Grégoire Carpentier and Jean Bresson
IRCAM
Music Representations Group
1 place Igor Stravinsky
F-75004 Paris, France
{Gregoíre.Carpentier,
jean.bresson}@ircam.fr

Interacting with Symbol, Sound, and Feature Spaces in Orchidée, a Computer-Aided Orchestration Environment

Until recently, orchestration has remained a relatively unexplored domain of computer music. The problem of musical orchestration could be stated as the art of combining timbres and pitches to create particular sound textures. More generally, though, orchestration comes into play as soon as timbral issues are addressed in instrumental music, and it brings forward the problem of linking the sonic space (i.e., the realm of timbres) with the symbol space of the formal compositional processes (consisting of notes, for example). Since the beginning of the 20th century, composing with "sounds" rather than "notes" has become a widespread practice (Erickson 1975), hence emphasizing the relevance of this problem in contemporary music creation. However, contrary to other techniques of musical composition, orchestration requires difficult-to-formalize and difficult-to-verbalize knowledge to assemble data from these symbol and sound spaces. Hence, for years, computer music systems stayed away from the complexity of this problem, and orchestration was never much more than an empirical activity.

Since its early years, computer music research has evolved in two main directions. On one hand, the goal was to provide composers with the ability to manipulate symbolic musical objects (e.g., notes, chords, rhythms, and melodies). On the other hand, researchers concentrated their efforts on sound processing, analysis, and synthesis, leading to a deeper comprehension of many aspects of sound and its perception. In the signal-processing field, the growing interest in automatic feature extraction (Peeters 2004) and the relationship between empirical features and perceptual dimensions (McAdams et al. 1995) paved the way towards a symbolic organization of sound. It also encouraged the emergence of music information retrieval, a field that aims to extract information from audio signals for the purposes of automatic classification and latent structure discovery. In the meantime, recent advances in the field of computer-aided composition introduced significant openings to integrate generation of electronic sound material into symbolic musical practices (Bresson and Agon 2007), making sound synthesis understandable as an interactive, modular, and formalized process.

Though very promising, these trends did not yet converge to a "unified theory," and complex connections between audio and symbolic data remain an open issue. The computer music community still misses a unified comprehensive framework that could bring together various levels of sound and music representations, integrating orchestration processes among the pioneering fields of new music research.

In this article, we present an orchestration system called Orchidée, and we focus on how user interaction may be of great help in making symbol and sound worlds communicate together. We introduce a general orchestration scheme in which each step is either supervised by an interaction process or driven by search procedures that jointly optimize objectives in symbol and audio spaces. All core computation procedures embedded into the Orchidée orchestration kernel communicate with computer music environments through a simple client/server architecture. We introduce innovative interfaces in OpenMusic (Assayag et al. 1999), Max/MSP (Puckette 1991), and MATLAB, both for the specification of the orchestration problem and for the exploration of its potential solutions. These interfaces allow a fine comprehension of the links between symbolic and audio data through feedback loops and preferences-inference mechanisms.

Computer Music Journal, 34:1, pp. 10–27, Spring 2010
© 2010 Massachusetts Institute of Technology.

This article is organized as follows. We first report previous research in the computer music community aimed at designing orchestration tools. We discuss their inner limitations and show how consequently they miss various aspects of orchestration complexity. We then suggest a multiple-viewpoint analysis of this complexity as a network of relations among sound, symbol, and feature spaces. From there, the orchestration problem is turned into a generic scenario for which the Orchidée client/server framework offers a solution. Each main step of this scenario (problem specification, search process, and exploration of solution space) is then separately detailed, and examples of graphical user interfaces (GUIs) are provided.

Related Work

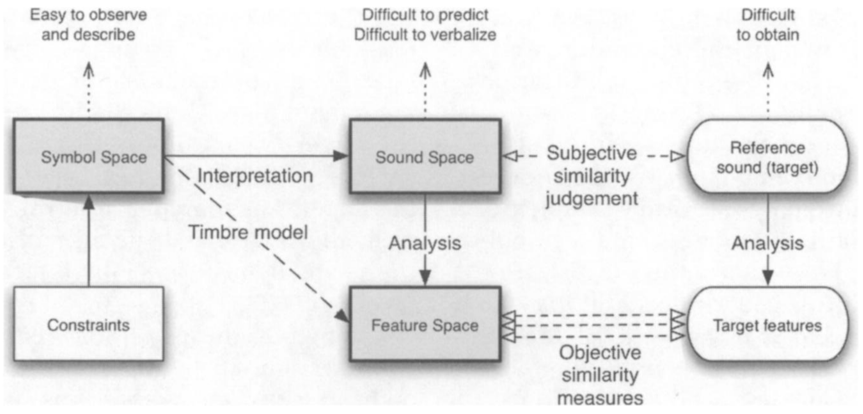
Though computer-aided orchestration is quite a new topic in the computer music field, at least three attempts have been made in past research for designing orchestration tools. They all share the same paradigm: The goal is always to discover sound combinations that "imitate" a target sound in the most convincing manner. Sound combinations are searched within an instrument sample database assumed to be large enough to encompass the sound potential of the orchestra. As we will see, our approach follows this trend, but it invokes innovative methods at each step of the orchestration process.

These state-of-the-art systems rely on the assumption that the target timbre—as well as that of each database item—can be characterized by some averaged spectrum or spectral envelope computed on the signal with appropriate spectral-processing techniques. Rose and Hetrik (2009) introduced an explorative and educational tool based on a singular value decomposition (SVD) and other basic linear algebra methods. Their algorithm approximates the target spectrum with a weighted sum of spectra picked from a palette of instrumental sound samples. To cope with orchestral constraints, the authors also suggested the CHI procedure, which first computes the set of all feasible combinations, and then ranks them on a distance-to-target criterion. Alternative approaches suggested by Psenicka (2003) and Hummel (2005) rely on iterative matching pursuit algorithms. Such methods first compute the target's spectrum and then find in a sound database the items whose spectrum best matches the target. The matched spectrum is then subtracted from the target, and the algorithm iterates until the residual spectrum falls under a given energy level.

Though all these methods require relatively low computation times, they miss the complexity of orchestration at various levels. First, they implicitly circumvent the combinatorial complexity of orchestration. The search for sound combinations in large databases is an NP-hard constrained optimization problem in regard to which iterative matching pursuit methods behave like greedy algorithms. Sounds are added one by one in combination until an optimization criterion cannot be further improved. Now, the lack of a backtracking mechanism with iterative methods most often results in local optima with uncertain quality (Hoos and Stützle 2005). The SVD approach suggested by Rose and Hetrick (2009) avoids these matters but is likely to output non-playable solutions, i.e., ones that are inconsistent with constraints of orchestral instrumentation. As for their alternate CHI method, it performs an exhaustive search on the set of feasible solutions; hence, it is only practicable on small-size problems.

Secondly, these state-of-the-art methods fail in considering timbre perception as a multidimensional phenomenon (McAdams et al. 1995; Jensen 1999). The optimization process is always driven by a single objective function, thus limiting the applications to rare situations in which the average spectrum holds the whole perceptual timbre information. Third, these methods offer poor control of symbolic features in orchestration proposals. The search is driven by the optimization of a spectral criterion, no matter the values musical variables (such as pitches) may take. It is therefore generally difficult to take advantage of the resulting solutions in real compositional processes. Last but not least, all of these orchestration tools are basic procedures rather than advanced compositional environments. They

Figure 1. Sound space, symbol space, feature space: a network of complex relations.



can take as input argument only a concrete, pre-recorded, target sound, yet this does not correspond to the real practice of orchestration. In most cases, the target timbre is nowhere but in the composer's mind, and it strongly relates with symbolic musical material that the state-of-the-art methods are unlikely to consider.

Why Orchestration Is a Complex Problem

Apart from purely combinatorial issues (the number of sound combinations potentially playable by an orchestra is virtually infinite), we believe the inner complexity of orchestration lies in the complex network of relations between sound and symbolic data. The historical orchestration treatises (Berlioz 1843; Rimski-Korsakov 1912; Koechlin 1943; Piston 1955; Adler 1989) have tried to "transcribe" some of these relations as a collection of "tricks" of which composers should be aware. However, they all refer to the aesthetics of a given time, and they hardly provide convincing scientific reasons that explain why some timbral arrangements sound "better" than others.

Sound Space, Symbol Space, Feature Space

Figure 1 helps in clarifying the complex relations between sound and symbolic materials. From a symbolic viewpoint, an orchestra is a discrete system that can be at each time easily described by a "score segment"—a set of discrete variables: instruments, notes, dynamics, playing styles, etc. Variable domains are implicitly restricted by a set of constraints reflecting the physical limitations of the instrumental playing.

When it comes to playing music, the current score segment is "projected" onto the sound space. Unlike the symbol space, the latter is difficult to observe, at least scientifically speaking. Music deals with sound perception, but verbalizing or formalizing this perception is quite a difficult task. It is well known in the music community that no suitable vocabulary is available for precise description of timbre. In addition, the sound space is hardly predictable: composers are limited in their faculties to imagine how timbre combinations sound, particularly when there is no prior experience to refer to and as the number of instruments under consideration increases. Besides the combinatorial issues of instrumental mixtures, unpredictability and non-objectivity in sound spaces are the main causes of orchestration's inner complexity.

