ON ARCHITECTURE AND FORMALISMS FOR COMPUTER-ASSISTED IMPROVISATION

Fivos Maniatakos * ** Gerard Assayag * Frederic Bevilacqua ** Carlos Agon*
* Music Representation group (RepMus)
** Real-Time Musical Interactions team (IMTR)
IRCAM, UMR-STMS 9912
Name.Surname@ircam.fr

ABSTRACT

Modeling of musical style and stylistic re-injection strategies based on the recombination of learned material have already been elaborated in music machine improvisation systems. Case studies have shown that content-dependant regeneration strategies have great potential for a broad and innovative sound rendering. We are interested in the study of the principles under which stylistic reinjection could be sufficiently controlled, in other words, a framework that would permit the person behind the computer to guide the machine improvisation process under a certain logic. In this paper we analyze this three party interaction scheme among the isntrument player, the computer and the computer user. We propose a modular architecture for Computer Assisted Improvisation (CAO). We express stylistic reinjection and music sequence scheduling concepts under a formalism based on graph theory. With the help of these formalisms we then study a number problems concerning temporal and qualitative control of pattern generation by stylistic re-injection.

1. INTRODUCTION

New computer technologies and enhanced computation capabilities have brought a new era in real-time computer music systems. It is interesting to see how artificial intelligence (AI) technology has interacted with such systems, from the early beginning until now, and the effect that these enhancements have had when setting the expectations for the future. In the 2002's review paper [1], computer music systems are organized in three major categories: (1) Compositional, (2) Improvisational and (3) Performance systems. Concerning the limits between what we call computer improvisation and computer performance, the authors of [1] state the following:

"... Although it is true that the most fundamental characteristic of improvisation is the spontaneous, real-time, creation of a melody, it is also true that interactivity was not intended in these approaches, but nevertheless, they could generate very interesting improvisations."

According to [1], first improvisation systems did not address directly interactivity issues. Algorithmic composition of melodies, adapatation in harmonic background, stochastic processes, genetic co-evolution, dynamic systems, chaotic algorithms, machine learning and natural language processing techniques constitute a part of approaches that one can find in literature about machine improvisation. However, most of these machine improvisation systems, even with interesting sound results either in a pre-defined music style or in the form of free-style computer synthesis, did not address directly the part of interaction with human.

During the last years, achievements in Artificial Intelligence and new computer technologies have brought a new era in real-time computer improvisation music systems. Since real-time systems for music have provided the framework to host improvisation systems, new possibilities have emerged concerning expressiveness of computer improvisation, real-time control of improvisation parameters and interaction with human. These have resulted to novel improvisation systems that establish a more sophisticated communication between the machine and the instrument player. Some systems have gone even further in terms of interactivity by envisaging a small participation role for the computer user.

However, as for the interaction design of many of these systems, the role of the computer user in the overall interaction scheme often seems to be neglected, sometimes even ignored. What is the real degree of communication that machines achieve to establish with the instrument players in a real-world improvisation environment? Is the study of bipartite communication between instrument player - computer sufficient enough to model real-world complex improvisation schemes? More importantly, do modern computer improvisation system really manage to exploit the potential of this new form of interactivity, that is, between the computer and its user? And what would be the theoretical framework for such an approach that would permit a double form of interactivity between (1) computer and instrument player (2) computer and its user?

In our research we are concerned with the aspects of enhanced interactivity that can exist in the instrument *instrument player - computer - user* music improvisation framework. We are interested particularly on the computer - user communication channel and on the definition of a theoretical as well as of a the computational framework that can permit such a type of interaction to take place in real-time. Such a framework differs from the conventional approach

of other frameworks for machine improvisation in that the computer user takes an active role in the improvisation process. We call the context we study *Computer Assisted Improvisation (CAI)*, due to the shift of the subject role from the computer to the computer user.

Later in this report, we propose a model for three-party interaction in collective improvisation and present an architecture for Computer Assisted Improvisation. We then introduce a formalism based on graph theory in order to express concepts within computer assisted improvisation. This formalism succeeds in expressing unsupervised, content-dependant learning methods as well as the concept of "stylistic re-injection for the pattern regeneration, furthered enriched with new features for user control and expressivity. Through a bottom-up approach we analyze real-world sequence scheduling problems within CAI and study their resolvability in terms of space and time complexity. In this scope we present a number of real world music improvisation problems and show how these problems can be expressed and studied in terms of graph theory. Finally we give notions about our implementation framework GrAIPE (Graph assisted Interactive Performance Environment) in the real-time system Max-MSP, as well as about future research plans.

