i-Berlioz: Interactive Computer-Aided Orchestration with Temporal Control

Eduardo R. Miranda[1], Aurélien Antoine[1], Jean-Michaël Celerier[2], and Myriam Desainte-Catherine[2]

[1] Interdisciplinary Centre for Computer Music Research (ICCMR), Plymouth University, UK
[2] Laboratoire Bordelais de Recherche en Informatique (LaBRI), University of Bordeaux, France
eduardo.miranda@plymouth.ac.uk
aurelien.antoine@postgrad.plymouth.ac.uk
jean-michael.celerier@labri.fr
myriam.desainte-catherine@labri.fr

Abstract. This paper introduces the beta version of i-Berlioz, an interactive CAO system that generates orchestrations from verbal timbre descriptors. This system relies on machine learning models for timbre classification and generative orchestration, and methods and tools to write and execute interactive scenarios. The text details the two main parts of i-Berlioz: its generative and exploratory engines, illustrated with an example of i-Berlioz operational processes. Such a system would ultimately aid the composition of musical form based on timbre.

Keywords. Computer-Aided Orchestration, Interaction, Generative Orchestration, Machine Learning.

1 Introduction

Computers have been programmed to make music as early as the beginning of the 1950's when the CSIR Mk1 computer was programmed in Australia to play back popular musical melodies [1]. The piece Illiac Suite for String Quartet, composed in late 1950's by Lejaren Hiller, in collaboration with mathematician Leonard Isaacson, at the University of Illinois, USA, is often cited as a pioneering piece of algorithmic computer music; that is, a piece involving materials composed by a computer. Its fourth movement, for instance, was generated with a probabilistic Markov chain [2]. A few years later, Hiller collaborated with Robert Baker to develop a piece of software for composition named MUsic Simulator Interpreter for COmpositional Procedures, or MUSICOMP [3]. MUSICOMP probably is the first system ever developed for computer-aided composition: "… it is a facilitator program. It presents no specific compositional logic itself, but it is capable of being used with nearly any logic supplied by the user." [4, p.1].

The burgeoning field of computer-aided composition has advanced considerably since MUSICOMP [5]–[10]. The computer has become ubiquitous in many aspects of mu-

sical composition. Nowadays musicians have access to a variety of software tools for composition, from user-friendly programming languages [11]–[13] and AI-based generators of musical ideas [14]–[16], to systems for generating and managing musical events interactively in real-time [17], [18].

Whereas the great majority of computer-aided composition systems currently available provide valuable tools for processing music in terms of pitches, rhythms, tempo and loudness, there is a generalized lack of tools for processing orchestration: that is, computer-aided processing of multi-instrumental properties and creation of unique timbres using acoustic instruments. Historically, this deficiency most probably is a legacy of the highly popular MIDI (Musical Instrument Digital Interface) communication protocol [19]. Originally developed in the 1980s to connect digital synthesizers (e.g., to control various synthesisers with one single keyboard controller), MIDI was quickly adopted by the computer music research community. Rather than representing musical sounds directly, MIDI encodes musical notes in terms of their pitches, durations, loudness, and labels indicating which instruments of the MIDI sound-producing device should play them. Although the musical possibilities of MIDI processing are vast, MIDI does not encode sounds per se, which renders it unsuitable for processing timbre.

We are interested in developing technology for computer-aided orchestration, or CAO. Despite the existence of a significant and diverse body of research into timbre and its defining acoustic features [20]–[25], there has been relatively less research into orchestral timbre emerging from the combination of various musical instruments playing simultaneously [26], [27]. Our research is aimed at furthering our understanding of orchestral timbre and building systems for CAO informed by such understanding. This paper focuses on the latter: it introduces the beta version of i-Berlioz, an interactive CAO system that generates orchestrations from verbal timbre descriptors.

Currently, i-Berlioz is capable of processing five timbre descriptors: breathiness, brightness, dullness, roughness and warmth. Given a timbre descriptor, the system generates clusters of musical notes with the instruments that would produce the required timbre, plus indications of how the notes should be played (e.g., usage of a specific bowing style for a string instrument). Moreover, i-Berlioz is able to generate sequences of such clusters for transitions between one timbral quality to another; for example, from very bright to less bright, or from dull to warm. The user can listen to the orchestrations and see their respective music notation. The resulting notation can be saved into a file, which can be edited by most music notation software.

