Projetopê: a guided free improvisation co-organized through the interactions between human musicians and a digital machine

Stéphan Schaub[1 2] **, Said Bonduki**[1*]

[1]Instituto de Artes - Unicamp
R. Elis Regina, 50 - Cidade Universitária, Campinas - SP

[2]Núcleo Interdisciplinar de Comunicação Sonora - Unicamp
R. da Reitoria, 165 - Cidade Universitária, Campinas - SP

schaub@unicamp.br, sabonduki@gmail.com

Abstract. *In the following contribution we present the overall outline of a musical improvisation environment in which musicians interact with a digital machine endowed with capacities to listen, to memorize, to communicate visual information and to synthesize sounds. The proposal innovates by shifting the focus away from real-time interactions to concentrate on the organization of larger time spans and on the production of a type of guided improvisation situated somewhere between the practice of entirely* free *improvisation and written,* open *composition. To achieve this goal, temporal segments are defined and their succession as well as their characteristics dynamically driven by the interactions between the improvisers and the digital machine.*
The name Projetopê *refers both to a subset of these possibilities and to an initial implementation programmed in Max/MSP. The latter presents itself as an "extended score", materialized on individual computer screens. Over the course of the performance the machine analyzes parameters such as pitch, register, duration and dynamics. It then uses the information to trigger the transitions between sections of the improvisation (thus making their durations variable) as well as to determine the pitches / harmonies it itself projects, usually between – occasionally also during – these same sections.*

1. Introduction

As soon as digital computers became sufficiently compact and (relatively) widespread, programmers / musicians started to develop and to explore performance environments in which a (human) instrumentalist could "interact" with a digital machine in the production of a common, improvised musical unfolding. George Lewis set the tone, so to speak, in the early 1980's [1] with his pioneering *Voyager* [2]: a computer program that endowed a machine with the capacities to "listen" to an ongoing musical stream and, based on this input alone, produce its own, moment to moment sonic contributions.

Since then many different approaches have been developed and explored that expand on this general idea [3], from Robert Rowe's *Cypher* [4], John Biles *GenJam* [5] to Assayag *et al.*'s *OMax* and *SoMax* [1]. Beyond their sometimes fundamental differences, all these systems share a common concern with the "present" of the performance, the moment at which, in other words, the "interactions" occur and the unfolding sound deployment takes its source.

The present contribution is part of a larger research project that seeks to approach human-computer co-improvisation from a different, and in many ways complementary angle. It focuses not so much on aligning the machine's sound production within what it understands about the performance's "present" (or its immediate vicinity) as on the ways in which larger time-frames can be defined "in the abstract" and *their* specific deployment be made to follow from the ongoing interactions between the improvisers and the machine.

With this context, we hope to motivate the overall features given to the project's first concrete implementation - and main focus of the present article: *Projetopê*. We will start our presentation, in the next section, with an exposition of some general principles that emerged from the first steps taken towards this objective. These include the definition of an extended notion of musical "score" (a succession of segments in time each potentially attached to different sets of performance-instructions and / or specific behavior from the part of the machine); the idea of making its display as well as its handling part of the machine's functions; and the listening and memorization capabilities required of the machine so as to let it interact with musicians without needing any external supervision. At this general level of description, the details of the sound-generation processes, are left open. There are indeed numerous ways in which such generation can be made a function of the information entering the machine at any given moment, of the particular "segment" of the performance one is currently in, or of a combination of the two. Let us also point out that the machine's output does not *necessarily* have to include sound. Its role can be restricted to that of a "conductor" helping organize an improvisation as it unfolds in time.

The term *Projetopê* refers both to set of strategies for organizing improvisation in interaction between humans and machines as well as to a given concrete realization of these strategies. The performance environment it provides invites a variable number of musicians to improvise freely together with a "machine" capable of listening to each participant individually, to memorize events, to

*Supported by Capes.

[1]For more information see https://forum.ircam.fr/projects/detail/omax-4/. Last accessed: 23/05/2025.

communicate visual information and to synthesize sounds, thus contributing to the global musical unfolding.

