Universidade de Évora - Instituto de Investigação e Formação Avançada

Programa de Doutoramento em Música e Musicologia

Área de especialização | Composição

Tese de Doutoramento

Comprovisador – a Real-Time Notation System. Improvisation, algorithmic composition and dynamic notation, for soloist(s) and ensemble.

Pedro Nuno Marreiros Louzeiro

Orientador(es) | Christopher Bochmann

António de Sousa Dias

Évora 2024

Universidade de Évora - Instituto de Investigação e Formação Avançada

Programa de Doutoramento em Música e Musicologia

Área de especialização | Composição

Tese de Doutoramento

Comprovisador – a Real-Time Notation System. Improvisation, algorithmic composition and dynamic notation, for soloist(s) and ensemble.

Pedro Nuno Marreiros Louzeiro

Orientador(es) | Christopher Bochmann

António de Sousa Dias

Évora 2024

A tese de doutoramento foi objeto de apreciação e discussão pública pelo seguinte júri nomeado pelo Diretor do Instituto de Investigação e Formação Avançada:

Presidente	Eduardo Lopes (Universidade de Évora)
Vogais	Carlos Caires (Escola Superior de Música de Lisboa do Instituto Politécnico de Lisboa) Christopher Bochmann (Universidade de Évora) (Orientador) Daniel Moreira (Instituto Politécnico do Porto) Isabel Maria Machado Abranches de Soveral (Universidade de Aveiro)

Évora 2024

Ao meu filho, Jorge Benjamim.

Agradecimentos

I should like to thank my PhD supervisors Christopher Bochmann and António de Sousa Dias for their advice and continued support.

This research is supported by FCT by means of a PhD studentship (POCH / EU).

I would also like to thank my family – first and foremost, my life companion Adriana – my parents and my in-laws. This work would not have been possible without their support.

A very special thank you to Carlos Caires and José Luís Ferreira for their invaluable support. To every musician who played in any of the comprovisation sessions carried out during this research I express my gratitude – there are too many names to name here (129, if one counts only the public sessions; actually, more), but I am grateful to all. Some musicians and friends who went above and beyond to help me with my endeavour – and thus have my deepest thanks – are: Alberto Roque, Andrea Agostini, Bruno Vítor Martins, Carlos Garcia, Carlos Guedes, Carlos Ramalho, Christophe Guerreiro, Constantin Basica, Daniele Ghisi, David Alves, Desidério Lázaro, Evis Sammoutis, Fábio Cachão, Fátima Vargas, Filipe Lopes, Filipe Valentim, Francisco Brazão, Francisco Serôdio, Georg Hajdu, George Georgiou, Gilberto Bernardes, Gonçalo Gato, Henrique Portovedo, Inês Nunes, João Barradas, João Centeno Moreira, João Costa, João Lourenço, Joaquim Nascimento, Jonathan Bell, Jonathan Shapiro, Jorge Ramos, Mariana Seara, Manuel Moreira, Miguel Azguime, Nelson Louzeiro, Ninon Gloger, Nora-Louise Müller, Paulo Gaspar, Pedro Calquinha, Pedro Centeno Moreira, Pedro Finisterra, Pedro Guerreiro, Pedro Sá, Phillipe Trovão, Ricardo Toscano, Rui Mourinho, Rui Penha, Rui Travasso, Sandeep Bhagwati, Sara Ross, Sérgio Leite, Tomás Moital, Vasco Ramalho, Vera Batista, Virginie Bove, Yiannis Christidis and Zé Eduardo. Furthermore, I would like to make a special reference to Andrea Agostini and Daniele Ghisi, the creators of the `bach` Library, for developing this excellent tool and for their outstanding support in troubleshooting and implementing the majority of my feature requests.

I dedicate this work to the memory of composer José Luís Ferreira, avid supporter of this research.

Contents

List of Figures

List of Tables

Acronyms

CAC Computer-Assisted Composition

DAW Digital Audio Workstation

EA Escola de Artes

ESML Escola Superior de Música de Lisboa (Lisbon College of Music)

FCT Fundação para a Ciência e a Tecnologia

GUI Graphical User Interface

HASGS Hybrid Augmented Saxophone of Gestural Symbiosis

HMI Human-Machine Interaction

ICMC International Computer Music Conference

IIFA Instituto de Investigação e Formação Avançada

ILLIAC Illinois Automatic Computer

IMS Interactive Music Systems

IRCAM Institut de Recherche et Coordination Acoustique/Musique

LAN Local Area Network

MFCCs Mel-Frequency Cepstral Coefficients

MIDI Musical Instrument Digital Interface

MIR Music Information Retrieval

ML Machine Learning

MuMe Musical Metacreation

NIME New Interface for Musical Expression

NMP Networked Music Performance

RTC Real-Time Composition

RTN Real-Time Notation

SMP Stochastic Music Program

TCP Transmission Control Protocol

UDP User Datagram Protocol

UE Universidade de Évora

URL Uniform Resource Locator

WAN Wide Area Network

Abstract

The present thesis is intended to contribute to a reflection on new ways of combining improvisation and composition aspects in a performance context, with special focus on the use of dynamic notation (animated in real time), through research based on artistic practice. To define this practice, we have chosen the term Comprovisation (in Portuguese: Comprovisação), a term that can be understood as a musical performance context in which elements of composition and improvisation coexist in aesthetic interdependency. In order to carry out this practice, a system – **`Comprovisador`** – was designed to enable mediated soloist-ensemble interaction using machine listening, algorithmic compositional procedures and dynamic notation, in a networked environment. As a soloist improvises, **`Comprovisador`**'s algorithms produce a score in real time that is immediately sight-read by an ensemble of musicians, creating a coordinated response to the improvisation. This interaction is mediated by a performance director who does so by manipulating algorithmic parameters. Implementation of this system requires a network of computers in order to display notation (separate parts) to each of the musicians playing in the ensemble. More so, wireless connectivity enables computers – and, therefore, musicians – to be far apart from each other, enabling space as a compositional element. Through the development of **`Comprovisador`** and the performance practice it enabled, a number of surrounding issues were researched and studied, namely, the use of composition algorithms in a real-time notation system, the suitable resources for mediating a comprovisation performance, and the effectiveness of a graphical synchronisation strategy within a dynamic notation interface. Further studies were made in order to assess the applicability of this system in an educational context regarding improvement of sight-reading skills. The findings of this research and their impact on the system's ongoing development – and, ultimately, its musical use – will be discussed herein.

Keywords: Comprovisation, Improvisation, Algorithmic Composition, Dynamic Notation, Network Musical Performance

Sumário

Comprovisador – um Sistema de Notação em Tempo-Real

Improvisação, composição algoritmica e notação dinâmica, para solista(s) e ensemble

Esta tese procura contribuir para uma reflexão sobre novas formas de aliar aspectos de improvisação e de composição em contexto musical performativo, com especial foco na utilização de notação dinâmica (animada em tempo real), através de investigação fundamentada numa prática performativa. Para definir essa prática, foi escolhido o termo Comprovisação, termo esse que pode ser entendido como um contexto musical performativo no qual elementos de composição e de improvisação coexistem em interdependência estética. Para levar a cabo essa prática, foi desenvolvido um sistema – `Comprovisador` – que permite mediar a interacção entre solista e ensemble, usando *machine listening* (escuta automática), procedimentos de composição algorítmica e notação dinâmica, num contexto de performance em rede. Em tempo real, enquanto um solista improvisa, algoritmos do `Comprovisador` produzem uma partitura que é imediatamente lida à primeira vista por um conjunto de músicos, criando uma resposta coordenada à improvisação. A interacção é mediada pelo director da performance, através da manipulação de parâmetros algorítmicos. A implementação deste sistema requer uma rede de computadores para exibir notação (partes separadas) a cada um dos músicos que tocam no ensemble. Para além disso, a conectividade sem fios permite que os computadores – e, logo, os músicos – estejam distantes uns dos outros, permitindo o uso do elemento espaço, na composição. Através do desenvolvimento do `Comprovisador` e da sua aplicação prática em ensaios e concertos, foi possível estudar questões como a aplicação de algoritmos de composição em tempo real, a

eficácia de uma interface de notação dinâmica com estratégias gráficas de sincronização, os recursos para mediação de uma performance de Comprovisação e as potencialidades de aplicação deste sistema em contexto pedagógico para desenvolvimento de competências de leitura à primeira vista. Os resultados desta investigação e seu impacto no desenvolvimento continuado do sistema – e, em última análise, a sua utilização musical – são discutidos neste documento.

Palavras chave: Comprovisação, Improvisação, Composição Algorítmica, Notação Dinâmica, Performance Musical em Rede

1

Introduction

This work was born primarily from the will to create musical performances that could merge, in real time, aspects pertaining to the fields of composition and improvisation. From the start, the main intent was to gather in performance musicians with different skill sets – namely, those trained in improvisation and those who excel at sight-reading and interpreting musical scores[1]. To designate this artistic goal, I adopted the term **comprovisation** – an amalgamation of the words composition and improvisation[2].

After studying a number of ways one could achieve this goal, I considered the use of computer resources enabling features such as dynamic staff-based notation, algorithmic composition, machine listening and network-distributed computing of real-time scores. Machine listening could enable real-time analysis of a soloist's

[1] For simplicity, despite the reductionism implied, the former group will be referred to as 'improvisers' and the latter as 'sight-readers'.
[2] The origins and definition of the term comprovisation will be discussed in Chapter 2 (Section 2.2).

improvisation; composition algorithms could generate consequent musical responses to the improvisation; and dynamic notation, presented in networked computer screens, could enable musicians to sight-read the generated responses. Moreover, algorithmic parameters could be manipulated in real time enabling a conductor/composer to mediate the interaction between improviser(s) and sight-readers – between soloist(s) and ensemble. Hence, I arrived at the research question that follows.

1.1 Research Question

The main question driving this research is:

> **How would a real-time composition system featuring dynamic staff-based notation contribute to the practice of comprovisation involving improviser(s) and sight-readers?**

To be able to answer this question, I have developed a computer system – `Comprovisador` – that enables mediated soloist-ensemble interaction using the aforementioned resources: machine listening, algorithmic compositional procedures, dynamic staff-based notation and network capabilities. In real-time, as a soloist improvises, `Comprovisador`'s algorithms produce a score that is immediately sight-read by an ensemble of musicians, creating a coordinated response to the improvisation. Interaction is mediated by a performance director through parameter manipulation.

Implementation of this system requires a network of computers in order to render and display individual score parts to each of the musicians playing in the ensemble. More so, wireless connectivity enables computers – and therefore musicians – to be far apart from each other, enabling space as a compositional element.

Some subsidiary questions must be considered to help steer the research in the right direction, namely:

- Is the concept of **real-time composition** indeed possible?
  - How does it differ from **improvisation**?
- What other **comprovisation** paradigms already exist inside and outside of the computer music field?
  - Which aspects from existing paradigms can or should be implemented in `Comprovisador` and how?
- Which features from other examples in the **animated notation** field can or should be adopted?

  - What are the benefits and drawbacks of staff-based notation when compared to graphic and other types of notation often favoured in the field?

- What choices in **algorithm conception** and **notation interface design** can be made to facilitate and integrate the sight-reading process in a real-time generative context?

  - How can the likelihood of sight-reading errors be mitigated or otherwise incorporated in the aesthetic conception?
  - How can synchronisation between musicians be enforced and to what degree?
  - How to manage, from a musical standpoint, the various delays contributed by processes such as input audio analysis, network data transfer, sound propagation[3] and most especially sight-reading?

- What **novel approaches** could be introduced to enable new and desirable aesthetic outcomes thus leading to advancements in the field?

The last of these questions is indeed the most important and also the most complex one. The answer to that question presupposes two conditions: (1) having extensive knowledge of the state of the art in the field and (2) having substantial performance experience with `Comprovisador` allowing for experimentation with new approaches, followed by reflective analysis.

The development model better suited to satisfy the latter condition is the iterative design model. Consequently, this was the adopted model for the development of `Comprovisador`. Through iterative design, new approaches can be conceptualised, implemented and then tested during performance. Following every performance, a reflective analysis takes place, concerning the musical outcomes, leading to a decision vis-a-vis the new approach to either discard it, redesign it or fine-tune it and then retest it. Also, the reflective analysis phase is very useful as a planning and writing phase, aiming at publishing research results. This has been done, in the context of this doctorate.

The present dissertation is therefore the result of practice-based research. To date, `Comprovisador` has been used in eleven public comprovisation performances, establishing the artistic body upon which this research is based. In this document, I intend to review the developmental process of `Comprovisador`, analysing the merits and disadvantages of the features that were implemented and – most importantly – discussing the musical outcomes achieved thus far, as well as future goals. The texts published throughout this research will be included here, albeit in an updated and expanded form.

[3] Note that musicians can be physically distant from each other.

Regarding aims and characteristics, a comparison shall be made between `Comprovisador` and a number of systems that are established in neighbouring or overlapping creative spaces. Such comparison shall demonstrate how and why `Comprovisador` occupies a unique space and thus may contribute to a pertinent reflection on the issue of **comprovisation with mediated soloist-ensemble interaction using dynamic notation (live scores) and real-time algorithmic composition**.

Furthermore, an assessment of `Comprovisador`'s affordances in other application fields shall be made. In the educational field, an investigation will take place aiming to assess the benefits this system can provide in the improvement of sight-reading skills. Also, in the field of traditional (deferred-time) electroacoustic composition, `Comprovisador` shall be explored as a source material generator, through real-time stochastic score generation, sight-reading of the generated score (human rendition) and audio recording of the rendition process, followed by deferred-time application of electroacoustic composition techniques on the recorded material.

1.2 Anticipated Problems

A readily available system or environment able to facilitate the practice of comprovisation performances in the format that I envisaged (see Section 1.1) is something that, to the best of my knowledge, did not yet exist – ergo, the development of a new system was pertinent. Such enterprise should ultimately enable a reflection upon the musical outcomes that may be attained through that innovative format and thus a contribution to knowledge. However, before being able to reach this goal and address specific gaps in knowledge (or even identify them), it was necessary to conceptualise and develop the system, test it, explore with and within it, and gain performance experience with every developmental iteration. On the subject of establishing a research problem, R. Lyle Skains offers an explanation for the difficulty in identifying knowledge gaps in practice-based research that is in line with the above:

> It can be difficult to identify gaps [in knowledge] when the researcher is engaged in an entirely new area or creative endeavor, as a basic level of knowledge and experience is required to, in essence, know what it is we do not yet know. [Practice-based research] is often a process of exploration and discovery, with many key insights arriving via serendipity, rather than as part of experiment design. [Ska18, p. 93]

In the field of computer music, whenever the goal is to produce musical improvisation based on human-

machine interaction, it is fairly common for the creative process to begin with the conception of a new system. Richard Dudas makes this observation in his article entitled "Comprovisation[4]: The various facets of composed improvisation within interactive performance systems" [Dud10].

> Very often when working with technology, it is the instrument[5] that must first be composed in order to have performance, and consequently, improvisation. A musical instrument, whether acoustic or electronic, can be defined as "a self-contained and autonomous sound-producing object that enables a musician to perform in a live situation" (citing Atau Tanaka). It is therefore the job of the electronic/computer musician to design and "compose" a rich computer music performance system. Such a system should not be designed to perform one lone task, as with a tool, but should be designed to evolve or metamorphose in the hands of a competent performer (...) [Dud10, p. 30]

It can thus be argued that the act of designing a new system is compositional in nature, given that every decision made regarding the development of its features will have a direct impact on the aesthetic outcome that is enabled. This applies to systems that use rule-based algorithms as well as those that use machine learning techniques. Moreover, according to the author, there is a compositional structure embedded in such a system (or instrument) which is used to define the musical progression of a work, even if the work is an improvisation [Dud10, p. 30]. In this manner, the author makes a connection between system design, composition and luthierie as well.

However, unlike the set of interactive music systems considered by the author, `Comprovisador` does not produce sound on its own – nor is it self-contained, as it entails a group of musicians who read the output of the system (a multi-part network-distributed score), converting their own reading (interpretation) into sound. Consequently, although `Comprovisador` permits a musician with a distinctive function (mediation) to perform on it in a live setting, it cannot be classed as an instrument.

It is of note that there exists a set of systems that do not conform to Rowe's **instrument paradigm** but rather to his **player paradigm**[6]. The key concept behind these systems is to have an artificial player interacting (often in a conversational form) with a human who plays a conventional instrument. This concept is present in `Comprovisador`, albeit with a key distinction: rather than an artificial, sound-producing player, `Comprovisador` can be regarded as an artificial, live-score-generating composer.

[4] Dudas' use of the term Comprovisation will be examined in Section 2.2

[5] In the original context, instrument is synonym of computer music interactive performance system – cf. Rowe's taxonomy: instrument paradigm (see Section 2.3.2).

[6] See Section 2.3.2 for the full classification system.

Nevertheless, it is important to reflect on the last sentence of the quotation above where Dudas calls for a high degree of complexity in the development of a rich computer music performance system. In fact, venturing into such an endeavour, I had to realise that much of my creative energy would go into programming, testing and debugging the system before it could go into actual music making. And since my training is not in computer science but rather in composition and improvisation (albeit with a technological background), that energy would also go into learning and investigating solutions for ordinary problems. These are not research problems but rather practical ones; notwithstanding, it is not uncommon for deeper problems to be uncovered in pursuit of a simple problem's solution.

Despite the admitted difficulty in the early identification of gaps in knowledge (inherent in practice-based research), it was possible to foresee a number of relevant problems from the start of the project. These were linked to the intent of integrating sight-reading in a real-time generative context[7], encompassing the following: 1) likelihood of sight-reading errors, 2) difficulty in synchronising musicians with regards to unpredicted, unpractised and unconducted scores, and 3) various delays caused by processes of computational, physical and cognitive nature.

Sight-reading is a highly specialized task, not truly real-time, in which *error* is always eminent and, as consequence, *anxiety* is always present. In order to employ sight-reading in a performance context, a delay or a special way of dealing with time may be implemented so as to mitigate these downsides. However, such a solution will have an impact on the aesthetic outcome and thus must be weighed. On the other hand, as shall be detailed in several parts of this dissertation, inserting and controlling a functional delay acting as a buffer for sight-reading – which I call Reading Time Window – is key to reduce error probability while enabling strategies for synchronisation and for managing intrinsic delays (by encapsulating them).

Instead of aiming to mitigate the likelihood of sight-reading errors, another solution is to adapt the aesthetic conception in such a way as to assimilate those errors, making them unnoticeable for listeners and inconsequential for musicians with regard to their stress levels. Examples of aesthetic adaptation include the use of high degrees of dissonance and/or entropy as a way of masking errors. Graphic notation can also be used enabling a situation where the very notion of error is, in many cases, challenged. These solutions are not mutually exclusive, meaning it is possible to have error risk mitigation and aesthetic adaptation strategies running concurrently and symbiotically.

Regarding the various delays that have the potential to impact synchronisation and/or the aesthetics of the

[7] See fourth subsidiary question and its sub-questions in Section 1.1

whole, the most critical one results from the cognitive processes entailed in sight-reading. When sight-reading a score, a musician's eyes fixate points in the score that are ahead of the music being played. This is known as the eye-hand span (or the eye-voice span in the case of singers) and can be defined as the amount of material, measured in number of notes, that could be correctly played following a certain note if the score was to disappear when that note was played [Slo74]. The eye-hand span is correlated with sight-reading proficiency, with the more skilled sight-readers showing larger spans [Slo74].

Eye-hand span can be interpreted as a memory buffer, by analogy with some computational processes. A buffer might be necessary to ensure a steady flow of data to a rendering algorithm in a situation where the acquisition of data may be fluctuating or darting. Just as larger eye-hand spans correlate to skilful sight-reading, larger buffers enable steadier data flow and thus less error-prone rendering. It comes with a cost, though: larger buffers mean greater latency. In real-time applications (such as real-time audio processing), this can be a problem – one that is usually resolved by reaching a compromise between stability (output quality) and low latency.

Besides varying considerably from individual to individual, eye-hand span also varies for each individual according to the degree of difficulty of a musical passage within a score. With static scores, each musician performs the necessary adjustments in span almost unmindfully. It should be noted that, in this context, musicians usually have the option of doing a quick visual inspection of the sheet music before beginning to play. By allocating some inspection time to identifying (and possibly even solving) eventual reading problems in advance, this option allows some level of optimisation of the eye-hand span at runtime.

With dynamic, live-generated scores, one major problem is that musicians do not have that option. In `Comprovisador`'s case, the idea of establishing a reading time window (suggested above) will require a preset amount of delay which may or may not be adequate for every musician's needs and for every musical situation. This can be problematic because a given reading time window size may lead to one of the following: 1) the window size (or delay amount) may be insufficient to accommodate a musician's typical eye-hand span; 2) it may be unsuited for a complex musical situation; or 3) it may be too large, resulting in unnecessary breaks in the musical discourse. In point of fact, different amounts of delay will impact the aesthetics in different ways.

`Comprovisador` can be considered a Networked Music Performance (NMP) system in that it relies on a local area network (LAN)[8] to enable distributed computing of real-time notation (RTN), prompting an ensemble of musicians to sight-read it. It is worth noting, however, that a large portion of NMP systems do not use any notation tools and instead rely on network streaming of audio, often over the Internet. LAN performances,

[8]Advancements have been made meanwhile in order to enable remote comprovisation performances, in the near future, using a wide area network (WAN) – the Internet.

notably when using wireless connectivity, can be carried out in non-standard space settings, as long as musicians are within reach of the network's Wi-Fi signal (or Ethernet cable), which usually means being in the same building. Generally in this situation, musicians and audience can hear the direct acoustic sound from other musicians, hence not requiring audio streaming.

NMP practitioners must often deal with synchronisation problems arising from processes of computational and physical nature. They must cope with network-induced latency (significantly aggravated with distances greater than 200 kilometres; also, when using Wi-Fi), acoustic delay (meaning sound propagation time, which becomes significant in local performances comprising spaces greater than 20 metres in length), hampered eye contact (by physical distance and/or architectural barriers), to name only the most significant. In NMP systems that employ RTN, as is the case with `Comprovisador`, this problems add up to the mentioned shortcomings of the sight-reading cognitive processes, making the issue of synchronisation a crucial one. Moreover, in a context of *actual* RTN (meaning live-generated and not just live-animated) it is virtually impossible to predict what comes next in the music (as one does in order to efficiently sight-read a piece of written repertoire, based on acquired musical grammar), let alone practice the score.

Such an "extreme sight-reading" context (a term coined by Jason Freeman [Fre08]) may be seen as problematic since it compromises the degree of complexity in a given musical phrase that a performer might consider manageable. This is so at least as far as staff-based notation is concerned and especially if various musical parameters are notated. One may fear that keeping the difficulty level always very low might yield dull and uninteresting results. Nevertheless, as Seth Shafer remarks:

> A wide breadth of creative work lies between the extremes of notational vacuum and parameter overload, with composers often attempting to balance one difficult parameter by making the other remaining parameters correspondingly easier. [Sha17a, p. 3]

Following Shafer's observation, a sensible way to manage complexity involves trading off the difficulty of different parameters, ensuring that the difficulty of several parameters does not increase at the same time. This should be a key consideration when developing a notation interface for a live-notation scenario.

Amongst the fields of NMP and RTN, in order to mask synchronisation gaps so they are not perceived by the listener, there is a tendency to opt for slow-attack sounds (e.g. Barbosa [Bar03, BCG05, BC11]), *pointillism* and long, step-free sounds (e.g. Cat Hope [HVWJ15, HWT18], whose colourful graphic scores often feature wavy

lines notating very long sounds rich in *glissandi*). A different approach is that used by Eingefeldt in his piece "An Unnatural Selection: Mvt 2 *Much Beauty is Before You*" for an ensemble of four musicians and generative animated notation, distributed in a LAN, presented at the Sound and Music Computing conference (SMC), in 2016. Here, a human conductor is employed, enabling the use of synchronised rhythms. From my perspective, either of these options represent aesthetic constraints: lines and points lead to motionlessness and entropy respectively, while Eingefeldt's option works better within a framework of regular meter.

In my research, I shall try and find ways to attenuate these constraints, working towards achieving a continuum of aesthetic possibilities, regarding rhythm and articulation, using staff-based notation to control harmony and contour. As noted above, by implementing a *reading time window* I seek to mitigate the processes that affect synchronisation, thus reducing those constraints – certainly uncovering others along the way, which will have to be addressed.

1.2.1 Opportunities

By addressing the problems stated above, this research should provide converse opportunities. More so, the adopted practice-based research methodology should help bring to light unforeseen problems and, sometimes via serendipity but always grounded in reflective analysis, encounter appropriate solutions. Here is a list of the most significant opportunities envisaged by this research:

- the conception of a flexible notation interface, capable of several specific real-time reading modes, each focusing on a unique strategy for incorporating the likelihood of sight-reading errors as an aesthetic feature (rather than a fault);
- the elaboration of a systematic approach to a reading time window, which shall enable a high rate of error mitigation during the sight-reading of real-time scores;
- the development of a graphical cueing strategy (bouncing ball), which will facilitate both score navigation and synchronisation, even when musicians are tens of meters apart and/or when their vision is blocked by architectural barriers;
- the development of an audio cueing strategy (adaptable metronome), which will further enhance synchronisation, thanks to the faster response of the auditory system, over the visual one; the designation "adaptable" implies that the metronome is able to adapt to situations of flexible tempo, unmetered rhythm or complex subdivision transitions;

- the development of another auditory cueing strategy (audio score), designed to aid singers in the process of fast pitch prediction and correct intonation, with special merit in microtonal contexts;
- the conception and implementation of a set of efficient constraint solvers (titled polifonadores) capable of providing, in real time, solutions for chord playability, with polyphonic instruments;
- and, ultimately, the creation of a new software system, which will enable the realisation of comprovisation performances where a conductor/composer is able to mediate the interaction between a solo improviser and an ensemble of sight-readers, and where complex interaction flow paths are enabled.

As mentioned above, I will also envisage the broadening of the system's application, adapting it as a tool for music students, for the development of sight-reading skills, accounting for different learning stages, ranging from the first few playable notes to advanced microtonal contexts. The same adaptation (a stochastic score generator) will also enable the system to function as a generator of pre-compositional material to use in fixed compositions.

On the subject of creative practice, there will be room for experimentation with other possibilities such as the use of augmented instruments to control expressive parameters, letting the soloist do some of the mediator's tasks for the benefit of interaction flow, aiming at a more consequent interplay (e.g. "Comprovisação nº 9", with Henrique Portovedo's Hybrid Augmented Saxophone of Gestural Symbiosis (HASGS) [Por19]). Another creative possibility I shall contemplate is to distribute precomposed scores over a network, where the affordance of using `Comprovisador` would be the ability to synchronise musicians on very complex situations (e.g. having multiple metronomic tempi and/or having musicians separated across a very large concert hall).

To conclude, the crucial role of anticipated issues and potential opportunities, whether foreseen or unforeseen, in this practice-based research should not be overlooked. They materialise from the research questions put forward in Section 1.1, especially the following two:

- What choices in **algorithm conception** and **notation interface design** can be made to facilitate and integrate the sight-reading process in a real-time generative context?
- What **novel approaches** could be introduced to enable new and desirable aesthetic outcomes thus leading to advancements in the field?

My primary research contribution arises from these two key inquiries and comprises reflection on a range

of potential solutions that require implementation and testing within various performance scenarios. The following list outlines the exploratory avenues that will be pursued in order to identify potential solutions.

- Expansion of aesthetic possibilities that embrace and/or mitigate the likelihood of sight-reading errors;
- Score navigation and ensemble synchronisation, supported by concurrent visual and auditory stimuli, facilitating the use of space as a compositional element and augmenting the range of rhythmic possibilities;
- Rhythm and articulation possibilities that go beyond the prevailing aesthetics of "lines and points", making sharp synchronised attacks a viable option, in metered and unmetered (flexible) rhythmic contexts;
- The inclusion of singers, ensuring correct intonation in various tuning systems, alongside producing appropriate choral timbre through stochastic generation of phoneme-based sung text;
- Idiomatically relevant use of chords (or double stops) according to instrument type;
- Educational use of generative algorithms for sight-reading enhancement;
- Comprovisation performances shaped from a *concertino-ripieno* interaction approach drawing from various practices inside and outside the computer music field (see Fig. 2.2).

1.3 Overview

This document comprises seven chapters and six appendices. Chapter 1 introduces the research question, outlining anticipated problems that this study aims to address. In Chapter 2, a broad contextualisation is provided, covering essential concepts, including the definition of Comprovisation. An examination of prevailing trends in Distributed Performance and Live Scores is also presented. Chapter 3 presents the `Comprovisador` system and its constituent parts while addressing the aesthetic goals that underlie its design. An account of the various steps in the iterative development model of `Comprovisador` can be found in Chapter 4. Following is Chapter 5, where the **modes of operation** developed and studied within the framework of **musical performance practice** are discussed. Chapter 6 explores the application of `Comprovisador`'s `Practice Mode` in educational scenarios, specifically in enhancing sight-reading abilities. Finally, Chapter 7 concludes the discussion, while indicating future directions. In the six appendices, relevant published work or submitted projects are included as support material.

Throughout the text, there are hyperlinks to short **Video Examples** to further illustrate both the behaviour of the dynamic score and the musical outcomes obtained in live performances. Each example will start and stop automatically at the correct time. Also, there is a replay button for convenience. It may happen that a Security Warning alert, like the one shown in Figure 1.1, appears the first time a hyperlink is clicked. If this occurs, select "Allow".

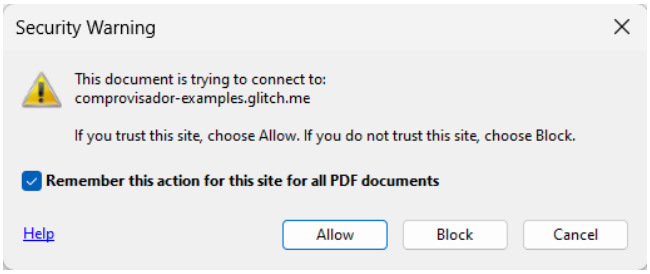


Figure 1.1: A Security Warning that may appear the first time a hyperlink is clicked – select "Allow".

A list of all video examples contained herein is provided via the following URL:
`https://comprovisador.wordpress.com/thesis-examples/` [Lou17a].

Audio/video recordings of most of `Comprovisador`'s public performances can be accessed through the project's website:
`https://comprovisador.wordpress.com/` [Lou17a]

1.3.1 Published Research

This dissertation includes material from the following items which were written and published during the course of the present research:

- Pedro Louzeiro. Real-time compositional procedures for mediated soloist-ensemble interaction: The `Comprovisador`. In Octavio A. Agustín-Aquino, Emilio Lluis-Puebla, and Mariana Montiel, editors, *Mathematics and Computation in Music: 6th International Conference, MCM 2017, Mexico City, Mexico*, pages 117–131. Springer International Publishing, Cham, 2017. [Lou17e]
- Pedro Louzeiro. The `Comprovisador`'s real-time notation interface. In *Proceedings of the 13th International Symposium on Computer Music Multidisciplinary Research*, pages 340–351, Matosinhos, 2017. [Lou17b]
- Pedro Louzeiro. Mediating a comprovisation performance: the `Comprovisador`'s control interface. In *Proceedings of the 43rd International Computer Music Conference*, pages 362–367, Shanghai, 2017. [Lou17d]

- Pedro Louzeiro. Improving sight-reading skills through dynamic notation – the case of **Comprovisador**. In Sandeep Bhagwati and Jean Bresson, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, pages 55–61, Montreal, Canada, 2018. Concordia University. [Lou18d]

- Pedro Louzeiro. The **Comprovisador**'s real-time notation interface (extended version). In Mitsuko Aramaki, Matthew E. P. Davies, Richard Kronland-Martinet and Sølvi Ystad, editors, *Music Technology with Swing*, pages 489-508. Springer International Publishing, Cham, 2018. [Lou18c]

- Pedro Louzeiro. Synchronizing to visual cues in a networked, real-time notation environment – **Comprovisador**. In Isabel Soveral and Fátima Pombo, editors, *Synchresis – Audio Vision Tales*, pages 165–172. UA Editora, Aveiro, 2019. [Lou19c]

- Pedro Louzeiro. Distributed Scores and Audio on Mobile Devices in the Music for a Multidisciplinary Performance. In *Proceedings of the 14th International Symposium on Computer Music Multidisciplinary Research*, pages 401–412, Marseille, 2019. [Lou19a]

- Pedro Louzeiro. Distributed Scores and Audio on Mobile Devices in the Music for a Multidisciplinary Performance. In Richard Kronland-Martinet, Sølvi Ystad and Mitsuko Aramaki, editors, *Perception, Representations, Image, Sound, Music*, pages 329–344. Springer International Publishing, Cham, 2021. [Lou21]

Moreover, research results were presented at the following conferences:

- Pedro Louzeiro. Improving sight-reading skills in a microtonal context using a real-time composition tool, July 2017. Presented at The 2nd European Saxophone Congress (EurSax'17). [Lou17c]

- Pedro Louzeiro. Synchronizing to visual cues in a networked, real-time notation environment – **Comprovisador**, November 2017. Presented at the International Conference Electroacoustic Winds 2017: SYNCHRESIS – Audio Vision Tales. [Lou17f]

- Pedro Louzeiro. Aspectos e potencialidades de um sistema de composição e notação em tempo-real – Comprovisador, April 2018. Presented at Conferências em Música e Musicologia do Centro de Estudos de Sociologia e Estética Musical – Pólo Universidade de Évora [Lou18a]

- Pedro Louzeiro. Aspects and potentialities of a real-time composition and notation system – Comprovisador, November 2018. Presented at Nova Contemporary Music Meeting – International Conference. [Lou18b]

- Pedro Louzeiro. Música Distribuída, May 2019. Presented at Semana da Composição 2019 – Escola Superior de Música de Lisboa. [Lou19b]

1.3.2 Scope and Methodology

Although this thesis is being submitted to achieve a Doctorate in Music and Musicology, with specialisation in Composition, a significant portion of the work presented here pertains to the field of Computer Music. Hence, it would be impractical to keep the focus of the discussion constantly on the musical side. Notwithstanding, efforts will be made so that a reader who is not an expert in computer science may still follow the discussed matters.

The nature of creative practice necessitates a certain level of subjectivity in research. It is impossible for an artist to remain completely objective when examining their own work. However, by reflecting on their practice, an artist can provide unique insights into specific aspects of their work. It is important to note, as maintained by Skains [Ska18], that reflection can be an imprecise technique as it relies on memory. R. Lyle Skains, a practitioner-researcher in the field of creative digital writing, makes a valuable recommendation on methodology, aiming at the memory issue:

> (...) I call for the employment of a self-directed form of ethnomethodology during the composition of the texts in the form of a research log (noting insights, process, difficulties), and draft materials and revision notes (which can later be analyzed as *in situ* utterances). [Ska18]

The methodology proposed by the author is indeed compatible with the iterative model adopted for the development of `Comprovisador`. Although Skain's research is primarily concerned with the cognitive processes underlying creative production, whereas my own is more focused on the technical processes and on the creative outcomes themselves, I nevertheless believe that there is merit in applying his proposed methodology to my own research.

2

Context

The `Comprovisador` system can be framed at the intersection of three distinct computer music performance spaces: **interactive music systems** (IMS), **networked music performance** (NMP) and **animated notation** (see Figure 2.1). The link between animated notation and NMP is a prevalent one since it is common for animated notation systems to distribute and synchronise individual score parts over networked computers, tablets or other devices. Such systems are often called score players as their role is to play back previously composed scores without any real-time changes (e.g. SmartVox [Bel18, BC19] and Decibel ScorePlayer [HVWJ15]). There are, of course, animated scores that do not require interactive or network features. Such scores may consist of animation video files which are played back from a single big screen as a score for the whole ensemble (e.g. Ryan Ross Smith's "Studies" – `http://ryanrosssmith.com/animatednotation.html`).

Systems that generate real-time scores (algorithmically and/or interactively) without network features populate

the space that is shared between animated notation and IMS (e.g. Nick Didkowsky's piece "Zero Waste" [WDH18] and Nicolas Collins' piece "Roomtone Variations" [Col13]). In this space, systems employ single screens displaying full scores (or solo scores if that is the case) which can be sight-read by one or few musicians (e.g. using a large computer monitor) or by an ensemble (e.g. using video projection) – a practice reminiscent of renaissance's *cantare super librum* not only because of the way musicians must gather around a single large score but also because improvisation is often part of the aesthetics.

Regarding the space exclusively shared between IMS and NMP, it accommodates systems that stream audio signals and/or control data across different nodes enabling local as well as remote generation and/or processing of signals. This can be done using various approaches and interactive topologies which are well documented in the literature (see [Bar03, Wei05, MSB19]). I shall not focus on this space as it does not regard animated notation – a key concept in my research.

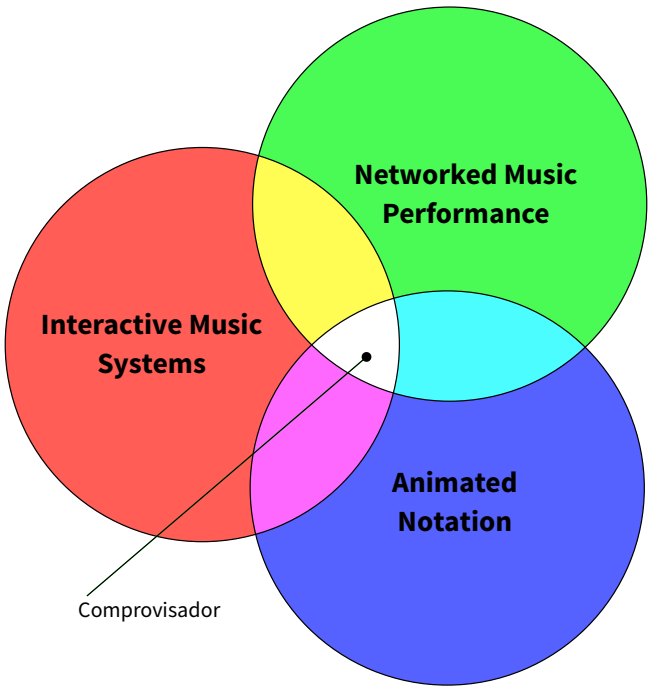


Figure 2.1: Three intersecting fields of computer music – space occupied by `Comprovisador`.

`Comprovisador` can be fitted in the intersected space because it accepts an improvised performance as input for algorithms that generate a real-time composed response – as is typical of a prevalent type of *interactive music system* (see Section 2.3.2) – but, whereas traditional IMS always output electronic or electro-acoustic sounds, `Comprovisador` outputs live-generated score parts (*animated notation*) which are distributed and synchronised across computer screens on a local area *network*.

It should be noted that other systems also exist at this intersection, albeit derived from distinct paradigms

within each of the three intersecting fields. One particularly relevant example is Quintet.net [Haj05, Haj08, HD09, JHV+17], an environment created by Georg Hajdu for composing and performing music on the internet. Quintet.net utilises the streaming of control data[1] (which is rendered as audio in real time on the client side, via virtual instruments), as it is more oriented towards performances over a WAN, whereas `Comprovisador` is more focused on LAN performances and therefore does not require (at the time of writing) the streaming of either control data or audio. In comparison to `Comprovisador`, which is a system formalised around a concrete model of interaction, Hajdu's software is an open environment for musical collaboration over a network. It supports both traditional and graphic notation, facilitating the integration of composed and improvised elements. Furthermore, both types of notation can be either fixed or live-generated. It is important to note, that the software is capable of processing musical streams algorithmically and incorporates custom-designed software patches for instrumental playback. Therefore, it seems fair to conclude that Quintet.net is more aligned with Rowe's instrument paradigm, in contrast to `Comprovisador`, where the live-score generating algorithms are more closely associated with the player paradigm. It is also worthy of note that both systems require the presence of a conductor, who is able to make decisions in real time that affect the musical outcome. On the other hand, as its name suggests, Quintet.net is tailored for use with five nodes, while `Comprovisador` is designed to accommodate larger ensembles. This comparison aims to demonstrate that `Comprovisador` possesses a distinctive set of characteristics that warrant further examination.

Figure 2.2 presents a concept map showing connections within and beyond the three mentioned spaces. It indicates which representative concepts of those spaces are shared with `Comprovisador`. Also, it demonstrates connections between these and other three music creation spaces: (1) computer aided composition, (2) composition of paper scores that incorporate context-dependent elements and (3) gesture languages for conducted improvisation or live composition.

Acknowledging the existence of such connections is important to understand the influence each field had on the development of `Comprovisador` and on the creative practice that is at the base of this research. Hence, it is important to clarify some key concepts that are present in this doctorate's main research question (Section 1.1) as well as in the concept map presented in Figure 2.2. To do so, I will look into the dualism inherent to **real-time composition**, analysing the concepts of composition and improvisation (Section 2.1). Subsequently, I will investigate the origins and usage of the term **comprovisation** according to three different fields (see Figure 2.2 – light-blue arrows) and discuss its definition (Section 2.2).

[1] As an alternative to streaming control data, there is the option of directly streaming audio using a system such as JackTrip [jac24, CC09].

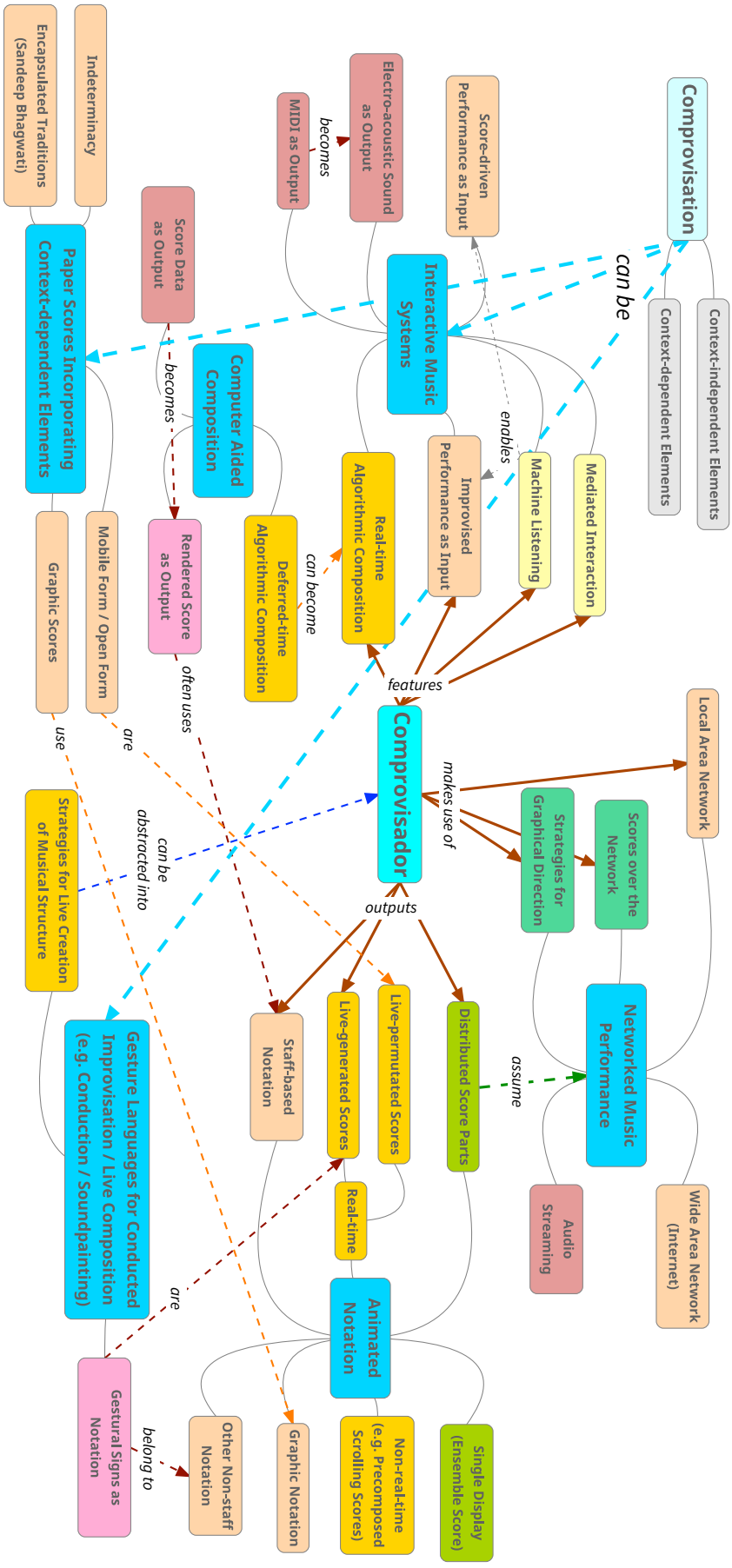


Figure 2.2: `Comprovisador`'s conceptual framework.

The various forms in which algorithms have been used in music constitute the background for the development of any novel system for music performance involving computers. An overview of those forms is pertinent and will therefore be made while indicating connections with `Comprovisador`. Furthermore, I shall examine how authors in the IMS field interpret the concept of **real-time composition** and how `Comprovisador` relates to that interpretation (Section 2.3).

Finally, I will explore the fields of **networked music performance** and **animated notation** focusing on the state of the art at the aforementioned intersection. I shall be able to demonstrate how `Comprovisador` addresses a specific creative space that is currently unoccupied (Section 2.4).

2.1 Composition versus Improvisation

Composition, according to the Oxford Dictionary, is "A creative work, especially a poem or piece of music" or it may be "The action or art of producing" such a creative work [oxf19]. **Improvisation**, according to the same source, refers to "The action of improvising" or to "Something that is improvised, in particular a piece of music, drama, etc. created spontaneously or without preparation" [oxf19]. Hence, given that a piece of music can be created through composition as well as through improvisation, it can be concluded that the main difference between the two actions is **preparation** or lack thereof. Nonetheless, musicians usually acknowledge the presence of preparation in the act of improvising and spontaneity in the act of composing. On this regard, Arnold Schoenberg has stated:

> [A]n improviser must anticipate before playing, and composing is a slowed-down improvisation; often one cannot write fast enough to keep up with the stream of ideas. [Sch50, p. 98]

Statements such as this may point to a distinction based on the **speed** at which each creative process takes place (deferred time – real time). Nonetheless, the fact that composing is an action carried out in deferred time allows a level of preparation inconceivable in improvisation. Improvisers must create musical materials while performing, i.e. in real time. Contrary to composers, they are not allowed the opportunity to reassess and rework their musical ideas before committing them to sound.

Consequently, composer and improviser experience time in a very distinct manner. Ed Sarath suggests that, for the improviser, "the present is heightened and the past and future are perceptually subordinated" [Sar96,

p. 1], while for the composer, “temporal projections may be conceived from any moment in a work to past and future time coordinates” [Sar96, p. 1]. He concludes that improvisation is driven by an *inner-directed temporality*, whereas composition is characterized by an *expanding temporality*. The contrasting directionality of the two **temporal conceptions** is indicated by the author as the main criteria for differentiating between improvisation and composition processes [Sar96].

The author considers a third category – *retensive-protensive temporality* – which involves the projection of awareness in past and future directions in a continuous framework. This conception may be invoked by improvisers when recalling past materials and implementing future-directed strategies. It can also be invoked when composers work out ideas through spontaneous performance, as part of a discontinuous process, or when creating a composition in a single real-time attempt, which the author calls *extemporaneous composition* [Sar96, p. 6].

Regarding the expanding temporality of the composer, Sarath writes:

> The composer may enter and freely traverse the past-present-future continuum of a work, assuming the vantage point of the future to review and possibly alter the past, or that of the past to view and rework the future. The temporality of the composer thus has both *cumulative* and *reversible* qualities, whereby relationships between events and their pasts and futures may be conceived [Sar96, p. 5].

Comparing the expanding temporality and its qualities with the inner-directed conception of the improviser where “the artist proceeds in a more moment-to-moment manner” [Sar96, p. 4], it can be concluded that the composer is allowed greater control over musical structure, not only on the horizontal axis, establishing a dramaturgy with connections of musical material over the time domain, but also on the vertical axis, through orchestration, synchronisation, polyphony and other techniques that require the composer to ’freeze’ time to work in a thorough manner.

Though it is true that a competent improviser, by invoking the retensive-protensive temporality, may use recurrence of relevant ideas or motifs throughout an improvisation, this is highly dependent upon memory, which is fallible and subject to all sorts of contingencies. Moreover, in practice, this can only be achieved in one time direction as there is no way to alter the past.

Also, in the context of free group improvisation, controlling the formal direction of a performance may be

challenging as each improviser will try to implement different future-directed strategies whose results may be incompatible (cf. [Gir18]), therefore, requiring adaptation. Consequently, retensive-protensive strategising is to a great extent inhibited in ensemble improvisation. As Sarath indicates, "the temporal awareness of ensemble improvisers tends to be directed to the vantage point most conducive to adaptive measures – the localized present" [Sar96, p. 5]. Hence, in this context dominated by the inner-directed conception, once a musical decision is made and a resulting event takes place becoming part of the unchangeable past, the only possibility is to take action in the localised present regarding the immediate future.

There is also the issue of **perception**. Can we as listeners discern between composed and improvised music? On that subject, Lehmann and Kopiez carried out a study where 102 college music students rated same-style compositions and non-compositions in order to test expert listeners' ability to discern between the two creative processes. The authors propose that " 'togetherness' and precision of an ensemble may indicate composition, while a higher degree of entropy could signal improvisation" [LK10, p. 579].

Here, it can be concluded that listeners tend to use vertical structure as a cue to recognise composition. That said, with a small group of free improvisers, it is possible to generate togetherness in a limited way (possible: synchronised attacks, repeated motifs; not possible: long/complex melodies or rhythms in unison) relying solely on spontaneous non-verbal communication – a task that becomes increasingly difficult with larger ensembles.

Still, one might ask: why is there not a reference in the study to cues regarding horizontal structure? Maybe it relates to the fact that, like the improviser, the listener relies upon memory to establish connections in the time domain. The method used in the study above relates to a real-life context where a listener experiences a musical work for the first time (e.g. a world premiere), yet there are situations where one listens to the same composition several times (e.g. listening to a favourite CD or attending a performance of Beethoven's fifth). In these situations, connections in the time domain become more apparent as long-term memory becomes engaged. In order to verify such a hypothesis, listeners would have to be subjected to a number of repetitions for each of the musical excerpts, preferably in different days.

However, the context of a world premiere is more akin to the kind of unrepeatable performance enabled by `Comprovisador` and thus more relevant to my research. Hence, I shall demonstrate how elements of vertical structure such as synchronisation, polyphony and orchestration were key to the design choices made (see Section 3.2).

There are, of course, other differences between both actions: composition often relies on notation whereas improvisation usually does not; a composed piece of music is, in most cases, a repeatable work whereas an

improvisation is ephemeral. Notwithstanding, these differences cannot be used as identifying criteria of one or the other form of creation. An improvisation can be recorded and therefore repeated, but it does not make it a composition – or does it? What if the recording is then transcribed and distributed as sheet music? Likewise, a piece can be composed using graphic notation[2] or textual instructions[3] where two performances of the same piece will sound inevitably different, hence, unrepeatable – can it still be considered composition? An improviser can use notation as a departing point for variation and proliferation – should it be called improvisation or rather interpretation?

When notation is involved, interpretation can often be viewed as a continuum between the strict rendering of prescribed pitches, durations and intensities and the freer act of improvising over a descriptive score. As an example, when interpreting baroque music, musicians are expected to execute ornaments which, in some cases, consist of a substantially larger amount of notes than the amount written in the score. At that point, it might be more appropriate to speak of improvisation – but where lies the boundary?

These questions serve the purpose of showing how difficult it is to find characteristics that unequivocally distinguish between composition and improvisation. Nevertheless, it remains true that composition, through the advantage of preparation, offers greater control over musical structure. Notation enhances this possibility, although, as implied above, notation is not a requirement for composition. Indeed, without ever using pen and paper or a computer, a composer may create a whole piece, memorising it while composing – often with the help of a musical instrument. I have used this approach myself, in the past. The piece can then be *aurally* transmitted and performed by a group[4]. However, notation undoubtedly facilitates transmission and rehearsing, particularly with large ensembles and sizeable compositions. More so, like an external memory addition to the composer's mind, notation enables the devising of musical structures of unparalleled complexity. It enables storage and iterative development of musical ideas, adopting all possible perspectives in the temporal dimension as well as in the general-to-particular continuum. Whether if an ultra-high level of complexity is a desirable quality is subject for another discussion.

Now, challenging a previous statement, I pose the following question: **can composition be carried out in real time?** One might argue that the two concepts contradict each other. My follow-up question would be: what if there is a real-time system (computational or otherwise) able to provide aurally perceivable structure in the vertical domain and possibly even in the time domain? Moreover, what if it incorporated a number of previously

[2] e.g.: Cornelius Cardew's "Treatise" [Car67].
[3] e.g.: Karlheinz Stockhausen's "Aus den sieben Tagen" [Sto68].
[4] In this context, it is important to remember that while not all musicians can read or write music, many can create it. Furthermore, not all traditions use music notation.

prepared elements while enabling essentially a real-time, context-dependent creation activity? I shall return to this question.

A reconciliation between the two concepts has been attempted in certain electronic and computer music practices. Joel Chadabe [Eig07], a pioneer in designing and performing with an analogue computer/synthesiser system, led an artistic practice which he called *interactive composition*, defining it as "a mutually influential relationship between performer and instrument" (as cited in [Eig07]). The word *interactive* entails the concept of a continuous framework where mutually influential relationships take place in real time. Therefore, it is intrinsic to improvisation rather than to composition. To address this contradiction in terms, Chadabe distinguishes interactive composition from instrumental improvisation as follows:

> [Interactive composition] is different from typical instrumental improvisation in that an improviser specifies and performs data in reaction to another improviser, but here the composer specifies and performs control strategies rather than data (...) The difference between instrumental improvisation and control strategy interaction is, then, primarily that of one's position in a control hierarchy, but it is true that in both cases the performer must act and react in realtime. [Cha77, p. 7]

In other words, an improviser engages with low-level musical elements whereas a composer manages high-level musical procedures. Chadabe's distinction uncovers a pertinent consideration: the level of abstraction of the processes controlled during performance may be key to the conception of real-time composition (RTC). Musical notes are low-level representations which can be abstracted into higher-level structures. A meta-level representation refers to the processes inherent to the creation of musical structures – the composition of the composition ([Gue18], citing Heinrich Taube). If a performer is controlling processes that generate high-level musical structures comprised of low-level elements, which is to say processes that constitute composition, then the performer is operating at a meta-level. Authors who explore RTC in some form often indicate the presence of meta-level representations [Row93, Ess95, Eig16, Bha17, BHE+18, Gue18] (see Section 2.3.2 – Real-Time Composition).

A reflection should also be made on Chadabe's assumption about instrumental improvisation. Instrumental improvisation requires embodiment, which requires education and training. Some improvisers consider themselves real-time composers and refer to embodiment as being a crucial aspect of the process. Bhagwati compares rule systems learned by improvisers – to the point of becoming embodied reflexes – to scores [Bha13a]. One can also compare them to algorithms, which are sets of instructions or rules that output musical structures. Thus, embodied reflexes are comparable to high-level representations of musical structures. Improvisers use

their 'algorithms' to be able to make musical decisions during the process of performing the music. Their decisions must not be based on a low note-by-note level – as that would not be efficient – but rather on a higher melodic-harmonic-rhythmic structure representation. It is true that the improviser has to execute all the notes – hence, his/her decisions entail the specification and performance of low-level data, as proposed by Chadabe. But I would argue, based on my experience as an improviser, that those lower-level decisions occur at an equally lower level of awareness. Here, I am making the assumption that there is some level of abstraction in improvisation – but is it meta? Perhaps not to the point enabled by interactive composition or RTC systems.

