action. When a sample was started, then it was played repeatedly until a correspond stop command was issued.

7 Conclusion

This contribution is not about a system able of autonomously producing music, but is related to *interactive composition* (Chabade, 1984), in which the production is the result of an interaction between performer and installation. A dancer/musician should be able to bring the system to an interesting state and play with it. However, our application shares some challenges with Evolutionary Music research field. In particular, we need a general enough music generation mechanism, that would not restrict the performer into a limited musical style (McCormack, 2005). Our proposed solution for this is *live coding*, as presented in section 5. The definition of the fitness function, i.e. the function used to evaluate the output music, has also been and is still a big bottleneck in evolutionary music (McCormack, 2005; Galanter, 2010). Our current solution, as described in section 5, is only a prototype, but might give some hints about how to infer subjective aesthetic evaluation from performer use of the system.

The host and the symbiont have been designed and implemented, as well as the part collecting the data from the sensors and the bridge to the audio

player. The host and the symbiont were integrated in the proposed commensal architecture. We reported experiments (recorded on videos) showing preliminary investigations of the possibilities given by the prototype.

Many questions remain open. In particular, how to obtained a more autonomous system that could be used by a single performer (the one wearing the sensors) and without the need of a second performer guiding the evolution of the host?

We have not enough feedback about how to co-evolve live a host-symbiont pair, during performances. The first commensal prototype, presented in this article, will facilitate this experimentation, and could lead to imagine complementary mechanisms to further develop the system into a fully fledged musical instrument.

We are also looking forward exploring the artistic landscape opened by this instrument. Although a lot of examples exist, as Variations V by Merce Cunningham and John Cage or Virus//Antivirus by Cie Lanabel – which uses a set of sensors similar to ours – the novelty of our commensal approach is related to finding ways to map the motion space onto the music space. Indeed, as the correspondences between sounds and moves are not chosen in advance, the performer would have to find their way through a musical landscape built on-the-fly just for and from them. We also wonder what kind of piece could born from the uninterrupted and ineluctable evolution of the music medium. Going beyond the artistic result itself, we also are curious of what could be the relationship between the performance and this non-deterministic, sometimes life-like instrument, that may *tire*, or that may *excite*. In short we look forward to a new exploration of relationship to real-time evolving technology in general.

Acknowledgements

This research has been supported by EU-FET grant EvoEvo (ICT-610427). The authors would like to thank Sophie Tabakov and Laurent Soubise of the Anou Skan Dance Company (`www.anouskan.fr`) who were early testers of the symbiont. Many thanks also to Leo Lefebvre and Anthony Rossi who contributed to the development of some modules used in the system. Other support: Christophe Rigotti is a member of LabEx IMU (ANR-10-LABX-0088).

References

Chabade, J. (1984). Interactive composing: An overview. *Computer Music Journal*, *8*(1), 22-27.

Crombach, A., & Hogeweg, P. (2007). Chromosome rearrangements and the evolution of genome structuring and adaptability. *Molecular Biology and Evolution*, *24*(5), 1130-1139.

EvoMove-team. (2016a). *EvoMove AnouSkan 1.* Retrieved from `https://www.youtube.com/watch?v=p_eJFiQfW1E`

EvoMove-team. (2016b). *EvoMove AnouSkan 2.* Retrieved from `https://www.youtube.com/watch?v=E85B1jJOiBQ`

Galanter, P. (2010). The problem with evolutionary art is. In *Proc. of the international conference EvoApplications: Applications of evolutionary computation* (pp. 321–330). Springer, Berlin Heidelberg.

Hickinbotham, S., & Stepney, S. (2016). Augmenting live coding with evolved patterns. In *Proc. of the international conference on evolutionary and biologically inspired music, sound, art and design* (pp. 31–46). Springer-Verlag, New York.

Hooper, L. V., & Gordon, J. I. (2001). Commensal host-bacterial relationships in the gut. *Science*, *292*(5519), 1115–1118.

Hooper, L. V., Midtvedt, T., & Gordon, J. I. (2002). How host-microbial interactions shape the nutrient environment of the mammalian intestine. *Annual review of nutrition*, *22*(1), 283–307.

Knibbe, C., Coulon, A., Mazet, O., Fayard, J.-M., & Beslon, G. (2007). A long-term evolutionary pressure on the amount of noncoding DNA. *Molecular Biology and Evolution*, *24*(10), 2344–2353.

McCormack, J. (2005). Open problems in evolutionary music and art. In *Proc. of EvoWorkshops: Applications of evolutionary computing* (pp. 428–436). Springer, Berlin Heidelberg.

McLean, A. (2014). Making programming languages to dance to: live coding with Tidal. In *Proc. of the ACM SIGPLAN international workshop on functional art, music, modeling & design* (pp. 63–70). ACM, New York.

Miranda, E. R., & Wanderley, M. M. (2006). *New digital musical instruments: Control and interaction beyond the keyboard.* A-R Editions, Middleton, Wisconsin.

Ogborn, D., Tsabary, E., Jarvis, I., Cardenas, A., & McLean, A. (2015). Extramuros: making music in a browser-based, language-neutral collaborative live coding environment. In *Proc. of the international conference on live coding.* ICSRiM, University of Leeds.

Peignier, S., Rigotti, C., & Beslon, G. (2015). Subspace clustering using evolvable genome structure. In *Proc. of the ACM genetic and evolutionary computation conference (gecco)* (pp. 575–582). ACM, New York.

Sagan, L. (1967). On the origin of mitosing cells. *Journal of theoretical biology*, *14*(3), 225–74.