2. BACKGROUND

One of the first models for machine improvisation appeared in 1992 under the name *Flavors Band* [4]. Flavors Band a procedural language for specifying jazz and procedural music styles. Despite the off-line character of the approach, due to which one could easily classify the system to the computer-assisted composition domain, the advanced options for musical content variation and phrase generation combined with musically interesting results inspired later systems of machine improvisation. In the computer assisted improvisation context, the interaction paradigm proposed by Flavors Band could be regarded as a contribution of major importance, due to the fact that it assigns the computer user with the role of the musician who takes high-level offline improvisation decisions according to machine's proposals. Adopting a different philosophy as for the approach, GenJam [6] introduced the use of genetic algorithms for machine improvisation, thus being the 'father' of a whole family of improvisation systems in the cadre of evolutionary computer music. Such systems make use of evolutionary algorithms in order to create new solos, melodies and chord successions. It was also one of the first systems for music companion during performance. Evolutionary algorithms in non supervised methods were introduced by Papadopoulos and Wiggins in 1998 [9]. More recent learning methods include reccurent neural networks [5], reinforcement learning [11], or other learning techniques such as ATNs (Augmented Transition Networks) [7] and Variable order Markov Chains [12].

Thom with her BAND-OUT-OF-A-BOX(BOB) system [2] addresses the problem of real-time interactive improvisation between BOB and a human player. Sever years later, the LAM (Live Algorithms for Music) Network's manifesto underlines the importance of interactivity, under the term *autonomy* which should substitute the one of *automation*. In [14] the LAM manifesto authors describe the Swarm Music/Granulator systems, which implements a model of interactivity derived from the organization of social insects. They give the term 'reflex systems' to systems where 'incoming sound or data is analysed by software and a resultant reaction (e.g. a new sound event) is determined by pre-arranged processes.' They further claim that this kind of interaction is 'weakly interactive because there is only an illusion of integrated performer-machine interaction, feigned by the designer'. With their work, inspired by the animal interaction paradigm, they make an interesting approach to what has prevailed to mean 'human - computer interaction' and bring out the weak point inside real-time music systems' design. However, even if they provide the computer user with a supervising role for the improvisation of the computer, they don't give more evidence about how a three party interaction among real musician - machine - computer user could take place. Merely due to the fact that they consider their machine autonomous. But if the human factor is important for a performance, this should refer not only to the instrument player but to the person behind the computer as well. At this point, it seems necessary to study thoroughly the emerging three party interaction context where each of the participants has an independent and at the same time collaborative role.

Computer - computer user interaction is studied instead in the framework of live electronics and live coding, under the 'laptop-as-instrument' paradigm. In [15], the author describes this new form expression in computer music and the challenges of coding music on the fly within some established language for computer music or within a custom script language. The most widespread are probably the aforementioned SuperCollider [19], a Smalltalk derived language with C-like syntax, and most recently ChucK, a concurrent threading language specifically devised to enable on-the-fly programming [20]. Live coding is a completely new discipline of study, not only for music but also for computer science, phsycology and Human Computer Interaction as well. Thus, it seems -for the moment- that live coding is constrained by the expressivity limitations of the existing computer languages, and that it finds it difficult to generalize to a more complicated interaction paradigm which could also involve musicians with physical instruments.

2.1 OMax and stylistic reinjection

An interaction paradigm which is of major interest for our research is this of the stylistic reinjection, employed by the OMax system [3]. OMax is a real-time improvisation system which use on the fly *stylistic learning* methods in order to capture the playing style of the instrument player. Omax is an *unsupervised, context-dependant* performing environment, the last stating that it does not have pre-accuired knowledge but builds its knowledge network on the fly according to the special features of the player's performance. The capacity of OMax to adapt easily to different music styles without any preparation, together with its ability to treat audio directly as an input through the

employment of efficient pitch tracking algorithms, make OMax a very attractive environment for computer improvisation.

It is worth pointing out that style capturing is made with the help of the Factor Oracle Incremental Algorithm, intruduced by Allauzen and al. in [13], which repeatedly adds knowledge in an automaton called Factor Oracle (FO). Concerning the generation process, this is based on FO and is extensively described in [3]. With the help of forest of Suffix Link Trees (SLTs) it is possible to estimate a Reward Function in order to find interconnections between repeated patterns, which is the key for the pattern generation. Through this method, one can construct a model that continuously navigates within an FO. By balancing linear navigation with pivots and adding a little of nondeterminism to the Selection Function decisions, the system can navigate for ever by smoothly recombining existing patterns. Due to the short-term memory effect, this recombination is humanly perceived as a generation, which, more importantly, preserves the stylistic properties of the original sequence. The creators of the system call this method *stylistic reinjection*: this method relies on recollecting information from the past and re-injecting it to the future under a form which is consistent to the style of the original performance. Based on this principle and employing further heuristic and signal processing techniques to refine accordingly selections and sound rendering, OMax succeeds musically in a collective, real-world improvisation context.

Finally, OMax provides a role for the computer user as well. During a performance, user can change on the fly a number improvisation parameters, such as the proportion between linear navigation and pivot transitions, the quality of transitions according to common context and rhythm similarity, define the area of improvisation and cancel immediately the system's event scheduling tasks in order to access directly a particular event of the performance and reinitiate all navigation strategies.