Returning to the question of including previously prepared elements in real-time context-dependent creation, here are some examples and their implications on the balance between composition and improvisation:

1. precomposed mobile passages (open-form);
2. precomposed harmonic-metric frameworks (e.g. jazz, many traditional music forms);
3. preprogrammed meta-level processes for musical structure generation upon input from an improviser (e.g. player-paradigm IMS, `Comprovisador`);

In the first example, there is a juxtaposition of elements in which the order is defined in real time and dependent on contingencies (via chance or decision making). One can also speak of juxtaposition of composition (in the form of previously prepared elements) and improvisation (referring to the processes guiding the real-time arrangement of those elements), although there is no doubt that the composition process is accomplished in deferred time. The second example is characterised by a superposition of layers: the composed framework over which the improvisation occurs. There is an issue of balance between the two, meaning each part can be more or less present in the actual musical rendition. To appreciate the balance depth, one may consider two extreme situations: a symphony orchestra playing a rich accompaniment to a soloist playing a form-restricted improvisation and a jazz quartet playing a standard tune in an exploratory way, going in and out of the framework. In the latter situation, it might be less clear what the composed part is – at least, from the listener's perspective – whereas in the former, it is usually clearer. But again, the composed part is unquestionably pre-existent. In the third example, the prepared elements consist in meta-composition specifications – not actual composition. The composition process is dependent on the improvisation, thus taking place in real time. Low-level data specification occurs only in the moment the input (improvised material) is processed.

It can be concluded that the concept of real-time composition is indeed valid. Still, there are some remaining problems with the nomenclature (cf. [Gue18]). The fact that the process is carried out in real time prohibits

the composer's expanding temporality Sarath proposes [Sar96] essentially because it is impossible to alter the past. By that logic, it should not be named composition. On the other hand, if context-independent (prepared) elements are used, the term improvisation might not be adequate either. It might still be possible to evoke the retensive-protensive temporality for the reason that decision making is often centralised (in a conductor-composer or in an algorithmic system) and, therefore, less susceptible to incompatibility of future-directed strategies as well as memory-driven recapitulations. That being said, every musical interaction creates a feedback loop where the decision maker is influenced by the output of the ensemble or system (except in the case of a cold, unsupervised generative system with no external input – ergo, non-interactive). This is the reason why these types of interaction create dialectic relationships. I would argue that this centralised decision making, with its intrinsic retensive-protensive temporality, places this musical context somewhere along the boundaries between composition and improvisation.

2.2 Comprovisation

To define the artistic practice under focus in this research, I have chosen the term *Comprovisation* as it suggests an amalgamation of the concepts of composition and improvisation. Several authors and musicians[5] have used this term to define approaches or practices that, in different ways, encompass elements from both fields. Below, the most relevant uses of the term are reviewed.

Conducted Improvisation

In the early eighties, Lawrence D. "Butch" Morris developed a system of gestures and signs for conducting an improvisation, being able to produce musical structure in real time. According to Thomas Stanley, "[f]or a time [Morris] had called his system *Comprovisation* (...)" [Sta09, p. 59]. Later, he coined (and trademarked) the term *Conduction* (short for conducted improvisation), which he defined as follows:

> Conduction®: The practice of conveying and interpreting a lexicon of directives to construct or modify sonic arrangement or composition; a structure-content exchange between composer/conductor and instrumentalists that provides the immediate possibility of initiating or altering harmony,

[5] A quick search in the discography database platforms Discogs [dis19] and AllMusic [all19] reveals about a dozen titles containing *comprovisation*. The earliest examples are from 1981 – Regan Ryzuk's *Trio Comprovisations* and *Fusion Quartet Comprovisations*. It is worth mentioning one track from "Butch" Morris entitled *The Bartók Comprovisation* and one CD from pianist Philip Thomas – *Comprovisation* – including works from John Cage, among others.

> melody, rhythm, tempo, progression, articulation, phrasing or form through the manipulation of pitch, dynamics (volume/intensity/density), timbre, duration, silence, and organisation [(order)] in real-time. [Mor]

This immediate possibility of initiating or altering musical features through manipulation of parameters is a common link between Morris' system and **Comprovisador**. Morris also referred to conduction as "an improvised duet for ensemble and conductor" [Rat13], which resonates, despite the obvious differences, with **Comprovisador**'s goal of enabling a conductor/composer to mediate the interaction between a solo improviser and an ensemble of sight-readers. The principal difference is the fact that in Morris' system all ensemble musicians are improvisers while in my system all ensemble musicians (not including soloist(s)) are classically trained and/or have excellent sight-reading skills: in Conduction, musicians must interpret the conductor's gestures and deliver corresponding musical actions; in **Comprovisador**, they must translate a generated score into sound.

Now, one can argue that the verbs to interpret and to translate are both synonyms of to read. In a way, Morris' gestures and signs are a score that he generates in real time and that the improvisers must follow. This relates to the discussion about the blurred boundaries between interpretation and improvisation. Another possible metaphor is the one given by Carlos Guedes while entertaining the possibility of real-time composition outside of computer music: "One can work with a group of musicians as if they were 'generative algorithms' and try to 'manipulate' their content generation by interacting with them through specific instructions." [Gue18]

Between 1985 and 2011, Morris carried out a total of 199 Conduction performances, all of them sequentially numbered [Mor]. He died in 2013. I never had the chance to meet him or attend one of his performances. I did, however, have the opportunity to attend a workshop of conducted improvisation guided by Olivier Benoît[6], in 2004. Moreover, I should point out my recent experience as a performer with the "Lisbon Soundpainting Orchestra"[7], conducted by François Choiselat. Through these experiences, I gained insight on a number of possible ways an improvisation could be structured in real time, from local action to global form, through conducting signs. This insight has proved relevant in the conceptualisation of **Comprovisador**'s algorithms and notation features, as shall be discussed in Chapters 3 and 5.

[6] Olivier Benoît was the director of "La Pieuvre" – an orchestra specialised in collective conducted improvisation, with links to the tradition of Butch Morris and John Zorn, among others [Ben99].

[7] Soundpainting is a "multidisciplinary live composing sign language for musicians, actors, dancers, and visual artists", created by Walter Thompson in 1974 [sou, Tho06]. The versatility of the Soundpainting language is remarkable: recently, it was tested as a device to control and direct a swarm of autonomous flying drones [CBC+18].

Computer Music Practice

In 2010, Richard Dudas wrote an article proposing the term *Comprovisation* in reference to "the balance between composition and improvisation with respect to interactive performance using electronic and computer-based music systems" [Dud10]. He considers two essential types of composition-improvisation relationships which are integral to electronic and computer music practice: "(1) composing an 'instrument' that can be improvised upon in performance, and (2) improvising with tools in order to create pre-compositional material" [Dud10]. Here, as mentioned in Section 1.2, composing an 'instrument' means designing and defining an interactive musical system – one that follows Rowe's *instrument paradigm* – where Dudas established a meaningful link between the process of composition and that of luthierie. Type two is associated with studio work and the process of exploration that leads to the generation of musical material to be used in a composition[8]. Yet, this process of instrumental exploration in search for musical ideas is not exclusive to the field of electronic and computer music, seeing that throughout History many composers of acoustic music have employed it. Likewise, there is a connection between type one relationships and a kind of musical work for acoustic instruments where the score demands that spontaneous elements be added during performance, although without the link between composition and luthierie. In this kind of work, the compositional structure is, of course, embedded in the score, designed in a way to allow the music to 'evolve or metamorphose' in different ways, according to the improvisational skills of 'a competent performer' (to use Dudas' words). Examples of such scores may be found in the work of composers like Earl Brown and Christian Wolff since the nineteen fifties. Moreover, it bears repeating that most of the music from the baroque period required performers to improvise ornamentation.

Dudas also admits the possibility of incorporating composed electronic material within an improvisational setting [Dud10] and he refers that computer musician George Lewis finds the coexistence of composed and improvised elements one of the most exciting aspects of including technology in music performance [Dud10, Roa85]. In my opinion, this coexistence of prepared and spontaneous types of music material is more suggestive of the term *comprovisation* than the two types of relationship the author focuses on. Moreover, the fact that this conception of *comprovisation* does not entail the use of technology is noteworthy. This is not to belittle technology and its affordances in this context, which in my view are substantial, but rather to approach a definition of *comprovisation* in an unbiased manner.

In light of this conception, I have developed within **`Comprovisador`** the possibility of deploying precomposed material[9] during a performance, in open-form style – one of the latest implementations to the system. Compared

[8] An example where **`Comprovisador`** is used in this type of creative process is given in Section 6.3
[9] In the form of staff-based dynamic score, not electro-acoustic sound – see Section 2.3.2

to a traditional open-form score for multiple players (where the order of the different sections would have to be sorted prior to a performance – which would defy the openness of the form – or otherwise pose problems of practicality), the main affordance of a live-score system consists in the ability to dynamically display a randomly selected score passage to all musicians at once.

This being said, **Comprovisador**'s primary design goal is not to include precomposed material in an improvised performance but rather to run algorithmic procedures on improvised material and deploy the compositional results in real time, thus establishing a dialectic relationship between improvisation and composed response. This is a type of composition-improvisation relationship (real-time application of compositional procedures on improvised material) that Dudas does not address in his article, despite it being an essential feature of many interactive music systems – namely those that conform to Rowe's *player paradigm*[10] (see Section 2.3.2). While it is true that algorithms of such systems had to be "composed" first (meta-composition), which is in agreement with Dudas' type one relationships, the real-time application of those algorithms on spontaneously created material is arguably a unique type of composition-improvisation relationship enabled by digital technology that deserves mention. On the other hand, one needs to accept the premiss of real-time composition in order to validate this possibility.

Ever since Dudas first used the term *comprovisation* in the Computer Music literature, some authors have employed it inseparably linked to interactive computer technologies – which, as I have tried to demonstrate, is an acceptation of the term that is biased and lacking. For example, Joshua Mailman writes:

> [S]ince the 1980s, but even more so recently, some technologically inclined musicians have been developing ways to 'improvise' complexity, by exploiting stochastic compositional methods pioneered originally by Iannis Xenakis in the 1950s, but now expanded and implemented with live interactive generative hardware-software technology. This has been called interactive composing or, since 2010, comprovisation, which is the use of technology to generate complexity that is spontaneously controllable. [Mai13, p. 357]

This author also defends that the concept of comprovisation can equally be applied to computer graphics generation and manipulation, especially when audio and graphics are coordinated systematically. In the article in question, he presents two "interactive dance systems that generate music and graphics spontaneously in response to hand and body movement" [Mai13].

[10] The interactivity flow path within **Comprovisador** is of greater complexity than that of traditional player-paradigm systems and will be subject of further discussion in the following chapters.

Despite the fact that his conception of comprovisation draws only from Chadabe's interactive composition (via Dudas), Mailman makes a good point in acknowledging the expanded possibilities of pairing stochastic compositional methods with interactive technology, in the generative art scene. In fact, rule-based stochastic procedures (see Section 2.3.1) have proved useful not only powering generative response methods but also transformative ones, where a real-time input is transformed to create variants (see Section 2.3.2). In this context, rule-based deterministic procedures and, more recently, machine learning (ML) techniques have also been applied (see Section 2.3.2).

Scores, Rule Systems and Contingencies

Composer Sandeep Bhagwati has used the term comprovisation beginning in 2004 to designate a composition practice he has led since 1996, consisting of complex, often cross-traditional scores for medium-to-large ensembles [Bha13c, p. 171]. Referring to comprovisation, he states:

> I use this term for all music that draws not only on the contingent moment of performance but also on context-independent rule systems or scores – and I believe that all music does both in some way. [Bha13a, p. 100]

The final remark asserts his conviction that no music ever is completely fixed (meaning that some elements will always depend on the particular moment of performance even in the most densely parametrised scores) as much as no improvisation ever is completely free (in the sense that it is inevitably bound to rule systems, embodied reflexes and inner representations that prompt a performer to play the next note) [Bha13a]. Hence, he equates improvisation rule systems with composition scores as being both context independent.

It is worthwhile noting that the author does not mention the use of technology within his explanation of the term. In his practice, he does often use computers and other technology in a very inventive way, as will be shown in Section 2.4.2, but he never refers to it as an inalienable feature of comprovisation. Nonetheless, in 2012, this connection is present in the title of an international research-creation workshop named "Comprovisations – Improvisation Technologies for the Performing Arts" [Bha12], organised by *matralab* – of which he is director at Concordia University in Montreal.

In order to better understand how he arrives at his definition of comprovisation (which will be discussed below), some important concepts inherent to his compositional (or better yet *comprovisational*) practice should be

examined. One such concept is that of *encapsulated traditions* [Bha13a, Bha13b]. These are "invented (i.e. composed) coherent rule-systems for improvisation that, like a tradition, function as generators for stylistically consistent improvisations" [Bha13b]. *Encapsulated* indicates that they are largely independent from each other in terms of structure, but are conceived to allow interaction with other such entities [Bha13a, p. 102].

His scores[11] consist of several of these entities which are layered to form *relational polyphony* and organised within the time domain to establish a dramaturgy. By *relational polyphony*, Bhagwati refers to a kind of polyphony where independent musical entities converge in a kind of aesthetical consilience, in contrast with *exegetic polyphony*, which concerns the unfolding of a central idea kernel [Bha13a, p. 101–102], [Bha13b]. On the process of rehearsing a score, he writes:

> Performers are given precise constraints and instructions for one or several encapsulated traditions, which they need to learn by heart and embody. These instructions can be very detailed, and are often illustrated by a written example of how a realisation could look (and sometimes even by an audio example). The musicians are not given explicit music, but rather blueprints on how to improvise within a certain framework. This demands a kind of personal practice very different from learning a written score on the one hand, but also very different from learning how to improvise on new material in one's own personal style. In fact, this approach to musical embodiment is closest to that of learning a North Indian *raag*. [Bha13a, p. 103]

With this technique, Bhagwati addresses the necessity of providing context-independent *communication standards* to large ensembles of improvising professional musicians (between 6 and 40) [Bha13c, p. 171].

Bhagwati gives an inclusive definition of comprovisation which attempts to acknowledge both oral, improvisatory traditions and the rich heritage of various traditions of written composition:

> [M]usical creation predicated on an aesthetically relevant interlocking of context-independent and contingent performance elements. [Bha13c, p. 171]

A simplified version, according to my own reading of his eloquent definition, would be:

> [A] musical performance context where both composed and improvised elements coexist in aes-

[11] Most of Bhagwati's comprovisation scores are available for download at **http://matralab.hexagram.ca/projects/#projectsSandeep** [Bha19].

> thetically relevant interdependency. [Lou18d, p. 2]

`Comprovisador` was indeed conceived as a tool to enable such musical performance contexts where solo improvisation and composed response are, in fact, interdependent: thanks to real-time composition algorithms, the composed response is highly dependent on incoming improvised material; and by virtue of a feedback loop, the improviser's decisions are affected by composed elements. One can say it forms a dialectical relationship, for a composed response could not exist without the improvisation and the improvisation could not be the same without the composed response. This interdependency is further extended by the presence of a mediator.

While conceiving a new comprovisation system founded upon real-time notation as well as aiming for the kind of interdependency noted above, aesthetic relevance should be the main concern when tailoring composition algorithms, choosing notation type and designing a notation interface. In Chapters 3 and 4, I shall address specific ways in which aesthetic relevance was considered during the development of `Comprovisador`.

Going back to the initial discussion, after having gained insight into ways of incorporating context-dependent elements into a composition (e.g. open form) and especially ways of structuring a group improvisation in real time (e.g. Soundpainting), I considered the adoption of paradigms that have flourished during the last two decades – namely, **animated notation** and **networked music performance** – as a means of bringing together improvisers and sight-readers in synergetic performances, benefiting from the 'realtimeness' of computational processes. This possibility implied the use of **machine listening techniques** as well as **algorithmic compositional procedures** in order to analyse the input from the soloist and to generate consequent musical responses. Both techniques are commonly found in music systems enabling human-machine interaction (HMI). Also, the idea of manipulating algorithmic parameters through a control interface came about and, with it, the figure of a performance mediator.

Finally, I have conceptualised the computer system under study in this doctorate – `Comprovisador` – which encompasses these features. In the sections below, I will discuss how `Comprovisador` squares with each music technology paradigm.

2.3 Algorithms in Music – a Historical Perspective

An algorithm is a finite sequence of unambiguous instructions for solving a problem or performing a calculation. Such procedures have been used since antiquity – a well-known example being Euclid's algorithm for finding the

greatest common divisor of two integers. An algorithm is said to be deterministic if, given a particular input, it always produces the same output. A non-deterministic behaviour is obtained when, for example, a degree of randomness is employed as part of the algorithm's logic.

In music composition, systematic procedures are often employed in the form of composition techniques. Harmonising a melody as a four-part chorale, for example, can be carried out by following a well-defined sequence of steps (see [Boc03, p. 40-42]). Examples of procedural composition techniques exist throughout History, from Machaut's isorhythmic motets in the fourteenth century to Boulez's total serialism in the fifties. Hence, as Gonçalo Gato writes:

> If a given technique can be strictly described and formalized in terms of the process it involves, then it can be regarded as an algorithm. [Gat16, p. 31]

Although the above considerations do not necessarily imply the use of computers, nowadays the term algorithm is linked to computing. And, indeed, if a composition technique can be formulated as an algorithm it can be realised by a computer. This is the premiss behind the advent of computer-assisted composition – a practice which I will look into in the following section.

In general, computer algorithms are useful for their ability to perform complex calculations with great precision and speed. With today's widely available technology, most algorithms for music (composition, sound synthesis and others) can be computed in real time or with negligible latency. This fact has allowed the emergence of real-time interactive music systems, capable of participating in live performances, fulfilling diverse aesthetic goals (see Section 2.3.2).

2.3.1 Deferred Time

The era of Computer-Assisted Composition (CAC) dawned with the creation of the *Illiac Suite for String Quartet* (1955-56), composed by Lejaren Hiller and Leonard Isaacson using the Illinois Automatic Computer (ILLIAC), located at the University of Illinois [DJ97, p. 374], [HI59]. The complete score, published in 1957, consists of four movements which result from the transcription of computer output (printed alphanumeric code) produced while carrying out a series of experiments. Those experiments were conceived by the authors in order to "determine whether (...) computers (...) can be used to generate music subject only to general instructions derived from various specified 'rules' of composition" [HI59, p. 2]. Their experiments ranged from monody and

first-species strict counterpoint to contemporary serial techniques and "highly unusual ways to produce radically different species of music" [HI59, p. 4].

Iannis Xenakis was another pioneer in this field. While Hiller and Isaacson's primary aim was the scientific research [HI59, p. 5], Xenakis sought specific aesthetic goals. In 1992, regarding the introduction of mathematics in his music, he wrote:

> [I]f, thanks to complexity, the strict, deterministic causality which the neo-serialists postulated was lost, then it was necessary to replace it by a more general causality, by a probabilistic logic which would contain strict serial causality as a particular case. This is the function of stochastic science. (...) As a result of the impasse in serial music, (...) I originated in 1954 a music constructed from the principle of indeterminism; (...) I named it "Stochastic Music". The laws of the calculus of probabilities entered composition through musical necessity. (...) They are the laws of the passage from complete order to total disorder in a continuous or explosive manner. [Xen92, p. 8-9]

In 1956-57, he composed *Achorripsis* implementing the thesis of minimum of constraints, which is based on probability theory. Parameters such as orchestral density, pitches, durations, successions, dynamics, glissando direction and speed are derived from probability distributions [Xen92, Har04].

Interested in the treatment of composition by machines, Xenakis translated the compositional scheme of *Achorripsis*' into a computer algorithm – the Stochastic Music Program (SMP). In 1962, the SMP was run in IBM-France's 7090 computer and six computer-assisted instrumental compositions were created [Xen92, Har04, DJ97]. Regarding the advantages of using computers in musical composition, Xenakis refers:

> Freed from tedious calculations the composer is able to devote himself to the general problems that the new musical form poses and to explore the nooks and crannies of this form while modifying the values of the input data. (...) With the aid of electronic computers the composer becomes a sort of pilot: he presses the buttons, introduces coordinates, and supervises the controls of a cosmic vessel sailing in the space of sound, across sonic constellations and galaxies that he could formerly glimpse only as a distant dream. [Xen92, p. 144]

As technology advanced, computer hardware became accessible to the general public, and new software introduced novel possibilities. In the late eighties, the PatchWork environment for CAC was created by M. Laurson,

J. Duthen, and C. Rueda at IRCAM, Paris. The environment introduced an intuitive graphical user interface (GUI) and featured the Lisp programming language. It was used by several renowned composers such as Brian Ferneyhough, Gerard Grisey, Magnus Lindberg, Tristan Murail, Kaija Saariaho among others [And11]. Current successors of the PatchWork environment are OpenMusic, also developed at IRCAM, and PWGL, developed at Sibelius Academy, Helsinki.

Many users of CAC environments tend to develop and use algorithms as generators of pre-composition material which they export to conventional notation software for further manual edition. For example, Gonçalo Gato frequently cherry-picks results by aurally assessing materials proposed by the algorithm [Gat16, p. 88, 100, 103].

In conclusion, CAC consists of a compositional practice that uses algorithmic procedures performed by a computer, typically in deferred time. Procedures can be deterministic or stochastic. Computers, with their ability to process data at high speed, allow composers to explore creative possibilities that would otherwise be out of reach. `Comprovisador` uses composition algorithms of both deterministic and stochastic types, but employs them in real time, displaying the output to musicians directly. This differentiating factor renders the system suitable for utilisation in live performance. At the same time, however, it prohibits further intervention by the composer: he/she is not allowed to edit or reject an output, although he/she may be able to modify the algorithmic parameters in time for the next output. In this regard, even though `Comprovisador`'s algorithms are designed as compositional procedures (or meta-composition), such a mode of operation is arguably closer to improvisation.

2.3.2 Real Time

Interactive Music Systems

An interactive computer music system, as proposed by Robert Rowe in 1993, is a system capable of changing its behaviour in response to musical input [Row93]. He offered a classification system built on a combination of three dimensions, attempting to identify musical motivations behind different types of input interpretation and methods of response. Here is an overview of Rowe's classification system [Row93]:

1. Drive;

   score-driven systems presuppose the use of some kind of stored music information to match against music arriving at the input;

performance-driven programs do not rely on a particular score.

2. Method of response;

   transformative methods take a live input and apply transformations to it to produce variants;

   generative methods use sets of rules to produce complete musical output;

   sequenced techniques use pre-recorded music fragments in response to real-time input.

3. Paradigm;

   instrument paradigm systems are concerned with constructing an extended musical instrument which exceeds normal instrumental response;

   player paradigm systems try to construct an artificial player, a musical presence with a personality and behaviour of its own.

Of the three broad classes of composition methods considered by the author, two are exclusively real-time: 1) transformation methods consist in performing algorithmic operations on representations emanating from an external source; 2) generation techniques produce a complete musical output from the operation of a compositional formalism, possibly aided by stored tables of fundamental material (e.g. basic scalar patterns, allowed duration values, etc.). Rowe points out a small but critical distinction separating both methods: the fact that transformation methods require live input and generation techniques do not. About this fine distinction which tends to blur easily [Row93], he concludes:

> [This distinction] seems to capture a noticeable difference in the way composers approach algorithmic methods: either the machine is changing something it hears or is generating its own material from stored data and procedures. [Row93]

According to Rowe's taxonomy, `Comprovisador` can be viewed as a performance-driven system that uses transformative algorithms. In regards to the paradigm, it is safe to say it is not an instrument, despite requiring a mediator who operates commands and parameter values via an interface – a feature that is not exclusive to this paradigm (cf. [Gio17]). On the other hand, it does not fit comfortably in the player paradigm, either. Performance-driven systems fitting the player paradigm and using transformative methods of response are often linked to the concept of human-machine interaction[12]. The computer interacts directly with the musician, outputting electro-

[12] Some examples of HMI systems will be discussed in Section 2.3.2.3 – Machine Listening.

acoustic sounds either synthesised, sampled or processed. In fact, this type of output is prevalent in most IMS that fit Rowe's classification system as a whole, regardless of their drive, method and paradigm.

In the case of `Comprovisador`, no actual sound is emitted by the computer. Instead, the computer coordinates the musical response by an ensemble of musicians who sight-read a live generated score. Hence, concerning the interaction feedback loop, the live score adds an additional step which causes a shift in the interaction focus: from human-machine to soloist-ensemble (human-human, with computer mediation). Like with player paradigm systems, the musical personality and behaviour of `Comprovisador` is indeed present but it may be obscured to the listener by the ensemble musicians' own presence.

Evidently, to the present day, many systems have been created with novel features that call for a new paradigm. These systems may involve different types of output such as live scores or other means of mediating human-human interaction. This will be subject of discussion in Section 2.4.

Real-Time Composition

Carlos Guedes discusses how musical practices within the field of computer music have shaken notions and concepts of traditional practices to a point that new definitions and taxonomies are emerging to address these basic notions [Gue18]. He writes:

> Departing from an initial intimate relation to traditional music concepts to describe computer music constructs such as "score," "orchestra," "instrument," "player;" using interaction metaphors such as "soloist with accompaniment," "conductor with orchestra," "Jazz combo," the field of computer music has expanded in ways that originated different avenues of musical expression as well as new concepts. One of them is real-time composition. [Gue18, p. 446]

Guedes defines real-time composition (RTC) as "a *Compositional practice* utilizing *interactive* music systems in which *generative* algorithms with a *non-deterministic* behavior are manipulated by a user during *performance*" [Gue17, Gue18]. The terms in italic are considered important keywords about RTC by the author. This definition accounts for systems such as *Kinetic Toolbox* – "a modular toolbox for real-time dynamic music generation" [Gue17, G+11] – which accept parametric input from a single user and where the output is the actual sound rendition of the generated algorithmic composition. Analysing this definition in relation to Rowe's taxonomy, we have that a RTC system is a performance-driven, generative instrument.

Let us take a closer look at a few of Guedes' keywords with some background perspective. Real-time interaction between musician and machine has emerged in the early seventies with works by Joel Chadabe and Sal Martirano [Eig07]. Both composers designed their own analogue computer/synthesiser systems with which they performed. This practice is referred by Dudas as one of the two basic species of composition-improvisation relationships regarding electronic and computer music – "Composing an 'instrument' that can be improvised upon in performance" [Dud10] (see Section 2.2) – although Chadabe's definition of his own practice focused on the performance context instead of the relation with instrument design. He called it *interactive composition* and defined it as "a mutually influential relationship between performer and instrument (as cited in [Eig07])". So, why composition and not improvisation since it is interactive (in real time)? Chadabe distinguishes interactive composition from instrumental improvisation as follows:

> [Interactive composition] is different from typical instrumental improvisation in that an improviser specifies and performs data in reaction to another improviser, but here the composer specifies and performs control strategies rather than data (...) The difference between instrumental improvisation and control strategy interaction is, then, primarily that of one's position in a control hierarchy, but it is true that in both cases the performer must act and react in realtime. [Cha77, p. 7]

In other words, an improviser engages with low-level musical elements whereas a composer manages high-level musical procedures. In Section 2.1, I tried answering the question "can composition be carried out in real time?" arguing that those were contradictory notions while acknowledging that aural perception of structure (both vertical and horizontal) and the use of previously constructed musical blocks could justify the hypothesis. Chadabe's distinction uncovers another aspect to consider: the level of abstraction of the musical features controlled by the performer. If the performer is controlling processes that generate musical structures comprised of low-level elements, which is to say processes that constitute composition, then the performer is operating at a metalevel.

As technology progressed, it became possible to use general-purpose computers to generate procedural music in real time[13], which meant that composers no longer needed to design the hardware of their instruments. Regarding software, the emergence of tools such as `Max` [PZS+] and Pure Data [Puc] – visual programming environments – gave many artists the possibility to design their own IMS.

In 1992, Karlheinz Essl began developing his Real-Time Composition Library (RTC-lib) for `Max`, which he used

[13] In the mid-eighties, computers were mainly used with MIDI-controlled synthesisers. Since the late nineties, affordable computers became fast enough to allow for live signal generation and processing.

in the composition of his *Lexikon-Sonate* – an interactive real-time composition for computer-controlled piano [Ess95]. This software composition does not need a musician with whom to interact: it uses 24 music-generating modules (each applying a specific compositional strategy) which are controlled by an internal conductor, generating a virtually endless, unrepeatable piece. Nonetheless, interactive performance is also possible with the user being allowed different levels of control (from random module-change requests to internal parameter specification within each module). To explore its interactive capabilities, the piece was premiered as a live broadcast during a radio program where "[t]he radio listeners (...) had the possibility to interact with the computer program by dialing a certain telephone number. Whenever a call came through, Lexikon-Sonate would change its compositional behavior by adding a new and randomly selected module into its combination chain. In this way the totality of radio listeners would 'govern' the form of the music, even though nobody could know the actual effect of their contribution." [Ess95] In Section 2.4, more recent examples of technology enabled audience mediation will be discussed.

Essl's Lexikon-Sonate can be considered generative music in that it consists of a generative system (a computer program) that will potentially produce a unique result each time it is deployed. This characteristics are mentioned by Philip Galanter in the context of generative art, which he defines as "any art practice where the artist uses a system (...) which is set into motion with some degree of autonomy contributing to or resulting in a completed work of art" [Gal03, p. 4].

According to Arne Eigenfeldt, the contemporary approach to generative art is Metacreation, which "looks at all aspects of the creative process and their potential for systematic exploration through software" [Eig16, p. 123]. In the generation of entire musical compositions through Musical Metacreation (MuMe), there is one aspect the author deems problematic: the development of musical form. In fact, he considers this a reason for continued human interaction in the dynamic generation of music. His approach to MuMe explores the use of *musebots*, which are "autonomous musical agents that interact in performance, messaging their current states in order to allow other musebots to adapt" [Eig16, p. 126]. Eigenfeldt's *Moments* is a generative installation using musebots and a *parameterBot* to generate an overall template of "moments", exploring Stockhausen's conception of Moment-form. The parameterBot determines the initial ensemble of musebots (each having preferred generative tendencies) and decides upon the overall duration of the composition, the number of sections and their proportions, as well as parameter values for a variety of features. The agents communicate their intentions and coordinate conditions for collaborative machine composition [Eig16, p. 123, 126].

In *Moments*, all musebots were created by the composer, although the musebot framework was originally conceived to allow interoperability of musebots coded by a community of developers and to coordinate

generative music software-only ensembles. More recent experiments have explored various possibilities of integrating human musicians into musebot performance [BHE+18, p. 19]. This relates to a long tradition of improvisation with IMS seeking to achieve a partnership between human and computer performers. These two types of system differ in terms of main interaction target (human-machine versus machine-machine) and algorithmic methods (transformative versus generative).

Finally, let us check how **Comprovisador** fits in Guedes' definition. **Comprovisador**'s algorithms employ deterministic as well as stochastic processes, mostly resulting in non-deterministic behaviour. They were not designed as generative algorithms but rather as transformative[14] ones as they require input from the soloist. Nonetheless, algorithmic parameters are manipulated by a user (mediator) during performance to foster interaction between improviser and ensemble. An important role of the mediator is to specify and supervise the development of the macrostructure – the musical form (cf. [Eig16]; also, see Section 4.1). One of **Comprovisador**'s main aesthetic goals is to facilitate the listener's perception of the relationship between the improvised source material and the composed response. This is achieved through the deployment of meta-composition algorithms in real time. In that sense, it is fair to say that **Comprovisador** is an interactive RTC system, although it does not fit a stricter definition of RTC like the one given by Guedes[15]. And that is because it involves features (namely, machine listening, dynamic notation with sight-reading performers and network implementation) that are not regarded as crucial for a RTC system. On the other hand, those are precisely the features that enable a kind of RTC that can foster human-human interaction mediated via computer algorithms.

Machine Listening

Machine listening (or computer audition) is a research field which comprehends a number of subfields such as speech recognition, auditory scene analysis and music information retrieval (MIR). However, in the specific context of interactive computer music systems, the term often refers to real-time audio feature extraction.

Commonly extracted features include pitch, duration and loudness (MIDI-level information – cf. [Gio17]) as well as spectral information. According to Artemi-Maria Gioti, MIDI-level information "is often described as score-level information. However, since most scores in the 20th and 21st century include instructions on

[14] Apart from the main transformative algorithms used in normal performance, **Comprovisador** has a built-in **Practice Mode** which is based on generative algorithms conceived to enable performers to get acquainted with the system's notation interface and its idiosyncrasies, in a simulated performance context. In addition, this generative tool has been successfully used in the improvement of sight-reading skills (see Section 6.1).

[15] Guedes ends up redefining real-time composition simply as *improvised composition with computers* [Gue18, p. 450] (cf. Dudas: Composed Improvisation within Interactive Performance Systems [Dud10]).

(extended) playing techniques and therefore timbre as well, MIDI-level information seems to be a more accurate description" [Gio17].

Interaction between computers and humans playing acoustic instruments is dependent on machine listening. One of the first examples produced was David Behrman's piece *On the Other Ocean* (1978). He used pitch-sensing circuitry connected to a microcomputer that controlled oscillator frequencies on a hand-made analogue synthesiser [Eig07].

The advent of the Musical Instrument Digital Interface (MIDI) standard in the early eighties enabled real-time extraction of performance information with MIDI enabled instruments, especially electronic keyboards. Such instruments are able to output a stream of performance events consisting mainly of note-on and note-off information (key press / release) as well as continuous control messages (triggered by e.g. *pitchbend* wheel, volume pedal, sliders, knobs, buttons, etc.). Various IMS developed then relied on MIDI instruments as their *sensing* stage[16]. In the following decade, affordable computers became capable of performing live signal analysis, consequently allowing the emergence of IMS that incorporated real-time audio feature extraction and acoustic instruments.

Besides MIDI-level information, in more recent years, many IMS rely on techniques that are able to extract spectral information (e.g. Mel-frequency cepstral coefficients (MFCCs)) and also gestural information in the *analysis* substage[17]. Regarding the *interpretation* substage, systems may employ data processing or Machine Learning (ML) techniques.

A good example of an interactive music system that can handle MIDI, audio and video data, relying on ML to carry out human-machine interaction is a project entitled "OMax", carried out by the research team "Musical Representations" of IRCAM [ABC+15]. OMax consists of a computer program capable of learning, in real time, the typical characteristics of a musician's improvisational style, as well as to play with him, in an interactive way (conforming to Rowe's player paradigm). For an overview of IMS categorized by the types of data involved in the *analysis* substage and the techniques used in the *interpretation* substage of their machine listening algorithms, see [Gio17]. For a broader view of approaches to computational improvisation, where a taxonomy of improvisational music systems is proposed, see [GKM+18].

[16]Rowe conceptualised the processing chain of an IMS in three stages: the *sensing* stage (collection of human performance data), the *processing* stage (data interpretation and response preparation (composition)), and the *response* stage (sound production / musical output) [Row93].

[17]Gioti redefines Rowe's processing chain, placing *analysis* (feature extraction) and *interpretation* (construction of higher-level representations of the human input) inside the *sensory processing* stage – the machine listening algorithm [Gio17]

Regarding the purpose of machine listening algorithms, Gioti writes: "'listening' in interactive systems goes beyond signal-level descriptors (sensory information), to higher-level representations of the audio input that are bound to aesthetic and/or compositional choices (symbolic information)" [Gio17]. In the development of `Comprovisador`, I have favoured MIDI-level information and rule-based data processing over MFCCs and ML, respectively. These choices are based on the proximity between input and output data types (MIDI-level information => score data) and the impression that rule-based algorithms allow me to formalise my own musical thinking, enabling a musical outcome that is arguably closer to my intent (how it would sound like had I written all the notes myself) than with ML techniques. Having said that, in the future I do intend to explore both spectral analysis and ML – MuBu[18] being a strong candidate for this application.

2.4 Distributed Performance and Live Scores – Current Tendencies

2.4.1 Networked Music Performance

Networked music performance is a practice that has emerged in the recent decades thanks to the development of computer network technologies and the creativity of musicians [HD09]. It consists on performance situations where a group of musicians interact over a network (LAN or WAN). This interaction can be achieved, for example, by audio streaming, score rendering or strategies for graphical direction. The different topologies of NMP have been studied by authors like Barbosa [Bar03] and Matuszewski [MSB19]. Another area of focus in NMP research is latency and the corresponding strategies for addressing it (see [BCG05, BC11]).

Below are some examples of works or systems that use a network as a medium for musical interaction. Some of these, such as the aforementioned Quintet.net [Haj05], also employ network-distributed animated notation and are therefore more pertinent to this investigation.

Decibel ScorePlayer [HVWJ15] is an iPad application developed by the Decibel New Music Ensemble, a group led by composer Cat Hope. This application enables network-synchronised scrolling of proportional colour music scores and audio playback. The ScorePlayer was originally designed for synchronisation of multiple iPads over a LAN. Subsequently, it was upgraded for use over a WAN through the incorporation of client-server functionality, thereby enhancing its telematic capabilities [JHV+17]. This development was showcased in a telematic concert held during the Sound and Music Computing Summer School 2016. The event involved the

[18] MuBu For `Max` is a toolbox for multimodal analysis of sound and motion, interactive sound synthesis and machine learning [SRS+09].

collaboration of three academic institutions: the Hochschule für Musik und Theater in Hamburg, Germany; Edith Cowan University in Perth, Western Australia; and Stanford University in California, United States [JHV+17]. While Decibel ScorePlayer facilitated score synchronisation, JackTrip ensured the delivery of multi-channel low-latency audio streaming. Furthermore, videoconferencing software enabled the real-time exchange of visual data between performers and audience members in the three locations. I was privileged to attend the concert in one of the locations (Hamburg), and I can attest to the success of the event, which presented a series of new works (graphic scores) by participants of the Summer School, featuring performers from three different countries across fifteen time zones.

Another relevant example of a score player system is Jonathan Bell's SmartVox [Bel18], which employs staff-based notation in conjunction with audio scores. SmartVox is a web-based distributed media player designed for use as a notation tool for choral practices. The audio-visual scores are created with the `bach` environment for `Max` and subsequently exported as movie files. In a choral context, singers hear (using earphones) and see their own part displayed in the browser of their smartphone. The whole is synchronised through the distributed state of the web application [Bel18].

In May 2019, a networked music performance dubbed Symphony for a Tunnel for 144 musicians and distributed score display system took place in Hamburg's St. Pauli Elbe Tunnel [HG19] – a 100-year old tunnel under the Elbe river (see Fig. 2.3). The performance featured Drawsocket [GH19], a recently developed system which was able to draw and synchronise scores across the required 144 iPads, connected via Wi-Fi. The system provides control over diverse media features of web browsers, notably SVG[19] which can be used to draw animated graphic notation. Through integration with MaxScore[20] [HG19], the system also enables common-practice notation. On the server side, Drawsocket uses `Max` as its primary controller interface and Node.js (Node for `Max` (N4M)) for server-client communication.

Of the examples provided, only SmartVox and Drawsocket make exclusive use of web technologies on the client side, as Decibel ScorePlayer is a native application for iPad. It is also pertinent to note my contribution to the work entitled `GarB'urlesco` [Lou21, Lou19a], which similarly employs web technologies (See Appendix D). This piece employs mobile devices in two distinct yet interrelated capacities: firstly, as a means of displaying scores on-screen, and secondly, as a method for multichannel audio distribution. This approach builds upon the methodologies explored in prior work by Cheng Lee, although these do not incorporate the use of music notation.

[19] Scalable Vector Graphics (SVG) is an Extensible Markup Language (XML)-based vector image format for two-dimensional graphics with support for interactivity and animation.

[20] MaxScore is a music notation library for the `Max` environment [com].

Lee proposes an approach for incorporating computer music and virtual reality practices into a multimedia performance installation requiring the audience members to use their own smartphones as 360-degree viewing devices [Lee17]. The author also proposes the use of wireless speakers carried around the venue as a means of achieving immersive sound and music effects in substitution for a multi-channel surround-sound system.

Finally, I should mention a.bel [CRRP16] – a system presented in 2015 at Casa da Música, in Porto, Portugal, in a concert where almost 1000 smartphones were used as musical instruments by the members of the audience. The event featured pieces by four composers – Carlos Guedes, José Alberto Gomes, Neil Leonard and Rui Penha – using different approaches to audience participation via their smartphones and the a.bel system.

In conclusion, it is important to highlight the various creative possibilities that NMP technologies have brought about. These include real-time audio streaming, which enables remote performance of music according to a number of distinct topologies; score distribution, which allows for synchronised navigation of multiple score parts; and multichannel audio distribution, which facilitates the creation of immersive sound spaces through a low-cost, simplified logistics approach. Additionally, NMP technologies have the potential to facilitate the integration of external inputs, such as audience participation, into the behaviour of generative algorithms. An early example of this is Essl's Lexikon-Sonate (1992), discussed in Section 2.3.2. The following section will demonstrate how dynamic notation has augmented this possibility, among other potential avenues.

2.4.2 Animated Notation

The first contact I add with animated notation was in 2003 during the attendance of a concert by the Kronos Quartet entitled "Visual Music" [kro03]. Their interpretation of Penderecki's "Quartetto per archi" involved a scrolling projection of the score in a very large screen for the audience to follow. The screen was divided by a vertical red bar which served as a synchronisation device for the musicians who had their backs turned to the audience (see Figure 2.4). This performance made me realise the potential of animated notation in solving musical problems – in this case, synchronisation in a proportional (non-metric) notation context – apart from its aesthetic possibilities in the visual field.

Since the late nineties, dynamic musical notation has been increasingly used in real-time music systems enabling various kinds of new interactive features – such as audience participation, where the audience is able to influence the behaviour of the algorithms [Fre08]. The advent of novel technologies, including tablets, laptops and video projectors, has opened up new pathways for creative expression through the use of animated

Figure 2.3: Rehearsal of Hajdu's "Symphonie im St. Pauli Elbtunnel" (May 2019), in Hamburg.

notation. The following works exemplify instances where algorithmic behaviour has been shaped by non-musical live inputs. A subsequent analysis will be presented on the diverse notational trends and their affordances.

"Flock" [Fre08, Fre07], a piece by Jason Freeman for saxophone quartet, video, electronic sound, dancers, and audience participation premiered in 2007, uses wirelessly connected PDAs[21] mounted on each player's instrument. Those devices display music notation generated from the locations of musicians, dancers, and audience members as they move and interact with each other. A significant number of compositions by this composer incorporate audience participation in a variety of ways (e.g. "Glimmer", presented further below).

"Peripatoi" [Pen11] is a musical composition for bass clarinet, vibraphone, piano and double bass, which incorporates a real-time score and video created by Portuguese composer Rui Penha. The video component comprised a quasi-real-time manipulated stream, which captured live occurrences outside the concert venue. The score elements, which had been prerehearsed, were live-permutated in response to the captured events.

[21] – personal digital assistants.

Figure 2.4: Penderecki's "Quartetto per archi" performed by the Kronos Quartet, in 2003.

The performance of "Multimedia Improvisation with Brain Waves for cello, live electronics and image processing" [YH17], by Hirayama and Yokoyama, was held during the 2017 International Computer Music Conference (ICMC), in Shanghai, and event which I was privileged to attend. The piece employs real-time visualisation and sonification of brainwaves, with the resulting data displayed as staff-based notation for the cellist to perform. During the course of the performance, the subject, who was wearing a brainwave sensor, was subjected to varying degrees of physical and psychological stress, which in turn affected the algorithms and resulted in a musical dramaturgy.

Types of notation

The nature of notation can be **descriptive** or **prescriptive**, depending on the degree of specification it conveys. It can also be somewhere in the middle. A prescriptive notation (such as the common-practice staff-based western notation) is suitable for fixed composition, with somewhat limited space for interpretation. A more descriptive notation will yield better results with music that calls for greater freedom of interpretation and even improvisation.

Graphic scores

Many approaches to dynamic notation tend to use animated graphic scores [Fre08, HVWJ15, Smi14a] and other kinds of non-staff notation precisely because they tend to be more descriptive in nature. But that is not the only reason: such approaches have a visual level that can be in itself an aesthetic goal, since it is common to have the animated notation projected for audiences to see. Many of these approaches rely on the improvisational skills of all performers to make their own interpretation of the score whereas **Comprovisador** – apart from the soloist (or soloists) – requires more traditional sight-reading skills from the ensemble performers, since it is based on staff notation. It should be noted that graphic notation can be very prescriptive, as well, depending on the approach.

One remarkable example of animated notation utilising graphic scores is "Õdaiko – a real-time score generator based on rhythm", created by Filipe Lopes [Lop09]. Lopes also developed a series of works titled "Do Desenho e do Som [Lop24]," which employs instruments and a software system bearing the same name. The aforementioned Decibel ScorePlayer [HVWJ15] is an illustrative instance of a system incorporating animated graphic scores.

Staff-base scores

The potential for employing staff-based notation in real-time applications has been made possible by recent advancements in software. Among these we find **MaxScore** [DH08], **INScore** [FDLO10] and the library used in **Comprovisador**: **bach**[22] [AG15, AG10].

A successor (one might say) of the PatchWork / OpenMusic family of CAC environments is **bach**: automated composer's helper – a library of tools enabling musical notation and computer-aided composition inside **Max** [AG15]. **Max** is a widely used visual programming environment for creating real-time applications. Thanks to the integration with **Max** and its real-time paradigm, **bach** has enabled and fostered new ways of using music notation and composition algorithms in a reactive and interactive fashion. **bach**'s website (http://www.bachproject.net) presents many creative and diverse projects carried out by users of the library [AG10]. Amongst those projects, we find a page dedicated to **Comprovisador** and a link to it's website.

[22] The **bach** library enables music notation and CAC tools inside the **Max** environment. Its most prominent objects are **bach**.score (for standard metric notation) and **bach**.roll (for proportional notation). Both objects are notation editors (meaning a user can interact with them via mouse and keyboard to create/modify the score) and score players (they can read back score data and drive a MIDI synth or similar). Moreover, they feature **Max** type inputs and outputs in order to be controlled by and/or to control other **Max** processes in real time [AG15].

Here, I must refer Hajdu's system quintet.net [Haj05] and works like "Roomtone Variations" [Col13] that use the Maxscore Library. "Zero Waste", by Nick Didkovsky [WDH18], was composed with a Java library (Java Music Specification Language – JMSL) developed by the author of the piece. JSML is also the core component of the Maxscore Library [DH08].

I should now state why I opted for `bach` and not the others: 1) it was free (at that time, Maxscore was not), and 2) it was native to `Max` (INScore could be made compatible via OSC).

Other non-staff scores

In "Glimmer", a composition for chamber orchestra and audience, Jason Freeman used coloured LED light tubes to convey pitch and dynamics information to performers [Fre08]. This can be effectively classified as a **light score**. The audience is invited to participate in the musical performance by waving LED light sticks, which are captured on camera, influencing the musical outcome. A computer program analyses the video data and assigns a value to each audience group. These values are then relayed to a corresponding group of musicians in the orchestra.

Besides light scores, other types of non-staff / non-graphic scores include **audio scores** [Bha18, Bel18], **haptic scores** [BCB+16] and **VR/AR scores** [San18]. By way of illustrating the concept of a haptic score, the :body:suit:score, developed by a research team led by Sandeep Bhagwati, is an essential reference point. The :body:suit:score is a wearable apparatus designed for use by mobile musicians. It is ergonomic, responsive and capable of transmitting sensor data and receiving control messages. It serves as a conduit for communication between composers and mobile musicians, enabling the transfer of information via embedded vibrotactile feedback devices located on the torso and extremities. Additionally, the suit is capable of collecting performance-related data (spatial position, audio descriptor analysis) and transmitting it wirelessly to the composer.

Notational purpose

Western traditional notation specifies the **resulting sound**, and this is so for many types of alternative notations. But even in western tradition we find notation types that specify **player action** instead – tablature being a good example. The work of Seth Shafer, notably his paper entitled "Performer action modeling in real-time

notation" [Sha17a] must be mentioned in this context.

Synchronisation strategies

Among systems that use graphical direction strategies, we find VizScore [Sha15, Sha17b], Decibel ScorePlayer, Quintet.net and MaxScore [JHV+17]. In ScorePlayer, the main strategy consists on scrolling the score from right to left under a fixed vertical line , while the other two systems feature a fixed score and a cursor that moves horizontally. Both strategies were adopted by `Comprovisador` but the former was abandoned at an early stage. Instead, a strategy was developed in which a bouncing ball is responsible for synchronising attacks and/or conveying a pulse (see Section 3.3.1). Programming of the bouncing ball incorporates motion laws that convincingly translate arsis and thesis sensations.

The field of animated notation is a rich one, characterised by a diverse range of approaches and techniques. While there are numerous examples that could have been included, the ones presented here are representative of the principal tendencies and may assist in better understanding the context in which Comprovisador is situated.

3

Presentation of the System

Concept

In broad terms, **`Comprovisador`** is able to listen to a solo improvisation and generate different musical responses in the shape of a real-time distributed score that is immediately sight-read by an ensemble of musicians. It employs: 1) machine listening techniques to perform real-time analysis of the soloist's improvisation; 2) algorithmic compositional procedures to generate consequent musical responses to the improvisation; and 3) dynamic notation, presented in networked computer screens, to allow musicians to sight-read the generated responses. Moreover, algorithmic parameters can be manipulated in real time, from a hardware terminal, enabling a conductor/composer to mediate the interaction between improviser(s) and sight-readers – between soloist(s) and ensemble. Wireless connectivity, which allows real-time communication of score data, also makes

it possible to place computers – and, therefore, musicians – apart from each other, allowing non-standard spacial settings.

3.1 System's Components

3.1.1 Hardware

In its simplest configuration, `Comprovisador` needs the following hardware equipment (see Figure 3.1):

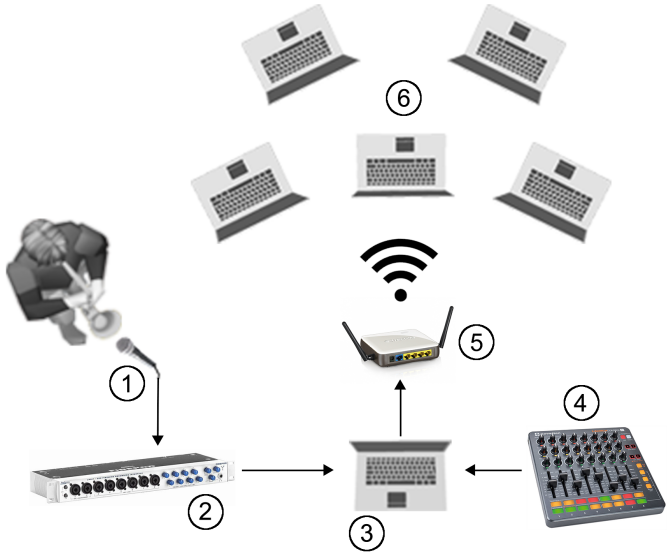


Figure 3.1: `Comprovisador`: hardware setup – core configuration.

① a [number of] microphone(s) – to capture the improvisation of the soloist(s) (only necessary for non-MIDI instruments);

② an audio/MIDI interface – to convert the analogue signal of the microphone(s) into digital signal and/or to input raw MIDI data;

③ a host computer – which applies compositional procedures upon the input it receives from the audio/MIDI interface and the control surface;

④ a control surface – through which algorithm parameters are manipulated;

⑤ a wireless router – which establishes communication between computers; and

⑥ a number of client computers – to render and display the animated score to the musicians in the ensemble.

This has been the core configuration in use since the first public performance ("Comprovisação nº 1"). Along the way, other components were added through iterative design, improving control and communication, and providing musicians with visual and audio cues (see Figure 3.2), namely:

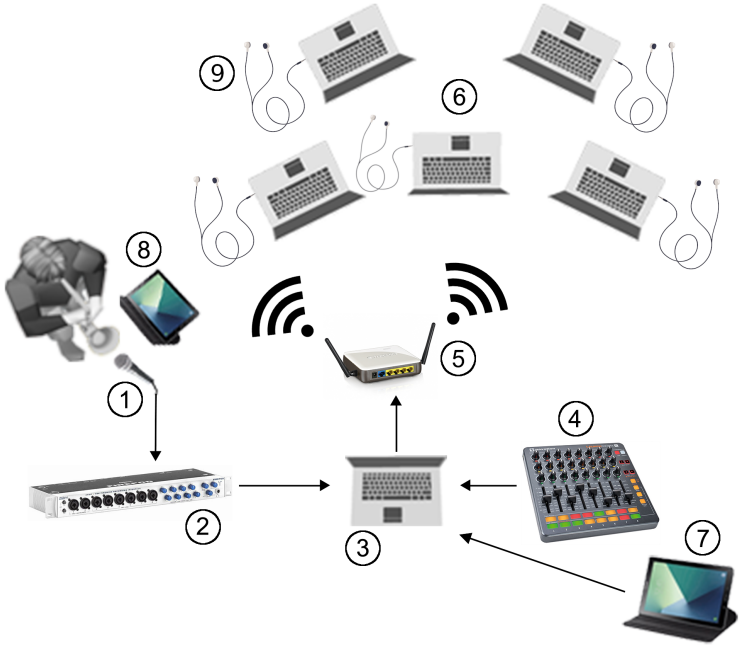


Figure 3.2: **`Comprovisador`**: hardware setup – browser-enabled tablets (for interaction or visualization via **`Miraweb`**) and earphone sets (for cue notes and adaptable metronome).

⑦ a tablet – used by the mediator for multi-touch control over a messenger module (since "Comprovisação nº 5");

⑧ a [number of] browser-enabled device(s) (optional) – facilitating communication of musical directions to the soloist(s) (since "Comprovisação nº 6"); also providing parameter visualisation to performers using augmented instruments[1] (since "Comprovisação nº 9");

⑨ a number of earphone sets – required for singers (since "Comprovisação nº 5"), allowing them to receive cue notes for intonation purposes; also required for instrumentalists (since "Comprovisação nº 11"), enabling them to follow an adaptable metronome (see below) for enhanced attack precision.

[1]The augmented instrument featured in "Comprovisação nº 9" – HASGS – is shown in Section 5.1.11. Also shown is the visualisation interface that was designed to provide the soloist with visual feedback of the parameter manipulation he performed.

Both ⑦ and ⑧ make use of **Miraweb** [Ben16], a **Max** package that enables visualisation/interaction with **Max** GUI (Graphical User Interface) objects via browser-enabled devices, such as tablets, smartphones or laptops. Regarding ⑨, the use of earphones allows singers to receive cue notes that effectively serve as an audio score (cf. [Bha18]) working in tandem with the visual score. It is useful for intonation purposes, especially so in microtonal contexts, with cue notes tuned to cent-level precision[2]. Moreover, recent developments have been made allowing each ensemble member (instrumentalists and singers alike) to pick up timing cues from an adaptable metronome through earphones. This metronome is able to adapt to an irregular pulse and, crucially, to a pulseless time-axis definition. It works in conjunction with the bouncing ball (and the singers' cue notes, if applicable) for enhanced attack precision. Both types of audio cue (cue notes and adaptable metronome) are, of course, specific to each player's part – as is the bouncing ball.