One important characteristic of i-Berlioz is its ability to support interactive design of musical form based on timbre. The system is able to hold multiple solutions for specific timbral targets on a time line. And it includes a conditional branching mechanism, which enables composers to specify different branching options and explore them interactively. The system supports the specification of orchestration strategies in a hierarchical fashion and the execution of various solutions can be inspected on the fly, while maintaining a global consistency.

i-Berlioz combines the work on analysis and classification of timbre within orchestral audio and machine learning developed at Plymouth University's ICCMR [28]–[30] and the research into methods and tools to write and execute interactive scenarios

developed at University of Bordeaux's LaBRI, conducted under the OSSIA (Open Scenario System for Interactive Applications) project [17], [31].
By way of related work, we cite the system Orchids, developed at IRCAM, Paris [32]. Orchids is aimed at producing orchestrations to imitate a given target sound; e.g., given the sound of a thunder the system would produce suggestions for imitating the timbre of a thunder with the orchestra. The system maintains a database of instrumental sounds. It extracts spectral information from the target sound and from each instrument of its database. This information is feed into a combinatorial search algorithm, which searches for groups of instruments whose combined spectrum matches that of the target sound [33]. Orchids outputs dozens of solutions that satisfy the matching criteria, but these solutions can still sound very different from each other. This requires the user to listen to dozens of solutions to select one. This process can be very tedious, ineffective and time-consuming, in particular when the user has a particular sound color, or a perceptual sound quality, in mind. We believe this pitfall can be addressed by designing a constraint-based filtering mechanism to narrow the solutions to a more specific sound quality. Hence the use of verbal sound description might improve this problem considerably.
The remainder of this paper is structured as follows. The next section presents an overview of the system's architecture and briefly describes how orchestrations are produced from a given sound descriptor. In order to ascertain that the system outputs an orchestration that satisfies a required characteristic it needs a method for categorizing trials, which will be detailed next. Then, we present the system's ability to generate sequences and transitions between timbral qualities.

2 System's Overview

The system comprises two main modules referred to as generative and exploratory engines, respectively. The functioning generative module is depicted in the flowchart shown in Fig. 1.
The system holds a comprehensive Instrument Database, which contains audio samples of single notes played by orchestral instruments. There are various versions for each note, with different playing techniques and dynamics. Given a Timbre Description, the system generates candidate solutions by assembling combination of notes from the database. With the Timbre Description the user can also provide additional information to constraint the combinatorial search space, such as a list of required instruments and number of notes. An example of a description could be: { bright, strings, 4 }, which means, a bright timbre using 4 strings instruments.

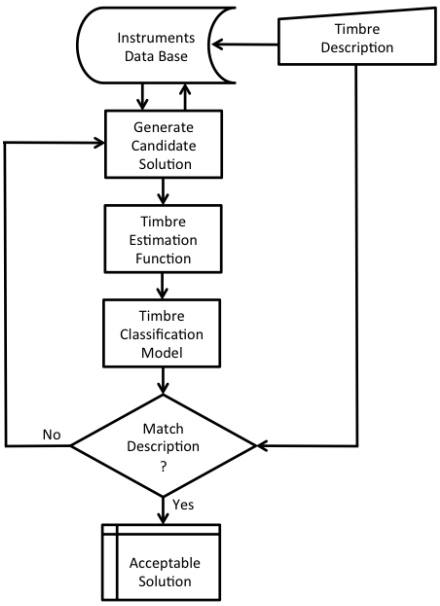

Fig. 1. Flowchart of the generative engine.

The Generate Candidate Solution module generates an audio file, which is rendered using the samples contained in the database. In order to estimate the perceptual quality of the candidate solution, the Timbre Estimation Function module analyses the spectrum of candidate audio file and the results of the analysis are relayed to the Timbre Classification Model, which establishes whether the candidate audio processes the required timbre characteristics or not. This process is repeated until a candidate solution matches the requirement.

The exploratory engine is depicted in Fig. 2. It is based on an extension plug-in to the *i-score* software which integrates the generative engine with the interaction scoring capabilities of i-score, described more in details in section 4. The general usage process is as follows: first, the user creates a score in which generative processes can be positioned in time. Such a score can contain branches which are specified graphically; these branch allows behaviors such as: “after the execution of a bright-to-warm transition, perform a warm-to-rough transition in one case and a warm-to-dull transition in another”. Then, during playback, cases can be selected according to external controls in real-time. Such controls can come from OSC or MIDI commands, in order to leverage external controllers to direct the choices. The chords produced by the generative engine are passed through the audio output ports of the process for direct listening. Besides, each i-Berlioz process provides the list of chords it played during the playback in a graphical view, so that they can be leveraged by the composer.

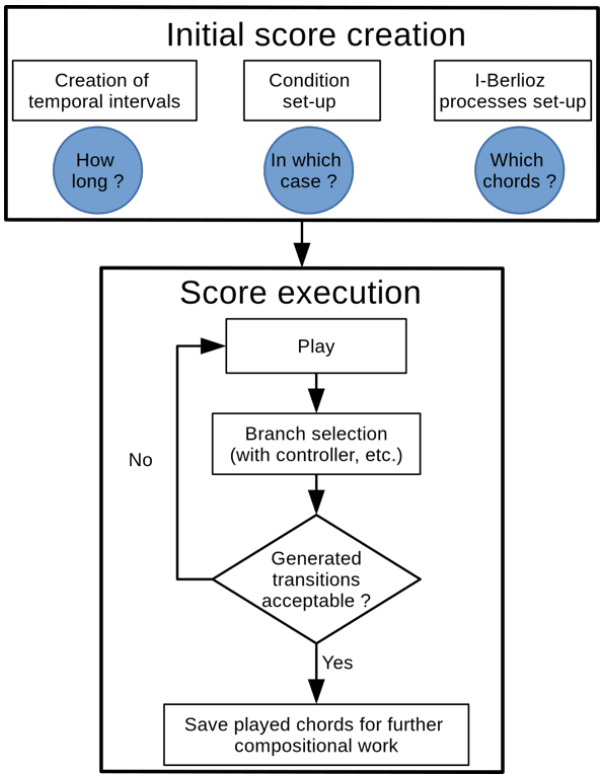

Fig. 2. Flowchart of the exploratory engine.

