REMEMBERING THE FUTURE : AN OVERVIEW OF CO-EVOLUTION IN MUSICAL IMPROVISATION.

David Plans Casal

University of East Anglia
Brunel University
david.plans@brunel.ac.uk

Davide Morelli

University of Pisa
info@davidemorelli.it

ABSTRACT

Musical improvisation is driven mainly by the unconscious mind, engaging the dialogic imagination to reference the entire cultural heritage of an improvisor in a single flash. This paper introduces a case study of evolutionary computation techniques, in particular genetic co-evolution, as applied to the frequency domain using MPEG7 techniques, in order to create an artificial agent that mediates between an improvisor and her unconscious mind, to probe and unblock improvisatory action in live music performance or practice.

1. DEMONS VERSUS BOUNDED RATIONALITY

> *"Composing is a slowed-down improvisation; often one cannot write fast enough to keep up with the stream of ideas."Arnold Schoenberg, "Brahms the Progressive", 1933, in Style and idea, 1950, as quoted in Nachmanovich, 1990.*

We believe that the processes behind musical improvisation, and therefore to a great extent those of composition, are not the result of an unbounded rationality at work, empowered solely by reasoning power, experience and musical training (Demons), but are more intrinsic, frugal and driven by a bounded rationality (5), influenced and sometimes entirely driven by the unconscious.

We see successful free improvisors (Jarrett, Parker, Bailey, etc.) as performing an impossible feat : creating music compositions out of thin air, and on the spot. Free improvisation is about listening and what Gladwell (6) calls 'thin-slicing'', in that an expert improvisor is able to actively listen to her environment (other musicians, the room, the echoes in her memory) and 'thin-slice' the content for clues she recognises as departure and arrival points, dialogic references and surprises, and then respond according to how her unconscious is directing her. Listening is a skill that can be acquired through training and matured through experience; and so might thin-slicing, if one were able to control the environment in which an improvisation happens, and involve learning agents built specifically to unblock the unconscious.

We propose to build such an agent, using methods inspired by Todd and Werner's work on genetic co-evolution algorithms (14) and the ABC group's theories on fast and frugal heuristics (5), as well as Michael Casey's MPEG7 feature recognition techniques (**? ?**) as implemented in his Soundspotter framework. Our work, needless to say, stands on the shoulder of giants. As well as Todd, Werner, Gigerenzer and Casey, we have benefited from the amazing vision of Thomas Grill, whose C++ framework for the Puredata environment allowed us to quickly prototype and think our way through our ideas with minimal programming pain, and from the amazing leaps of progress made by others, from Lewis' 'Voyager' (10) to Miranda's mimetic agents (11) and cellular automata systems.

Our criteria for this agent are: It must take input from live music improvisation as its main body of data and primary control device, and it must enable the player to navigate a map of unconscious musical gestures (musical phrases and their timbral, rhythmic interrelationships) by providing an evolving 'mirror' to her playing.

Many artificial agents have been built to provide independent and collaborative music improvisors, and we will outline a few that have influenced our research below; we will however firstly examine some of the further issues that have influenced the design of ours, whom we will call Frank, in honour of Todd and Werner's Frankensteinian Methods.

Our criteria for this agent are:

- It must take input from live music improvisation as it main body of data and primary control device.
- It must enable the player to navigate a map of unconscious gestures by providing an evolving 'mirror' to her playing.

Many artificial agents have been built to provide independent and collaborative music improvisors, and we will outline a few that have influenced our research below; we will however firstly examine some of the further issues that have influenced the design of ours, whom we will call Frank, in honour of Todd and Werner's Frankensteinian Methods.

1.1. Remembering the Future

Improvisation happens in an environment full of snap judgments, where previous experience, cultural heritage and current information acquired through listening all help enable the improvisor to make decisions quickly.

Snap judgments can be made in a snap because they are light in processing expense and frugal in nature (6; 5), and successful decision making in improvisation relies on a carefully nurtured balanced between bounded (deliberate) and unbounded (instinctive, unconscious) rationalities. In instinctive behavior, thin slices of experience are captured and processed by the unconscious to give us ready answers to questions which need an immediate answer, such as 'If I don't put my hand forward, will that door slam into me¿, or 'Do I like this person enough to trust them with my child for 5 minutes?', or 'Is the violin player about to reference the motif I introduced 3 minutes ago, and should I join in¿.

