Philippe ESLING

ATO-MS : Abstract Temporal Orchestration - Modular Structure

Software documentation

IRCAM
1, place Igor Stravinsky
F-75004 Paris

ATO-MS : Abstract Temporal Orchestration - Modular Structure,Software documentation

This report was prepared by
Philippe ESLING

Supervisors
Carlos AGON

Philippe ESLING
Equipe Representations Musicales
Institut de Recherche et Coordination Acoustique / Musique (IRCAM)
1, Place Igor Stravinsky
F-75004 Paris
France

www.ircam.fr
Tel: (+33) 6 32 58 91 08
E-mail: esling@ircam.fr

Release date:	03 / 2012
Category:	1 (public)
Edition:	First
Comments:	This report is an internal technical report for IRCAM.
Rights:	©Philippe ESLING, 2012

Contents

Chapter 1

Installation

This is an experimental prototype distribution, this document is intended to help you through the installation steps and usage of this software. This procedure is quite easy but will require a reboot of your computer during the process.

1.1 Installation procedure

1. Install a MySQL Server, the latest OSX 10.6 version is available in this folder, or you can download one for your OS at : http://dev.mysql.com/downloads/mysql/ (For OSX 10.7, you will have to download MySQL for OSX 10.6 - 64 bit)
2. Add the MySQL.prefPane to your preferences panel (it will allow you to quickly check that the SQL server is running at startup), you can also use the startup item package.
3. Reboot your computer
4. Check that the SQL server is running in your preferences
5. Launch the Ato-ms.pkg installation package and follow instructions. Warning : The 'script execution' step is responsible for importing and structuring the knowledge database. Because of the current size of feature knowledge (˜1.5 Gb), this step may take a long time before completion (˜ 5-10 minutes).
6. Start the Ato-me.app in your Applications folder.

1.2 Contents of the package

The installation package should provide the following :

1. Ato-ms server application (*Applications/Ato-ms.app*)
2. Ato-me client application (*Applications/Ato-me.app*)
3. Temporal indexes structure (*Library/Preferences/IRCAM*)
4. SQL database architecture

The sound file is a separate distribution which entails *Studio On Line* (SOL) and *Vienna Symphonic Library* (VSL) sound databases.

1.3 Troubleshooting

If you encounter any problem or the application doesn't seem to work, two files are available to narrow down the problem

- /tmp/atomsInstallReport.txt
- /tmp/atomsInstallOut.txt

If the installation was made successfully but the server does not seem to start, you can try to manually launch the server to see the console output by following this procedure

1. Start the *Ato-me.app* from your Applications folder
2. Open a terminal window
3. Go to the folder root :
   cd /Applications/Ato-ms.app/Contents/MacOS/
4. Run the main script :
   ./atoms.sh

If problems persist, you can send a mail directly to <esling@ircam.fr>

Chapter 2

Main modules

2.1 General view

When launching the orchestration client, the first minimal configuration (cf. Figure 2.1) gradually appears to the screen which contains :

- The ***toolbar*** panel (cf. Section 2.2) which allows to quickly access different features of the client.
- The ***title*** panel which contains the main button for launching an orchestration search.
- The ***server*** panel (cf. Section 2.3) which allows interaction with the computing server.

After launching the *Ato-me* application, the client will automatically launch the orchestration server (*Ato-ms*) and start establishing the communication protocol with it. The whole software being constructed around a modular architecture, when the client and server have established communication, a new panel while appear underneath in order to provide *target* interaction. (cf. Section 2.4). Apparition of this panel also signify that the server is running correctly.

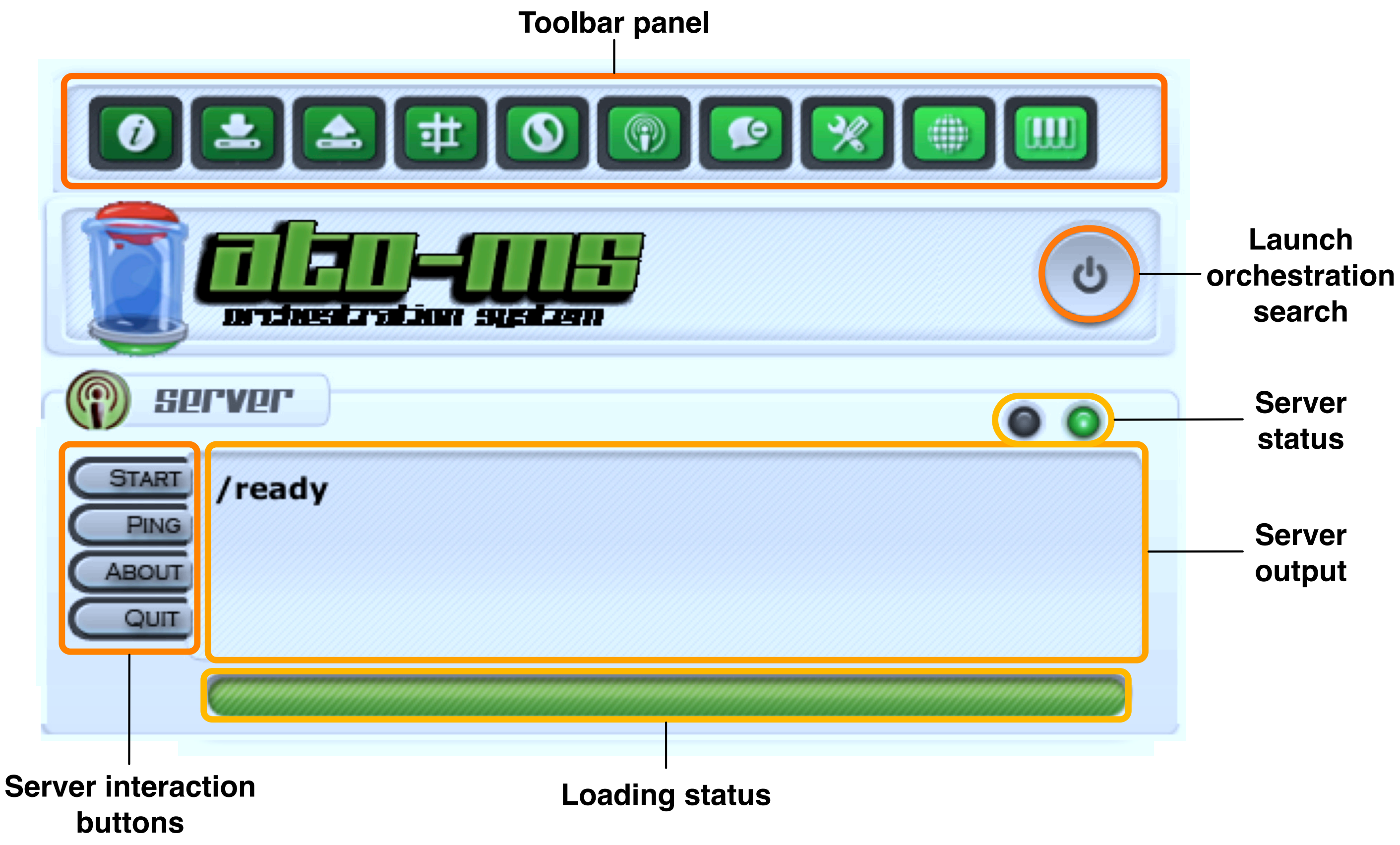
Once every parameters for the search have been set to desired values, hitting the orchestration button will start the search procedure and therefore make a new panel appear on the right to show the *results* of the algorithm. (cf. Section 2.9)

2.2 Toolbar

The toolbar can be found on top of the main panel. It is designed to allow quick and easy access to different parts of the software. Figure 2.2 shows a summary of the meaning for each button. We quickly explain the functionalities related to each button of the toolbar in this section and provide pointers to more detailled explanations inside this document.

Informations

The information button opens a panel which contains the current version number of the software as well as authorship information and contact adress.



/ready

Figure 2.1: First minimal configuration of the client upon launching. It contains the *toolbar* panel (cf. Section 2.2), the *title* panel with a quick button to launch orchestrations and the *server* panel (cf. Section 2.3) which gives the current server state and incoming raw messages.

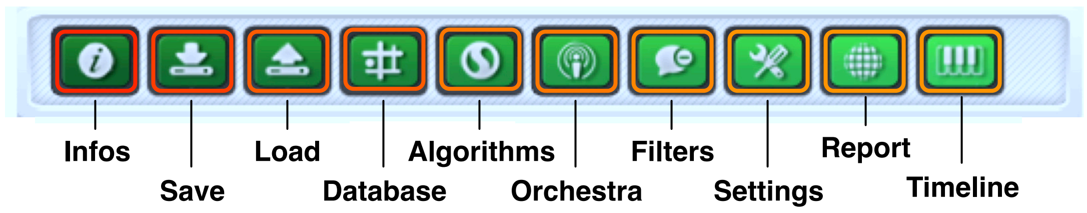


Figure 2.2: Toolbar summary for quick access to different functionalities of the software.