However, if we are not able to precisely characterize objects in sound spaces, we still can compare them. Many psychoacoustic experiments involving the building of timbre spaces rely on this pairwise sound-comparison principle (McAdams et al. 1995). Introducing a "reference sound" in Figure 1 is the first step to an organization of the sound space, as sounds may then be arranged according to their similarity with the reference. This is why all current orchestration systems use a target sound as the main input: A landmark is needed to measure distances in the sound space. However, instead of dealing with subjective similarities at the sound level, these systems prefer a spectral-based representation for computing distances. As previously noted, this approach implicitly makes the assumption that the average spectrum holds all timbral information and that timbre similarity and spectral distance are equivalent, which is not straightforward.

Perceptual features may help here. These features are automatically extracted from audio signals (Peeters 2004) and can easily be correlated with perceptual dimensions. For instance, the spectral centroid correlates with perceptual brightness, and attack time correlates with percussive aspects of sound (McAdams et al. 1995). Objects in the sound space may then be compared to a given reference sound on the basis of their objective feature values (reflecting different perceptual dimensions). But objectivity is not without cost: Feature space can have a high dimensionality, and the choice of an appropriate distance is still a complex problem. It has been shown (Carpentier 2008) that merging feature distances into a single dissimilarity value is impossible without prior information on listening preferences. We show in the next section how a multicriterion approach is an elegant way to cope both with the multidimensionality of timbre perception and the unpredictability of each dimension's relative importance.

Theoretical Formalization

From now on, we will assume the existence of a sound database reflecting the sound possibilities of the orchestra. (The instruments must have been recorded at various pitches, playing styles, and dynamics.) A computer-aided orchestration system will then suggest combinations of sounds in this database according to a given problem specification. We will also assume that each sound s of the database is associated with a set of audio features $(d_1(s), \ldots, d_K(s))$ that reflects various aspects of its timbre. Examples of such features are spectral centroid, log-attack time, main resolved partials, mel spectrum, noisiness, etc.; see Tardieu (2008) for a detailed review. We will call the combination of both the sound database and the feature database *instrumental knowledge*.

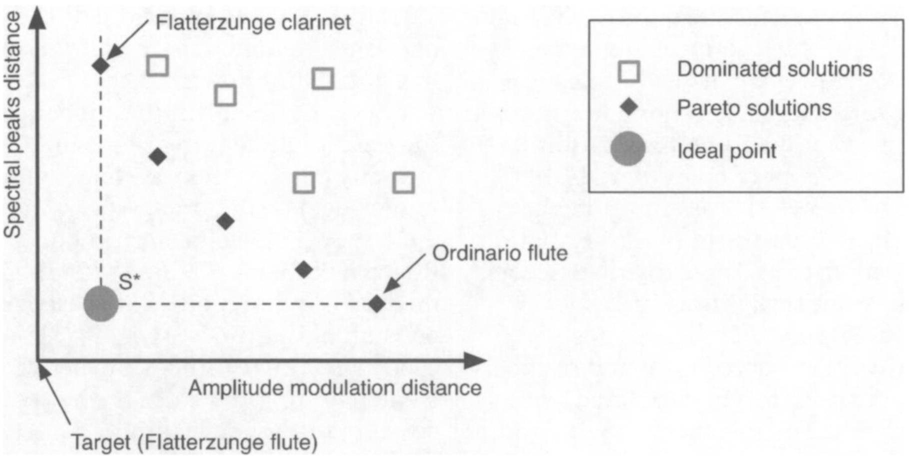
Now imagine a timbre model that can predict the perceptual features associated with any state S (i.e., any combination of sounds in the instrumental knowledge) of the orchestra in the symbol space. Such models have been introduced in Tardieu and Rodet (2007) and Tardieu (2008). Given a sound target T, timbre models also estimate the vector of perceptual dissimilarities $(D_1^T(S), \ldots, D_K^T(S))$ along each perceptual dimension between S and T. Hence, finding the state S^* that sounds as close as possible to the target timbre is left to the following multicriterion minimization problem (Ehrgott 2005):

$$S^* = \left\{ \arg\min_S D_k^T(S), k = 1, \ldots, K \right\} \qquad (1)$$

In multicriterion optimization, S^* is called the *ideal point*. It is a configuration that simultaneously optimizes all criteria. In most real-life cases, S^* does not exist, and solving Equation 1 then consists of finding a set of tradeoff states that realize different compromises among the conflicting objectives. Such solutions are called *Pareto solutions* (or *efficient solutions*, or *optimal solutions*). A solution S is optimal if and only if there is no other solution in the search space that achieves better values than S on every criterion D_k^T. Figure 2 illustrates these concepts. Here, solutions are simple sounds rather than sound combinations to make the Pareto approach easier to understand. Axes are distances to the target (here, a flutter-tongue [*Flatterzunge*] flute sound) along two criteria: spectral peaks distance and amplitude modulation distance. Hence, the target timbre is the origin. The flutter-tongue clarinet is the closest solution regarding amplitude modulation, whereas the normal flute achieves the lowest spectral peaks distance. Unfortunately, there is no solution S^* in the search space that combines those two properties.

The multicriterion approach is an appropriate paradigm when the relative weight of each desired criterion cannot be known in advance. Orchestration (and more generally, timbral similarity) falls into

Figure 2. Bi-criteria timbre similarity. Axes represent distances to the target (here, a flutter-tongue or Flatterzunge *flute sound) timbre along two criteria: spectral peaks distance and amplitude modulation distance.*



this category of problems. Depending on targets and composers, some regions of the Pareto solution set will be preferred to others, according to specific and personal preferences of perceptual objectives. We will call them *listening preferences* in the remainder of this article.

In Carpentier (2008), we suggested an appropriate genetic algorithm (Goldberg 1989) for efficiently solving Equation 1. Eventual global symbolic constraints on musical variables are also tackled with an innovative local search procedure. As the present article mainly deals with interfaces and user interaction, we will not go into details on these methods. We will simply refer to them to as *main search procedures*. Interested readers may consult Carpentier et al. (in press).

General Framework Architecture

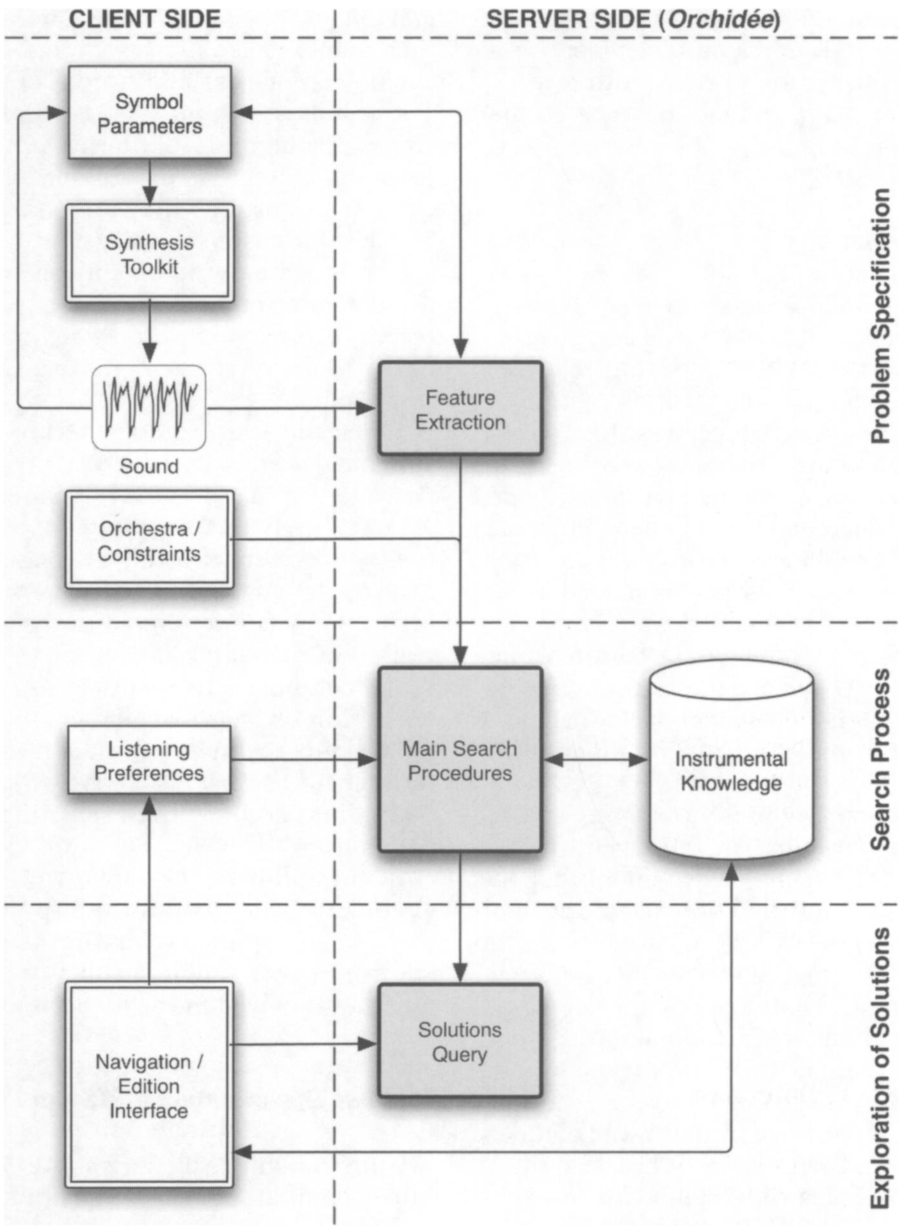
Introducing feature-based timbre models "between" sound and symbol spaces let us address the inner unpredictability and non-objectivity of the sound space by a multicriterion combinatorial optimization problem. Without oversimplifying the complexity of orchestration, we thus turned our problem into a general theoretical framework for which various efficient methods were introduced. However, major difficulties remain at two levels at least.