In the work of the improvisor, in her practice, there is an inescapable need to unblock unconscious action, so that these snap judgments can occur and meaningful musical material emerge. Improvisors such as Evan Parker rarely practice from a notated score, and choose instead to focus on gestural devices that have developed in their playing during decades of practice and live performance with others. His is then a self-contained ecology, where Lewis' dialogic imagination (10) can work unencumbered by the (sometimes essential) constraints of the score, composer, player cycle; but, it relies heavily on an almost completely *exploratory* process and ecological reality, which takes decades to evolve to the mature point where the process is almost solely E-creative (4; 2).

In trying to unblock, we need the agent to be free from the traditional bounds of composition. As George Lewis points out (10):

> "If we do not need to define improvised ways of producing knowledge as a subset of composition, then we can simply speak of an improvising machine as one that incorporates a dialogic imagination."

Frank tries to activate the dialogic processes of the improvisor's mind, in particular the quicksilver heuristics involved in finding improvisational pathways within musical material through instrumental practice. Our aim is to enable a state of flow in the player, in which her dialogic imagination can be receptive to the kind of motivic/harmonic play mature Jazz musicians experience.

Behind any unconscious action, there is encyclopedic knowledge that we cannot necessarily access through willed action, and this points at an important issue: really skilled improvisors are able not just to recall on demand past events and current motivic/harmonic changes; they are also able to 'remember' the future: they can project their imagination into future events. The essential process behind this kind of projection into time is typical prefrontal cortex activity: humans and some animals use it to predict whether a gap is too long to jump over, a challenger too fierce to fight, or a crossing to dangerous to attempt. We use our previous experience, and play the possible event (successful crossing or getting run over) in our minds. The combination of prefrontal simulation and experiential memory could be called an unconscious remembering or replay of an event which may (fight) or may not (flight) happen: this is why we call it remembering the future.

Unconscious remembering, or noetic (8; 9) (to know that an event occurred without remembering) memory, is, we propose, at the heart of dialogic interplay in musical improvisation, and the design of our system will attempt to prod the human improvisor to better understand the temporal connections underlying this process.

1.1.1. Creating a door to the unconscious

Goldstein, Gigerenzer and Todd's work (7; 5) on the recognition heuristic, the simplest of their fast and frugal heuristics, which proves that efficient decision-making does not need very large amounts of information and can also rely on lack of knowledge, can be linked to Jacoby's unconscious recollection (noetic) as explained above. It is clear that in an environment where we are forced to act on unconscious data to make a decision, we will make links that simply are not, and have never been there; when pushed, we invent.

We propose that simply giving a musician an ongoing evolutive stream of mirrored (feeding back and forth from human to agent) sound gestures could potentially trigger a frugal process of recognition, and the E-creative processes. These could in turn help to navigate her unconscious to focus and direct (deliberate thinking) improvisational and compositional processes. Through the same process (thin-slicing) that we follow when selecting fruit at a market or choosing a mate, she could select from incoming streams of music gestures, as though 'shoppping' for her own bits of unconscious dialogic metadata (links to other music gestures, by same player or someone else).

This paradigm, where we propose Frank fits, is meant to activate the dialogic imagination of an improvisor through live practice.

Our objective is to lead the player to unfound links between motivic/harmonic material, such as the links Schonberg mentioned when writing about his Chamber Symphony, which Gartland-Jones and Copley quote when illustrating the possible uses of a goal-directed GA agent (4). Schoenberg saw two completely disconnected themes, and would have erased theme b, but opted to wait:

> 'About twenty years later, I saw the true relationship. It is of such a complicated nature that I doubt whether any composer would have cared deliberately to construct a theme in this way; but our subconscious does it involuntarily.' (13)

As with the recognition heuristic, we want the improvisor to 'benefit from their own ignorance' (5) p.57 and to discover the hidden relationships between themes.

1.2. Previous Methodologies

Evolutionary computing has, by now, a long record of application in musical research; to date, it remains generally focused on either computer music or musical cognition concerns (**?**). We will not address the whole background of this work here, but instead will focus on the technniques that inspired our work.