Save Session

Save the current orchestration session (*.orch file*) to a file on the hard drive (cf. Section 4.2) and save the corresponding client state.

Load Session

Load an orchestration session (*.orch file*) from the hard drive (cf. Section 4.2) and restore the corresponding client state.

Database

Opens the database panel which gives access to database inspection and also the temporal query system (cf. Chapter 3).

Algorithms Settings

The algorithm settings panel allows to define which search algorithm to use and also to set its internal parameters (cf. Section 2.8).

Orchestra

The orchestra panel allows to define the instrumental staff that will be used for the orchestration search procedure (cf. Section 2.7).

Filters

The filters allow to reduce the search space by filtering sounds to be used based on their symbolic or spectral attributes (cf. Section 2.6).

Settings

General settings of the system allows mainly to set the audio drivers and to update and define the sound database (cf. Section 2.7).

Report

The embedded report system allows to send bug reports, results or functionnality ideas to a collaborative server (cf. Section 4.3).

Timeline

The timeline object is a first attempt to provide an explicit handling of orchestral macro-articulations. (cf. Section **??**).

2.3 Server

The whole orchestration system is based on a *client / server* architecture.

- The *Ato-me* application acts as client with a user interface which allows to specify the problem (through orchestra, filters and algorithms), set the input objectives (targets and features), select solutions and overall use every possible kind of interaction offered by the system.
- The *Ato-ms* application is the computing core of the system. It is responsible of mainting diverse

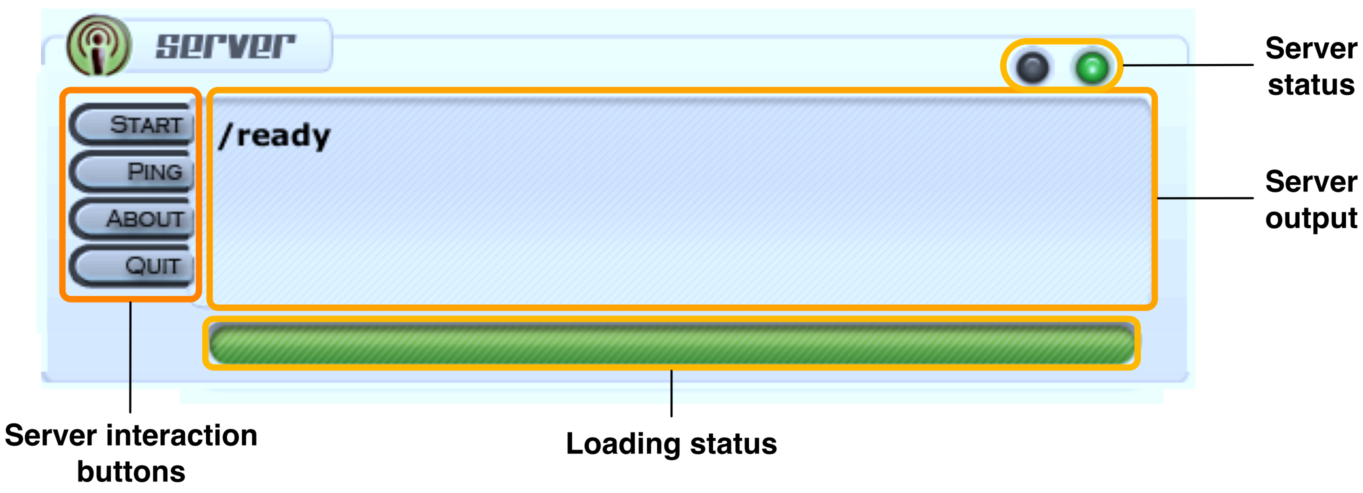


Figure 2.3: The server panel allows to handle communication and interaction with the server. The status is displayed by two leds and a large text window allows to see incoming raw messages.

data structures and allows to run several search algorithms as well as taking into account the different search constraints imposed by the user.

The server interaction can be controlled from the server panel (cf. Figure 2.3). A set of buttons on the left part manage the basic interaction :

Start	Launch the server (ignored if the server is running)
Ping	This send a checking message to the server, to ensure that it is correctly running. The server replies with a */ready* message and both leds blink successively
About	This message retrieves the information message containing the version number of the server
Quit	Shuts down the server (ignored if the server is not running)

On this panel can also be found two leds that indicate the current status of the server. A steady red light indicates that the server is ready to receive information. A blinking red light indicates that the server is running but currently performing computation. Finally a steady red light indicates that the server may be closed or encountered an error. This panel also features a text box which prints all incoming raw messages from the server. Finally a loading bar allows to see the current advance over long computation tasks.

2.4 Target

In the definition of the orchestration problem, the target is the goal to be approximated as closed as possible by the algorithm. It is therefore based on approximating jointly several timbre properties. The advantages of the Ato-ms system is that this approximation can be made separately over temporal shapes, mean values or standard deviations. Furthermore, for the definition of the target (and therefore the features themselves) several possibilities are given to the user.

- Using a single target sound file and selecting its properties
- Using some features from a first target sound file and then, thanks to the ***multi*** mode, use other features from another sound file.
- Modifying any features shapes and values from a target and possibly mix it with original features.

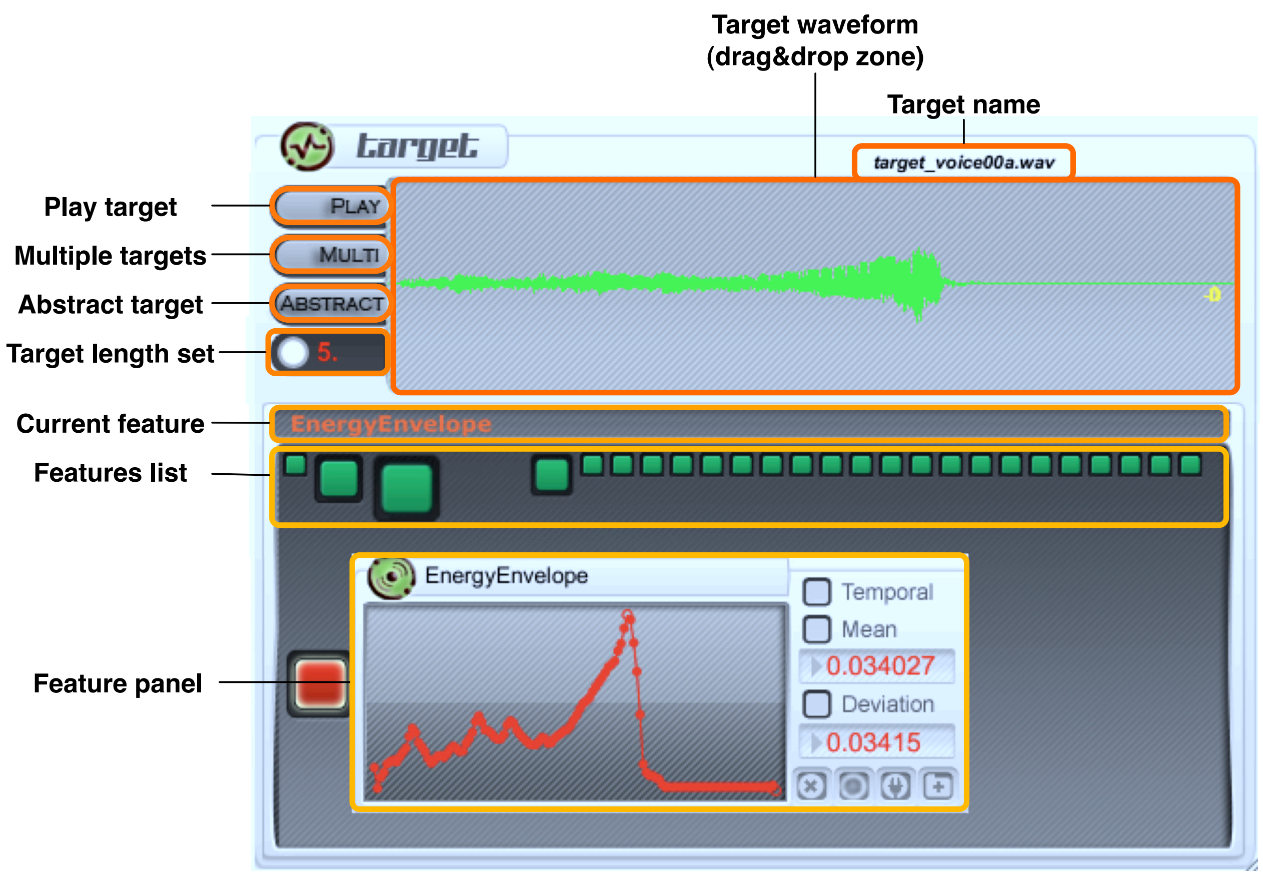


Figure 2.4: The target panel allows to set orchestration objectives either through multiple sound files or by directly drawing desired temporal shapes (*abstract* mode).