First arises the issue of a general specification of an orchestration. Remember that the goal is to find a combination of sound samples whose features best match a set of target features according to a multidimensional perceptual distance. The concept of target can therefore have various different meanings. It can be considered as a concrete sound (e.g., a natural or instrumental recording) from which target features may be extracted. More generally, it can also be a more or less abstract idea of a resulting sound mixture to be created, which does not exist before it is actually produced with the orchestra. Hence, the question of defining a sound target as the initial base of the orchestration process is actually linked to the much more general problem of the musical conception of sounds in the composers' mind. In this case, when no target sound is available, we provide a set of synthesis tools that allow for its generation in the context of a computer-aided composition environment. (We introduce these tools in later sections of this article.)

Second, a multicriterion problem leads to a set of efficient solutions rather than a unique optimum. Depending on the problem and on the number of objectives, this optimal set might be very large. Hence, helping users in the discovery of configurations satisfying their personal needs is another major issue.

In these two representative steps, our belief is that user interaction has a key role to play. Figure 3 formalizes the interactions in our system through a client/server architecture. Interfaces and user interaction are on the client side, and the Orchidée

Figure 3. Generic orchestration scenario through the Orchidée client/server framework. Double-lined boxes denote user interfaces.



server (i.e., the orchestration kernel) handles the core computational procedures. The client–server communication is established using simple OSC messages (Wright 2005).

The double-line boxes denote user interfaces (Orchidée clients). Once the server is running, one or several of such client applications or interfaces can send their data and information (target specification, search constraints, and preferences). They can also run search procedures and queries about their results or about the state of the server. This architecture therefore opens a wide range of possibilities and approaches on the clients' sides, for instance in the definition and synthesis of sound targets, or in the visualization and navigation in the solution space, which both represent key interaction areas in the framework we propose.

Orchestration Scenario

At the start of the orchestration process, the user specifies the initial problem data: the composition of the orchestra, some specific constraints relative to the orchestra or to the expected solutions, and above all the sound target. Orchidée receives this target as a sound file that contains either a preexisting sound (as in previous researchers' orchestration software) or a newly synthesized one from a client interface, and analyzes it to extract and return target features. The target feature set should be considered as a "sound abstraction" or a "sound class": there might be many sounds corresponding to the feature values, and the sound used to specify the target is one of them. The goal then is to discover other instances of this sound class made out of instrumental samples.

In the next section, we present a target specification interface developed in the OpenMusic computer-aided composition environment that acts as an Orchidée client and allows the user to generate and progressively refine the target sound from symbolic parameters. During the various steps of target generation, the features extracted by the server can be queried by the client application and integrated as symbol data in the specification process.

In addition to the generation of target features, the problem can also be specified by orchestral composition and instrumental constraints. It has been shown in previous work (Carpentier 2008; Carpentier et al. in press) that constraint programming is an appropriate framework for modeling the current compositional context. Our OpenMusic-based client provides programmatic tools and GUIs to specify these additional problem-specification data.

Target features and context constraints are therefore the inputs to the orchestration problem, which is addressed in Orchidée as a constrained multicriterion combinatorial optimization task (see the previous discussion). The main search procedures then combine genetic search and local search to explore various optimal regions of the search space. The search is split into two collaborative processes. On one hand, a multicriterion genetic algorithm tries to approximate in a reasonable time the efficient solutions (i.e., the Pareto set) of Equation 1. These solutions display different tradeoffs between the optimization criteria corresponding to possible user listening preferences. On the other hand, a local search algorithm tries to find configurations that fit the musical requirements expressed as symbolic constraints.

When the search process terminates, the server stores and eventually returns a set of sound combinations that all match the target timbre. Subsequently, we present another client interface that allows the user to interact with this set of solutions and to investigate it according to different criteria. According to the user's actions in this interface, relevant regions of the search space (i.e., regions that contain sound combinations that meet the user's aesthetic needs) can be identified and provided to the server. From there, the main procedures might be redrawn to intensify the search in particular regions, now taking into account the user's implicit listening preferences.

In the following sections, we introduce the Orchidée clients used for the specification of the problem and for the exploration of its solution space, and we further explore details of the interaction schemes involved in the orchestration process.

Problem Specification and Sound Target Definition

The first Orchidée client was created in the OpenMusic computer-aided composition environment, including a set of tools and interfaces dedicated to the specification of the orchestration target and constraints. Although the use of a visual programming language like OpenMusic may suggest some kind of automated process, an important proposal conveyed in this environment is the strong user interaction and control over the creation and execution of compositional processes. Hence, the use of OpenMusic as a client of our orchestration server allows us to integrate formalized aspects carried out by means of programming tools, as well as an interactive orientation of the overall process through editors and bidirectional communications between OpenMusic and Orchidée. Moreover, the problem data are not strictly fixed by the initial specification: a progressive fine-tuning of the target information and of the orchestration constraints generally takes place after each step of the orchestration search process.

Sound Target

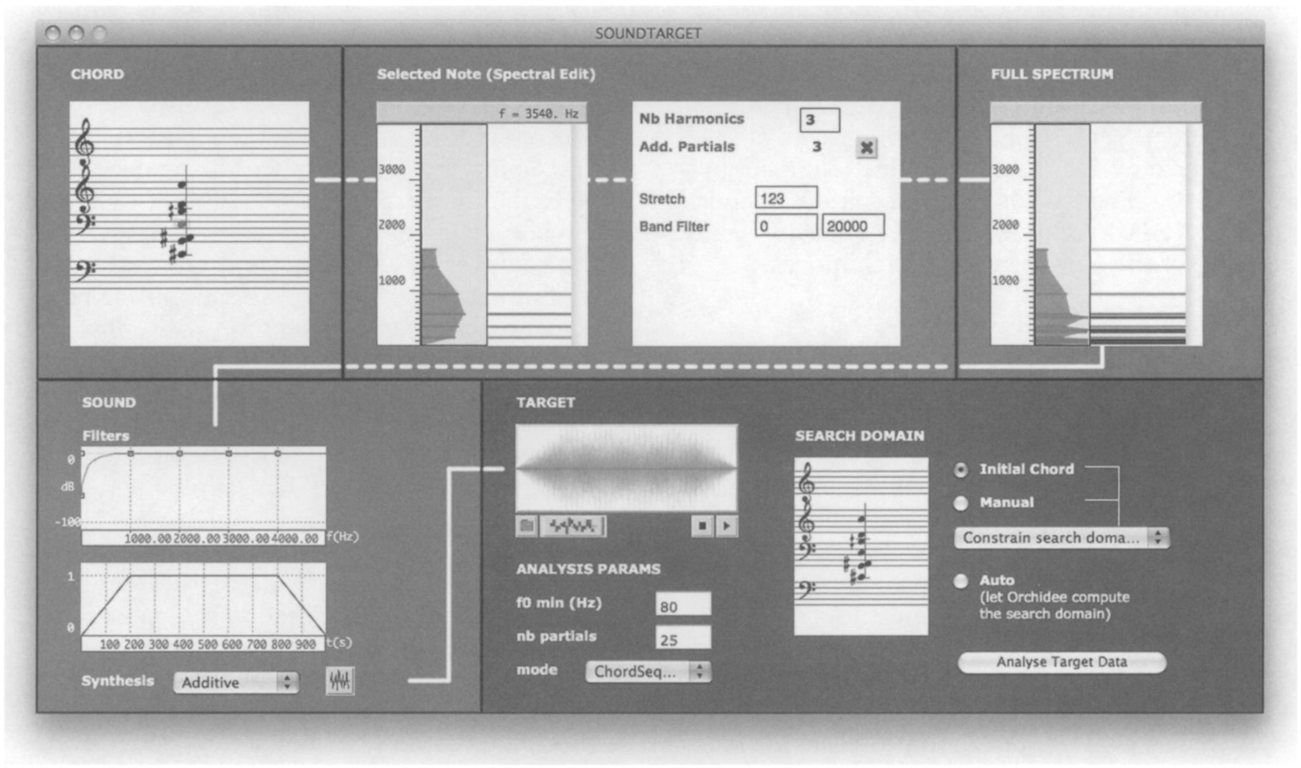
As stated earlier, the definition of an orchestration target is closely related to the problem of the musical representation of sound, which has already been addressed from various perspectives in computer music literature and software. The first and perhaps most intuitive approaches are based on graphical representations, and they generally assume a bi-dimensional time/frequency referential on which sound textures and events are "drawn." In this approach, there is a correspondence between the graphical representation and the perceptual dimensions of sound. However, the sounds resulting from the related systems are quite restrictive in comparison to the realm of sounds that can be created by a computer and, even more so, to the realm of sounds that can be imagined by a composer. An opposite approach would represent the sound in a compositional context as a process being created by the composer. With several underlying conceptual distinctions, this approach principally corresponds to the programming and visual programming languages and environments such as Music-N/Csound (Mathews 1969), SuperCollider (McCartney 1996), and Max/MSP (Puckette 1991). Sound creation in these environments can be extended to a more musical level, because the sound is formalized as a part (or as the object) of a compositional process (Bresson and Agon 2007). However, this task is neither easy nor very intuitive and can easily lead composers to waste time and energy on building the target even before thinking about how to orchestrate it.