Typically, one client computer is used for every two performers – which is analogous to the standard arrangement in an orchestra, with two players per stand. This configuration allows for an efficient use of stage space and technical resources. In some cases, though, it is convenient to use one computer for each performer[3]. Such cases include the use of large-sized instruments (e.g. piano, pitched percussion), the distribution of musicians across a large space and the use of earphones (see ⑨ above). Hence, the recent implementation of the adaptable metronome, requiring earphones for all players, brought a limitation in regards to the availability of the dual-instrument per computer configuration. To overcome this limitation, it may be possible for pairs of players to share one set of earphones, possibly having to split its cord or attach extensions. This possibility has not yet been tested in real-world conditions, even though **Comprovisador** is capable of sending individual cues through each channel of the client computer's audio port.

The system is fully reconfigurable and flexible regarding instrumentation of the ensemble, in regards to number and kind. Testing in real-world conditions has shown it is compatible with Mac OSX and Windows systems. Machines with 64-bit processors, 13" or greater displays and dedicated graphics processors (for optimal OpenGL rendering) are recommended, although it will adequately run on modern laptops with multi-core processors and integrated graphics.

3.1.2 Software

Software for this system is being developed in **Max** [PZS+], with extensive use of the **bach** library [AG15] for its notation features, CAC tools and **Max** integration. The system consists of two applications: one which runs on

[2] For an example of such a context, see "Comprovisação nº 6" in Section 4.2.

[3] In Section 3.3, differences between single- and dual-instrument configuration are discussed regarding the notation interface.

the host computer and another which is instantiated on each of the client computers.

The host application (`Comprovisador.host`) is responsible for receiving and analysing the input from the soloist(s), calculating the compositional procedures and responding to commands from the conductor/composer. The client application (`Comprovisador.client`) is in charge of rendering the generated score and displaying it to the musicians (see Figure 3.3).

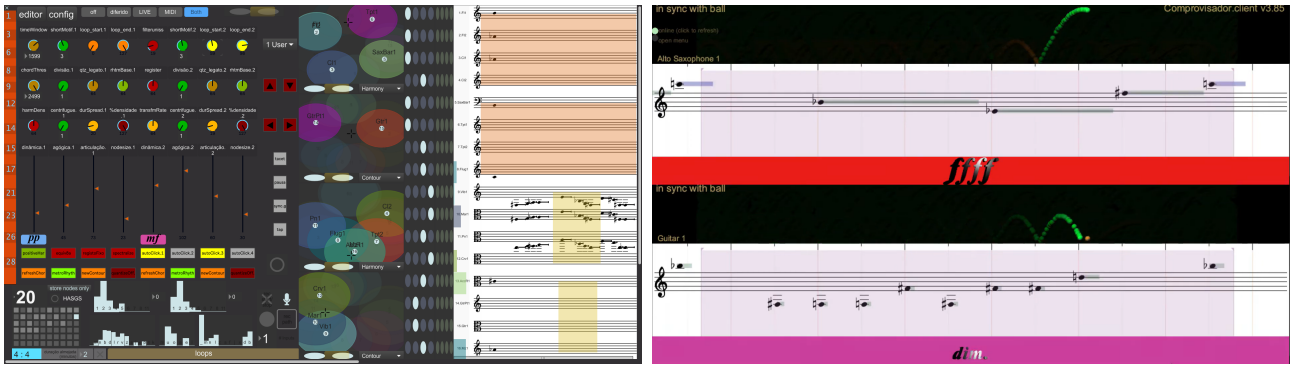


Figure 3.3: `Comprovisador.host` (left) and `Comprovisador.client` (right) applications' main windows.

Host Application

The host application consists of multiple modules (see Figure 3.4), namely:

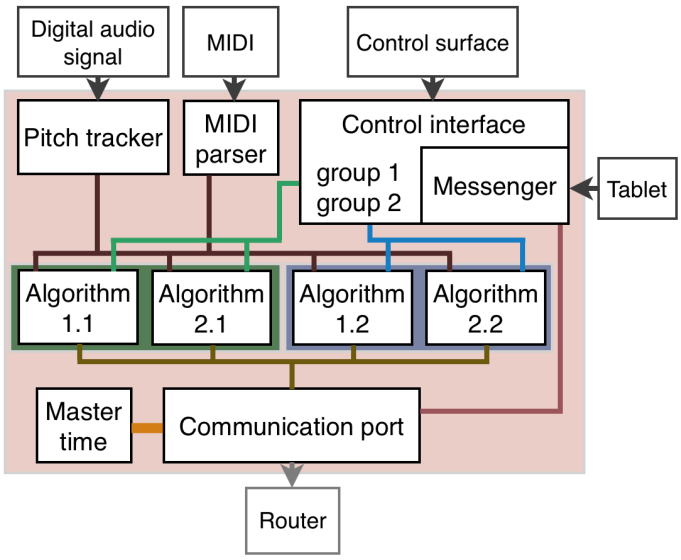


Figure 3.4: `Comprovisador`: host application overview.

pitch tracker – here, musical notes played by the soloist are deciphered in real-time from the digital audio signal input;

MIDI parser – in the case of MIDI enabled instruments, a MIDI parsing module is used instead of the pitch tracker; both polyphonic and multi-channel input are accepted;

control interface – this module consists of two control groups containing a total of four slots for algorithms; algorithmic parameters are manipulated in real-time by the conductor/composer; the control interface provides graphical feedback for all commands performed on the external control surface (mirroring) and it is possible to store and recall parameter presets; it also provides information to its operator about ongoing algorithmic procedures;

messenger – inside the control interface lies a Miraweb [Ben16] enabled messenger submodule, which can be operated through a multitouch-enabled device like a tablet; recipient selection is made easy by selecting the group to which they are allocated or the instrument family they belong to; although there are default message options available, new ones can also be typed; additionally, notation signs can be used for ornaments, articulations, and special techniques besides textual messages; these can also be selected automatically at random at a specified rate (see Figure 3.5);

compositional algorithms – there are two distinct algorithms – **Harmony** and **Contour** – which are instantiated in all four slots of the control groups (each slot can host any of the two algorithms[4]); instruments can be assigned to any of these four instances, which work in parallel; each algorithm generates different musical responses (broadly, chords[5] and melodic contours) when receiving pitch and parametric data; furthermore, each algorithm has two main variations; generated musical responses take into account idiomatic aspects of the assigned instruments such as range (and whether it is dependent on dynamics), polyphonic capabilities, etc.;

communication port – here, generated musical data are sent via UDP or TCP protocols to client computers; data are rendered into musical notation in every client application;

master time – this module implements an adaptation of Roger Dannenberg's Time-Flow concept [BD99, Dan17] capable of measuring network latency, setting a global network time and keeping events synchronised across the network through time-stamping.

Many aspects of the host application – communication port as well as parts of algorithms and control interface – are automatically configured on startup. This feature enables the system to be flexible regarding instrumentation. Implementation details of this feature are described in Section 3.5.2.

[4] To ensure future resilience, the system has been updated to allow up to ten algorithms to be allocated in each slot. This includes the two current and eight potential future algorithms.

[5] There is also still a submodule for generating sung text using stochastic rules. It works in tandem with algorithm **Harmony** when required. At present, there is no method in place to generate sung text with algorithm **Contour**.

Figure 3.5: **Comprovisador**: messenger window (Miraweb enabled). Tapping / clicking a family or group (e.g.: Brass) selects all relevant instruments. Tapping / clicking a predefined message (e.g.: "con sord") sends it to selected instruments.

Client Application

In addition to notation rendering, the client application also carries out some algorithmic tasks that could in theory be performed by the host application. Examples of such tasks include the quantisation used in the Quantum Loop mode (see Section 3.4.4), the transposition required for all transposing instruments, and the respelling of accidentals to avoid augmented and diminished intervals where possible. The goal of this task decentralization is to unburden the host computer's CPU and to keep the wireless data traffic as lightweight as possible.

Another important feature of the client application is the **Practice Mode**. Initially, this tool was developed in order to enable performers to get acquainted with the system's notation interface and its idiosyncrasies. This way, even before the first rehearsal, performers were able to experience sight-reading in a simulated performance context, being subject to unpredictable note patterns (thanks to a random walk algorithm) and to a specific cueing strategy – the bouncing ball (and, more recently, the adaptable metronome). Moreover, this tool can play a significant role in a new way of improving sight-reading skills, in a broader educational context (see Chapter 6).

The client application is structured as follows (see Figure 3.6):

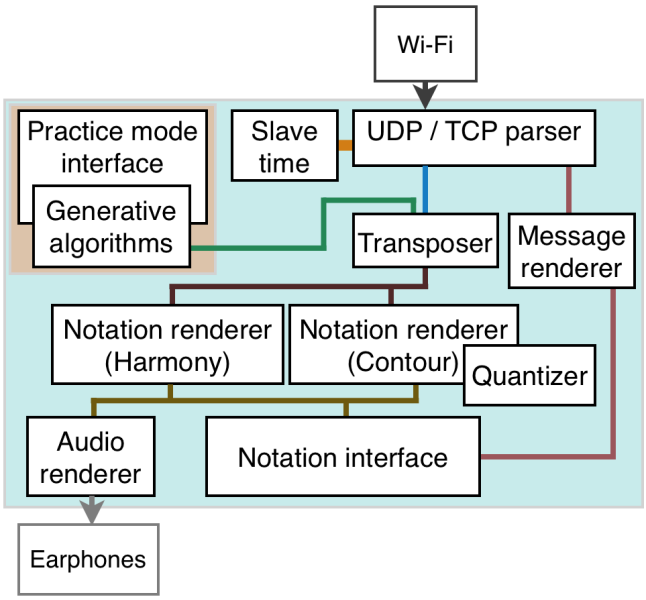


Figure 3.6: `Comprovisador`: software - client application overview.

UDP / TCP parser – arriving network data gets appropriately parsed;

slave time – this module is the client counterpart of the host's **master time** module; it enables local prediction of global time between global time updates;

message renderer – arriving messages are displayed in the notation interface;

transposer – for transposing instruments, pitch information is diverted from other score data and transposed, before being rendered as notation;

practice mode interface – a GUI that lets the user manipulate algorithmic parameters for the `Practice Mode`; with this tool, the user can customize the level of difficulty of the generated score, for practice purposes;

generative algorithms – the stochastic behaviour of these algorithms is intended to emulate real-world performance situations within the `Practice Mode`;

notation renderers – notes and their durations are set in place according to particular rules of the active algorithm; these modules also control the audio score renderer and the behaviour of the bouncing ball; furthermore, Harmony renderer handles the generated sung text;

quantiser – whenever a musical phrase needs to be quantised (in order to be presented in standard rhythmic notation) this module is activated;

audio score renderer – for singers or for general practice, this module renders an audio score from the notation generated in real time; singers may listen in advance to sounds as cues for upcoming notes (see below);

since recently, this module also renders an adaptable metronome; both cue notes and metronome adapt to different time conceptions, metronomic or chronometric, working in conjunction with the bouncing ball;

notation interface – the actual graphical interface on which the animated notation is drawn, including the staves and notes, the bouncing ball, dynamics and text messages (see Section 3.3 for a detailed description of the notation interface).

It should be clarified that cue notes for singers consist of a pair of sounds for each note on the staff: one sound that is played in advance, giving the singer the note to be sung, and another that is played during the whole length of the written note, aiding both in intonation and in timing accuracy of onset and ending. The former is rendered with a rapidly decaying complex waveform; for contrast, the latter uses a sustained simple waveform. In addition, both convey the appropriate dynamics through their volume level, aiding in this domain as well.

Video Example 3.1 below demonstrates the audio score in action. For context, this example was produced with the `Practice Mode`[6], using the Bohlen-Pierce scale. Also visible in the example is the generated sung text. This video example was used as part of a submission to a Call for Scores for the TENOR2020/2021 conference (see Appendix E). The submission was accepted; however, the piece was not performed due to restrictions from the COVID19 pandemic.

Video Example 3.1

Comprovisador.client: audio score

3.2 Aesthetic Goals and Design Considerations

3.2.1 Notation Type

Although there are several different approaches to real-time notation as shown in Chapter 2, most choices fall into two broad categories: staff-based notation and non-staff notation[7]. The latter, being more descriptive in nature, has many advantages: it encourages performers to be creative in translating non-conventional signs into

[6] For that reason, musical directions such as “frullato” and “trillo” should be excused in this situation.

[7] One could say graphic notation, but that term would not encompass works where notation goes beyond the scope of graphical signs – e.g. Freeman’s light scores [Fre08] or Bhagwati’s haptic and audio scores [BCB+16, Bha18]

sound and music, it relieves performers of the responsibility of having to play prescribed notes and, potentially, it embodies an aesthetic value as a visual or multimedia experience (e.g. through video projection).

On the other hand, while it is true that staff-based notation may put performers in a less creative and less forgiving situation (and the idea of projecting the score so that the audience may follow the performers' potential mistakes would likely generate further anxiety), it is also true that it enables a greater compositional control over certain musical parameters – namely, pitch and harmony – which are of interest to me. Wrong notes as well as timing discrepancies and other audible mistakes are bound to occur. But it is possible to take this error expectation into account and somehow incorporate it in the aesthetics of the piece.

A good example of this incorporation is Nick Didkovsky's "Zero Waste" [Fre08, Haj16, WDH18], for sight-reading pianist and real-time transcription algorithm. In this piece, the performer sight-reads two initial measures of software-generated music while the algorithm transcribes the performer's rendition. The transcription is immediately displayed to the performer and the process repeats itself. Both performer and algorithm are expected to fail in order for proliferations of the initial gesture to take place. As Georg Hajdu points out [Haj16], the abstract, chromatic quality of the material selected for the opening bars prevents an error from being perceived as such. Instead, error becomes the shaping force of the piece.

During early development stages of `Comprovisador`, the concept of "extreme sight-reading" proposed by Jason Freeman [Fre08] had an influence on the choice of using staff-based notation. The influence came not from a particular example in the article, but rather from my reading of the concept expressed in the title. Moreover, from an aesthetic point of view, an initial aim was to ensure that the listener would perceive the composed material as originating from the improvisation of the soloist. The chosen notation type seemed to be suitable for this purpose due to its advantage in the specification of pitch. Strategies for the design of a functional notation interface have thus been devised, taking into account the problem of error and all related issues. The element of time was found to be crucial in this conception, as will be exposed in Section 3.2.2.

3.2.2 Synchronised Attacks

In a hybrid type of performance such as comprovisation, it is presumable that an informed listener will be looking for clues as to what is being improvised and what is being composed – and even trying to assess the effectiveness of the notation system. As referred in Section 2.1, Lehmann and Kopiez have concluded that a listener trying to discern between composed and improvised music may associate 'togetherness' and precision of an ensemble

with composition and entropy with improvisation [LK10]. In this line of thought, I find synchronisation to be an effective way to let the listener perceive organization as opposed to chaos, hinting at what is being composed in real-time. Additionally, my prior engagement with Soundpainting and other forms of conducted improvisation has provided insight into the potential of synchronised attacks – the simplicity of implementation through an appropriate command gesture and the efficacy of the musical outcome despite individual choices on which note to contribute with.

In a synchronised attack, even if a few notes are false or missing, there is no way the listener can tell, for the score is known only to the player. What shines through is the sense of cohesion, which can signal composition. And, as mentioned before, a mistake can become a plausible shaping force – in this case, by influencing the improviser's playing (as shall be discussed in Chapter 5).

This does not imply that insufficient thought has been devoted to the harmonic dimension – quite the opposite (the proof being in the name given to the algorithm in question: **Harmony**), as it too constitutes a form of sound organisation that listeners can grasp. Moreover, pitch organisation constitutes a key advantage over gesture languages for conducted improvisation or live composition. Details on the harmonic procedures carried out by this algorithm are given in Section 3.4.3.

In order to have synchronised attacks in an extreme sight-reading context[8], the issue of time is of great importance. Firstly, musicians need time to recognize each note or group of notes (or, as John Sloboda would phrase it, to register pitch symbols in memory [Slo76]); secondly, they need time to prepare the notes on their instrument; lastly, they need to be precisely cued – and effective cueing involves very specific timing. And often motion, such as when cueing is performed by a conductor. In any of these three steps, problems may arise leading to delays and jeopardizing synchronisation. Hence, establishing a reading time window and implementing a visual cueing device (consisting of a bouncing ball) were my first design choices[9]. Both would have to be time-adjustable, according to musical goal and/or technical difficulty. Ideally, the reading time window should be sufficiently large to accommodate a musician's typical eye-hand span (cf. [Slo74]), without being such as to cause unnecessary breaks in the musical discourse (as discussed in Section 1.2).

This reading mode, labelled "in sync with green ball", is demonstrated in Video Example 3.2. With respect to the reading time window, it can be seen that first a note appears on the score and shortly afterwards the green ball

[8] Such a context need not involve dynamic notation to be considered extreme; one can imagine a première, before an audience, of a conventional paper-score composition for orchestra, where the parts have been printed at the last minute.

[9] To be precise, the first implementation of a visual cueing device, used only in "Comprovisação n° 1", consisted of an indicator that bounced in place, being stationary in the x-axis as the score would scroll underneath it from right to left, but the principles of a reading time window and a bouncing motion indicator were already in place.

bounces, preparing the attack. The reading time window is therefore the interval between the arrival of a note and its attack. For each new note, all the calculations required to initiate this set of events take place the instant before the note is drawn onto the staff. It is worth noting that the single-instrument layout used here is well suited to accommodate the extended multi-staff system being employed.

Video Example 3.2

Comprovisador.client: single-instrument layout, in sync with green ball.

3.2.3 Rhythmic Exploration

In order to have synchronised attacks within the framework of proportional notation, there are at least two possible time conceptions that can be used: chronometric time and metronomic time. The former is suitable for a pulseless time-axis definition, yielding a more irregular type of rhythm but one that can, in theory, be more consequent to the soloist's contribution. The first approach that came to be developed within `Comprovisador` is based on this conception. It requires measuring every inter-onset interval[10] of the notes performed by the soloist and testing each against a specified threshold. Whenever the threshold is exceeded, a phrase ending is flagged and a synchronised attack is triggered as a response. The rhythm that results from the sequence of responses is therefore closely linked to the soloist's phrasing. Such rhythm tends to be irregular[11], especially so if the threshold value is being expressively modified during performance. Lower threshold values make the algorithm more reactive generating results that are denser and more intrusive to the soloist's discourse, leading to active rhythmic interactions. By contrast, higher values give more space for melodic expression.

The latter conception is achieved here by defining a metronomic period and subordinating the synchronised attacks to the resulting tempo grid[12]. Rhythmic input from the soloist is disregarded in this mode (although pitch is still considered, for harmonic purposes). Instead, he or she may eventually lean towards the imposed tempo by influence of the ensemble response. The advantage of this conception in terms of rhythmic exploration lies in the fact that different multiples of the metronomic period can be used interchangeably, thus generating[13] rhythmic phrases comprising diverse durations, or rather diverse inter-onset intervals, in regular pulse.

[10] One could use the term durations, but that would be imprecise for non-*legato* playing.

[11] – unless the soloist is making an undue effort to impose a steady pulse.

[12] See Section 3.3.2 for an explanation on the behaviour of the bouncing ball in this particular mode: In Sync with Green Ball (Grid) – Harmony, Variation 2.

[13] Indeed, this reading mode, or rather the algorithmic variation associated with it, has a more generative character then the previous one. The harmonic content of the output is still influenced by the soloist's input, but if at a given moment he or she stops playing, the algorithm will keep generating content based on the pitch set of the last input phrase. More on this in Chapter 5.

Video Example 3.3 below shows the notation interface in dual-instrument layout where each instrument belongs to a specific group. Both groups have independent rhythms (sets of inter-onset intervals), although linked to the same metronomic grid[14]. It should be emphasised that each group is bound to allocate several instruments playing together the same rhythm with different notes (synchronised attacks).

Video Example 3.3

Comprovisador.client: dual-instrument layout, in sync with green ball (grid).

The subdivision of a pulse is a rhythmic feature that can be more challenging to specify in proportional notation, especially when different durations and/or notes are used - as in a melody. In such cases, it might be more appropriate to use traditional notation (but perhaps see Motivic Exploration II below). However, splitting a note occupying one pulse into equal subdivisions (as when turning a crotchet into four semiquavers by applying two strokes across its stem) is not quite so difficult. By using colours and numbers in a way that is easily intuitive, it is possible to specify the number of parts into which a given note should be divided (like tuplets), ensuring cohesive responses from the ensemble (see Video Example 3.4). This number can, of course, be manipulated by the mediator in expressive ways. More so, using two groups, different subdivisions of the same pulse can be overlaid, resulting in polyrhythmic patterns.

Video Example 3.4

Comprovisador.client - tuplets example.

As I understand it, these rhythmic exploration resources (controlled gradation between space and density in a pulseless time-axis definition, rhythmic phrases in regular pulse, overlaid rhythms, tuplets and polyrhythmic patterns), reinforced by cohesive attacks, represent strong compositional cues for the listener. Furthermore, they constitute a colourful rhythmic palette available to the mediator.

3.2.4 Motivic Exploration

Apart from cohesive attacks and rhythmic exploration, other strategies can elicit the perception of a compositional process – one of them being motivic exploration. If the listener is confronted with a melodic fragment being played simultaneously by various instruments and/or transformed in a coherent manner, he or she

[14] Defining an independent metronomic period for each group, resulting in a polytempo effect, is also viable.

might perceive it as composition. Here, simultaneity refers to a given short time interval we perceive as present (specious present – see [Poi15]). It does not imply unison or homophony but rather polyphony and micropolyphony.

This textural procedure, if done with no concern for synchronisation and no obligation regarding meter or rhythm, allows musicians to serenely read the score and render the melody with lower probability of mistakes (research shows that most errors in sight-reading are related to rhythm [Zhu14, Wri05]) and arguably less anxiety. At the same time, a dense texture will help in disguising the occasional missed note. This led to the implementation in `Comprovisador` of a reading mode featuring proportional notation, a looping melodic fragment, a linear cursor and the label: "loop (non-sync)".

The following example (Video Example 3.5) shows the notation interface in dual-instrument layout. Each instrument is allocated to a specific group, each with independent contours (melodic fragments). A number of expressive transformations are applied to each contour – looping region (size and boundary position), playing rate, articulation and dynamics. Not featured in this example is the possibility of transforming the contour using techniques such as expansion, contraction and transposition – those will be demonstrated in Chapter 5 using snippets from live performances. It should be noted that each group is usually composed of several instruments. Here, they would play the same contour, perhaps in a different transposition, though without synchronisation, for the rate suggested by the green cursor is merely descriptive – and deliberately out of sync among clients. Towards the end of the example, the second instrument (guitar 1) is reallocated to algorithm Harmony (reading mode: "in sync with green ball"). It serves to demonstrate the scenario in which two players using the same client computer experience contrasting reading modes.

Video Example 3.5

Comprovisador.client: dual-instrument layout, loop (non-sync).

Motivic Exploration II

Prior to "Comprovisação nº 9", progress was made to resolve the conundrum of specifying rhythmic subdivision with melodic function in proportional notation (reading mode: "in sync with green ball"), yielding positive outcomes. The concept is founded on the idea of *acciaccatura*, in which a group of short notes that have no theoretical duration are executed immediately before the primary note. The size of the groups may vary, at the mediator's discretion, from one to eight notes, as shown in Video Example 3.6. Each *acciaccatura* is built on

the most recent melodic intervals played by the soloist and transposed so that the primary note aligns with the intended chord note (algorithm Harmony).

Thanks to the effective cueing devices, the sight-reading musician can predict the downbeat accurately and hit the primary note precisely. Although some inaccuracies may occur with the short notes played before the main note, these mistakes are hardly perceptible given the speed at which these notes are played. The aesthetic possibilities of this resource are explored in Chapter 5.

Video Example 3.6

Comprovisador.client - acciaccatura example.

3.2.5 Standard Rhythmic Notation

The ability to create rich and well-organised textures, incorporating melodic, rhythmic, and harmonic components, as well as formal elements such as repetition and variation, is desirable. Standard notation allows all that while adding two new levels of time: pulse and rhythmic subdivisions. The problem lies in the fact that the more elements are added, the more difficult sight-reading becomes and the more exposed musicians feel.

A progressive approach to the various elements may offer the answer. Let us imagine a musician learning a new piece of music: if they encounter a difficult passage, they might focus solely on the notes, repeating the passage several times until they are sure to play all the correct pitches; and only then will they try and play those pitches in precise rhythm and tempo. Emulating this process, when in `Comprovisador` standard rhythmic notation is activated, the notes that were previously displayed in proportional notation will be kept the same, enabling the performer with the chance to focus solely on the new element: rhythm. Conversely, once a particular rhythmic phrase has been introduced, melodic variations may occur while keeping the rhythm unchanged.

The reading mode that handles standard rhythmic notation through quantisation has been labelled “quantum loop”, a tongue-in-cheek reference to quantum mechanics[15]. Video Example 3.7 does not present the process of quantising a contour originally introduced in proportional notation. Instead, it focuses on a few rhythmic effects that can be achieved: two independent loops of different size, bound to the same pulse (polymeter); manipulation of the loop region boundaries; and agogics manipulation (polytempo). As with “loop (non-sync)”,

[15] “In physics, quantisation (...) is the systematic transition procedure from a classical understanding of physical phenomena to a newer understanding known as quantum mechanics.” [qua23]

melodic transformations using expansion, contraction or transposition are also possible (not shown here, as these will be discussed in Section 5.1.9).

Video Example 3.7

Comprovisador.client: quantum loop.

3.2.6 Instrumentation and Musical Form

In music, timbre is often considered one of the aspects that can most easily evoke an emotional response from listeners and, perhaps for this reason, has always been used by composers to contribute to the perception of musical structure. This is most evident when a piece originally composed for a single instrument, such as the piano, is orchestrated. Here, the composer might use different timbral combinations for different musical sections or even for different functions (e.g. melody versus accompaniment) within a section.

Instrumentation is often a major element in real-time systems that seek to organise collective improvisation, where it is used as a way of "shaping freedom in improvised music" [Gir18] (see also [Gou18]).

In the earliest phase of `Comprovisador`'s development (until its debut in "Comprovisação nº 1"), instrumentation was controlled stochastically by means of a single parameter – density. If this parameter was set to its maximum value, all ten available instruments in the ensemble would be involved in the next response; if it was set to, say, twenty precent, two of the instruments would be selected at random to play the response. Furthermore, the instrumentation was fixed (hardcoded). Still, despite the frugality of control options, some decisions in which musical form was emphasised by instrumentation were made prior to performance (i.e. composed) with interesting results (see Appendix B).

In the subsequent phases, various features were developed allowing for a detailed control of both instrumentation and form, notably, the possibility of:

predefining a new set of instruments for each session , with automatic configuration of parameters such as range, transposition (transposing instruments), and others;

predefining a musical form , by creating and storing a sequence of parameter **presets** – having the potential to produce different levels of texture contrast with each call of a preset, since instrumentation, expression parameters and, crucially, the type of algorithmic response are all subject to change at once;

controlling two independent groups with complementary musical functions – each group allocating separate algorithmic instances, and each of these allocating specific instruments;

defining instrument clusters within each algorithmic instance, enabling manual as well as automated switching between timbral sets (i.e. contrast) in consecutive responses of the same algorithmic instance (e.g. three flutes followed by two trumpets followed by two clarinets);

generating sung text through a stochastic, parametrically controlled algorithm, enabling singers to vocalise identical phonemes in a particular chord, resulting in a consistent choral sound;

communicating via text messages to a specific subset of instruments at once (e.g.: @strings > pizzicato; @woodwinds > frullato);

producing textural effects , e.g. *pointillism*, achieved with a special resource named **centrifugue** (a play on the words centrifuge and fugue);

performing expressive manipulation of various parameters (including the old *density*, now renamed **%densidade**), potentially resulting in a gradual transition between structural points in the musical form.

Examples of these possibilities shall be explored in Chapter 5. For a detailed review of the Control Interface and available parameters, see Section 3.4 below.

3.2.7 Practice Mode and Performers' Feedback

Development and enhancement of these and other features of `Comprovisador` was only possible thanks to the feedback of musicians who tested the system in rehearsals and performances. In this regard, feedback was also gathered from the performers' interaction with `Comprovisador`'s `Practice Mode`, which informed further refinements to the system based on the following parameters: reading time window, maximum melodic leap and note flux (inverse to inter-onset interval)[16]. As mentioned above, this tool was implemented as a way to enable performers to get acquainted with the system's notation interface and its idiosyncrasies – including its different reading modes and the bouncing ball as a cueing strategy. It featured a random walk algorithm capable of generating unpredictable note patterns and an elementary GUI for parameter control.

The `Practice Mode` was especially useful in situations where musicians and developer were in different locations. The tool allowed to obtain valuable feedback from a distance and perform bespoke enhancements

[16] A systematic approach to this information is discussed in Chapter C.

in time for the first rehearsal (see Section 4.2 – "Comprovisação nº 4").

Several musicians who tested the initial iterations reported that the `Practice Mode` was useful for developing sight-reading skills. This was seen as an opportunity to expand the use of the system into the field of education. As a result, I conducted two studies: one aimed to evaluate the improvement of sight-reading skills in a microtonal context among a group of professional saxophonists [Lou17c], while the other focused on students who played various instruments in a more general sight-reading context [Lou18d]. The latter is presented in Chapter 6.

At the same time, Joaquim Nascimento, a guitarist who played in "Comprovisação nº 3", asked for permission to use `Comprovisador`'s `Practice Mode` in his research for his Master's degree. His study focused on a group of young guitar students who used the `Practice Mode` to improve their instrumental sight-reading skills [Nas19]. A summary of the results is also presented in Chapter 6.

A complete redesign of the `Practice Mode`'s GUI was carried out, driven by Nascimento's request and my interest in adapting the system as a tool for music students. The main design consideration was to allow users to adjust the difficulty level of the algorithmic output according to their proficiency level. This was accomplished by adding further controllable parameters and permitting users to save their preferred settings. Details will be provided in Chapter 6.

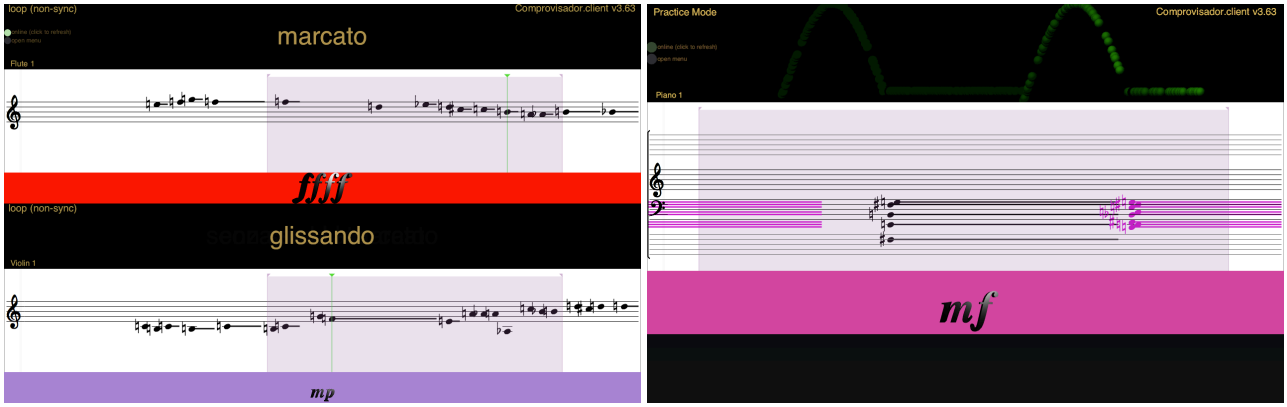
3.3 Notation Interface Design

3.3.1 Overview of the Notation Interface

As mentioned before, the notation interface was conceived in order to have one client computer for every two instruments, regardless of range or transposition of the instruments used. A single instrument per computer layout may be preferable in some cases.

Graphical objects in the interface adjust perfectly to all modern laptop screens, irrespective of their size, definition, and geometry, and of the instrument layout used. This is achieved using `JavaScript` inside `Max` to instantiate and position all graphical objects.

Comparing layouts of the two different configurations (see Figures 3.7a and 3.7b; also, see Video Examples 3.2



(a) Dual-instrument layout. (b) Single-instrument layout.

Figure 3.7: `Comprovisador.client`: dual- versus single-instrument layout.

and 3.3), it can be concluded that there are some advantages in single-instrument layout. On the one hand, multi-staves can be used, while on the other, both the dynamics bar (below the staff) and the direction bar (above the staff) can take on larger dimensions, ensuring faster information detectability and better legibility, being these good principles of graphical interface design [wik]. This space optimization was motivated by musicians' suggestions and was found to have a positive impact in performance.

The notation objects consist of a combination of `bach.roll` and `bach.score` objects. While the former renders proportional durations, the latter renders standard rhythmic notation [AG15].

The dynamics bar is a coloured bar rendered in OpenGL over which dynamics text is displayed. Again, regarding good principles of graphical interface design, both background colour and text size (3D space) change accordingly to the level of dynamics, in a reactive fashion. The colour that symbolises ***pppp*** is cyan and the one attributed to ***ffff*** is red. Any level in between will assume a proportional mixture of the two colours, maintaining the same perceived level of brightness. Concerning text, whenever the level is being changed, the words ***cresc.*** or ***dim.*** appear and move forward or backward in a three-dimensional space (see Figure 3.8). This feature is achieved using OpenGL (namely, Jitter object `jit.gl.text3d` [PZS+]. These reactive features were highly valued by musicians, as they reported being able to easily identify the dynamics level while maintaining full focus on the musical notes.

Figure 3.8: `Comprovisador.client`: dynamics bar – text size (3D space) and background colour.

Regarding the direction bar, it also features OpenGL graphics in order to render at a high frame rate (around 60

fps) a small bouncing ball which allows for musicians to synchronise attacks and play in a given tempo. Using the same OpenGL context, musical direction terms and other information are displayed. To ensure detectability, each new entry will pulsate in bright white, as demonstrated in various video examples in this chapter.

As illustrated in Figure 3.9, the motion described by the ball derives from a sine function returning absolute values. This type of function convincingly translates a motion of fall and ricochet, often used by conductors (see [Col12] or [Mei09]). This has been tested against other synchronisation strategies and musicians had a better response to the bouncing ball approach. Regarding the ball's fading trail (which was not present in the early versions of `Comprovisador`), testers reported that it facilitated the perception of the bouncing motion (preparatory beat) and the moment of impact (downbeat), especially when the object was not being looked at directly.

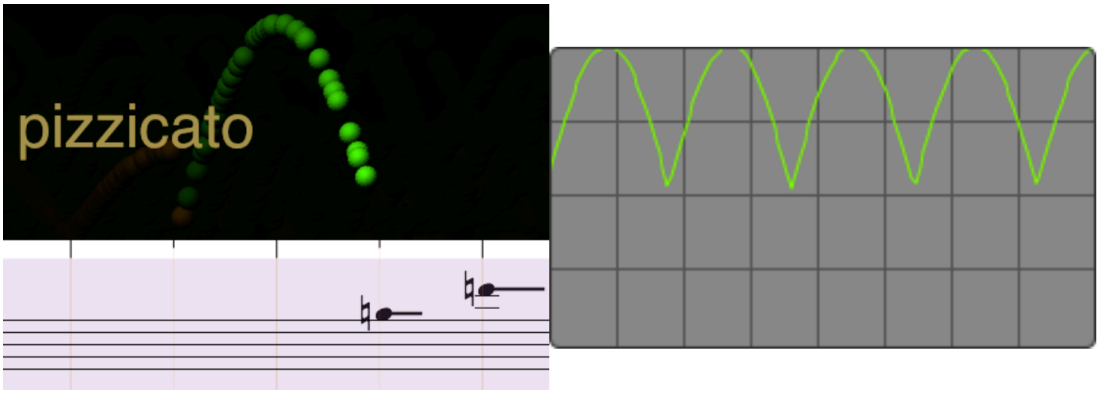


Figure 3.9: `Comprovisador.client`: direction bar (left) versus "folded" sine wave (right).

3.3.2 Reading Modes

As a result of the design considerations and aesthetic goals discussed in Section 3.2, four reading modes have been implemented. They correspond to the two variations of each of the two algorithms. Here is a summary of their characteristics (refer to the respective video examples below each list):

Mode 1: in sync with green ball – algorithm Harmony, variation 1.

- proportional notation;
- notes are written in real-time, from left to right;
- a bouncing ball shows the player when to play each note;

- the player should start each note precisely when the ball rebounds on the note;
- the length of the bounce (the period) adapts to each inter-onset interval that is less than or equal to one second (which is the default period); for greater intervals, the appropriate pre-bounce time is added[17];
- a two-click metronome (played through earphones) is synchronised with the bouncing ball – the first click coincides with the start of the bounce, whereas the second is played at the peak of the bounce, halfway through;
- a reading time window of about a second and a half[18] is calculated so that the player has time to read and prepare each note on their instrument;
- when a note or its duration line stretches off the play region (a fixed darker rectangular area which represents a domain of 5 seconds), it reappears at the beginning of the same play region – this feature replaces traditional page turning;
- notes that have already been played are erased in order to free staff space for new notes to be written;
- the ball will move horizontally over a long note's duration line, stopping at its end or bouncing to the next note, if there is one;
- for singers (refer to Video Example 3.1):
  - a cue note is played through earphones – it is synchronised with the second metronome click, aiding in melodic and rhythmic accuracy;
  - a second sound is played during the whole length of the written note, aiding in intonation;
  - text to be sung, made of rule-based generated phonemes, appears beneath each note.

Video Example 3.8

Comprovisador.client: mode 1 - in sync with green ball – (Harmony - variation 1).

Mode 2: in sync with green ball (grid) – algorithm Harmony, variation 2.

- the same as Mode 1 except for the fact that there is an underlying metronomic tempo for all attacks;
- a grid representing the underlying tempo is shown in the staff (see Figure 3.10);

[17] Refer to Figure C.1, in Chapter C, for further details on this.
[18] Other values can be assigned to the reading time window, with the default being 1600ms, as discussed in Chapter C.

- during long notes or rests, instead of moving horizontally or disappearing, the ball continues to bounce in tempo assuming an orange colour instead, while the vertical amplitude of its movement is reduced – thus, simulating a conductor's passive gesture of tempo keeping [GG04];
- whenever a response from the player is demanded (active gesture) the ball becomes green again and bounces higher (also, the metronome is activated);
- the reading time window will adjust to the nearest multiple of the metronomic period[19];

Video Example 3.9

Comprovisador.client: mode 2 - in sync with green ball (grid) – (Harmony - variation 2)

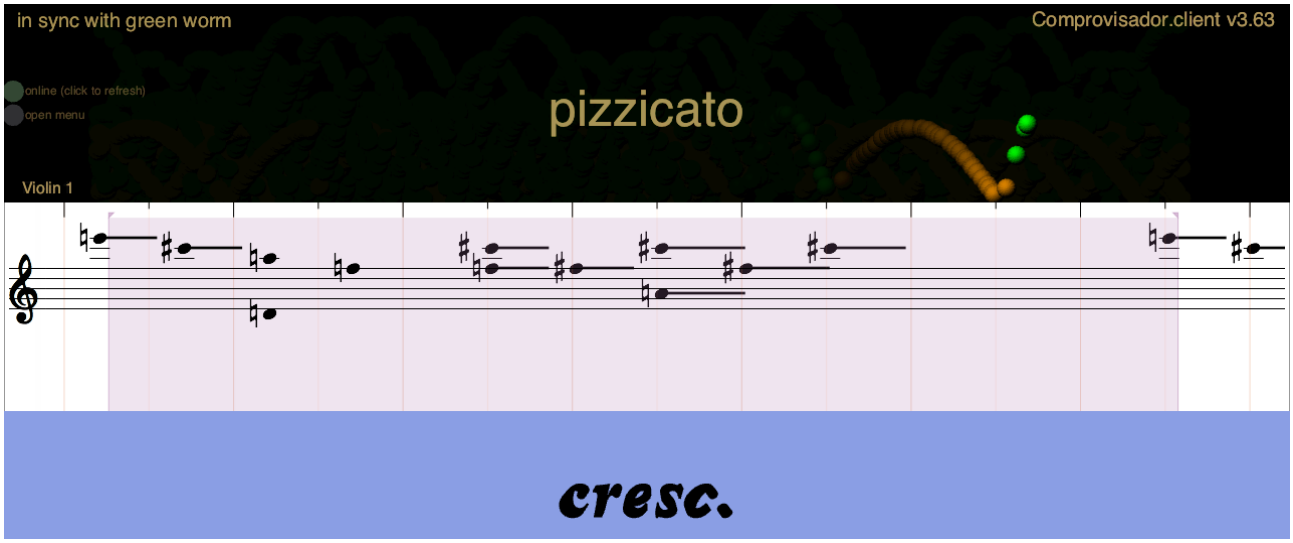


Figure 3.10: `Comprovisador.client`: orange ball (passive gesture) and grid (underlying tempo).

Mode 3: loop (non-sync) – algorithm Contour, variation 1.

- proportional notation;
- melodic contours appear, all notes at once;
- the player should loop through the notes framed inside the play region which in this case can be dynamically adjusted to any arbitrary portion of the displayed melody (refer to Figure 3.7a);
- a vertical green line (cursor) cycles through the play region so to give the player an idea of the intended playing rate, although to synchronise with the line is not mandatory;

[19] This allows the pulse to be synchronised across various algorithm instances (including the quantised variation of Contour) by employing the "sync" and "tap-tempo" functions, as mentioned in Section 3.4.4 below.

- above all, the player should not attempt to synchronise with their fellow musicians; in fact, there is an intended rate discrepancy in each client's cursor in order to help avoid synchronisation between players;

Video Example 3.10

Comprovisador.client: mode 3 - loop (non-sync) – (Contour - variation 1)

Mode 4: quantum loop – algorithm Contour, variation 2.

- standard rhythmic notation;
- quantised melodic contours appear, all notes at once, fitted in eight beats;
- the player should play in tempo with the green ball, which in this case aims at the beginning of every beat (instead of at every note);
- the player should loop through the notes framed inside the play region which can be dynamically adjusted to any number of beats;
- instruments allocated to the same control group (see Section 3.1.2) will always be in the same beat and in the same tempo – hence, they should play in sync[20];
- when two instruments are allocated to different control groups, one of three scenarios may arise (refer to Video Example 3.7):
    1. both instruments are in sync, although potentially playing different sampled contours – polyphony;
    2. both instruments are playing in the same tempo but performing different-sized loops – polymeter / isorhythm; or
    3. the two instruments are playing in different tempi – polytempo.

Video Example 3.11

Comprovisador.client: mode 4 - quantum loop – (Contour - variation 2)

Regarding note appearance, in response to requests from musicians, the colour of the duration line in proportional notation was altered to a translucent green, as presented in Figure 3.11 (made with version v3.85). This

[20] All instruments allocated to a given algorithmic instance within a given control group will have identical dynamics and, except in mode 3, will play synchronously, but not necessarily the same pitches (see Sections 3.4.3 and 3.4.4).

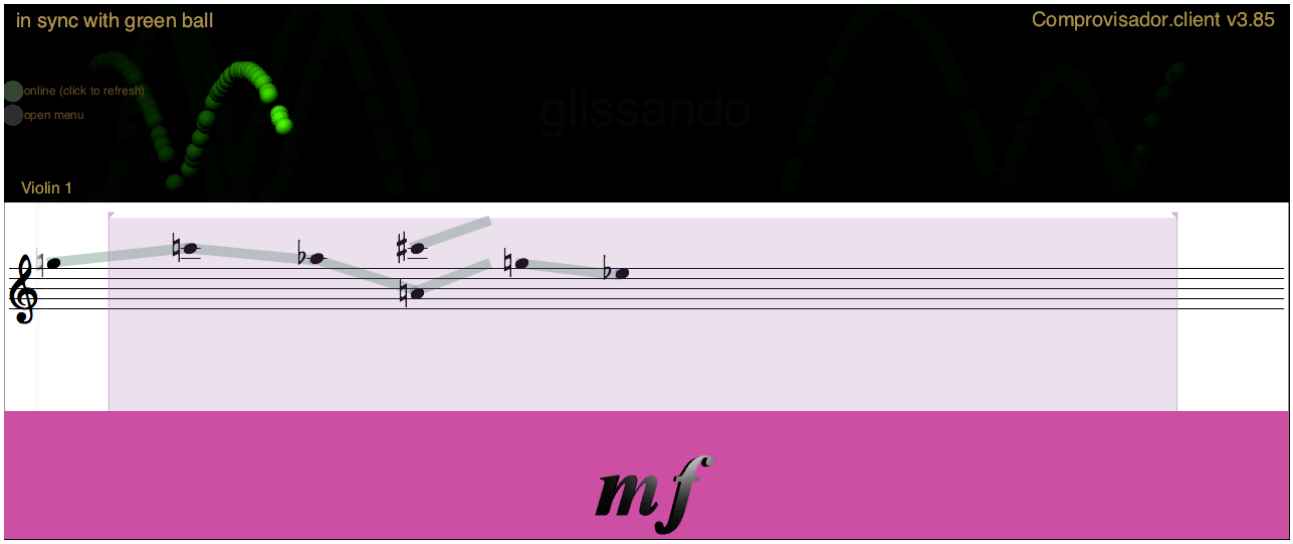
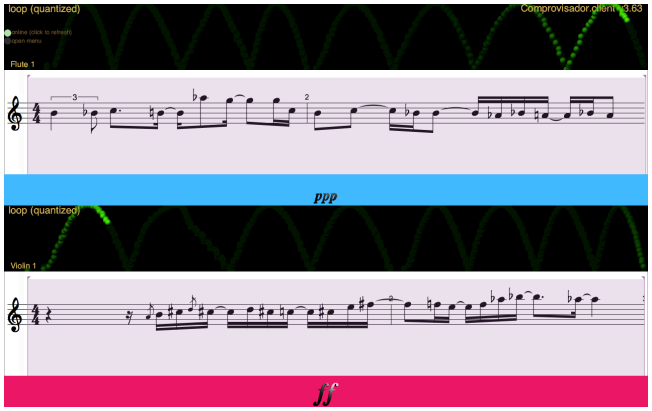


Figure 3.11: `Comprovisador.client`: In Sync with Green Ball (v3.85) – translucent duration lines with *glissandi*.

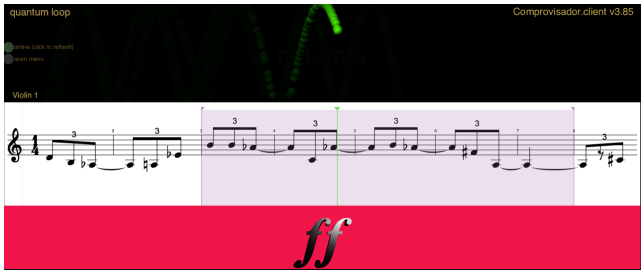
was carried out to minimise confusion with either staff lines or ledger lines, which can be observed in Figure 3.10 (rendered with version v3.63). In addition, the duration lines can now be slanted to indicate glissandi.

In regards to rhythm, the standard notation renderer has been enhanced at the quantiser level (version v3.85). Here, instead of writing two $\frac{4}{4}$ measures, as shown in Figure 3.12a, the algorithm writes eight $\frac{1}{4}$ measures, as demonstrated in Figures 3.12b and 3.12c. This allows two things: 1) complex patterns are conveniently delineated by bar lines, and thus easier to decipher (see Figure 3.13); 2) long notes unfold into tied crotchets, simplifying the process of counting the beats – which is crucial when a loop causes a long note to be truncated, causing confusion for a musician attempting to read it. This can be observed in Figures 3.14a and 3.14b: the former shows the loop region beginning in an ambiguous, white portion of the measure, clipping an uncertain duration of the start of a dotted minim; the latter presents the loop region beginning unequivocally at a specific bar line. Furthermore, it is now possible to select between quavers, quaver triplets and semiquavers as a rhythmic base (minimal note values) for the quantisation. Examples using triplets and semiquavers are shown in Figures 3.12b and 3.12c. As for the metronome, it is capable of adjusting to the rhythmic base being used and accounts for the tempo. At very slow tempi, musicians will hear all the pertinent beat subdivisions (semiquavers or semiquaver sextuplets). At moderate tempi, the metronome generates quavers or quaver triplets, and at faster tempi, it produces only beat units. Video Example 3.11 above demonstrates this feature, using a semiquaver base.

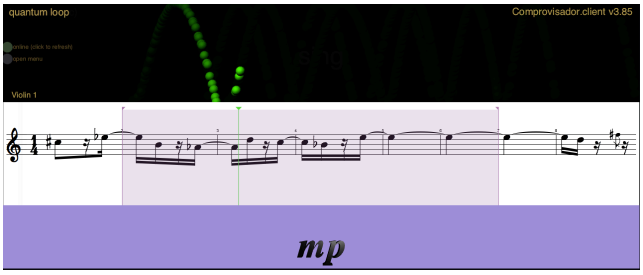
The notation interface of `Comprovisador` has benefitted from the concurrent development of the `bach` library. As new features became available in each iteration of the library, they were integrated into in the system. These encompass the ability to use articulation and ornament signs, together with an algorithm for respelling



(a) v3.63 – 2×$\frac{4}{4}$

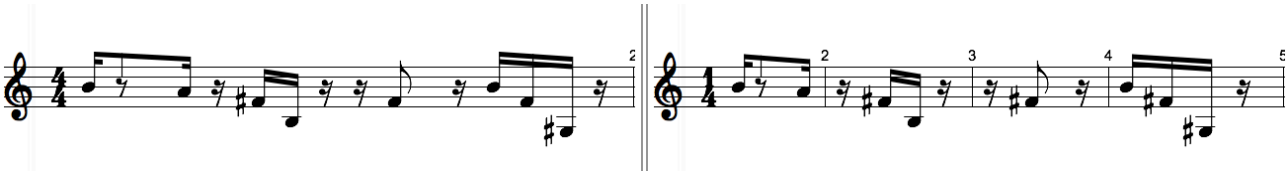


(b) v3.85 – 8×$\frac{1}{4}$ – triplets.



(c) v3.85 – 8×$\frac{1}{4}$ – semiquavers.

Figure 3.12: `Comprovisador.client`: Quantum Loop – older version (a) versus newer version (b and c).

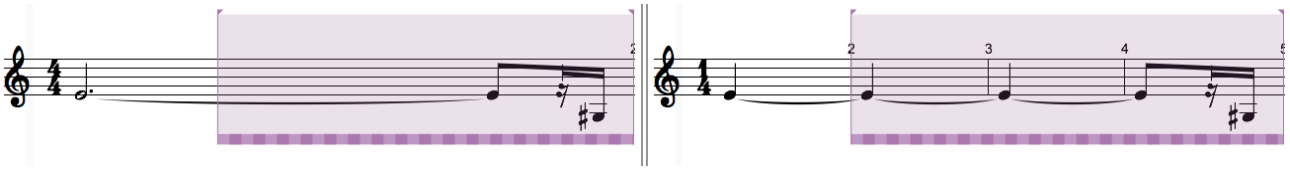


(a) in $\frac{4}{4}$ (original);

(b) in $\frac{1}{4}$ (new);

Figure 3.13: Quantiser enhancements: complex patterns conveniently delineated by bar lines (in $\frac{1}{4}$).

accidentals in a more musical way, especially in atonal contexts. It is reasonable to suggest that the collaboration between myself and the developers of `bach`, in the form of bug reports and feature requests, has led to improvements in the library that ultimately benefit the community of `bach` users. For example, the option to change the colour of a note's duration line is now available following my request and has been used by composers such as Jonathan Bell (see [BC19]). The respell algorithm has also been improved as a result of our collaboration.



(a) in $\frac{4}{4}$ – loop region starts at an uncertain point; (b) in $\frac{1}{4}$ – loop region starts at a specific bar line;

Figure 3.14: Quantiser enhancements: long notes unfolding into tied quarter notes (in $\frac{1}{4}$).

3.4 Algorithms and Control Interface Design

3.4.1 Overview of the Control Interface

An exploded view of `Comprovisador.host`'s Control Interface is shown in Figure 3.15. It consists of several blocks with designated functions. At the centre, there is a vertical block for the activation of algorithms and allocation of instruments. Algorithms are instantiated into four slots. The **instrument allocator** consists of a grid that has four columns, each of which corresponds to one of the four algorithmic slots. The number of rows in the grid corresponds to the number of instruments that are specified in the initial configuration. The mediator can adjust the switches in the grid as needed, to allocate each instrument to the appropriate slot. The square objects with ovals enclosed within (commonly known as `nodes`) are probability weighting objects that offer various means for controlling the instrumentation[21] , within an algorithmic instance. The **part unifier** feature enables the selection of instruments to play the same part (doubling), making it particularly useful for choir or string sections. It allows for up to four distinct parts to be specified (instruments selected in the same column belong to the same part).

To the left lies the `control surface mirror` comprising various knobs, sliders and buttons that offer graphical feedback (i.e. mirroring) for all commands performed on the external control surface. These graphical objects also accept direct interaction with the mouse.

There are two independent control groups, each controlling two algorithmic instances simultaneously. For instance, modifying the parameter **nodesize.1** (fader 4) will influence the size of the nodes in `nodes 1` and `nodes 2` objects simultaneously (group 1), while **nodesize.2** (fader 8) will impact `nodes 3` and `nodes 4` (group 2).

Located at the bottom left corner is a **preset manager**, which allows parameter presets to be stored and recalled. The left and right arrows on the external control surface facilitate navigation through the presets. Next to the

[21] See below a summary of the instrumentation control.

preset manager, there is a cluster of **multislider** objects that allow control over probability weights. There are two **multisliders** above that handle the weights of rhythmic multiples (refer to Harmony algorithm, variation 2) and three below that handle the weights of different constituents of the sung phonemes: the opening consonant, the vowel, and the closing consonant.

On the right-hand side, there is the **reference score**, which has been introduced to monitor algorithmic procedures. On the left border, there is a vertical **progress bar** which includes a concealed **musical form script**. It becomes visible by clicking on the progress bar or typing "t". This script or text score comprises a list of instructions accompanying each saved preset. The number linked with each preset/instruction is constantly visible on the progress bar, even if the remainder of the script is undisclosed, allowing performance time and musical form to be kept under control. The intended duration of the performance can be specified in minutes within the proper setting, which influences the rate of the progress bar.

Other elements in the main window of **Comprovisador.host** include access to the system configuration and a multitrack audio/data recorder with a minimalistic GUI. Additionally, the **messenger module** is available by swiping with two fingers on the trackpad, although it is unnecessary when using a tablet and thus omitted from the image. The recorder is capable of producing recordings of multiple audio tracks, while synchronously capturing all outbound network data, for subsequent analysis. This allows for later re-rendering and analysis of the scores.

Following is a summary of the instrumentation control (relevant terminology in italics):

- there are two control groups, each with two slots for algorithms;
- there are[22] two separate algorithms that can be instantiated in any of the four slots, which run in parallel;
- instruments can be ***allocated*** to a specific instance of an algorithm and, by inheritance, to one of the two control groups;
- allocated instruments may be considered ***available to play*** only if a probability weight greater than zero is attributed to them (through the **nodes** object[23] , in conjunction with the **%densidade** parameter – see Figure 3.17);

[22] – currently.

[23] Each *node* within the **nodes** object represents an instrument, with the probability weight increasing as the crosshairs move closer to the centre of the node. If the crosshairs are positioned outside the node boundary, the probability weight is zero. It is worth noting that the size of the nodes is affected by modifying the **nodesize** parameter, which, in turn, impacts the relative distance between the crosshairs and each node centre, thereby affecting the weight distribution.