- Starting from a void set (***abstract*** target mode) and directly drawing shapes to optimize.

Of course all these modes can be mixed together depending on the specific needs of a user. The target panel is therefore divided into two different parts. (cf. Figure 2.4)

The top part of the target panel contains the main target control options and display the waveform of the last used sound file. The waveform area is also a drag&drop zone, so that users can simply drag their desired soundfiles inside for faster input. The controls on the left provide the following interactions

Play Play the soundfile that has been dragged into the drop zone (ignored otherwise).

Multi This button is an activation switch to enable the multiple targets mode. It works in the following manner :
First use a sound or abstract target (without *multi* engaged) and select some interesting features. Then activate the *multi* mode by pressing this button. Now whenever a new sound or abstract target will be input, previously selected features will be retained as long as the multi button is engaged.

Abstract The abstract mode allows to define a target without the need to provide a sound file. By clicking this button, the system will analyze a segment of silence and provide all spectral descriptors of a second of silence. Starting from there, the user can start drawing new shapes.

Length This define the target length (in seconds) for optimization. This allows another level of flexibility where the original spectral features can be spread over the timeline. This is especially useful for abstract targets where there is a need to impose a sense of temporality.

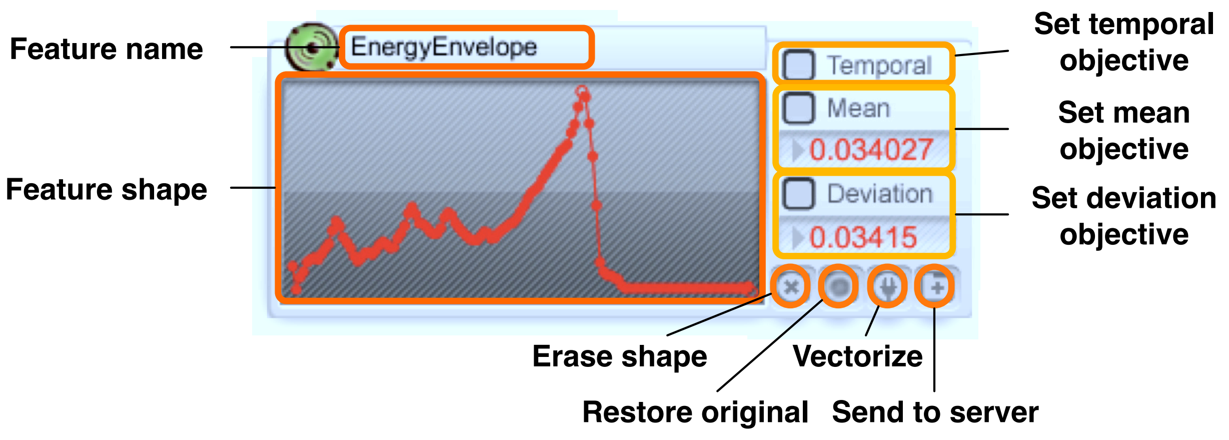


Figure 2.5: Features panels allow to select, modify or input objectives. The corresponding choice can either be to optimize the temporal shape (normalized in the range [-1, 1]), optimize the mean value or optimize the standard deviation. It is also possible to optimize any combinations of these three separately. For each shape, the options at the bottom right allows to edit the shape. The *vectorize* option will simplify the temporal shape by quantifying it to 16 temporal points.

2.5 Features

Features panels allow to select, modify or input spectral feature objectives. The corresponding choice can either be to optimize the temporal shape (normalized in the range [-1, 1]), optimize the mean value or optimize the standard deviation. It is also possible to optimize any combinations of these three separately. Figure 2.5 summarizes the action available from the features panel.

Selecting one of the checkbox on the right indicates that the corresponding feature should be optimized by the search algorithm. The mean and deviation values can be directly chaged from this panel. For each shape, the options at the bottom right allows to edit the shape.

Erase Erase all time points and empty the current temporal shape

Restore Restore the temporal shape from the last feature analysis

Vectorize Simplify the temporal shape by quantifying it to 16 temporal points.

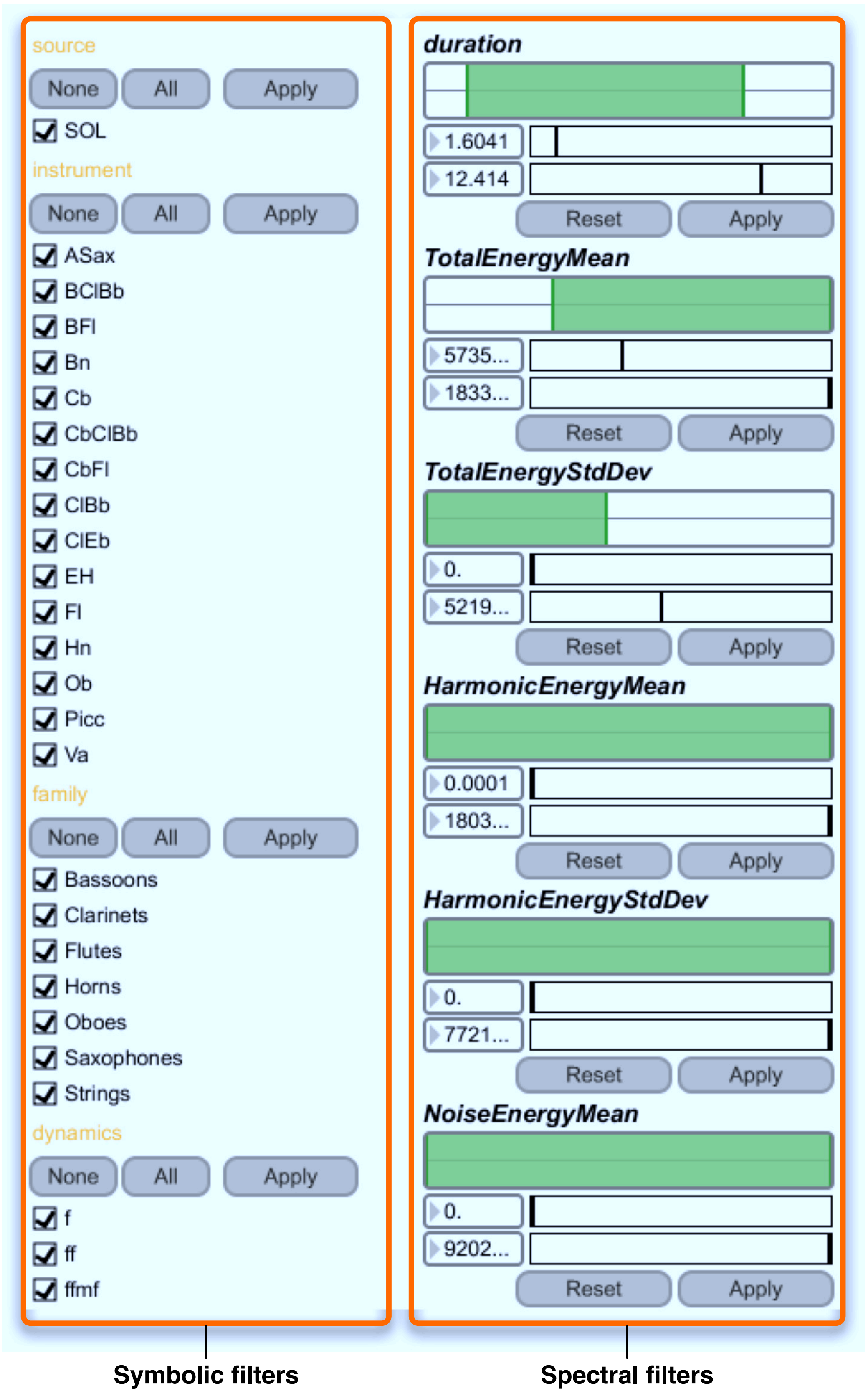
Send This step is ***mandatory*** after changing a temporal shape. This allows to confirm new changes to the temporal shape.

2.6 Filters

Filters are a simple class of constraints which can be imposed on the search procedure. The idea is to constrain the search space to use only certain types of sounds by selecting only the values that should be used throughout the search. The filters panel (cf. Figure 2.6) is divided into the symbolic filters (left) and the spectral filters (right).

Symbolic filters are used to constrain the sounds used based on their symbolic descriptions. (eg. *playingMode*, *note*, *dynamics*). For example, the orchestration solutions can be constrained to use only *mezzoforte* sounds to approximate the target. The top buttons (cf. Figure 2.7) allow to directly select or deselect all values. Each checkbox can be changed independently. Once the modifications are made, the filter must be applied to the server

Spectral filters are used to constrain the sounds used based on the values of their spectral features. For example, the orchestration solutions can be constrained to use only sounds which have a very high deviation in energy (eg. *cresendo*, *decrescendo*, *tremolo* and so forth) to approximate the target. The



source
- ✓ SOL

instrument
- ✓ ASax
- ✓ BClBb
- ✓ BFl
- ✓ Bn
- ✓ Cb
- ✓ CbClBb
- ✓ CbFl
- ✓ ClBb
- ✓ ClEb
- ✓ EH
- ✓ Fl
- ✓ Hn
- ✓ Ob
- ✓ Picc
- ✓ Va

family
- ✓ Bassoons
- ✓ Clarinets
- ✓ Flutes
- ✓ Horns
- ✓ Oboes
- ✓ Saxophones
- ✓ Strings

dynamics
- ✓ f
- ✓ ff
- ✓ ffmf

duration
1.6041
12.414

TotalEnergyMean
5735...
1833...