2.2 The Continuator

An other system worth mentioning is The Continuator [12]. Based on variable-order Markov models, the system's purpose was to fill the gap between interactive music systems and music imitation systems. The system, handling except for pitch also polyphony and rhythm, provides content-dependent pattern generation in real-time. Tests with jazz players, as well with amateurs and children showed that it was a system of major importance for the instrument player - computer interaction scheme.

3. MOTIVATION

In this point, it is interesting to have a look over interaction aspects between the musician and the machine. Maintaining style properties assures that similar type of information travels in both directions throughout the musician - machine communication channel, which is a very important prerequisite for interaction. Moreover, diversity in musical language between machines and physical instrument playing, is one of the main problems encountered by evolutionary improvisation systems. OMax deals well with this inconvenience, in the sense that the patterns being regenerated and the articulation within time is based on the music material played the performer. However, when for interaction one should study not only information streams but also design of modules responsible for the *Interpretation* of received information [14]. In the case of the instrument player, human improvisors can interpret information according to their skills developed by practicing with the instrument, previous improvisational encounters and stylistic influences. These skills allow -or not- reacting to surprise events, for instance a sudden change of context. Concerning the machine, OMax disposes an interpretation scheme that stores information in the form FO representation. The question that arises concerns the capability of this scheme to handle surprise. This question can be generalized as follows: Can stylistic reinjection approach adapt its pre-declared generation strategies to short-time memory events? The last implies the need for an specifically configured autonomous agent capable of detecting short-time memory features of a collective improvisation, understanding their musical nuance and transmitting information to the central module of strategy generation.

Concerning stylistic reinjection, this approach currently permits a several amount of control of the overall process. We described in previous section the computer user's role in the OMax improvisation scenario. However, It seems intriguing to investigate further the role the computer user can have in such a scenario. For instance, wouldn't it be interesting if the computer user could decide himself the basic features of a pattern? Or if he could schedule a smooth passage in order that the computer arrives in a specific sound event within a certain time or exactly at a time?

We believe that the study of three-party interaction among can be beneficial for machine improvisation. In particular, we are interested in the 'forgotten' part of computer - computer user interaction, for which we are looking forward to establishing a continuous dialog interaction scheme. Our approach is inspired from the importance of the human factor in a collective performance between humans and machines, where humans are either implicitly (musicians) or explicitly (computer users) interacting with the machines. Though this three party interaction scheme, computer user is regarded as an equal participant to improvisation.

With respect to existing systems, our interest is to enhance the role of the computer user from the one of supervisor to the one of performer. In this scope, instead of controlling only low level parameters of computer 's improvisation, the user is set responsible of providing the computer with musical, expressive information concerning the structure and evolution of the improvisation. Inversely, the computer has a double role: first, the one of an augmented instrument, which understands the language of the performer and can regenerate patterns in a low level and articulate phrases coherent to user's expressive instructions, and second, the one of an independent improvisor, able to respond reflexively to specific changes of music context or to conduct an improvisation process with respect to a pre-

specified plan.

Our work consists of setting the computational framework that would permit such operations to take place. This includes:1) the study of the interaction scheme, 2) the architecture for such an interaction scheme 3) an universal representation of information among the different participants and 4) a formalism to express and study specific musical problems, their complexity, and algorithms for their solution.

4. ARCHITECTURE

In this section we analyze three-party interaction in CAI and propose a computational architecture for this interaction scheme.

4.1 Three party interaction scheme

In figure 1, one can see the basic concepts of three party interaction in CAI. The participants in this scheme are three: the musician, the machine (computer) and the performer. Communication among the three is achieved either directly, such as the one between the performer and the computer, or indirectly through the common improvisation sound field. The term sound field stands for the mixed sound product of of all improvisers together. During an improvisation session, both the musician and the performer receive a continuous stream of musical information, consisting of a melange of sounds coming from all sources and thrown in the shared sound field canvas. They manage to interpret incoming information through human perception mechanisms. This interpretation includes the separation of the sound streams and the construction of an abstract internal representation inside the human brain about the low and high level parameters of each musical flux as well as the dynamic features of the collective improvisation. During session, the musician is in a constant loop with the machine. He listens to its progressions and responds accordingly. The short-time memory human mechanisms provides her/him with the capacity to continuously adapt to the improvisation decisions of the machine and the evolution of the musical event as a group, as well as with the ability to react to a sudden change of context.

On the same time, the machine is listening to the musician and constructs a representation about what he has played. This is one of the standard principles for human machine improvisation. Furthermore, machine potentially adapts to a mid-term memory properties of musician's playing, thus reinjecting stylistically coherent patterns. During all these partial interaction schemes, the performer behind the computer, as a human participant, is also capable of receiving and analyzing mixed-source musical information, separating sources and observing the overall evolution.