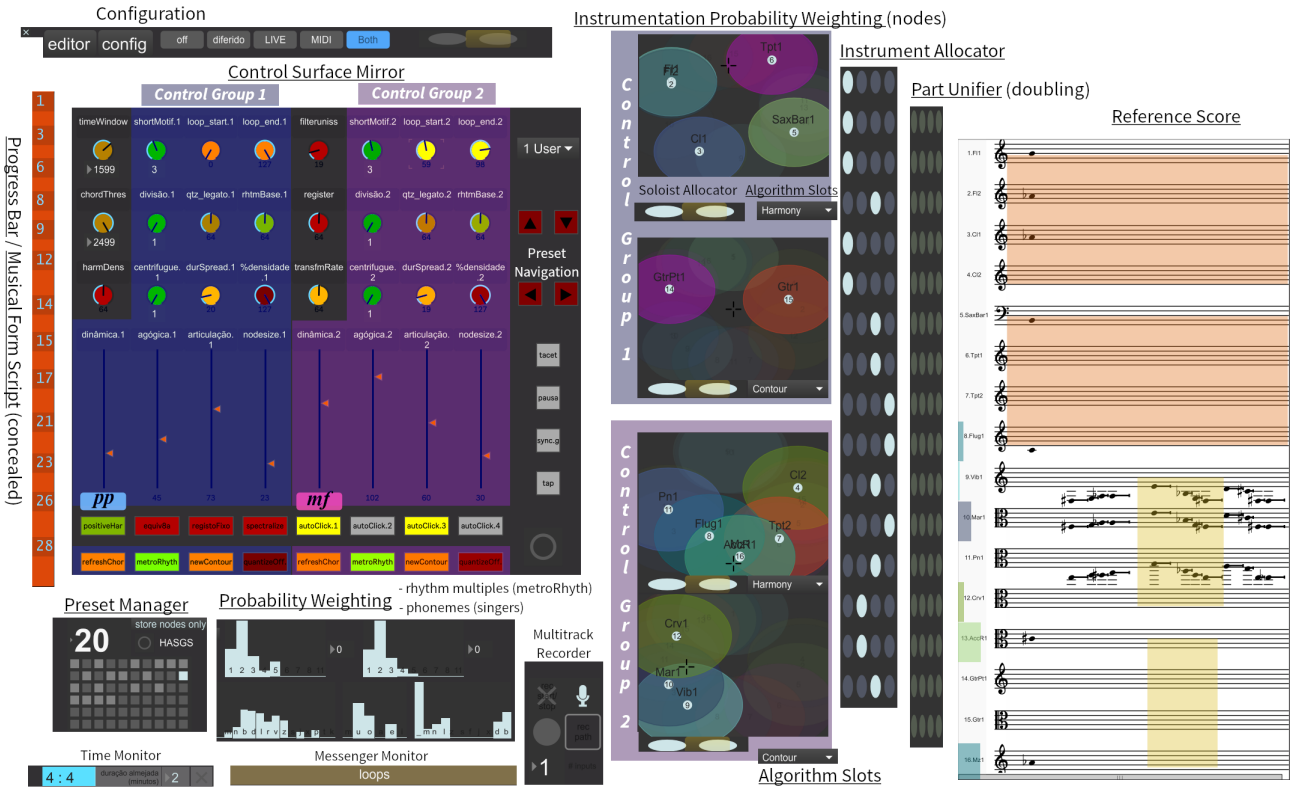


Figure 3.15: `Comprovisador.host`: control interface – exploded view.

- any instrument that is available to play can be ***assigned to play*** by the algorithm on output;
- if the **%densidade** parameter is set to its maximum value, all instruments that are available to play (i.e., all those selected via the `nodes` object[24]) will automatically become ***assigned to play***.

3.4.2 Parameters and Procedures

In addition to selecting a particular variation of an algorithm, there are several parameters that can be manipulated in real time (via the control surface) to produce different musical outcomes. Each controllable parameter can be saved within a preset for later recall at the touch of a button, allowing a wide range of contrast levels in musical transitions as well as firm control over musical form (see **preset manager** block and **preset navigation** buttons, in Figure 3.15). Furthermore, the preset manager can dispatch targeted messages to players upon preset triggering. This resource is highly effective in prescribing a contrasting attitude.

By curating a sequence of meticulously designed parameter presets, the mediator can effectively plan a musical

[24] The `nodes` object allows the mediator to apply different probability weights to different instruments. In effect, when **%densidade** is at its maximum, `nodes` acts as an instrument selector, excluding any instrument with a probability weight of zero.

(a) left-hand fingers on faders 1–4 (control group 1); right-hand fingers on faders 5–8 (control group 2);

(b) left-hand fingers on knobs (group 1); right-hand fingers on buttons (group 2);

Figure 3.16: Control surface unit (Novation Launch Control XL) with mediator's hands.

form, prior to performance. This is an act of **composition** in and of itself. In contrast, during live performance, manipulating parameters in real time is closer to **improvisation**, and triggering presets, when done according to the **musical form script** and following the **progress bar**, can be seen as **interpretation** (see discussion on Composition versus Improvisation, in Section 2.1, and on Real-Time Composition, in Section 2.3.2).

Regarding the configuration of the control surface, it was necessary to come up with a layout that would be both practical and efficient. This layout would have to provide for a balanced and intuitive way to control expressive parameters in any performing context, considering all possible combinations of active algorithms. Since the chosen control surface[25] has 8 well-sized faders (which are ideal to operate gradual transitions between parameter states) and 16 conveniently placed buttons (which are perfect for alternating between states or triggering transformations) (see Figure 3.16), the solution for this was to assign half of those controllers to control group 1 and the other half to control group 2. Therefore, faders 1 to 4 (left-hand side) control parameters named **dinâmica**, **agógica**, **articulação** and **nodesize** (related to dynamics, agogics, articulation and instrumentation) of group 1 while faders 5 to 8 (right-hand side) control the same named parameters of group 2. Figure 3.17 illustrates the correspondence between parameter names and their respective knobs, faders, or buttons, as aligned with the `control surface mirror` layout.

It is important to point out that each group may control two different algorithms at the same time (hence the

[25] – Novation Launch Control XL. Section 4.1.1 covers the reasons for acquiring this unit.

efficiency of this layout), although it is possible (and maybe sensible) to activate only one of them at a time, in each group. It should also be noted that musical parameters of different algorithms are in fact different procedural parameters and may behave slightly differently, despite having the same name. That being said, **dinâmica** will always impact dynamics, **articulação** will always control the relative length of notes, and **agógica** will always affect the rate of events. The nuances of this fader's functionality are detailed in Sections 3.4.3 and 3.4.4.

3.4.3 Algorithm Harmony

Generally speaking, Harmony generates chords from the notes played by the soloist in his or her last musical phrase [Lou17e]. The notes of the generated chords are automatically distributed to the assigned instruments, written to their corresponding notation interfaces, and played subsequent to the allotted reading time window.

There are two approaches on how the soloist's notes are selected and then recombined to generate chords. If the **positiveHarm** button is set via the control interface[26], new chords will be generated from notes that were recently **played** by the soloist. On the contrary, if **negativeHarm** is chosen, new chords will be generated from notes that were recently **avoided** by the soloist.

The global parameter[27] **harmDens**, which refers to harmonic density, enables variation between incorporating all input notes and filtering out some before executing the algorithm to construct the output chord. It does not affect the number of assigned instruments; instead, for every discarded note, it evenly replicates a retained input note.

When constructing a chord, three modes of register transposition are available. Firstly, setting the toggle[28] **equiv8a** allows for notes to be transposed one or more octaves to fit the range of the assigned instrument. Secondly, setting the toggle **registoFixo** means that notes will not be transposed, resulting in any notes beyond the assigned instrument's range being filtered out. Finally, the default register transposition mode (applied when no toggle is set) involves finding the modulus of the soloist's phrase between its extreme notes. When in the default mode, all transpositions adhere to the most recent modulus, which is subject to change with each phrase.

[26] For every named parameter or function mentioned in this section, refer to Figure 3.17.

[27] Global parameters affect both control groups simultaneously. These correspond to the knobs in the first and fifth columns, the top row of rectangular buttons, as well as the buttons located on the right-hand side. Refer to the colour coding in Figure 3.15.

[28] Some interface buttons operate as toggles, switching between states when pressed. Other buttons have a momentary behaviour such as triggering an event when pressed.

Figure 3.17: `Comprovisador.host`: control interface; detail of the `control surface mirror` block with the names of all parameters.

To achieve a smooth voice leading between chords, transposition of a note for a given instrument will always take into account the register of the preceding note played by the same instrument. To ensure variety, sudden changes in the overall register are feasible. This task can be accomplished through manual flipping of the **register** knob or automatic adjustment in response to the soloist's most recently played pitches. However, the automation feature is subject to a threshold specified by the **transfmRate** knob, which can suppress it altogether.

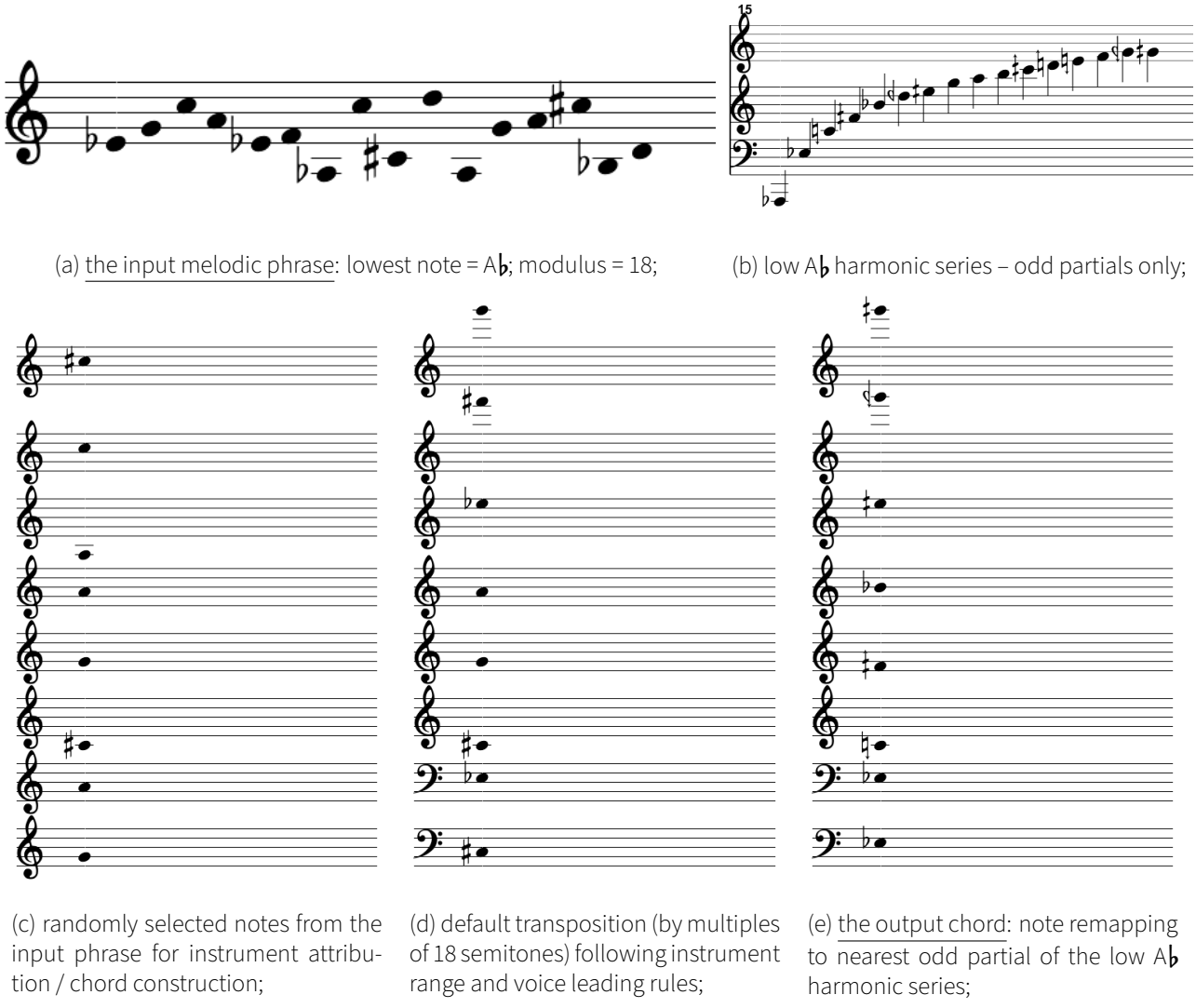
Still on the subject of voice leading, a recent development has been implemented to avoid melodic unisons when a short leap alternative is possible (e.g. when the modulus is small). Without this feature, there was a risk of excessive note repetition, leading to a monotonous effect. The leap threshold may be regulated using the **filteruniss** knob.

For singers, the algorithm generates a text to be sung, consisting of rule-based generated phonemes. Each phoneme is linked to the chord generated simultaneously. Consequently, all voices will sing the same phoneme for every given chord.

During the preparation of "Comprovisação nº 7", a new development permitted the incorporation of micro-tonality in the form of spectral chords, performed by choir singers. Construction of these chords is attained by remapping each transposed note to the nearest odd partial of the lowest input note, subsequent to its transposition to the lowest available octave, as illustrated in Figure 3.18. This mode is activated by setting the **spectralize** toggle.

Switching between the two variations of Harmony is done by a toggle. When it is set to **threshRhythm**, new chords are automatically triggered at the end of a phrase, according to the threshold set with the fader **agógica**. The fader **articulação** regulates the duration of chords. When set to the maximum, a chord will last until a new chord is initiated, thus producing a legato effect. For the **agógica** parameter, when set to its highest level, the threshold will be infinite, i.e. no new chords will be triggered. The use of the infinite threshold in conjunction with **articulação**'s legato setting enables chords to potentially have infinite duration. Notwithstanding, at any point, the mediator can manually trigger a chord using the **refreshChord** button, bypassing the threshold. More importantly, within this infinite threshold setting, upon clicking an instrument node or cluster of nodes, a chord is instantly output for the selected instruments, effectively transforming the `nodes` object into a versatile instrumentation interface. Additionally, halting or dismantling a chord can be accomplished by pressing the **pausa** button (a global pause function) or deselecting instruments (all at once or one at a time) within the `nodes` object.

When the variation toggle is in **metroRhythm** state, a metronome is turned on forcing all attacks to adhere to



(a) the input melodic phrase: lowest note = A♭; modulus = 18;

(b) low A♭ harmonic series – odd partials only;

(c) randomly selected notes from the input phrase for instrument attribution / chord construction;

(d) default transposition (by multiples of 18 semitones) following instrument range and voice leading rules;

(e) the output chord: note remapping to nearest odd partial of the low A♭ harmonic series;

Figure 3.18: Algorithm **Harmony** – spectral chord generation procedure.

it. Rhythmic durations are then calculated according to an editable set of probability weights (see Figure 3.15, probability weighting – rhythm multiples). The rate of the metronome is set with the fader **agógica**. The percentage of the rhythmic durations that contain sound, or relative duration, is modified using the fader **articulação**, in combination with the knob **durSpread**. The latter, as its name implies, allows variability in the relative duration of consecutive chords.

3.4.4 Algorithm Contour

The algorithm Contour captures the last musical phrase of the soloist and, after procedures of truncation, filtration and register transposition have taken place, writes the phrase in the notation interfaces of the assigned

instruments. The written phrase is to be played in loop and may undergo transformations, such as contraction or expansion, subject to a threshold specified by the **transfmRate** knob, which has the ability to inhibit the occurrence of such transformations. These automatic transformations are always consequent to whatever the soloist meantime played. Furthermore, the mediator can use the up and down buttons on the control surface to trigger a transposition of the contour by, a random interval of up to 3 semitones.

The default modulus-based register transposition approach is also used in Contour algorithm, when generating a new phrase or transforming an existing one. It is important to distinguish between two types of transposition: the transformation that shifts the original contour slightly for the sake of musical interaction and the register transposition that adjusts the contour to the range of the designated instrument.

Rhythm quantisation

Variation 2 of this algorithm (triggered by the **quantizeOn** toggle) consists of the quantisation of a captured phrase, which allows musicians to play in sync. If a given phrase is quantised, its original notes remain the same. Players can thus focus entirely on the rhythm, as they are familiar with the notes already. As per the latest system update, durations are fitted into eight $\frac{1}{4}$ measures (refer to Figures 3.12b and 3.12b). The minimum rhythmic values for quantisation can be selected from quavers, quaver triplets and semiquavers (via parameter **rhtmBase**).

Further melodic transformations may continue to occur in this variation as well, subject to the **transfmRate** threshold.

In both variations, the fader **agógica** sets the loop rate (playing speed / tempo). Additionally, a tap-tempo function enables the synchronisation of the pulse across all algorithms. Two taps on the **tap** button (refer to Figure 3.17) will establish the period/tempo for all slots, regardless of the algorithm instantiated in each. This action automatically selects variation 2 for all slots – i.e. **metroRhythm** and **quantizeOn** for Harmony and Contour respectively.

In the outlined quantisation process, it is important to note that while the rhythmic quantities of the original phrase are to some degree preserved, the rhythmic qualities[29] may end up fairly distorted. This is because tempo information is not taken into account when capturing the original phrase. Rather than a issue, this is an aesthetic choice.

[29] – cf. [Boc06]

Due to problems identified during performance with **Comprovisador** (see Section 5.1.9), a new and improved method for handling standard rhythm notation is presently under development (see Section 7.1.1).

In algorithm Contour, there is currently no means of producing sung text in either of the two variations. A straightforward strategy has meanwhile been devised, although it has not been put into action yet. The strategy involves assigning a single phoneme to every note within the contour. Phoneme changes and melodic transformations would occur simultaneously.

3.5 Technical Implementation

3.5.1 Network and Login

With regards to networking, three approaches are in place to handle different situations: UDP – for data needing low latency and high throughput (but where occasional dropped packets are tolerable); TCP – for data needing to be transferred reliably (but tolerating an occasional delay); and multicast UDP (using the Java class **net.maxhole**). The latter is used only at the beginning of a session in order for the host computer to announce its IP address to all client computers and to query theirs, as suggested by Ben Nevile [Nev06].

The instrument configuration window presents users with a few drop menus where they can choose their instrument, starting by family. All the other drop menus automatically assume the normal values for the chosen instrument (e.g. treble clef and E♭ transposition for Alto Saxophone). If players one and two choose the same instrument, different player numbers are automatically assigned. However, if more players are using the same instruments in other client computers, a decision has to be made regarding player numbering.

When users press the **Confirm** button, each client sends its IP address and port number(s) (for a single instrument or a couple of them) to be gathered by the host. Port numbers are specified according to the table shown in Appendix A.

3.5.2 Startup Auto-Configuration

Along with the initial IP address gathering, many aspects of the host application are automatically configured on startup, by means of a script that looks up an instrumentation list in crossed-reference to an instrumentation

dictionary, both stored in text files. The former consists of a simple list of the instruments to be used in a session while the latter contains a large set of information specific to each instrument (family, range, transposition, clef, dynamic range mapping, strings tuning, initial IP port number, etc.). This allows for the system to be flexible, yet idiomatically correct, in terms of instrumentation.

4

Iterative Design

4.1 From Concept to Performance

In the early plans made for `Comprovisador`, which date back to mid 2014, a control interface was not under consideration – at least, not for the purpose of real-time interaction. Instead, the original concept was to implement a complex logical system that would be able to change algorithmic parameters according to predefined musical events. Those events would be triggered by the improviser (for example: by playing a specific note sequence) and, for each new performance, a new set of rules would have to be programmed (in order for it to be considered a different piece).

The downside of this concept was the fact that it wouldn't be flexible enough to cope with the unpredictable nature of improvisation. The clash between the rigidity of such a system and the spontaneity of an improviser

could potentially translate into problems in the development of musical form. As discussed in Section 2.3.2, Eigenfeldt considers the systematisation of form a complex task which justifies the incorporation of human interaction in various MuMe systems. Therefore, the necessity for human mediation during performance was recognized and consequently, the concept of a control interface was developed. Hence, the need for a human intervention during performance was acknowledge and so the idea of a control interface came about.

A rudimentary version of **`Comprovisador`** was used in a live performance entitled "Comprovisação nº 1" which took place in 18 May 2015 at the Escola Superior de Música de Lisboa (Lisbon College of Music) (ESML), featuring Ricardo Toscano – saxophone, the author – direction, and Camerata Silva Dionísio – a 10 piece woodwind ensemble (see Figure 4.1).

Figure 4.1: "Comprovisação nº 1": performance, Composition Week, ESML, 18/05/2015.

In Figure 4.2, we see the early design of the control interface as used in "Comprovisação nº 1". A hardware control surface was used for precise, tangible operation, with the advantage of enabling simultaneous parameter manipulation, while the graphical interface provided visual feedback. At the top right, a tab containing five buttons was used for algorithm activation. The five available algorithms were named *contour*, *harmony*, *klangfarben*, *sombra* and *finalChord*. The characteristics of each of the algorithms are explained later in Appendix B. Each algorithm automatically triggered a predefined directive, although it was possible to write a different one at any moment.

In the center right block, five sliders were used to manipulate expressive and algorithmic parameters as follows:

- *dinâmica* – to manipulate the dynamics level;

- *agógica* – to manipulate the rate of events;

- *articulação* – to manipulate the relative length of notes;

- *densidade* – to control the density in terms of how many (but not which) of the 10 instruments would actually play (randomly chosen);

- *comprimento* – to control the length (in number of notes) of a given melodic contour.

The bottom right block contained three buttons that were used as follows (from left to right): to realize a new chord, on demand; to create a momentary silence; to alternate between the type of harmonic field (see *positiveHarm* / negativeHarm in Section 3.4.2).

The interface also featured a recorder, a timer and a messenger block containing a few predefined musical direction terms.

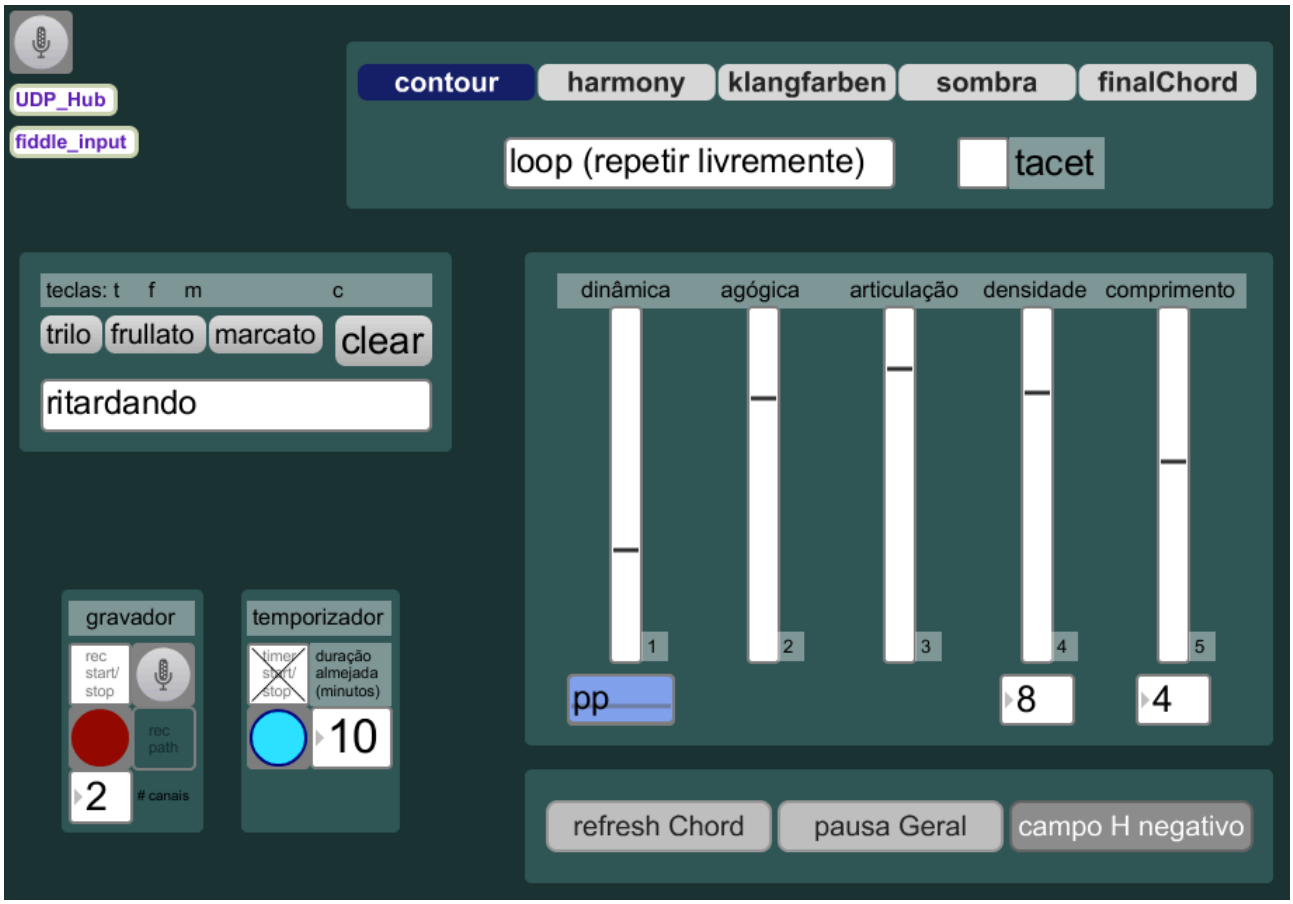


Figure 4.2: `Comprovisador.host`: early version's control interface as used in "Comprovisação nº 1".

Despite the simplicity of both system and interface, it was possible to carry out an interesting performance that went beyond demonstrating proof of concept. Moreover, the performance allowed for a reflection concerning the aspects needing improvement so to render the system more flexible, more reliable and more musical. Hence, a set of developments was planned out (please, refer to the last table of Appendix B), implemented over time and tested during the subsequent performances.

4.1.1 Developments

Many of the developments made since "Comprovisação nº 1" did not concern control interface aspects (for example: functioning of algorithms, networking issues, notation interface layout, etc.) and will not be addressed here. Instead, only those having an impact on the system's control capabilities will be listed.

Flexible instrumentation. The control interface and most of the system's modules were rethought in order to accommodate the possibility of flexible instrumentation – as opposed to the fixed, hardcoded instrumentation approach of "Comprovisação nº 1". The solution found was an auto-configuration script approach based on instrumentation list and dictionary discussed in Section 3.5.

Instrumentation control. Along with flexibility of instrumentation (in terms of system configuration) came the need and opportunity to implement a way of controlling instrumentation from the performance standpoint. A set of graphical objects was laid out to facilitate control over instrument assignment.

Parallel algorithm running. Instrument assignment led to multiple algorithms running side by side, with specific parameter settings and instrument groupings. The control interface had to account for dedicated controls for each instance of each algorithm.

Algorithm variations. The original five algorithms were streamlined into two. Variations as well as new controllable parameters were developed for each of the two in order to obtain a wider range of possible musical responses. The algorithms and their variations can be outlined in a summarized manner:

- Harmony (algorithm) – generates chords from notes played by the soloist; output: proportional notation, each note / chord written in real-time; players should synchronise each attack with a bouncing ball (see Section); a reading time window is factored in for each new note / chord;

  - threshRhythm (variation 1) – notes / chords are output at the end of a musical phrase, after a set

threshold;

- metroRhythm (variation 2) – there is an underlying metronomic tempo for all attacks; a grid representing the underlying tempo is shown in the notation interface;

- Contour (algorithm) – captures melodic contours as played by the soloist and applies transformations; output: all notes at once; contours should be played in loop, taking a dynamic loop region into account;

- quantizeOff (variation 1) – output: proportional notation; players should *avoid* synchronising with others;

- quantizeOn (variation 2) – output: standard notation, players should *try* synchronising with others;

Reference score This important debugging tool was implemented as a way to monitor the output of algorithmic procedures.

Control surface integration. Contemplating the pros and cons of a flexible interface design that could be configured to work with any type of control surface versus an integrated design where a specific unit must be used, a decision was made towards the latter, given the advantages in terms of layout efficiency (refer to Figures 3.15 and 3.17). After careful consideration, a control surface that fitted `Comprovisador`'s needs was acquired – the Novation Launch Control XL. The key factors were layout and number of controls as well as LED light indicators (programmable via `Max` [LCX14]).

Control groups. Regarding layout, as explained in Chapter 3, two distinct control groups were created enabling simultaneous control over four algorithmic instances.

Routing enabled messenger. With this tool, it is now possible to quickly choose a group of recipients (according to instrument family, for example, or to algorithm assignment) and send them a message (an instruction – refer to Figure 3.5). The messenger window is also a Miraweb [Ben16] server which means it can be controlled from any kind of multitouch device.

Preset manager. Finally, a preset manager was created in order to facilitate control over form and readiness for interaction.

4.2 Milestones

Since 2015, `Comprovisador` has been used in public performances in eleven different occasions, involving upwards of 120 musicians.

Each performance has been preceded by development stages and short periods of rehearsal. Development choices and feature enhancement were only possible thanks to the feedback of musicians who tested the system in these real-world situations. In Comprovisação nº 5 (see below), the rehearsal stage spanned over a four-month period of weekly rehearsals with an ensemble consisting of 12 ESML students (flute, oboe, alto saxophone, tenor trombone, bass trombone, tuba, electric bass, marimba, piano, two singers – soprano and mezzo – and violin) and served as a test field for ongoing development with frequent discussion over musician experience.

It is fair to say that every public performance has served as a developmental milestone, every time leading the system towards its maturity. Following bellow are a few examples of such milestones as well as problems that were identified and opportunities they created for further development.

Comprovisação nº 1

May 2015, Composition Week, ESML (Portugal). This first performance was important for demonstrating proof of concept. Five compositional algorithms were designed, among which 'Harmony' and 'Contour'. The other three were later deemed unnecessary, as these two were further developed. The system was initially designed for a fixed, "hardcoded" instrumentation and featured limited control over algorithmic parameters.

The notation interface contained two side-by-side notation viewers (serving a pair of musicians from each laptop) which were found to be too narrow to accommodate longer musical gestures (see Figure 4.3). Also, a note scrolling approach was used but came to be uncomfortable for reading due to jitter – musicians reported.

Comprovisação nº 2

May 2016, Composition Week, ESML (Portugal). The longest period of time between two performances ran between the first and the second, during which a great deal of changes to the system took place. Those changes enabled flexible instrumentation through auto-configuration features (see Section 3.5.2) and increased control over algorithmic parameters. Wide notation viewers (on top of each other) were implemented and the note

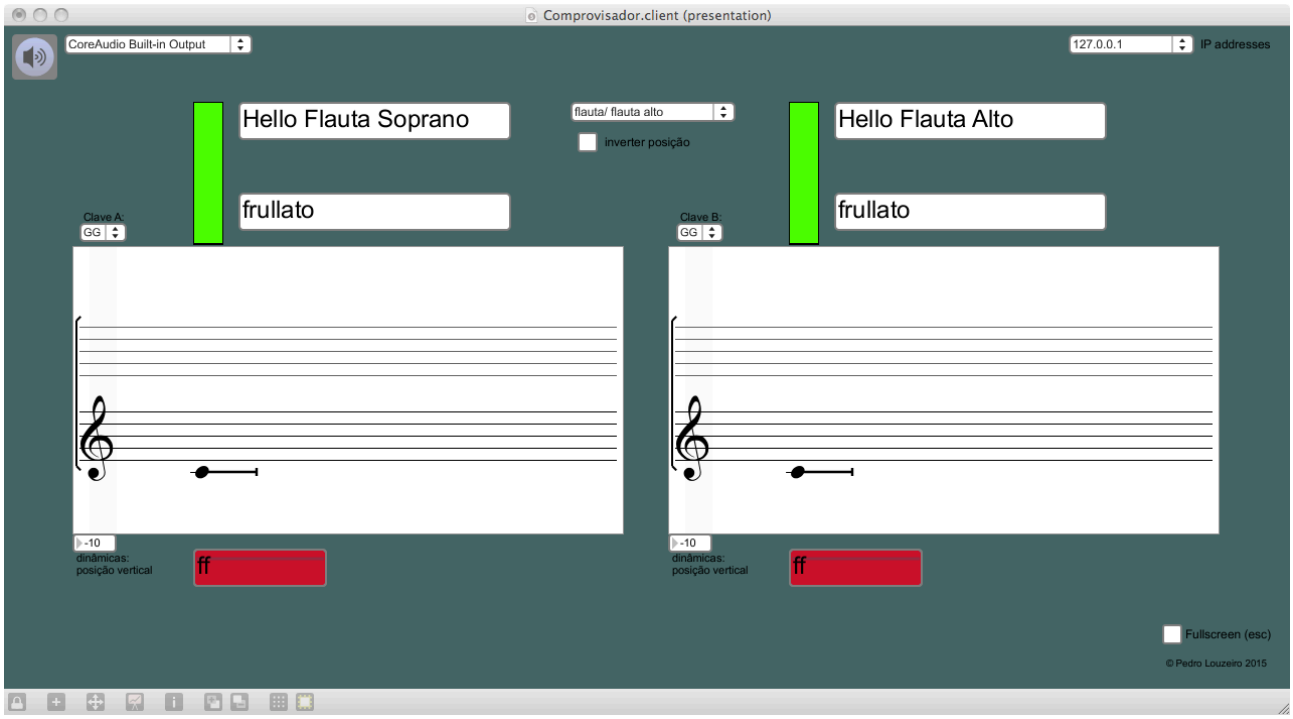


Figure 4.3: `Comprovisador.client`: old notation interface (as of Comprovisação nº 1).

scrolling approach was replaced by one where the notes wrap around to the beginning of the staff. A visual synchronisation strategy consisting of a bouncing ball (discussed in Section 3.3) was implemented using the `drawsprite` feature of `Max` object `lcd`.

During the performance, some stability issues were brought out by the large number of participating musicians (30) and client computers (16). The causes were discovered later. The lack of a preset mechanism to help in coping with the increase of controllable parameters was also felt.

Comprovisação nº 3

2016, Festival Música Viva, O'Culto, Lisbon (Portugal). A rudimentary preset system and slightly better control over rhythm were set in place. The solo xylophone performance was subjected to electro-acoustic manipulation of the instrument's sound by a dedicated performer (José Luís Ferreira) using a discrete system.

Comprovisação nº 4

2016, Sound and Music Computing Conference, Kampnagel, Hamburg (Germany). For this performance, parallel algorithm running and separated control groups were implemented. These features allowed for richer and better structured musical textures by assigning specific instruments to specific musical functions (for example, strings playing long ***mp*** chords and woodwinds playing *staccato* ***ff*** melodic fragments).

Algorithm Contour was upgraded to take durations into account and other improvements which made it possible to capture and display longer musical phrases, taking full advantage of the wide notation viewer.

Other novelties consisted of harmonic notation (for piano, strings and vibraphone) and multi-percussion specific notation.

Fine-tuning of this features was done taking musicians' feedback into account. The `Practice Mode` was especially useful in this situation since the musicians and the developer came from different locations. The tool allowed to obtain valuable feedback from a distance and perform bespoke enhancements in time for the first rehearsal.

Comprovisação nº 5

2017, ESML (Portugal). As a result of a longer period of rehearsals and ongoing development, stability issues were properly identified and corrected.

New features include quantisation / standard rhythmic notation and further enhancement of algorithms including cross-algorithm tempo synchronisation. Algorithmically generated lyrics text was also implemented and used, granting all singers with the same phonemes for simultaneous notes. This has proven to be an interesting aesthetic feature as was assessed during rehearsals and performance. Multiple soloists and a non-standard spacial setting (musicians played in two stairways, over 50 meters apart from each other) were two key features of this performance.

An especially important aspect consisted in the development of 1) an improved preset mechanism, as well as 2) a Miraweb [Ben16] enabled messenger module. The former allowed musical form to be planed ahead which in turn enabled a greater level of readiness for interaction, from the mediator's standpoint. The latter made it possible to quickly choose a group of recipients (according to instrument family, for example, or to group

assignment) and send them a message (usually a technique term or instruction).

Yet one problem remained to be solved: synchronisation (which is never perfect in a real-time composition network environment based on sight-reading) was sometimes undermined by poor graphics processing in some of the available computers. This caused the bouncing ball to be rendered at a low frame rate which had a noticeable impact on the timing accuracy of players. As of Comprovisação nº 4, the `lcd` approach had been replaced with a `jsui` object using `MGraphics` system [Gro11], but was found not to have made a significant difference and, in some cases, performance seemed even worse.

Comprovisação nº 6

2017, II European Saxophone Congress (EurSax'17), Casa da Música, Porto (Portugal). To improve synchronisation, a complete reconstruction of the notation interface using OpenGL hardware acceleration was undertaken. It was home tested by a number of musicians who agreed to continue collaborating in the project after Comprovisação nº 5 and used in performance since Comprovisação nº 6. Reconstruction had to account for the fact that layering of different graphical `Max` objects with transparent background would no longer be possible. Thus, all graphical features (with exception of the notation objects) had to be integrated in OpenGL programming, which turned out to be beneficial towards all aspects of the interface (not just synchronisation), according to musicians' reports.

The preset module was further improved, alongside the messenger module. Since Comprovisação nº 6, it is possible to automatically send a set of individualized messages whenever a new preset is called. This is an effective tool for managing musical form, where instructions can be simultaneously sent to all performers (including the soloist) concerning their specific role in a given musical context.

Comprovisação nº 7

2017, IV Peças Frescas Açores Festival, Colégio Church, Ponta Delgada (Portugal). This performance for nine singers, five instruments and solo saxophone used 14 client computers in a successful way. Singers sang from the choir loft and instrumentalists were positioned in different points of the altar and lateral naves.

Microtonal notation was used for singers, with the aid of earphones to ensure good intonation, enabling the use of spectral harmony (refer to Video Example 5.6). Also, the new centrifugue feature was introduced.

Comprovisação nº 8

2017, Symposium on Computer Music Multidisciplinary Research (CMMR), Real Vinícola, Matosinhos. In this performance, seven musicians (four saxophones, piano, electric bass and steel drums) were positioned around the audience. All seven were sight-readers but four of them were also improvisers, alternating between the roles of soloist and regular ensemble member. Microtonality was also used with the saxophones.

Comprovisação nº 9

2018, Sound and Music Computing Conference, Rialto Theatre, Limassol (Cyprus). This performance featured HASGS – an augmented instrument played by the soloist – to explore the possible synergies between a real-time composition and notation system (`Comprovisador`) and a hybrid acoustic-control augmented instrument to enhance the level of interactivity (see Section 5.1.11).

After the OpenGL implementation, it became more noticeable that network jitter was also negatively affecting synchronisation. To overcome this issue, a time-stamp based system was implemented before "Comprovisação nº 9", in accordance with Dannenberg's concept of "time-flow" [Dan17] (see Section D.3.1).

Another upgrade was the quantiser enhancement mentioned in Section 3.4.4. Instead of writing two $\frac{4}{4}$ bars, the algorithm now writes eight $\frac{1}{4}$ bars.

New algorithmic possibilities were introduced, namely *acciaccature*, tuplets and a random ornament selector.

Comprovisação nº 10

2018, Centro Cultural de Lagos – pátio, Lagos (Portugal). Precomposed passages were deployed, resembling open-form aesthetics. The passages were restricted to loops of up to eight beats.

Regarding the use of space, musicians from the ensemble played in the galleries on the 2nd floor, spread over a length of 40 metres. Furthermore, the soloist was playing a MIDI instrument capable of sending three note streams simultaneously (three MIDI channels: right hand, chords and bass).

Comprovisação nº 11

2022, Escola Secundária de Loulé – átrio do bloco E, Loulé (Portugal). The adaptive metronome made its debut, used in concurrence with the bouncing ball for improved synchronisation between musicians.

This performance took place inside a spacious hall (600m^3). Musicians were spread over three floors and staircases, surrounding the hall. The performance featured interaction with a group of art students, as they live-painted a mural.

Another debut was a System Launcher application with automated checklist – implemented to avoid mishaps potentially caused by a moment of inattention.

Before the concert, testing of the time-flow approach was conducted with sixteen laptops playing a synchronised metronome through their loudspeakers during ten minutes, with seventeen musicians carefully listening. No jitter or flam effect was perceived which safely indicates sub-twenty millisecond precision. These results are deemed successful.

5

Musical Performance Practice

In this chapter, the analysis will focus on various musical examples extracted from live performances featuring `Comprovisador`. This approach aims to provide an understanding of the **modes of operation** used and their relevant aesthetic results. These modes of operation were derived and examined as an implementation of the principles of practice-based research. That is, they were created and studied in authentic, real-world performance environments. To fully comprehend the various modes of operation, it is necessary to be acquainted with the effect of each parameter that is accessible from the control interface, in addition to relevant terminology outlined in Section 3.4.

5.1 Modes of Operation

5.1.1 Preset Editing (Composing)

As stated in Chapter 3, the curation of parameter presets is a critical part of composing and ultimately delivering a successful comprovisation performance. The significance of instrumentation has also been highlighted, particularly the ability to define instrument clusters within those presets. Using the `nodes` interface to perform definitions graphically, adjusting one node at a time with a mouse, can be a tedious task that often results in unpolished outcomes. The need for an editing mode to automate certain aspects of this process was therefore identified.

Video Example 5.1 showcases the Node Cluster Edit Mode of `Comprovisador`, which accelerates the node clustering process through parameter adjustment. Switching to this mode will alter the role of all knobs and faders on the control surface[1], rendering them appropriate for editing operations.

The **nodes_center** parameter facilitates the rotation of nodes or node clusters along a concentric trajectory. Parameter **num_cluster** allows nodes to be clustered by number. A cluster can be made up of consecutive or interlaced nodes, and this is determined by two different regions of the **num_cluster** knob. The **Δphase_node** parameter modifies the clustering phase. For instance, it can alter the sequence from [1,2,3] [4,5,6] to [2,3,4] [5,6,1]. In Node Cluster Edit Mode, there is a dedicated **node_size** knob for each of the four slots.

The creation and adjustment of node clusters across multiple slots can be easily achieved in this mode. This is a helpful tool for designing presets to be used in performances. Notwithstanding, it is also possible to manually place nodes in a non-concentric arrangement, which may at times be more advantageous depending on the desired outcome.

Video Example 5.1

Comprovisador - Node Cluster Edit Mode.

After allocating the instruments into the appropriate algorithmic slots and control groups, and arranging the nodes as intended, the following step towards preset creation is defining the initial parameter states for each control group. The reference score can provide an insight into the kind of algorithmic response to be expected.

[1] Fourteen additional templates are available on the control surface mirror. These can serve as future operating modes should new algorithms necessitate a fresh parameter configuration. Parameter names can easily be entered into the corresponding controller labels.

In order to communicate the intended musical attitude to the players, it is useful to create targeted messages to be dispatched upon preset triggering. When possible, it is advisable to experiment in a rehearsal setting and make necessary refinements to the initial parameter states while exploring various possibilities of manipulation. Taking notes on the **musical form script** is beneficial at this stage. Musicians will also value the rehearsal experience as it allows them to anticipate the algorithmic behaviour, even though they cannot study notes and passages as they typically would during a traditional rehearsal due to the notation being generated in real time.

5.1.2 Harmony

Algorithm Harmony was used for the first time in "Comprovisação nº 1". At that time, there was only one transposition mode available, based on octave equivalence. On the other hand, the **positiveHarm**/**negativeHarm** approach had already been implemented.

In the initial segment of Video Example 5.2, extracted from "Comprovisação nº 1", chords are formed using notes that the soloist recently played (**positiveHarm**). In the last third of the video, negativeHarm is activated shortly before the *frullato* effect can be heard. In the example, it is audible that **positiveHarm** promotes stability, whereas **negativeHarm** elicits harmonic movement. In the former case, since only the soloist can decide whether to stay in the same harmony or move away from it, it is the soloist who usually initiates the changes in the harmonic field. Nevertheless, there is a possibility of error arising from either the machine listening algorithm or the sight-reading process of the musicians, which can also lead to a change in the harmonic field. In the case of **negativeHarm**, all ensemble responses are intentionally altering the harmony. As the soloist responds to this alteration by transitioning towards the new harmony, the ensuing ensemble response will once again modify the harmony – thus forming a spiralling cycle. In addition, there has been a reduction in the phrase ending threshold (parameter **agógica**), which was performed by the mediator, resulting in an increase in rhythmic density and a faster rate of harmonic change.

The interaction feedback loop demonstrated here deserves attention. The soloist provides input for the system through his improvisation; the system's algorithms use this input along with the mediator's parametric input to generate a score; this score is sight-read by the ensemble musicians, enabling them to deliver a coordinated response to the improvisation. Subsequently, the soloist's performance is influenced by this response, while the mediator is influenced by all. As discussed in Chapter 2, this constitutes a dialectical relationship in which the decision-making of the different actors is affected by the actions of others in an interactive manner.

Video Example 5.2

Comprovisação nº 1 - algorithm Harmony.

Video Example 5.3 provides a clear demonstration of the infinite threshold setting (**agógica** parameter at its maximum), enabling prolonged chords. The triggering of new chords is initiated by the mediator. In the given example, it is noticeable that the harmony progresses more when the soloist performs chromatic or "outside" passages (jazz nomenclature). Additionally, there are numerous leaps in the individual voices, which is not a disadvantage in this quasi-orchestral setting, but could be problematic in a context where the texture is reduced. It should be noted that the algorithm's voice leading feature had not yet been implemented.

Throughout the performance of "Comprovisação nº 2", the system experienced intermittent instability owing to undetected software glitches and subpar hardware, specifically an outdated wireless router. These issues have since been resolved. Near the end of the video example, a brief episode of instability resulted in the violins getting trapped in repeating a pattern after the rest of the ensemble had stopped playing. To an unassuming listener, it could have appeared to be an intentional musical gesture.

Video Example 5.3

Comprovisação nº 2 - final chords.

Let us now examine two video examples from "Comprovisação nº 5", the first performance to include **space as a compositional element**. Video Example 5.4 corresponds to the beginning of section 1 of the performance. Here, the instruments to the left of the solo saxofonist (group 1) played long ***mp*** notes while the instruments to the right (group 2) played short ***ff*** notes. Furthermore, both groups struck the notes simultaneously, creating the impression that the first group was providing a gentle sustain to the sharp and rapidly decaying attack of the second group (to use synthesiser nomenclature). This tells us that two control groups were being used, each with a unique instance of algorithm Harmony and a specific set of instruments allocated to it. Several parameters were preset to different values in order to to obtain the contrasting outputs, except for **agógica**, which needed to be identical in both in order to achieve simultaneous strikes. It should be noted that while the instruments to the right of the soloist (group 2) consistently played in *tutti*, the others always played in pairs: flute and alto saxophone, two singers or violin and oboe. Also, while the order of the pairs was randomly chosen, the pairs were fixed (preset) – this is a demonstration of the automated switching of timbral sets mencioned in Chapter 3. The operation of this feature, named **autoClick**, is explained below. Furthermore, it is worth noting that the default modulus-based transposition mode was in use.

Regarding Video Example 5.5, at the start of section 2 where a pianist replaces the previous soloist, the type of harmonic response changed significantly from the previous example. Here, the **registoFixo** transposition mode was activated. The resulting harmony aligns more closely with the soloist's performance, as the notes remain in their original register and are not transposed. In addition, the texture is less dense, as any notes outside of the range of their assigned instrument are filtered out instead of being transposed. To contrast with the previous section, the soloist received encouragement to maintain harmonies for longer durations, and the **registoFixo** strategy certainly facilitated this. The fact that his playing style (like that of many pianists and other players of harmonic instruments) relied considerably on left-hand rhythmic filling was also helpful in achieving the goal of prolonged, almost static harmonies. This is due to the fact that continuous filling prevents the crossing of the phrase-ending threshold, ultimately preventing the triggering of any chords. The statism and clarity of texture achieved allowed the effective use of subtle timbral nuances, such as vowel modulation by singers, flute trills and *frullato*, as well as melodic lines triggered by the algorithm Contour, which initiate near the end of the excerpt.

Returning to the previous example, numerous timing discrepancies are evident in the synchronised attacks. These were due to the reasons discussed in Section 4.2, namely the use of the `MGraphics` system, which was unable to render the bouncing ball at a satisfactory frame rate on some of the available machines, and the lack of a network clock implementation (see Time-flow concepts in Section D.3.1).

Video Example 5.4

Comprovisação nº 5 - default transposition mode; two groups, sharp attack and sustain; autoClick.

Video Example 5.5

Comprovisação nº 5 - registoFixo transposition mode; statism.

Having covered the three transposition modes and the **positiveHarm**/**negativeHarm** approach, let us now focus on an example involving the **spectralize** function. As outlined in Section 3.4.3, this functionality was originally implemented in "Comprovisação nº 7", featuring a choir composed of nine singers and five instruments. It was later tested in "Comprovisação nº 8" with saxophones, but the resulting sound could not be classified as spectral due to the musicians relying on their intuition to play the microtonal notation. The outcome was interesting from a microtonal perspective, but it fell short of the proper just intonation achieved by the choir, thanks to the use of cue notes via earphones. It would be worthwhile to investigate the impact of this cuing device on instrumentalists.

In Video Example 5.6, the choir performs spectral chords using phonemes that have a high likelihood of the vowel sound 'ʊ'[2], a low likelihood of nasal consonants such as 'm' or 'n', and hardly any fricative or plosive sounds. This probability weighting is achieved through the appropriate interface object (refer to Figure 3.15) and then stored in the relevant preset.

At a certain point in the example, the instrumentalists are prompted to improvise whenever a note appears in their score rather than playing the designated note. This way of controlling the texture of an improvisation will be further discussed below, in Section 5.1.14.

Video Example 5.6

Comprovisação nº 7 - just intonation chords (spectralize function).

5.1.3 Contour

The basic concept of the Contour algorithm is depicted in Video Example 5.7, which shows the beginning of "Comprovisação nº 1". Short melodic fragments are captured from the soloist's performance and assigned to instruments of the ensemble when the mediator presses the **newContour** button or increases the density parameter. The latter results in more instruments being added to the musical texture with updated contours, enabling a few of these to be stacked for a polyphonic effect. There is a point in the video where the texture is reduced, and only a pair of clarinets can be heard playing. This moment, to me, is one of beauty that came about purely by chance, as the algorithm did not allow for the selection of the precise instrumentation at this stage, as detailed in Section 3.2.6. Parameters **dinâmica** and **agógica** were also manipulated to good effect. The expressive manipulation of texture, dynamics and agogics, while simplistic, is perceptible and capable of fostering soloist-ensemble interaction.

In regard to rhythm, notes were uniformly positioned across the musical staff in proportional notation, therefore conveying no indication of rhythmic value. Moreover, the side-by-side notation viewers (refer to Figure 4.3) could not accommodate extended musical phrases or manipulation of loop regions. Musicians were requested to improvise the rhythm as a method of compensation, yet the level of creativity applied to the task appeared to be insufficient.

[2] – or 'oo', as in 'book'.

Video Example 5.7

Comprovisação nº 1 - algorithm Contour.

The following example (Video Example 5.8) is from a rehearsal that occurred before "Comprovisação nº 2" and demonstrates that although there were no updates made to the representation of rhythmic values, other aspects were enhanced. Specifically, wide, stacked notation viewers were implemented, enabling the adjustment of note duration lines in response to **articulação** parameter manipulation, as well as the stretching/compressing of note spacing along the x-axis (i.e. temporally) in response to **agógica**. Additionally, a `nodes` object offered partial control over instrumentation, albeit without provision for group allocation. The video suffers from poor image quality, but it effectively demonstrates the scores in action, as well as the musical response of individual musicians. Furthermore, the preset manager had not been implemented yet, resulting in all mediator actions being improvised.

Video Example 5.8

First rehearsal with João Moreira - prior to Comprovisação nº 2.

Video Example 5.9 is the continuation of Video Example 5.5, from "Comprovisação nº 5" and demonstrates the impact of pitch-wise contraction and expansion transformations on the initial contour in response to the soloist's input. These transformations are grounded on the modulus computation of the preceding played phrase. As a result, the output will contract or expand accordingly, while also being transposed to the boundaries of the same modulus.

In the given example, the piano in the background plays the same melodies as the oboe and violin but intentionally not in sync, as prescribed by the non-sync directive. It was not intended for the oboe and violin to synchronise their playing, but it may have been challenging to avoid synchronisation due to their close proximity. All things considered, the outcome is aesthetically pleasing.

Video Example 5.9

Comprovisação nº 5 - transformations to the contour in response to soloist input.

Video Example 5.10 provides a compelling illustration of how mediation can impact the soloist's performance, promoting musical interaction. Here, the soloist uses a three-note motif resulting from the mediator's manipulation of the loop size, as well as prior algorithmic transformations. It's essential to note that the soloist was the

original creator of the melodic contour, which led to the development of the three-note motif. It is clear from these examples that employing two control groups to generate two complementary musical functions enhances the musical texture.

Video Example 5.10

Comprovisação nº 5 - manipulation of the contour and motif appropriation by the soloist.

In the following example from "Comprovisação nº 7" (Video Example 5.11), algorithm Contour applies a different type of transformation. Instead of the transformation occurring in reaction to the soloist input, it is determined by the mediator's interaction with the up/down buttons on the control surface. Upon each press of such buttons, the contour transposes upwards or downwards to a random interval up to 3 semitones. As it is the case with soloist-induced alterations, register re-transposition may take place to provide variety. It is crucial to distinguish between the two types of transposition: the transformation that shifts the original contour slightly for musical discourse progression and the register transposition that adjusts the contour to the range of the assigned instrument.

An interesting aspect of this example is the harmonic symmetry that arises from the modulus of a fifth and the audible superposition of this interval, further enhanced by the textural augmentation that emerges from the transformation.

Video Example 5.11

Comprovisação nº 7 - harmonic symmetry within algorithm Contour; transformation by transposition.

The performance of "Comprovisação nº 10" was characterised by numerous interactions supported by the Contour algorithm. The example below (Video Example 5.12) shows some of these interactions. The soloist clearly allows himself to be influenced by the outcome, creating space to hear (and let the audience hear) the response to his input. Due to the sometimes virtuosic nature of the input, the ensemble members appear to struggle at times[3], thus generating tension for the audience. However, this is not necessarily undesirable as long as there are sufficient moments of release.

The captured contour initially heard in the given example is appropriate for showcasing a distinct outcome (in contrast to the earlier example) that can be achieved through the default register transposition mode. The

[3]The flautist and pianist actually perform rather well, but unfortunately the sound recording does them no favours.

original melody, in D Major, spans over a minor tenth in the middle to low register. It commences on the low dominant melodic degree, rises to the high dominant, and goes through the lower mediant. To match the range of the violin, the response had to be transposed to a higher register. As a result, the melody is played a minor tenth higher in F Major, leading to a bitonal moment without conflicting registers.

Video Example 5.12

Comprovisação nº 10 - various contour loops.

5.1.4 AutoClick

Video Example 5.13 below demonstrates the technical side of the **autoClick** function. The mouse remains stationary throughout while incoming notes reposition the crosshairs within the **`nodes`** object, creating the illusion of clicking. Parameter **agógica** has been set to the minimum threshold, leading to high responsiveness. The instrument nodes have been arranged in four clusters of four, equidistant from the object's centre. When the **autoClick** function is enabled, a specific cluster is automatically selected by the crosshairs just before the algorithm produces a chord, which occurs every time a phrase ending is identified. This process results in automated switching between timbral sets. The crosshairs move in polar coordinates, changing azimuth and maintaining the distance from the centre. Parameter **nodesize** can be adjusted to allow clusters to intersect or have gaps between them. A rest is produced when the crosshairs hit a blank space.