TotalEnergyStdDev
0.
5219...

HarmonicEnergyMean
0.0001
1803...

HarmonicEnergyStdDev
0.
7721...

NoiseEnergyMean
0.
9202...

Figure 2.6: Filters allows to constrain the search space by selecting only the values that should be used in the search. The filters panel is divided into the symbolic filters (left) and the spectral filters (right)

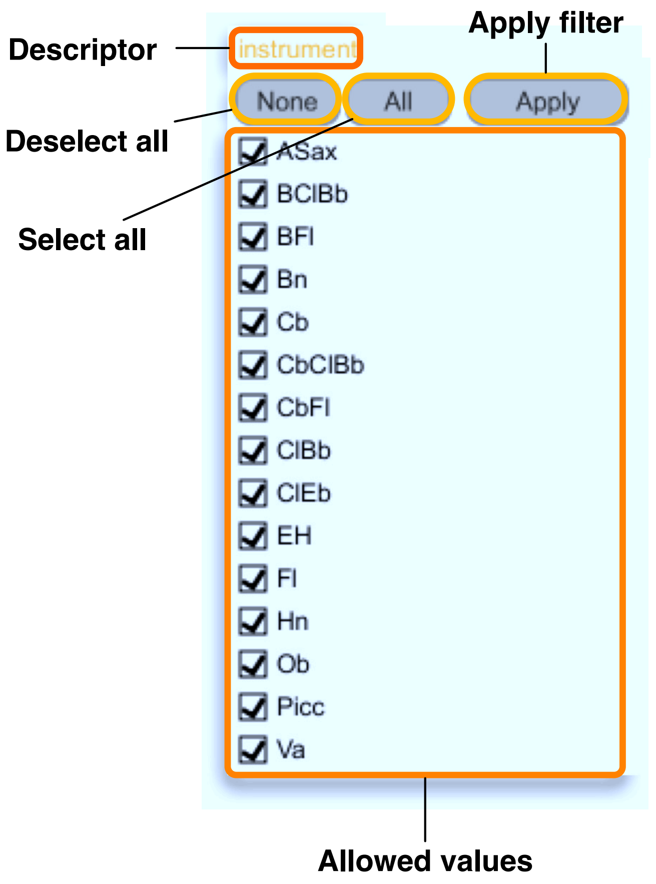


Figure 2.7: Symbolic filters are used to constrain the sounds used based on their symbolic descriptions. The top buttons allow to directly select or deselect all values. Each checkbox can be changed independently. Once the modifications are made, the filter must be applied to the server

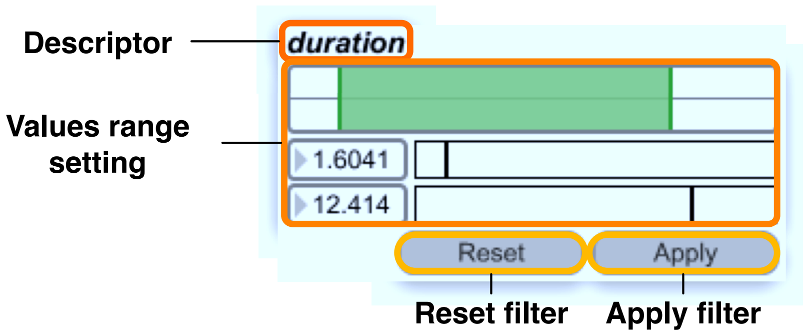


Figure 2.8: Spectral filters are used to constrain the sounds used based on the values of their spectral features. The middle bar allows to directly impose a range of allowed values. These values can be set up more finely by using the two other sliders. Once the modifications are made, the filter must be applied to the server.

middle bar (cf. Figure 2.8) allows to directly impose a range of allowed values. These values can be set up more finely by using the two other sliders. Once the modifications are made, the filter must be applied to the server.

2.7 Orchestra

The orchestra panel allows to define which instruments should be used in the orchestration proposals and their numbers. The top panel (cf. Figure 2.9) provides a preset system which allows to save and load orchestra configurations for faster handling. The interesting part to understand is that the orchestra is defined through its players mainly. Therefore, one player can be assigned to multiple instruments (that will therefore not play at the same time). The user should define an instrumental configuration and how many players follow this configuration and then add them. After adding players, independent remove buttons are available at the right of each player.

2.8 Algorithms

The modular structure of the *Ato-ms* system (cf. Figure 5.1) allows for several completely different search algorithms. That way, the user can select its preference The algorithm panel allows to select which search algorithm should be used as well as its internal parameters. Currently two algorithms are available, but more may be coming. The *Sub-space genetic* algorithm is based on hypervolume-based genetic evolution and the *Optimal Warping* algorithm is based on a combination of multi-objective and time series matching algorithms. The optimal warping search is slower than the previous one but should provide more complex and diverse solutions.

2.9 Results

Once the search procedure has been launched by the user, the *Ato-ms* server will start computation. Meanwhile, a new panel appears in the client, that signify that the search is undergoing. This panel allows to parse through the *current generation* of solutions. As the algorithms are based on evolutionary procedures, several generation will be created and the client automatically updates the solutions based on the advances of the server. However, the user can always go back to older generations even if the proposals will most certainly approximate less closely the target features. The results panel (cf. Figure 2.11) is divided in two distinct parts. The left part allows to parse through solution and see spectral features and optimization distances, while the right part allows to see the symbolic score for each solution.

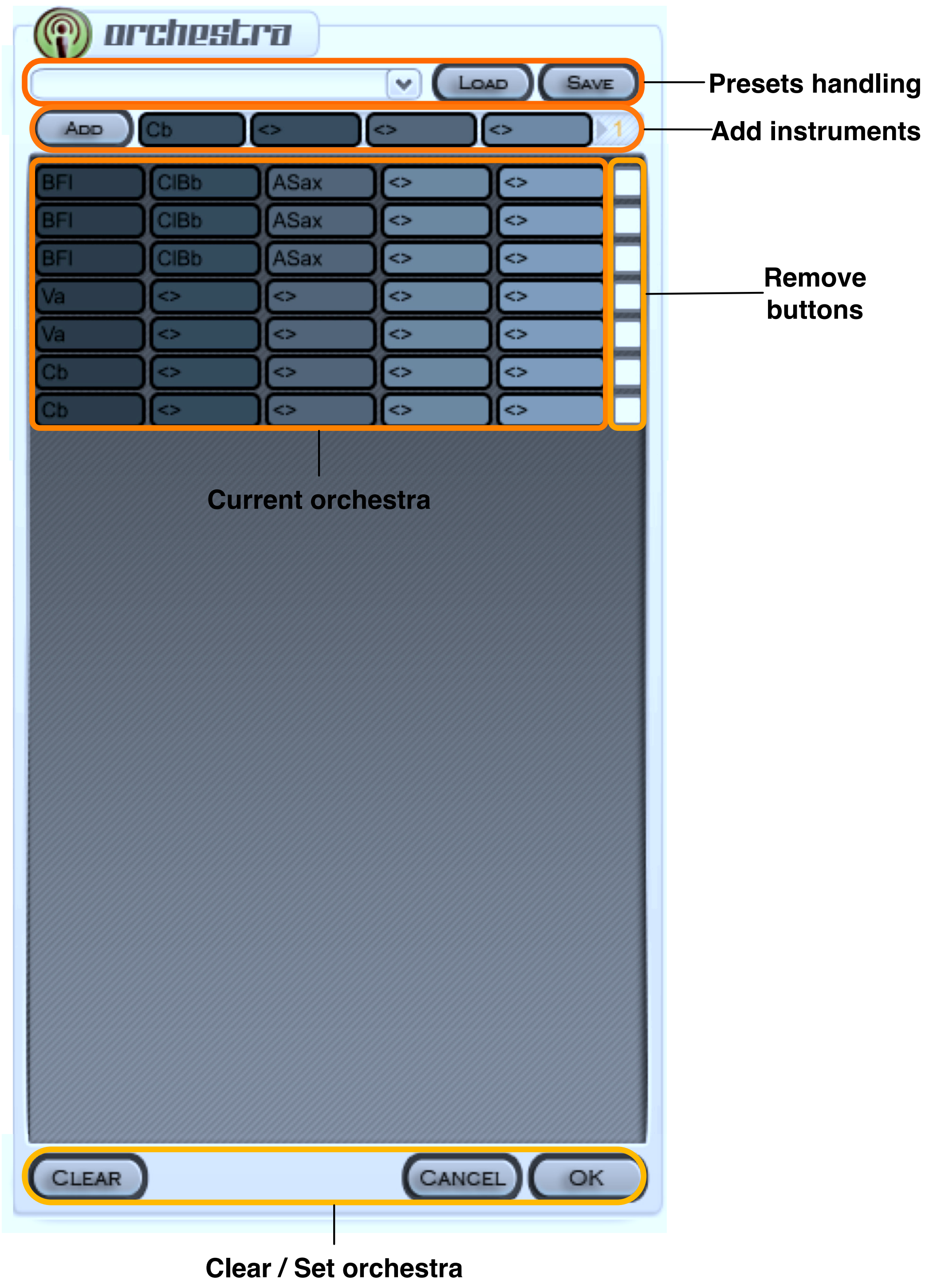


Figure 2.9: The orchestra panel allows to define the players and their instruments. The top part also provides a preset system to load and save orchestra configurations.