The target definition interface we created in OpenMusic takes advantage of both previously mentioned approaches. It uses different tools previously developed for the control of sound synthesis in this environment, principally of the OMChroma system (Agon et al. 2000), which provides powerful high-level structures for the design of pitch structures and the synthesis of complex sounds.

The *SoundTarget* is an OpenMusic object dedicated to the specification of the orchestration target. It provides a convenient interface that hides most of the complex underlying sound processing and programming issues while maintaining the potential of a programmable system. As with most of the OpenMusic objects, the *SoundTarget* can be built and edited by programming means or by using a dedicated graphical interface. The *SoundTarget* editor gathers symbolic, spectral, temporal, and functional representations of the different components of a target (see Figure 4).

Initially, a *SoundTarget* object is built from a chord containing the primary structural pitches of the target. This chord is edited in the upper-left part of the editor window. Each note is associated to a spectrum, i.e., a single partial that can be extended either with a given number of harmonics or by user-defined partials. Some additional items are visible in the "Selected Note" frame of the target editor in Figure 4, which allows for the algorithmic processing of the spectral components (e.g., filtering, stretching, shifting, and setting of the relative amplitudes). This additional processing permits creating a wide variety of timbre structures whose richness and complexity result from pre-formalized processes and experiments. Predefined functions are provided for this purpose. As we will show, user-defined functions can also be plugged in at this stage. The spectra of the chord components are then brought together into a global spectrum ("Full Spectrum," in the upper-right part of the window) to which a spectral envelope filter and a temporal envelope can be applied (bottom-left part of the window). Finally, a concrete sound is synthesized from the resulting target data, using one of several available synthesis techniques (additive, FM, etc.) This sound is our orchestration target; it is sent with additional spectral or symbolic information to the orchestration server for complementary analysis and extraction of perceptual features.

Figure 4. The OpenMusic SoundTarget editor.



f = 3540. Hz
Nb Harmonics 3
Add. Partials 3
Stretch 123
Band Filter 0 20000
Synthesis Additive
f0 min (Hz) 80
nb partials 25
mode ChordSeq...

The strength of the *SoundTarget* object also lies in its easy integration into the OpenMusic visual programs. Indeed, its essential components (i.e., the initial chord, the spectral components of the chord's notes, and the filtering envelopes and parameters) can be created by upstream processes or derived from existing OpenMusic objects. In other words, the target creation can be integrated into a larger-scale compositional framework. Figure 5 shows a *SoundTarget* being created from a patch. Before sending target data to the Orchidée server, the *SoundTarget* object may still be edited and refined manually through its editor.

The functional components of the *SoundTarget* editor (i.e., the functions plugged in and related to the spectral unfolding of the chord pitches) can also be set algorithmically and parameterized when building the sound target. In this case, these spectral-processing modules can consist of any convenient function to process the underlying pitch structure. They can come from a predefined library provided with the orchestration tool, or they can be previously composed/programmed by the user in the computer-aided composition environment (either as Lisp functions or as visual programs). In Figure 6, a *SoundTarget* is now created with additional spectral processing functions designed as visual programs.

Notice that it is also possible to build the *SoundTarget* directly from a sound file previously created or imported in OpenMusic, thus allowing one to specify the orchestration target as a pure "concrete" sound. In this case, or if the target creation and synthesis interfaces are not needed anymore, the *SoundTarget* editor can be set to a more compact visualization mode, as shown in Figure 7.

The sound-target information submitted by the client is considered at different levels by the Orchidée server. In any case, perceptual features are extracted from the sound file for use as optimization

Figure 5. Building a SoundTarget object in an OpenMusic visual program. The initial chord is derived from an algorithmic process, and the different filters are derived from a library of predefined formant filters. Double-clicking the SoundTarget box opens the editor, a part of which is visible at the right of the figure.

Figure 6. Creating a SoundTarget from symbolic data and functions for the spectral processing of this data. A band-pass filter is specified with default frequencies, as well as another predefined function and a third user-defined function (the "ramp" editor is open at the right) that randomly modifies the amplitude of the spectral components.