The novelty of our architecture relies mainly on the concept behind performer - machine's communication. Instead of restricting the performer's role to a set of decisions to take, our approach aims to subject the performer in a permanent dialog with the computer. In other words, instead of taking decisions, the performer 'discusses' his intentions with the computer. First, he expresses his intentions to the system under the form of musical constraints. These constraints concern time, dynamics, articulation and other musical parameters and are set either statically or dynamically. As a response, the computer proposes certain solutions to the user, often after accomplishing complex calculi. The last evaluates the computer's response and either makes a decision or launches a new query to the machine. Then the machine has either to execute performer's decision or to respond to the new query. This procedure runs permanently and controls the improvisation. An other important concept in our architecture concerns the computer's understanding of the common improvisation sound field. This necessity arises from the fact that despite for computer's ability to 'learn', in a certain degree, the stylistic features of the musician's playing, the last does not stand for the understanding of the overall improvisation. Thus, There has to be instead a dedicate mechanism that assures interaction between the machine and the collective music improvisation. Moreover, such a mechanism can be beneficial for the performer - machine interaction as well, as it can make the computer more 'intelligent' in his dialog with the performer.

4.2 General architecture for Computer Assisted Improvisation

However, for the conception of an architecture for CAI that permits three party interaction, there are a couple of important issues to take into account. First, the fact that the proposed interaction scheme is gravely constrained in time, due to the fact that all dialogs and decisions are to be taken in real time (though in the soft sense). Second, the computer should be clever enough to provide *high-level, expressive* information to the performer about the improvisation, as well as high level decision making tools.

The internal architecture for the computer system is shown in figure 2. This architecture consist mainly of six modules which can act either concurrently or sequentially. On the far left we consider the musician, who feeds information to two modules: the pre-processing module and the short-term memory module. The pre-processing module is responsible for the symbolic encoding of audio information and stylistic learning. On the far right part of the figure we see the renderer, the unit that sends audio information to the collective sound field.

The short-memory processing module serves the understanding of the short-term memory features of collective improvisation. In order to reconstruct internally a complete image for the improvisation's momentum, this modules gathers information both from the representation module and the scheduler; the first in order to know what is being played by the musician and the second for monitoring computer's playing in short-term. It is possible that in the future it will be needed that the short-term memory processing module will also include an independent audio and pitch tracking pre-processor in order to reduce the portion of time required for the detection of surprise events.

In the low-center part of figure 2 one can find the interaction core module. This core consists of a part that is responsible for interfacing with the performer and a *solver* that responds to his questions. The performer lances queries

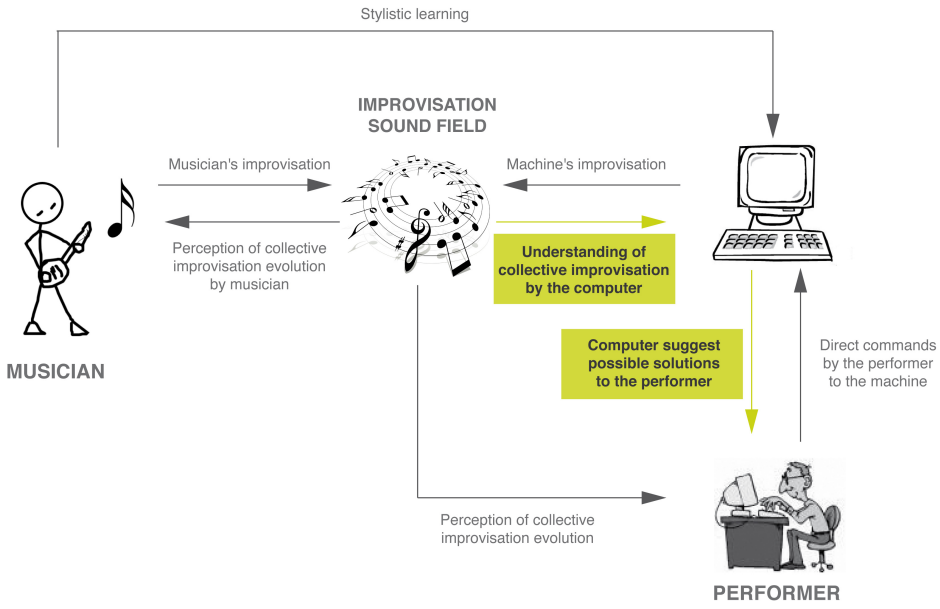

Figure 1. Three-party interaction scheme in Computer Assisted Improvisation. In frames (yellow) the new concepts introduced by the proposed architecture with respect to conventional interaction schemes for machine improvisation.

under the form of constraints. In order to respond, the solver attires information from the representation module. The time the performer takes a decision, information is transmitted to the scheduler. Scheduler is an intelligent module that accommodates commands arriving from different sources. For instance, a change-of-strategy command by the performer arrives via to the interaction core module to the scheduler.