Two excellent surveys and general inquiries into the use and general application of genetic algorithms in music (out of many others) are Gartland-Jones and Copley's 'The Suitability of Genetic Algorithms for Musical Composition' (4) and Burton and Vladimirova's 'Generation of Musical Sequences with Genetic Techniques' (3), both of which focus on methodologies (theirs and others) that attempt to use genetic algorithms to generate musical material. Some, such as Biles' 'GenJam' (1), work within premises such as 8th-note derivation within strict Jazz timelines, others, such as the IndagoSonus system, attempt to bypass the fitness bottleneck through GUI-driven evolutionary targets. In the case of Todd and Werner's coevolution principle, the generation of musical material is based on populations of hopeful singers and critics coevolving at the same time. In the case of Lewis' 'Voyager', with its legacy of Forth programming, and rule-based structure, we see a competent improvisor, but one that is necessarily fixed within the numerical MIDI domain (as are most others), and not as able to capture the gestural nuances embedded in timbre variation that can occur within musical improvisation.

We do not here have the space to outline each in turn. Todd and Werner's genetic co-evolution algorithm became our choice of implementation for Frank, due to its emphasis on evolving criticism, an essential part of the thin-slicing machine (Frank) we wanted to build, and of cultural heritage as a phenomenon. However, as pointed out by Miranda, Todd, and Kirby (12), within Todd's co-evolution, which evolves hopeful male singers and female critics in parallel, there is a 'puzzling fundamental question' which is left unaddressed: where do the expectations of the female critics come from? We will address this question in our system in a brute, fundamental way: by allowing the human improvisor to determine the scale of expectancy as a variable. Since the improvisor's live input has a direct effect on the female genotype, this gets around the expectancy provenance.

2. TECHNICAL IMPLEMENTATION

For the rest of this paper, we will refer to one particular use case of Frank, for consistency purposes. In this case, one human player at any instrument (in this case, piano) will be the live input, through normal analog to digital conversion feeding into the Puredata environment, within which we host the objects (written in C++, using Flext) that constitute our agent, Frank. The player is given a Puredata patch to control some of the facets of Frank, such as initial lexical database creation and starting the GA process.

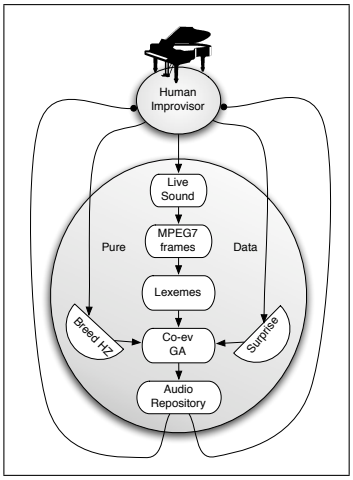

Figure 1. Frank : a high-level overview of the framework.

The Frank framework consists of the following elements, which feed into each other in sequence as the live sound input comes into Puredata:

- MPEG7 feature extraction
- Acoustic Lexemes database creation from clustered MPEG7 frames
- Co-evolution GA, taking live sound, and two other variables as input
- Audio repository, which can be static or built from live sound

A high-level overview of Frank's design and data flow can be seen in figure 1, which outlines the four steps above and shows where human input and reception happen.

2.1. Co-evolving strings of MPEG7 vectors

In our implementation of Todd's co-evolution (14), we decided to address what Todd calls the structure versus novelty trade-off by focusing on novelty or creativity, and isolating structure to the functions of the matching algorithms using Casey's methods. In this way, navigating the musical solution space would be a question of finding structure within evolved solutions, and not before it (thus avoiding setting a priori knowledge of the musical space, as rules).

We should here point out the difference between our implementation of co-evolution, and Todd and Werner's; in section 4.2 of their Frankensteinian paper (14), 'Co-evolving hopeful singers and music critics', from which we took most of our inspiration, they outline their third scoring method (or fitness/expectation system), the 'surprise preference scoring' method. Briefly, every female builds an expectation matrix *while* listening to a male's song. We have not, at this stage, implemented this scoring method, and have focused solely on similarity, so that we could more easily manage the progression from bare Soundspotter methods to co-evolving features. We aim to

implement suprise preference in a coming version, so to allow for internal gene movement.