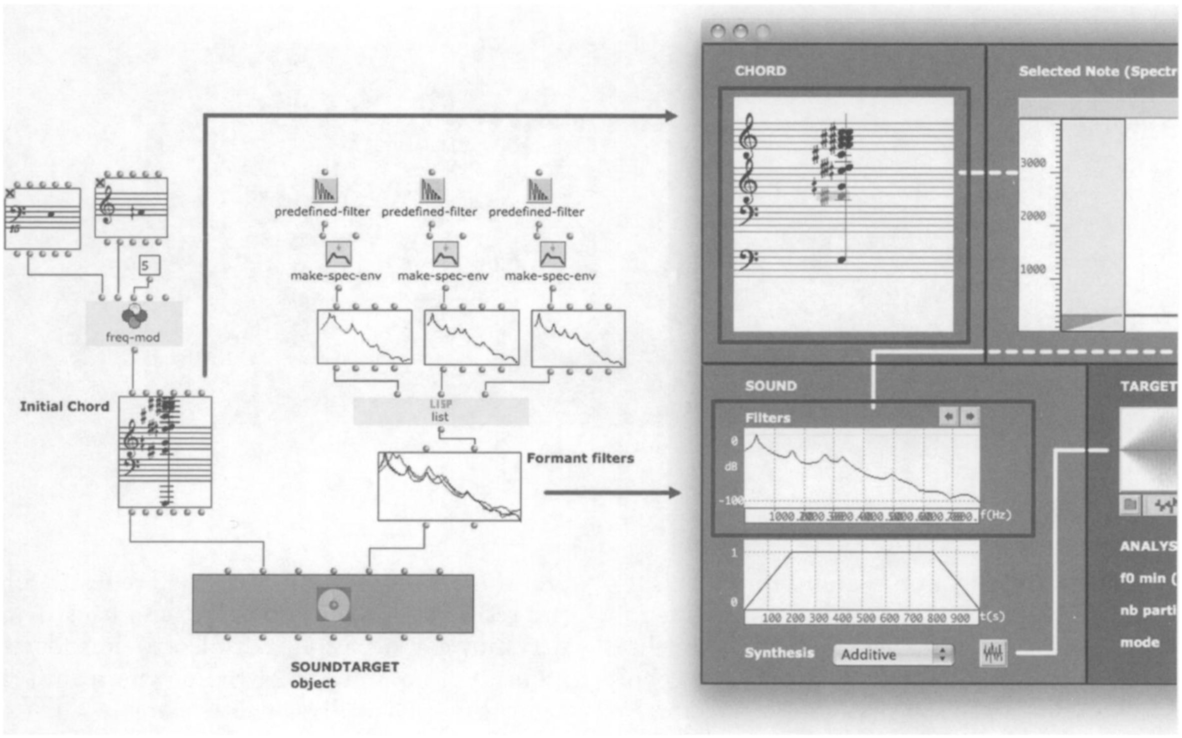


Figure 5

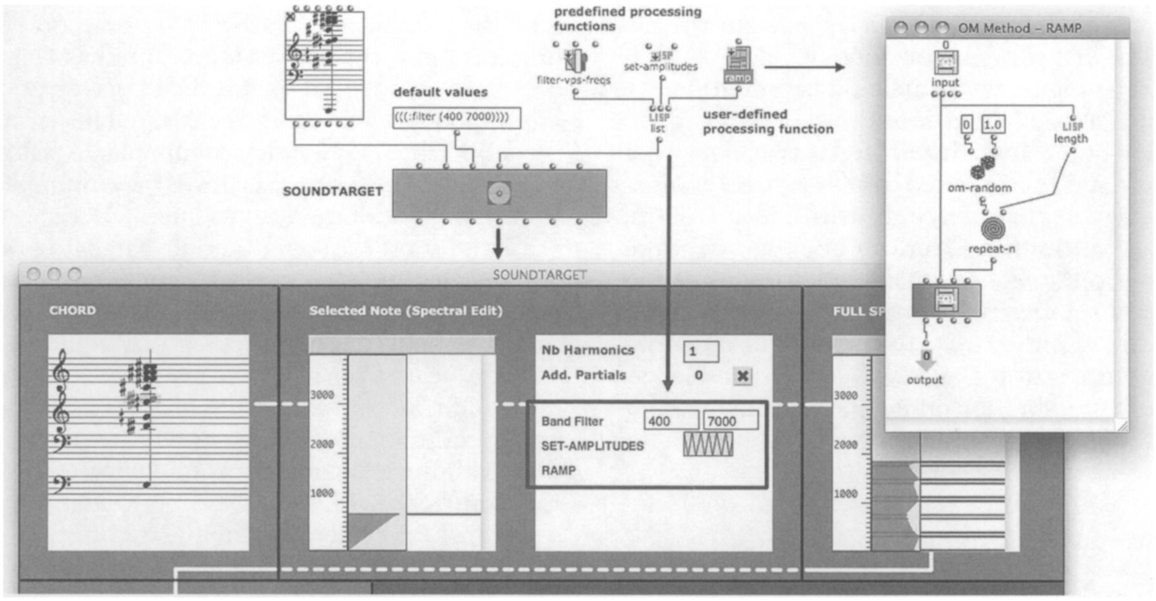
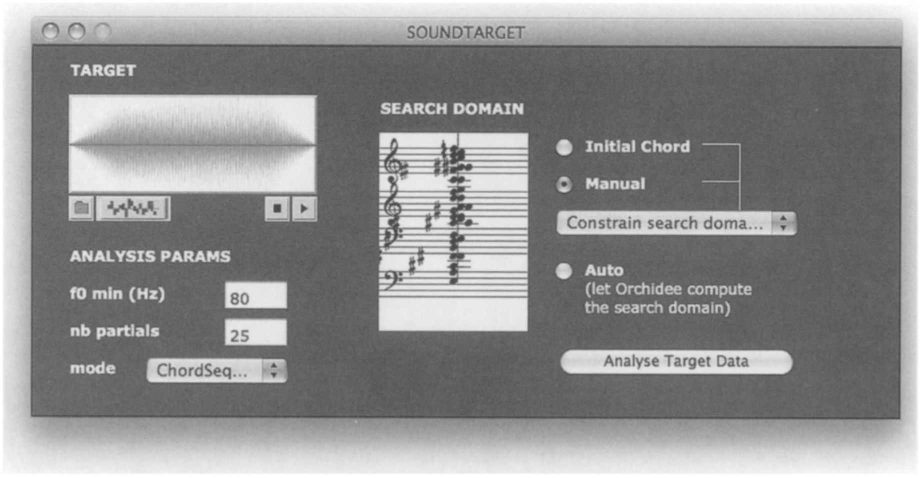


Figure 6

Figure 7. Compact visualization of the SoundTarget *editor.*



criteria in the orchestration search procedures. If provided, symbolic data can be used as a filter to limit the possible pitches that may appear in the orchestration proposals (the "Search Domain" chord seen in Figures 4 and 7). If no chord is specified, Orchidée automatically computes a set of pitches from the target spectrum. Because it belongs to the symbol space, it can be returned to the target specification environment and progressively refined during the overall process (see the bottom-right part of the *SoundTarget* editor window).

Before going further, it should be noted that the tools introduced in this section only allow for the synthesis of a static timbre target, that is, without any temporal evolution. This restriction is not imposed by the system architecture depicted in Figure 3, but by the nature of Orchidée's timbre features itself. Though efforts are currently being made to exploit temporal aspects of timbre in automatic orchestration, Orchidée still describes sounds with features averaged over time. Hence, it is only suitable for orchestrating single time slices.

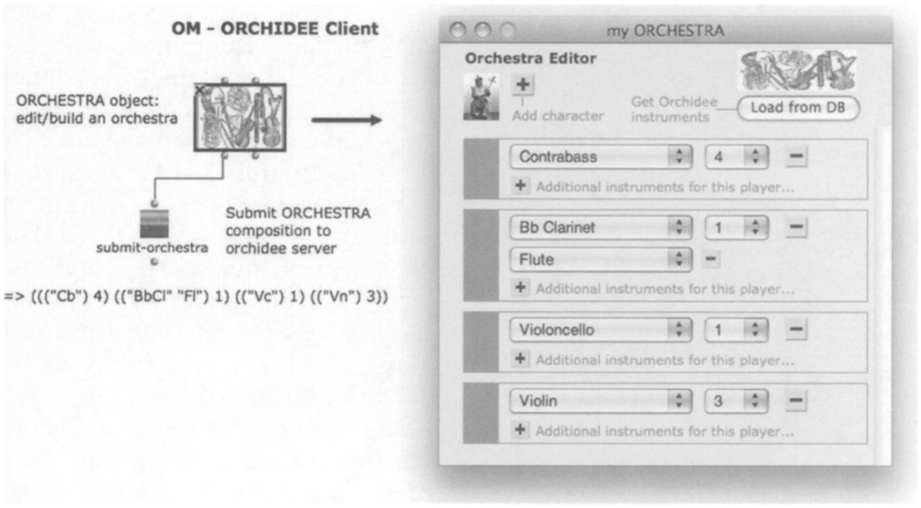
Orchestra and Constraints

Another essential element of the orchestration problem is the composition of the orchestra. The preliminary specification of this orchestra allows the research domain of the process to fit with a real musical situation, but also to considerably reduce the complexity of the downstream search algorithms. Basically, an orchestra is a list of "characters" (or players), where each character can play one or several instruments.

Figure 8 shows an example of an *Orchestra*, another object created in OpenMusic and provided with its orchestra editor. In this example the orchestra is composed of four contrabass players, one clarinet/flute player, one cello player, and three violin players. The instruments are chosen from a predefined list of possible instruments. At the upper-right part of the orchestra editor, the command "load from DB" allows the user to query Orchidée about the contents of the sound-sample database to set this list of instruments accordingly. The function box labeled "submit-orchestra" formats the orchestra information as a simple message and allows it to communicate to the Orchidée server.

In addition to the instruments of the orchestra, extra requirements can be stated corresponding to practical musical constraints or dedicated to driving the search algorithms toward particular sets of solutions. Example of such constraints can be for instance "at least two violins must play the same note," "there must be fewer than two G-sharps in the solution," and so on. The patch in Figure 9

Figure 8. Interface for the specification of an orchestra in OpenMusic. The orchestra object on the left can be edited using the editor visible on the right of the figure. The box "submit-orchestra" translates it into a message to be sent to Orchidée.


OM - ORCHIDEE Client
ORCHESTRA object: edit/build an orchestra
Submit ORCHESTRA composition to orchidee server
submit-orchestra
my ORCHESTRA
Orchestra Editor
Add character
Get Orchidee instruments
Load from DB
Additional instruments for this player...

=> ((("Cb") 4) (("BbCl" "Fl") 1) (("Vc") 1) (("Vn") 3))

Contrabass 4
Bb Clarinet 1
Flute
Violoncello 1
Violin 3

is a complete orchestration process implemented as an OpenMusic visual program, integrating additional constraints into the search procedure (top right).

Exploration of the Solution Space

The orchestration search procedures in Orchidée tackle two distinct problems at the same time: a multicriterion optimization problem on perceptual feature distances, and a constraint satisfaction problem on musical variables. Each of these problems can have many solutions, especially with a large number of perceptual features. Hence, appropriate methods are needed to help the user in the fast discovery of musically relevant orchestration proposals. Here again, user interaction is the core feature.

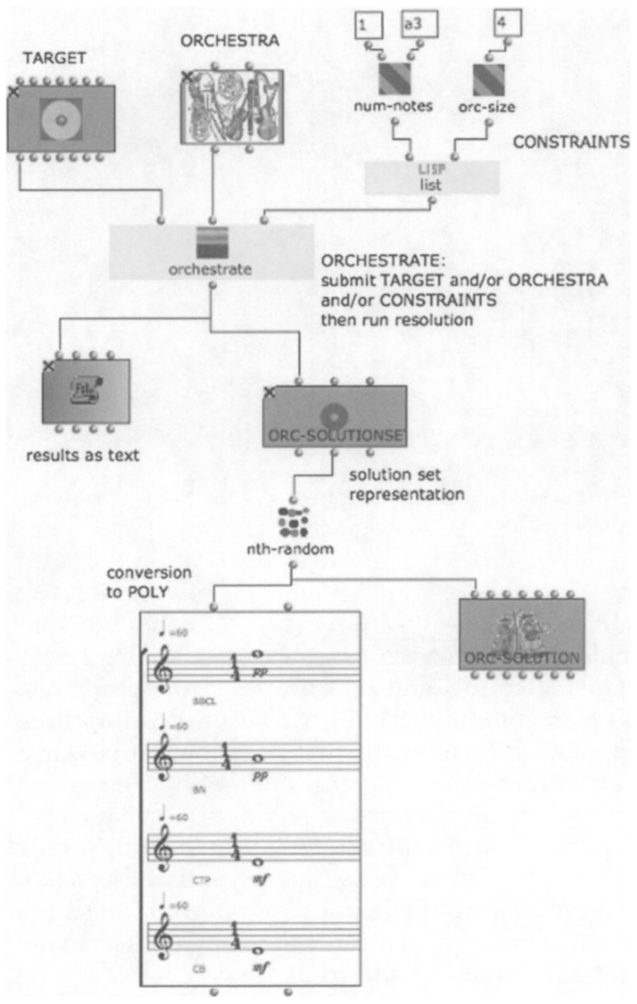
Multiple Space Exploration

As with the target specification procedure, we believe that simultaneous access to the heterogeneous spaces of orchestration has to be encouraged for exploring the solution set. To enhance an intuitive understanding of the complex relations between symbol, feature, and sound spaces, we put forward a set of interconnected exploration modules as depicted in Figure 10.