3 Timbre Descriptors and Classification Model

The system supports five descriptors of timbre, namely: breathiness, brightness, dullness, roughness and warmth. These descriptors can be retrieved from audio sources by performing various analyses methods, which are detailed below:

- Breathiness: in order to measure the level of breathiness of an audio file, we calculate the amplitude of the fundamental frequency against the noise content, and also the spectral slope [34], [35]. The larger the ratio between the amplitude of the fundamental and the noise content, the breathier the sound.
- Brightness: the acoustic correlates for the attribute brightness are the spectral centroid and the fundamental frequency [36], [37]. The higher the spectral centroid and the fundamental frequency, the brighter the sound.
- Dullness: as with brightness, in order to measure the dullness of a sound we need to calculate its spectral centroid. However, in this case, the lower the value of the spectral centroid the duller is the respective sound [38].

- Roughness: is measured by calculating the distance between adjacent partials in critical bandwidths and the energy above the 6th harmonic. A rough sound is characterized be a short distance between critical bandwidths; theoretically, the shorter the distance the rougher the sound [39]–[41].
- Warmth: the warmth of a sound is measured by calculating its spectral centroid and retrieving the energy of its first three harmonics. A low spectral centroid and a high energy in the first three harmonics suggest that the sound is warm [42].

In order to establish whether a solution candidate possesses the required characteristics or not we developed a method to automatically classify audio samples according to timbre description as described above. The difficulty here is that there is no agreed metrics for classifying orchestral audio samples in terms of timbre properties. Therefore, we developed our own comparative scale for each descriptor as follows: 250 audio recordings of various different well-known orchestral pieces have been analyzed; e.g., Beethoven's *5th Symphony*, Vivaldi's *Four Seasons*, Debussy's *Suite Bergamasque*, and Saint-Saëns' *Carnival of the Anim*als, to cite but four. Each of these pieces was sliced five times into audio samples lasting for 1, 2, 3, 4, and 5 seconds, respectively. Here, the analysis of longer audio samples would not provide accurate values as some acoustic features are time related, therefore, it is essential to split them into short audio samples for analysis purposes. Furthermore, the different lengths have been chosen to match the durations of outputs typically generated by computer-aided orchestration systems. This data gathering resulted in performing timbre estimations onto 236 632 audio files, thus, compiling a dataset composed of 236 632 values for each descriptor. The analysis of this large number of samples enabled us to establish a scale for each attribute, and, thus, be able to normalize the data among the five timbre descriptors. Furthermore, the scale for each timbre attribute is continually calibrated as new audio files are analyzed.

The development of a comparative scale for the different descriptors enabled us to input timbre values into a machine-learning algorithm, using a Support Vector Machine (SVM) supervised learning model [43]. SVM models try to find the separation between the different categories with a gap that is as wide as possible to create a delimited space for each category. Then, when a new value is presented, SVM models estimate the space in which the value sits, thus predicting the category it belongs to. SVM methods have been successfully applied in various applications, such as face detection [44] and speaker recognition [45] to name but two.

Supervised learning algorithms are dependent of a labelled training dataset. Therefore, the initial step was to create a set of examples that will then be used to train the SVM classification model. Here, examples consisted of calculated timbral values of short audio files as input data, and their dominant perceptual quality, represented by verbal attributes, as the desired output category. The training samples have been selected from the large training dataset created for the comparative scale mentioned previously. Here, the 250 orchestral audio files that have scored the highest values for each

verbal attribute have been selected and manually labeled by the authors with their corresponding attribute. For instance, the 250 samples with the highest values for the attribute brightness have been chosen and labeled 'brightness'. In total, the corpus training contained 1,250 samples labeled accordingly.

The SVM algorithm has been implemented using the svm.SVC function, which is a SVM method for classification task, taken from the Scikit-Learn v0.18 library [46], with parameters kernel type = RBF (for Radial Basis Function), RBF kernel coefficient $\gamma = 0.2$, and penalty parameter C=1.0. Fig. 3 shows a normalized confusion matrix created to estimate the performance of the classification model generated by the SVM algorithm. Here, the training samples consisted of 90% of the training corpus (1125 samples), and 10% of the training corpus (125 samples) were selected as testing samples. Using this training dataset, the svm.SVC function produced a success rate of 0.976, which means that the classification model predicted the correct verbal attribute 97.6% of the time. Users have the ability to calibrate the classification models by listening and labelling a selection of audio samples, which are then processed by the SVM algorithm. This method, inspired by the reinforcement learning techniques, allows the users to input their own perception levels into the learning process, thus improving the accuracy of the suggested classifications.