We give our system a division of tasks: the male population in our genetic algorithm produces many answers to the musical space question (an incoming query by way of real-time audio, such as a piano chord). The female population criticises those answers, isolates winners, and breeds with them. Just as in Todd and Werner's idea, this process is about generating answers, testing those against some criteria and repeating the process. Our objective was for those criteria to evolve in real-time, and not be set by the system maker. We saw that using MPEG7 vectors (provided by Casey's Soundspotter methods), essentially frames in the musical spectra of ongoing real-time audio derived from FFT analysis, could provide us both with an ongoing influence and set of criteria, but also with a genotypical unit with which we could start the process of evolution. For example we could assign a number of incoming concatenated MPEG7 frames to be our female genotype, which would trigger imitation, and let co-evolution take over from there.

The ability of co-evolution to generate synchronic diversity (**?**) through the process of sexual selection (speciation) would then save our system from eradicating diversity and reaching a 'perfect' solution, which would be musically uninteresting.

2.2. Creation of the Lexemes Database

It would have been unfeasible to simply take the MPEG7 floating point vector numbers, as they are too large (64 of them per frame); we needed to take the MPEG7 matches to incoming audio and simplify them for our genotype. The k-means algorithm offers a simple clustering method, which we chose to apply to Frank's design. Hashing might be needed for very large datasets, but we were confident k-means would perform well for smaller (up to 2 hours) of music.

We then decided to cluster MPEG7 frames using k-means, labeling them 'lexemes', following the Casey convention. Every MPEG7 frame consists of both audio data and MPEG7 features data. We kept the features data only, and thus reduced the dataset we would have to deal with even further. These clusters form our working genotype: a musical gesture. Figure 2 outlines the lexemes creation process in the context of the whole system.

The essential process is: after MPEG7 feature extraction has concluded, features are stored in a database. At this point, the k-means algorithm is used to cluster the features stored in this database; we use the Euclidean distance between features to derive these clusters, and once the k-means algorithm has produced an optimised set of clusters, the center of each cluster becomes a lexeme. These then become our lexemes database.

Creating a database of lexemes by feeding Frank an existing static audio file, and then analysing and tagging incoming, live audio as lexemes, gives us a working framework, with the potential for a common dialogic lexicon to emerge over time. We tested this using the use case

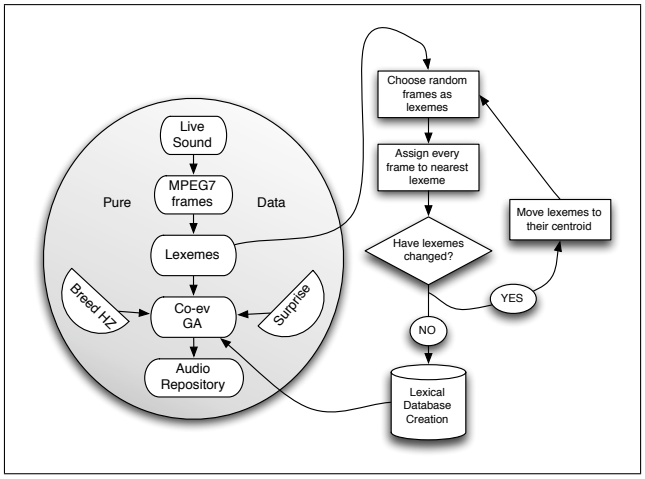

Figure 2. Figure 2 : lexemes creation.

above, and gave Frank its first bit of music: Luciano Berio's 'Omaggio a Joyce'. We were at this point able to query by matching live input, with a much reduced data set, and could query whole musical gestures by forcing Frank to look at particular lexemes (giving it the lexeme ID) and navigating the cluster around it.

Once we arrived at a system design that would allow us to breed populations of lexemes, by applying the genetic co-evolution algorithm on top of Soundspotter existing C++ methods, the need arose to find a navigational space for our data: we choose 2 dimensional matrices, which would allow us to compute the Ecludiean distance between lexemes, to allow our system to consider issues of musical form over time. In the next section, we explore the lexeme database creation and issues surrouding it further.

2.3. Witness the Fitness : Frank's core job

Having achieved a lightweight and simple enough clustering method, we had a working framework for our genotype, and set out to implement our version of the co-evolution algorithm, to breed populations of individuals with sequences of these lexemes as their genotype.

