Fivos Maniatakos

8 Designing Three-Party Interactions for Music Improvisation Systems

Abstract: For years, a major challenge among artists and scientists has been the construction of music systems capable of interacting with humans on stage. Such systems find applicability in contexts within which they are required to act both as independent, improvising agents and as instruments in the hands of a musician. This is widely known as the player and the instrument paradigm. During the last years, research on Machine Improvisation has made important steps towards intelligent, efficient and musically sensible systems that in many cases can establish an interesting dialog with a human instrument player. Most of these systems require, or at the very least encourage, the active participation not only of instrument players but also of computer users-operators that conduct actions in a certain way. Still, very little has been done towards a more sophisticated interaction model that would include not only the instrument player and the machine but also the computer operator, who in this case should be considered as a computer performer. In this paper we are concerned with those aspects of enhanced interactivity that can exist in a collective improvisation context. This is a context characterized by the confluent relationship between instrument players, computers, as well as humans that perform with the help of the computer, onstage. The paper focuses on the definition of a theoretical as well as a computational framework for the design of modern machine improvisation systems that will leave the necessary space for such parallel interactions to occur in real-time. We will study the so-called three party interaction scheme based on three concepts: first, with the help of the computer, provide an active role for the human participant, either as an instrument player or as a performer. Secondly, create the framework that will allow the computer being utilized as an augmented, high-level instrument. Lastly, conceive methods that will allow the computer to play an independent role as an improviser-agent who exhibits human music skills.

Keywords: Machine Improvisation, Computer Aided Improvisation, Three Party Interaction, Automata

8.1 Introduction

A major aspect of the computer’s contribution to music is reflected in interactive systems. These are computer systems meant to perform together with human performers on stage. It seems that there are at least two a priori reasons for such systems to exist and they are both straightforward to understand: sound synthesis and algorithmic processing. By expanding the acoustic instruments with a supplementary
layer, interactive systems added a completely new dimension to the way music is generated and perceived onstage. On the other hand, the computers efficiency in executing complex algorithms, otherwise impossible to test manually, has allowed the introduction of computational paradigms into the world of composition and improvisation. These two concepts are known as the instrument and the player paradigm respectively. With musical demands increasing over the years, interactive systems have come on age, only to be constrained by the limitations imposed by both the theoretical background and the technically plausible.

Machine musicianship (see (Rowe, 1992)) is regarded as the key to the success of interactive systems. The first reason for this is straightforward: due to their apparent association with human skills, computer skills have to be advanced enough so as to substantiate the computer’s raison d’être in the music field. The second reason is perhaps the most important: making the computer more musically intelligent can help people engage with music. Consequently, through a human-oriented approach, machine musicianship can be seen as a means of improving audiation, which, according Gordon in (Gordon, 1980), is “an individual’s ability to communicate through music”. Machover (Oteri, 1999) has wonderfully expressed the computer’s role in enhancing musical sociability: the deployment of computers as a way of reducing practical inconveniences and of increasing the level of communication that can be achieved through music, combined with a vision of engaging more people in music regardless of their dexterity level, makes it nothing less that appealing. In (Rowe, 2001), Rowe expresses a similar thought: “... Interactive systems require the participation of humans making music to work. If interactive music systems are sufficiently engaging as partners, they may encourage people to make music at whatever level they can.”

These ideas are also discernible in the history of machine improvisation systems. In the beginning, research seemed to focus more on the development of autonomous, self-contained virtual improvisers and composers (Mantaras & Arcos, 2002). Algorithmic composition of melodies, adaptation to harmonic background, stochastic processes, genetic co-evolution, dynamic systems, chaotic algorithms, machine learning and natural language processing techniques were a few of the approaches that were followed and that one can still find in 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 adequately refer to the part of interaction with humans.