The description of its features is the focus of section 3. The overall principles presented are roughly equivalent to the information the musicians are given to navigate through *Projetopê*'s environment over the course of a specific performance. In accordance with the nature of the present gathering, particular emphasis will be given to the concrete solutions encountered, which are the focus of section 4. For our first prototype, the main framework selected has been Max/MSP [2], together with the Bach library [3]. Some constraints involved in developing the project demanded particular solutions, which ultimately shaped the first phase of its implementation. Such is the case in the use of the MIDI protocol and digital sound synthesizers.

We will then wrap up our presentation with some remarks about the performance of *Projetopê*, some possible extensions and perspectives for future research.

2. Some General Principles

Before getting into any detail, let us rephrase the questions underlying the project as a whole in the following way: can we program a computer to partake in the *organization* of the overall formal deployment of an otherwise free improvisation? Can this organization be made a function of the *interactions* occurring contextually, over the course of a particular session's deployment between the machine and the improvisers?

A first step towards answering these follows from the observation that to organize the temporal unfolding of a given improvisation beforehand is to devise some kind of a musical "score". We define the latter, in the most general possible terms, as organizing the performance in a series of sections characterized / distinguished from on another by some performance instructions, concrete (musical or visual) occurrences and/or behaviors on the part of the machine. The duration of each section and its characteristics can be completely preordained and known to all beforehand. But these elements can also be left open and their specific "values" or "form" be functions, at each step, of what is happening, or of what has happened, over the course of a *specific* performance.

For the latter to occur we have to postulate that the machine is endowed with a capacity to listen to the ongoing improvisation. Things can also be taken a step further by adding that it is also endowed with a capacity to memorize aspects of past events. Neither is trivial from a technological perspective, a point we will get back to further down. But assuming such capacities are at hand, various strategies can be devised to render the score "dynamic" and dependent on the contextual information these capacities bring in. The transition from one section to the next might be triggered, for instance, by the machine's recognition of some specific musical configuration. While the dimension impacted here is the duration of the sections, similar principles can extend to the determination of the characteristics attached to them. The options here are vast. Even more so when one considers how broadly we defined the term "characteristics".

Focusing more specifically on the "performance instruction" leads to considerations about another set of features that can contribute to our objectives, namely, the capacity of the machine to communicate visual information in "real time" to the performer(s), ideally, through individual tablets or computer screens. By placing both the handling and the display of performance instructions into the digital realm makes it possible, along lines similar to the one mentioned above, to make the score - as a succession of performance instructions - depend on contextual information.

An important extension of the latter follows from letting the elements displayed by the machine reach beyond the performance instructions *per se* to include information relative to its own perception of the present, its memory of the past or even its anticipation of what its future actions might be.

The general framework being presented, we may now turn our attention to our first concrete implementation / prototype of an improvisation environment of the type just described.

3. *Projetopê*

The "score" of our *Projetopê* prototype alternates between two types of sections. During the first, which we will call *main sections*, the performers are invited to express themselves freely, remaining attentive to each other's outputs as well as to the information they receive from the computers screen, in the production of a collective sound totality. During the second, which we will call *transition sections*, a single chord is synthesized and played by the machine. While the performers may play over that chord, they are generally invited to mark a pause, entering again only at the start of the next *main section*. Occasionally, a chord heard over a *transition section* might extend, in part or in whole, over the following *main section*.

The durations of the *transition sections* are relatively short and (at least at the present state of development) part of the parameters input beforehand. Those of the main sections, on the other hand, are more extended. The transition out of one and into the next (transition) section, is made dependent on the alignment of all the performers with a specific configuration, as follows.

Throughout the performance, the machine locates each performer's production on a grid defined by register (divided into "low", "medium", "high") and intensities (also divided into "low", "medium", "high"). We call each of the nine possible register / intensity combinations an "area" (see Figure 1 below with each *area* numbered from 1 to 9). Each main section specifies, for each performer, a particular *area* as being the one in which to remain over

[2] Available on: `http://cycling74.com/products/max/` last accessed: 20/05/2025.