We have 2 populations: males and females. Every generation each female will choose a male and breed. Every male can breed more than once but can also not breed at all if no female choses him. The fitness function implements how a female will chose her male.

Let us clarify, at this point, how the female genotype is constructed, as it represents the essential input from the lexemes database into the general population: after feature extraction is put into a data array, the use of the k-means algorithm gives us a clustered centers; these become the lexemes, and are stored in their own array. When this array is full, a new genotype is created, and this becomes the female genotype and is inserted into the population as a new individual.

When a male and a female breed they create a new string randomly taking part of the genotype from the mother

and part from the father (the crossover), and the new individual can be both male or female (randomly). Mutations then occur, and this produces new musical material in the form of phenotypes, or winning individuals. The GA process runs many times per second, since we want to our solutions to evolve over time, and to produce fluent musical production.

Winning individuals are then given back as queries by ID to Soundspotter, which can then point to the right sequence of MPEG7 frames. At this point, winners are proposed to the human player in the form of live sound.

2.3.1. The fitness function in detail

Our fitness function matches the male lexeme string (genotype) against the female one. A first step involves creating a matrix expressing the probability of finding a particular lexeme in a particular position, so we have as many columns as the lexemes in a string and as many rows as the number of lexemes in our database.

This is one of the reasons why we need to cluster lexemes, so that we just need a row for each group of lexemes instead of a row for each lexeme, lowering memory and cpu usage.

We fill this matrix with statistical data taken from the female's genotype: we make several copies of it, starting from different positions (close to each other in the matrix) and we use them as a statistical source. Here is an example. Let us say we have an original string: 1 2 3 3 3. If we derive from this string starting at a different position, we could get 2 3 3 3 1, and if we do it again, 3 1 2 3 3.

Figure 3 shows two tables with the statistical data we gain from this process, the second one with the normalised data.

	Pos1	Pos2	Pos3	Pos4	Pos5
Lex1	0.3	0.3	0	0	0.3
Lex2	0.3	0.3	0.3	0	0
Lex3	0.3	0.3	0.6	1	0.6

Female statistical data

	Pos1	Pos2	Pos3	Pos4	Pos5
Lex1	1	1	0	0	0.5
Lex2	1	1	0.5	0	0
Lex3	1	1	1	1	1

Normalised female statistical data

Figure 3. Figure 3 : Female genotype statistical data.

After normalisation of the data, we can compare the male's genotype using this matrix, to see how close it is to the female's. If we take an example male genotype of 2 2 3 3 2, compare it to 2 2 3 3 3, we know it will score 1 + 1 + 1 + 1 + 0 = 0.8 (4 out of 5 = 0.8). For comparison, a random string would statistically score 0.66, and a perfect copy of the genotype would score 1. If we take another male with 2 1 2 3 3, scoring 0.9, the female would prefer this one over the former. This latter one is in fact very close to a simple translation of the female genotype, at this point.

To implement this fitness function, and the matrix statistics shown above, we used two important techniques: imprecise pattern matching, and weight matrices, to give us recognition of similar as opposed to just identical strings in the case of the former, and to achieve this similarity recognition in a fuzzy way, in the latter.

In order to achieve the computations above, we used the Euclidean distance between lexemes, storing these in our matrices, in order to derive degrees of similarity.

3. OVERVIEW AND CONCLUSIONS

3.1. Overview of Frank in practice

When our improvisor starts working with Frank, she has several things to do: she will first have to either load previously recorded sound, or start live recording, into a Puredata table. Then, giving Frank an 'extract' message will begin the initial lexical database creation. At this point Frank works to minimise the average distance between frames and lexemes. The improvisor can then start the co-evolution process by sending a 'startGA' message, which will initiate a thread running the GA up to 10 times a second. After this, she can send a further 'ga' message, which will prompt Frank to start listening to her playing, feeding her output into the population as new genotypes (lexemes), choosing winners from the population and playing those back to her. At this point, she can affect the direction of evolution through two important variables: Surprise, the degree of similarity the females expect from the males (this is our brute force answer to co-evolution's 'puzzling question'), and Breeding Frequency, which controls the maximum number of generations Frank will deal with in one second. The latter allows the improvisor some control over the speed of general change in the populations.