Each orchestration solution is characterized by a set of musical variables (e.g., note, dynamics, playing styles) in the symbol space, as well as a set of perceptual descriptors in the feature space and a set of objective functions values in the criteria space. The usual representation for the optimal solutions of a multicriterion problem is the criteria space. As far as orchestration is concerned, it describes perceptual distances to the target along different timbral dimensions. It differs from the feature space, as multidimensional features (e.g., the spectral envelope) may be merged into a single scalar value, i.e., a single objective function to optimize.

Figure 11 displays an example of coupling different views for navigation in the solution set. This interface is an Orchidée client in MATLAB that allows the visualization of orchestration proposals in different spaces. Symbol, feature, and criteria spaces communicate with each other to propagate user actions. When selecting a solution or a group of solutions in a given space, the corresponding solution or group is highlighted in

*Figure 9. Running Orchidée from OpenMusic. This patch submits a target, an orchestra, and additional constraints to the orchestration server, then runs the search procedures and stores the resulting solutions either in a text buffer, dedicated data structures (*orc-solution*), or a* poly *object (i.e., in score notation). The constraints at the right state that there must be an A3 in the solution, and that there can be at most four instruments playing.*



the other two. Hence, an intuitive and empirical understanding of the relations between the heterogeneous orchestration spaces is possible. Any orchestration solution selected from any space can be simulated (i.e., projected in the sound space) thanks to the instrumental-knowledge sound samples.

In the upper-left corner in Figure 11, the orchestration-efficient solutions are plotted on a hierarchical cluster tree in the symbol space. Symbolic clustering is a convenient way to represent regions of the solution set as more or less abstract schemas. Solutions are gathered in a same cluster when they share common symbolic characteristics that we call a "schema." The higher the node in the tree, the more abstract the schema. The number of edges between two solutions in the tree reflects the symbolic dissimilarity between them.

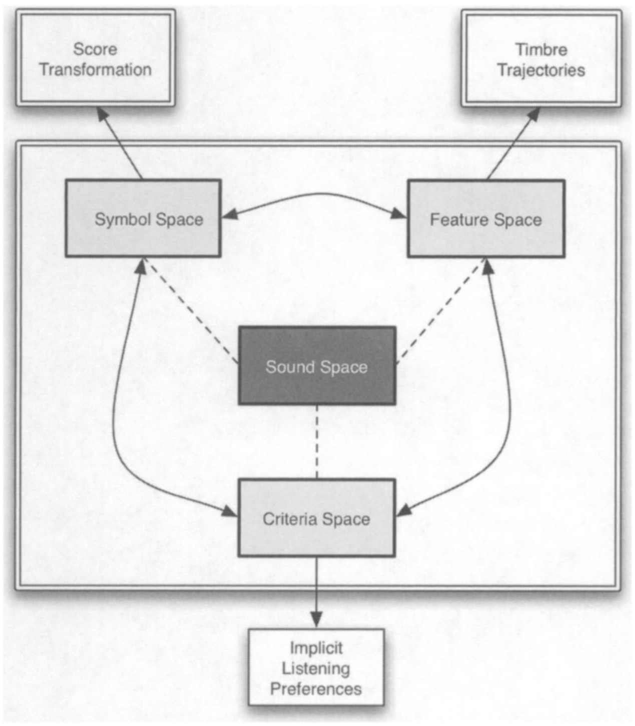
All solutions enclosed within the "Selected Subtree" rectangle in Figure 11 share the following schema explicated in the right column of the symbol space module: B-flat clarinet (BbCl) plays a quartertone above A-sharp5, bassoon (Bn) plays F-sharp4, etc. Pitches played by violin 1 (Vn) and cello (Vc) differ within the group, and violin 2 is always silent. The selected solution cluster is "propagated" into the criteria space (lower-left module) that reflects the relative distances of each solution to the target timbre (located at the origin), as well as into the feature space in the lower-right window. (Spectral envelopes are observed here; note that each orchestration in the group has roughly the same envelope.) In the upper-right corner, a representative solution (i.e., the cluster center) of the selected group is shown as a set of sound-file names and can be simulated in the sound space upon user request by simply playing a mixdown of the corresponding sound files.

Transformations in the Symbol Space

As shown in Figure 10, each of the three enclosed modules in the navigation interface can also be the starting point of a specific transformation process. From the symbol space, orchestration proposals can be edited either manually or automatically by updating the constraint structure. Figure 12 is an example of a mixing-console-like editing interface in Max/MSP that allows a fine-tuning of the symbolic parameters as well as the balance between each component in the orchestration.

Modifications in the symbol space can also be performed by adding constraints. Figure 13 shows how an orchestration proposal is gradually transformed into another through the addition of extra constraints. The initial solution is an imitation of a sung vowel played by string instruments. Using

Figure 10. Navigating in orchestration proposals via interconnected exploration modules (symbol space, feature space, criteria space). The three boxes outside the large rectangle concern transformations of the corresponding spaces, as described in the text.



the constraint-satisfaction local search engine embedded in Orchidée (and recording the trace of the algorithm resolution), we gradually turned the initial chord into another orchestration played by brass and woodwinds only. The whole passage is displayed on a score in Figure 14. Note that, owing to the local search approach, there is only one single instrument change for each bar, therefore ensuring the continuity in the overall timbral evolution.

Trajectories in the Feature Space

Timbre trajectories can also be computed from the feature space. The goal here is to find a sequence of orchestrations of which the perceptual features evolve along a predetermined path. This principle was used by composer Jonathan Harvey with whom we collaborated for the writing of his recent piece *Speakings*, for orchestra and electronics (Nouno et al. 2009). From a pre-recorded sound, we generated various orchestration solutions with different constraint sets and thus obtained a wide range of values along the different dimensions of the feature space. We then used a local search algorithm to find a "path" in the overall solution set in such a way that all the overall loudness, brightness, and the spectral similarity with the most prominent target peaks increase over the evolution. The whole orchestration process is fully detailed in Carpentier (2008).

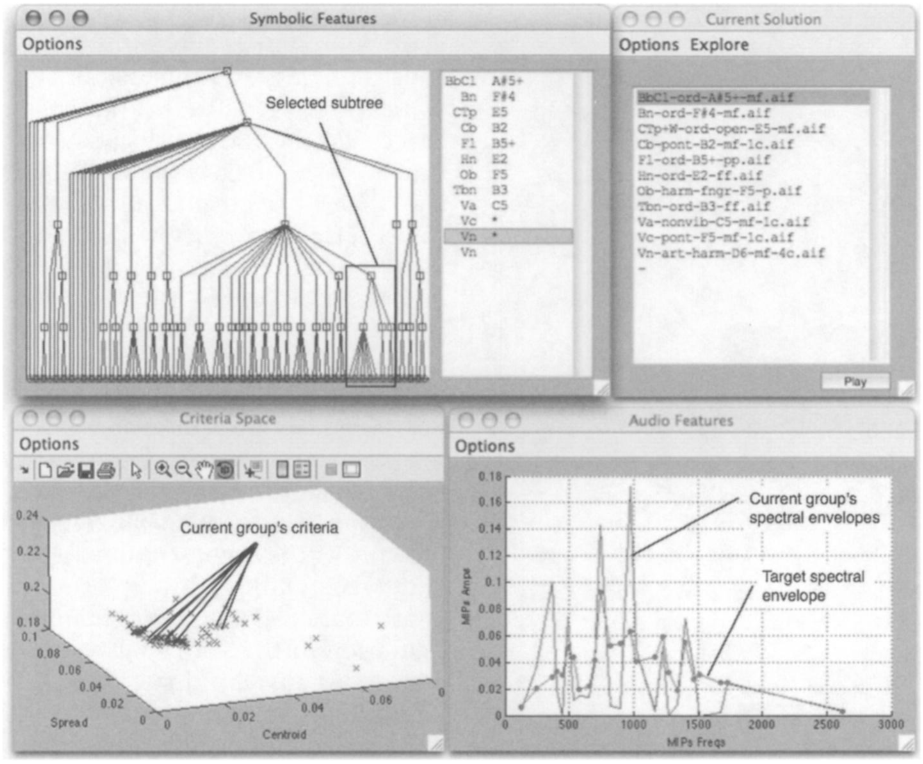
Learning Listening Preferences from User Choices

Last, transformation processes can also be initiated from the criteria space. Assume that a first drawing of the orchestration algorithm returns the Pareto approximation set plotted on Figure 15. Each configuration is *Pareto-optimal*: There is no configuration in the search space explored so far that dominates any solution of the approximation set on both criteria. Solution 1 in Figure 15 improves upon solution 3 on the first criterion but is worse on the second. Now imagine that we ask a user to pick the configuration within this set that is considered closest to the target. It is obvious that choosing solutions in the upper-left side means that objective (a) is prominent, whereas solutions in the lower-right side indicate that objective (b) is more important.