[3] For more information, see `https://www.bachproject.net/`. Last accessed: 20/05/2025.

a minimal period of time for the transition to occur. The latter, however, will only happen if *all* performers agree to it and also remain on their respective *transition areas*. In other words, the transitions are decided collectively and signaled musically through particular positions held within the intensity / register space.

The chords that are projected during the transition sections first follow a process of "accumulation" with each new chord adding pitches to the one heard previously. When the number of pitches in the chord increases to a set, maximum number, the process is reversed, and pitches are gradually subtracted from the chord. The operation is continued until only one pitch remains, marking the end of the performance.

The actual pitches included in each transition chord are determined by the combination of two factors. Over the course of a main section, each performer contributes a fixed number of pitches that will alter - add or subtract from - the previously heard chord. This number varies for each performer and with each section, taking first positive, then negative values. At the same time, each performer can emphasize particular pitches by playing them over longer durations than any other. The machine remains attentive to which pitches are thus singled out by each and periodically updates this information. At the moment of transition, the machine collects all the pitches thus emphasized, adding or subtracting them from the pitch-content of the previous chord.

All the while, the machine provides information about the current "state" of the performance to each improviser through individual computer screens. Concerning the chords synthesized and projected at the transitions, the information provided is as follows: the notes that are currently included in the transition chord (in traditional staff notation); how many pitches the performer will add or subtract from it by the end of the current section and which notes the performer has played longest – and thus emphasized - at the current moment of the section (also in staff notation). Concerning the transitions, the machine indicates, to each performer, what his or her current *transition area* is as well as those of the remaining musicians. It also indicates to each, at any given moment, in which area of the register / intensity grid it perceives the sonic production to be located.

The performers involved in a particular realization of *Projetopê* are thus *collectively* responsible for triggering transitions as well as for the pitch content of the chords sounded during these transitions. Much of the sounds heard over the course of the main sections remain freely improvised. The machine, however, helps shape the performance as a whole, inviting each participant to negotiate its "present" while also taking into consideration information about its potential future developments.

4. Patch development

Projetopê focuses on real-time analysis of a musical performance, with the captured data informing the work's unfolding structure. After defining which performance parameters would be analyzed, we determined which formal aspects they would influence. To implement this, the calculations performed on data from the *satellite patches* are consolidated in the *central* patch, allowing the performance flow to be coordinated. Communication between patches is handled by the *udpsend* and *udpreceive* objects, which transmit data over a network. Within this framework, a chain of operations analyzes the notes played during the performance, their compliance with the selected parameters, and how they influence the progression of the work.

The structure of the digital setup is built on the interaction between a *central patch* and several *satellite patches*. The former receives data from the latter, each of which extracts and processes information from an individual musician's performance as described previously and forwards relevant data to the *central patch*. It is through the analysis of this aggregated data that moments of transition and the harmonic content used in the electronic synthesis are determined, thus giving shape and coherence to the performance as a whole.

Defining the functions required for the latter to occur led to the design of an initial patch structure that remains in place in the current version. This structure is organized into operational modules, each responsible for a specific task within the data processing flow. These modules can be changed or adapted to meet possible future demands as the project is further developed. Thus, each patch is organized around the following modules:

1. Definition and storage of performance **parameters**;
2. **Machine listening** and capture of data from performers;
3. **Analysis** of captured data;
4. **Visualization** of analyzed data in graphical, numerical, and symbolic notation;
5. **Communication** between performers' machines;
6. Sound **synthesis**.

Each module processes incoming data in real time, forming a chain that begins with machine listening and proceeds through the exchange of information between the various processing modules. The following sections describe the functionalities of each module, as well as the flow and exchange of information between them, enabling the performance to occur as designed. For the implementation of the proposed modules, the *Bach* library for Max has been selected, as it provides robust tools for list processing and visualization in traditional staff notation.

Below, we describe the procedures for each module in the *central patch*. Note that the *satellite patches* have reduced functionality, considering that computation and control are taken care of in the central one.

4.1. Defining and Storing Performance Parameters

A key feature of *Projetopê* is the flexibility with which its performance parameters may be set. These include indi-

vidual parameters specific to each performer and global parameters governing the form and transitions of the performance as a whole. The latter are controlled by the *central patch*.