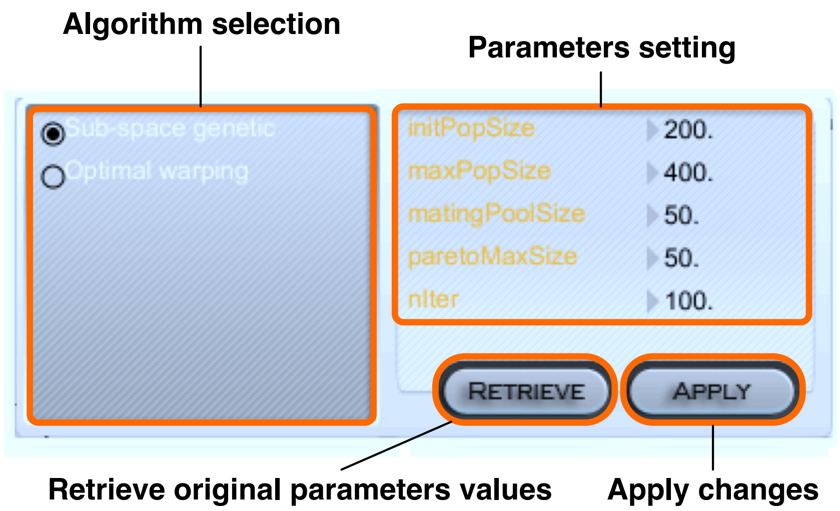


Figure 2.10: The algorithm panel allows to select which search algorithm should be used as well as its internal parameters. Currently two algorithms are available, but more may be coming thanks to the modular architecture of this software.

The left panel (cf. Figure 2.12) allows to see the position of the solutions on the optimization map (and therefore how solutions approximate different objectives more or less). Underneath this map, the user can select two of the spectral descriptors and see the temporal shape of the currently selected solution. Finally, the bottom part allows the user to select any past or current generations and one of the solution inside this generation.

The right part of this panel is a symbolic score representation based on the *Bach* software. It allows to directly see the score corresponding to a solution and (if the sound database is installed on the hard drive) to play the sound mixture straightforwardly. One of the main advantages of this representation is that it can be used in conjunction with the *timeline* object (cf. Section 4.1) that allows to place several solutions on a longer time scale and therefore experiment with orchestral macro-articulations. It is therefore possible to copy, paste, add or modify any part of this representation. Furthermore this representation includes an automatic symbolic search system. This allows for example to change the pitch, duration or dynamics of any note and the system will find the corresponding sound file in the database. This can allow full transposition or rescaling of the solutions. The bottom part include interaction buttons to modify the representaiton

X Perform temporal rescaling of the score

Y Perform height rescaling of the score

Autozoom (switch with *manual*) If the autozoom is engaged, the representation will automatically perform a rescaling of the solution on the X and Y dimensions.

Empty (swich with *leave*) When this option is turned on, the unused instruments are removed from the score. This allows faster parsing of the solutions, however it is advise to left the empty staves when using in conjunction with the timeline (to keep the right order of instruments.

Play (switch with *autoplay*) When this option is turned on, the solutions will be played simultaneously while parsing the map.

2.10 Settings

The settings panel provides interaction with the basic settings of the system. It is therefore possible for the user to set

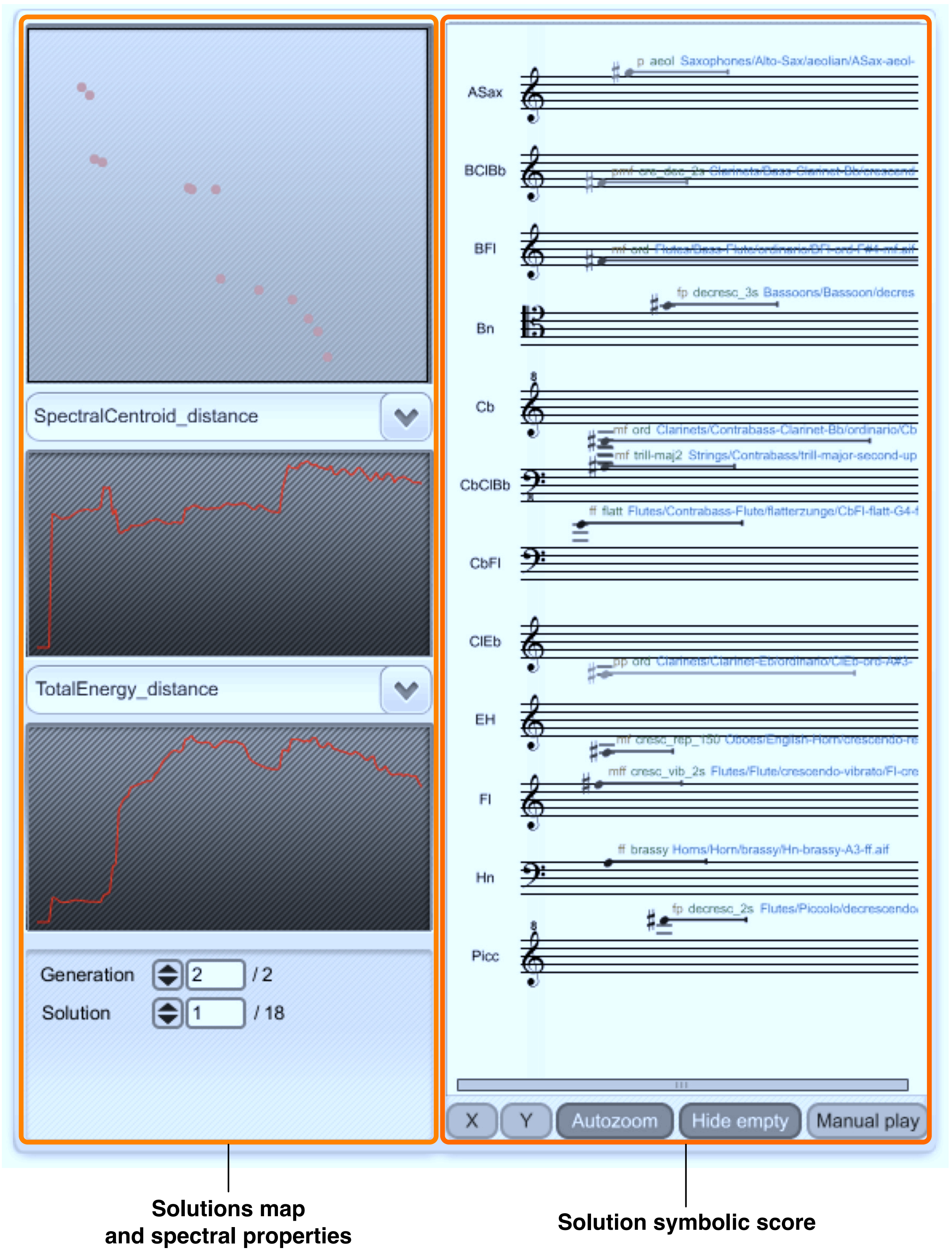
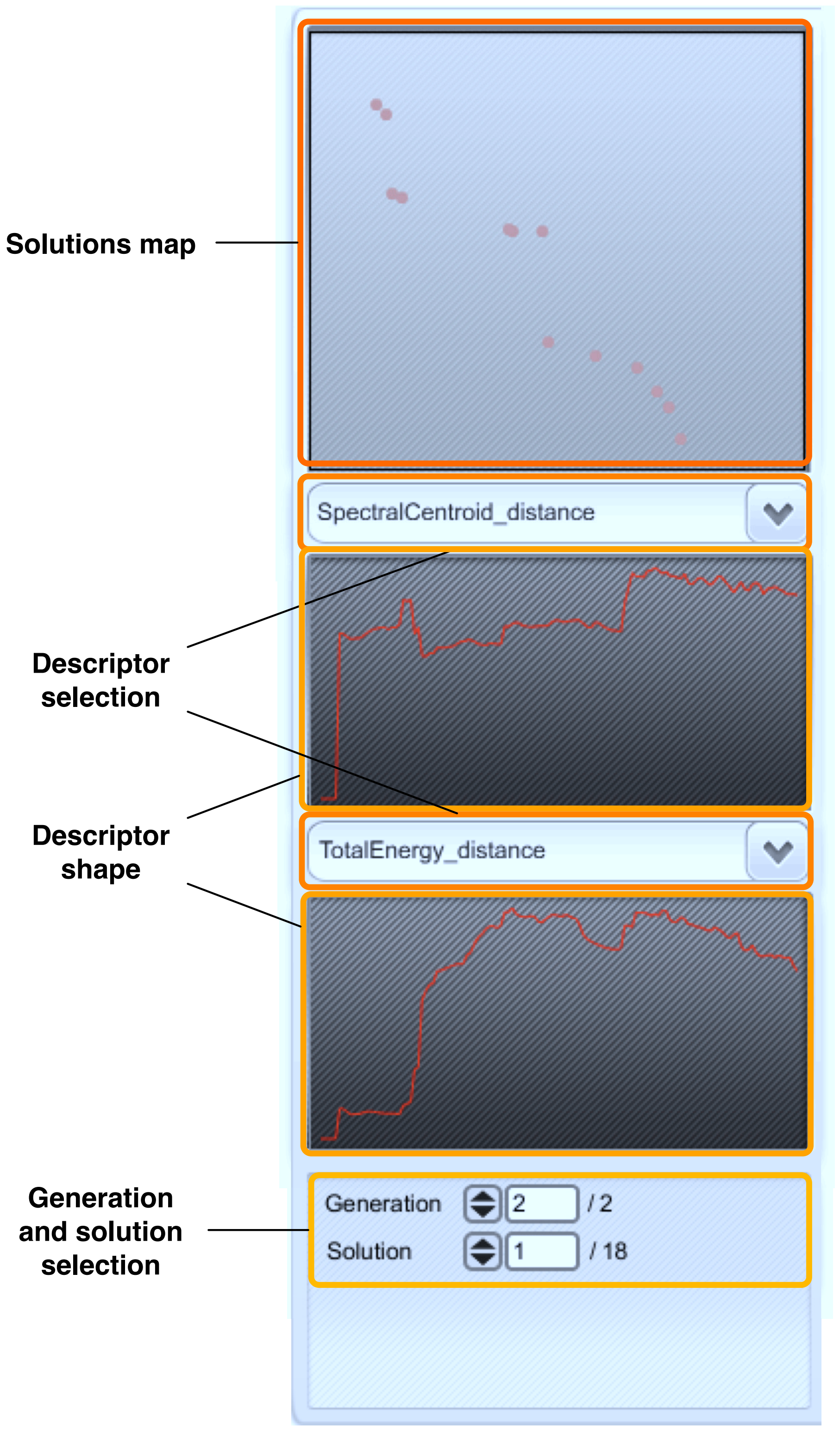