The multicriterion search procedure in Orchidée embeds a preference-modeling mechanism that allows redrawing the algorithm toward a specific direction of the search space. If, for instance, solutions 1, 2, or 3 are picked by the composer as the best, the algorithm will concentrate on region A (or, respectively, B or C) in the next run. Roughly speaking, this is achieved by favoring configurations that dominate the intermediary best solution in the selection step. This choosing/redrawing process can be repeated as many times as necessary to converge on perceptually relevant configurations. Directions of optimization can obviously change over time, depending on the output of each drawing. This is the interactive loop depicted in Figure 3.

Figure 11. An example of multiple viewpoint navigation interface in MATLAB.

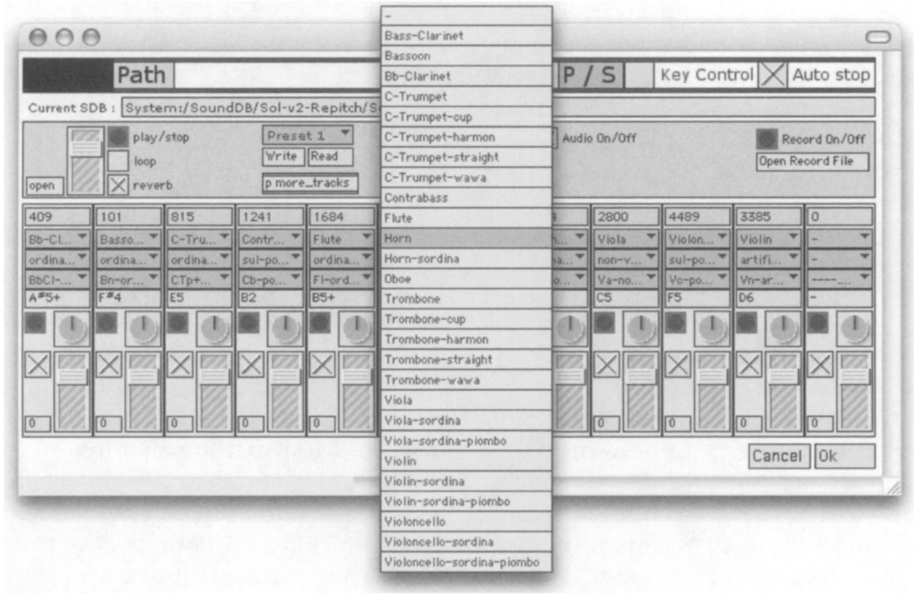
Figure 12. A Max/MSP interface for the manual edition of symbolic parameters.



```
BbCl  A#5+
  Bn  F#4
 CTp  E5
  Cb  B2
  Fl  B5+
  Hn  E2
  Ob  F5
 Tbn  B3
  Va  C5
  Vc  *
  Vn  *
  Vn
```

```
BbCl-ord-A#5+-mf.aif
Bn-ord-F#4-mf.aif
CTp+W-ord-open-E5-mf.aif
Cb-pont-B2-mf-1c.aif
Fl-ord-B5+-pp.aif
Hn-ord-E2-ff.aif
Ob-harm-fngr-F5-p.aif
Tbn-ord-B3-ff.aif
Va-nonvib-C5-mf-1c.aif
Vc-pont-F5-mf-1c.aif
Vn-art-harm-D6-mf-4c.aif
-
```

Figure 11



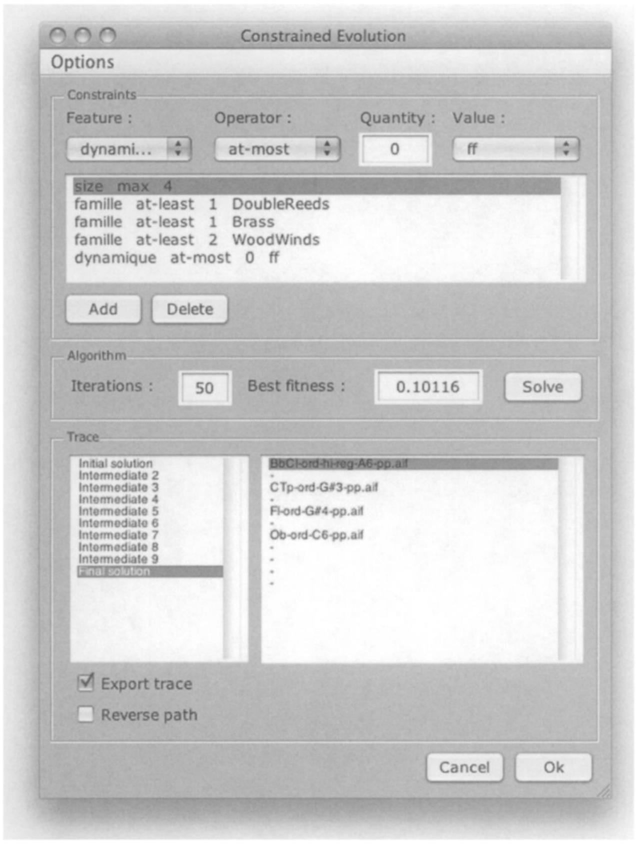
Current SDB : System:/SoundDB/Sol-v2-Repitch/S

```
409       101       815       1241      1684      ...  2800      4489      3385      0
Bb-Cl...  Basso...  C-Tru...  Contr...  Flute     ...  Viola     Violon... Violin    -
ordina... ordina... ordina... sul-po... ordina... ...  non-v...  sul-po... artifi... -
BbCl-...  Bn-or...  CTp+...   Cb-po...  Fl-ord... ...  Va-no...  Vc-po...  Vn-ar...  ----...
A#5+      F#4       E5        B2        B5+       ...  C5        F5        D6        -
```

```
-
Bass-Clarinet
Bassoon
Bb-Clarinet
C-Trumpet
C-Trumpet-cup
C-Trumpet-harmon
C-Trumpet-straight
C-Trumpet-wawa
Contrabass
Flute
Horn
Horn-sordina
Oboe
Trombone
Trombone-cup
Trombone-harmon
Trombone-straight
Trombone-wawa
Viola
Viola-sordina
Viola-sordina-piombo
Violin
Violin-sordina
Violin-sordina-piombo
Violoncello
Violoncello-sordina
Violoncello-sordina-piombo
```

Figure 12

Figure 13. Changing an orchestration played by strings into an orchestration played by woodwinds and brass by changing the constraint structure on the variables of the symbol space.



dynami... at-most 0 ff

```
size max 4
famille at-least 1 DoubleReeds
famille at-least 1 Brass
famille at-least 2 WoodWinds
dynamique at-most 0 ff
```

Iterations : 50 Best fitness : 0.10116

```
Initial solution        BbCl-ord-hi-reg-A6-pp.aif
Intermediate 2          -
Intermediate 3          CTp-ord-G#3-pp.aif
Intermediate 4          -
Intermediate 5          Fl-ord-G#4-pp.aif
Intermediate 6          -
Intermediate 7          Ob-ord-C6-pp.aif
Intermediate 8          -
Intermediate 9          -
Final solution          -
                        -
```

[✓] Export trace
[ ] Reverse path

Conclusion

A formalized approach to the complexity of musical orchestration has been presented. Although the problem of orchestration could be stated as the art of combining instrumental timbres together, a complexity analysis raises the global issue of linking the symbol space of the formal compositional process to the sound space. We have shown in this article that user interaction can be of great help in making symbol and sound worlds communicate.

To this end, we presented an orchestration system based on a client/server architecture that allows this interaction to take place at each different step of the orchestration process. On the server side, the orchestration kernel embeds advanced sound analysis and constrained multicriterion techniques that jointly optimize objectives in symbol and audio spaces. Original features such as the modeling of user preferences or the possibility to investigate different regions of the search space are combined with user interfaces on the client side, providing composers with an orchestration tool that overcomes most of the limitations of the few currently existing systems.

Successful practical results and enthusiastic feedback from composers provide evidence that a significant step has been made in music research. In particular, our collaboration with (among others) composer Jonathan Harvey on his latest piece *Speakings* (performed 19 August 2008 by the BBC Scottish Orchestra, under the direction of Ilan Volkov, in Royal Albert Hall, London) was extremely convincing and encouraging. The orchestral texture at the beginning of the third part of the piece was completely conceived with Orchidée (Nouno et al. 2009). The orchestrated passage suggested by the system was "pasted" into the final score with only minor revisions, though a few changes were necessary to cope with voice leading or practical playing issues. It should be noted that the composer also took into consideration most of the dynamics indications (and their frequent variations) of the Orchidée-generated score. Mr. Harvey even left a note to the director at the beginning of the passage: "Great care to respect the dynamics." Orchidée's innovative techniques were therefore very helpful in finding the finest balance in the dynamics of the instruments.