The scheduler is responsible for examining what was supposed to schedule according to the previous strategy and organizes a smooth transition between the former and the current strategy. Sometimes, when contradictory decisions nest inside the scheduler, the last may commit a call to core's solver unit in order to take a final decision. It is worth mentioning that the dotted-line arrow from the short-term memory processing module towards the scheduler introduces the aspect of *reactivity* of the system in emergency situations: when the first detects a surprise event, instead of transmitting information via the representation module -and thus not make it accessible unless information reaches the interaction core-, it reflects information directly to the scheduler with the form of a 'scheduling alarm'. Hence, via this configuration, we leave open in our architecture the option that the system takes over *autonomous* action under certain criteria. The last, in combination with those mentioned before in this section establishes full three party interaction in a CAI context.

5. FORMALISMS FOR COMPUTER ASSISTED IMPROVISATION WITH THE HELP OF GRAPH THEORY

Further in this section, we address music sequence scheduling and sequence matching and alignment, two major problems for CAI. After a short introduction, we give formalisms for such problems with the help of graph theory.

5.1 Music sequence scheduling

Concerning the notion of *music scheduling* is usually found in literature as the problem of assigning music events to a particular time in the future, commonly within a real-time system [16]. Scheduling of musical events is one of the main functionalities in a real-time system [17] where the user is given the opportunity to plan the execution of a set of calculi or DSP events, in a relative or absolute manner, sporadically or sequentially. In a parallel study of the process of scheduling in music computing and other domains of research such as computer science and production planning, we could reason that for the general case described above, musical scheduling refers to the single machine scheduling and not the job-shop case.

We define *Music Sequence Scheduling* as the special task of building a sequence of musical tasks and assigning their execution to consecutive temporal moments.

In our study, we are interested to music sequence scheduling in order to reconstruct new musical phrases based on symbolically represented music material. Our objective is to conceptualize methods which will help the user define some key elements for these phrases, as well as a set of general or specific rules, and to leave the building and scheduling of the musical phrase to the computer. In an improvisation context, the last allows the performer taking crucial decisions in a high level; on the same time, the computer takes into account performer's intentions, sets up the low-level details of the phrase generation coherently to the user choices and outputs the relevant musical stream. For instance, a simple sequence scheduling prob-

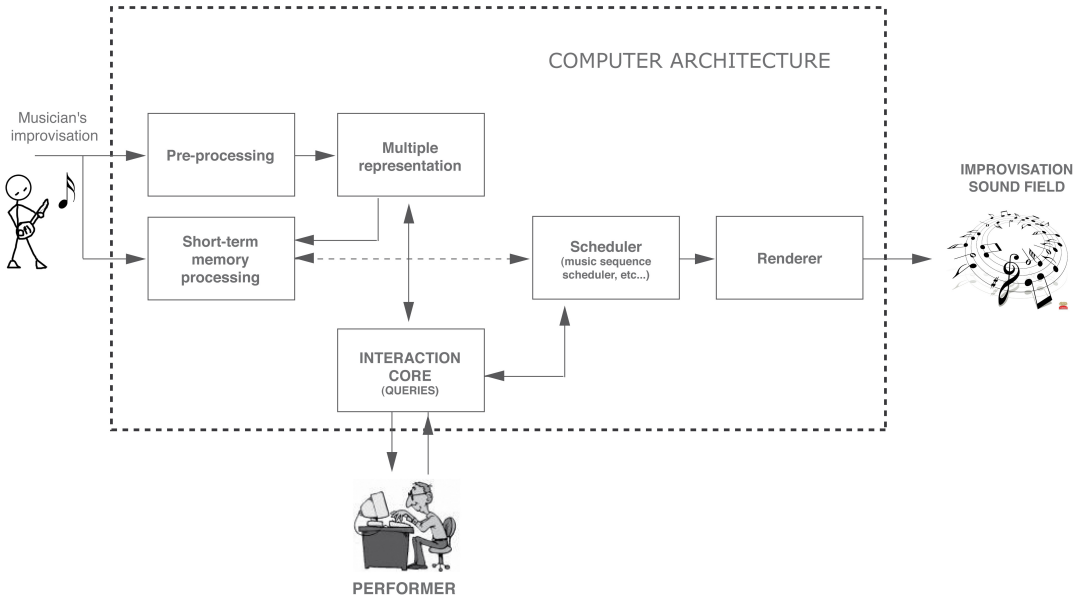

Figure 2. Overal Computer Architecture for CAI

lem would consist of navigating throughout a FO with the same heuristic such as the one used by OMax system, under the additional constraint that we would like to reach a particular state x in a particular moment t.

5.2 Music sequence matching and alignement

Music sequence matching concerns the capacity of the system to recognize if sequence is within its known material. Music sequence alignment is the problem of finding sequences within its corpus which are 'close' to a sequence of reference. The last pre-assumes the definition of a distance function according to certain criteria. Both are not in the scope of this paper, but are mentioned for reasons of clarity.