Nowadays, the incorporation of human participants in the process is almost non-negotiable. During the last years, research on Machine Improvisation has made important steps towards intelligent, efficient and musically sensible systems, based on the instrument or the player paradigm, or both. In many cases, such improvisation systems manage to establish an interesting dialog with a human instrument player and, in many cases, encourage the active participation of a computer user that conducts actions in a certain manner.
Still, very little has been done towards a more sophisticated interaction model that would include not only the instrument player and the machine but also the machine operator (computer user or performer). Do modern computer improvisation systems really manage to utilize the potential of this new form of interactivity, that is, between the computer and its user? And what would be the theoretical framework of such an approach that would permit a double form of interactivity between (a) the computer and instrument player (b) the computer and the performer?

In this work we are concerned with those aspects of enhanced interactivity that can exist in a modern music improvisation context. This is a context characterized by the presence of instrument players, computers, as well as humans that perform with the help of the computer. It is therefore the focus of this paper to define a theoretical as well as a computational framework that will leave the necessary space for such interactions to occur in real-time. In the next sections we will study the so-called three party interaction scheme and the architecture of GrAIPE (Graph Aided Interactive Performance Environment) improvisation system that is based on three basic concepts: first, with the help of the computer, provide an active role for the human participant, either as an instrument player or as a performer. Secondly, create the framework that will allow the computer to be utilized as an augmented, high-level instrument. Lastly, conceive methods that will allow the computer to play an independent role as an improviser-agent who exhibits human music skills.

8.2 Interactive Improvisation Systems

8.2.1 Compositional Trends and Early Years

The exploration of random processes for music composition has been of great interest to many composers of the last century. Cage is one of the first composers to underline, through his compositional work, the importance not so much of the resulting sound but of the process necessary for its creation. It was as early as 1939 with his work Imaginary Landscape that he detached himself from the musique concrete paradigm of sound transformation in controlled environments, thus stressing the importance of both the creation process and the randomness that occurs in less controlled performance contexts. It is precisely this trend that we come across in his later works, such as William’s Mix (1953) and Fontana Mix (1958) as well as in studio works of other composers of the time, such as Reich’s Come Out (1966) and Eno’s Discrete Music (1975) and in live performance projects by both Cage and Tudor, namely Rainforest and Variations V (see (Eigenfeldt, 2007)) for more details on the above compositional works).

While algorithmic composition has placed the composition process at the centre of attention, it was the “complexification” of this process that was indented to become
the basic constituent of what was identified as interactive composition. Interactive composition focused on how to make the composition process complex enough so as to assure the unpredictability of the result. This paradigm, first introduced by Joel Chadabe (see (Chadabe, 1984)), grounded the computer as an indispensable component of the composition process and as the medium to achieve “creative” randomness through algorithmic complexity. The basic idea found behind interactive composition is the one of a sophisticated “instrument” whose output exceeds the one-to-one relationship with the human input by establishing a mutually influential relationship between the performer and itself. The partial unpredictability of information generated by the system, combined with the partial control of the process from the part of the performer, creates a mutually influential relationship between the two, thus, placing both into an interactive loop.

8.2.2 Interacting Live With the Computer

Interactive composition has signalled the beginning of a new era in computer music systems, providing a new paradigm for human computer interaction in music composition and performance.

The Voyager interactive musical environment is a personal project by trombonist and composer George Lewis (Lewis, 2000). The software, first written in Forth language at the early 1980s, was intended as a companion for onstage trombone improvisation, by guiding automatic composition through real-time analysis of the various aspects of the trombone players performance. According to Lewis, Voyager follows the player paradigm in Rowe’s taxonomy, in the sense that it is designed to generate complex responses through internal processes thus humanizing its behaviour.