Individual parameters include the number of pitches each performer contributes to the transition chord in a given section. If set to a negative value, this parameter instead determines the number of pitches removed from the transition chord. Other parameters determine the transitions: the position the performer must occupy in the register–intensity grid (see Figure 1) and the required time they must remain in that area for the transition to occur. These parameters are updated with each new *main section*.

The global parameters, managed by the *central patch*, specify the total number of sections in the piece, the number of pitches in the chord deployed at each *transition section*, its duration, and the synthesis parameters that are used in its rendition.

4.2. Listening and Data Capture

The design of the project required a data capture model capable of updating the system in real time, ensuring that each newly played pitch is immediately incorporated into the patch's processing. To meet this requirement, we used the MIDI (Musical Instrument Digital Interface) protocol, which is supported in the current version of the patch. MIDI was opted for the purpose of simplifying the data inputs, which permit easy verification of the data flow inside the patch. The MIDI capabilities of the instruments used during early stages of development were key in making this choice. The MIDI data captured includes the pitch of each note attack / release (and thus, duration) and its velocity (attack intensity). These parameters form the foundation of the processing chain.

Given the choice of MIDI, objects from the *Bach* library were used to transcribe each performer's playing in real time. Notably, the *bach.roll* and *bach.transcribe* objects were used. The latter receives MIDI data from an instrument and constructs a score in *bach.roll*. This process operates continuously, so that each performer's playing is both recorded live and simultaneously displayed in the form of a score. This provides precise temporal capture of events, a crucial feature for the data analysis stages required for the performance to happen.

4.3. Visualization

The processed data are presented to performers in three different forms: numerical values, symbolic notation, and bar graphs. A central element of the visualization is the performer's position within the grid at any given moment. The performer's current position within the nine possible areas of the grid is thus highlighted in real time, in yellow as shown in Figure 1, with register represented on the x-axis and intensity on the y-axis. A second matrix indicates the area that needs to be occupied for the next transition to happen. Once a transition occurs, this target is updated with the data corresponding to the next section.

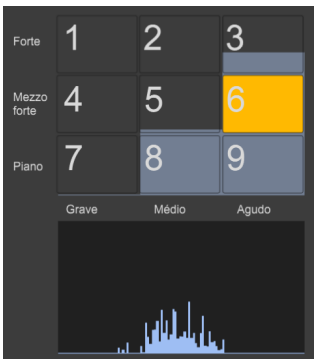

Figure 1: Transition grid

The light gray bar graph visualization seen in the background of the matrix in Figure 1 reflects how the musician's performance over the recent past is distributed over the three regions of the register. In Figure 1 for instance, the low register has not been played, the medium one roughly 35 percent of the time and the high register roughly 65 percent of the time. This information is calculated in real time, using a fixed time window to estimate the proportion of time the performer spends in each region of the register. To give the performer a clear notion of how their notes are spread across their register, the lower graph, shows pitch-by-pitch data across the instrument's register during the whole section. As an example, in Figure 1, the lower graph (in blue) shows a concentration of notes in the middle register, while the upper graph reveals that the pitches played most recently were concentrated in the upper register.

There are three symbolic notation visualization displays that shows as many distinct streams of information:

1. The real-time transcription of the performer's playing;
2. The pitches he or she would contribute to (or subtract from) the next transition chord, were the section to end at that moment;
3. The chord composed of pitches drawn from all of the performer's contributions over the course of the performance as collected by the machine.

The *bach.roll* and *bach.score* objects enable this symbolic visualization. To allow performers to see the pitches that would contribute to the chord (were the section to end at that moment), *bach.roll* is also used. It represents the pitches that have been played with the overall longest duration (in milliseconds) during the current section, with this duration represented proportionally by a horizontal bar. This is useful for the performer to gauge how long he or she must play a given pitch to replace another one in the set. An instance of this is shown in Figure 2.