5.3 Formalisms

The structures used for learning in existing improvisation systems, even if effective for navigation under certain heuristics, are too specialized to express complex problem of navigation under constraints. For instance, a FO automaton is sufficient when for agnostic navigation under certain heuristics among its states, but fails to answer to problems of scheduling a specific path in time. In order to be able to express diverge problems of music sequence scheduling, alignment or more complex problems, we are obliged to use more general graph structures than the ones used in content-dependent improvisation systems. The advantage of this approach is that our research then can be then generalized to include other graph-like structures. Our method focuses on regenerating material by navigating throughout graph structures representing the corpus. Due to the reasons mentioned before we will express all stylistic reinjection related problems under the graph theory formalism.

Formally we describe the process of music sequence scheduling in the context of stylistic reinjection as follows:

Consider now a continuous sound sequence S. Suppose that for this sequence it is possible to use an adaptive segmentation method to segment S in n chunks according to content and a metric $m = f(d)$, $d \in D$, where D a set of descriptors for S. Each m causes different segmentation properties i.e different time analysis t_m. A symbolic representation of S would then be $S_m(t_m)$, $1 \leq m \leq M$, where M the number of metrics and $t_m = 0, 1, .., n_m \; \forall m \in M$.

Axiom 1 The musical style of a continuous sound sequence S can be represented by a countably infinite set P with $|P| \neq 0$ of connected graphs.

Definition 1

We define *stylistic learning* as a countably infinite set $F = \{Sl_1, Sl_2, .., Sl_n\}$, $|F| \neq 0$ of mapping functions $Sl_i : \mathbf{S}_m \rightarrow G_i(V_i, E_i)$, where $\mathbf{S}_m = S_{mt_m} \forall t_m \in [0, n_m]$ of sequence S for a metric m, $1 \leq m \leq M$, G_i a connected graph with a finite set of vertices V_i and edges E_i as a binary relation on V_i.

Definition 2 We define as *stylistic representation* the countably infinite set $P = \{G_i(V_i, E_i) : i \neq 0\}$ of digraphs.

Definition 3 We define as *sequence reinjection* a selection function
$R_{seq} : (G_i, q) \rightarrow \mathbf{S}_m$ with $R_{seq}(G_i, q) = \mathbf{S'_m}$ and $S'_{mt_m} = S_{mt'_m}$, q a number of constraints $q = h(m)$, $m \in (1, M)$.

Definition 4 We define as *musical sequence scheduling* as a scheduling function $R_{sch} : (R_{seq}, T_s(R_{seq})) \rightarrow \mathbf{S}_m$, with T_s the setup time for the sequence reinjection R_{seq}.

With these formalisms we can now begin to study stylistic representation, learning and sequence reinjection with the help of graphs. These issues now reduce to problems of constructing graphs, refining arc weights and navigating along the graphs under certain constraints.

In section 4.2 we underlined the importance of the short-term memory processing module. Even while the standard functionality, it should be employed with the mechanism to quickly decode information that has lately been added

to the representation, make comparisons with earlier added performance events and find similarities.

Definition 5 We define *Music Sequence Matching (MSM)* as a matching function $M : (\mathbf{S}_m, \mathbf{S'}_m) \rightarrow [0,1]$, where $\mathbf{S}_m$, $\mathbf{S'}_m$ symbolic representations of sequences S and S' for the same metric m.

Definition 6 We define *Music Sequence Alignment (MSA)* the alignment function $A : (\mathbf{S}_m, \mathbf{S'}_m, q) \rightarrow R$ where

$$A(\mathbf{S}_m, \mathbf{S'}_m, q) = \min\{\sum_q c_q x_q\}, \quad (1)$$

$\mathbf{S}_m, \mathbf{S'}_m$ symbolic representations of sequences S and S' respectively for the same metric m, q a number of string operations, $c_q \in \mathbb{R}$ a coefficient for an operation q and $x_q \in \mathbb{Z}$ the number of occurrences of operation q.

With the help of the previous definitions we are ready to cope with specific musical problems.

6. A SIMPLE PROBLEM ON STYLISTIC REINJECTION IN CAI

Problem

Let a musical sequence S symbolically represented as S_n, $n \in [0,N]$, and $s,t \in [0,N]$ a starting and target point somewhere within this sequence. Starting from point s, navigate the quickest possible until the target t, while respecting sequence's stylistic properties.

Definition 7 With the help of axiom 1 and definitions 1, 2, we apply stylistic learning and create a stylistic representation digraph $G(V,E)$ for S_n with set of vertices $V = \{0,1,..N\}$ and E the set of edges of G.