Rowe’s Cypher system (1990) reflected the motivation to formalize concepts on machine musicianship for the player paradigm (see (Rowe, 1992, 2001, 1999)). This system can be regarded as a general set of methods that cope with four basic areas of machine musicianship: analysis, pattern processing, learning, and knowledge representation. Rowe’s ideas have had major influence on machine improvisation systems of the last decade. A typical example of such systems are those employing Lexicography-based methods, such as suffix trees, while there are also other, non pattern-based systems that may produce similar results. Thom with her BAND-OUT-OF-A-BOX (BOB) system (Thom, 2000) addresses the problem of real-time interactive improvisation between BOB and a human player. In other words, BOB is a “music companion for real-time improvisation. Thom proposes a stochastic model based on a greedy search within a constrained space of possible notes to be played. Her system learns these constraints -hence the stochastic model- from the human player by means of an unsupervised probabilistic clustering algorithm. Its approach is more performer than designer based, as it adapts to the playing of the instrument player. Furthermore, the system has a high degree of autonomy, using exclusively non-supervised methods.
An interaction paradigm which is of major interest is the one of “stylistic reinjection”, employed by the OMax system (Assayag & Bloch, 2007; Assayag, Bloch, Chemillier, Cont, & Dubnov, 2006). OMax is a real-time improvisation system which uses on the fly stylistic learning methods in order to capture the playing style of the instrument player. OMax learning mechanism is based on the Factor Oracle (FO) automaton (introduced in (Allauzen, Crochemore, & Raffinot, 1999)). OMax provides a role for the computer user as well, in the sense that during a performance, the user can change on the fly a number improvisation parameters. 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 it an attractive environment for computer improvisation.

Another system worth mentioning is The Continuator (Pachet, 2002). Based on suffix trees, the system’s purpose was to fill the gap between interactive music systems and music imitation systems. Given a mechanism that assures repositioning from a leaf to somewhere inside the tree, generation can be seen as the concatenated outcome of edge labels, called “continuations”, emitted by the iterated navigation down the branches of the tree. Tests with jazz players, as well as with amateurs and children confirmed the importance of the system for the instrument player – computer interaction scheme. In (Pachet & Roy, 2011) the authors exploit the potential of integrating general constraints into Markov chains. Moving towards this direction, they reformulate the Markov probability matrix together with constraint satisfaction practices so as to generate steerable Markovian sequences.

Schwarz’s CataRT project is an efficient, continuously evolving real-time concatenative sound synthesis system, initially conceived in the cadre of the instrument paradigm. After five years of continuous development, CataRT is now capable of dealing with the problem of finding continuations. It has also recently started challenging the CSS’s potential to function as an improvisation companion in the cadre of the player paradigm (Schwarz & Johnson, 2011).

8.3 Three Party Interaction in Improvisation

We use the term three-party interaction to refer to the ensemble of interactions arising from a modern, collective improvisation context. In Figure 8.1, one can see the basic concepts of the three party interaction scheme. The participants in this scheme are: the musician, the machine (computer) and the computer performer. Communication among the three is achieved either directly, such as in the case between the performer and the computer, or indirectly through the common improvisation sound field. During an improvisation session, both musician and performer receive a continuous stream of musical information, consisting of a melange of sounds coming from all sources and thrown into the shared sound field canvas. Incoming information is then interpreted through human perceptual mechanisms. This interpretation involves
the separation of the sound streams and the construction of an abstract internal representation inside the human brain on the low and high level parameters of each musical flux, as well as on the dynamic features of the collective improvisation. During a session, the musician is in a constant loop with the machine. She/he listens to its progressions and responds accordingly. The human, short-time memory mechanisms make it able for her/him to continuously adapt to the improvisation decisions of the machine and the evolution of the musical event as a group, as well as to react to a sudden change of context.

At the same time, the machine is listening to the musician and constructs a representation on what she/he has played. This is one of the standard principles for human machine improvisation. Furthermore, the machine potentially adapts to the mid-term memory properties of the musician’s playing, thus re-injecting 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 concepts hidden behind the communication between performer and machine. Instead of restricting the performer’s role to a set of decision making actions, our approach aims at putting the performer in a position where she/he is in a constant dialog with the computer. This means, instead of taking decisions, the performer ‘discusses’ his intentions with the computer; the computer, in its turn, makes complex calculi and proposes certain solutions to the performer. The latter evaluates the computer’s response and either makes a decision or lances a new query to the machine. Conversely, the machine either executes the performer’s decision or responds to the new query. This procedure runs continuously and controls the improvisation.