Video Example 5.13

Comprovisador.host – demonstration of the autoClick function.

Since its inception in "Comprovisação nº 5", this approach has been used extensively. In "Comprovisação nº 7", various examples of its application successfully produced pleasing musical results, ranging from smooth timbral transitions to striking density contrasts. An example featuring smooth timbral transitions is presented in Video Example 5.14, wherein the nodes are clustered according to the gender of the singers. The timbral aspect is further enhanced by the different phonemes applied to each sung chord.

Video Example 5.14

Comprovisação nº 7 – clustering (male versus female).

The following example (Video Example 5.15) is a revisit of the sharp attack / gentle sustain gesture first explored in "Comprovisação nº 5" (refer to Video Example 5.4). In the first part of the example, the woodwinds and piano perform brief emphfortissimo notes; the choir sings prolonged emphpianissimo notes, taking turns between male and female voices and moments of silence (when the crosshairs fall into the void of the `nodes` object). The roles are then reversed in terms of dynamics and duration (articulation parameter), but the choir still employs the same type of **autoClick** strategy (men, women or silence, but perhaps with a higher probability of silence). Moreover, this section exhibits a lower reactivity level than the first section, likely due to a higher threshold or to the soloist playing less actively.

It is worthy of note that the level of synchronisation in "Comprovisação nº 7" surpasses that in "Comprovisação nº 5". The usage of openGL for GPU rendering of the bouncing ball was the main contributing factor to the positive outcome, although it should be noted that the quality of the available laptops was generally better. Furthermore, a slight difference in cohesion was noticed between singers and instrumentalists – the former used earphones to receive cue notes[4], which had a beneficial effect.

Video Example 5.15

Comprovisação nº 7 – two groups (attack/sustain) and clustering.

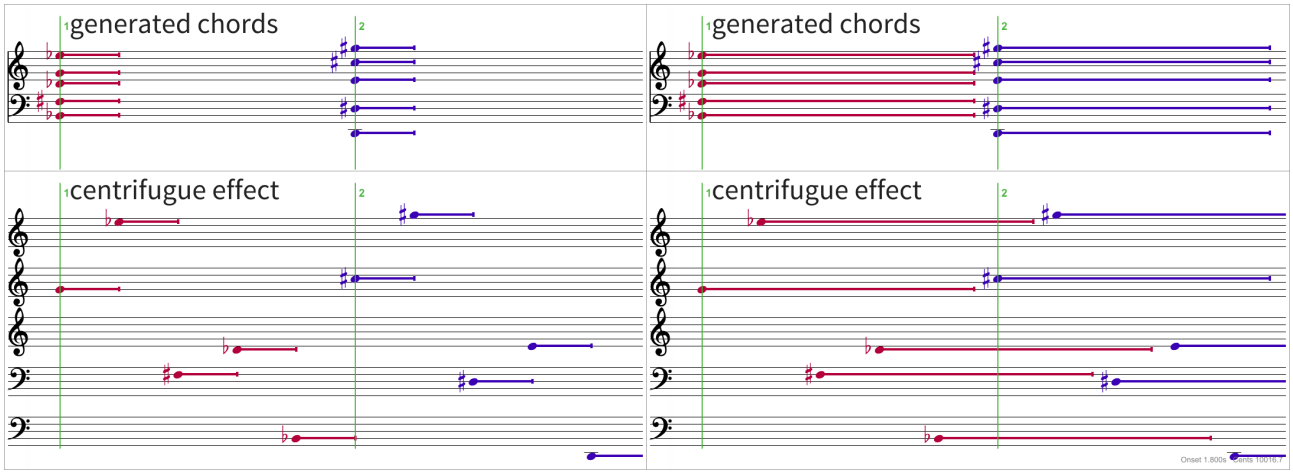
5.1.5 Centrifugue

As previously mentioned in Chapter 3, **centrifugue** is a play on the words centrifuge and fugue. This effect is illustrated in Figure 5.1. When the **centrifugue** parameter is set to its maximum value, individual notes of each chord are uniformly distributed over the selected rhythmic duration[5] for that chord. If there are no changes to its value or to the instruments assigned to play, the order of the attacks is retained for the next chord, resulting in a series of chords being played as a cycle of individual attacks. If the value is between the minimum and the maximum[6], for example, three out of five, the software will try to evenly distribute all available instruments into three lots. This results in three evenly spaced attacks (subchords) for each generated chord, with two lots of two instruments and one lot of one (see Figure 5.2). The order of the lots is maintained for subsequent chords until the value of the **centrifugue** parameter is changed or the list of instruments assigned to play is altered.

[4] The adaptable metronome had not yet been implemented at that time.

[5] Rhythmic duration refers to the allotted multiple of the metronomic period rather than the chord's actual duration, which is affected by the **articulação** parameter. This parameter sets the actual duration to some proportion of the rhythmic duration.

[6] The minimum value of the **centrifugue** parameter is always one (effect off), while its maximum value is always equal to the number of instruments assigned to play.



(a) with **articulação** set to 20% – no overlapping occurs; (b) with **articulação** set to 92% – overlapping occurs;

Figure 5.1: **centrifugue** effect (parameter value set to 100%): individual notes of each chord are uniformly distributed throughout the selected rhythmic duration.

In Video Example 5.16 below, the **centrifugue** effect is being employed create a texture based on the concept of pitch overlay. At the beginning of the excerpt, the **%densidade** parameter is set to a low value, which means that only a few instruments or voices are assigned to play or sing each time a new chord is triggered. It also means that the instrumentation will inevitably change with each new chord, as the algorithm assigns a subset of the available instruments to play. Long durations are employed while **centrifugue** is at its maximum. This produces a very spacious texture with single-note attacks, overlapping of note durations (refer to Figure 5.1b), and evolving instrumentation. The overlap is such that it is impossible to tell when one harmony begins and another ends – they slowly blend into one another. Nor is it possible to predict a sequence of timbres, although the low-density texture allows interesting timbral qualities to emerge.

Video Example 5.16

Comprovisação nº 7: centrifugue enabling pitch overlay.

Video Example 5.17 below demonstrates the **centrifugue** effect used in conjunction with other parameters to create a density *crescendo*. The example begins exploring the same concept as the previous one, but evolves differently. Gradually, the number of assigned instruments is increased by manipulating the **%densidade** knob. To maintain a similar or slightly increased rate of attacks, it is necessary to reduce the **centrifugue** value. This, however, results in subchords of increased note density (refer to Figure 5.2). It is possible to hear some cohesive attacks of this kind, midway through the example. The crescendo is carried on with an increase in the **centrifugue** value and a decrease in the rhythmic durations[7], for maximum rate of events. At this stage, all sixteen instruments

[7] **Duração** is the parameter that enables control over the metronomic period, whereas the multiples are controlled via the

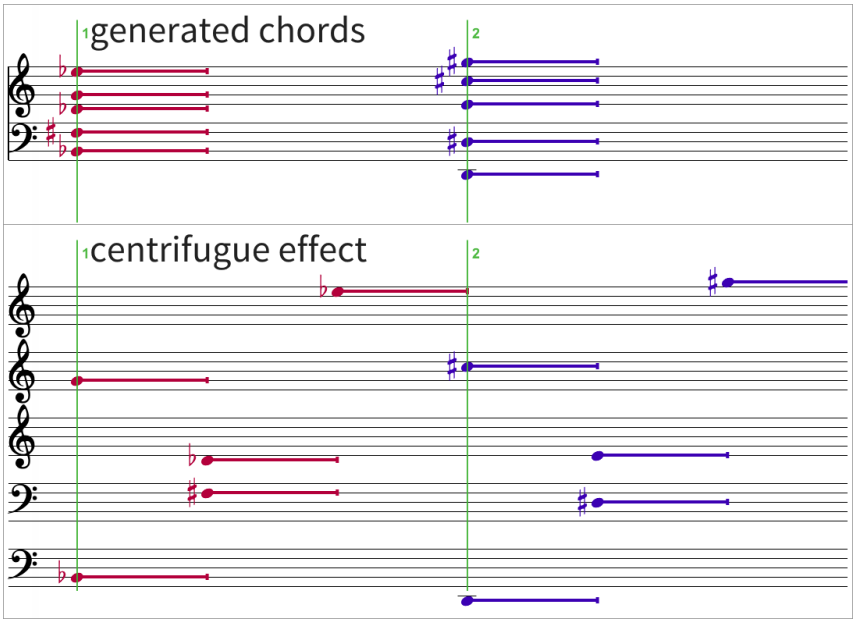


Figure 5.2: **centrifugue** parameter value set to 3 out of 5: individual notes of each chord coalesce to form three balanced subchords that are evenly spread across the selected rhythmic duration (with **articulação** set to 33%).

are assigned to play. From that point onwards, the climax is reached by increasing not only **dinâmica** but **articulação** as well, resulting in a dense texture of overlapping sounds. A new resource is then used for contrast: unison, short, ***ffff*** attacks (explained below).

Video Example 5.17

Comprovisação nº 11: centrifugue evolving into a density crescendo.

In other performances with **`Comprovisador`**, the **centrifugue** effect is used in different manners. Before addressing those examples, it is important to mention Thompson's Soundpainting [Tho06] and the work of Olivier Benoît [Ben99] – both were an inspiration in the development of this resource. *Pointillism* is a prevalent strategy in Soundpainting, but perhaps more influential was Benoît's use of a gesture that he referred to as sweeping (in french, *balayer*). This gesture consisted in moving his arm rapidly in a semi-circle, always pointing towards the ensemble; musicians had to react quickly and play a short sound whenever his arm "swept" by them.

The first iteration of the **centrifugue** was in fact an attempt to recreate the *balayer* effect. It was based on the graphical object **`nodes`** with its crosshairs rotating automatically. As the crosshairs passed on top of the instrument selection circles, a note would be triggered for the relevant instrument (see interface demonstration

`multislider` interface.

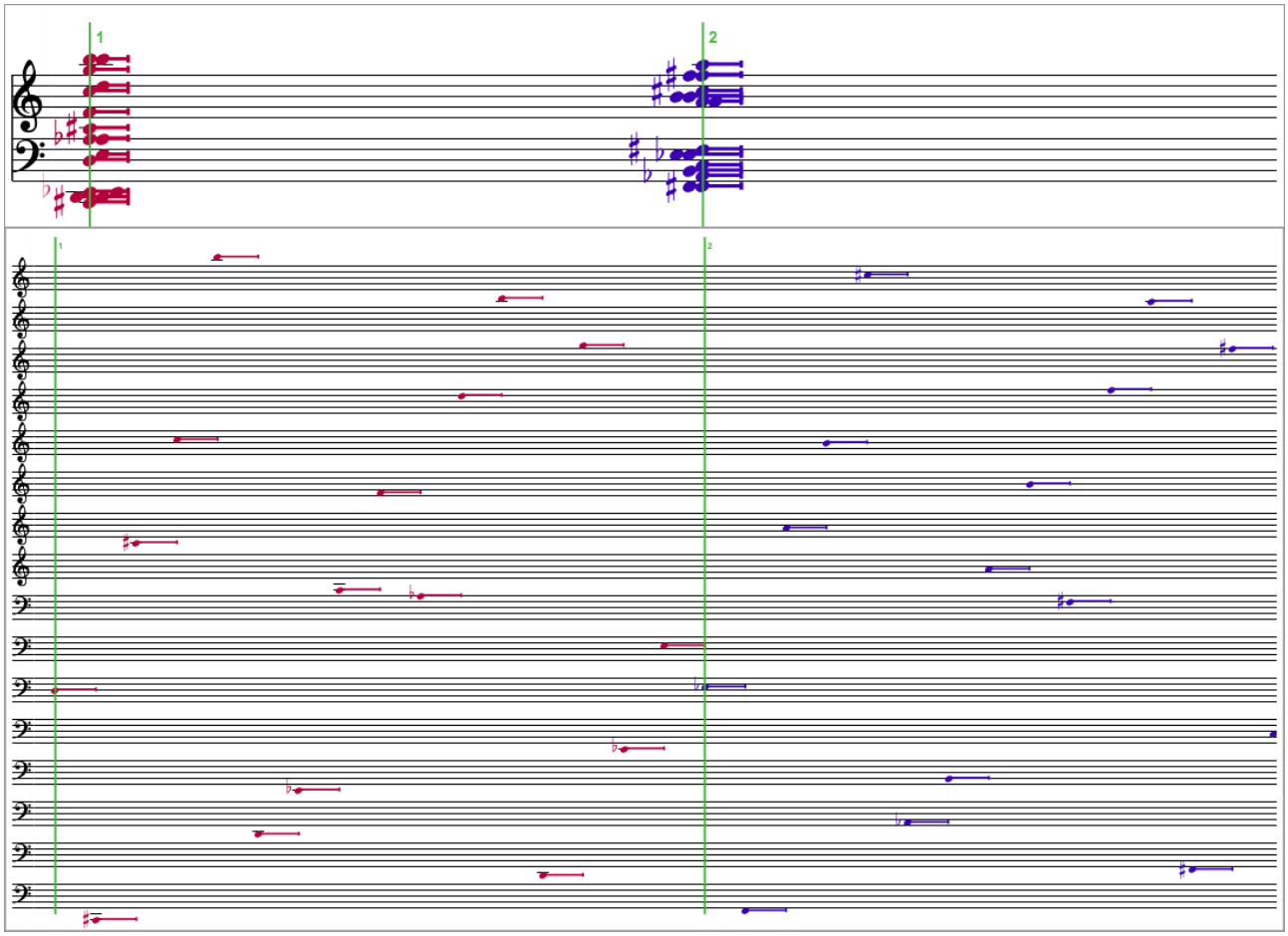


Figure 5.3: A *pointillistic* effect achieved with **centrifugue** at its maximum and sixteen instruments assigned to play.

in Video Example 5.18). This approach was later abandoned due to its adverse effect on CPU performance, as it depended on graphics performance to generate symbolic output, which was not computationally efficient. The effect can be heard in Video Example 5.19 extracted from "Comprovisação nº 2".

Video Example 5.18

Comprovisação nº 2: centrifugue prototype - autorotate interface demonstration.

Video Example 5.19

Comprovisação nº 2: centrifugue prototype - sweeping effect.

In Video Example 5.20, sourced from "Comprovisação nº 10", the **centrifugue** effect is used with a slow setting. The audience perceives the cycle of seven timbres and corresponding sound locations. To provide diversity to

the texture, ornaments are used. The messenger module includes a random function that automatically selects from various ornament types (refer to Figure 3.5).

Video Example 5.20

Comprovisação nº 10 - slow centrifugue with ornaments.

5.1.6 Tuplets

Section 3.2.3 analysed the potential to divide a note occupying one pulse into equal subdivisions. This approach allows for the creation of polyrhythmic effects by overlaying different subdivisions of the same pulse using two groups. Video Example 5.21 below, from "Comprovisação nº 10", has been selected to illustrate a number of issues.

1. Following a pre-composed section in E Dorian, which is audible at the start of the video, the guitar allocated to algorithm Harmony - variation 2 - plays "quavers" at a consistent tempo. This is due to the **divisão** parameter being set to 2. The guitar incorporates E Dorian notes in response to the recent improvisation on the E Dorian pattern by the soloist, indicating that perhaps the **equiv8a** transposition mode was in use. However, when the violin enters, there is a harsh dissonance which may indicate a shift to the default transposition mode which dampens the mood. This is one example of a limitation that cannot be controlled when drawing up a list of presets based on an idea of a future improvisation, the direction of which it is impossible to foretell.

2. When musicians switch between tuplets, the transition can be inaccurate. This limitation has since been overcome by implementing the new adaptable metronome (cf. Video Example 5.22)

3. A conflict may arise if the soloist plays rhythmically while the ensemble tries to follow the machine's tempo by relying only on visual cues, rather than auditory ones. The situation is exacerbated when musicians are physically separated from each other, as was the case[8]. The conflict may result in a deviation of the soloist's tempo from that of the system. It can be concluded that a metronome would be advantageous in aiding ensemble members to remain in sync with the system, being more effective than visual cues. This would then allow it to be easier for the soloist individually to compensate for any drifting that may

[8]The group of seven musicians was dispersed throughout a gallery that spanned over 40 metres. Sound propagating over such distances introduces a level of latency that significantly hampers rhythmic interplay.

occur between his tempo and the system's, as his occasional drifting would not affect other musicians' perception of the system's tempo. Still, in certain environments, this issue may not be as straightforward as setting up a metronome, as discussed below in Section 5.1.9.

Video Example 5.21

Comprovisação nº 10 - tuplets.

Moving on to "Comprovisação nº 11", Video Example 5.22 showcases the use of the tuplet-based polyrhythm approach supported by the adaptable metronome to assist in complex modulations between tuplet ratios, resulting in high accuracy. The metronome anticipates the appropriate subdivisions to be played, thereby positively influencing performance.

Compared to the previous example, this example demonstrates a significant increase in accuracy. It is worth noting that the musicians in this example are in eleventh and twelfth grade, while the musicians in the previous example were professionals. Despite this, the tuplet transitions are executed more proficiently in this case. This is evidence of the effectiveness of the adaptable metronome.

It should be pointed out that there are two groups, allowing two simultaneous tuplets (polyrhythms), and that the **autoClick** function is active, allowing automatic switching between timbral sets.

Video Example 5.22

Comprovisação nº 11 - tuplets with adaptable metronome (enhanced precision).

5.1.7 Acciaccature

The two following examples from "Comprovisação nº 9" are intended to demonstrate the *acciaccatura* effect obtained with the **shortMotiv** parameter in the **threshRhythm** variation (refer to Figure 3.17). Video Example 5.23 is suggestive of imitative counterpoint. The melodic interplay is aesthetically appealing. There is also some ornament exploration with good results. In Video Example 5.24, the *acciaccatura* effect results in more intricate outcomes, which can be attributed to the soloist's employment of a less conventional playing technique.

Video Example 5.23

Comprovisação nº 9 - acciaccature and ornaments.

Video Example 5.24

Comprovisação nº 9 - acciaccature with more complex input.

In “Comprovisação nº 10”, there is a noteworthy example of *acciaccature*, with ample space to appreciate the distinct entrances and the soloist’s interaction (see Video Example 5.25). A notable moment occurs when the flute imitates the soloist, who responds with a modified reimitation of the theme.

Video Example 5.25

Comprovisação nº 10 - acciaccature.

A crucial example on this topic can be found in “Comprovisação nº 11” (see Video Example 5.26). Again, the perceived “togetherness” (referencing Lehmann and Kopiez [LK10]) is remarkable, characterised in this example by *acciaccature* played in pairs of similar instruments – a fact which denotes the use of the **autoClick** function. Given that there is no implied pulse, since it is based on the **threshRhythm** variation, nor is the duration of the short notes quantifiable, it is remarkable that the clarinets can produce such closely matched phrases – while sight-reading. It must be attributed to the consistency of the preparatory beat provided by the metronome.

Video Example 5.26

Comprovisação nº 11 - acciaccature in pairs.

5.1.8 Advanced Slot Usage / Part Unifier

The system enables the operation of four algorithmic slots simultaneously, although it is uncommon to use more than two at once. This is because there are no autonomous controls for each individual slot, but two control groups that each oversee two slots. Nonetheless, with some ingenuity, it is feasible to fully utilise the slots to produce effective results in terms of musical texture.

A potential scenario is a chorus-like texture, where each slot represents a choral part. Use of the Part Unifier is required to facilitate this, allowing for a selection of instruments to play the same part (i.e. doubling). This device permits up to four distinct parts to be specified, making it simple to designate a single united part for every algorithmic slot. Although the Part Unifier is capable of operating with any mode of register transposition, the most conspicuous one in this instance would be **equiv8a**.

Before proceeding with the idea of employing all four slots, let us examine using solely one slot with algorithm Harmony and allocating all four unified parts to it. The algorithm would generate four notes for each chord and distribute one note to each unified part, resulting in a four-part homophonic texture. The Part Unifier would accomplish the doubling of parts.

By allocating each unified part to a particular slot, each with an instantiation of algorithm Harmony - variation 2, we attain the benefit of creating independent rhythms for each part. In addition, ensuring parameter **agógica** is identical across control groups guarantees that the tempo will be shared across all slots/parts. As an example, the first part may include sopranos and a flute, while the fourth part may comprise basses, bassoon, and piano. This is the case in Video Example 5.27, extracted from "Comprovisação nº 7". The example concludes with all instruments allocated to the Part Unifier's first column, resulting in a *tutti* unison.

Video Example 5.27

Comprovisação nº 7 - chorus.

Video Example 5.28, taken from "Comprovisação nº 11", illustrates an alternative application of the *tutti* unison technique: sharp attacks. This segment of the performance contrasts with the dense *pointillistic crescendo* of the previous segment (refer to Video Example 5.17).

Video Example 5.28

Comprovisação nº 11 - tutti unisons, sharp attacks.

The *finale* of "Comprovisação nº 11" revisits that of "Comprovisação nº 7", with the exception that the soloist opted to perform solely in C Major (see Video Example 5.29 and 5.30).

Video Example 5.29

Comprovisação nº 11 - C Major (chorus).

Video Example 5.30

Comprovisação nº 11 - finale in C Major chord.

5.1.9 Quantum Loop

Since its inception, the Quantum Loop mode encountered many problems, comprising technical hurdles (some of which were due to bugs in **bach** that appear to have been resolved) and design issues with the notation interface. The concept operates by allowing loop transformations (alterations to the position of loop region boundaries) and/or melodic transformations (contraction, expansion or transposition) to occur while musicians are playing, causing the notes to change before their eyes. This inefficiency can lead to performer anxiety and disruptions in the musical discourse.

A number of developments have been implemented at the quantiser level, as discussed in Section 3.3.2 and used in the examples below. These have improved rhythmic readability, mitigating certain problems. At the time of writing, there is a solution being created to upgrade the standard rhythmic notation interface design. This solution is based on a model that has already demonstrated favourable results in another system, albeit with a fixed composition (refer to Appendix D). Here, musicians read a score in which measures are dynamically updated in a cycle to avoid disrupting the reading process. The measures that have been read are replaced at a distance of half a system (half a cycle). A working prototype of this upgrade, adapted for live generated scores, is presented in Section 7.1.1 of the final chapter. This novel solution will resolve the issues of discontinuity and the related performer anxiety, though it will result in a delay in the manipulation's outcome.

Video Example 5.31 shows the application of the Quantum Loop mode in "Comprovisação nº 11". The quantiser enhancements mentioned above have a beneficial impact on the musical outcome as they prevent unsuitable rhythmic complexities. The metronome is in use to enable synchronisation among ensemble members. However, there seems to be a discrepancy between brass and woodwind instruments that may be linked to the software, as it stays constant and repetitive like a canon a semiquaver apart – potentially due to instability within the module that measures network time.

The mediator attempted soloist-ensemble synchronisation using the tap-tempo function, based on the preceding pulse imposed by the soloist. However, substantial drifting occurred between the two parties due to the size of the space (which exceeded 600m^3 in volume) inducing unmanageable levels of latency and reverberation. Additionally, speech noise[9] from the audience may have had an adverse effect.

Video Example 5.31

Comprovisação nº 11 - quantum loop.

[9] This noise would increase further, as evidenced in the above examples (5.29 and 5.30). .

As referenced in Section 5.1.6, implementing a metronome in this scenario facilitates precise tempo communication for the ensemble members. However, it cannot ensure flawless synchronisation between the soloist and ensemble due to the soloist solely relying on the ensemble's performance to gauge the system's tempo. The issue lies in the latency produced by sound propagation across long distances. Is the use of a metronome for the soloist a potential solution to this conundrum? If so, how would it affect the improviser's spontaneity?

One factor that might have helped to minimise the drift problem in this situation is **Comprovisador**'s messaging capabilities. Sadly, this feature was not appropriately configured for the soloist in this performance. Possibly a suggestion like "play less and listen more" could have been beneficial.

Below is a sample of "Comprovisação nº 9" (Video Example 5.32) in which the Quantum Loop mode was employed with two layers (groups). This instance is deemed successful due to the cohesive rhythmic performance, even within the context of agogics manipulation[10]. Perhaps, this success is due to the musicians performing together on stage, particularly the woodwinds who shared the same music stand (i.e. the same laptop in dual-instrument layout). Performing in close proximity makes it easier to work together to compensate for drift when it occurs in relation to the visual cueing device (the bouncing ball). It ought to be highlighted that the adaptable metronome had not yet been implemented.

Video Example 5.32

Comprovisação nº 9 - quantum loop - two layers (groups) and agogics manipulation.

5.1.10 Precomposed Passages - Towards Open Form

Since "Comprovisação nº 10", precomposed passages can be used with some features and limitations[11]:

- it is based on the current Quantum Loop mode;
- each passage is limited to eight beats in duration;
- loops can be specified and manipulated within a passage;
- it is possible to regulate the instrumentation (by muting/unmuting) during the performance and to create pauses (mute all); this is achieved by using the **nodes** object.

[10] Incidentally, if the parameter **agógica** is lowered to zero in algorithm Contour, a fermata sign is displayed above the notes. This is how the mediator directed the ending of this performance.

[11] Those limitations will no longer be present when the upgraded standard rhythmic notation interface becomes operational.

There are four instances of precomposed passages in “Comprovisação nº 10”, with the initial one being the E Dorian passage mentioned earlier (refer to 5.21). All precomposed passages have been taken from “A Mãe que Chovia” [12] a children’s musical tale I wrote specifically for this ensemble. The passages are labelled “Night”, “Wind”, “Rain”, and “Parachute”. This was a practical reuse of previously rehearsed material.

Video Example 5.33 illustrates the “Wind” passage composed of contrasting/complementary melodic fragments. The example begins just before the precomposed part, with the violin melody (imitated by the soloist) marking the beginning of the segment. During the example, the mediator manipulates the instrumentation, thereby affecting the texture. The deliberate gaps created by the management of the instrumentation should be noted, as they generate a specific musical gesture - a feature of the Open Form aesthetics. Meanwhile, they also permit the soloist to interact. At a certain point, the various fragments amalgamate into a cohesive whole: the complementarity becomes apparent.

Video Example 5.33

Comprovisaçao nº 10 - precomposed passage: “Wind”.

Video Example 5.34 features the “Rain” passage, with rapid overlapping arpeggios. By manipulating the boundaries of the loop region, changes in harmonic field are effectively achieved alongside changes in harmonic rhythm.

Video Example 5.34

Comprovisaçao nº 10 - precomposed passage: “Rain”.

Finally, the “Parachute” passage that concludes the performance of “Comprovisação nº 10” is presented in Video Example 5.35. It features an appealing accompanied melody and undergoes live instrumentation tweaks. The soloist’s rhythmic exploration in this example is outstanding.

This example is also a confirmation of the fact that the bouncing ball on its own did not ensure a drift-free synchronisation when the musicians were at a distance from each other, as seen earlier in Section 5.1.6. The soloist appeared to seamlessly adjust to the ensemble’s average tempo, making it seem effortless. This may be due to his proximity to the contrabass, which played *marcato*, or to his extensive experience with improvisation. In contrast, there is a discernible rhythmic tension between the guitarist and the contrabassist, who were situated

[12] – “The Mother who Rained” (my translation). An original tale by José Luís Peixoto.

at the opposite ends of the ensemble, leading to noticeable drifting. The use of a metronome could have potentially solved this issue. This experience is one of the reasons why I decided to include a special metronome in **Comprovisador**.

Video Example 5.35

Comprovisaçao nº 10 - precomposed passage: “Parachute”.

5.1.11 Augmented Instruments as Controllers

“Comprovisação nº 9” was conceptualised with the aim of using an augmented instrument – specifically, the Hybrid Augmented Saxophone of Gestural Symbiosis (HASGS) developed by Henrique Portovedo – taking advantage of the possible synergies between a real-time composition and notation system (**Comprovisador**) and a hybrid acoustic-control augmented instrument to enhance the level of interactivity. Therefore, HASGS functioned as a musical interface with dual purpose. Firstly, it fed **Comprovisador**’s algorithms with improvised musical material performed on the acoustic side of the instrument. Secondly, it controlled several of its parameters by using embedded controllers and sensors. This effectively claimed some of the performance director’s mediation tasks for the benefit of interaction flow.

HASGS keypad allowed the soloist to navigate through **Comprovisador**’s presets according to the plan and subject to his momentary desire, while its other controllers (ribbon, trigger button, knobs, pressure and acceleration sensors) enabled him to control parameters such as dynamics, density, register and agogics, among others. A miraweb-enabled graphical feedback interface shown in Figure 5.4 was designed to enable the soloist to receive visual updates on the controls used on HASGS (a feature which was deemed valuable) and to receive messages from the control interface of **Comprovisador**. Another module called HASGS_hub has been specifically designed to enable the remapping of parameters within the HASGS controllers. This provides the soloist with the flexibility to manipulate specific parameters available with each preset. By empowering the soloist with control over selected parameters of either expressive or compositional nature, the interplay achieved was more significant. Moreover, the performance mediator became more aware of the macrostructure while in control of parameter mapping.

Some examples taken from “Comprovisação nº 9” were already shown in Sections 5.1.7 and 5.1.9. Below are three examples that specifically demonstrate the interaction of the HASGS controllers.

Figure 5.4: A miraweb-enabled visualisation interface for HASGS controller feedback and `Comprovisador` messaging. Messages from the host are displayed at the top. The parameter names assigned to each controller are subject to remapping upon preset change. The preset number and timer can be viewed at the bottom left.

In Video Example 5.36, the soloist manipulates the parameters **divisão.1** and **divisão.2**, which establish the tuplet ratio for each group. The tablet displaying the graphical feedback of the manipulation is visible in front of the soloist in the video.

Video Example 5.36

Comprovisação nº 9 - tuplet ratio mediated by the soloist via HASGS (graphical feedback visible).

Video Example 5.37 demonstrates the communication of a preset change – the arrival of a message is signalled by an orange flash – which readies the soloist for a contrasting musical event.

Video Example 5.37

Comprovisação nº 9 - preset change - communication with the soloist via message.

Finally, in Video Example 5.38 the soloist moderates the contour rate. The graphical white bar, which provides feedback on the **agógica** parameter, is visible in the video. We can immediately discern the effect of this parameter on the speed of the flautist's performance. More so, it is noticeable when the soloist intentionally reduces the rate to make room for his delivery. This is perhaps one of the best indications that a significant interplay was attained through the combination of these two systems.

Video Example 5.38

Comprovisação nº 9 - contour rate mediated by the soloist via HASGS.

5.1.12 Preset Alternation - Composite Gestures

While the concept of creating a musical form based on contrasting sections initially sparked the idea of implementing presets in `Comprovisador`, alternative methods include employing a combination of switchable presets to form composite musical gestures centred around the principle of causation. Communication with the soloist must be unconstrained for this method to be effective, given that the created composite gesture will rely on his or her actions.

"Comprovisação nº 10" commenced with such a composite gesture where the first preset yielded a sharp, *tutti* attack as a response and the second involved layering of individual, prolonged notes, achieved by interacting with the `nodes` object (see Video Example 5.39). The messages that were dispatched with each preset intended only to convey the algorithmic response but ended up becoming a score, prescribing a specific gesture from the soloist. Though it may appear that the soloist's playing style is influencing a unique algorithmic response, this is not the case. The mediator is actually operating like a conductor, directing the distinct components of the composite gesture. Rather, the soloist is simply interpreting the communication that accompanies the preset alteration, doing so in a very effective way.

Video Example 5.39

Comprovisação nº 10 - composite gesture: sharp hit & layered smooth chord.

Video Example 5.40 presents a segment featuring alternating *tutti* unisons and a *pointillistic* effect in "Comprovisação nº 10". The similarity between the soloist's gesture in the opening section and this instance is striking. This also contributes to the construction of musical form and is an expression of Sarath's concept of *retensive-protensive temporality* [Sar96] discussed in Section 2.1.

Video Example 5.40

Comprovisação nº 10 - composite gesture: tutti unisons & pointillism.

5.1.13 Techniques and Ornaments

Special techniques and ornaments are important as they alter the texture of the sound at a micro level, potentially enhancing changes in the overall texture. On many occasions in performances with `Comprovisador`, it can be observed that these changes often promote interaction with the soloist in such a striking way that he or she feels compelled to respond with a change of the same kind, thus creating a strong interactive bond.

Such a bond, present in Video Example 5.2 shown above (see Section 5.1.2), occurs when the soloist responds to a trill played by the ensemble, which was prescribed by the mediator, by also playing a trill, or when the mediator responds to a high note played by the soloist with a *crescendo*. There is also a moment wherein the soloist reacts to a dissonance (a reading mistake, maybe?) by altering his timbre and intonation.

These coordinated launches of ornaments or techniques are very impactful in terms of listener perception. In the context of free improvisation, such coordinated changes in timbre, register, dynamics, agogics - in short, texture – are seldom. Dialogue and interactive appropriation of ideas are more prevalent, but this paradigm does not allow for coordinated changes. Since these techniques can be easily prompted via predefined messages or notation signs, and have an immediate effect on musical texture, they constitute a significant resource in comprovisation. Refer to Figure 3.5 to view the different technique directives that are available for dispatch to a particular group. It should be noted that additional messages may be typed at any time. As mentioned above, the messenger module has a random function that automatically selects from a variety of different types of ornaments. Also see 5.23 from “Comprovisação nº 9” and 5.20 from “Comprovisação nº 10” where ornaments are used in a more polyphonic way.

5.1.14 Conducted Improvisation

In Video Example 5.6, presented in Section 5.1.2, there is a section wherein several instrumentalists engage in improvisation, prompted by the directive “Each note actually means... a free improvisation”. Using any of the available resources that allow for instrumentation and textural control (`nodes` interaction, **autoClick**, **centrifugue** and related parameters), as well as dynamics, it is possible to shape a group improvisation, as is audible in the example. Another possible indication is to improvise around each note, which is somewhere between an ornament and a free improvisation.

Some elements of surprise can be achieved by using the prompt ”Each note actually means...” used as a prefix.

In the following example (Video Example 5.41) from “Comprovisação nº 7”, musicians are instructed to “speak”, leading to an amusing reaction from the audience.

Video Example 5.41

Comprovisação nº 7 - speak.

Amusement can also arise within the ensemble, as shown in Video Example 5.42, where the instruction was to “strike the handrail”. For the surprise to bear fruit, the musicians had no knowledge of the assigned task.

Video Example 5.42

Comprovisação nº 11 - strike handrail.

Not everything always goes according to plan. During “Comprovisação nº 11”, the musicians were instructed to “speak”, which inadvertently encouraged the inattentive audience to speak even louder. This trend persisted until the conclusion of the performance. It is worth noting that this happened in a secondary school setting.

6

Use in Educational Context

6.1 Improving Sight-Reading Skills through Dynamic Notation

This section is based on the following publication: [Lou18d]. The initial parts were omitted to avoid repetition.

A possible new direction

Carlos Guedes states that one of the goals on the development of real-time composition applications is “to open a new and potentially revolutionary way of education and active enculturation with unfamiliar musical styles” [Gue17]. What about dynamic notation applications? Can they play a significant role in a new way of improving sight-reading skills?

The motivation on addressing issues related to sight-reading has evolved from two directions: 1) as a qualified solfege teacher with over fifteen years of experience, I have been interested in ways to help students improve their skills, and 2) as a creator, while developing a real-time notation system and putting it into action during rehearsals and performances, I have worked in collaboration with competent sight-readers, looking into ways of improving the system's notation interface in order to meet and expand their abilities.

The system – **Comprovisador** – was originally designed to carry out comprovisation performances using real-time algorithmic composition and dynamic staff-based notation. To engage in such musical practice, performers are expected to have excellent sight-reading skills as well as the ability to adapt to new performance situations. A **Practice Mode** was created within **Comprovisador** to help performers improve those skills while getting acquainted with the system's notation interface. Later, this was seen as an opportunity to broaden the system's application and adapt it as a tool for music students.

In order to assess the system's applicability in this new educational context, a user study with quantitative and qualitative data is discussed herein.

Music sight-reading has long been a subject of research in the field of music cognition. Many authors have pointed out pattern recognition and understanding of musical structure as a few of the most important skills among good sight-readers [Bea38, Wol76, Slo74, Slo76, WUF97]. Pianist Boris Goldovsky, interviewed by Thomas Wolf, said: "you read only a fraction of the notes and you guess at the others. A good sight-reader gets a total image of a page and extrapolates what is going on exactly" [Wol76]. Evidently, such an ability can only come from being familiarized, through years of training, with the rules and patterns common to a certain style of music, a certain repertoire. Also, this statement is based on the assumption that the sight-reader will have the chance to at least take a glance at the whole music page before beginning to play. But most dynamic score sight-readers do not have that luxury Ḧence, they have to develop other skills in order to become successful at that task. Could generative algorithms, such as the one implemented in **Comprovisador**'s **Practice Mode**, be of aid to the development of those skills?

While searching for applications or systems that use dynamic notation and aim for sight-reading improvement I did not find anything relevant. There are great amounts of smartphone applications intended for music notation learning and some do use dynamic score technology. Yet, the majority uses previously written (coded) music excerpts and it is rare to find one that joins dynamic notation technology with the power of generative algorithms.

In July 2017, during a talk at the 2nd "European Saxophone Congress", the possibility of using **Comprovisador**'s **Practice Mode** as a way for saxophonists to improve sight-reading skills in a microtonal context was pre-

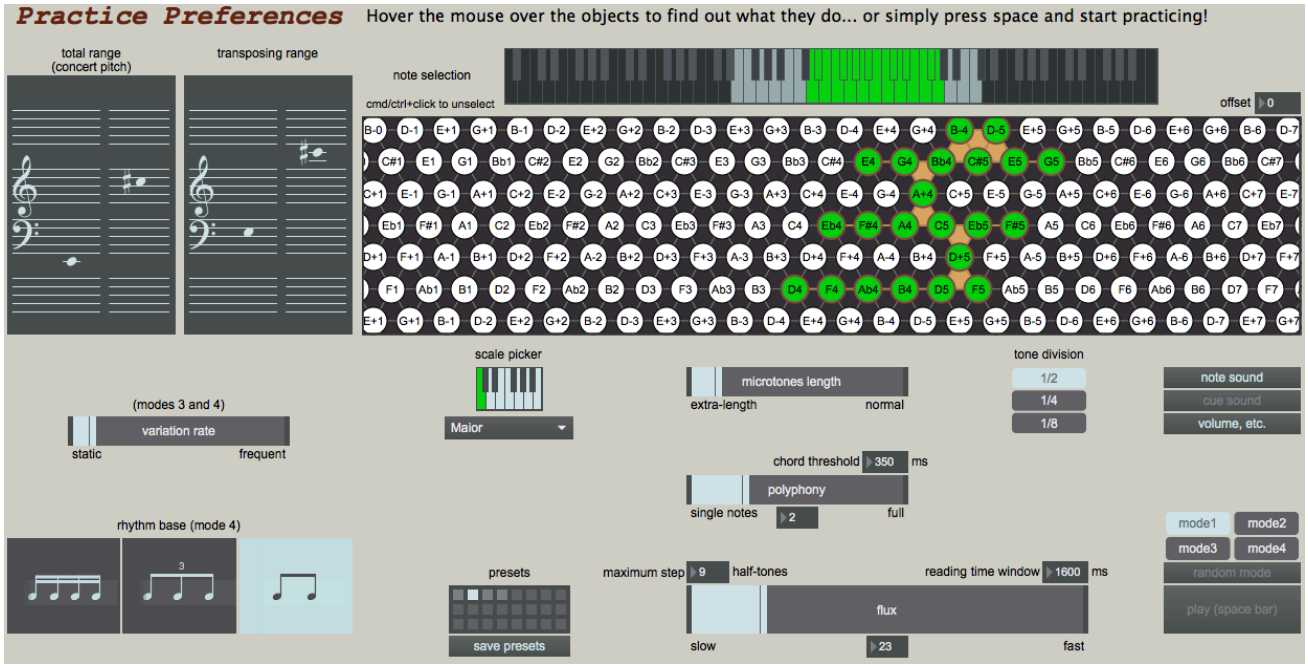


Figure 6.1: **`Comprovisador.client`** - practice preferences window.

sented [Lou17c]. A trial had been carried out with a small group of professional saxophone players and results were presented during the talk. Some adaptations were done to the system in order to be possible to collect user practice data for study. No other changes were made. Results pointed to potential benefits in using the application but it became clear that a progressive learning approach strategy would have to be devised.

Such a strategy was indeed planned out aiming not only at the microtonal issue but also at a more general context. Its implementation consisted on designing a new GUI for the **`Practice Mode`** with more controllable parameters and the possibility of storing user presets (see Figure 6.1 and Table 6.1). The goal was to enable the user to match the difficulty level of the algorithmic outcome to his or her degree of proficiency.

During the implementation of this GUI, another trial was carried out – this time with music students and teachers of different instruments – in order to assess the usefulness of this tool in a generic music education context [1]. However, parameters controlling standard rhythm notation were not yet implemented when the trial took place.

[1] It is worth noting that this tool should never be considered as a substitute for actual repertoire sight-reading, which is the best way to acquire pattern recognition skills.

parameter name	parameter description
range	allows control of range in concert pitch and in transposition (automatically set when choosing an instrument)
note selection	a selectable keyboard allows turning on or off certain notes or even whole registers
microtone selection	microtones can be hand-picked from a `bach`.tonnetz object
microtones length	enables the user more time to stabilize fingering and tunning whenever a microtone is output
tone division	selects all notes matching the set tone division
scale picker	selects all notes matching the chosen scale
polyphony	sets maximum, ranging from single notes to full polyphony (value depends on the instrument)
chord threshold	sets a threshold in milliseconds under which no chords are allowed (only single notes)
reading time window	adjusts the sight-reading window in milliseconds
maximum step	sets the maximum melodic step in half-tones
note rate or "flux"	ranging from slow to fast (proportional notation)
rhythm base	minimal units for standard notation (allowing creation of simple patterns, and to progress)
variation rate	limits the occurrence of variations of a melody (in loop mode), ranging from static to frequent
user presets	enables the user to store and recall parameter presets

Table 6.1: `Comprovisador.client` – `Practice Mode`'s parameter list (user controlled). New parameters are marked in bold.

6.1.1 Method

The trial was carried out in a music school in Portimão (south of Portugal), with 14 participants, 9 of which were students and 5 were teachers, playing the following instruments: saxophone (3 participants), violin (4), piano (3), guitar (2), double bass (1) and trombone (1). Students' ages ranged from 13 to 18. The level of experience of the participants was heterogeneous, as can be inferred by the age range and by the fact that it mixes students and teachers. Yet, none had had experience with microtonality.

Participants were individually asked to sight-read from the computer screen without any detailed explanation. As they were playing, some parameters would be manipulated in an attempt to match their proficiency level and, while doing so, I would explain what each parameter was meant to do. Towards the end of the exercise, it was explained how parameters could be stored as user presets for later recall as a way to keep track of progress. Participants were then asked to explore this feature in conjunction with the parameters previously manipulated.

With instruments that enable microtonal playing, an approach to the matter was carried out, activating only one microtonal note (in some cases, two notes) [2] and limiting the range so that the algorithm would focus on the

[2] Some solfege books [Hau77, Jol00] address note reading through a block-building approach where, for example, lesson 1 features

	Notes	Micr.	Std.R	Prp.R	Dyn.S
N Valid	14	11	14	14	14
Mean	5,14	4,55	4,14	4,57	4,57
std.Dev.	,770	,873	1,351	,938	1,158
Min.	4	3	2	3	3
Max.	6	6	6	6	6

Table 6.2: Assessment of `Comprovisador.client` as a tool for sight-reading skills improvement. Categories: standard notes, microtones, standard rhythm, proportional rhythm, experience with dynamic notation systems. Rating: from 1 to 6 (1 being not useful and 6 being very much useful).

register surrounding the chosen microtonal note. Also, longer duration time was assigned to this same note in contrast to regular notes, this way allowing stabilization.

Participants were directly observed and were videotaped while playing, for further observation. After the exercise was complete (which took around 15 minutes per participant), they filled up a form containing three sections: quantitative assessment, qualitative assessment and suggestions.

6.1.2 Results and Discussion

From observation, it was possible to perceive that all participants, with the exception of two students, were able to figure out (by themselves or needing very little explanation) how to play in sync with the bouncing ball.

In all cases, with proportional notation it was possible to match the parameter settings to the proficiency level of each individual so that it always became an interesting sight-reading challenge.

The progressive microtonal approach, starting with known notes / fingerings and adding only a selected microtonal note (assigned with a longer duration), was regarded as successful (from observation, backed by answers to the form). Violinists seemed to struggle a bit more than other instrumentalists but I was not able to find a relevant cause for that contrast. Pianists obviously did not experience this part of the exercise.

As expected, it was observed that work needed to be done in the standard rhythmic notation part, in order to enable beginner students with a viable tool.

From form responses and regarding quantitative assessment of the application as a tool for sight-reading skills

only notes C and D, lesson 2 introduces note E, and so on. This approach may be useful when applied to any type of exotic notation – as microtonal is for a large number of musicians, students and professionals alike.

improvement (see Table 6.2), results were encouraging in the category of standard notes. Results were also positive in the categories of proportional notation, experience with dynamic notation systems, and microtones. Although, the latter had less three responses (pianists). The lowest rated category was standard rhythm, as expected.

There was an optional category "other" where two participants (both of them wind instrument teachers) added "tunning", rating it with the highest score. They highlighted the benefit of playing in tune with the sound produced by the computer.

Regarding qualitative assessment of functionality and appearance, the responses were the following. The bouncing ball was considered useful / effective / helpful, except for two students who deemed it confusing. The dynamics bar was considered useful / effective / legible, but nonetheless some participants reported it to be too fast / difficult to comply with / very challenging; one participant highlighted the 3D animation as a good solution. Verbal instructions had very similar responses.

Regarding the observed and reported ease to synchronise with the bouncing ball, it is in line with Richard Picking's findings on his study where he compares three types of animated time-location tracker, in the context of reading music from computer screens (versus reading from paper). The subjects of his study reported the "jumper tracker" (which is analogous to `Comprovisador`'s bouncing ball) to be the preferred one [Pic97].

My preliminary conclusion taken from observation and commentaries is that young students tend to ignore dynamics and verbal instructions – and they are fine with it. Advanced students and teachers tend to get a bit frustrated when not able to comply with everything (notes, dynamics and instructions) but also feel rewarded when they do.

There were many voluntary commentaries and suggestions. The preset management system and GUI for parameter control were regarded as having good configuration / ease of use / good control over "excess of randomness". Pianists complained about insufficient spacing between staves. There is actually only enough space for the central C line – which is standard in many computer music applications that use GF staves – but pianists are not necessarily used to it. Some participants mentioned that the duration line should be of a lighter color because it interferes with the perception of the staff-lines. This is now fixed, as shown in Figure 3.11. An interesting suggestion was to implement a way to have harmonic structures as a base for the generative algorithm.

Without surprise, many comments about standard rhythm were made, for example: "Everything is changing all

the time due to excess of variations", or "It needs patterns".

To sum up, results seem encouraging (although they have to be put in perspective regarding the small sample size) suggesting there are advantages in the use of `Comprovisador.client`'s `Practice Mode` as a way of improving certain sight-reading skills, with special focus on skills pertaining to the dynamic notation realm. Regarding the least explored field – standard rhythm – I believe there is equal potential, now that the GUI's development is complete.

6.1.3 Moving Forward

Much work has been done, meanwhile, in terms of correcting the reported issues, namely the color of the duration line, which is now translucent green, as well as the standard rhythm controls. Apart from the controls, standard rhythm was enhanced at the quantizer level[3]. Here, instead of writing two $\frac{4}{4}$ measures, the algorithm writes eight $\frac{1}{4}$ measures. This allows two things:

1. complex patterns are conveniently delineated by bar lines and thus easier to decipher;

2. long notes unfold into tied quarter notes, making it easier to count the beats – which is especially important when a loop is set in a way that a long note becomes truncated.

This can be seen in Figure 3.12c: there are three tied quarter notes that would otherwise be written as a dotted half note. The loop region is excluding the third quarter note. If it was written as two $\frac{4}{4}$ measures, the loop region would end in an ambiguous, white portion of the measure, corresponding to the duration of the dot, which would be confusing for the reader.

In the medium term, I might pursue the suggestion of implementing a way of having the generative algorithm obey a harmonic structure. This structure could be cyclic or generative.

One goal, of course, is to do further testing, if possible with a larger sample size and during a longer period of time, so to be able to measure actual learning progressions and observe commonalities that might emerge among multiple participants.

[3] These improvements have been discussed in more detail in Chapter 3

6.2 The Contribution of Joaquim Nascimento

This section is original and consists of a reflection on the findings of Joaquim Nascimento in his study [Nas19].

With regard to the usefulness of the **Comprovisador** as an educational tool, it is pertinent to mention the study conducted by guitarist Joaquim Nascimento in his Music Education Master's thesis, titled "O Contributo do **Comprovisador** na Leitura Musical à Primeira Vista" [Nas19] (The Contribution of **Comprovisador** in Music Sight-Reading – my translation).

As stated in Chapter 3, Joaquim Nascimento participated as an ensemble member in "Comprovisação nº 3". He expressed great enthusiasm for using **Comprovisador.client**'s **Practice Mode** to improve his own sight-reading skills following the concert. Later, he expressed an interest in using this tool to carry out research for his Master's degree. The research focused on a group of young guitar students who utilised the technology to enhance their instrumental sight-reading abilities. Due to my own motivation to assess the system's educational potential, I granted him permission to use **Comprovisador.client** in his research and opted to collaborate by modifying the system to best serve the necessities of young music learners and the researcher. The primary design consideration aimed at enabling users to modify the complexity of the algorithmic output according to their proficiency level. This goal was achieved by integrating additional controllable parameters and providing users with the ability to save their preferred settings, as detailed above in Section 6.1 (refer to Table 6.1).

During our collaboration, I suggested a step-by-step approach, beginning with a limited pitch set consisting of a few open strings and then gradually introducing a few stopped notes to the set. This approach is rooted in the concept of gamification, whereby the objective is to gain skills by overcoming increasingly challenging levels. My suggestion was accepted, and a variety of observation methods were used. In the ensuing paragraphs, some of the conclusions[4] from Nascimento's research are presented.

The author has concluded that **Comprovisador** has a facilitating effect on the development of sight-reading skills. This is based on the correlation between anticipation and sight-reading abilities. The author found that students were focused on playing the notes without concerning themselves with the use of the correct fingering. This resulted in the students making real-time decisions to use alternative fingerings [Nas19].

The use of **Comprovisador.client**'s **Practice Mode** in music sight-reading practice led the author to deduce that individuals naturally assume a correct posture which can be linked to empathy towards the object.

[4] A citation of the original work is presented in Appendix F, in Portuguese.

This results in a positive and relaxed attitude towards mistakes, without any tension in response to potential errors [Nas19].

The author observed that students prepared their left hand meticulously by positioning their fingers in close proximity to the notes that could be expected. This facilitated technical improvement of their left hand as well as the acquisition of new skills, resulting in fingering automation and successful sight-reading. According to the author, this practice facilitated a seamless connection between vision and execution [Nas19].

It has been observed that students were fully engaged in a state of flow while working with `Comprovisador`. The criteria for the state of flow were met, according to the author: clear objectives, high concentration, immediate feedback, and a sense of control over the situation. It is worth noting that the activity proposed was neither too simple nor too complicated, and it focused on a particular field of action [Nas19].

The author reports a noteworthy enhancement of engagement and motivation among the students as a result of their interaction with the software. High levels of consistent motivation were observed. Ultimately, the systematic use of the "game"[5] was found to increase students' interest in practising music sight-reading [Nas19].

These findings authenticate the outcomes obtained in the paper [Lou18d], presented in Section 6.1, with the added benefit of a more extensive sample size and prolonged observation period.

Based on these findings and my previous experience using `Comprovisador` in rehearsals with teenage students, I recently submitted a project to the Loulé Conservatoire - Francisco Rosado (refer to Appendix F). The project aims to establish small ensembles of younger students who will work with `Comprovisador` to learn new repertoire and improve their instrumental sight-reading abilities in the context of animated notation. The anticipated benefits of the proposed pedagogical model are described in Section F.4 of Appendix F. The success of this project could translate into a positive impact on society.

6.3 A Generative Tool for Precomposition Based on Sight-Reading

The `Practice Mode` has demonstrated its versatility by serving a range of purposes beyond its original design, which aimed to familiarise performers with `Comprovisador`'s notation interface and its idiosyncrasies in a simulated performance setting. As described above, it has been successfully used in an educational context

[5] – referring to the gamification of the sight-reading process enabled by `Comprovisador.client`'s `Practice Mode`.

to improve sight-reading skills and to increase motivation for this task. Personally, I can testify to the usefulness of this resource for enhancing sight-reading ability on the guitar and, additionally, for solidifying both standard and unconventional harmony fingerings in muscle memory.

Another unintended purpose that the `Practice Mode` served for me in relation to chord sight-reading practice was to generate precomposed material for a fixed media work entitled "Mahjong". This piece was then used as the electroacoustic basis for a homage for choir and electronics entitled "#ascoisasdozé", dedicated to the composer José Luís Ferreira. This piece was premiered at the ESML on 9 May 2019. The `Practice Mode` was used to stochastically generate chords, which were then sight-read on the guitar and recorded. The audio recording of the sight-reading performance was then subjected to improvisation using granular synthesis software for further cherry-picking[6] and editing in a DAW (Digital Audio Workstation).

[6]Gonçalo Gato [Gat16] employs algorithmic generation of precompositional material and subsequent cherry-picking in his process. However, his algorithms tend to be deterministic in nature and produce symbolic output material.

7

Conclusions

At the start of my doctoral research, my aim was to develop a system capable of facilitating real-time interaction between a soloist and an ensemble, where the musical discourse of the former is highly improvisational, while that of the latter is algorithmically organised. The system `Comprovisador` was therefore designed to enable mediated soloist-ensemble interaction employing machine listening, algorithmic compositional procedures and dynamic notation in a networked environment.

My research was driven by the question of how a real-time composition system featuring dynamic staff-based notation could enhance the practice of **comprovisation** involving improvisers and sight-readers. The aim of this study was to investigate the potential benefits of this system within the context of musical performance predicated on composed and improvised elements coexisting in aesthetically relevant interdependence.

Several aesthetic objectives informed the design of the system. These objectives were constructed by considering the significant similarities and disparities between composition and improvisation, and founded on listener perception. A number of anticipated problems that could arise from the unique idiosyncrasies of extreme sight-reading in a networked environment were also addressed.

Eleven public performances were conducted using `Comprovisador`, forming the corpus for this practice-based research. The system was subject to iterative development, with the resultant development and exploration of specific **modes of operation** in real performance settings.

Through a series of experimental rehearsal stages and analysis of the resulting performances, this study aimed to shed light on how this innovative approach to real-time notation could broaden musical horizons, encourage experimentation, and improve the overall experience of comprovisation performance.

Finally, the study explored a parallel implementation of the system in the domain of music education, yielding positive outcomes.

It was noted that `Comprovisador` occupies a unique position at the junction of three distinct creative spaces relevant to computer music performance. The system takes an improvised performance as input to algorithms that generate a composed response in real time, a key characteristic of **player-paradigm Interactive Music Systems**. With regards to output, `Comprovisador` produces live-generated score parts (**Animated Notation**[1]) that are distributed and synchronised across computer screens on a local area network (**Networked Music Performance**). This presented a chance to explore the different crossroads between these three artistic domains, alongside the numerous connections to neighbouring practices. Moreover, it was an opportunity to draw on disciplines that involve non-digital methods of comprovisation, including **paper scores incorporating context-dependent elements** and, specifically, **gesture languages for conducted improvisation and live composition** (an area in which I had some practical knowledge).