We define as *Music Transition Graph (MTG)* the digraph G with the additional following properties:

1. G is connected with no self loops and $|V|-1 \leq |E| \leq (|V|-1)^2 + |V| - 1$
2. every $e_{i,j} \in E$ represents a possible transition from vertex i to j during navigation with cost function $w_\tau(i,j)$
3. for every $i \in [0,N-1]$ there is at least one edge leaving vertex i $e_{i,i+1}$.
4. let d(i) the duration of a musical event $S_i \in \mathbf{S}$, $d_0 = 0$. The cost function of an edge $e_{i,j} \in E$ is defined as:
   - $w_\tau(i,j) = d(j)$, if $j = i+1$, $\forall i,j \in (0,N)$
   - $w_\tau(i,j) = 0$, if $j \neq i+1$, $\forall i,j \in [0,N]$
   - $w_\tau(0,1) = 0$.

Solution to problem

Let a path $p =< i_0, i_1, .., i_k >$ in a graph G' and the weight of path p the sum of the weights of its constituent edges:

$$w(p) = \sum_{j=1}^{k} w(i_{j-1}, i_j). \quad (2)$$

We define the shortest-path weight from s to t by:

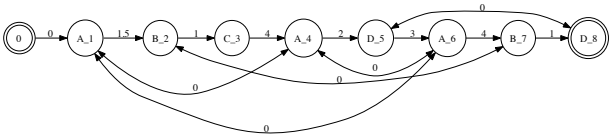

Figure 3. A Music Transition Graph for melody ABCADABD. Arc weights correspond to the note duration of the source. Bidirectional arcs above and below nodes connect patterns with common context.

$$\delta(s,t) = min\{w(p) : s \rightsquigarrow^p t\}. \quad (3)$$

Let a MTG G to stylistically represent the sequence S_n. This graph is connected, with no self loops and no negative weights. Given that, from definition 7, $w_\tau(i,j)$ is strictly by the duration $d(i,j)$ of graph events, our problem consists of finding a MTG's shortest path problem. The solution for the MTG shortest path exists and can be found in polynomial time. One of the most known solutions to this problem can be found with the help of Dijkstra's algorithm, with complexity $O(\log V)$. Dijkstra's algorithm does not solve the single pair shortest-path problem direcly, it solves instead the single-source shortest path problem, however no algorithms for the single pair shortest path problem are known that run asymptotically faster than the best single-source algorithms in the worst case.

Corollary from solution

Given a music sequence, the MTG representation permits solving the problem of accessing from a musical event s a musical event t within a music sequence S in the less possible time. Thus, by recombining musical events within S, we can reproduce a novel sequence from s to t. On the same time, this sequence respects the network of the MTG graph, hence the stylistic properties of the original music sequence.

Application 1

Suppose a music melody $S\{A,B,C,A,D,A,B,D\}$, with durations $d_S\{1.5,1,4,2,3,4,1,2.5\}$.

We construct a MTG $G(V,E)$ with edges $e(i,j) \in E$ with for $e(i,j) : j \neq i+1$ the arc connect common context according to the metric $m = PITCH$ (figure 3).

Suppose that our problem is to find the quickest possible transition from vertex $s = 2$ (note B) to vertex $t = 8$ (note D). To solve our problem, we can apply Dijkstra's algorithm.

We apply the algorithm for our sequence S and state D. When the algorithm terminates we have the shortest paths for all graph vertices. We can resume that for our problem the solution is the path $p =< B_2, B_7, D_8 >$. For the navigation along a path, we ignore one of two interconnected components. Hence, the final path is $p' =< B_2, D_8 >$.

We presented the formalisms and the solution to the simplest sequence scheduling problem. In our research, we focus on a number of problems that we are treating with the same methodology. These problems are combination of problems in sequence scheduling and sequence alignment domain. A list of the more important ones that we are dealing with is as follows:

1)Find the shortest path in time from a state s to a state t (examined).

2) The same with problem 1 with the constraint on the length of common context during recombinations.

3) Find the shortest path in time from a state s to a given sequence .

4) Find the continuation relatively to given sequence (Continuator).

5)The same with problem 1 with the additional constraint on the total number of recombinations.

6)Find a path from a state s to a state t with a given duration t, with $t_1 \leq t \leq t_2$.

7)Find a path from a state s to a state t with a given duration t, with $t_1 \leq t \leq t_2$ and with the additional constraint on the total number of recombinations (problem 5 + 6)

7. GRAIPE FOR COMPUTER ASSISTED IMPROVISATION

Our algorithms and architecture, are integrated in an under development software under the name GrAIPE. GrAIPE stands for Graph Assisted Interactive Performance Environment. It is an ensemble of modular objects for max-msp implementing the architecture presented in section 4.2. Concerning GrAIPE's design and implementation, our basic priorities are intelligent interfacing to the performer and efficient well-implemented algorithms for concurrency, interaction and the system's core main functions. Whether the software is under development, an instantiation of GrAIL has already taken place under the name PolyMax for machine improvisation, simulating with success 10 concurrent omax-like improvisers.

8. CONCLUSIONS - FUTURE RESEARCH

In this report we tried to set the basis for a novel, three-party interaction scheme and proposed a corresponding architecture for Computer Assisted Improvisation. Employing the approach of stylistic learning and stylistic interaction, we operated to formalize this interaction scheme under formalisms inspired from graph theory. We then discussed a simple music sequence scheduling problem.