![Figure 8.1: Three-Party Interaction Scheme for collective improvisation. In yellow the interactions held according to the human and the instrument paradigm](image-url)
The second concept concerns the computer’s understanding of improvisation. This necessity derives from the fact that even though the computer is able to learn, at least to a certain extent, the stylistic features of the musicians playing, still the understanding of the overall improvisation process remains beyond its grasp. Therefore, there has to be a dedicated mechanism that will assure the interaction between the machine and each of the participants of the improvisation. Moreover, such a mechanism can prove to be beneficial for the performer – machine interaction as well, as it can make the computer more ‘intelligent’ in its dialog with the performer.

As mentioned in Section 1, building frameworks with the intelligence to tackle simultaneously all issues arising from three party interaction is not straightforward. Most systems are based on just one method and, although they adapt well to a restricted subset of requirements, they usually fail to provide a complete framework for dealing with the musicianship of improvisation agents. For instance, a system may show increased accompaniment and solo improvisation skills for a particular music style, whilst at the same time fail to adapt to other musical styles.

On the other hand, hybrid systems that use different modules in parallel - each based on a different approach- failed to provide clear answers to the problem. Tackling a variety of problems demands heteroclite actions arising from distinct methods and representations. Dispersing resources to numerous, distinct strategies simultaneously, increases the computational cost. As the whole system has to make everything work together in real-time, the computational cost may be prohibitive. One way towards a universal framework for machine improvisation agents is to find common elements among the different approaches and bring them together. Nevertheless, the main reason for the difficulty for these heuristics to work together and cooperate for achieving solutions to composite problems, is the lack of a universal, music-dedicated representation. Such a representation would allow all different methods to be regrouped and work in reduced memory cost.

Thus, the technical requirements of this representation framework should be the following:

- It should provide tools for sequence generation coherent to the stylistic features of the improvisation as established by the ensemble of the participants,
- It should entail mechanisms to detect modes of playing of each of the participants during improvisation, to detect the short-term, expressive features and be capable of responding accordingly,
- It should have mechanisms allowing human-driven, stylistically coherent sequence generation, in order to interact with the performer,
- It should be computationally efficient, fully customizable and expandable with respect to the different types of interaction that it can establish either with the instrument player or with the human operator.
8.4 The GrAIPE Improvisation System

GrAIPE is a software built to support the previously mentioned requirements for improvisation in the three party interaction context. “GrAIPE” stands for Graph Assisted Interactive Performance Environment. GrAIPE is a data-driven system, in the sense that it does not need any a priori knowledge to improvise, but learns directly from the content. Its mechanism is based on the simple principle of detecting and representing similarities of musical motifs inside the sequences, fed to the system during the learning process. With the help of this representation, the system is then capable of detecting musically meaningful continuations between non-contiguous motifs inside the sequence. When in improvisation mode, the system navigates inside this representation either linearly or non-linearly. In the first case, it reproduces exactly a part of the original sequence, whereas in the second, it jumps (reinjects) to a distant point of the sequence, thus producing a musical motif that is musically interesting even though it does not exist in the original sequence. By balancing these two navigation modes, GrAIPE manages to produce novel music material by meaningfully recombing motifs of the original sequence.

What is new about GrAIPE in comparison with conventional improvisation systems is the mechanism for controlling interaction in improvisation. This mechanism is based on the representation structure that lies beneath GrAIPE, the Music Multi Factor Graph (MMFG) introduced in (Maniatakos, 2012). The MMFG is a graph-like structure for representing musical content. It contains a number of nodes equal to the number of successive musical events that occur in the given sequence, as well as weights that determine possible reinjections.