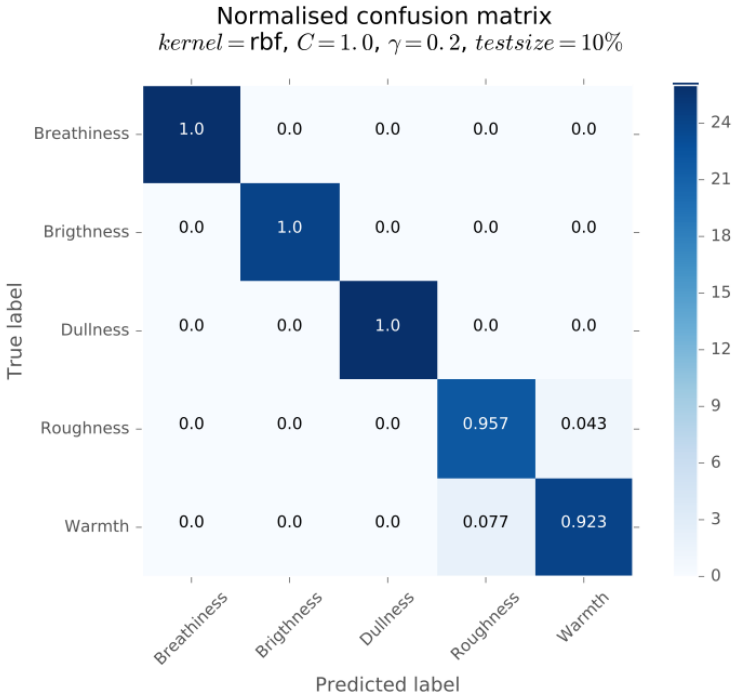

Fig. 3. Normalized confusion matrix of the SVM classification model created from rescaled training dataset, with test size = 10% (1125 training samples, 125 testing samples).

3.1 Solution Candidate Examples

This section illustrates the definition and generation of a solution candidate. As mentioned previously, i-Berlioz utilizes timbre descriptors as output decision parameter in the generative engine. Other parameters can be manipulated by the users to define their musical ideas and guide the generative processes. As additional constraints, the user can specify the group of instruments, its organization, and the types of interval between the sets of notes of a candidate solution; for instance, a type of chord such as major, minor, whole tones, semi-tone clusters, and so on. These parameters can also be randomly assigned should the user wish so.

For ease of description and discussion, the following examples are constrained to string instruments organized as a standard string quartet (Contrabass, Violoncello, Viola, and Violin). The first example was defined with the following parameters: { bright, strings, 4, major }. Here, the system was instructed to generate a candidate possessing a bright timbre and composed of a major chord with 4 notes produced by strings instruments. A candidate solution matching these parameters is shown in Fig. 4a, while Fig. 4b shows another candidate solution, but it does not match the timbre descriptor 'bright'. Fig. 5 presents a second example with parameters { warm, strings, 4, major }, with a candidate matching the timbre descriptor constraint and one not producing a combination of notes presenting characteristics of a warm sound. With these two different examples, one can observe that with similar parameters, the sonic characteristics of generated instrument combinations can be very different, highlighting the large search space of instruments combination, one of the challenges of computer-aided orchestration.

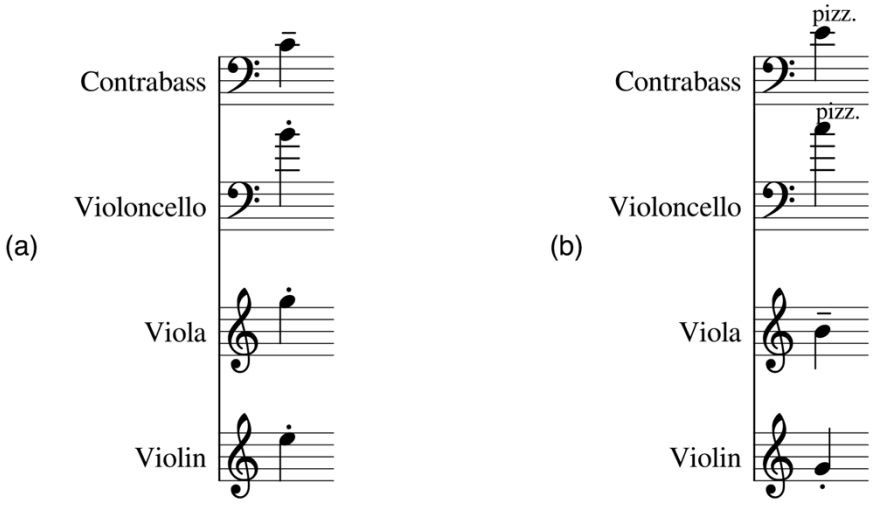

Fig. 4. (a) presents a candidate solution composed of a major chord with 4 notes produced by string instruments matching the descriptor 'brightness' while (b) shows a candidate solution generated with same parameters but not matching the descriptor.