Upon reflecting on the intricacies of real-time composition and comprovisation, my attention turned to listener perception, and thus to aspects such as synchronisation and cohesion, which also relate to the aforementioned gesture languages. I also developed an appreciation for the advantages of staff-based notation in controlling pitch and harmony.

While examining the prevailing aesthetics in the fields of NMP and RTN, I observed that they are characterised

[1]– Animated Notation may be referred to as Real-Time Notation, but it is important to differentiate between truly live-generated and merely live-animated notation. In the case of `Comprovisador`, this would be live-generated RTN.

by slow-attack sounds, pointillism and long, stepless sounds. The objective is to conceal any synchronisation gaps so that they are not perceived by the listener. In terms of score type, graphic and other non-staff scores seem to be preferred by a large proportion of RTN practitioners. It has the advantage of being more descriptive than prescriptive, allowing more freedom of interpretation. As my focus had shifted towards synchronisation and staff-based notation, I deemed it valuable to experiment with a novel approach to observe the outcomes.

Given my aim to incorporate sharp, cohesive gestures into a real-time generative context grounded in the sight-reading of staff-based scores, I anticipated some problems. These encompassed: 1) the high likelihood of sight-reading errors, 2) the challenge of synchronising musicians when faced with unpredictable, unpractised, and un-conducted scores, and 3) the delays arising from computational, physical, and cognitive processes.

These challenges presented opportunities for a dialectic process that contributed to shaping my aesthetic objectives. Potential methods for reducing the occurrence of sight-reading errors and incorporating this occurrence into the aesthetics were examined. Among the most relevant methods, I highlight a versatile notation interface capable of supporting a range of real-time reading modes, a systematic approach to a reading time window, a graphical cueing strategy in the shape of a bouncing ball that facilitates both score navigation and synchronisation, and an audio score comprising cue notes (for singers) and an adaptable metronome.

`Comprovisador` was designed iteratively, with **aesthetic goals** in mind at every step. Following is a summary of these goals.

As previously stated, a preference was early on given to staff-based notation, as it allowed greater control over specific musical parameters, albeit at the cost of limiting performer creativity and flexibility. Additionally, synchronised attacks and cohesive musical gestures were noted as preferred techniques due to their potential to evoke a sense of order versus disorder (composition versus improvisation).

This kind of rhythmic precision can be implemented within the framework of proportional notation, where two possible time conceptions can be used – chronometric time and metronomic time – leading to a range of rhythmic exploration possibilities. In a metronomic time conception, rhythmic exploration can mean musical phrasing with different proportional note values or, conversely, using subdivisions of the beat (tuplets) to allow juxtaposition and/or superposition of moderately complex tuplet ratios. In relation to chronometric time, it has the characteristic of producing a more irregular type of rhythm that can be more interactive vis-à-vis the soloist's input.

The use of standard rhythmic notation has presented some problems, as it makes sight-reading more difficult by adding different elements at the same time. Despite its inherent problems, some of the main advantages of standard rhythmic notation are the ability to create well-organised textures that incorporate melodic, rhythmic and harmonic components, as well as formal elements such as repetition and variation. Other goals associated with the implementation of standard rhythmic notation within `Comprovisador` include the exploration of effects such as polyphony, polymeter and polytempo, as well as the use of precomposed passages that borrow from the aesthetics of open form.

Motivic exploration is one of several strategies that can foster the perception of a compositional process. This objective has been accomplished through the use of fluid rhythmic techniques based on proportional notation, which allow the performer to focus on the melodic content, without the constraints of standard rhythmic notation. Motivic exploration was also attained through a melodic resource known as *acciaccatura*, implemented in `Comprovisador` using proportional notation, but effectively cued by the bouncing ball and the adaptable metronome.

Controlling instrumentation and musical form have been long-standing objectives that have continually been refined. Some significant features that have been prioritised include: predefining a new set of instruments for each session, predefining (composing) a musical form using a preset manager, controlling two independent groups with complementary musical functions, defining instrument clusters for timbral switching, generating sung text for a consistent choral sound, communicating via text messages, producing textural effects and performing expressive manipulation of various parameters for transitioning between structural points in the musical form.

One particular goal was to deploy a device, named `Practice Mode`, to familiarise musicians with the notation interface and unique features of the system. This device proved beneficial in two unforeseen ways: firstly, it facilitated the remote gathering of valuable feedback from musicians, resulting in personalised enhancements to the system, and secondly, it became a resource capable of supporting the educational field in enhancing sight-reading skills.

In light of the **lessons learned** from this research, resolving the question of how to handle the error has emerged as pivotal. A real-time notation system relying mainly on staff notation involves a considerable amount of error expectation. Sight-reading is a difficult task and errors of pitch and timing are bound to occur. Thus, such a system must consider incorporating this error factor as part of its aesthetics.

In **Comprovisador**, incorporation of the error factor is achieved through various methods. The Harmony algorithm includes a reading time window, typically providing sufficient preparation time for the player to read and prepare the note in his or her instrument. However, this may be impacted by multiple factors such as note rate, instrument type, and performer experience. The aesthetic potential of the reading time window emerges when the soloist can anticipate the response delay and engage with it accordingly.

Variation 1 of algorithm Contour demands that musicians refrain from synchronising with one another. This allows for a more laid-back approach from the sight-reader towards the displayed melody, potentially avoiding melodic errors, while simultaneously creating a potentially captivating heterophonic texture.

When the quantised variation is activated, the notes of the original melody remain unchanged. This allows the performer to concentrate exclusively on the new rhythm. From the listener's point of view, I find it interesting to perceive the transition from unsynchronised to synchronised and vice versa. However, more work is needed to refine the standard rhythmic notation interface, which aims to resolve issues like interruptions caused by melodic transformations and loop manipulation, alongside the eight-beat cap that currently impacts the use of precomposed passages. Such interruptions have a detrimental effect on sight-reading, negatively impacting musical fluidity. Thankfully, a new and improved metered notation interface is currently in development (see Section 7.1.1 below).

This research was predominantly focused on exploring and analysing the various modes of operation that were devised to achieve the specified aesthetic objectives, within the framework of practice-based research. In the succeeding paragraphs, these insights are elucidated.

Extensive thought has been devoted to the control of instrumentation. Using four slots within both control groups, any combination of the two algorithms may be deployed simultaneously, allocation distinct instrument sets and carrying out distinct responses that enhance each other. As a simple example, certain instruments could play a soft drone while undergoing dynamics modification, just as others may skim across a high-pitched melodic fragment while undergoing agogics mediation. This has the potential to create intriguing musical structures, particularly when pre-planned with the preset manager to master the musical form.

In fact, one key finding from this research is the significance of the capability to edit presets and to use them for composing and ultimately delivering a successful comprovisation performance. Accordingly, the automated capabilities provided by the Node Cluster Edit Mode were deemed favourable. Node clusters permit the use of both manual and automated switching between timbral sets, and it is possible to create and modify node

clusters across various slots in this mode.

The autoClick function enables the automated switching of timbral sets. When activated, the crosshairs within the **nodes** object automatically select a specific cluster, enabling the algorithm to assign a chord to the corresponding set of instruments. The application of this feature has achieved satisfactory musical outcomes, varying from seamless timbral transitions to significant density contrasts.

Comprovisador has a rich palette of harmonic generation devices at its disposal. The different approaches to note selection and register transposition in chord generation have proven useful in creating coherent and contrasting harmonic fields. In particular, the default modulus-based register transposition mode creates a blend of symmetry and unpredictability that I find attractive. However, the fixed register mode (no transposition) is optimal for responding to soloists playing harmonic instruments, depending on their playing style. It is able to produce a harmonic field that is very coherent, almost reverberant, and even akin to real-time orchestration. The resultant harmony aligns more closely with the soloist's performance, as the notes remain in their original register and are not transposed. Moreover, harmony with spectral features can be produced by employing just intonation, and other tuning systems such as Bohlen-Pierce can be conveniently incorporated.

The reading mode associated with Variation 1 of the Contour algorithm has undergone considerable modification throughout its developmental phases. This has led to an augmented capacity to stimulate diverse degrees of musical expressiveness among performers. During its initial stages, it did not indicate rhythmic values, as notes were evenly distributed across the staff in proportional notation. From this point, it evolved to represent of the rate of playing by stretching or compressing the space between the notes along the x-axis. It also allowed for articulation by adjusting the length of the note duration lines. Currently, it is possible to modify the size and position of a loop region, and to display rhythmic durations proportionally. Moreover, it is possible to realise algorithmic transformations of a contour in reaction to input from the soloist or through mediator intervention. This feature has proven to be advantageous for the musical collaboration during several performances, at times contributing to the creation of a dialectical relationship between the soloist and ensemble (between improvisation and composition).

Various rhythmic, melodic and textural effects have been explored through easily applied parameters or functions that can be accessed via corresponding parameter knobs. Parameter **divisão**, used within Variation 2 of algorithm Harmony (**metroRhythm**), divides a note into the required number of subdivisions of the established pulse – or tuplets. The use of **divisão** in two separate control groups generates polyrhythmic patterns, assuming both groups share the same pulse. In some cases, achieving precise transitioning between tuplet ratios has

posed a challenge to musicians – even professionals. The implementation of an adaptable metronome that can assist musicians with such transitions by playing the appropriate subdivisions was considered beneficial, according to the musical results obtained in "Comprovisação nº 11".

Parameter **shortMotiv**, employed in Variation 1 of algorithm Harmony (**threshRhythm**), produces an *acciaccatura* consisting of the required number of notes by using the most recent melodic intervals played by the soloist. The phrase is transposed to resolve on the designated chord note. The *acciaccatura* effect can lead to diverse results, ranging from contemplative to active melodic interplay, evoking imitative counterpoint, to more complex contour outcomes. The extent of the effect largely hinges on the soloist's rate of events and their use of a more or less conventional playing technique. Interestingly, the application of *acciaccature* to pairs of instruments can yield a remarkably cohesive result. It can therefore be seen as an effect with high potential for promoting interactivity and a sense of musical organisation.

The **centrifugue** function can be deployed within Variation 2 of algorithm Harmony (**metroRhythm**). In conjunction with other parameter settings, such as **agógica**, **articulação**, **%densidade** and the number of instruments available to play, **centrifugue** can produce a series of usable effects with distinct characteristics. For example:

- a cycle of individual attacks within a chord;
- a cycle of subchords within a chord;
- low-density pitch overlay, enabling slow blending of harmonies and emergence of timbral qualities (irregular cycle);
- a slow, contemplative cycle of individual timbres and sound locations (regular cycle);
- a *pointillistic* effect with density manipulation; or
- a polyphonic effect, when used with ornaments (all successive entrances within the cycle will perform the same ornament – imitation) or a prompt to "improvise around each note".

The initial design of this feature had relied on graphical automation within the **`nodes`** object, resulting in an impressive sweeping effect (*balayer* – cf. [Ben99]). However, this approach had demonstrated suboptimal computational efficiency.

The messenger system can be used to quickly and effortlessly send instructions to a group of musicians. Prescribing ornamentation or specific techniques to a particular group of performers through conventional notation

signs or simple text instructions can have a noteworthy impact on musical texture, given the coordinated nature of the response. Both the audience and soloist can clearly discern this effect, thus reinforcing the perception of musical structure and the interaction feedback loop.

The messaging system provides a further advantage by enabling the mediator to encourage musicians to improvise and control the scope of their improvisation (using the directive "Each note is actually a free improvisation", among others). Group improvisations can be shaped by utilising the available resources facilitating instrumentation and textural control, such as `nodes` interaction, **autoClick**, **centrifugue** alongside related parameters.

Communication of musical instructions to the soloist is a resource offered through the messenger module. Although initially deemed optional, performance experience has demonstrated the aesthetic advantages of this resource and the consequential detriment when it is not utilised, as it can help resolve musical problems in real time.

The contribution of this type of communication to the musical quality of the performance "Comprovisação nº 10" is noteworthy. Here, a combination of switchable presets facilitated the creation of composite musical gestures centred on the principle of causation. Every preset, as deployed by the mediator, automatically dispatched a specific message to the soloist, who interpreted it akin to a musical score.

The control interface of `Comprovisador` exhibits great flexibility by enabling the control of instrumentation, phoneme-based text for singers, and algorithmic response, in addition to the management of parameter presets, which allows for both the programming of musical forms and the creation of composite musical gestures. Unique textures can be realised through the use of the Part Unifier feature and an unconventional approach to algorithm slot allocation. In addition, when the soloist employs an augmented instrument containing built-in controllers, the control interface operation can be divided between the mediator and the soloist for the benefit of a more consequent interplay.

Several unforeseen challenges have emerged that were not previously anticipated. These issues have been addressed through iterative approaches until a satisfactory solution was found. This has been the case with the bouncing ball cueing strategy and, more generally, with synchronisation.

At an early stage, inadequate graphics processing was found to result in the bouncing ball being rendered at a low frame rate, adversely affecting the timing accuracy of players. After testing various graphics rendering

systems, it was determined that OpenGL, which offered hardware acceleration, was the most effective, improving synchronisation among musicians. This presented a chance for an enhanced reworking of the entire notation interface using OpenGL.

Despite the significant improvement brought about by the OpenGL implementation, it became apparent that Wi-Fi network jitter was also having a negative impact on synchronisation. This problem was addressed by implementing a system based on time-stamps in conjunction with an approach similar to the Network Time Protocol, following Dannenberg's concept of "time-flow". Testing was conducted and the results were successful.

However, in a real-world scenario (during the performance of "Comprovisação nº 10"), it was found that the synchronisation between the musicians was unsatisfactory in large space settings, when relying solely on the bouncing ball. An adaptable metronome was therefore developed, which had a positive effect in complex musical situations. The bouncing ball is still considered to be a useful device, taking on the secondary role of score navigation.

Yet, during "Comprovisação nº 11", which took place in a spacious hall (exceeding 600m^3 in volume), there was a timing conflict between the soloist's tempo and that of the ensemble. In this case, the synchronisation between the members of the ensemble was not compromised - a sign that the metronome, in conjunction with the previous measures, was effective. But it is also a sign that certain scenarios require special consideration, despite a system that performs effectively.

These conclusions represent a synthesis of the measures trialled and evaluated under varied performance scenarios. **It is shown that the system's unique design allowed the implementation of several modes of operation that led to the achievement of aesthetic innovations**. Therefore, these findings indicate the potential of `Comprovisador` to broaden musical horizons and promote experimentation. Consequently, this offers a stimulating prospect for future studies in the field.

7.1 Future Directions

The following section aims to address unresolved issues and provide hints for future exploration based on the insights gained.

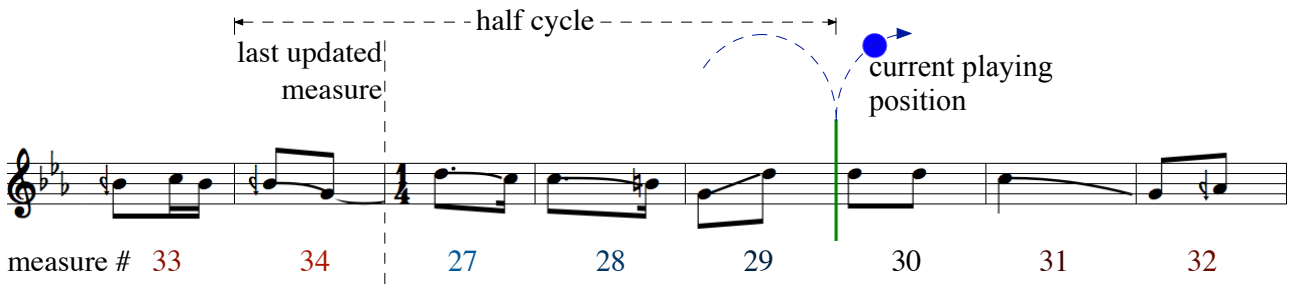


Figure 7.1: Score with measures dynamically updated in cycle and bouncing ball.

7.1.1 System Improvement

Improved Metered Notation Interface

A new way of dealing with metered notation has been developed but not yet fully implemented in **Comprovisador**. It has however been tested in a real-world setting within a different system subsidiary to **Comprovisador** (see Appendix D). In this setting, the score was synchronised to multichannel electroacoustic tracks. This new approach is thus well suited for precomposed music and it is undergoing modifications to enable its use with real-time generative scores (see Video Example 7.1 below).

Here are its characteristics (see Fig. 7.1):

- measures are unitary and may include any denominator ($\frac{1}{4}$, $\frac{3}{8}$, $\frac{3}{16}$, $\frac{5}{16}$, etc.);
- the number of measures per system depends on screen resolution and is optimized for 8 measures on the more common screens;
- measures are dynamically updated in a cycle to avoid disrupting the reading process – measures that have been read are replaced at a distance of half a system (half a cycle)[2].

Since the 10th performance of **Comprovisador**, precomposed material can be deployed at any given time and can be worked as one would in an open form piece. Any number of precomposed loops can be uploaded to the clients but the duration of each loop is currently restricted to a maximum of 8 beats. This feature was developed using the Quantum Loop framework.

Since its introduction, the Quantum Loop mode has encountered several problems. These include technical

[2]For a standalone interactive example of the score in action, visit **https://glitchscore.glitch.me/**. It might be necessary to reload the page and/or wait a few moments for the musical staff to appear.

hurdles and design issues with the notation interface. The conception enables loop transformations (alterations to the position of loop region boundaries) and/or melodic transformations (contraction, expansion, or transposition) to occur while musicians are playing, resulting in notes changing before their eyes. This inefficiency has lead to performer anxiety and disruptions in the musical discourse.

The new solution will prevent the discontinuity issues – and associated performer anxiety – in exchange for a delay in the outcome of the manipulation. It is, indeed, analogous to the proportional notation modes one and two, with their proven reading time window approach and the notation wrapping around to the beginning of the staff.

However, a new problem may be on the horizon. Will that delay affect the audience's perception of causality? Will it not be confusing to hear a soloist-induced transformation four beats after the triggering event? Perhaps a reading window of four beats is redundant. Two beats might be sufficient, or it might be possible to adjust the window size automatically according to the tempo, with faster tempi requiring more beats.

Video Example 7.1

Comprovisador's enhanced metered notation interface prototype demonstration.

New algorithm

In 2020, I created four pieces for solo flute, commissioned by Maria João Cerol, to be premiered on 4 March 2020 at the Colégio Almada Negreiros, FCSH, Universidade Nova, Lisbon. These pieces were composed using a CAC (deferred time) algorithm that I created for this purpose, using **Markov chains** applied to intervals. The algorithm takes a short contour as input and returns a complete composition of proliferations based on simple and compound intervals of the contour (inspired by the compositional techniques of Christopher Bochmann). The algorithm was developed for use in a real-time environment and could be integrated into `Comprovisador` (as initially intended). The advantages of this approach could result in a higher capacity to generate melodies from the initial input, resulting in the production of new and constantly changing material while remaining connected to the original input.

Figure 7.2 displays the opening section of the third composition, entitled "Fachada". The initial numbered intervals indicate the original contour which serves as the input.

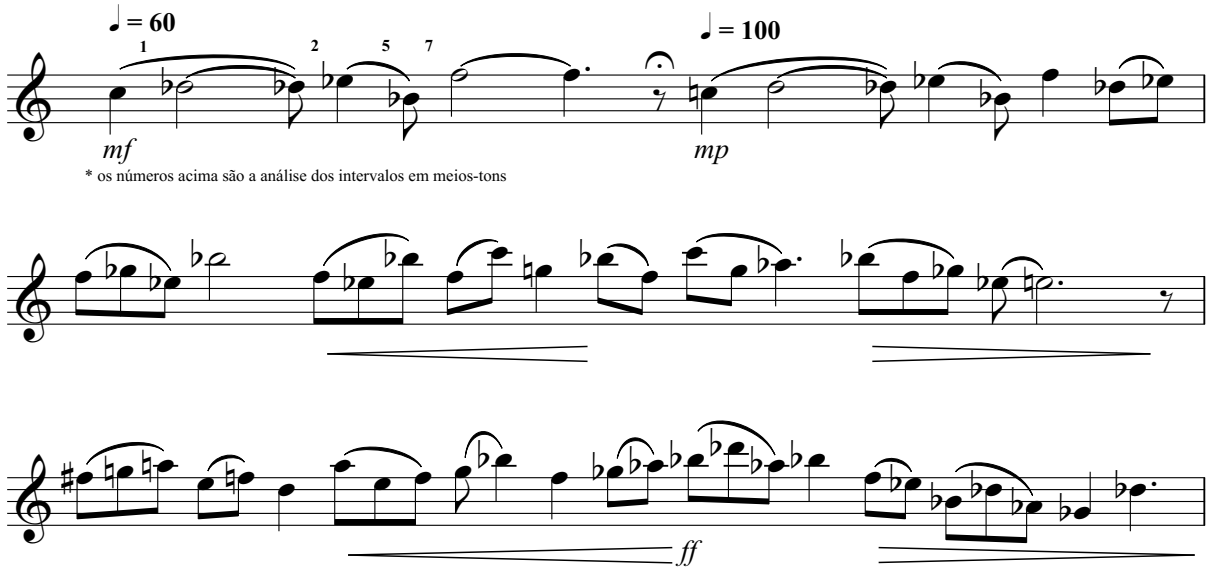


Figure 7.2: "Fachada" (extract), for flute – generated by an algorithm based on Markov chains, inspired by the compositional techniques of Christopher Bochmann.

7.1.2 New Musical Goals

An ultimate creative goal is to plan and perform a dramatised work in the shape of a *comprovised* opera. Such a work would involve a performance in which the singer(s) have complete freedom in their approach to delivering the *libretto* while maintaining meaningful interactions with the instrumental accompaniment. This is made possible by the system's ability to store and recall presets of algorithmic parameters. This significant feature – the preset manager – was introduced out of the essential need to manage musical structure and a growing number of controllable parameters experienced during the execution of "Comprovisação nº 2" (refer to Chapter 4). The preset manager, linked to an annotated timeline, enables the prior composition of the musical form by specifying the algorithmic response types characterising each section. The improvisational, contextual input will vary, and the algorithmic response will differ. However, the nature of the response will be predictable and directly impact the discourse of the soloist. And in the case of a dramatised work, meaningful correlations between the emotions expressed in the text and the musical mood specified in the corresponding presets could be established.

A

Initial IP Port Numbers (Table)

The following table presents the initial IP port numbers assigned to each instrument type. Note that a further number is added for each player of the same instrument. In this way, each instrumental part is given a unique identifier (e.g. trumpet in Bb, player 3 => 52000 + 250 + 3 = 52253). Note that a further number is added for each player of the same instrument. Importantly, the table has ample space for expansion, allowing for the incorporation of unplanned instruments with ease. As it currently stands, the system can be used with a large orchestral ensemble (e.g. 150 violins) without conflict.

	Woodwinds	Brass	Percussion	Keyboard	Singers	Strings
+	50000	52000	54000	56000	58000	60000
0	Piccolo	French Horn	Timpani	Harp	Soprano	Violin
50	Flute	Wagner Tb. Bb	Woodblocks	Piano		
100	Alto Flute	Wagner Tb. F	Tom-toms	Piano (4H hi)	Mezzo	
150	Bass Flute	Picc. Trumpet	Multi-perc.	Piano (4H lo)		Viola
200	Oboe	Trumpet D		Celesta	Alto	Cello
250	Oboe in A	Trumpet Bb	Glockenspiel	Harpsichord		Contrabass
300	Eng. Horn	Flugelhorn	Vibraphone	Harmonium	Tenor	CB (5str)
350	Heckelphone	A trombone	Tubular Bells	Organ (man)		CB solo (F#)
400	Clarinet Eb	T trombone	Gongs	Organ (ped)	Baritone	Gamba
450	Clarinet Bb	B trombone	Xylophone	Synth		
500	Clarinet A	Euphonium	Xylorimba	Accordeon RH	Bass	
550	C. Bassetto	Tuba	Marimba	Accordeon LH		
600	Bass Clarinet			Melodica	Child Sop.	
650	CB Clarinet			Mandolin		
700	Bassoon			Guitar	Child Mez.	
750	Contrabassoon			Guitar (7str)		
800	Sopranino Sax			Portuguese Guitar	Child Alto	
850	Soprano Sax			Bass Guitar		
900	Alto Sax			Electric Guitar		
950	Tenor Sax			Electric Guitar (7str)		
1000	Baritone Sax			Electric Bass		
1050	Bass Sax			E. Bass (5str)		
1100	SS Recorder			E. Bass (6str)		
1150	S Recorder					
1200	A Recorder					
1250	T Recorder					
1300	B Recorder					
1350	CB Recorder					
...						

Table A.1: Initial IP port number according to instrument type

B

Um sistema para coordenação improvisação/composição, com intérpretes humanos

This article is in Portuguese, as it was written at the beginning of this research and has not been published (although it was presented to a jury in the first year of the PhD). The first sections have been omitted to avoid redundancy. The algorithms used in "Comprovisação nº 1" are analysed, which helps to contextualise the development of the current algorithms. Practical and aesthetic assessments are also made. Attention should be directed towards the last table, which includes almost all of the implemented resources and functionalities. The history of these resources and functionalities can be found in Chapter 4 and their results have been analysed in Chapter 5.

ABSTRACT

`Comprovisador` is a system that enables the coordination between the improvised musical discourse of a soloist and the musical response of an ensemble of human performers during a musical performance, using real-time algorithmic composition and dynamic notation. The system presented utilises multiple networked computers and has undergone testing in a concert setting. This has permitted the assessment of the primary objectives and identification of practical and aesthetic challenges, along with their resolutions.

Keywords: musical improvisation; algorithmic composition; dynamic notation; graphic interface.

Comprovisador: um sistema para coordenação entre improvisação livre e composição algorítmica em tempo-real, com intérpretes humanos

3. ALGORITMOS DE COMPOSIÇÃO EM TEMPO-REAL

Tradicionalmente, no campo da composição assistida por computador, existem dois tipos principais de procedimentos de composição algorítmica (Dodge & Jerse, 1997) – procedimentos estocásticos e procedimentos determinísticos. Os algoritmos que concebemos para o "Comprovisador" empregam ambos os tipos de procedimentos, mas talvez com maior incidência dos segundos.

Gostaríamos de relembrar que a nossa primeira preocupação estética foi no sentido de assegurar que o ouvinte seria capaz de perceber que o material composto algoritmicamente era originado na improvisação do solista, e que a nossa prioridade inicial se centrou nas questões de campo harmónico e textura.

Olhemos, então, para os algoritmos desenvolvidos até este momento e para as principais características do seu modo de funcionamento.

3.1 - "contour"

Directiva de leitura: *loop* (repetir livremente).

Resumo dos parâmetros manipuláveis: dinâmica; agógica; articulação; densidade; comprimento; botão "refresh Chord"; botão "pausa Geral".

Modo de funcionamento:

- são geradas frases melódicas a partir dos últimos intervalos tocados pelo solista (procedimento determinístico) – as frases deverão ser tocadas em *loop*;
- o comprimento das frases (determinado pela manipulação do respectivo *fader*/ *slider*) varia entre 2 e 7 notas;
- a nota de partida, para cada instrumento, será uma das 5 notas diferentes mais recentemente tocadas pelo solista, sendo a instrumentação gerida por um procedimento estocástico;
- o registo da nota de partida de cada frase é escolhido a partir de um âmbito adequado a cada instrumento, através de um procedimento estocástico;
- se a densidade (manipulável) for superior a 5 instrumentos, haverá repetição de notas de partida (salvo diferença de registo);
- se a densidade for inferior a 5, seguramente, uma das notas de partida será a última que o solista tocou, sendo a(s) restante(s) nota(s) de partida escolhida(s) aleatoriamente de entre as 4 ainda disponíveis;

- caso alguma das notas da frase ultrapasse o registo do instrumento, a mesma é suprimida da frase;
- a manipulação do parâmetro agógica faz com que as notas da frase se afastem ou se aproximem, graficamente (afastamento = *rit.*; aproximação = *accel.*);
- a manipulação do parâmetro articulação faz com que as barras que representam a duração se alonguem ou se encurtem proporcionalmente (sem barra = *staccatissimo*; barra aflorando a nota seguinte = *legato*);
- o botão "refresh Chord" (ver fig. 8, campo ③, 1º botão) reconstrói a textura, criando um novo conjunto de frases, partindo dos actuais parâmetros (últimos intervalos, notas diferentes mais recentes, densidade e comprimento) – consoante o valor da densidade, pode dar-se o caso da instrumentação mudar por completo;
- a manipulação suave do parâmetro densidade permite adicionar ou remover à textura instrumentos um a um – a cada instrumento adicionado à textura por esta via será atribuída uma frase actualizada, o que permite uma transformação gradual da textura;
- o botão "pausa Geral" (ver fig. 8, campo ③, 2º botão) limpa todos os visores de notação (todos os instrumentistas param de tocar) até um novo conjunto de frases ser gerado;
- a textura (e o campo harmónico inerente) nunca muda se não for com a intervenção do operador da interface, através de uma das formas acima descritas.

3.2 - "harmony"

Directiva de leitura: ao sinal (ataca ou corta).

Resumo dos parâmetros manipuláveis: dinâmica; densidade; botão "refresh Chord"; botão "pausa Geral"; botão "campo H" (positivo / negativo); botões "trilo", "frullato", etc.

Modo de funcionamento:

- são gerados acordes a partir das notas tocadas pelo solista (procedimento determinístico);
- a instrumentação de cada acorde é gerida por um procedimento estocástico que tem em conta o parâmetro densidade;
- cada instrumentista toca apenas uma nota (longa) de cada vez, atacando-a de forma sincronizada com os outros instrumentista, respeitando os sinais de entrada ou de corte fornecidos pelo objecto [pulsador];
- as durações são monitorizadas – as notas tocadas pelo solista que ultrapassarem 749 milissegundos de duração (≥750ms) são interpretadas como respirações da melodia e servem para desencadear um novo acorde – ao contrário do que acontece no algoritmo "contour", aqui, não é necessária a intervenção do operador da interface para haver uma mudança de campo harmónico;
- a duração de um silêncio é somada à da nota que o antecede – desta forma, uma nota de 700ms de duração seguida se um silêncio igual ou superior a 50ms produz o mesmo efeito do caso anterior;
- o tempo de pulsação utilizado pelo [pulsador] (sinal de entrada ou de corte) é de 800ms – o que leva a concluir que o novo acorde será atacado 1550ms (800+750) após o início da nota que o desencadeia;

- se for escolhido o campo harmónico positivo (ver fig. 8, campo ③, 3º botão), o acorde é composto com as 5 notas diferentes mais recentemente tocadas;
- se for escolhido o campo harmónico negativo, o acorde é composto com as 4 notas diferentes que há mais tempo não são tocadas, às quais se junta a última a ser tocada;
- se a densidade for superior a 5 instrumentos, haverá repetição de notas (salvo diferença de registo);
- se a densidade for inferior a 5, seguramente, uma das notas será a última que o solista tocou – mesmo para o caso do campo harmónico negativo – sendo a(s) restante(s) escolhida(s) aleatoriamente de entre as 4 ainda disponíveis;
- o registo de cada nota é escolhido a partir de um âmbito adequado a cada instrumento, através de um procedimento estocástico;
- o botão "refresh Chord" reconstrói o acorde, tendo em conta os actuais parâmetros (notas diferentes mais/menos recentes e densidade) – consoante o valor da densidade, pode dar-se o caso da instrumentação do acorde mudar por completo;
- o referido botão pode ser usado de forma recursiva, gerando grupos de acordes com durações mais curtas do que a norma, neste algoritmo – durações essas que nunca serão inferiores a 800ms, tendo em conta o tempo de pulsação usado no [pulsador];
- o botão "pausa Geral" limpa todos os visores de notação (todos os instrumentistas param de tocar, ao sinal de corte do [pulsador]) até novo acorde ser gerado;
- o uso de efeitos, tais como trilo, frullato ou outros, pode ser muito eficaz com este algoritmo – existem botões e atalhos de teclado (ver fig. 8, campo ④) que permitem o envio de mensagens predefinidas com indicações para alguns desses efeitos; alternativamente, a mensagem pode ser digitada directamente na caixa de texto respectiva.

3.3 - "klangfarben"

Directiva de leitura: cânone.

Resumo dos parâmetros manipuláveis: articulação (*x*duração).

Modo de funcionamento:

- como o nome sugere, este algoritmo baseia-se na técnica conhecida como melodia de timbres;
- funciona como um cânone à distância fixa de 1000ms;
- cada nota tocada pelo solista é atribuída a um instrumento à sorte;
- a atribuição de cada nota é feita em tempo-real, o que significa que o músico dispõe exactamente de 1000ms para a decifrar e preparar;
- são evitadas repetições de instrumento, directas ou alternadas, na atribuição de notas;
- o registo de cada nota é escolhido a partir de um âmbito adequado a cada instrumento, através de um procedimento estocástico;
- o visor de notação é usado em modo de partitura rolante (*scrolling*), onde as notas são vistas como se se deslocassem da direita para a esquerda, devendo ser articuladas no instante em que cruzam uma linha vertical fixa, de cor verde (barra de leitura), sendo que a própria nota fica verde, nesse momento;

- o [pulsador] é usado de forma síncrona para ajudar na precisão do momento do ataque – utiliza-se o mesmo tempo de pulsação que vimos anteriormente (800ms), o que significa que o arranque da pulsação se dá 200ms após a atribuição da nota;
- as durações de cada nota, salvo manipulação, são as originais;
- o parâmetro articulação é usado de forma análoga ao que acontecia no algoritmo "contour", no sentido em que a sua manipulação faz com que as barras de duração se alonguem ou se encurtem – contudo, o termo articulação não se aplica verdadeiramente, na medida em que se tratam de notas isoladas, em cada instrumento; na prática, trata-se de uma multiplicação/divisão das durações originais, criando respostas que podem ser mais *staccato* que o original ou então com sobreposição de timbres;
- as dinâmicas são as originais, sendo inscritas por debaixo de cada nota, no próprio visor de notação.

3.4 - "sombra"

Directiva de leitura: responsório.

Resumo dos parâmetros manipuláveis: não aplicável.

Modo de funcionamento:

- este algoritmo baseia-se no funcionamento do anterior, no que diz respeito à utilização do modo de partitura rolante, com a excepção de que não há linha vertical verde (barra de leitura), nem se pretende o mesmo tipo de ataque sincronizado;
- funciona como um cânone a uma distância não fixa;
- todas as notas tocadas pelo solista aparecem em todos os visores, num fluxo contínuo da direita para a esquerda;
- a resposta do cânone é feita por todos os instrumentos, em heterofonia – pretende-se que cada músico execute a resposta livremente, num tempo ligeiramente diferente do dos seus colegas, procurando criar desfasamentos;
- o registo, para cada instrumento, corresponde a uma transposição de oitava (ou múltiplo desta), criando um resultado bastante mais "tonal" que os algoritmos anteriores;
- as dinâmicas são as originais, sendo inscritas por debaixo de cada nota, no próprio visor de notação.

3.5 - "finalChord"

Directiva de leitura: acorde final.

Resumo dos parâmetros manipuláveis: dinâmica.

Modo de funcionamento:

- é gerado um acorde, construído por camadas – uma nota de cada vez;
- o acorde só estará completo quando todos os instrumentos estiverem a tocar;
- as durações das notas tocadas pelo solista (somadas às dos silêncios que eventualmente lhes seguirem) que ultrapassarem 1499 milissegundos são interpretadas como finais importantes de frase e são adicionadas ao acorde – dessa

forma, o solista tem tempo suficiente para decidir se permite que determinada nota seja usada na construção do acorde ou se avança para mais uma frase melódica;

- o registo de cada nota é escolhido a partir de um âmbito adequado a cada instrumento, através de um procedimento estocástico;
- o operador da interface não tem forma de intervir na construção do acorde;
- a dinâmica pode ser manipulada;
- como o nome indica, este algoritmo foi pensado para finalizar uma performance.

4. APRECIAÇÕES PRÁTICAS E ESTÉTICAS

A primeira fase de desenvolvimento do presente sistema coincidiu com a preparação da performance "Comprovisação nº1". Os ensaios serviram, na maior parte das vezes, para testar o desempenho dos diferentes módulos do sistema, à medida que iam sendo criados ou melhorados. Musicalmente, foi possível ter uma percepção dos tipos de interacção solista-máquina-(operador)-intérpretes-solista (em ciclo) que poderiam resultar de maneira mais interessante. Com base nessa percepção, alinhavámos algumas estratégias para a estruturação formal da performance.

A apreciação global que fazemos, após o concerto, é que os objectivos prioritários foram alcançados, na medida em que concebemos um sistema que funciona de maneira estável e previsível, e é capaz de, em tempo-real, partindo de um discurso musical improvisado, criar uma resposta musical organizada, coerente, e passível de ser interpretada, no momento, por um grupo de músicos instrumentistas.

Simultaneamente, damo-nos conta de que seria interessante implementar, no curto prazo, algumas soluções ao nível da utilização do ritmo e da gestão algorítmica da instrumentação. Igualmente, estão identificados obstáculos de nível prático que, uma vez resolvidos, permitirão um funcionamento mais eficaz e uma utilização mais gratificante do sistema.

Na tabela que se segue, encontram-se descritos os recursos ou funcionalidades que, na sequência das apreciações referidas, já lográmos implementar ou pretendemos implementar a curto ou médio prazo.

recurso ou funcionalidade	modo de implementação	**Ø** - obstáculo visado / **√** - ganho pretendido / **!** - problemas antecipados	estado / previsão
Auto-programação do módulo "Porta de Comunicação".	Instanciação dinâmica (através de *script* - objecto [thispatcher]).	**Ø** Inflexibilidade ao nível da instrumentação, por ser fixa, pré-programada; **√** Possibilidade de redefinir a instrumentação de uma performance ou de um ensaio com uma antecedência de poucos minutos, através da edição de uma tabela.	implementado
Automatização do registo dos endereços IP dos computadores cliente junto da porta de comunicação.	Comunicação via UDP no sentido cliente-anfitrião com identificação: instrumento+IP.	**Ø** Tempo gasto no entediante processo de digitação manual dos endereços IP de todos os computadores cliente, em cada sessão; **√** Obtenção da plena configuração do sistema de comunicação, de forma automática, com uma intervenção mínima dos instrumentistas; **√** Avaliação do desempenho do sistema através do envio e retorno de mensagens *ping*.	em fase avançada de implementação
Registo dos dados de notação musical produzidos durante uma sessão.	Captura de todas as comunicações de saída (objecto[f0.data _seq]).	**Ø** Impossibilidade de analisar, em tempo-diferido, a composição gerada e o funcionamento dos algoritmos; **√** Reconstrução da composição gerada, para efeitos de análise; **√** Possibilidade de montagem video.	em fase avançada de implementação
Partitura de retorno.	Re-encaminhamento dos dados de notação para um objecto [bach.roll] auto-configurado a partir da instrumentação definida.	**Ø** Impossibilidade de visualizar, em tempo-real, a totalidade das partes instrumentais; **√** Retorno visual de todas as partes geradas algoritmicamente, para efeitos de monitorização do processo; **√** Facilidade em realizar ajustes na programação dos algoritmos, através da análise do retorno visual geral; **!** Será conveniente fazer uma avaliação do consumo de recursos de CPU.	em fase inicial de implementação
Novo desenho da interface de leitura – 1.	Substituição dos dois visores lado a lado por um só visor para ambos os instrumentos (sistema de pentagramas).	**Ø** Constrangimento em termos de espaço horizontal de leitura; **√** Optimização do espaço horizontal de leitura através da distribuição das notas, pelo espaço, evitando a aglomeração das mesmas, em casos de gestos rápidos e/ou com muitas notas; **!** Será necessário modificar partes da programação em todos os algoritmos.	a implementar a curto-prazo
Novo desenho da interface de leitura – 2.	Substituição do modo de partitura rolante (*scroll*) pelo modo de partitura em *loop* (notas fixas; barra de leitura móvel; actualização em *loop* do conteúdo).	**Ø** Constrangimento em termos de desempenho gráfico, que se traduz em constrangimento na velocidade do fluxo de notas, que por sua vez limita o uso de durações curtas (as notas ficam aglomeradas); **√** Melhoria na performance do CPU e no desempenho gráfico; **√** Melhoria na legibilidade; **√** Possibilidade de uso de durações mais curtas. **!** Será necessário re-programar pequenas partes nos algoritmos que usam este modo.	a implementar a curto-prazo

recurso ou funcionalidade	modo de implementação	**Ø** - obstáculo visado / **√** - ganho pretendido / **!** - problemas antecipados	estado / previsão
Utilização de ritmo no algoritmo "contour".	Reprogramação do algoritmo de modo a aproximar e/ou afastar horizontalmente grupos de notas.	**Ø** Monotonia rítmica na repetição cíclica das frases melódicas geradas; **√** Obtenção, numa dada frase, de notas agrupadas e outras afastadas, por forma a serem interpretadas com proporções rítmicas distintas; **√** Possibilidade de manipular a proporção gráfica entre notas aproximadas (agrupadas) e afastadas;	a implementar a curto-prazo
Gestor de instrumentação.	Inserção de um bloco na interface de controlo, que fará uso do objecto [nodes]; Programação do algoritmo de gestão.	**Ø** Relativa monotonia tímbrica devido ao uso repetido do mesmo procedimento de gestão; **√** Diferentes pesos de probabilidade para grupos (famílias) de instrumentos; **√** Possibilidade de manipular os referidos pesos via objecto [nodes] ou através de um *joystick* externo;	a implementar a médio-prazo
Disparo em tempo-real, por meio de eventos previamente determinados, de estruturas musicais pré-compostas.	Composição baseada em registos de performances anteriores, utilizando cadeias de Markov de 2ª ou 3ª ordem; Programação de algoritmo dedicado ao disparo em tempo-real; Instanciação, na aplicação cliente, do objecto [bach.score], que permite a notação rítmica convencional.	**Ø** Dificuldade em conceptualizar um algoritmo que produza, em tempo-real, um resultado de alto nível de complexidade rítmica e polifónica mas que seja passível de ser interpretado sem falhas, por músicos humanos, num contexto de leitura à primeira vista; **√** Incorporação, em tempo-real, de estruturas musicais de alto nível de complexidade rítmica e polifónica previamente estudadas pelos intérpretes; **√** Proximidade estética face ao conceito de base do projecto, via 3 vertentes: 1. composição algorítmica (utilização de cadeias de Markov), 2. composição a partir de uma improvisação (ainda que em tempo diferido), 3. notação dinâmica em tempo-real (ainda que a música esteja previamente composta e estudada).	a implementar a médio-prazo

recurso ou funcionalidade	modo de implementação	**Ø** - obstáculo visado / **√** - ganho pretendido / **!** - problemas antecipados	estado / previsão
Geração, em tempo-real, de estruturas musicais baseadas na análise estatística da performance em curso.	Analise estatística da performance, recorrendo a cadeias de Markov de 2ª ou 3ª ordem; Programação de algoritmo dedicado à análise estatística e à geração das estruturas musicais.	**Ø** Inexistência de procedimentos generativos que sejam capazes de, a partir de um dado momento, continuar a compor sem intervenção directa solista; **√** Riqueza formal, graças à possibilidade de ter momentos em que o solista não intervém; **√** Riqueza formal, graças à possibilidade de ter uma resposta musical que reflecte um estímulo que poderá ter ocorrido muito antes; **!** Será conveniente salvaguardar a dificuldade de leitura e/ou de execução da música gerada desta forma.	a implementar a médio-prazo

Tabela 1 - Comprovisador: recursos e funcionalidades a implementar, a curto e médio prazo.

Resta dizer que, para além das melhorias que descrevemos na tabela 1, temos algumas ambições estéticas que tentaremos alcançar a longo prazo, nomeadamente, a possibilidade de:

- ter vários algoritmos em funcionamento simultâneo, direccionados a grupos instrumentais distintos (orquestração em tempo-real);
- construir campos harmónicos a partir de soma/multiplicação de intervalos, por oposição ao actual sistema que se baseia na equivalência de 8a;
- testar/adaptar o sistema para funcionamento com:
    - formações orquestrais (implica grande quantidade de computadores cliente);
    - instrumentos com outras características (harmónicos/rítmicos);
    - vários solistas na mesma performance.

C

Synchronising to Visual Cues

This chapter draws almost entirely on the following publication: [Lou19c], with some parts omitted to avoid repetition.

C.1 Assessing the Effectiveness of the Bouncing Ball

For a coordinated musical gesture to be perceived as such, some kind of synchronisation strategy must be at play. In a traditional context of ensemble playing, musicians simply react to each other's movements and/or those of a conductor – and this is possible due to the physical proximity in which these interactions occur. In

certain cases, proximity may allow a musician to effectively react to sounds produced by another musician, be it breath sounds, attack transients or dynamic nuances.

Evidently, this strategy becomes useless when musicians are tens of meters apart or when their vision is blocked by columns or even walls, as is often the case in performances carried out with `Comprovisador`. The use of a graphical synchronisation strategy – a visual cue – may address this problem but, as we shall discuss, there are many issues to consider if we aim at a musically satisfying solution. These issues are intrinsic to the networked music performance paradigm (e. g.: network latency, acoustic delay (speed of sound), lack of eye contact) and the real-time notation one (e. g.: timing discrepancies induced during the individual process of sight-reading, lack of experience with graphical cues, hesitation).

`Comprovisador` uses a bouncing ball approach as synchronisation strategy. Compared to other common strategies (e. g.: scrolling score, linear cursor, etc.) It has the advantage of using downward vertical acceleration and ricochet to convey a sense of pulse, of arsis and thesis, simulating the motion of a conductor's hand. At the same time, by bouncing towards the appropriate note or bar-line, it enables score navigation.

In our previous research ([Lou17b]), we have found poor graphics performance on slower machines to have a negative impact on synchronisation – in cases where the frame rate dropped below a certain value, musicians reported they were no longer able to perceive the bouncing motion and, thus, the moment of attack. In the first two versions of our bouncing ball implementation (based on `Max`'s [lcd] and [jsui] objects, respectively), graphics were rendered in the CPU and this fact did not allow for a better performance. After switching to OpenGL, a technology that takes advantage of hardware acceleration by rendering in the graphics card, we were able to reverse those impacts, easily achieving rates of 60 frames per second, while enhancing other aspects of the interface regarding information detectability and legibility (dynamics and textual instructions). This improvement was assessed by musicians who had rehearsed and performed in "Comprovisação no 5" (using the [jsui] implementation) and continued collaborating in the project [Lou17d]. Musicians who performed in the subsequent three "Comprovisações" also reported being able to synchronise to the bouncing ball without effort.

Although musicians' reports were encouraging, we felt there was still room for improvement, namely regarding the issue of network latency – which we will explain bellow. Also, we devised an experiment in order to measure the effectiveness of our synchronisation strategy, which will be subject for discussion herein.

C.1.1 Reading Time Window

Establishing a reading time window and implementing a visual cueing device (the bouncing ball) were our first design choices (see Section 3.2.2), within algorithm Harmony. The reading time window (r.t.w.) is adjustable according to musical goal and/or technical difficulty. The period of the bouncing ball must be less than or equal to the length of that window. Also, it should not be greater than 1 second as the bouncing motion would become too slow to be effective. To give a practical example, if the reading time window is set to 1600 milliseconds and the bounce period (b.p.) is set to 1 second, from the moment a given note is displayed there is a pre-bounce time of 600 ms followed by a bounce time of 1000 ms (see Figure C.1, note E). Although the interval between the display of a note and it's actual onset is always the same (r.t.w.), in many cases, previous notes are still displayed on the screen and have to be played during that window of time; likewise, new notes might be appearing meanwhile. In these cases, pre-bounce time and bounce period change dynamically, depending on the inter-onset intervals (i.o.i.): if i.o.i. is less than r.t.w. but greater than b.p., then pre-bounce time is shortened; if i.o.i. is less than b.p., then pre-bounce time equals zero and bounce period is shortened.

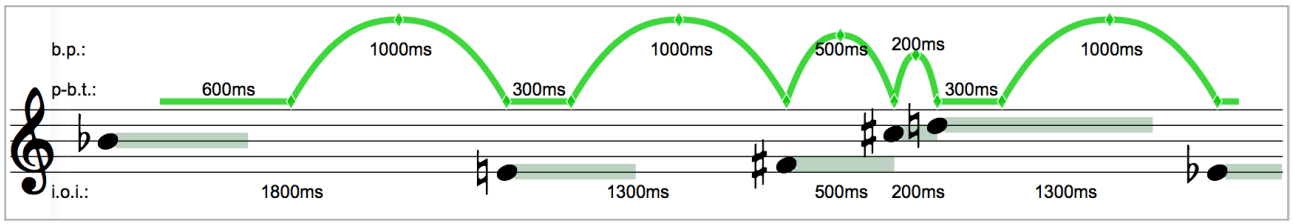


Figure C.1: Bouncing ball path – bounce period (b.p.), pre-bounce-time (p-b.t.) and inter-onset interval (i.o.i). Note: although all notes are visible in the image, individual notes are displayed in left-to-right sequence, according to their inter-onset intervals; assuming a reading time window of 1600ms, each note is displayed 1600ms before it should be played (i.e. ahead of the bouncing ball's current horizontal position by a horizontal distance representing 1600ms).

C.1.2 Latency Compensation

Latency is one of the most frequent problems in systems for networked music performance. If we consider performances over the internet with musicians connecting from different continents, latency may reach average values of over 100ms, which render rhythmic interplay a true conundrum. This fact alone has major implications on the aesthetics of a performance.

In local area networks, average latency values tend to be much smaller, especially when using cabled ethernet. Nevertheless, it is not possible to predict its exact value – and more so with wireless connections. This uncertainty causes problems.

From the beginning, we opted for wireless connections because they are easier to set up and do not involve extra cost in network cables. Soon, we discovered that although average latency values would be below 20ms, it was not uncommon to have messages arriving up to 200ms later.

In algorithm Harmony, as soon as a chord is generated in the host application, data representing each note are immediately sent to clients to be rendered and displayed (and to be played after the length of r.t.w.). If we could assume that latency values would always be below 20ms across different machines within the network, there would not be a real need to manage it because it is within an acceptable threshold for this kind of network applications (see [LK04]). An onset threshold of 50ms is usually acceptable in the context of ensemble music (see [Ras79]) and, although keeping in mind musicians would not be able to play perfectly in sync with the bouncing ball, there would still be a margin of at least 30ms.

In this scenario, a latency compensation approach based on the average latency value for each client could potentially increase this margin. But if very occasionally a given note on a given client arrives 200ms later than other notes on other clients (i.e., with a delay 10 times greater than the average), we would have an undesired discrepancy despite the implemented latency compensation. Thus, this approach would not solve the problem of occasional outliers.

Thus, we have looked into Dannenberg's concept of "Time Flow" [Dan17] which aims at solving synchronisation problems in real-time music and media systems. The author describes four approaches to synchronisation in increasing levels of sophistication: Synchronisation Levels 0 through 3. Level 0 approach "is simply to eliminate as much latency as possible, assume that latency is zero, and operate in real time" [BD99] – so, our previous approach relates to this level.

While the higher sophistication levels of synchronisation are conceptualized for more complex systems involving different kinds of media processing and timing accuracy needs, level 1 is the most relevant for addressing the differences in communication time we were experiencing. It consists on applying time-stamps to events, computing the events in advance within a "control stage" and delivering the computed events with time-stamps to a "rendering stage". There, events are delayed according to time-stamps in order to produce accurately timed output. Since it presupposes a delay in the rendering stage and our algorithm already has a delay in that stage (the reading time window), it was possible to implement this approach. But for time-stamps to have actual meaning within a network, all machines must agree on the time. This issue can be addressed with the use of a clock synchronisation protocol such as the one described by Brandt and Dannenberg [BD99]. We have implemented an adaptation of their approach which was then tested in the experiment described in

Section C.1.3.

Our implementation consists on setting a master clock on the host and a slave clock on each client. The slave clock continuously adjusts itself to match the master clock, which is done in timed intervals, via clock messages. In between those clock messages, the slave clock tries to predict when the next message will occur. Median latency values (instead of arithmetic mean) are used for latency compensation and a prediction threshold is used in order to detect outliers: messages arriving outside the threshold are ignored.

C.1.3 Method

In order to assess the bouncing ball's effectiveness as a synchronisation strategy, the following experiment was devised: four musicians were placed in two separate rooms in such a way that none had visual contact with their neighbor (see Figure C.2). Each musician had a computer running a client of `Comprovisador`. Another computer was used to run the host application of `Comprovisador` and to record audio. Participants were professional musicians – a saxophonist, a trombonist, a guitarist and a pianist, and they had no prior experience with real-time notation.

Musicians had to sight-read dynamic notation displayed in their computer screens. Staff-based proportional notation (without meter or traditional duration symbols) was used (see Figure 3.7b). The notes were generated in the host computer – by `Comprovisador`'s algorithm Harmony – and were to be played homophonically as chords[1].

Without reference of any kind other than the bouncing ball, musicians had to rely solely on this graphical cue in order to know when to play a note. In this manner, if musicians were able to play in a fairly synchronised way, our experiment would determine that the bouncing ball approach was suitable for its goal.

In order to measure the onset thresholds of each chord and, at the same time, to assess the improvements enabled by the "time flow" implementation, sound was recorded simultaneously in 12 tracks: four tracks recorded the host computer's own rendering of the score (internally/ digitally); another four tracks recorded the output of each client computer (mini-jack out); and the remaining four tracks recorded the live acoustic instruments (see Figure C.3). Each instrument had its own microphone, each client computer had a cable connecting from its headphones output, and all 8 cables were connected to the host computer's audio interface, so everything was recorded in sync in order to be measurable through waveform analysis.

[1] In some cases, piano and guitar had actual chords of up to four notes to play.

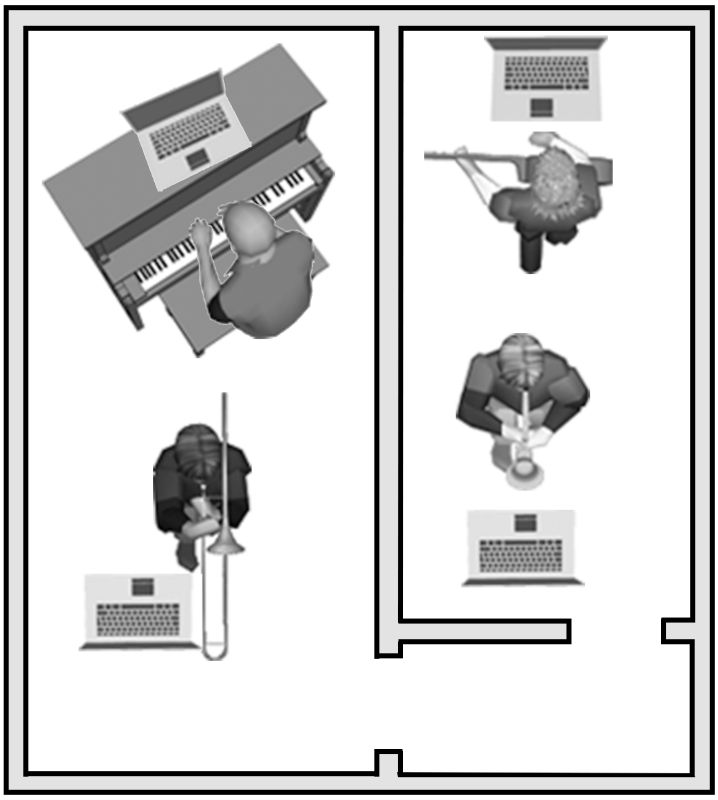


Figure C.2: Recording session – separate rooms, no eye contact.