![Figure 8.2: Top: Music sequence: S = \{Do, Mi, Fa, Do, Re, Mi, Fa, Sol, Fa, Do\}. Bottom: MFG-Cl for sequence S. There is one-to-one correspondence between the notes and the states of the MFG (each note event is exactly one state). Arrows in black and grey color represent transitions (factors) and suffix links respectively.](image)

The key behind MMFG’s novelty is its double identity, being simultaneously a Repeat Non-Linear (RNL) graph and a Multi Factor Graph (MFG) automaton (for both see...
Due to its identity as a RNL graph, it manages to exhaustively detect all possible continuations inside a learned sequence, sort them according to their musical quality and schedule reinjections through time. By being a MFG automaton, MMFG provides the tools for finding a given pattern inside a sequence with minimal computational cost. Furthermore, as shown in (Maniatakos, 2012), MMFG is the result of merging the two structures together by taking advantage of their common properties as structures. Building MMFG is accomplished at almost no additional memory cost than the one needed for RNL and, in any case, far less than building the graph and the automaton separately.

With the help of MMFG, it is possible to resolve not only problems for pattern matching and sequence scheduling but also composite problems that arise when combining the two categories of problems together. GrAIPE allows the user to declare a number of constraints relating to the improvisation and, by employing a number of algorithms called “primitives”, proposes improvisation scenarios compliant with the user’s constraints.

The table of Figure 8.3 illustrates the steps of the scheduling process for music sequence generation with the help of MMFG’s primitives. MMFG provides a variety of strategies for declaring the source, the destination and the generation parameters. Concerning the designation of the source, this can be done through pure temporal parameters, such as in absolute terms or at a specified time interval. Alternatively, it can also be done by declaring a source state locally, or by satisfying the condition that a specific pattern (chroma, rhythmic, or timbral) is recognized. In a similar manner we can declare the target of the scheduled sequence, with the help of explicit (state) or implicit local parameters (for example, at a certain motif inside the sequence). A number of different configurations can be performed to specify the temporal parameters connected to the target. For instance, one can ask the system to go to a specific target “immediately” (by forcing the transition to the target state), to transit smoothly towards a specified target “as quickly as possible”, or in terms of absolute time or in a time interval (i.e improvise smoothly and arrive to a target t within 5 seconds). Finally, there can also be many other possible parameters in order to specify the characteristics of the generation process. These parameters rely on filtering transitions from/to a state according to quality parameters, or by applying quality constraints to the entire path. In the table of Figure 8.3 there are also methods for balancing the tradeoffs of stylistic coherency vs. novelty and looping vs. originality.

This parameterization of the scheduling process with the help of MMFG provides a framework of large-scale control over the generation process. The great advantage of this approach is the fact that, once the scheduling parameters are defined, generation can then be done automatically without any additional intervention from the part of the user. By declaring long-term goals and leaving the note-level generation -according to a set of rules- to the computer, it is possible to achieve precise control on the temporal aspects of generation. This facilitates generation in the improvisation
context, where precise temporal control is indispensable, but not straightforward to achieve through conventional methods.

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>How</th>
<th>domain constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>at state $s$</td>
<td>at moment $T$</td>
<td>immediately bounded below</td>
<td></td>
</tr>
<tr>
<td>at the actual state</td>
<td>in $\Delta T$ time</td>
<td>as quickly as possible</td>
<td>bounded above</td>
</tr>
<tr>
<td>at pattern $p$</td>
<td>at rhythmic pattern $q_r$</td>
<td>at second $T$</td>
<td>double bounded</td>
</tr>
<tr>
<td>at rhythmic pattern $p_r$</td>
<td>approximately $\Delta T$ time</td>
<td></td>
<td>limited repetitions</td>
</tr>
</tbody>
</table>

Figure 8.3: Panorama of control parameters for generation. The scheduling process is divided into 3 basic steps: source, target and path (from, to, how). Both source and target can be implied either through local (state) or contextual (pattern) parameters. The scheduled path is described with the help of constraints, which can be local (intra-state) and global (inter-state), or may concern information about coherency/novelty tradeoff and the rate of repeated continuations. Problems of different classes can be combined, each time giving a different generation method

8.5 General Architecture for Three Party Improvisation in Collective Improvisation

The internal architecture of GrAIPE is shown in Figure 8.4. This architecture consists mainly of six modules which can act either concurrently or sequentially. On the far left we consider the musician, who feeds information into 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 side of the figure we see the renderer, the unit that sends generated audio out to the environment.
The MMFG structure lies inside the Multiple representation module, and entails the long and short-term stylistic features of the learned material. This material can be either pre-learned sequences or the improvisation flows of the instrument players that the systems learns on-the-fly.