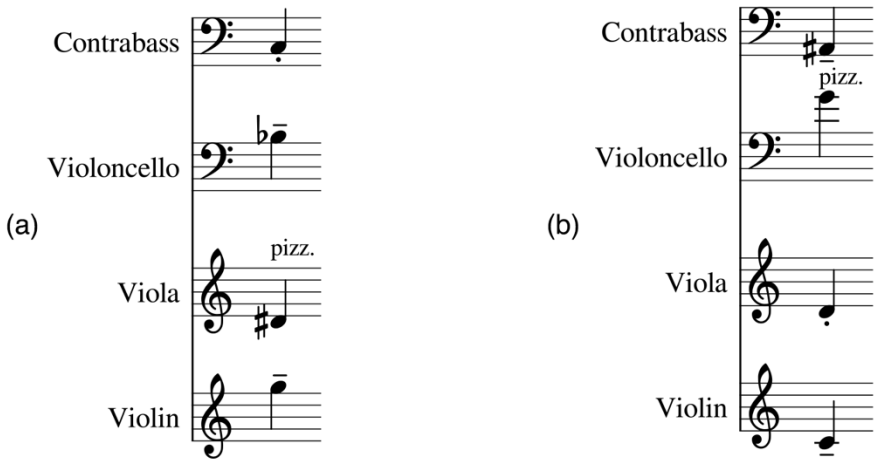

Fig. 5. (a) presents a candidate solution composed of a minor chord with 4 notes produced by string instruments matching the descriptor 'warmth' while (b) shows a candidate solution generated with same parameters but not matching the descriptor.

4 Sequencing and Transitions

At the basis of the scheme for making transitions introduced above is a sophisticated engine for writing and executing temporal constraints, based on a formalism for the authoring of interactive scores developed during the course of the OSSIA research project at LaBRI [17]. This formalism manifests itself mainly in the *i-score* software, which is both a visual editor and an execution engine for such interactive scores.

The system is based on the following building blocks: temporal interval, temporal condition, instantaneous condition, and process. Temporal intervals are organized in directed acyclic graphs, with instantaneous conditions being the vertices of such graphs, and the direction of time being the direction of the graph. Temporal conditions allow to trigger groups of instantaneous conditions, which in turn stops and starts previous and following intervals. During the execution of an interval, processes can take place: these can be automations, sound files and effects, scripts, and so on.

The elements of the OSSIA model is shown in Fig. 6. The execution takes place as follows: the interval A runs for a fixed duration. When it ends, an instantaneous condition is evaluated: if it is false, the branch with B will not run. Otherwise, after some time and the playback of an audio file, the execution of B ends up in a flexible area centered on a temporal condition. If there is an interaction, B stops and D starts. Else, D starts when the maximum bound of B is reached. Like after A, a condition chooses whether G will execute. If G executes, an automation and another computation are both executed for a fixed duration. Meanwhile, C started executing at the same time than B. C is waiting for an interaction, without any time-out. When the interaction happens, the two conditions following C are evaluated; the truth value of each will imply the execution of E and F. Finally, when H executes, a hierarchical sub-score starts.

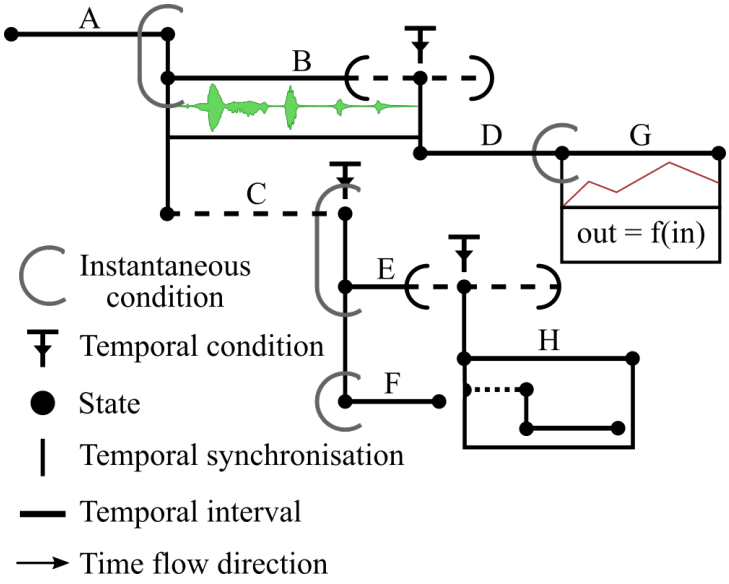

Fig. 6. Elements of the OSSIA model.