C.1.4 Results

In Figure C.3, the four waveform tracks in the middle stack correspond to the output of each client computer. It is visible that all onsets fall within a 20ms window – and this is typical throughout the whole recording. Thus, we can assume our "time flow" implementation works as expected, preventing occasional longer communication delays to cause onset discrepancies.

Nonetheless, although 20ms might be considered an adequate value for this application, it is not as good as the results obtained by Brandt and Dannenberg (1999). Also, the bottom stack (internal host rendering of the score) shows discrepancies of the same order of magnitude. This suggests variable latency is being induced at an earlier stage – during the chord generation or, more probably, during the polyphony constraint solving (used for piano and guitar chords).

Comparing the bottom and middle stacks, we see a delay of around 70ms. This could be owing to audio processing latency induced by the clients' sound cards (digital-to-analogue conversion) and the host's audio interface (analogue-to-digital conversion), since the audio of the bottom stack was internally recorded without

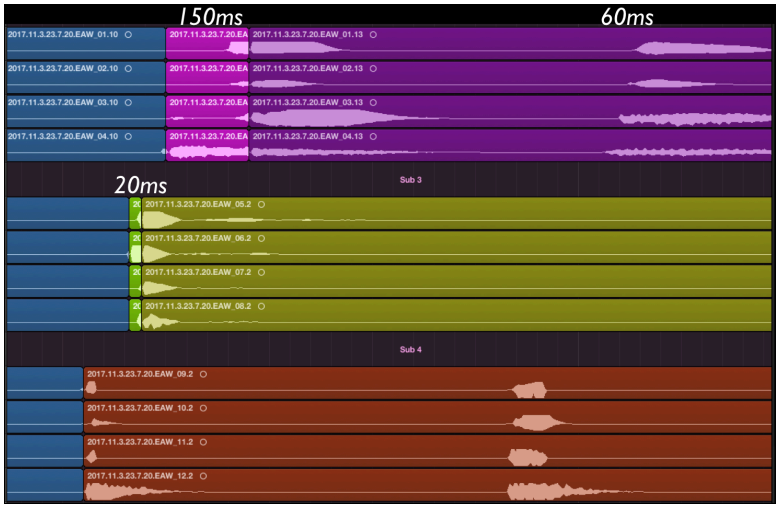


Figure C.3: Waveform analysis – bottom stack: host computer's rendering of the score; middle stack: output of each client computer; top stack: live acoustic instruments.

any D-A or A-D conversion (hence, no latency). Furthermore, all client computers had different specifications which could also explain the 20ms onset span observed in the middle stack, considering latency induced at the audio processing stage.

Looking into the top stack (acoustic instruments), we can see on this example an onset span of around 150ms. But if we were to ignore the piano (the fourth track), we would have all onsets within a 30ms window. In the following chord, all onsets were evenly distributed along a 60ms window (with intervals of around 20ms between onsets).

These examples were selected from a section of the recording where the sight-reading context was not very difficult. During this section we observed typical onset spans of 100ms or less, averaging to around 80ms. In more difficult contexts (for example, with a fast note rate or with unexpected melodic leaps), average onset spans tended to be larger.

C.1.5 Conclusions and Future Work

Considering the outcome of our experiment, we find that the bouncing ball approach can be safely used in sight-reading contexts of moderate difficulty yielding synchronisation results that we judge as musically acceptable. If we take into account the fact that participants had no prior experience with real-time notation, we think it is safe to assume these results can be improved through practice. It would be interesting to repeat this experiment in similar conditions and with the same musicians after a practice period in order to compare results.

Regarding factors that create difficulty in sight-reading, and although this experiment was not designed to assess this issue in particular, we were able to observe four difficulty vectors: note rate, melodic leaps, polyphony, and reading-time window.

Reading-time window is easy to assess: the musician needs time to read ahead to be able to decipher a note and prepare it in their instrument. Values under 1 second are insufficient; values over 2 seconds become unnecessarily large. From trial and error, we have found the sweet spot to be around 1600ms.

Polyphony should also be easy to determine: a single note is always easier to decipher than a pair of notes, which is easier than three notes, etc.

Regarding melodic intervals, unisons are arguably the easiest to sight-read; large leaps are generally considered more difficult than small steps, but there are exceptions (e.g.: a restricted pitch set might facilitate sight-reading large leaps, thanks to pitch recurrence).

We find note rate harder to assess. On one hand, if we consider an isochronous stream of notes, then, for a given musician, there should be a given threshold beyond which he or she will start feeling stressed. In this case, we can say that difficulty is proportional to note rate. On the other hand, with small groups of two to four notes followed by rests, we can achieve smaller inter- onset intervals than in the previous case. With larger groups of notes, difficulty levels will rise quickly. Hence, note rate should be analyzed as a two-dimensional vector where we cannot find a single threshold point but a threshold curve instead.

In order to determine thresholds for these vectors – or combinations of vectors – another experiment would have to be designed, using a similar method but organizing algorithmic parameters in a linear way. Then, it would be possible to collect sufficient data to perform a thorough analysis that could yield more conclusive results.

Finally, taking into account the main results of this research, since we were able to observe that the average onset span was fairly stable for easy-to-moderate sight-reading contexts (around 80ms) tending to ramp up in more difficult scenarios, it would make sense to use this parameter to signal the threshold for a given difficulty vector.

D

Distributed Scores and Audio

This is the integral text of the following publication: [Lou21]. This publication introduced an animated score system, wherein the measures are dynamically updated in a cycle so as avoid any disruptions in the reading process. This system forms the foundation for the improvements being devised for Comprovisador, as detailed in Chapter 5.

Distributed Scores and Audio on Mobile Devices in the Music for a Multidisciplinary Performance

Abstract: In an attempt to uncover the strengths and limitations of web technologies for sound and music notation applications, driven by aesthetic goals and prompted by the lack of logistic means, the author has

developed a system for animated scores and sound diffusion using browser-enabled mobile devices, controlled by a host computer running `Max` and a web server. Ease of deployment was seen as a desirable feature in comparison to native application computer-based systems – such as `Comprovisador`, a system which has lent many features to the one proposed herein. Weaknesses were identified motivating the design of mitigation and adaptation strategies at the technical and the compositional levels, respectively. The creation of music for a multidisciplinary performance entitled `GarB'urlesco` has served as a case study to assess the effectiveness of those strategies. The present text is an extended version of a paper presented at CMMR 2019, in Marseille.

D.1 Introduction

Recently, we were invited to compose music for a multidisciplinary performance entitled `GarB'urlesco`, which includes elements of theatre, dance, costume design and music (Fig. D.1) [Lou19a]. In addition to the new music created for this purpose, the performance also includes pieces of baroque music, played on period instruments. The narrative explores the sociocultural contrasts that can be observed in the Algarve. Algarve is a southern Portuguese region marked by densely populated coastal areas and scattered rural communities, and with an economy driven by a very seasonal tourist activity. To enhance the idea of contrast in the composition, it seemed aesthetically relevant to include electro-acoustic elements clashing with the period instruments. However, there were no logistical or financial resources for this, especially regarding sound projection. Not allowing ourselves to give up, we looked for alternatives in recent examples in the field of NMP where mobile devices are used as musical instruments, sound projectors, animated scores or as an interface for audience participation.

Regarding our previous experience, as of 2015 we have been developing `Comprovisador`, a system designed to enable mediated soloist-ensemble interaction using machine listening, algorithmic compositional procedures and dynamic notation, in a networked environment [Lou17e, Lou18c, Lou17d]. In real-time, as a soloist improvises, `Comprovisador`'s algorithms produce a score that is immediately sight-read by an ensemble of musicians, creating a coordinated response to the improvisation. To this date, `Comprovisador` has been used in ten public performances. In each performance, the system is presented with new features. The introduction of composed electro-acoustic tracks in synchronisation with the animated scores is to be expected soon.

Like any other tool, `Comprovisador` has its strengths and its weaknesses. Weaknesses are often problems for which a solution has not yet been found or represent a trade-off from other aspects considered more important. One such case is the fact that `Comprovisador` is not compatible with mobile devices (namely, tablets). Not only

Figure D.1: **GarB'urlesco** – a multidisciplinary performance combining theatre, dance, costume design and music (early and contemporary), played on period instruments.

does it require proper computers but it also requires software installation procedures[1] that are not very complex for someone experienced, but can be so for someone who is not computer oriented. Thus, it can be tedious or discouraging for users and it always entails increased preparation time when preparing a concert. These issues would be problematic in the context of **GarB'urlesco** and would not solve the fundamental issue of sound projection logistics.

For **Comprovisador**'s objectives, processing speed and reliability of laptop computers running native applications outweigh ease of deployment of tablets running web applications. Other applications with different objectives may benefit more from a mobile device-based approach. Apart from musical performance with real-time generated scores, we envisage possibilities in the educational field – Soundslice [HRO19], a web platform where users can learn to play pieces of music through dynamic notation synchronised with Youtube videos is a good example, but there are also possibilities in the fields of ear training and sight reading (see [Lou18d]).

In the field of music and multimedia networked performance, there are many examples that use mobile devices with as many different strategies. Following are only a few examples.

"Flock" [Fre08, Fre07], a piece by Jason Freeman for saxophone quartet, video, electronic sound, dancers, and

[1] Both host and client applications of **Comprovisador** run in the **Max** environment [PZS+] using the **bach** library [AG15, GA17] and also Java.

audience participation premiered in 2007, uses PDA's mounted on each player's instrument. Those devices display music notation generated from the locations of musicians, dancers, and audience members as they move and interact with each other.

Decibel ScorePlayer [HVWJ15] is an iPad application developed by the Decibel New Music Ensemble, a group led by composer Cat Hope. This application enables network-synchronised scrolling of proportional colour music scores and audio playback. It has been used by many composers worldwide.

Cheng Lee proposes an approach for incorporating computer music and virtual reality practices into a multi-media performance installation requiring the audience members to use their own smartphones as 360-degree viewing devices [Lee17]. The author also proposes the use of wireless speakers carried around the venue as a means of achieving immersive sound and music effects in substitution for a multi-channel surround-sound system.

Composer Jonathan Bell, who also has used bluetooth speakers as networked sound projectors, has created a system called SmartVox [Bel18] – a web-based distributed media player as notation tool for choral practices. The audio-visual scores are created with the **bach** environment for **Max**. In a choral context, singers hear (using earphones) and see their own part displayed in the browser of their smartphone. The whole is synchronised through the distributed state of the web application.

In May 2019, a networked music performance dubbed Symphony for a Tunnel for 144 musicians and distributed score display system took place in Hamburg's St. Pauli Elbe Tunnel [HG19]. The performance featured Drawsocket [GH19], a recently developed system which was able to draw and synchronise scores across the required 144 iPads, connected via Wi-Fi. The system provides control over diverse media features of web browsers, notably SVG[2] which can be used to draw animated graphic notation. Through integration with MaxScore[3] [HG19], the system also enables common-practice notation. On the server side, Drawsocket uses **Max** as its primary controller interface and Node.js (Node for **Max** (N4M)) for server-client communication. Of the examples given here, only SmartVox and Drawsocket use purely web technologies on the client side.

Finally, we should mention a.bel [CRRP16] – a system presented in 2015 at Casa da Música, in Porto, Portugal, in a concert where almost 1000 smartphones were used as musical instruments by the members of the audience. The event featured pieces by four composers – Carlos Guedes, José Alberto Gomes, Neil Leonard and Rui Penha – using different approaches to audience participation via their smartphones and the a.bel system.

[2] Scalable Vector Graphics (SVG) is an Extensible Markup Language (XML)-based vector image format for two-dimensional graphics with support for interactivity and animation.

[3] MaxScore is a music notation library for the **Max** environment [com].

In this paper, based on a case study – the music composed for `GarB'urlesco`, we will attempt to demonstrate a possible application for web resources (HTML5, JavaScript (JS) and open-source libraries) which includes animated precomposed scores and distributed sound diffusion on a local network. Since synchronisation is not easily achievable in this context, we will discuss certain mitigation and adaptation strategies that were adopted, regarding the technical and the compositional sides, respectively. The compositional strategies also took into account the sound characteristics of this type of devices.

The motivation for carrying out this practical application is therefore related to research on easily deployable solutions and sharpened by the will to perform electro-acoustic music in a context with scarce logistical resources.

D.2 HTML5-based Solutions

General advantages of mobile devices for animated score and sound applications include ubiquity (most especially in the case of smartphones), ease of transportation and set-up (from one's pocket directly to the music stand), and a fair processing power despite being small and lightweight. As for the advantages of web-based software we count being completely cross-platform and cross-type (laptops, tablets, smartphones, etc.), having no need for additional software installation (any required libraries are loaded by the browser at runtime), being free of charge, capable of performing animation at an adequate frame rate (HTML5 Canvas element / `window.requestAnimationFrame()` method, which we find to have a superior performance than `Max`'s `jsui` or `lcd` options (cf. [Lou18c])), and including access to powerful open-source tools for graphics and sound (p5.js, p5.sound [p5j], tone.js [ton]) and for interfacing with `Max` (Miraweb and its underlying library – xebra.js [xeb18]).

This approach also has disadvantages, specifically the inability to use UDP[4] and the unavailability to the best of our knowledge of an open-source[5] music notation library with suitable quality for generative applications[6]. Furthermore, there are important timing issues with three different origins: 1) network latency, which is aggravated by the use of the Transmission Control Protocol (TCP) (since UDP as an alternative is unavailable); 2) imprecise JavaScript clock (accessing the audio subsystem's hardware clock through the Web Audio API can

[4] UDP (User Datagram Protocol) is a communications protocol used primarily for establishing low-latency and loss-tolerating connections between network applications.

[5] Paid solutions are not included in the scope of this research.

[6] At the cmmr 2019 conference, we have learnt about the Guido Project [gui20], an open source project that encompasses a music notation format, a score rendering engine and various music score utilities. The Guido engine is a library that can be embedded on different platforms using different programming languages, among which JavaScript. We find both the music notation format and the score rendering engine to be adequate for the type of generative applications we envisage.

improve precision although not to our desired levels, as will be discussed in Section D.3.1); and 3) latency originating in other factors.

Considering the above disadvantages, can we nonetheless accomplish interesting musical results by implementing mitigation and adaptation strategies? This question is what we attempt to answer in the following sections.

D.3 Case Study

`GarB'urlesco` is a multidisciplinary performance with elements of theatre, dance, costume design and music (both early music and purposely composed music), using period instruments and a traditional cane flute, composed electronics for networked mobile devices and animated score. The score uses standard notation which is dynamically updated also featuring a bouncing ball cueing system for synchronisation between instrumentalists and electronics.

Regarding space, the devices are arranged according to Fig. D.2 (left). Smartphones are hidden under the audience's chairs, which are arranged in double rows on each side of the room. Device number 1 (a tablet) is the only device in charge of displaying the score and it sits on the flautist's music stand.

The intent of hiding the devices in close proximity to the audience is to create immersive sound effects while causing some strangeness (hearing sound but not being able to see its source) and to compensate for the weak sound output of these devices.

In addition, there is a host computer running `Max` that coordinates sound and score events, being also connected to a small 2.1 sound system, located in the musicians' space, enabling low frequency sound effects.

We have designed a system that uses xebra.js to establish WebSocket[7] communication between `Max` and the browser of each device. Besides `Max`, the host computer runs a server hosting two HTML files, two JavaScript libraries (xebra.js and tone.js), various audio files and the score (comprised of png files, designed in `Max` using the `bach` library [AG15] for notation and CAC features – see Section D.3.1). One of the HTML files is destined for audio playback only (see Fig. D.3) and the other for displaying the score as well (device number 1, Fig. D.4).

The loading steps occur as follows:

[7] WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection.

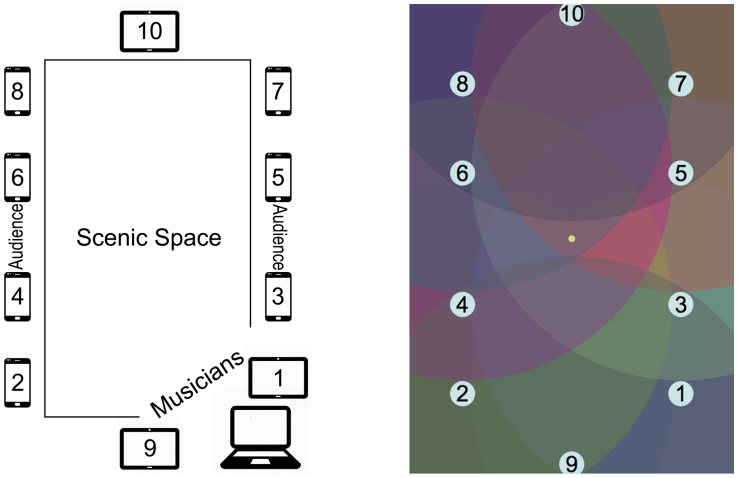


Figure D.2: Left: space arrangement of mobile devices in `GarB'urlesco`; right: Swarm Spatialiser GUI - `nodes` `Max` object.

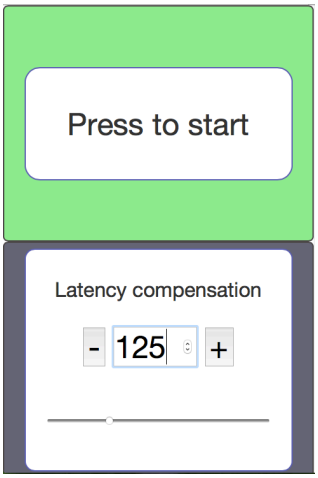


Figure D.3: HTML-based GUI for default devices – audio playback only.

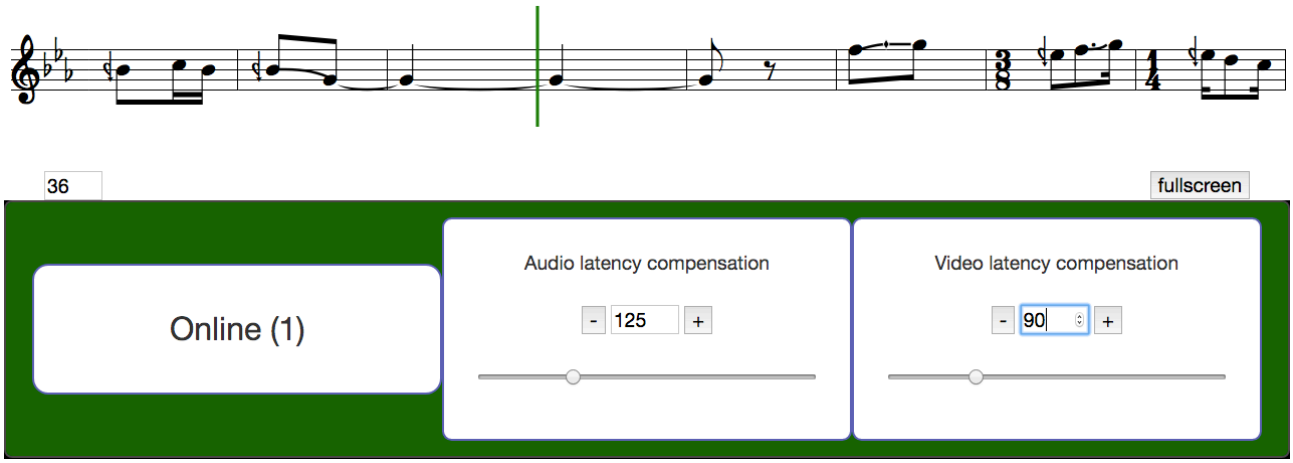


Figure D.4: HTML-based GUI for device 1 – audio playback and score display.

- the browser of each device is pointed to the appropriate HTML file;
- the user is prompted to input a unique label for easy identification by `Max`;
- the relevant audio files and, if applicable, the score png files are then loaded by the script;
- meanwhile, a WebSocket connection is established with `Max`;
- by user interaction ('press to start' button), the audio context is started up, `Max` is notified of the client's status (online) and initial clock synchronisation takes place;
- clock sync continues in background (preventing drift), during performance.

D.3.1 Technical Problems and Mitigation

Battery operated devices turn off their screens and other resources when left idle with no user interaction. In our case, it makes sense to leave the screen off as it saves up battery power, avoiding the need to plug everything to the mains, also helping conceal the devices. The problem is that other resources such as sound output and Wi-Fi connection also go to sleep or run in low consumption mode, thus hampering performance.

One possible workaround for this issue is to play a dummy sound file every few seconds, triggered from `Max` over the network. This keeps both resources awake[8]. The dummy sound file must contain some audio data

[8] – in devices running older Android versions – see Section D.4.

otherwise it is dismissed by the system. On the other hand, we obviously could not do with an audible repeating sound. Hence our sound file contains a very short (100ms), low amplitude (-21dB) and low frequency (30Hz) sine wave which cannot be heard from the devices' speakers though having enough information to trick the system into keeping both resources active.

In **Comprovisador**, we had implemented a system to attain suitable synchronisation which was based on Dannenberg's concept of "time-flow" [Dan17] – specifically, level 1 synchronisation[9] which uses time-stamps. For time-stamps to have actual meaning within a network, all machines must agree on the time. This problem can be addressed by using a clock synchronisation approach similar to the Network Time Protocol (NTP).

NTP is based on low-latency UDP. However, browsers do not allow the use of this protocol for security reasons, which raises problems for applications like ours.

After running a few tests with the WebSocket protocol we have realized that, in a local network, the round-trip time values were very unpredictable, ranging between below one millisecond and above two hundred milliseconds. Hence, we have developed an algorithm that queries the server time repeatedly, during 2 seconds. By probability, the fastest round trip within this interval will be under eighty milliseconds – although very often under one millisecond. By selecting this iteration we maximise precision and use the reported server time to make the necessary adjustments to the client's clock. Every few minutes the algorithm runs again to compensate for clock drift. The algorithm is presented below.

```
function queryNow(){    // Called repeatedly during a 2s interval
  if (querying) {
    var wrong_time = now(); // client time is assumed wrong
    var message = ["queryNow",wrong_time];
    sendMax(message);       // query Max to get server time
  }
}
function timedResponse(t){  // Called upon Max response
  var wrong_time = t[0];    // Max echoes the client time
  var reported_time = t[1]; // and reports server time
```

[9] This concept aims at solving synchronisation problems in real-time music and media systems. The author describes four approaches to synchronisation in increasing levels of sophistication: Synchronisation Levels 0 through 3. Level 1 consists on applying time-stamps to events, computing the events in advance within a "control stage" and delivering the computed events with time-stamps to a "rendering stage". There, events are delayed according to time-stamps in order to produce accurately timed output.

```
  var received_time = now(); // Client time-stamps the response
  var round_trip = received_time - wrong_time;
  if (round_trip < best_lap) {
    best_lap = round_trip;  // Store the fastest iteration
                            // calculate our offset
    ntp_off = reported_time - round_trip/2 - wrong_time;
  }                         // after 10ms, recall queryNow
  Tone.context.setTimeout(queryNow, 0.01);
}
function queryEnd() {       // Called when the 2s interval ends
  querying = false;
  Tone.Transport.seconds += ntp_off; // adjust client time
}
```

On the server side, time is obtained with the **Max cpuclock** object, which is the most accurate [PZS+]. On the client side, we obtain it with **Tone.Transport** (from the tone.js library) which is based on Web Audio's **audioContext.currentTime** property. It is indeed more accurate than the JS clock. Nevertheless, it is not as accurate as we would like it to be. A simple experiment consisting on scheduling a repeated event every 500ms with **Tone.Transport.scheduleRepeat** using various latency hint values revealed noticeable jitter in every device we have tested (Android only).

We have encountered additional latency originating in other factors at a later stage than the network and the Web Audio clock. The latency value varies from device to device, being higher on older Android devices (cf. [MOL+19]) – sometimes surpassing 400 milliseconds. In iOS devices, values were more consistent – around 100 milliseconds. Within the same device it stays relatively stable (see Section D.4).

To tackle this problem we have implemented a graphic user interface (GUI) allowing manual adjustment of the latency compensation for audio (see Fig. D.3) and also for video (score version) , which in many cases required a different value (see Fig. D.4). To perform this adjustment, one needs to be physically placed between the sound output of the server (hard-wired) and the client device; then, we trigger a repeating sharp sound on both devices and adjust the value until the resulting sound is perceptually centred. Since our approach uses time-stamps delaying the sound onset by one second in relation to triggering, it is possible to account for all latency contributions (network transfer, software and hardware processing) within a safe margin. In the future, this procedure can be automated using the server audio input to measure the client output delay and calculate

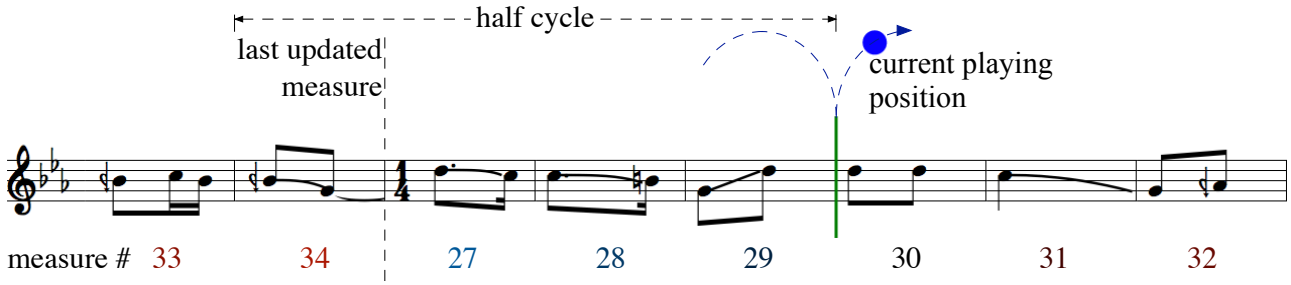


Figure D.5: Score with measures dynamically updated in cycle and bouncing ball.

the adjustment.

As for the score, since we were unable to find a suitable JS notation library, we have turned to the idea of using image files. This is possible thanks to an image-export feature included in the recently-released version 0.8.1 of **bach**[10], on which we were allowed to beta-test a pre-release.

Our approach is well suited for precomposed music and can, in the future, be modified for real-time generative scores. Here are its characteristics (see Fig. D.5):

- measures are unitary (1/4, 3/8, 3/16, 5/16, etc.);
- each measure consists of an independent png file of a constant pixel size;
- the number of measures per system depends on screen resolution and is optimized for 8 measures on the more common screens;
- measures are dynamically updated in cycle in such a way as not to disturb the reading process – replacing the ones that have been read at the distance of half a system (half a cycle)[11].

This approach was adapted from a previous one recently introduced in **Comprovisador**, the difference being that, in **Comprovisador**, notation is generated and drawn in real-time using **bach** objects.

Another approach inherited from **Comprovisador** is the bouncing ball as a visual synchronisation and score navigation device. Other authors have used similar approaches (see [Sha15, ZA18]) and consider them preferable when comparing to other types of score navigation strategies [Pic97].

[10] The **bach** library enables music notation and CAC tools inside the **Max** environment. Its most prominent objects are **bach**.score (for standard metric notation) and **bach**.roll (for proportional notation). Both objects are notation editors (meaning a user can interact with them via mouse and keyboard to create/modify the score) and score players (they can read back score data and drive a MIDI synth or similar). Moreover, they feature **Max** type inputs and outputs in order to be controlled by and/or to control other **Max** processes in real time [AG15].

[11] For a standalone interactive example of the score in action, visit **https://glitchscore.glitch.me/**.

In our web-based application, two overlaid canvas elements are used: one holds the score while the other renders the bouncing ball. The former is redrawn the least number of times possible – once only whenever the measure is updated and only in the corresponding rectangle. The latter is redrawn around 60 times per second (with JS's `requestAnimationFrame()` method). It would be possible to use only one canvas but it seemed unnecessary to redraw all the png images in every frame.

Since rendering of the bouncing ball is done by a different clock[12], a drift may occur. The correction for such drift is obtained as follows:

- `Max` host sends the message `syncScore(n, t)` every few measures, where `n` is the measure to reach at time `t`;
- if at `t` the bouncing ball is lagging behind, then jump forward to `n`;
- else if ball is early, reduce velocity by a factor in order to reach `n+1` on time `t+period` (skew).

The last step of this algorithm helps avoiding jitter, which would occur if the bouncing ball had to jump backward.

D.3.2 Compositional Problems and Adaptation

Several problems are faced when composing mixed music with mobile devices as sound projectors:

low power output – a smartphone cannot balance with an acoustic instrument;

poor quality – absence of low frequency spectrum and some degree of distortion; audible dynamic compression in iOS devices;

latency – even with our mitigation strategies, it is not possible to have the required level of timing precision to perform phase-accurate spatialisation.

Regarding the lack of power, we adapt by using number – at least one device for every four people – and textural reduction, using electronics mainly against solo instruments. On the issue of poor quality, we can restrict to

[12] Trying to render graphics with the more precise Web Audio clock would result in a much lower frame rate. Synchronisation is nonetheless controlled by the Web Audio clock using the Tone.Draw.schedule() method of the tone.js library (see [Wil13, ton]).

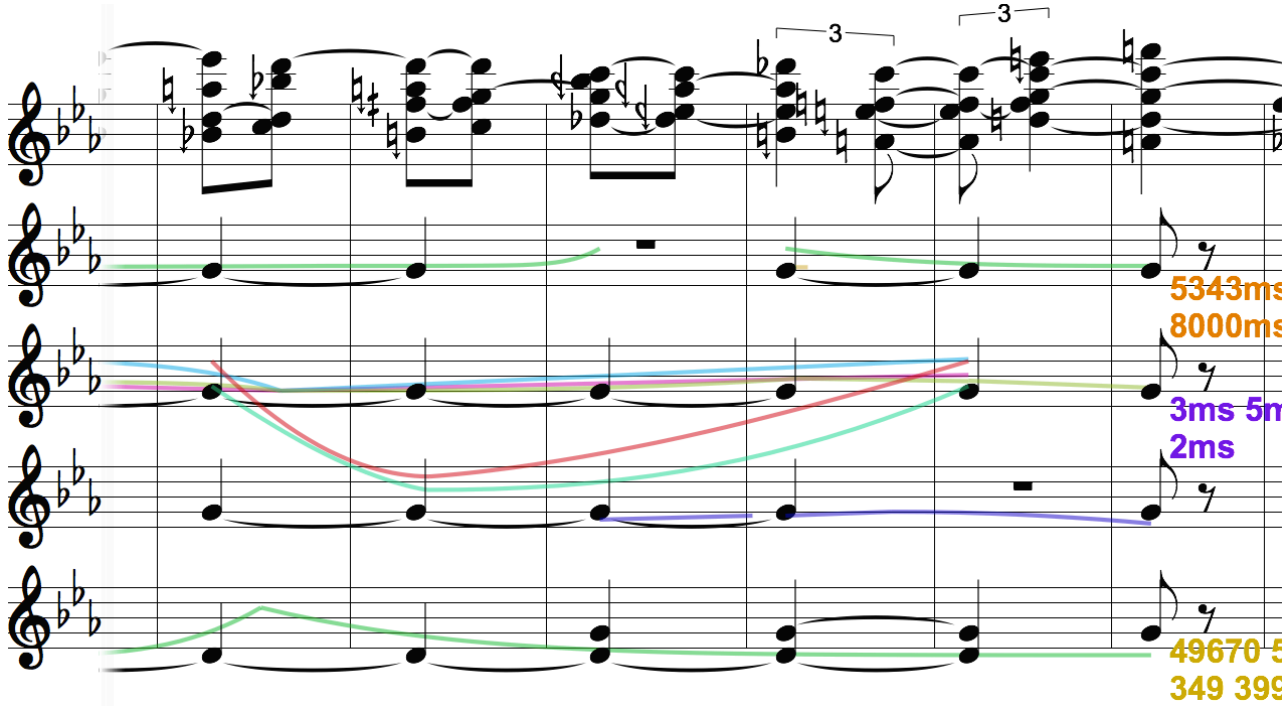


Figure D.6: A **bach**.score object featuring slots of type *function* (coloured lines) and type *intlist* (coloured numbers).

using high-pitched sounds. As for latency, "if you can't beat it, join it", which in this case means to embrace non-simultaneity – to use granulation-based effects.

In fact, we have built a granular synthesiser controlled from a **bach** score. Thanks to **bach**'s slot system, it allows sequencing all the granular synth's temporal parameters, as well as the transposition parameters (the musical notes) in the same notation environment used for the instrumental parts (see Fig. D.6).

Although phase-accurate spatialisation is not feasible, we have conceived a strategy thus allowing to convey spatial sensations encompassing temporal inaccuracies. It uses the characteristics of granulation – namely, the probability of a given grain to be emitted by a particular source.

In Fig. D.2 (right), we see a GUI made with the object **nodes**. Here, each ellipse represents the field of influence of each source while the cursor's position in relation to each ellipse represents the probability weight of a grain, at a given moment, to be emitted by the respective source.

To avoid phase problems due to temporal inaccuracies, a grain can never be emitted by more than one source simultaneously. The amplitude is based on the same weight assignment (**nodes** GUI) but with normalised and scaled values.

This system is intended to simulate, to a certain extent, a graphic system of particles where each particle revolves around a point in space, with apparent free will. Although we never see the centre point, we get a sense of where it is by the way particles express their desire towards it. The same happens in our granular spatialiser – which we name Swarm Spatialiser.

However, when we apply an automated trajectory generator to our GUI to get a stream of probability weights, and then apply further calculations of norm and scale to those values before feeding them to the granular synth, our system's CPU utilization becomes dangerously high. We have solved this by recording the stream of weights (post-calculations) to a **bach**.roll object as slot content of type llll[13]. Then, when we read back the values from **bach**.roll as a lookup table, we avoid performing all the calculations within the trajectory generator (Lissajous curves and other functions), the **nodes** GUI (translation of coordinates into probability weights and graphic rendering of the GUI object) and beyond. Furthermore, we can access the recorded information in **bach**.roll and perform simple time stretching and compressing operations in order to obtain slower or faster trajectories.

Another advantage of using **bach** is its microtonal support which we have taken advantage of (see Fig. D.6). In fact, some traditional melodies were used as a basis for the composition and those melodies contain various microtonal inflections. Therefore, it seemed relevant to extrapolate that feature to the remaining musical elements. Bach works in midicents, which our granular synth accepts. Regarding notation, it supports pitch breakpoints in the duration lines (see Figs. D.4 and D.5), allowing glissandi to be defined and played back.

D.4 Results

The system was assessed by composer and musicians during rehearsals (April to July 2019) and performance (6 and 7 July 2019).

The animated score was found to perform suitably, with faster performance of the bouncing ball when compared to previous experiments in the **Max** environment. Also, the appearance of the staff, despite being comprised of individual image files, is seamless.

Synchronisation between flautist and electronics guided by the bouncing ball was considered effective, but it must be taken into account that there were no events requiring very precise synchronisation (as part of our adaptation strategy). Other instruments' lines were subject to that of the flute and were easy to follow along

[13] llll stands for Lisp Like Linked List.

with the paper version of the score.

Manual latency compensation was found relatively easy to adjust sensorially. Once adjusted, there was some probability of a change in the latency value, causing the device to be slightly out of sync. In slower devices running older Android versions, this was more noticeable. However, this probability was not very significant. Moreover, our compositional strategy accounts for these imprecisions.

The dummy sound workaround for keeping resources active was effective with many of the devices available to us. However, it did not work with iOS devices nor with devices running Android version 8 or later. On iOS devices, the solution was to simply set the display to not enter sleep mode. Recent Android devices revealed more disadvantages: developer options had to be enabled and devices had to be connected to a power source in order to stay awake during the show.

At the sequencing level, the probability-based granular spatialiser was considered effective, being analogous to a graphic particles system simulating swarm intelligence. The use of the **bach**.roll for trajectory sequencing has enabled better performance by eliminating the need to perform a substantial number of calculations, replacing it with a lookup table approach.

Furthermore, using **bach** to sequence our granular synth's pitch and temporal parameters has proved advantageous in regards to the integration with the instrumental melody representation. It is noteworthy that the slot system enables a kind of interaction similar to the one enabled in Digital Audio Workstations (DAW) for automation control, with the advantage of a deeper integration with notation. On top of that, it allows us to sequence data for instruments created in **Max** which therefore are not available in any DAW software.

Finally, at the composition level, our adaptation strategies have made it possible to circumvent inherent problems in this type of system and achieve results with pleasing aesthetics. Regarding the sound spectrum restriction and the instrumental texture reduction, our choice was to use granulated instrumental sounds taken from the ensemble in order to create a dialogue with the respective acoustic instruments, as a duet. This approach offered timbre cohesion and balance, overcoming the poor quality and sound power of the devices. With regards to sound design, the system adapted well to the effects required by the narrative – namely, 'countryside' (represented by the bells of a flock of sheep), 'sea' (filtered white noise and seagulls), 'frantic tourism and its bipolar seasonality': extreme calmness vs chaos (smartphone ringtones, camera shutters, DTMF tones[14] and glitchy sounds). These sounds were chosen firstly based on their energy in the high spectrum and

[14]Dual-tone multi-frequency (DTMF) signaling is a telecommunication signaling system using the voice-frequency band over telephone lines between telephone equipment and other communications devices and switching centers.

secondly because they were in some way associated with the concept of flock – sheep, seagulls, tourists with their smartphones and cameras – and therefore appropriate for our Swarm Spatialiser. Sensations of sound source displacement were felt clearly around the room as expressed by a few audience members who came to us once the show had ended.

Also appreciated was the satirical use of the ringtones emitted by actual smartphones: it starts with only one phone ringing, thus leading the audience to think someone forgot to turn off their phone (remember: the devices are hidden under the audiences' chairs). After a short while and a blatant ringtone crescendo, the purpose becomes obvious.

In addition to the mobile devices, it was possible to obtain sound effects with stronger dynamics and lower frequencies by using a 2.1 sound system. Effects consisted in certain soundscapes and reinforcement of instrumental passages.

`GarB'urlesco` was conceived for a particular venue, in Lagos, Portugal. If the possibility ever comes of performing it elsewhere, some adaptation will likely be needed. For example, a larger venue will require a larger number of devices – which could raise problems regarding network capability – or the use of portable speakers, which would change the original sound characteristics.

D.5 Conclusions

A system of this nature can hardly compete with native application computer-based systems for distributed music in contexts where the music requires great accuracy and reliability. A notation system like `Comprovisador` takes advantage of the timing accuracy of `Max`'s scheduler and `bach`'s CAC tools enabling distributed computing of real-time generated scores with great flexibility and scalability. An array of speakers will deliver the full range of audible frequencies with the dynamic range of an orchestra and unsurpassable timing precision. But these systems involve significant preparation time and/or logistic resources.

Hence, in situations where it is possible to adapt the music creation to the idiosyncrasies of a web-based system, it is possible to take advantage of its best features (notably, ease of deployment) and achieve aesthetically pleasing results.

The case study presented herein showed the usefulness of some of the features available in the `bach` library for

achieving the desired results. On one hand, the new image-export feature allowed us to create a dynamic, nice-looking score on the browser using png files as discrete measures. On the other hand, the slot system present in the **bach**.score and **bach**.roll objects allowed integrated sequencing of parameter values and microtonal notation for our granular synth and respective Swarm Spatialiser. These latter tools were crucial in the accomplishment of the outlined compositional strategies.

E

Score Submission: Climate-Migrants

This text aims to show the possibility of a planned performance with the `Comprovisador` and the integration of an animated score for movement (the singers are choreographed in real time), in which the musical form is laden with metaphors. This is an updated version of the score submitted to the TENOR2020/2021 conference. The 2020/2021 submission was accepted, but the piece was not performed due to restrictions caused by the COVID19 pandemic. The updated version was submitted to the TENOR2022 conference, but unfortunately was not accepted. The original version featured Nora-Louise Müller as the soloist playing a Bohlen-Pierce clarinet. This is why Video Example 3.1 included in Chapter 3 is based on the Bohlen-Pierce scale.

TENOR 2022 – Score submission

Author / Composer

Pedro Louzeiro

Universidade de Évora, Portugal

Centro de Estudos de Sociologia e Estética Musical (CESEM), Lisbon, Portugal

Conservatório de Música de Loulé – Francisco Rosado (CML-FR), Portugal

plouzeiro@uevora.pt

Title and details of the piece

Climate Migrants*

for kinetic vocal improvisers, instrumental-vocal ensemble and the Comprovisador system (real-time composition and notation)

Duration: 10'00''

Performers:

- soloists: Singers from **Neue Vocalsolisten**
- ripieno: **Ensemble de Comprovisadores** – students from CML-FR, Portugal

Keywords: comprovisation, real time, staff-based score, audio score, movement score, microtonality.

* – This is **modified** version of the piece submitted to TENOR2020, which was accepted but not performed due to the remote nature of the conference imposed by COVID19 restrictions.

About the piece

Our planet is warming up. We hear about it on the news at an increasing rate but, in our daily lives, we are not so often confronted with the effects of such warming (… yet). All seems well, except for perhaps an unusually long bushfire season or an atypical golfball-sized hail storm. Thus, we go about our lives, taking everything for granted, from our food supply chain to our ever-growing economy. Many believe that, if we keep our "business as usual", we will be entering a state of runaway climate change very soon. And they say it will come with catastrophic consequences, namely the disruption of our food systems, which will inevitably cause societal collapse. Without food or land, massive quantities of people will be forced to migrate in order to survive. Demographic pressure will increase in certain parts of the world, leading to conflict and suffering.

This piece is a homage to the climate migrants of today and tomorrow. It features vocal improvisers (the soloists), an instrumental=>vocal** ensemble (the ripieno) and a real-time composition and notation system – Comprovisador. In real time, as the soloists improvise, Comprovisador's algorithms produce a music score which is immediately sight-read by the instrumentalists in the ripieno, creating a coordinated response to the improvisation. Apart from the music score, a movement score is also used, enabling the soloists to synchronously move to defined positions on stage, revealing shapes and providing the audience with kinetic auditory sensations. The soloists will view the movement score in handheld **browser-enabled** devices. Furthermore, musical interaction is mediated by a performance director via the systems's interface.

For theatrical effect, instrumentalists from the ripieno are placed in islands (e.g. the reeds island; the brass island) on **and off** stage. **At a given point in the piece (figuratively: **at climate breakdown**), most of the instrumentalists from the ripieno will become singers and **migrate** to the stage, following the movement score, causing ever increasing **demographic pressure**. Moreover, these musicians will sing in a microtonal tuning system – a metaphor for alien culture.

The music score involves two concurring dimensions: staff-based notation and audio cues, conveyed through individual computer screens and earphones – thus enabling microtonality for singers. The sung text

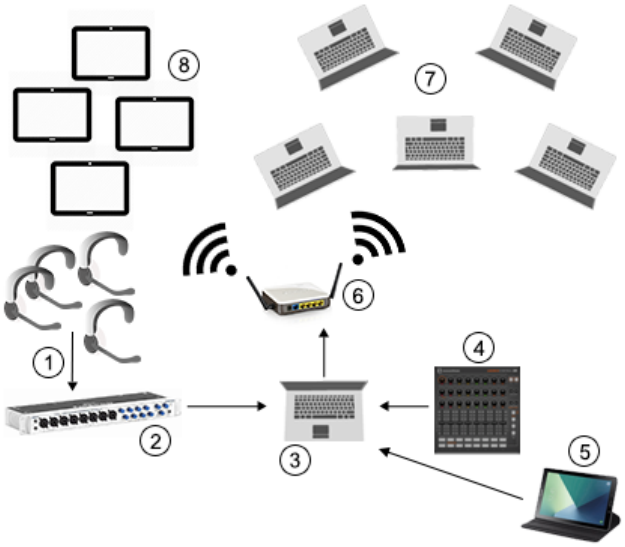
is generated through a rule-based process including weighted random selection of phonemes. The movement score consists of a graphic score which is animated according to oscillator functions. It works well in tablets and other browser-enabled devices.

Detailed description of the system

Implementation of Comprovisador (1) requires a network of computers in order to present audio-visual notation (separate parts) to each musician of the ripieno. Moreover, browser-enabled devices can be used to visualise the movement score (soloists). Wireless connectivity allows for devices – and therefore singers – to move freely, enabling space and kinetics to be used as compositional elements. A host computer centralizes algorithmic tasks accepting pitch input from the improvisers and parametric input from the mediator. It then transmits the output to all client computers, which render the music score. The movement score, coordinated by the host computer, consists of a web-app which is loaded by handheld browser-enabled devices.

Comprovisador's hardware configuration and data flow (see image):

1. wireless headset (or lavalier) microphones for pitch tracking – one for each soloist;
2. an audio interface;
3. a host computer;
4. a control surface;
5. a tablet – for instant messaging control;
6. a wireless router;
7. client computers (with earphones) for audio-visual score distribution – one for each ripieno musician;
8. tablets for movement score distribution – one for each soloist.



Please note: items 3, 4, 5 **and 7** will be provided by the proponent; items 2 and 6 should be provided by the organisation; items 1 and 8 must be provided by the organisation.

All software components of Comprovisador have been developed in Max (https://cycling74.com) using objects from the Bach library (http://www.bachproject.net) for notation rendering and for several algorithmic compositional tasks. The movement score makes use of the jit.mo library.

Following are some of Comprovisador's strong points:

Easy set up – With regards to networking, three approaches are in place to handle different situations: UDP, TCP, and Multicast UDP. The latter is used only at the beginning of a session in order for the host computer to automatically query the IP addresses of all client computers, making setting up a seamless process.

Distributed computing – In addition to notation rendering, the client application also carries out some algorithmic tasks that could in theory be performed by the host application. The goal of this task decentralisation is to unburden the host computer's CPU and to keep the wireless data traffic as lightweight as possible, raising the theoretical maximum number of connected clients.

Instrumentation flexibility – Many aspects of the host application are automatically configured on startup by means of a script that looks up an instrumentation list in crossed-reference to an instrumentation dictionary, both stored in text files. The latter contains a large set of information pertaining to each instrument (family, range, transposition, clef, dynamic range mapping, strings tuning, initial IP port number, etc.). This allows for the system to be flexible, easy to set up, yet idiomatically correct, regarding instrumentation. For

singers, this correctness means the system will never output a very low note if the dynamics are set to a high value and vice-versa.

Choral sound – Also for singers, the text to be sung is algorithmically generated in real time. Hence it is possible to have all singers sing identical phonemes, providing instant cohesive choral sound.

Reliable synchronisation – Following Dannenberg's (2) time-flow concepts (namely Level 2 Synchronisation), Comprovisador's synchronisation within the network is enforced by a NTP-like algorithm which is calibrated automatically (using an audio cable once during setup) to account for individual hardware latency differences between devices.

Effective cueing – The client application features three cueing strategies that work together to achieve a high level of effectiveness: 1) audio cues ensure good intonation and melodic guidance (singers), while assisting in attack precision; 2) a visual cueing device consisting on a bouncing ball enables score navigation while simulating a conductor's tempo gesture (see image); 3) a two-click metronome (which adapts to the period of the bouncing ball) amplifies attack precision (tutti). In contexts where no tempo unit is defined, the period of the bouncing ball (and metronome) is dependent on the inter-onset interval (see demonstration video), which will vary according to the improvisation. The level of accuracy these redundancies provide enable complex rhythmic effects such as **micropolyrhythms** and **microphasing** (e.g.: 16 musicians, each playing a note every 400ms, each out of phase by 25ms (400/16) resulting in a granular effect), which add up to the possibilities in harmony and tuning systems.

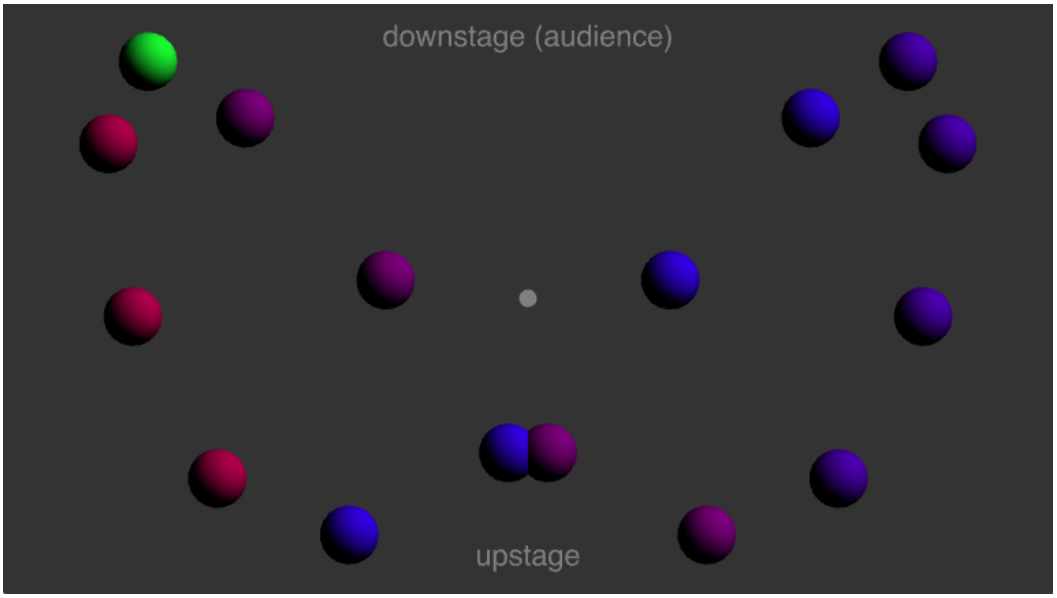
Open form – Comprovisador now allows triggering of precomposed passages during performance.

Preset manager – A thoughtfully outlined performance plan can be attained through creating presets of algorithmic parameters and corresponding movement / stage positions. Each preset yields different types of musical response, ranging from reactive synchronised *tutti* impacts to intricate micropolyphonic textures.

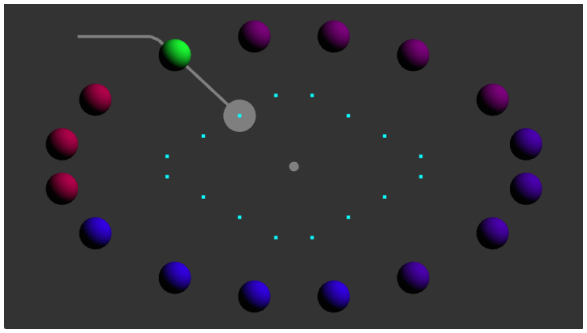
Practice tool – The client application also features a practice tool that allows performers to get acquainted with the system's notation interface and its idiosyncrasies and to further improve their sight-reading skills. It is currently being studied as an educational tool with students from CML-FR to extend the work presented during TENOR 2018, in Montreal (3).

Outline of the **movement score**:

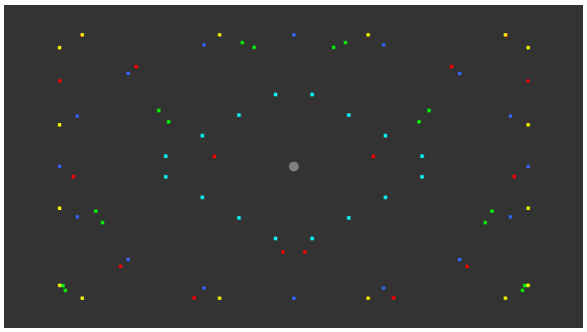
- point of view of singer facing audience (top of screen = downstage);
- colour-coded voice type (**soprano**, **alto**, **tenor**, **bass**) and highlighted **self**;



- a path with target is drawn prior to movement onset; each singer should mimic their avatar's trajectory in real time;
- soloists: text messages such as "SOLO", "DUET" or "TACET" can be displayed in the canvas, allowing the mediator to be in control of texture;
- ripieno: during motion, movement score window is in fullscreen; music score relies only on audio cues (staff notation not visible) and is restricted to easier music (e.g.: prolonged notes; repetitions);



- floor markings: only 5 structural constellations are marked (5 colours – see image below); score only needs to show relevant markings for each trajectory destination (see image on the right);
- score's aspect ratio **can be adjusted** according to stage proportions;
- trajectory timings **can be scaled** in function of stage size.



Motion paths do intercept, especially towards the end of the piece when more singers join the stage. This seeks to represent human pressure and conflict. Singers will be instructed on how to move when paths cross, intentionally avoiding face contact (potentially rubbing elbows and backs) while holding a laptop PC (ripieno) or a tablet (soloist).

Performers required

Soloists: Singers from **Neue Vocalsolisten** – minimum: 2; recommended: 4.

Ripieno: **Ensemble de Comprovisadores** – 10 to 16 students from CML-FR.

Technical specifications

Each CML-FR student will bring a Windows laptop PC (property of CML-FR) with **Comprovisador.client fully installed and tested**. Currently, this ensemble performs **weekly rehearsals** with this setup and will continue so at least until June 2022.

Regarding instrumentation, in most cases each student will their own instrument. However, the organisation is expected to provide a few large instruments, namely:

- **1 piano**
- **1 keyboard** (a clavinova-style or synth-style one)
- **2 mallet perc. instruments** (recommended: vibraphone + marimba; non-mallet alternatives accepted);
- **1 accordion** (if possible)
- **1 baritone saxophone or bass clarinet** (if possible)

Floor markings: **60 minutes** (5x16 = 80 points; final measurements / xy-coordinates depend on stage dimensions)

Setup time: **45 minutes**.

Rehearsal time: **two one-hour rehearsals** (one for movement and system acquaintance; another for the whole piece) plus a **half-hour dress-rehearsal**.

Equipment to be provided by the **composer / CML-FR**:

- **1 laptop** (host) – (*Apple MacBook Pro, 13-inch*);
- **1 control surface** – (*Novation Launch Control XL*);
- **1 tablet** – (*Samsung, 10-inch*);
- **10 to 16 laptops** (clients) – (*HP i7, 16-inch*);
- **10 to 16 sets of earphones** – (one for every laptop);

Equipment to be provided by **TENOR 2022** (*****or singers):

- **1 audio interface** – 4 mic inputs or greater; USB 2.0 or FireWire;
- **1 Wi-Fi router** – dual band, 1750Mbps or better;
- **2 to 4 wireless headset** (or lavalier) **microphones**[1];
- **2 to 4 tablets*** – Android or iOS;
- **1 small table and chair** – (interface / composer);
- **power supply** – for host computer, router and interface;
- **power supply "station"** – for battery top up of client laptops, prior to performance (backstage);
- **coloured tape** – floor markings, 5 different colours (e.g.: Red, Green, Blue, Cyan, Yellow);

Personnel needs:

- **stage crew** – placement of coloured floor markings according to final measurements / xy-coordinates (to be provided by the author).

[1] The requested microphones will be used to acquire pitch data from the soloists as input for the algorithmic system. If the organisation wishes capture and record or amplify the sound, overhead mics should be used.

Biography

Pedro Louzeiro is a Portuguese composer, born in 1975.

Currently a PhD candidate in Évora University, Portugal, he is doing research in the field of dynamic notation systems with focus on mediated soloist-ensemble interaction, under the supervision of Dr. Christopher Bochmann and Dr. António de Sousa Dias. He was awarded a PhD Studentship by the Portuguese Foundation for Science and Technology (FCT).

He received his Master's Degree in composition from Évora University, in 2013, and his Bachelor's in Music Education from the Lisbon College of Music, in 2002.

He was awarded 2nd prize in the II International Composition Competition "Goffredo Petrassi" (Rome, Italy, 2012) and was distinguished with honourable mentions in the X International Composition Competition "Carl von Ossietzky" (Oldenburg, Germany, 2010) and the IV International Composition Prize "Fernando Lopes-Graça" (Cascais, Portugal, 2014).