![Overall Computer Architecture of GrAIPE](image)

**Figure 8.4:** Overall Computer Architecture of GrAIPE

The short-term memory processing module is responsible for detecting the short-term memory features of the overall improvisation. In order to internally reconstruct a complete image for the improvisation’s momentum, this module gathers information both from the representation module and the scheduler; from the first in order to know what is being played by the musician and from the second so as to monitor the computer’s short-term playing.

In the lower side of Figure 8.4 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 her/his questions. The performer lances queries under the form of musical constraints. She/he can ask, for instance, about the existence of navigation paths -a.k.a generated sequences- according to a certain target state, which are subject to a combination of a number of global and local constraints (see Figure 8.3). In order to respond, the solver attires information from the representation module. The minute the performer takes a decision, information is transmitted to the scheduler.

The Scheduler is an intelligent module that accommodates commands arriving from different sources. For instance, a change-of-strategy command by the performer arrives via the interaction core module to the scheduler. The scheduler is responsible for examining what was supposed to be scheduled 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 the 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 to emergency situations:
when the first detects a surprise event, instead of transmitting information via the
representation module -and thus not making it accessible unless information reaches
the interaction core-, it reflects information directly to the scheduler in the form of a
’scheduling alarm’. Hence, via this configuration, we leave open in our architecture
the option for the system to take autonomous actions under certain criteria. The last,
in combination with what has been mentioned earlier in this section, establishes in
full the three party interaction in a CAI context.

8.6 Interaction Modes

8.6.1 GrAlPE as an Instrument

The interaction core encloses all stylistic generation schemes relative to the instru-
ment paradigm. The computer user-performer can combine all different methods
and approaches discussed in Section 4 to interact with material learned before or
during the improvisation session and control a stylistically-coherent improvisation.
The dialogue with the computer is supported with methods based on the recognition
of patterns and sequence scheduling, through the use of musical constraints (time-
to-target, path constraints, etc.). There exist an infinite number of different possible
configurations for such an interaction to occur. Here, we present two basic tracks: one
for graphic and one for physical interfaces.

A very simple but efficient graphical interface is built on the concept shown in
Figure 8.5. The vertical axis represent the “reference time” of the sequence modelled
by MMFG, with the possibility that the size of the last will gradually increase from
the right during the improvisation. The horizontal axis represents the “improvisa-
tion time”, which is the time elapsed from the beginning of the improvisation. Con-
sequently, each point of this field corresponds to one moment of the improvisation
time, as well as to a position inside the MMFG. Due to the axis configuration, a linear
navigation without any continuations would graphically result in a linear function
\[ y(x) = x + b \]. Programming time goals, with the help of such a graphical interface, is
straightforward. Just by selecting a point in the field, the scheduling parameters of
“where” and “when” for the target state get instantly defined. The advantage of this
approach is that it allows for an overview of the entire improvisation time with respect
to the regions of MMFG that have already been used for improvisation, as well as for
a view of linearity vs. non-linearity and thus coherency vs. novelty. A generalized
interaction would consist in defining a number of points from which the generation
should pass in the future. Finally, by attaching more information at each point or in
between points, it is possible to further refine the generation process with additional constraints.

The integration of a physical instrument-like interface may also result in interesting human-computer interactions. Let’s take, for instance, the case of a midi piano. Such an instrument facilitates the insertion of patterns into the generation system. We could then use these input patterns as a guide for automated generation. By using the “as quick as possible” primitive of 1 we could imagine a scenario where the user indicates only the target patterns and the computer tries to rapidly assign to these patterns by generating the rest of the material while either keeping, or not, the stylistic features of the sequence. By generalizing the above approach, it is easy to imagine various interactions based on the augmented instrument paradigm. These interactions are based on the principle that the user provides note input that is not played as is, but it is used as first material for the computer to generate stylistically coherent content. Such an approach could be of interest to people that are not experts or have not reached the end of their learning curve for a particular music style. By implying their musical ideas in the form of a sketch, and with the help of the computer, they are able to participate, at least to a certain degree, in a collaborative improvisation context and enjoy making music with other people.