SpectralCentroid_distance
TotalEnergy_distance
Generation 2 / 2
Solution 1 / 18
ASax
BClBb
BFl
Bn
Cb
CbClBb
CbFl
ClEb
EH
Fl
Hn
Picc
p aeol Saxophones/Alto-Sax/aeolian/ASax-aeol-
fp decresc_3s Bassoons/Bassoon/decres
mf ord Clarinets/Contrabass-Clarinet-Bb/ordinario/Cb
mf trill-maj2 Strings/Contrabass/trill-major-second-up
ff flatt Flutes/Contrabass-Flute/flatterzunge/CbFl-flatt-G4-f
pp ord Clarinets/Clarinet-Eb/ordinario/ClEb-ord-A#3-
mf cresc_rep_150 Oboes/English-Horn/crescendo-re
mff cresc_vib_2s Flutes/Flute/crescendo-vibrato/Fl-cre
ff brassy Horns/Horn/brassy/Hn-brassy-A3-ff.aif
fp decresc_2s Flutes/Piccolo/decrescendo
X
Y
Autozoom
Hide empty
Manual play
Solutions map
and spectral properties
Solution symbolic score

Figure 2.11: The results panel allows to parse through the orchestration proposals in real time.



Generation 2 / 2
Solution 1 / 18

Figure 2.12: The left panel shows optimization information, spectral descriptor and solution parsing.

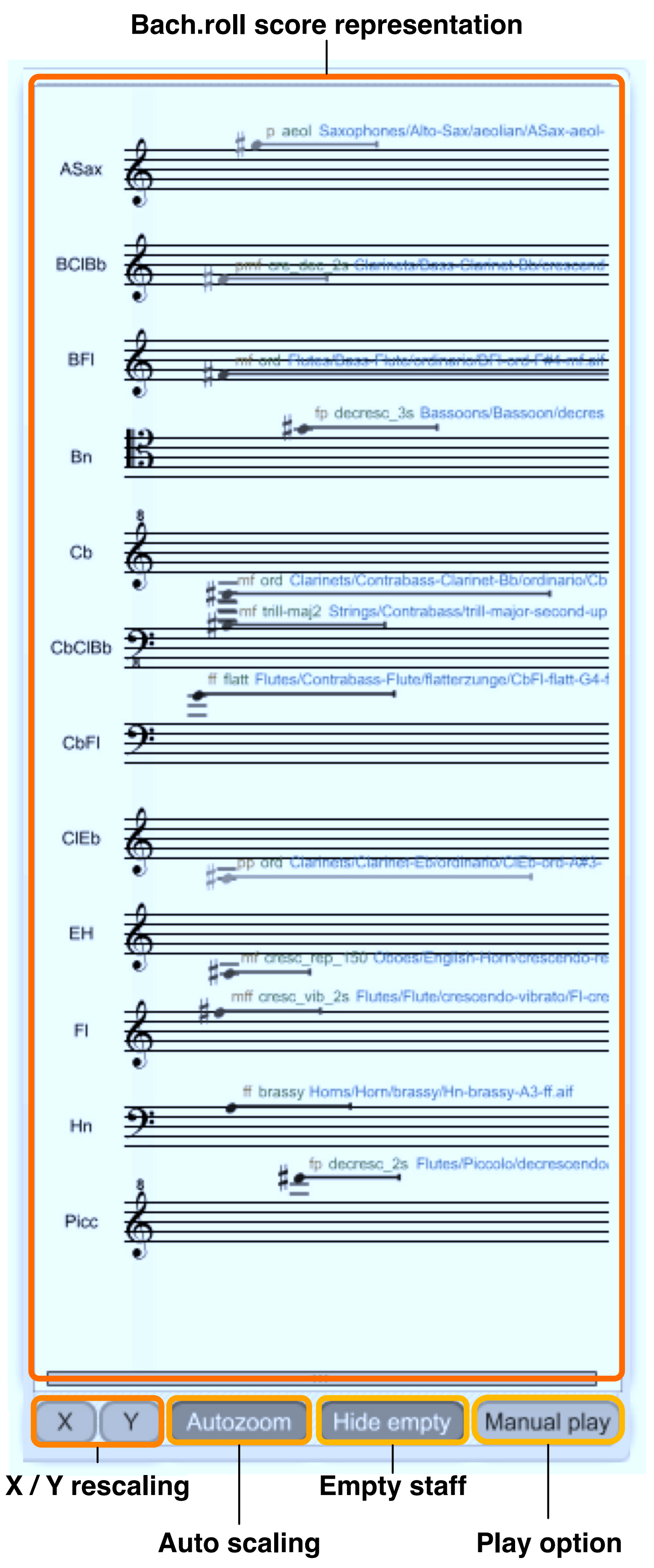

Bach.roll score representation
ASax
p aeol Saxophones/Alto-Sax/aeolian/ASax-aeol-
BClBb
BFl
Bn
fp decresc_3s Bassoons/Bassoon/decres
Cb
mf ord Clarinets/Contrabass-Clarinet-Bb/ordinario/Cb
mf trill-maj2 Strings/Contrabass/trill-major-second-up
CbClBb
ff flatt Flutes/Contrabass-Flute/flatterzunge/CbFl-flatt-G4-
CbFl
ClEb
pp ord Clarinets/Clarinet-Eb/ordinario/ClEb-ord-A#3-
EH
mf cresc_rep_150 Oboes/English-Horn/crescendo-re
mff cresc_vib_2s Flutes/Flute/crescendo-vibrato/Fl-cre
Fl
ff brassy Horns/Horn/brassy/Hn-brassy-A3-ff.aif
Hn
fp decresc_2s Flutes/Piccolo/decrescendo
Picc
X
Y
Autozoom
Hide empty
Manual play
X / Y rescaling
Empty staff
Auto scaling
Play option

Figure 2.13: The score representation allows to directly see the symbolic representation of solutions. The bottom buttons can modify the aspect of this representation.