His most relevant works as a composer include a concerto for trumpet and symphonic band called "Proclamação", commissioned for the commemoration of the 1st Centenary of the Portuguese Republic, and "Água – a Seiva da Terra", a symphonic poem commissioned to mark the World Water Day.

He has carried out various performances with the "Comprovisador" system, which he develops in the scope of his PhD programme, presenting his work in international conferences such as: Sound and Music Computing (SMC2016, Hamburg, Germany, and SMC2018, Limassol, Cyprus), Computer Music Multidisciplinary Research (CMMR2017, Matosinhos, Portugal, and CMMR2019, Marseille, France), Internacional Computer Music Conference (ICMC2017, Shanghai, China) and International Conference on Technologies for Music Notation and Representation (TENOR2018, Montreal, Canada).

Video demonstration of the score in action

The following link provides a series of examples of the Comprovisador's score in action. The total time of the examples does not exceed 5 minutes (each example will automatically stop in due time).

Please note that some of the examples were made with older versions of the system, using different instruments. The newest examples, featuring the newest advancements, are identified as such in a caption on the top of the page.

Apart from the score demonstration, there are a few examples from an older piece entitled "Comprovisação nº 7" which is pertinent to the current submission. In this piece, 14 client computers were used without any technical or musical issues. Moreover, the piece features 9 singers and 5 instruments (plus one soloist), performing in two separate spaces and employing some of the techniques that will also be used in "Climate Migrants" – for example: groupings (male Vs. female voices, left Vs. right, voices Vs. instruments, etc.), microtonality (a short passage using spectral chords / just intonation), text with distinct probability weights assigned to each preset, among others.

https://climate-migrants.glitch.me/

References

(1) Pedro Louzeiro. The Comprovisador's real-time notation interface (extended version). In Mitsuko Aramaki, Matthew E. P. Davies, Richard Kronland-Martinet, and Sølvi Ystad, editors, *Music Technology with Swing*, pages 489–508. Springer International Publishing, Cham, 2018.

(2) Roger B. Dannenberg. Time-flow concepts and architectures for music and media synchronization. In *Proceedings of the 43rd International Computer Music Conference*, pages 104–109, Shanghai, 2017.

(3) Pedro Louzeiro. Improving sight-reading skills through dynamic notation – the case of Comprovisador," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – Tenor'18*, pages 55–61, Montréal, 2018.

F

Projecto b): Classe das Notas Animadas

Below is a project that was recently submitted to the **Conservatório de Música de Loulé – Francisco Rosado**, in Portugal (hence it is written in Portuguese), to be implemented during the 2023/2024 school year. At the same time, a proposal named ***Projecto a): Ensemble de Comprovisadores*** was put forward, continuing the homonymous project carried out in 2021/2022, which resulted in the previously mentioned performance "Comprovisação nº 11".

The Pedagogical Model of Rehearsal presented in ***Projecto b)*** (se Section F.4) makes use of `Comprovisador`'s advanced real-time resources. These resources, such as loops, instrumentation management, bouncing ball, adaptable metronome, cue notes and `Practice Mode`, have the potential to enhance productivity during

rehearsals, especially for students who are not yet very autonomous. This approach represents a significant novelty in the field.

F.1 Preâmbulo

Este projecto destina-se a alunos que se encontrem a frequentar o ensino básico (2º e 3º ciclo) no Conservatório de Música de Loulé – Francisco Rosado (CML-FR), a partir do 5º ano de escolaridade, e visa a criação de pequenas **Classes de Conjunto** que trabalharão **repertório criado para o efeito** através do sistema **`Comprovisador`**, sob a forma de **partituras animadas**. Com efeito, este sistema, tendo sido originalmente concebido para a prática de performances de comprovisação (ver Projecto a)), integra algumas ferramentas que se constituem como mais-valias no âmbito da pedagogia musical, nomeadamente:

- uma ferramenta de estudo (chamada **`Practice Mode`**) que gera uma partitura animada para efeitos de desenvolvimento das competências de leitura à primeira vista, onde o grau de dificuldade[1] pode ser ajustado em tempo real, através de parâmetros algorítmicos;
- a possibilidade de reproduzir partituras animadas previamente criadas (quer seja pelo próprio sistema, quer seja por um compositor humano), usando os diferentes modos de animação do sistema (bolinha saltitona, loops, notação rítmica proporcional ou tradicional).

Assim, os alunos que participarem neste projecto terão a oportunidade de tocar peças de conjunto especificamente compostas para o nível instrumental **de cada aluno do grupo**, mesmo em grupos heterogéneos. Quer seja por questões de ritmo de aprendizagem, quer seja pelas idiossincrasias de cada instrumento, é muito comum, nos primeiros anos de formação, haver diferenças assinaláveis no nível de desempenho instrumental de alunos do mesmo grupo etário. A abordagem através do **`Comprovisador`** e das suas possibilidades de modificação da partitura em tempo-quase-real, permite **uma diferenciação pedagógica e uma verdadeira integração**.

Outra vantagem deste sistema, em paralelo com o trabalho sobre o repertório criado, é o uso do **`Practice Mode`** que permite desenvolver a prática da leitura à primeira vista, igualmente, num contexto de grupo primado

[1] Seja qual for o contexto generativo – **`Practice Mode`** ou performance de comprovisação – a notação musical gerada pelo **`Comprovisador`** tem sempre em consideração aspectos idiomáticos do instrumento em causa (p. ex., tessitura, registo mais confortável em função da dinâmica, etc.). No caso do **`Practice Mode`**, é ainda possível restringir o número de possibilidades de notação (p. ex., apenas as notas dó e sol, apenas cordas soltas, apenas graus conjuntos, etc.).

pela diferenciação pedagógica.

Finalmente, uma dimensão interessante deste projecto seria a possibilidade de dinamizar a **classe de Análise e Técnicas de Composição**, incentivando os alunos a escrever pequenas peças para as diferentes classes de conjunto formadas.

O proponente do projecto tem experiência profissional no campo da leccionação de classes de conjunto de alunos do 5º ano[2], com instrumentação variada e muito própria e, por conseguinte, com a necessidade de criação de repertório específico. O modelo pedagógico usado nessa experiência passada consistiu, ao início, na composição em sala de aula, no quadro, de pequenos fragmentos musicais que, aula após aula, foram tomando forma e transformando-se em peças completas. O conjunto de peças compostas pelo professor e apresentadas pelos alunos foi publicado, integrando um artigo científico [Lou14], onde foi feita uma reflexão acerca das preocupações pedagógicas inerentes à composição de cada uma das peças. De resto, no ano lectivo de 2021-2022, o proponente compôs uma peça para os seus alunos da turma de 5º ano, no âmbito da disciplina de Formação Musical, no CML-FR, que foi lida pelos mesmos com sucesso moderado, durante a sua última aula.

Sobre as valências do `Comprovisador` enquanto ferramenta didática, é pertinente referir a investigação de Mestrado em Ensino de Música levada a cabo pelo guitarrista Joaquim Nascimento, intitulada "**O Contributo do `Comprovisador` na Leitura Musical à Primeira Vista**" [Nas19]. Nas conclusões da sua investigação, pode ler-se o seguinte:

> Sendo que lê melhor à primeira vista quem consegue fazer melhores antecipações, pode concluir-se que o `Comprovisador` foi um facilitador de tal competência (…). (…) o foco do aluno estava em tocar a nota, e não em utilizar a digitação x ou y, (…) ou seja, o aluno opta por uma alternativa de digitação em tempo real.
>
> (…) A prática da leitura musical à primeira vista utilizando o Practice Mode do `Comprovisador` permitiu-nos concluir existir a adoção natural de uma postura mais correta, podendo relacionar este comportamento com uma empatia perante o objeto (…) traduzindo-se [na adoção de] uma atitude positiva, não constrangedora, perante a falha, não existindo tensão perante os erros que possam ocorrer.
>
> (…) o aluno demonstrou uma preocupação em preparar a mão esquerda, colocando os dedos o mais perto possível das notas que previa aparecerem, contribuindo isso para o aperfeiçoamento

[2] Academia de Música de Lagos.

> técnico da mão esquerda, bem como para o progresso de novas competências, traduzindo-se em automatismos digitais criando a ligação final entre a visão e a execução.
>
> (…) registou-se estarmos na presença de um aproximado estado de fluxo (…), em que os alunos se concentraram ativamente. Saliente-se que as condições para esse estado de fluxo se encontram reunidas no trabalho com o `Comprovisador`, ou seja: os objetivos são claros; existe um alto grau de concentração e foco num determinado campo de ação; o feedback é imediato; a atividade proposta nunca é demasiado simples ou complicada; existe uma sensação de controlo sobre a situação.
>
> (…) A interação com o software revelou-se uma mais valia na motivação e no empenho (…). (…) existiu homogeneidade no que concerne aos níveis motivacionais, os quais se mantiveram elevados na realização da tarefa proposta em aula. Foi possível verificar o aumento do interesse dos alunos pela prática da [leitura musical à primeira vista], através da utilização sistemática do "Jogo".

F.2 Objectivos

Os objectivos do projecto, diferenciados por tipo de participante, são definidos da seguinte forma:

- para os alunos do projecto:
  - desenvolver competências de execução musical em grupo;
  - adquirir e melhorar competências de leitura à primeira vista, no instrumento;
  - consolidar a técnica instrumental;
  - ganhar competências de leitura com partituras animadas;
  - obter experiência performativa;
- para alunos de ATC (ou outros voluntários):
  - ganhar experiência no acto de compor, com preocupações de âmbito didático;
  - ter a oportunidade de ouvir as suas composições tocadas e de fazer revisões, de acordo com o seu julgamento estético e com o feedback do orientador;
- para o orientador:

- melhorar as suas competências composicionais e instrumentais – aplicadas ao sistema `Comprovisador`, enquanto ambiente de composição e enquanto instrumento digital;
- optimizar o sistema ao nível do desempenho computacional (reprogramação / depuração);
- com base na prática artístico-pedagógica desenvolvida, reflectir sobre as valências e limitações estéticas e didácticas do sistema `Comprovisador`.

F.3 Implementação

Os destinatários deste projecto são **alunos do ensino básico**, de qualquer instrumento, a partir do 5º ano de escolaridade.

As composições serão levadas a cabo, em primeira instância, pelo orientador. De resto, os alunos da classe de ATC serão convidados a dar as suas contribuições ao repertório dos grupos formados.

Para além do ensaio das peças compostas, haverá lugar à prática da leitura à primeira vista, fazendo uso das ferramentas generativas do `Comprovisador`.

Quanto à periodicidade, pretende-se que os ensaios sejam **semanais**, com duração de 45 minutos (um tempo lectivo), em horário a definir. Idealmente, os ensaios dos diferentes grupos deverão ocorrer no mesmo dia da semana, em articulação com o projecto a), de modo a optimizar a logística da montagem do material. Havendo constrangimentos que impossibilitem a realização de ensaios semanais, poderá ser estudada uma periodicidade mensal.

No que respeita ao espaço físico, o local mais adequado para a realização dos ensaios deste grupo é o **Auditório do Solar da Música Nova**, embora outros espaços possam ser considerados.

Em termos de recursos materiais, para além dos instrumentos musicais, será necessário **um computador portátil por participante**[3] (alunos + professor). O orientador levará o seu próprio material, a saber: um tablet, um router, um controlador MIDI e um teclado MIDI.

No que se refere a recursos humanos, por forma a optimizar o tempo de ensaio, será necessária uma pessoa, em coordenação com o orientador, para preparar o espaço e os recursos materiais (nomeadamente, com-

[3] Em certos casos, se houver um número elevado de participantes, estes podem agrupar-se dois a dois por cada computador.

putadores), durante 15 minutos, duas horas antes do início do (primeiro) ensaio (o tempo restante servirá para montagem da parte técnica, a cargo do orientador). Será igualmente necessário desmontar e arrumar os materiais, após o término do (último) ensaio – 15 minutos.

Relativamente aos outputs do projecto, pretende-se a realização de **pelo menos três concertos públicos**, com a participação dos vários grupos, a realizar ao longo do ano lectivo de 2023-2024.

F.4 Modelo Pedagógico dos Ensaios

O foco dos ensaios incidirá no trabalho sobre o repertório criado para cada grupo. As possibilidades tempo-real presentes no `Comprovisador`, como sejam a definição de ciclos (loops) e a gestão da instrumentação, permitirão abordar situações comuns de ensaio com maior conveniência e pragmatismo, resultando em ensaios mais dinâmicos e produtivos. Imagine-se a seguinte situação, num ensaio tradicional:

- é encontrada uma dificuldade musical, numa passagem;
- o maestro manda parar, informando os alunos sobre:
  - o problema musical a resolver;
  - o número de compasso a partir do qual irão recomeçar;
  - quais os instrumentos que deverão tocar / descansar;
  - (repetir as informações para os alunos que se mostrarem desatentos);
  - levantar a batuta e dar a entrada;
  - repetir até que o problema encontrado esteja corrigido.

Uma situação análoga, com o `Comprovisador`, seria resolvida em movimento, da seguinte forma:

- é encontrada uma dificuldade musical, numa passagem;
- o maestro activa a função de loop, abrangendo os compassos em causa, e:
  - isola os instrumentos relevantes (desactivando os restantes);
  - manipula o andamento, se necessário (e.g., lento, acelerando gradualmente);

  - envia mensagens específicas, se necessário* (e.g., @trompete: o fá é ♯);

- a música não chega a parar, durante a resolução do problema, os alunos mantêm-se concentrados e o trabalho é mais produtivo.

De resto, a navegação da partitura e a sincronização são asseguradas pelas funcionalidades **bolinha saltitona**, **metrónomo adaptável** e **notas guia** (via earphones), pelo que será muito raro um aluno perder-se, mesmo havendo manipulação agógica. As notas guia têm ainda a vantagem de ajudar o aluno a auto-corrigir problemas de leitura (*) e de afinação, em articulação com a função de loop.

Será reservado algum tempo (cerca de 10 minutos por ensaio) para a prática da leitura à primeira vista, utilizando as ferramentas generativas do `Comprovisador`. Para uma demonstração do `Practice Mode` em uso por jovens estudantes em contexto de leitura à primeira vista, veja-se o Exemplo Vídeo F.1. A melodia deste exemplo foi gerada em tempo-real, usando apenas notas da tonalidade de Ré M. As estudantes presentes neste exemplo nunca tinham tido contacto com o `Comprovisador`.

Video Example F.1

Duas estudantes lêem à primeira vista uma melodia gerada pelo Practice Mode.

F.5 Avaliação do Projecto

A observação directa dos ensaios e dos concertos que vierem a ser realizados permitirá avaliar se os objectivos propostos foram atingidos.

O feedback informal dos espectadores dos concertos será também um indicador ponderável na avaliação.

Espera-se que o **desenvolvimento da leitura à primeira vista dos alunos**, sobretudo no instrumento, seja positivamente impactado, o que poderá ser avaliado pelos professores de instrumento e também de formação musical, no contexto das suas aulas.

Por fim, a reflexão sobre estes indicadores será um contributo importante para a investigação que o autor do `Comprovisador` leva a cabo, no âmbito do seu doutoramento, em torno das potencialidades dos sistemas de notação em tempo-real.

Bibliography

[ABC+15] G. Assayag, G. Bloch, M. Chemillier, B. Lévy, and S. Dubnov. Omax, 2015. URL: **http://repmus.ircam.fr/omax/home** [cited 2017/04/04].

[AG10] Andrea Agostini and Daniele Ghisi. Bach: Automated composer's helper, 2010. URL: **http://www.bachproject.net** [cited 2017/04/01].

[AG15] Andrea Agostini and Daniele Ghisi. A max library for musical notation and computer-aided composition. *Computer Music Journal*, 39(2):11–27, 2015. **doi:10.1162/COMJ_a_00296**.

[AG18] Andrea Agostini and Daniele Ghisi. Pitches in bach. In Sandeep Bhagwati and Jean Bresson, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, pages 128–137, Montreal, Canada, 2018. Concordia University. URL: **https://www.tenor-conference.org/proceedings.html**.

[all19] "allmusic", 2019. URL: **https://www.allmusic.com/search/all/comprovisation%20comprovisations** [cited 2019-01-31].

[And11] Dimitris Andrikopoulos. Pwgl. an introduction to the program and some practical applications. In *International Conference Musica Nova VIII Janacek Academy of Music and Performing Arts*, Brno, 2011.

[Ant14] Elliott Antokoletz. *A History of Twentieth-Century Music in a Theoretic-Analytical Context*. Routledge, New York, 2014. URL: **https://books.google.pt/books?id=qrkTAwAAQBAJ**.

[Ape95] T. Apel. Max and max for live patches and externals, 1995. URL: **http://vud.org/max/** [cited 2017/01/31].

[Auy09] Sunny Y. Auyang. *Engineering – An Endless Frontier*. Harvard University Press, 2009. URL: **https://books.google.pt/books?id=XOa4iwk7cR4C**.

[Bar03] Álvaro Barbosa. Displaced soundscapes: A survey of network systems for music and sonic art creation. *Leonardo Music Journal*, 13:53–59, 2003. **doi:10.1162/096112104322750791**.

[BC11] Álvaro Barbosa and João Cordeiro. The influence of perceptual attack times in networked music performance. In *Audio Engineering Society 44th International Conference - Audio Networking*, San Diego, 2011. URL: **http://repository.uwl.ac.uk/id/eprint/5266/**.

[BC19] Jonathan Bell and Benedict Carey. Animated notation, score distribution and ar-vr environments for spectral mimetic transfer in music composition. In Cat Hope, Lindsay Vickery, and Nat Grant, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'19*, pages 7–14, Melbourne, Australia, 2019. Monash University. URL: **https://www.tenor-conference.org/proceedings.html**.

[BCB+16] Sandeep Bhagwati, Isabelle Cossette, Joanna Berzowska, Adam Basanta, Julian Stein, Joseph Browne, Alexandra Bachmeyer, Felix Del Tredici, Sarah Albu, Marcello Giordano, John Sullivan, Deborah Egloff, Marcelo Wanderley, and Julian Klein. Musicking the body electric: The "body:suit:score" as a polyvalent score interface for situational scores. In *Proceedings of the 2nd International Conference on Technologies for Music Notation and Representation*, page 121—126, Cambridge, UK, 2016.

[BCG05] Álvaro Barbosa, Jorge Cardoso, and Gunter Geiger. Network latency adaptive tempo in the public sound objects system. In *Proceedings of the 2005 Conference on New Interfaces for Musical Expression (NIME)*, page 184–187, Vancouver, 2005. **doi:10.5555/1085939.1085991**.

[BD99] Eli Brandt and Roger B. Dannenberg. Time in distributed real-time systems. In *Proceedings of the 1999 International Computer Music Conference*, pages 523–526, Beijing, 1999. Carnegie Mellon University. URL: **http://repository.cmu.edu/compsci/498/**.

[Bea38] Kenneth L. Bean. An experimental approach to the reading of music. *Psychological Monographs*, 50(6):i–80, 1938. **doi:10.1037/h0093540**.

[Bel18] Jonathan Bell. Audiovisual scores and parts synchronized over the web. In Sandeep Bhagwati and Jean Bresson, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, pages 17–23, Montreal, Canada, 2018. Concordia University. URL: **https://www.tenor-conference.org/proceedings.html**.

[Ben99] Olivier Benoît. La pieuvre, 1999. URL: **http://obenoitmusic.free.fr/pieuvre.htm** [cited 2018/12/10].

[Ben16] Andrew Benson. Content you need: Miraweb, 2016. URL: **https://cycling74.com/articles/content-you-need-miraweb** [cited 2017/04/13].

[Bha08] Sandeep Bhagwati. Towards interactive onscreen notations for comprovisation in large ensembles. In *Proceedings of the Symposion for Systems Research in the Arts and Humanities*, Baden-Baden, 2008.

[Bha12] Sandeep Bhagwati. Comprovisations – improvisation technologies for the performing arts, 2012. Research-Creation Workshop. Hexagram Concordia & matralab, Concordia University Montréal. URL: **http://matralab.hexagram.ca/comprovisations2012/about/** [cited 2019/01/24].

[Bha13a] Sandeep Bhagwati. Comprovisation – concepts and techniques. In Henrik Frisk and Stefan Östersjö, editors, *(Re) thinking Improvisation: Artistic Explorations and Conceptual Writing*, Doctoral Studies and Research in Fine and Performing Arts, pages 99–104. Malmö Academy of Music, Lund University, 2013.

[Bha13b] Sandeep Bhagwati. Comprovisation – concepts and techniques, 2013. signale graz – Konzertreihe für Elektroakustische Musik, Algorithmische Komposition, Radiokunst und Performance – program notes. URL: **https://signale.kug.ac.at/signale-graz/konzerte/0111.html** [cited 2019-02-03].

[Bha13c] Sandeep Bhagwati. Notational perspective and comprovisation. In Paulo de Assis, William Brooks, and Kathleen Coessens, editors, *Sound & Score: Essays on Sound, Score and Notation*, pages 165–177. Leuven University Press, Leuven, 2013.

[Bha17] Sandeep Bhagwati. Vexations of ephemerality. In *Proceedings of the International Conference on Technologies for Music Notation and Representation*, pages 161–166. Universidade da Coruña, may 2017. **doi:10.5281/zenodo.924185**.

[Bha18] Sandeep Bhagwati. Elaborate audio scores: Concepts, affordances and tools. In Sandeep Bhagwati and Jean Bresson, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, pages 24–32, Montreal, Canada, 2018. Concordia University. URL: **https://www.tenor-conference.org/proceedings.html**.

[Bha19] Sandeep Bhagwati. matraprojects – Projects by Sandeep Bhagwati, 2019. matralab – Concordia University in Montréal. URL: **http://matralab.hexagram.ca/projects/#projectsSandeep** [cited 2019-02-03].

[BHE+18] Andrew Brown, Matthew Horrigan, Arne Eigenfeldt, Toby Gifford, and Daniel Field. Interacting with musebots. In *2018 New Interfaces for Musical Expression (NIME) Conference Proceedings*, pages 19–24, Blacksburg, 2018. URL: **http://hdl.handle.net/10072/378315**.

[Boc03] Christopher Bochmann. *A Linguagem Harmónica do Tonalismo*. Juventude Musical Portuguesa, Lisboa, 2003.

[Boc06] Christopher Bochmann. Para uma formação actualizada. *Revista da Associação Portuguesa de Educação Musical*, pages 28–40, 2006. URL: **http://hdl.handle.net/10174/2915**.

[Car67] Cornelius Cardew. *Treatise*. Gallery Upstairs Press, Buffalo, 1967.

[CBC+18] Nadine Couture, Sébastien Bottecchia, Serge Chaumette, Mateo Cecconello, Josu Rekalde, and Myriam Desainte-Catherine. Using the soundpainting language to fly a swarm of drones. In Jessie Chen, editor, *Advances in Human Factors in Robots and Unmanned Systems*, pages 39–51, Cham, 2018. Springer International Publishing.

[CC09] Juan-Pablo Cáceres and Chris Chafe. JackTrip: Under the hood of an engine for network audio. In *Proceedings of International Computer Music Conference*, page 509–512, San Francisco, California, 2009. International Computer Music Association, International Computer Music Association.

[CE18] Linda Candy and Ernest Edmonds. Practice-Based Research in the Creative Arts: Foundations and Futures from the Front Line. *Leonardo*, 51(1):63–69, 02 2018. URL: **https://doi.org/10.1162/LEON_a_01471**, **arXiv:https://direct.mit.edu/leon/article-pdf/51/1/63/1578280/leon_a_01471.pdf**, **doi:10.1162/LEON_a_01471**.

[Cha77] Joel Chadabe. Some reflections on the nature of the landscape within which computer music systems are designed. *Computer Music Journal*, 1(3):5–11, 1977.

[Col12] J.F. Colson. *Conducting and Rehearsing the Instrumental Music Ensemble: Scenarios, Priorities, Strategies, Essentials, and Repertoire*. Scarecrow Press, 2012. URL: **https://books.google.pt/books?id=TsbP1knYKikC**.

[Col13] Nicolas Collins. Roomtone variations, 2013. Premiered October 2013, Mills College, Oakland, CA. URL: **https://youtube/JrqMmY8ikuA** [cited 2020-08-12].

[Col16] Dave Collins. *The Act of Musical Composition: Studies in the Creative Process*. SEMPRE Studies in The Psychology of Music. Routledge, London, 2016. URL: **https://books.google.pt/books?id=KXzeCwAAQBAJ**.

[com] MaxScore. Hochschule für Musik und Theater Hamburg. URL: **http://www.computermusicnotation.com/** [cited 2020-08-10].

[CRRP16] Alexandre Resende Clément, Filipe Ribeiro, Rui Rodrigues, and Rui Penha. Bridging the gap between performers and the audience using networked smartphones: the a.bel system. In *Proceedings of the International Conference on Live Interfaces*, Brighton, 2016. URL: **http://users.sussex.ac.uk/~thm21/ICLI_proceedings/2016/Papers/Short_Papers/127_Abel.pdf**.

[Dan17] Roger B. Dannenberg. Time-flow concepts and architectures for music and media synchronization. In *Proceedings of the 43rd International Computer Music Conference*, pages 104–109, Shanghai, 2017. URL: **https://www.cs.cmu.edu/~rbd/papers/timeflow2017.pdf**.

[DFLO09] Christophe Daudin, Dominique Fober, Stéphane Letz, and Yann Orlarey. The guido engine – a toolbox for music scores rendering. In *Proceedings of the Linux Audio Conference*, pages 105–111, Parma, Italy, 2009. URL: **https://guido.grame.fr/papers/lac2009.pdf**.

[DH08] Nick Didkovsky and Georg Hajdu. Maxscore: Music notation in max/msp. In *Proceedings of the International Computer Music Conference*, Belfast, 2008. URL: **http://hdl.handle.net/2027/spo.bbp2372.2008.034**.

[dis19] Discogs, 2019. URL: **https://www.discogs.com/search/?q=comprovisation&layout=med** [cited 2019-01-31].

[DJ97] C. Dodge and T.A. Jerse. *Computer Music: Synthesis, Composition, and Performance*. Schirmer Books, New York, second edition, 1997. URL: **https://books.google.pt/books?id=eY_BQgAACAAJ**.

[Dud10] Richard Dudas. Comprovisation: The various facets of composed improvisation within interactive performance systems. *Leonardo Music Journal*, 20:29–31, 2010.

[Eig07] Arne Eigenfeldt. Real-time composition or computer improvisation? a composer's search for intelligent tools in interactive computer music. In *Proceedings of the Electroacoustic Music Studies Conference*, Leicester, 2007.

[Eig14] Arne Eigenfeldt. Generative music for live performance: Experiences with real-time notation. *Organised Sound*, 19(3):276–285, 2014. **doi:10.1017/S1355771814000260**.

[Eig16] Arne Eigenfeldt. Exploring moment-form in generative music. In *Proceedings of the Sound and Music Computing Conference 2016*, pages 123–128, Hamburg, 2016.

[Ess95] Karlheinz Essl. Lexikon-sonate. an interactive realtime composition for computer-controlled piano. In *proceedings of the II Brazilian Symposium on Computer Music*, 1995. URL: **http://www.essl.at/bibliogr/lexson-sbcm.html**.

[FC10] Jason Freeman and Andrew Colella. Tools for real-time music notation. *Contemporary Music Review*, 29(1):101–113, 2010. **doi:10.1080/07494467.2010.509599**.

[FDLO10] D Fober, C Daudin, S Letz, and Y Orlarey. Partitions musicales augmentées. In *Actes des Journées d'Informatique Musicale*, Rennes, 2010.

[FOL17] Dominique Fober, Yann Orlarey, and Stéphane Letz. Towards dynamic and animated music notation using INScore. In Vincent Ciciliato, Yann Orlarey, and Laurent Pottier, editors, *Linux Audio Conference*, pages 43–51, Saint-Etienne, France, 2017. CIEREC. URL: **https://hal.archives-ouvertes.fr/hal-02158969**.

[FOLM19] Dominique Fober, Yann Orlarey, Stéphane Letz, and Romain Michon. A tree based language for music score description. In *Proceedings of the 14th International Symposium on Computer Music Multidisciplinary Research*, pages 737–744, Marseille, 2019. URL: **https://cmmr2019.prism.cnrs.fr/downloads.html**.

[Fre07] Jason Freeman. Flock, 2007. URL: **http://distributedmusic.gatech.edu/flock/** [cited 2019-05-06].

[Fre08] Jason Freeman. Extreme sight-reading, mediated expression, and audience participation: Real-time music notation in live performance. *Computer Music Journal*, 32(3):25–41, 2008. URL: **http://www.jstor.org/stable/40072645**.

[G+11] Carlos Guedes et al. Kinetic, 2011. INESC TEC / FEUP - SMC Group. URL: **http://smc.inesctec.pt/kinetic/** [cited 2018/01/04].

[GA17] Daniele Ghisi and Andrea Agostini. Extending bach: A family of libraries for real-time computer-assisted composition in max. *Journal of New Music Research*, 46(1):34–53, 2017. **doi:10.1080/09298215.2016.1236823**.

[GA18] Daniele Ghisi and Carlos Agon. dada: Non-standard user interfaces for computer-aided composition in max. In Sandeep Bhagwati and Jean Bresson, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, pages 147–156, Montreal, Canada, 2018. Concordia University. URL: **https://www.tenor-conference.org/proceedings.html**.

[Gal03] Philip Galanter. What is generative art? complexity theory as a context for art theory. In *Proceedings of the 6th Generative Art Conference*, Milan, 2003.

[Gat16] Gonçalo Gato. *Algorithm and Decision in Musical Composition*. PhD thesis, Guildhall School of Music and Drama, 2016. URL: **http://openaccess.city.ac.uk/17292/**.

[GG04] E.A.H. Green and M. Gibson. The expressive gestures. In *The modern conductor*. Pearson Prentice Hall, Upper Saddle River, seventh edition, 2004.

[GH19] Rama Gottfried and Georg Hajdu. Drawsocket: A browser based system for networked score display". In Cat Hope, Lindsay Vickery, and Nat Grant, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'19*, pages 15–25, Melbourne, Australia, 2019. Monash University. URL: **https://www.tenor-conference.org/proceedings.html**.

[Gio17] Artemi Maria Gioti. Machine listening in interactive music systems: Current state and future directions. In *Proceedings of the 43rd International Computer Music Conference*, pages 216–220, Shanghai, 2017.

[Gio18] Artemi-Maria Gioti. Neurons: An interactive composition using a neural network for recognition of playing techniques. In Philippe Pasquier, Oliver Bown, and Arne Eigenfeldt, editors, *Proceedings of the 6th International Workshop on Musical Metacreation (MUME 2018)*, Salamanca, 2018.

[Gir18] Kevin Gironnay. Exploring with ilÉa ensemble: Shaping freedom in improvised music. In Sandeep Bhagwati and Jean Bresson, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, pages 50–54, Montreal, Canada, 2018. Concordia University. URL: **https://www.tenor-conference.org/proceedings.html**.

[GKM+18] Toby Gifford, Shelly Knotts, Jon McCormack, Stefano Kalonaris, Matthew Yee-King, and Mark d'Inverno. Computational systems for music improvisation. *Digital Creativity*, 29(1):19–36, 2018. **doi:10.1080/14626268.2018.1426613**.

[Gou18] Vincent Goudard. John, the semi-conductor: A tool for comprovisation. In Sandeep Bhagwati and Jean Bresson, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, pages 43–49, Montreal, Canada, 2018. Concordia University. URL: **https://www.tenor-conference.org/proceedings.html**.

[Gro11] Darwin Grosse. Jsui-mgraphics patch-a-day, 2011. URL: **https://cycling74.com/forums/jsui-mgraphics-patch-a-day** [cited 2017/04/23].

[Gue17] Carlos Guedes. Real-time composition, why it still matters: A look at recent developments and potentially new and interesting applications. In *Proceedings of the 43rd International Computer Music Conference*, pages 162–167, Shanghai, 2017. URL: **http://www.icmc2017.com/en/download.html**.

[Gue18] Carlos Guedes. Composing and improvising. in real time. In Mitsuko Aramaki, Matthew E. P. Davies, Richard Kronland-Martinet, and Sølvi Ystad, editors, *Music Technology with Swing*, pages 445–453, Cham, 2018. Springer International Publishing.

[gui20] The Guido project, 2020. Grame-CNCM. URL: **https://guidodoc.grame.fr/about/** [cited 2020-08-10].

[Haj05] Georg Hajdu. Quintet.net: An environment for composing and performing music on the internet. *Leonardo*, 38(1):23–30, 2005. **doi:10.1162/leon.2005.38.1.23**.

[Haj08] Georg Hajdu. Real-time composition and notation in network music environments. In *Proceedings of the International Computer Music Conference*, Belfast, 2008.

[Haj16] Georg Hajdu. Disposable music. *Computer Music Journal*, 40(1):25–34, 2016. **doi:10.1162/COMJ_a_00342**.

[Har04] James Harley. *Xenakis: His Life in Music*. Routledge, 2004. URL: **https://books.google.pt/books?id=hHXqbJGEM4EC**.

[Hau77] Bernard Haultier. *Nouveau Solfège*, volume 1. Editions Musicales Hortensia, Paris, 1977.

[HD09] Georg Hajdu and Nick Didkovsky. On the evolution of music notation in network music environments. *Contemporary Music Review*, 28(4-5):395–407, 2009. **doi:10.1080/07494460903422313**.

[HD18] Georg Hajdu and Nick Didkovsky. Maxscore: Recent developments. In Sandeep Bhagwati and Jean Bresson, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, pages 138–146, Montreal, Canada, 2018. Concordia University. URL: **https://www.tenor-conference.org/proceedings.html**.

[HG19] Georg Hajdu and Rama Gottfried. Networked music performance in the old elbe tunnel. In Cat Hope, Lindsay Vickery, and Nat Grant, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'19*, pages 55–60, Melbourne, Australia, 2019. Monash University. URL: **https://www.tenor-conference.org/proceedings.html**.

[HI59] Lejaren Hiller and Leonard Isaacson. *Experimental Music: Composition with an Electronic Computer*. McGraw-Hill, 1959. URL: **https://archive.org/details/experimentalmusi00hill**.

[HRO19] Adrian Holovaty, Corey Richardson, and Edward O'Riordan. Soundslice, 2019. URL: **https://www.soundslice.com** [cited 2019-05-06].

[HV11] Cat Hope and Lindsay Vickery. Screen scores: New media music manuscripts. In *Proceedings of the International Computer Music Conference 2011*, pages 224–230, Huddersfield, 2011. URL: **http://hdl.handle.net/2027/spo.bbp2372.2011.045**.

[HVWJ15] Cat Hope, Lindsay Vickery, Aaron Wyatt, and Stuart James. The decibel scoreplayer - a digital tool for reading graphic notation. In Marc Battier, Jean Bresson, Pierre Couprie, Cécile Davy-Rigaux, Dominique Fober, Yann Geslin, Hugues Genevois, François Picard, and Alice Tacaille, editors, *Proceedings of the First International Conference on Technologies for Music Notation and Representation – TENOR'15*, pages 58–69, Paris, France, 2015. URL: **https://www.tenor-conference.org/proceedings.html**.

[HWT18] Cat Hope, Aaron Wyatt, and Daniel Thorpe. Scoring an animated notation opera – the decibel score player and the role of the digital copyist in 'speechless'. In Sandeep Bhagwati and Jean Bresson, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, pages 193–200, Montreal, Canada, 2018. Concordia University. URL: **https://www.tenor-conference.org/proceedings.html**.

[jac24] JackTrip, 2024. JackTrip Labs, Inc. URL: **https://www.jacktrip.com/** [cited August 22, 2024].

[JHV+17] S. James, C. Hope, L. Vickery, A. Wyatt, B. Carey, X. Fu, and G. Hajdu. Establishing connectivity between the existing networked music notation packages quintet.net, decibel scoreplayer and maxscore. In *International Conference on Technologies for Music Notation and Representation*, A Coruña, 2017.

[Jol00] Jean-Clement Jollet. *Jeux de Rythmes et Jeux de Clés*, volume 1. Billaudot, Paris, 2000.

[KB18] David Kim-Boyle. Reframing the listening experience through the projected score. In Sandeep Bhagwati and Jean Bresson, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, pages 181–185, Montreal, Canada, 2018. Concordia University. URL: **https://www.tenor-conference.org/proceedings.html**.

[Kim18] Jin-Ah Kim. Thoughts on sandeep bhagwati's comprovisation – concepts and practices. *Circuit – Musiques Contemporaines*, 28(1), 2018. English

Supplements (online). URL: https://revuecircuit.ca/articles/28_1/51.04-thoughts-on-sandeep-bhagwatis-comprovisation/.

[Kli18] Christian Klinkenberg. A combination of graphic notation and microtonal pitch notation in the video score of the opera "the cross of the engaged". In Sandeep Bhagwati and Jean Bresson, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, pages 186–192, Montreal, Canada, 2018. Concordia University. URL: https://www.tenor-conference.org/proceedings.html.

[kro03] Kronos quartet – visual music, 2003. URL: https://kronosquartet.org/concerts/details/1910.

[LCX14] Launch control xl programmer's reference, 2014. URL: https://global.novationmusic.com/support/downloads/launch-control-xl-programmers-reference-guide.

[Lee17] Cheng Lee. Multimedia performance installation with virtual reality. In *Proceedings of the 43rd International Computer Music Conference*, pages 347–350, Shanghai, 2017. URL: http://www.icmc2017.com/en/download.html.

[LK04] Nelson Lago and Fabio Kon. The quest for low latency. In *Proceedings of the International Computer Music Conference*, Miami, 2004.

[LK10] Andreas Lehmann and Reinhard Kopiez. Can expert listeners hear if a piece is improvised or composed? In *Proceedings of the 11th International Conference on Music Perception and Cognition*, pages 577–580, Seattle, 2010.

[Lop09] Filipe Lopes. Õdaiko: Real time score generator. *Master's thesis. The Hague*, 2009.

[Lop24] Filipe Lopes. Filipe lopes, 2024. URL: https://www.filipelopes.net/ [cited August 22, 2024].

[Lou14] Pedro Louzeiro. Pequenas peças para flauta, piano e guitarras. 9:1–47, 2014.

[Lou17a] Pedro Louzeiro. Comprovisador – about, 2017. URL: https://comprovisador.wordpress.com/.

[Lou17b] Pedro Louzeiro. The Comprovisador's real-time notation interface. In *Proceedings of the 13th International Symposium on Computer Music Multidisciplinary Research*, pages 340–351, Matosinhos, 2017. URL: http://cmmr2017.inesctec.pt/proceedings.

[Lou17c] Pedro Louzeiro. Improving sight-reading skills in a microtonal context using a real-time composition tool, July 2017. Presented at The 2nd European Saxophone Congress (EurSax'17).

[Lou17d] Pedro Louzeiro. Mediating a comprovisation performance: the Comprovisador's control interface. In *Proceedings of the 43rd International Computer Music Conference*, pages 362–367, Shanghai, 2017. URL: **http://www.icmc2017.com/en/download.html**.

[Lou17e] Pedro Louzeiro. Real-time compositional procedures for mediated soloist-ensemble interaction: The Comprovisador. In Octavio A. Agustín-Aquino, Emilio Lluis-Puebla, and Mariana Montiel, editors, *Mathematics and Computation in Music. MCM 2017. Lecture Notes in Computer Science*, volume 10527, pages 117–131. Springer International Publishing, Cham, 2017. **doi:10.1007/978-3-319-71827-9_10**.

[Lou17f] Pedro Louzeiro. Synchronizing to visual cues in a networked, real-time notation environment – Comprovisador, November 2017. Presented at the International Conference Electroacoustic Winds 2017: SYNCHRESIS – Audio Vision Tales.

[Lou18a] Pedro Louzeiro. Aspectos e potencialidades de um sistema de composição e notação em tempo-real – Comprovisador, April 2018. Presented at Conferências em Música e Musicologia do Centro de Estudos de Sociologia e Estética Musical – Pólo Universidade de Évora.

[Lou18b] Pedro Louzeiro. Aspects and potentialities of a real-time composition and notation system – Comprovisador, November 2018. Presented at Nova Contemporary Music Meeting – International Conference.

[Lou18c] Pedro Louzeiro. The Comprovisador's real-time notation interface (extended version). In Mitsuko Aramaki, Matthew E. P. Davies, Richard Kronland-Martinet, and Sølvi Ystad, editors, *Music Technology with Swing. CMMR 2017. Lecture Notes in Computer Science*, volume 11265, pages 489–508. Springer International Publishing, Cham, 2018. **doi:10.1007/978-3-030-01692-0_33**.

[Lou18d] Pedro Louzeiro. Improving sight-reading skills through dynamic notation – the case of Comprovisador. In Sandeep Bhagwati and Jean Bresson, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, pages 55–61, Montreal, Canada, 2018. Concordia University. URL: **https://www.tenor-conference.org/proceedings.html**.

[Lou19a] Pedro Louzeiro. Distributed scores and audio on mobile devices in the music for a multidisciplinary performance. In *Proceedings of the 14th International Symposium on Computer Music Multidisciplinary Research*, pages 401–412, Marseille, 2019. URL: **https://cmmr2019.prism.cnrs.fr/downloads.html**.

[Lou19b] Pedro Louzeiro. Música distribuída, may 2019. Presented at Semana da Composição 2019 – Escola Superior de Música de Lisboa.

[Lou19c] Pedro Louzeiro. Synchronizing to visual cues in a networked, real-time notation environment – Comprovisador. In Isabel Soveral and Fátima Pombo, editors, *Synchresis – Audio Vision Tales*, pages 165–172. UA Editora, Aveiro, Portugal, 2019. URL: **http://uaeditora.marka.pt/eBook/Synchresis-audio-vision-tales/9789727895915**.

[Lou21] Pedro Louzeiro. Distributed scores and audio on mobile devices in the music for a multidisciplinary performance. In Richard Kronland-Martinet, Sølvi Ystad, and Mitsuko Aramaki, editors, *Perception, Representations, Image, Sound, Music*, pages 329–344, Cham, 2021. Springer International Publishing.

[Mac16] Robin Maconie. *Other Planets: The Complete Works of Karlheinz Stockhausen 1950–2007*. Rowman & Littlefield Publishers, Lanham, 2016. URL: **https://books.google.pt/books?id=2gp5DQAAQBAJ**.

[Mai13] Joshua B Mailman. Improvising synesthesia: Comprovisation of generative graphics and music. *Leonardo Electronic Almanac*, 19(3):352–384, 2013. URL: **http://journals.gold.ac.uk/index.php/lea/article/view/88**.

[Man13] P. Manning. *Electronic and Computer Music*. Oxford University Press, 2013. URL: **https://books.google.pt/books?id=ryet1i-8OlYC**.

[Mat19] Benjamin Matuszewski. Soundworks - a framework for networked music systems on the web – state of affairs and new developments. In *Proceedings of the Web Audio Conference (WAC) 2019*, pages 65–70, Trondheim, Norway, 2019. URL: **https://hal.archives-ouvertes.fr/hal-02387783**.

[McA99] Stephen McAdams. Perspectives on the contribution of timbre to musical structure. *Computer Music Journal*, 23(3):85–102, 1999. URL: **http://www.jstor.org/stable/3681242**.

[Mei09] Gustav Meier. *The score, the orchestra, and the conductor*. Oxford University Press, New York, 2009. URL: **https://books.google.pt/books?id=-SZ50-TGmnIC**.

[MOL+19] Romain Michon, Yann Orlarey, Stéphane Letz, Dominique Fober, and Catinca Dumitrascu. Mobile music with the faust programming language. In *Proceedings of the 14th International Symposium on Computer Music Multidisciplinary Research*, pages 371–382, Marseille, 2019. URL: **https://cmmr2019.prism.cnrs.fr/downloads.html**.

[Mor] Lawrence D. “Butch” Morris. Conduction.us. URL: **http://www.conduction.us/index.html** [cited 2019/01/11].

[Mor15] Luigina Mortari. Reflectivity in research practice: An overview of different perspectives. *International Journal of Qualitative Methods*, 14(5), 2015. **doi:10.1177/1609406915618045**.

[MSB19] Benjamin Matuszewski, Norbert Schnell, and Frédéric Bevilacqua. Interaction topologies in mobile-based situated networked music systems. *Wireless Communications and Mobile Computing*, 2019:68–76, 2019. **doi:10.1155/2019/9142490**.

[Nas19] Joaquim Nascimento. O contributo do Comprovisador na leitura musical à primeira vista, 2019. Dissertacção de Mestrado.

[Nev06] Ben Nevile. Networking: Max talking to max, 2006. URL: **https://cycling74.com/tutorials/networking-max-talking-to-max** [cited 2017/04/13].

[Nym99] Michael Nyman. *Experimental Music: Cage and Beyond*. Music in the 20th century. Cambridge University Press, second edition, 1999. URL: **https://books.google.pt/books?id=QEBzEhzAkYwC**.

[oxf19] English oxford living dictionaries, 2019. Oxford University Press. URL: **https://en.oxforddictionaries.com** [cited August 22, 2024].

[p5j] p5.js. Processing Foundation. URL: **https://p5js.org** [cited 2019-05-06].

[Pen11] Rui Penha. peripatoi, 2011. URL: **https://ruipenha.pt/?s=peripatoi** [cited August 22, 2024].

[Pic97] Richard Picking. Reading music from screens vs paper. *Behaviour & Information Technology*, 16(2):72–78, 1997. **doi:10.1080/014492997119914**.

[Poi15] Robin Le Poidevin. The experience and perception of time, 2015. URL: **https://plato.stanford.edu/entries/time-experience/** [cited 2018/04/12].

[Por19] Henrique Portovedo. HASGS: Generating visual and graphical feedback. In Isabel Soveral and Fátima Pombo, editors, *Synchresis – Audio Vision Tales*, pages 14–21. UA Editora, Aveiro, Portugal, 2019. URL: **http://uaeditora.marka.pt/eBook/Synchresis-audio-vision-tales/9789727895915**.

[Puc] Miller Puckette. Pure data. URL: **https://puredata.info** [cited 2019/09/19].

[PV90] António Pinho Vargas. Con(di)vergências: as relações entre o jazz e a música contemporânea. *Colóquio/Artes – Gulbenkian*, (85), 06 1990. URL: **https://www.researchgate.net/**

publication/342184014_Condivergencias_as_relacoes_entre_o_jazz_e_a_musica_contemporanea_1990.

[PZS+] M. Puckette, D. Zicarelli, R. Sussman, J. K. Clayton, J. Bernstein, B. Nevile, T. Place, D. Grosse, R. Dudas, E. Jourdan, M. Lee, and A. Schabtach. Max 7: Documentation. URL: https://docs.cycling74.com/max7/ [cited 2017/01/31].

[qua23] Quantization (physics), 2023. Wikipedia. URL: https://en.wikipedia.org/wiki/Quantization_(physics) [cited August 22, 2024].

[Ras79] R. A. Rasch. Synchronization in performed ensemble music. *Acta Acustica united with Acustica*, 43(2):121–131, 1979. URL: https://www.ingentaconnect.com/content/dav/aaua/1979/00000043/00000002/art00005.

[Rat13] Ben Ratliff. Butch morris dies at 65; creator of 'conduction', 2013. The New York Times (online). URL: https://www.nytimes.com/2013/01/30/arts/music/butch-morris-dies-at-65-creator-of-conduction.html.

[RCAS16] Cristina Rottondi, Chris Chafe, Claudio Allocchio, and Augusto Sarti. An overview on networked music performance technologies. *IEEE Access*, 4:8823–8843, 2016. URL: https://ieeexplore.ieee.org/abstract/document/7769205.

[RM16] Simon Rose and Raymond MacDonald. Improvisation as real-time composition. In Dave Collins, editor, *The Act of Musical Composition - Studies in the Creative Process*, pages 187–214. Routledge, London, 2016.

[Roa85] Curtis Roads. Improvisation with george lewis. In Curtis Roads, editor, *Composers and the computer*. W. Kaufmann, Los Altos, CA, 1985.

[Row93] Robert Rowe. *Interactive Music Systems: Machine Listening and Composing*. MIT Press, 1993. URL: https://wp.nyu.edu/robert_rowe/text/interactive-music-systems-1993/.

[Row01] Robert Rowe. *Machine Musicianship*. MIT Press, Cambridge, Mass., 2001. URL: https://mitpress.mit.edu/books/machine-musicianship.

[San18] Giovanni Santini. Linear (live-generated interface and notation environment in augmented reality). In Sandeep Bhagwati and Jean Bresson, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, pages 33–42, Montreal, Canada, 2018. Concordia University. URL: https://www.tenor-conference.org/proceedings.html.

[Sar96] Ed Sarath. A new look at improvisation. *Journal of Music Theory*, 40(1):1–38, 1996. **doi:10.2307/843921**.

[Sch50] Arnold Schoenberg. *Style and Idea*. Philosophical Library, New York, 1950. URL: **https://books.google.pt/books?id=lKcnAAAAMAAJ**.

[SCJT16] Evan Strasnick, Austin Chambers, Liu Jiang, and Xiaonan Tong. Pianolens: An augmented reality interface for piano instruction, 2016. Stanford University – other projects. URL: **https://cs.stanford.edu/people/estrasni/otherprojects/pianolens.html** [cited 2019-02-06].

[Sha15] Seth Shafer. VizScore: An on-screen notation delivery system for live performance. In *Proceedings of the International Computer Music Conference*, pages 142–145, Denton, TX, 2015. URL: **http://hdl.handle.net/2027/spo.bbp2372.2015.027**.

[Sha17a] Seth Shafer. Performer action modeling in real-time notation. In Helena Lopez Palma, Mike Solomon, Emiliana Tucci, and Carmen Lage, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation*, pages 117–123, A Coruña, 2017. Universidade da Coruña. URL: **https://www.tenor-conference.org/proceedings.html**.

[Sha17b] Seth Shafer. *Recent Approaches to Real-Time Notation*. PhD thesis, University of North Texas, 2017.

[Ska18] R. Lyle Skains. Creative practice as research: Discourse on methodology. *Media Practice and Education*, 19(1):82–97, 2018. **doi:10.1080/14682753.2017.1362175**.

[Slo74] John Sloboda. The eye-hand span - an approach to the study of sight reading. *Psychology of Music*, 2(2):4–10, 1974. **doi:10.1177/030573567422001**.

[Slo76] John A. Sloboda. Visual perception of musical notation: Registering pitch symbols in memory. *Quarterly Journal of Experimental Psychology*, 28(1):1–16, 1976. **doi:10.1080/14640747608400532**.

[Smi14a] Ryan Ross Smith. Animated notation dot com, 2014. URL: **http://animatednotation.com** [cited 2017/07/01].

[Smi14b] Ryan Ross Smith. Ryan ross smith – scores, 2014. URL: **http://ryanrosssmith.com/scores** [cited 2020-08-12].

[sou] Soundpainting – the art of live composition. URL: **http://www.soundpainting.com/** [cited 2018/12/10].

[SRS+09] Norbert Schnell, Axel Roebel, Diemo Schwarz, Geoffroy Peeters, and Riccardo Borghesi. Mubu & friends — assembling tools for content based real-time interactive audio processing in max/msp. In *Proceedings of the International Computer Music Conference (ICMC 2009)*, Montréal, 2009.

[Sta09] Thomas Taylor Stanley. *Butch Morris and the Art of Conduction*. PhD thesis, University of Maryland, 2009.

[Sto68] Karlheinz Stockhausen. *Aus den sieben Tagen*. Universal Edition, Vienna, 1968.

[the95] Thesaurus.com, 1995. URL: **https://www.thesaurus.com** [cited August 22, 2024].

[Tho06] W. Thompson. *Soundpainting: The Art of Live Composition. Workbook*. Number vol. 1. W. Thompson, 2006. URL: **https://books.google.pt/books?id=m6BMnQEACAAJ**.

[Tho19] Philip Thomas. Philip thomas, 2019. URL: **https://www.philip-thomas.co.uk** [cited 2019-02-03].

[ton] tone.js. URL: **https://tonejs.github.io** [cited 2019-05-06].

[Vic10] Lindsay Vickery. Mobile scores and click-tracks: Teaching old dogs. In *Proceedings of the Australasian Computer Music Conference*, pages 63–70, Canberra, 2010. URL: **https://www.lindsayvickery.com/research.html**.

[WDH18] David Whyte, Nick Didkovsky, and Stefan Hutzler. Zero waste: Mapping the evolution of the iterative sight-reading of a piano score. *Music Theory Spectrum*, 40(2):302–313, 2018. **doi:10.1093/mts/mty019**.

[Wei05] Gil Weinberg. Interconnected musical networks: Toward a theoretical framework. *Computer Music Journal*, 29(2):23–39, 2005. **doi:10.1162/0148926054094350**.

[wik] User interface design. URL: **https://en.wikipedia.org/wiki/User_interface_design** [cited 2015/08/20].

[Wil13] Chris Wilson. A tale of two clocks - scheduling web audio with precision, 2013. URL: **https://www.html5rocks.com/en/tutorials/audio/scheduling/** [cited 2019-07-19].

[Win04] Gerhard E. Winkler. The real-time score. a missing-link in computer-music performance. In *Proceedings of Sound and Music Computing*, Paris, 2004. URL: **http://www.smcnetwork.org/resources/smc2004**.

[Wol76] Thomas Wolf. A cognitive model of musical sight-reading. *Journal of Psycholinguistic Research*, 5(2):143–171, Apr 1976. doi:10.1007/BF01067255.

[Wri05] Brenda Wristen. Cognition and motor execution in piano sight-reading: A review of literature. *Update: Applications of Research in Music Education*, 24(1):44–56, 2005. doi:10.1177/87551233050240010106.

[WUF97] Andrew J. Waters, Geoffrey Underwood, and John M. Findlay. Studying expertise in music reading: Use of a pattern-matching paradigm. *Perception & Psychophysics*, 59(4):477–488, Jun 1997. doi:10.3758/BF03211857.

[xeb18] xebra.js, 2018. Cycling74. URL: https://cycling74.github.io/xebra.js/index.html [cited 2019-05-06].

[Xen92] Iannis Xenakis. *Formalized Music: Thought and Mathematics in Composition*. Harmonologia series. Pendragon Press, 1992. URL: https://books.google.pt/books?id=y6lL3I0vmMwC.

[YH17] Masao YOKOYAMA and Haruka HIRAYAMA. Improvised multimedia contents by visualization and sonification of brainwaves. *Reports of the Technical Conference of the Institute of Image Electronics Engineers of Japan*, 16.04:56–58, 2017. doi:10.11371/wiieej.16.04.0_56.

[ZA18] Slavko Zagorac and Patricia Alessandrini. ZScore: A distributed system for integrated mixed music composition and performance. In Sandeep Bhagwati and Jean Bresson, editors, *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, pages 62–70, Montreal, Canada, 2018. Concordia University. URL: https://www.tenor-conference.org/proceedings.html.

[Zhu14] Katie Zhukov. Exploring advanced piano students' approaches to sight-reading. *International Journal of Music Education*, 32(4):487–498, 10 2014. doi:10.1177/0255761413517038.

UNIVERSIDADE DE ÉVORA
INSTITUTO DE INVESTIGAÇÃO
E FORMAÇÃO AVANÇADA

Contactos:

Universidade de Évora

Instituto de Investigação e Formação Avançada — IIFA

Palácio do Vimioso | Largo Marquês de Marialva, Apart. 94

7002 - 554 Évora | Portugal

Tel: (+351) 266 706 581

Fax: (+351) 266 744 677

email: iifa@uevora.pt