The compositional process proposed in this paper only requires the use of instantaneous conditions in order to choose a specific sequence amongst the possible ones, and of the i-Berlioz process. This process leverages the generation engine to create and listen to chords according to a specific transition. The following parameters can be specified by the composer:

- The start and end target descriptors
- The duration of each orchestration cluster, or chord
- The number of instruments
- The set of instrument available for generating an orchestration cluster

When the process runs, at regular intervals fixed by the requested chord length, the generation engine receives a chord query. When a correct chord has been generated, its sound file is reported to the i-Berlioz process which plays it. Fade-outs are applied to the generated sound files to prevent clicks in the sound.

In order to perform a cross-fade between the start and end descriptors, the following algorithm is applied:

- On the first tick, generate ten chords and use their attribute value, for instance ‘brightness’ to find a minimal and maximal expected value for the rest of the generation.
- Then, at every tick, generate ten other chords and find the one with the attribute that matches the expected progression the closest over all chords generated, so that the first generated chord has a maximal attribute, and the chord generated at the middle of the process has a minimal attribute.
- For the second half of the process, perform the same method but going from the min to the max instead.

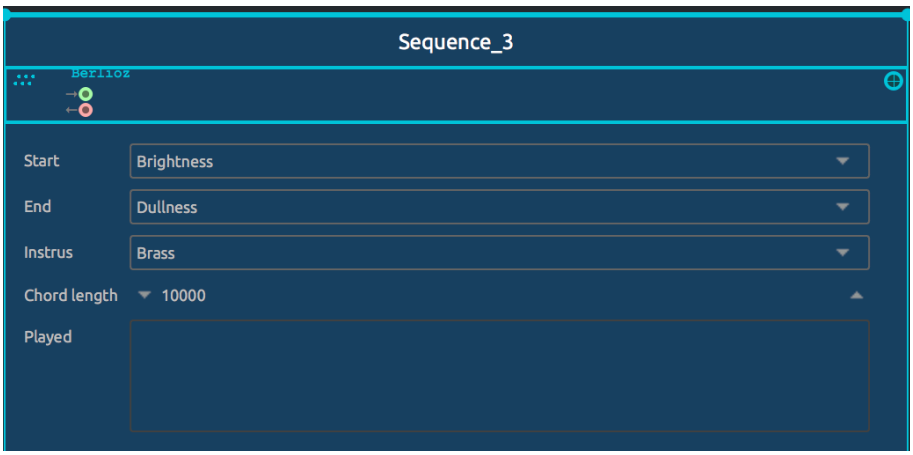

Fig. 7. The i-Berlioz process embedded in an *i-Score* object.

4.1 An example of making transitions with i-Berlioz

This section proposes to illustrate the processes for creating sequences of instrument combination transitions with i-Berlioz with a detailed example. First, considering a single run of the process shown in Fig. 7, which consists of generating a sequence of chords from brightness to dullness, the following timbral values for each chord composing the sequence were obtained:

Brightness	0.81291	0.78052	0.66858	0.58725	0.4997
Dullness	0.62474	0.71709	0.80002	0.82594	0.87524

Here, the sequence comprised of ten chords divided in two parts. First, i-Berlioz generated five brass instrument chords matching the timbral descriptor ‘brightness’, which were then concatenated in descending order. Then, five chords matching the attribute ‘dullness’ were generated and concatenated in ascending order. In other words, the system was required to generate a sequence going from ‘very bright’ to a ‘little bright’, then from a ‘little dull’ to ‘very dull’.

Now, considering the scenario example presented in Fig. 8. Here, i-Berlioz was asked to generate three sequences using the number of instruments defined by the curve drawn in the *i-score* automation box displayed at the top—in this example, using two to four instruments. The first sequence consisted of generating instrument combinations from dullness to breathiness with brass instruments. The second sequence had an instantaneous condition (shown in the OSSIA model in Fig. 6). Here, the selection of the i-Berlioz box to generate the sequence 2 depended on a condition. In this example, we used a MIDI hardware and put the condition on the MIDI values of a fader. If the fader was up, the box named *Sequence_2a* was selected. If the fader was down, *sequence_2b* was selected. Here, the fulfillment of the condition was manually defined by moving a fader on a MIDI controller. However, conditions can also be the result of

different parameters, such as temporal or number of instruments to combine for example. Then, for the third sequence, the i-Berlioz box *Sequence_3* had a temporal condition which meant to start the generation of this sequence after 15 seconds. This last sequence was composed of brass instrument combinations from brightness to dullness.

Fig. 8. A scenario example operation of i-Berlioz.

5 Concluding Remarks

We are interested in developing technology for computer-aided orchestration, or CAO. Our research is aimed at furthering our understanding of orchestral timbre and building systems for CAO informed by such understanding. This paper introduced the beta version of i-Berlioz, an interactive CAO system that generates orchestrations from verbal timbre descriptors.

Currently, i-Berlioz is capable of processing only five timbre descriptors: breathiness, brightness, dullness, roughness and warmth. Obviously, this is only a starting point. Our goal is to provide flexibility for user-defined vocabularies. However, this is a challenging task because timbre is a difficult concept to formalize, even more so orchestral timbre, as there has been relatively less research into orchestral timbre emerging from the combination of various musical instruments playing simultaneously.