To create the final chord for each *transition section*, the *central patch* collects and consolidates the pitches that have been played the longest by each performer and sends the result back to the *satellite patches*, making this information available to each performer using the *bach.score* object. This process repeats for each section,

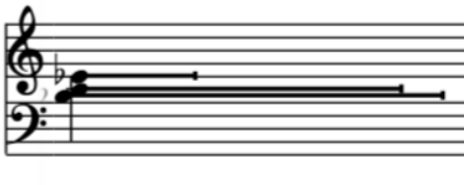

Figure 2: Bach.roll object visualization of the pitches that have been played the longest

constantly transforming the chords played at each *transition section*. The data processing logic is described in the next subsection.

4.4. Extracting performance data

The first stage of data processing focuses on evaluating the overall duration of the pitches played by each performer during a given *main section*. In order to evaluate the ones that were deployed over the longest time periods overall, the individual durations of their appearances are added together. To achieve this, a dedicated processing chain of analyses was implemented.

The *bach.roll* object serves as the primary data structure, as its visualization function relies on the capture and storage of performance information in real time. From the *bach.roll* data set, pitch, duration, and onsets are continuously extracted. In order to ease error correction and overall legibility, the MIDI pitch numbers are converted into "symbolic" notation ranging from B0 to Bb9. The duration (in milliseconds) of each pitch event is also deduced from the onsets of its corresponding "note on" and "note off".

A *coll* object thus continuously receives data pairs consisting of pitch name and duration. This object contains 128 lines and an non-determinate number of columns. Each line corresponds to a pitch name. At each new dataset entering, its duration value is added in the next available column of the line corresponding to its pitch name (see left hand side of Figure 3).

In a second *coll* object this information is periodically consolidated, summing the durations corresponding to each pitch name (see central column of Figure 3), The result is then sent to the visualization mentioned previously (lower part of Figure 1). Finally, the data is sorted in descending order relative to the consolidated durations (see right hand side of Figure 3) and the n pitch names with the highest duration values are sent to the second visualization (Figure 2), with n corresponding to the number of notes contributed by the performer during the current *main section* to the chord that will be played during the next *transition section*.

A second set of calculations determines the average dynamic level and the distribution of pitches across registers. Let us consider pitch first. Three separate *coll* objects are used, each corresponding to one of the three registers: "low", "middle", and "high". Since register

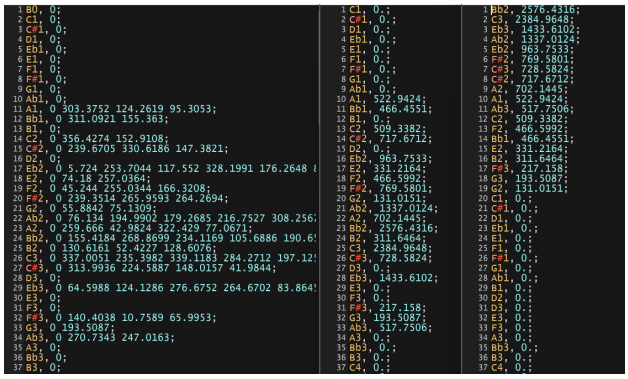

Figure 3: Lists of pitch durations

boundaries vary across instruments, the patch provides adjustable limits for each region. Using this division, the system calculates the percentage of pitches played within each register, a metric that is crucial for determining the performer's current position on the pitch-intensity grid. A similar strategy is applied to the division of the velocity values into "low", "middle", and "high", respecting the particularities of each instrument. The average velocity of the pitches played within a sliding temporal window is computed determining the current position within the grid.

4.5. Section transitions

The *central patch* continuously monitors whether the performers have met the conditions required to trigger a transition. When they are all positioned in the correct *transition areas* and remain there for a predefined minimal duration, the transition to the next section is initiated. A countdown to the next *transition section* then appears on all the patches.

The transition areas and minimal durations to trigger the countdown are parameters that can be treated as a formal element of the "composition" of the performance, with each session characterized by its own configurable combination of parameters.

Relative to the transitions, each interface displays three key elements: the performer's current real-time position within the grid, the next transition area, and those of the other musicians. In the latter case, the area is referred to in numerical form so as to avoid the need to display additional grids. The display in real-time of the musician's position on the grid shows how the system is interpreting their performance, allowing them to adjust their playing as seems fit. The underlying analysis is calculated within a rolling time window, which can be adjusted to control sensitivity: a shorter window makes the system more responsive to small variations, while a longer window, by averaging over more data, stabilizes the position against minor fluctuations.