Database

Root	This allows to change the root directory where to find the sound files database. This step is mandatory if the user will to play and hear the different solutions proposed by the system.
Update	When the user adds new sound files, this allows to compute their spectral descriptor and add their features to the search process.
Import	This allows to directly import a compressed database file (.db file) which follows the right SQL architecture.
Create	This function allows to create a new database from scratch.

Server

Harmonic	Harmonic filtering allows to restrain the search space by automatically deriving the note used from the partials analysis.
Print	When this option is turned on, all incoming OSC messages will be print in the Max window.
DSP	This gives access to audio drivers and related settings.

Chapter 3

Database querying

In addition to the orchestration search algorithms, the *Ato-ms* features an innovative “intelligent database” system. This system allows to perform queries for sound files based on the temporal evolution of their spectral descriptors. Therefore, when searching for specific properties of a sound, it is possible to directly draw the desired temporal shapes of a specific spectral descriptor and the system will retrieve the six top matches from the database.

3.1 Introduction

From the toolbar, it is possible to access this system by clicking on the database icon. This opens the first database panel (cf. Figure 3.1) that is divided into three parts depending on the type of queries that can be made

Values	This query panel allows a simple parsing of database fields, queryable fields and the corresponding values. (cf. Section 3.2.1)
Temporal	This panel is used for temporal shape querying based on different spectral descriptors. (cf. Section 3.2.3)
Sample	This part allows to check the samples inside the database depending on the instrument, note and playing style. (cf. Section 3.2.2)

3.2 Querying

We detail in this section each type of querying available.

3.2.1 Values querying

The values querying panel is intended to have a quick lookout over every existing values in the database.

Fields	This allows to see all fields in the database (even those on which queries cannot be performed)

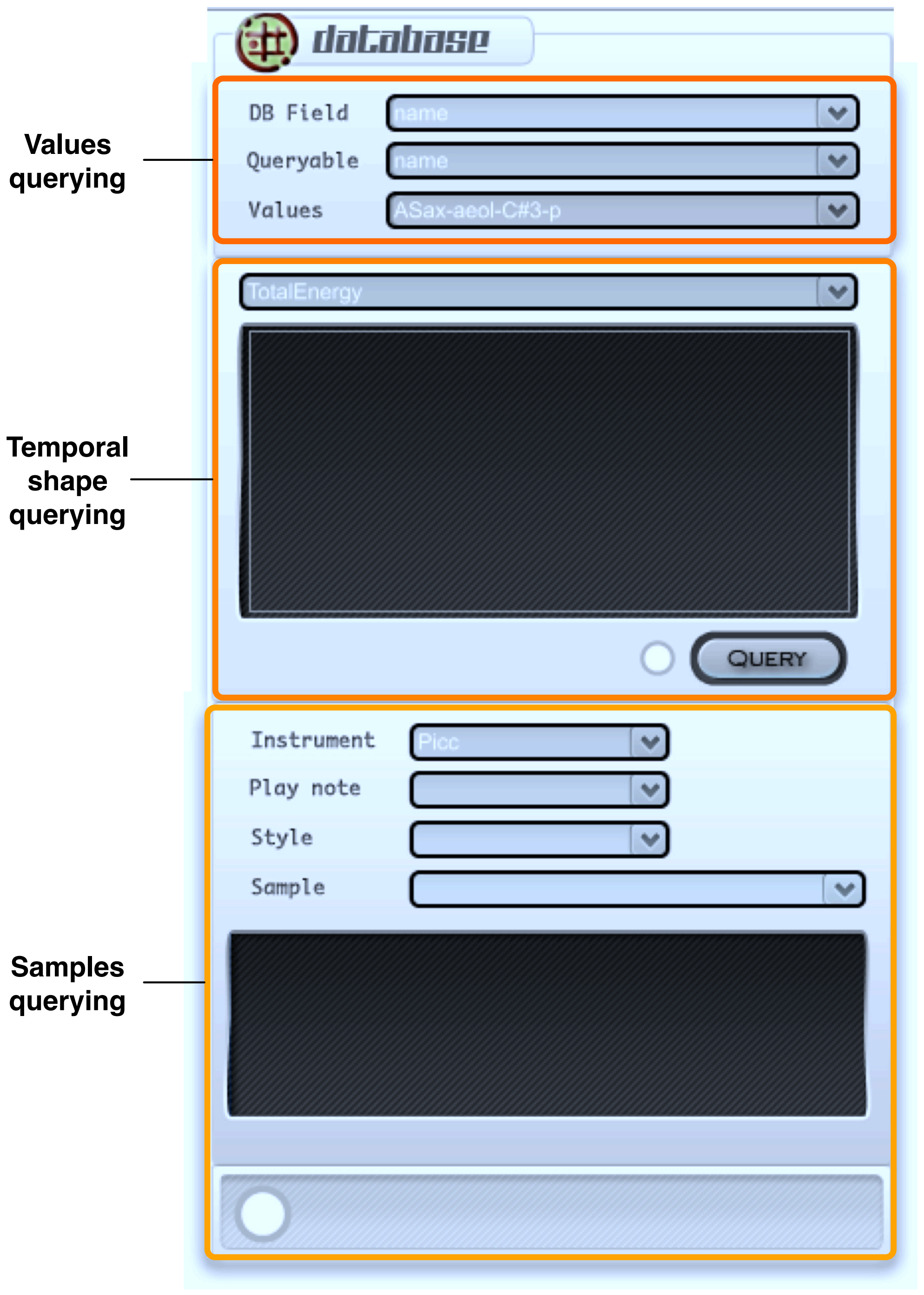


Figure 3.1: The database panel gives access to three different sub-systems for querying

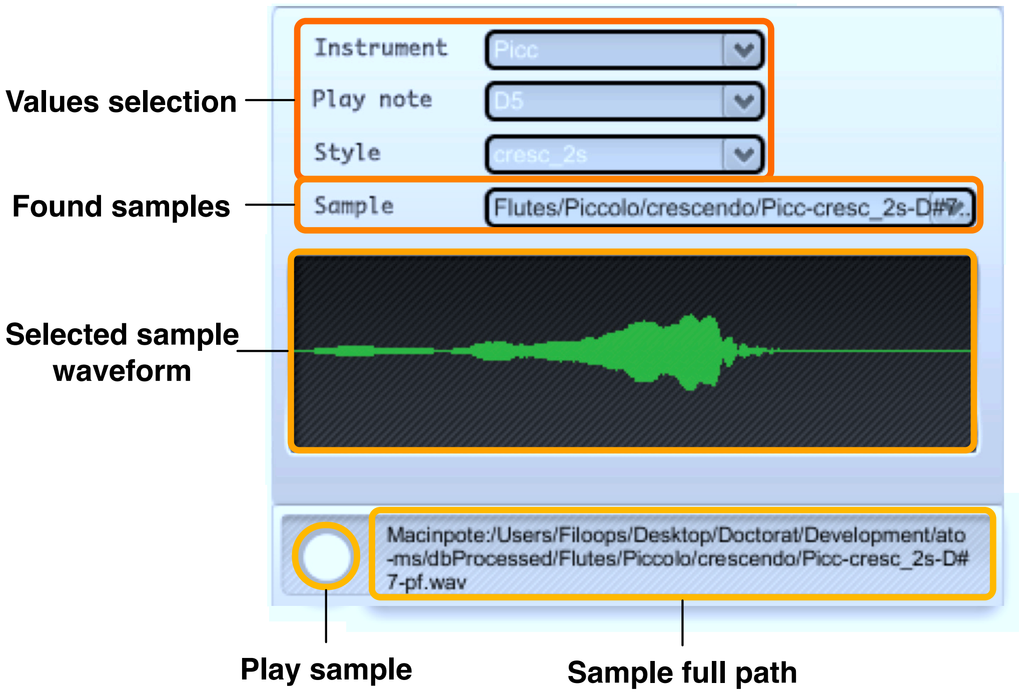

Values selection
Instrument
Picc
Play note
D5
Style
cresc_2s
Found samples
Sample
Flutes/Piccolo/crescendo/Picc-cresc_2s-D#7...
Selected sample waveform
Macinpote:/Users/Filoops/Desktop/Doctorat/Development/ato-ms/dbProcessed/Flutes/Piccolo/crescendo/Picc-cresc_2s-D#7-pf.wav
Play sample
Sample full path

Figure 3.2: The samples querying panel allows to parse through all the samples in the sound database based on symbolic categories.

Queryable This field allows to parse only queryable fields of the database. When a descriptor is selected, all its unique values are displayed in the next field.

Values This field shows all existing values for the last descriptor selected in the *Queryable* field.

3.2.2 Samples querying

The samples querying panel (cf. Figure 3.2) allows to find the samples based on their symbolic categories. Therefore, the first fields allow to select, the instrument, its note and playing style. Based on these informations, several samples are proposed in the corresponding field mostly based on different dynamics and variations of the sample. Finally, the bottom part of the panel prints the full path where the sample can be found and the left button allows to play the corresponding sample.