Still more work need to be developed with respect to the Exploratory Engine (Fig. 2). Here we introduced how we are exploring the potential of the OSSIA time-based formalism to aid the design of sequences and transitions of timbral events in time, which we believe would ultimately aid the composition of musical form based on timbre. Interactive scores, as presented in [47], allows a composer to write musical scores in a hierarchical fashion and introduce interactivity by setting interaction points. This would enable different executions of the same score to be performed, while maintaining a global consistency by the use of constraints on either the values of the controlled parameters, or the time when they must occur.

References

[1] P. Doornbusch, "Computer Sound Synthesis in 1951: The Music of CSIRAC", *Computer Music Journal*, Vol. 28(1), pp. 10-25, 2004.

[2] L. Hiller and L. M. Isaacson, *Experimental Music: Composition with an Electronic Computer*, New York, USA: McGraw-Hill, 1959.

[3] L. Hiller and R. Baker, "MUSICOMP: MUsic Simulator-Interpreter for COMpositional Procedures for the IBM 7090". *Technical Report 9*, University of Illinois at Urbana-Champaign, Experimental Music Studios. USA, 1963.

[4] R. Baker, "MUSICOMP: Music Simulator-Interpreter for COMpositional Procedures for IBM 7090 electronic digital computer", *Technical Report 9*, Urbana, USA: University of Illinois Experimental Music Studio, 1963.

[5] A. Agostini and D. Ghisi, "Real-Time Computer-Aided Composition with bach", *Contemporary Music Review*, Vol. 32(1), pp. 41-48, 2013.

[6] G. Assayag, C. Rueda, M. Laurson, C. Agon, and O. Delerue, "Computer-Assisted Composition at IRCAM: From PatchWork to OpenMusic", *Computer Music Journal*, Vol. 23(3), pp. 59-72, 1999.

[7] J. Bresson, C. Agon, and G. Assayag, *The OM Composer's Book 3*, Paris: Éditions Delatour, France, 2016.

[8] J. D. Fernández and F. Vico, “AI Methods in Algorithmic Composition: A Comprehensive Survey”, *Journal of Artificial Intelligence Research*, Vol. 48, pp. 513-582, 2013.
[9] E. R. Miranda, *Composing Music with Computers*, Oxford, UK: Focal Press, 2011.
[10] C. Roads, *The Computer Music Tutorial*, Cambridge, USA: The MIT Press, 1996.
[11] J. Bresson, C. Agon, and G. Assayag, “OpenMusic: visual programming environment for music composition, analysis and research”, *Proceedings of the 19th ACM International Conference on Multimedia*, pp.743-746, Scottsdale, Arizona, USA, 2011.
[12] A. Farnell, *Designing Sound,* Cambridge, USA: The MIT Press, 2010.
[13] T. Winkler, *Composing Interactive Music: Techniques and Ideas Using Max*, Cambridge, USA: The MIT Press, 2001.
[14] J. A. Biles, “Improvizing with Genetic Algorithms: GenJam”. In: E. R. Miranda and J. A. Biles (Eds.) *Evolutionary Computer Music*. London, UK: Springer, 2007.
[15] M. Gimenes and E. R. Miranda, “An Ontomemetic Approach to Musical Intelligence”, In: E. R. Miranda (Ed.) *A-Life for Music: Music and Computer Models of Living Systems*, pp. 261-286. Middleton, USA: A-R Editions, 2011.
[16] F. Pachet, “The Continuator: Musical Interaction With Style”, *Proceedings of International Computer Music Conference (ICMC2002),* Gotheborg, Sweden, 2002. Available online: https://www.csl.sony.fr/downloads/papers/uploads/pachet-02f.pdf (Accessed on 18 Dec 2017).
[17] J-M. Celerier, P. Baltazar, C. Bossut, N. Vuaille, and J-M. Couturier, “OSSIA: Towards a unified interface for scoring time and interaction”, *Proceedings of TENOR 2015 Conference*, Paris, France, 2015. Available online: https://hal.archives-ouvertes.fr/hal-01245957 (Accessed on 18 Dec 2017)
[18] M. Toro-Bermúdez and M. Desainte-Catherine, “Concurrent constraints conditional-branching timed interactive scores”, *Proceedings of Sound and Music Computing*, Barcelona, Spain, 2010. Available online: http://smcnetwork.org/files/proceedings/2010/65.pdf (Accessed on 18 Dec 2017).
[19] P. White, *Basic MIDI*, London: SMT - Sanctuary Publishing, 2003.
[20] J-B. Barrière, *Le Timbre: Métaphore pour la Composition*, Paris, France Christian Bourgois, 1991.
[21] J. M. Grey, *An Exploration of Musical Timbr*e, PhD Thesis, Stanford University, USA, 1975.
[22] H. Helmholtz, *On the Sensations of Tone as a Physiological Basis for the Theory of Music*, New York, USA: Dover Publications, 1954.
[23] J. F. Schouten, “The Perception of Timbre”, *Reports of the 6th International Congress on Acoustics,* Vol. 76, pp.10, 1968.
[24] K. Siedenburg, I. Fujinaga, and S. McAdams, “A Comparison of Approaches to Timbre Descriptors in Music Information Retrieval and Music Psychology”, *Journal of New Music Research*, Vol. 45(1), pp. 27–41, 2016.
[25] E. Zwicker and H. Fastl, *Psychoacoustics: Facts and Models*, London: Springer, 2013.