Figure 8.5: Graphical Interface for the interaction core. The vertical and the horizontal axis represent the reference and the improvisation time respectively. We suppose that the system has learned the sequence $S$ of Figure 8.2 through the MMFG (on the vertical axis). Linear improvisation corresponds to the linear function $y(x) = x + b$ (vectors $[0, A]$, $[A', B]$, $[B', C]$, $[C', D]$). Continuations through external transitions correspond to vectors $[A,A']$, $[B,B']$, $[C,C']$ and are due, each time, to the structure of MMFG. A point in this 2−D plane specifies the “where” (reference time at the vertical axis) and the “when” (improvisation time at the horizontal axis). The user can guide the improvisation process by setting points -thus Q constraints- in this 2−D plane. In the case shown in the figure, we may consider that the user, at time 0, selected the point D. This means that the system should improvise in such way so as to find itself after 17 notes at note fa. The path $0A'BB'CC'D$ shows the sequence that has been selected by the computer in order to fulfil the constraints set by the user. The produced sequence is the one shown in the figure below the horizontal axis.
8.6.2 GrAIPE as a Player

The reactive component includes strategies inspired from the player paradigm. It consists of methods that assign autonomy to the system so as to participate as an independent agent. It is implemented inside the interaction core and the scheduler module.

Nonetheless, in order for such an autonomous agent to exist, the last needs to perform a thorough understanding of the improvisation environment. In terms of stylistic generation, this is translated as the capacity to build internal representations and detect events arising from every flow in the improvisation context. Methods for representation of multiple flows and with respect to various music parameters, as well as methods for detection of patterns, are both natively supported by MMFG through the strategies described in (Maniatakos, 2012). With the help of these methods, it is then possible to envisage strategies for programming general modes of interactions in the musical level that this agent should apply. For example, it is possible to schedule general responses of the system that are triggered by detected events. In this sense, control methods of 1 can serve as the means by which to design one or more behaviors and interactions modes, that the system will eventually apply depending on the evolution of the improvisation.

Another useful approach arises from a predictive utilization of the recognition component. By using a memory window for the recent past of the instrument players’ flows, one can use the results of the pattern detection of the MMFG to predict future events. Then, by combining the last with the above strategies of detect and act, it is possible to anticipate future behaviors of the agent by associating future scenarios to predictions.

Of particular interest in the design of the reactive component are the methods for combining material occurring from different sequences. The capacity of a system to combine material arising from different participants of the improvisation, as well as the capacity to retrospect its own improvisation, are crucial for the complexification of the agent’s behaviour. In a context of an improvisation of more than one instrument players, the system analyzes in parallel the multiple flows and can improvise according to heuristics for balancing the generated material among the different flows learned.

8.7 Conclusion

The interest in the present study of the three-party interaction context for modern improvisation arose from the double role of the computer operating, on occasion, as an independent improvisation agent and as an instrument- depending on the type of human intervention -as an instrument player or a computer performer. This resulted in a large number of potential interactions that can simultaneously occur among the
improvisation participants. By relying on the MMFG structure, we establish a set of tools and methods for dealing with the two paradigms: the one of the computer as an independent agent and the second as a high-level musical instrument. Similarly to former sequential models for sequence generation, MMFG-based generation is the result of navigating inside the graph-like representation provided by the MMFG. The novelty of our approach lies in our successful effort to provide a complete set of tools in order to control the musical result of these heuristics and methods. The conception of these toolset follows a bottom-up arrangement, in the sense that simple tools (primitives) are defined, in a first place, in the form of queries over the MMFG representations, and more complex ones arise from the combination of the former.

References