The Orchidée client developed in OpenMusic is currently being enhanced with new features for the manipulation and exploration of solution spaces. Eventually, the library OM-Orchidée will allow one to easily use and experiment with orchestration processes, fully integrated in more general compositional frameworks. In the future, the client/server Orchidée architecture will allow for the modular development of other specific client interfaces for tackling more and more complex compositional needs. Convincing examples of real-life orchestrations made with Orchidée are available online at http://recherche.ircam.fr/equipes/repmus/carpentier.

Figure 14. Score of the timbre evolution generated in the symbol space by changing the constraint structure (Figure 13). Starting from a chord played by strings (first bar), the sound slightly moves towards the wind choir. Only one instrument changes at each bar.

Figure 15. Guessing listening preferences from user's choices.

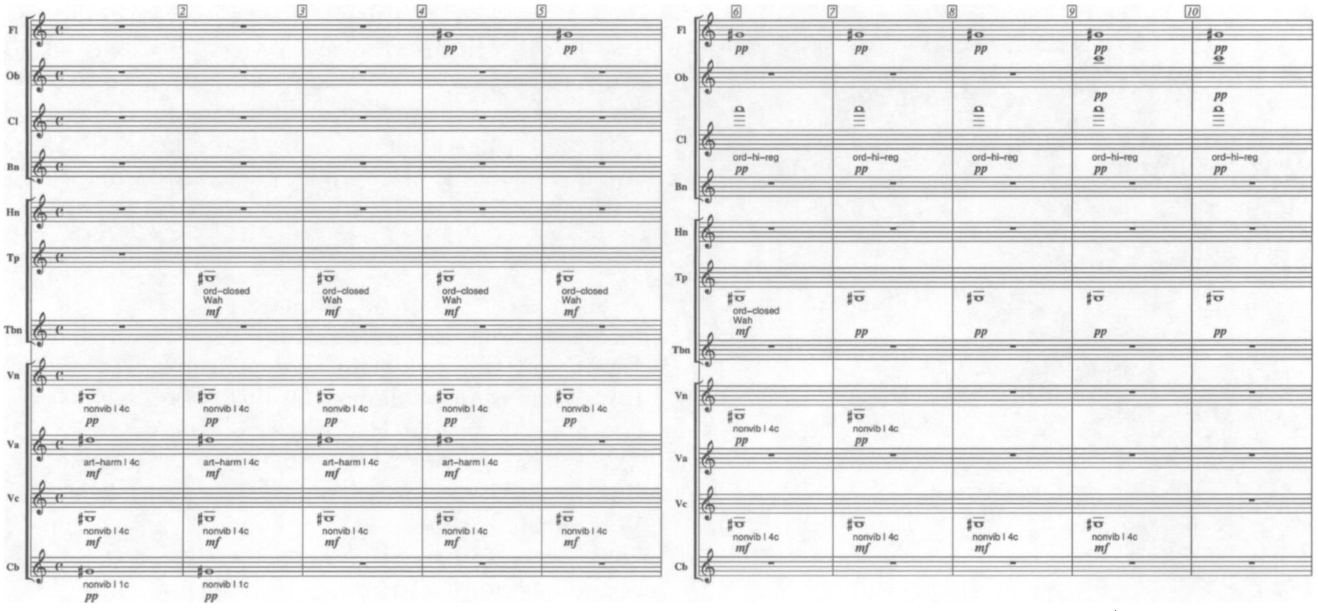
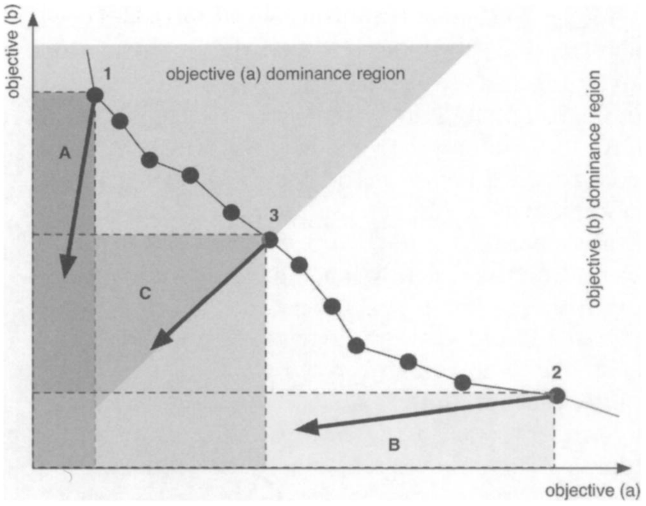


Figure 14



objective (b)
objective (a) dominance region
objective (b) dominance region
objective (a)
1
2
3
A
B
C

Figure 15

Acknowledgments

The authors would like to thank composer Yan Maresz and researchers Damien Tardieu and Hugues Vinet for their daily support in the development of Orchidée. We also would like to mention Georges Bloch, who kindly proofread this article. This work has been partly supported by the French National Research Agency (ANR) in the framework of the Sample Orchestrator project.

References

Adler, S. 1989. *The Study of Orchestration*. New York: Norton.

Agon, C., et al. 2000. "High-Level Musical Control of Sound Synthesis in OpenMusic." *Proceedings of the 2000 International Computer Music Conference*. San Francisco, California: International Computer Music Association, pp. 332–335.

Assayag, G., et al. 1999. "Computer-Assisted Composition at IRCAM: From PatchWork to OpenMusic." *Computer Music Journal* 23(3):59–72.

Berlioz, H. 1843. *Traité d'instrumentation et d'orchestration*. Paris: Henri Lemoine.

Bresson, J., and Agon, C. 2007. "Musical Representation of Sound in Computer-Aided Composition: A Visual Programming Framework." *Journal of New Music Research* 36(4):251–266.

Carpentier, G. 2008. "Approche computationnelle de l'orchestration musicale, optimisation multicritère sous contraintes de combinaisons instrumentales dans de grandes banques de sons." PhD Thesis, University of Paris 6.

Carpentier, G., et al. In press. "Solving the Musical Orchestration Problem using Multicriterion Constrained Optimization with a Genetic Local Search Approach." *Journal of Heuristics.*

Ehrgott, M. 2005. *Multicriteria Optimization,* 2nd ed. Vienna: Springer.

Erickson, R., 1975. *Sound Structure in Music.* Berkeley: University of California Press.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning.* Upper Saddle River, New Jersey: Addison-Wesley.

Hoos, H., and T. Stützle. 2005. *Stochastic Local Search: Foundations and Applications.* San Francisco, California: Morgan Kaufmann.

Hummel, T. 2005. "Simulation of Human Voice Timbre by Orchestration of Acoustic Music Instruments." *Proceedings of the 2005 International Computer Music Conference.* Available online at www.thomashummel.net/english/pdf/simulation.pdf.

Jensen, K. 1999. "Timbre Models of Musical Sounds." PhD Thesis, University of Copenhagen.

Koechlin, C. 1943. *Traité de l'orchestration.* Paris: Max Eschig.

Mathews, M. 1969. *The Technology of Computer Music.* Cambridge, Massachusetts: MIT Press.

McAdams, S., et al. 1995. "Perceptual Scaling of Synthesized Musical Timbres: Common Dimensions, Specificities, and Latent Subject Classes." *Psychological Research* 58:177–192.

McCartney, J. 1996. "Rethinking the Computer Music Language: SuperCollider." *Computer Music Journal* 26(4):61–68.

Nouno, G., et al. 2009. "Making an Orchestra Speak." *Proceedings of the Sound and Music Computing Conference – SMC'09.* Available online at smc2009.smcnetwork.org/programme/pdfs/221.pdf.

Peeters, G. 2004. *A Large Set of Audio Features for Sound Description (Similarity and Classification).* Technical Report, IRCAM. Available online at recherche.ircam.fr/equipes/analyse-synthese/peeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf.

Piston, W. 1955. *Orchestration.* New York: Norton.

Psenicka, D. 2003. "SPORCH: An Algorithm for Orchestration Based on Spectral Analyses of Recorded Sounds." *Proceedings of the 2003 International Computer Music Conference.* Available online at quod.lib.umich.edu/cgi/p/pod/dod-idx?c=icmc;idno=bbp2372.2003.056.

Puckette, M. 1991. "Combining Event and Signal Processing in the MAX Graphical Programming Environment." *Computer Music Journal* 15(3):68–77.

Rimski-Korsakov, N. A. 1912. *Principles of Orchestration.* New York: Maximilian Steinberg.

Rose, F., and J. Hetrick. 2009. "Enhancing Orchestration Technique via Spectrally Based Linear Algebra Methods." *Computer Music Journal* 33(1):32–41.

Tardieu, D. 2008. *Modèles d'instruments pour l'aide à l'orchestration.* PhD Thesis, University of Paris 6.

Tardieu, D., and X. Rodet. 2007. "An Instrument Timbre Model For Computer Aided Orchestration." *Proceedings of the 2007 Workshop on Applications of Signal Processing to Audio and Acoustics.* New York: Institute of Electrical and Electronics Engineers, pp. 347–350.

Wright, M. 2005. "Open Sound Control: An Enabling Technology for Musical Networking." *Organised Sound* 10(3):193–200.