3.2. Conclusions

We believe Frank is successful in enacting what the ABC group call the recognition heuristic is hard at work: a thin-slicing environment where each musical gesture produced by the live improvisor is answered by many possible solutions by Frank, so that the hidden motivic and harmonic relationship we want the improvisor to discover becomes the Criterion of the heuristic. Frank then becomes the Mediator, and in its Surrogate Correlation through the suprise/similarity and breed frequency functions, influences the probability of recognition in the improvisor, whose mind in turn uses the recognition heuristic to infer the Criterion (the hidden relationship).

However, in preliminary evaluation of Frank in live performance, we have found that it doesn't at this point allow for deep insight of a complex performer's own musical language. It is a complete prototype; all its components are finished and working in their present state, it is able to create unexpected and non-obvious solutions to musical behaviour and its created materials are coherent with the given musical context. However, its hypothesis is in very early stages of testing. In the next stages of investigation, we intend to monitor ventromedial prefrontal cortex activity testing using functional MRIs, to show blood

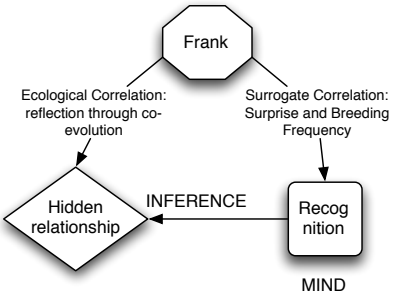

Figure 4. Figure 3 : The recognition heuristic enabled by Frank.

flow levels to that region of the brain during improvisatory interaction with Frank. We estimate that an initial group of ten to fifteen test cases will give us a reliable body of data, from which to begin formulating the actual effectiveness of Frank as capable of stimulating the recognition heuristic we outline, and of ultimately unblocking unconscious action in improvisation successfully. The addition of variable length lexemes, and the implementation of surprise in the female population's fitness evaluation, is being considered as essential progress for the eventual completeness of the algorithm.

References

[1] BILES, J. A. Genjam: A genetic algorithm for generating jazz solos. In *International Computer Music Conference* (1994 1994), p. 131.

[2] BODEN, M. *Computer models of creativity*. Handbook of Creativity. Cambridge University Press, 1998-10-28 1998, p. 351.

[3] BURTON, A. R., AND VLADIMIROVA, T. Generation of musical sequences with genetic techniques. *Computer Music Journal 23*, 59–73.

[4] GARTLAND-JONES, A., AND COPLEY, P. The suitability of genetic algorithms for musical composition. *Contemporary Music Review 22* (2003), 43–55.

[5] GIGERENZER, G., TODD, P. M., AND GROUP, A. R. *Simple Heuristics That Make Us Smart (Evolution Cognition)*. Oxford University Press Inc, USA, 2000-10-26 2000.

[6] GLADWELL, M. *Blink: The Power of Thinking Without Thinking*. Penguin Books Ltd, 2006-02-23 2006.

[7] GOLDSTEIN, D. G., AND GIGERENZER, G. Models of ecological rationality: The recognition heuristic. *Psychological review 109* (01 2002), 75–90.

[8] JACOBY, L. L. A process dissociation framework: Separating automatic from intentional uses of memory. *Journal of Memory and Language 30*, 5 (1991/10), 513–541.

[9] JACOBY, L. L., TOTH, J. P., AND YONELINAS, A. P. Separating conscious and unconscious influences of memory: Measuring recollection. *Journal of Experimental Psychology: General 122* (06 1993), 139–154.

[10] LEWIS, G. E. Too many notes: Computers, complexity and culture in voyager. *Leonardo Music Journal 10* (12 2000), 33–39.

[11] MIRANDA, E. R. *Mimetic Development of Intonation*. 2002.

[12] MIRANDA, E. R., KIRBY, S., AND TODD, P. M. On computational models of the evolution of music: From the origins of musical taste to the emergence of grammars. *Contemporary Music Review 22* (2003), 91–111.

[13] STEIN, L. *Style and Idea, Selected Writings of Arnold Schoenberg*. St. Martin's Press, 1975 1975.

[14] TODD, P., AND WERNER, G. Frankensteinian methods for evolutionary music composition in griffith, 1999.