[26] V. Alluri and P. Toiviainen, “Exploring perceptual and acoustical correlates of polyphonic timbre”, *Music Perception: An Interdisciplinary Journal*, vol. 27(3), pp. 223–242, 2010.
[27] J-J. Aucouturier, *Ten experiments on the modelling of polyphonic timbre*, PhD Thesis, University of Paris 6, France, 2006.
[28] A. Antoine, and E. R. Miranda, “Computer Generated Orchestration: Towards Using Musical Timbre in Composition”, *Proceedings of the 9th European Music Analysis Conference (EuroMAC 9 – IX CEAM)*, Strasbourg, France, 2017.
[29] A. Antoine, and E. R. Miranda, “Musical Acoustics, Timbre, and Computer–Aided Orchestration Challenges”, *Proceedings of the 2017 International Symposium on Musical Acoustics (ISMA)*, Montréal, Canada, 2017.
[30] A. Antoine, D. Williams, and E. R. Miranda, “Towards a Timbral Classification System for Musical Excerpts”, *Proceedings of the 2nd AES Workshop on Intelligent Music Production*, London, UK, 2016.
[31] T. De la Hogue, T. Baltazar, M. Desainte-Catherine, J. Chao, and C. Bossut, “OSSIA : Open Scenario System for Interactive Applications”, *Proceedings of Journées de l’Informatique Musicale*, Bourges, France, 2014.
[32] P. Esling and A. Bouchereau, *Orchids: Abstract and temporal orchestration software*. IRCAM, Paris, 2014. Available online: http://repmus.ircam.fr/_media/esling/orchids-documentation.pdf (Accessed on 18 Dec 2017).
[33] D. Tardieu, G. Carpentier, and X. Rodet, “Computer-aided orchestration based on probabilistic instruments models and genetic exploration”, *Proceedings of International Computer Music Conference (ICMC 2007)*. Copenhagen, Denmark, 2007.
[34] J. Hillenbrand and R. A. Houde, “Acoustic correlates of breathy vocal qualitydysphonic voices and continuous speech”, *Journal of Speech, Language, and Hearing Research*, Vol. 39(2), pp. 311–321, 1996.
[35] M. A. Landera and R. Shrivastav, “Effects of spectral slope on perceived breathiness in vowels”, *The Journal of the Acoustical Society of America*, Vol. 119(5), pp. 3339–3340, 2006.
[36] E. Schubert, J. Wolfe, and A. Tarnopolsky, “Spectral centroid and timbre in complex, multiple instrumental textures”, *Proceedings of the International Conference on Music Perception and Cognition*, pp. 112–116. Evanston, USA, 2004.
[37] A. C, Disley, D. M. Howard, and A. D. Hunt, “Timbral description of musical instruments”, *Proceedings of International Conference on Music Perception and Cognition*, pp. 61–68, 2006.
[38] D. B. Bolger, *Computational models of musical timbre and the analysis of its structure in melody*. PhD Thesis, University of Limerick, Ireland, 2004.
[39] E. Terhardt, “On the perception of periodic sound fluctuations (roughness)”, *Acta Acustica*, Vol. 304, pp. 201–213, 1974.
[40] v. W. Aures, “A procedure for calculating auditory roughness”, *Acustica*, vol. 58(5), pp. 268–281, 1985.
[41] H. Fastl and E. Zwicker, *Psychoacoustics: Facts and Models*, London: Springer, 2007.
[42] R. Pratt and P. Doak, “A subjective rating scale for timbre”, *Journal of Sound and Vibration*, Vol. 45(3), pp. 317–328, 1976.

[43] C. Cortes and V. Vapnik, “Exploring Perceptual and Acoustical Correlates of Polyphonic Timbre”, *Machine Learning*, Vol. 20(3), pp. 273–297, 1995.
[44] E. Osuna, R. Freund, and F. Girosit, “Training support vector machines: an application to face detection”, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 130–136. IEEE, 1997.
[45] V. Wan and W. M. Campbell, “Support vector machines for speaker verification and identification”, *Proceedings of IEEE Workshop on Neural Networks for Signal Processing X*, pp. 775–784. IEEE, 2000.
[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and J. Vanderplas, “Scikit-learn: Machine learning in Python”, *Journal of Machine Learning Research*, Vol. 12(Oct), pp. 2825-2830, 2011. Available online: http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html (Accessed on 18 Dec 2017).
[47] A. Allombert, G. Assayag, and M. Desainte-Catherine, “A system of interactive scores based on Petri nets”, *Proceeding of the 4th Sound and Music Conference - SMC 2007,* pp. 158–165, Lefkada, Greece, 2007.