Graph approach to CAI appears to be promising for modeling three-party interaction in a real-time non supervised improvisation environment that includes a 'silicon' participant. Not only does it permit a formalization of CAI problems in relation with space and time complexity, but it also approaches timing and capacity issues with widely accepted time-space units (for instance, milliseconds) that can make explicit the connection of our theoretical results with real-time system own formalism. This can be proved extremely practical for the future when integrating our theoretical results to real-time environment, in contrast with other formalisms such as in [18] that despite of penetrating complex interactivity issues, their temporal analysis in abstract time units makes this connection more implicit. Furthermore, graph formalization allows transversal bibliography research in all domains where graphs have been employed (production scheduling, routing and QoS etc.), and thus permit the generalization of music problems to universal problems and their confrontation with algorithms that have been studied in this vast both theoretic and applicative domain of graph theory.

In the recent future we are focusing on presenting formal solutions for the problems list in the previous section. Of particular interest are approximate solutions to problems 5, 6, 7, which are NP-hard for the general case.

Concerning development, GrAIPE has still a lot way to run until it fits with the requirements set in section 4.2. Even though already with a scheduler, a basic visualization module and a scripting module, these modules are being re-designed to adapt to the new research challenges. Other modules are a constraint-based user interface and its communication with a solver which is under development.

9. REFERENCES

[1] De Mantaras, R., Arcos, J., "AI and Music. From Composition to Expressive Performance" , AI Magazine, Vol. 23 No.3, 2002.

[2] Thom, B., "Articial Intelligence and Real-Time Interactive Improvisation", Proceedings from the AAAI-2000 Music and AI Workshop, AAAI Press, 2000.

[3] Assayag, G. , Bloch, G., "Navigating the Oracle: a Heuristic Approach", Proc. ICMC'07, The In. Comp. Music Association, Copenhagen 2007.

[4] Fry, C., "FLAVORS BAND: A language for Specifying Musical Style", Machine Models of Music, p. 427-451, Cambridge, MIT Press, 1993.

[5] Franklin, J. A. Multi-Phase Learning for Jazz Improvisation and Interaction. Paper presented at the Eighth Biennial Symposium on Art and Technology, 13 March, New London, Connecticut, 2001.

[6] Biles, A. GENJAM: A Genetic Algorithm for Generating Jazz Solos. In Proceedings of the 1994 International Computer Music Conference, 131137. San Francisco, Calif.: International Computer Music Association, J. A. 1994.

[7] Miranda, E., "Brain-Computer music interface for composition and performance", in International Journal on Disability and Human Development, 5(2):119-125, 2006.

[8] Graves, S.,"A Review of Production Scheduling", Operations Research, Vol. 29, No. 4, Operations Management (Jul. - Aug., 1981), pp. 646-675, 1981.

[9] Papadopoulos, G., and Wiggins, G. 1998. "A Genetic Algorithm for the Generation of Jazz Melodies". Paper presented at the Finnish Conference on Artificial Intelligence (SteP98), 79 September, Jyvaskyla, Finland, 1998.

[10] Giffler, B., Thompson, L., "Algorithms for Solving Production-Scheduling Problems", Operations Research, Vol. 8, No. 4, pp. 487-503, 1960.

[11] M., Cont, A., Dubnov, S.,Assayag, G., "Anticipatory Model of Musical Style Imitation Using Collaborative and Competitive Reinforcement Learning", Lecture Notes in Computer Science,2007.

[12] Pachet,F., "The continuator: Musical interaction with style". In proceedings of International Computer music Conference, Gotheborg (Sweden), ICMA,2002.

[13] Allauzen C., Crochemore M., Raffinot M., "Factor oracle: a new structure for pattern matching, Proceedings of SOFSEM'99, Theory and Practice of Informatics, J. Pavelka, G. Tel and M. Bartosek ed., Milovy, Lecture Notes in Computer Science 1725, pp 291-306, Berlin, 1999.

[14] Blackwell, T., Young, M. "Self-Organised Music". In Organised Sound 9(2): 123136, 2004.

[15] Collins, N., McLean, A., Rohrhuber, J., Ward, A., "Live coding in laptop performance", Organized Sound, 8:3:321-330, 2003.

[16] Dannenberg, R., "Real-TIme Scheduling And Computer Accompaniment Current Directions in Computer Music Research", Cambridge, MA: MIT Press, 1989.

[17] Puckette, M., "Combining Event and Signal Processing in the MAX Graphical Programming Environment", Computer Music Journal, Vol. 15, No. 3, pp. 68-77, MIT Press, 1991.

[18] Rueda, C., Valencia, F., "A Temporal concurrent constraint calculus as an audi processing framework", Sound and Music Computing Conference, 2005.

[19] http://www.audiosynth.com/

[20] http://chuck.cs.princeton.edu/