4.6. Synthesis

The chords heard during each *transition section* is produced by the digital synthesis module. The pitches stored in the *bach.score* object provide the basic data of this operation, which can be implemented in various ways. In the

current version of the project, sound synthesis is performed using the *vst~* object, which supports *Virtual Sound Technology* (VST) plugins—commonly used in *Digital Audio Workstations* (DAWs). In Max, the *vst~* object enables direct use of audio plugins. Thus, the *bach.score* object sends the chord-data to the *vst~* object, which then synthesizes the sound. Available synthesis parameters include overall duration and sound envelope.

5. Directions and perspectives

At least two areas have been identified for future development: capturing acoustic sounds directly (rather than relying on MIDI) and implementing a synthesis method that does not depend on VSTs. Expanding the listening module to support non-MIDI instruments would broaden the project's applicability, while further development of synthesis options would provide a wider palette of sonic outcomes. The MIDI protocol has proven useful to stablish the patch´s data flow, allowing the emergence of a system ready to receive sound analysis information, while the harmony create from the data gathered from the performance can be organized in many different ways for the use of sound synthesis.

The modular organization of the patch allows individual processing stages to be replaced without disrupting the overall system, making it easy for other users and groups of performers to customize. There is no fixed limit to the number of performers that can participate; however, supporting larger ensembles may require certain adjustments.

6. Final remarks

The development of *Projetopê* represents a significant contribution to the field of musical co-improvisation between a machine and human improvisers. It proposes a performative environment in which the machine is no longer merely an immediate sound generator, but acts as a structural co-organizer of the musical form.

The integration of listening, memory, visualization, and sound synthesis, distributed in a modular, scalable system, enables the exploration of a variety of configurations as well as new modes of interactions between humans and computational systems in performance contexts.

The this first prototype, many parameters given fixed values from the onset. Most of them, however, such as *transition areas*, the duration of the *transition sections*, the number of pitches contributed by each performer to the latter's chords, etc. could also be envisioned as resulting from functions that take characteristics of the performance as it unfolds in time as its input variables. The possibilities of making a computer crash in mid-performance are obviously endless.

By shifting the focus from the "present moment", characteristic of more "traditional" approaches, to an expanded formal perspective based on sections with dynamic, parameterized transitions, the project makes room for more complex improvisational practices, involving not only musical gestures but also formal decisions emerging from collective listening and mediation by the machine.

The technical solutions adopted here, while proven effective as an initial prototype, also point toward new avenues for future developments. The modular design and flexibility on the number of performers make *Projetopê* a versatile platform which we hope, will open to collaboration with diverse groups and artistic contexts.

The project remains under development, with the aim of deepening the dialogue between human improvisation and computational agency, and thereby contributing to the advancement of experimental musical practices grounded in human–machine interaction.

7. Acknowledgments

The authors would like to thank all that have been involved, in one way or another, in the elaboration and testing of *Projetopê*: the musicians Alexandre Zamith Almeida, Andre Martins, Manuel Falleiros and Mateus Santin Mendes; the researchers and programmers Danilo Rossetti, Guilherme Zanquetta Lalier Avila, Ivan Eiji Simurra, Lucas Hernandes Pureza and Mikhail Malt, as well as staff if the *Nucleo interdisciplinar de comunicação sonora* (NICS-UNICAMP) and of the *Instituto de Arte - Música* (IA Musica-UNICAMP).

References

[1] Curtis Roads. *From composers and the computer*. A-R Editions, 1st edition, 1985.

[2] George E. Lewis. Too many notes: Complexity and culture in voyager. *Leonardo Music Journal*, 10:33–39, 2000.

[3] Toby Gifford et al. Computational system for music improvisation. *Digital Creativity*, 29(1):19–36, 2018.

[4] Robert Rowe. Machine listening and composing with cypher. *Computer Music Journal*, 16(1):43–63, 1992.

[5] John Biles. Genjam: A genetic algorithm for generating jazz solos. *International Computer Music Conference Proceedings*, 94:131–137, 1994.