3.2.3 Temporal querying

The temporal querying panel (cf. Figure 3.3) is based on an innovative "intelligent database" system. This system allows to perform queries for sound files based on the temporal evolution of their spectral descriptors. Therefore, when searching for specific properties of a sound, it is possible to directly draw the desired temporal shapes of a specific spectral descriptor and the system will retrieve the six top matches from the database. The top field of this panel allows to select the desired spectral descriptor. The middle part is based on a breakpoint function and allows to draw any type of temporal evolution required. Finally the bottom of this panel allows to erase the temporal shape or launch the query.

3.3 Results

When a temporal query is performed on the database, the panel automatically expands to show the six top matches of the query and their related spectral features. (cf. Figure The results are the best in terms of time series distances but are shown in alphabetic order.

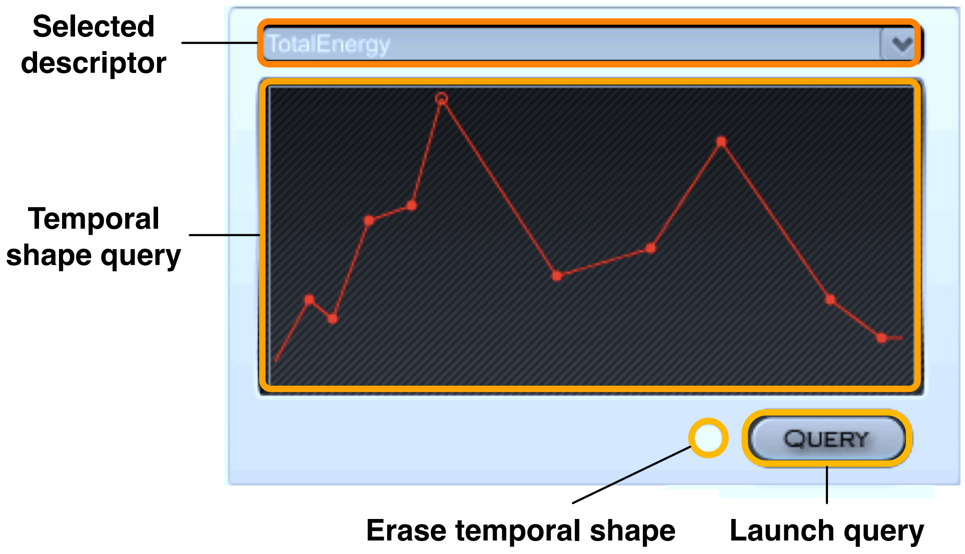
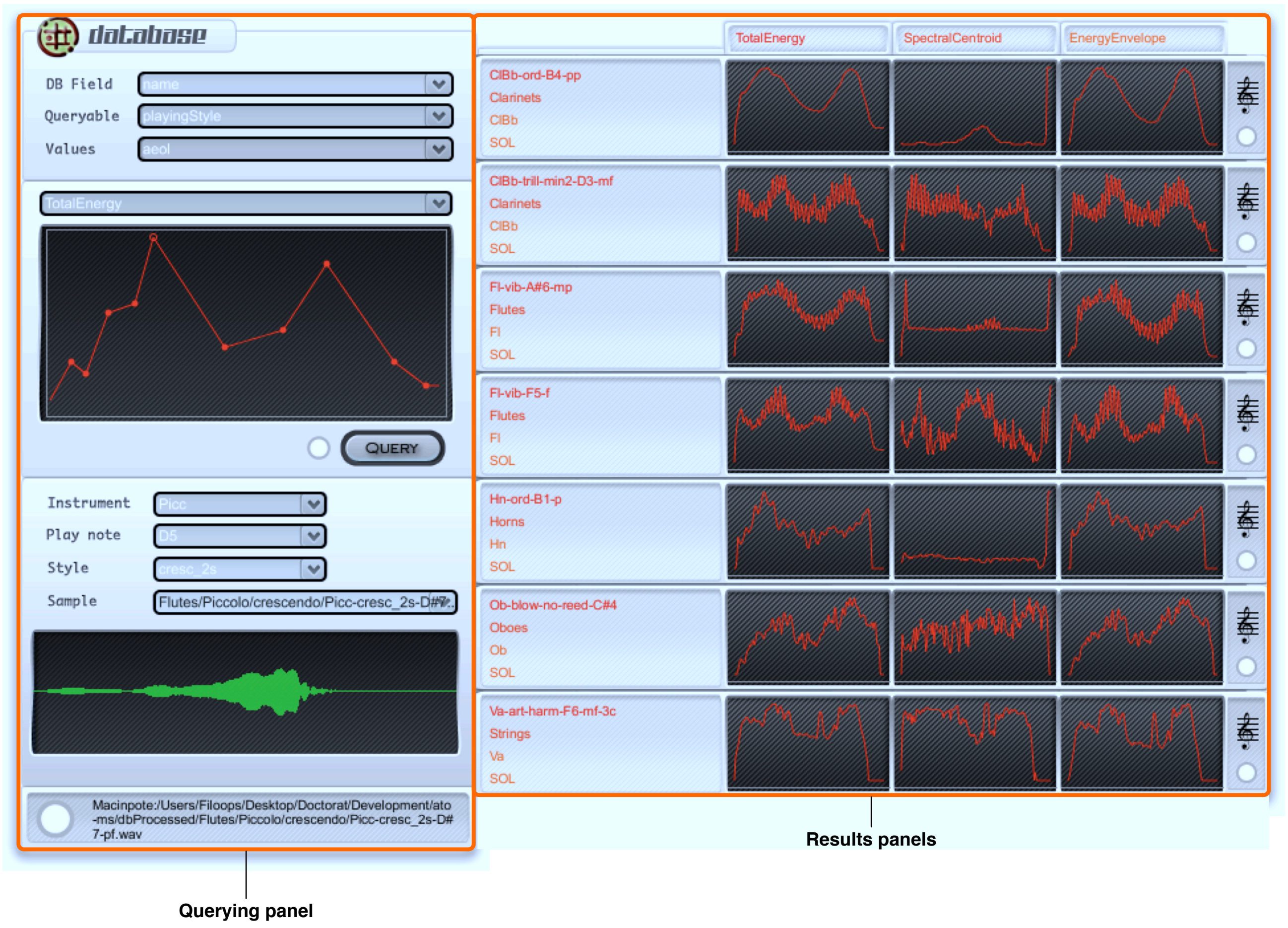


Figure 3.3: The temporal query panel allows to perform queries based on the temporal shapes of spectral descriptors.



DB Field: name
Queryable: playingStyle
Values: aeol

TotalEnergy

Instrument: Picc
Play note: D5
Style: cresc_2s
Sample: Flutes/Piccolo/crescendo/Picc-cresc_2s-D#7...

Macinpote:/Users/Filoops/Desktop/Doctorat/Development/ato-ms/dbProcessed/Flutes/Piccolo/crescendo/Picc-cresc_2s-D#7-pf.wav

	TotalEnergy	SpectralCentroid	EnergyEnvelope
ClBb-ord-B4-pp Clarinets ClBb SOL			
ClBb-trill-min2-D3-mf Clarinets ClBb SOL			
Fl-vib-A#6-mp Flutes Fl SOL			
Fl-vib-F5-f Flutes Fl SOL			
Hn-ord-B1-p Horns Hn SOL			
Ob-blow-no-reed-C#4 Oboes Ob SOL			
Va-art-harm-F6-mf-3c Strings Va SOL			

Figure 3.4: Once a temporal query has been issued on the database, the database panel automatically expands to show the six top matches in alphabetic order.

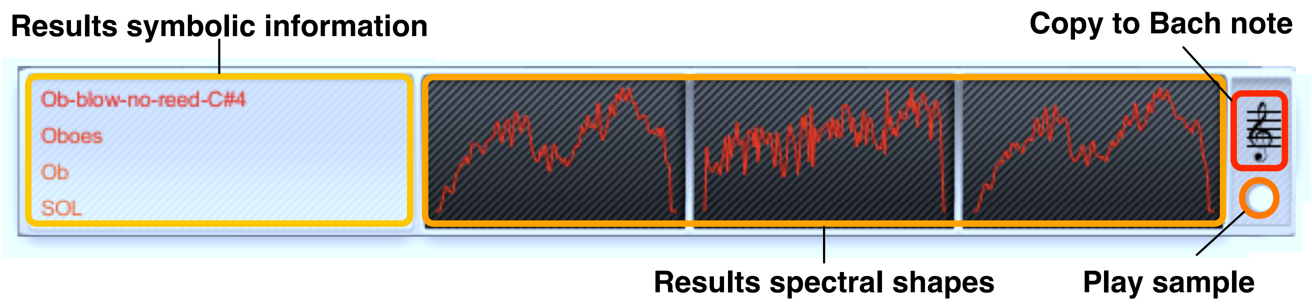


Figure 3.5: A single result window provides symbolic information, spectral shape and interaction controls.

For each result are provide symbolic and spectral information as well as interaction controls. (cf. Figure 3.5)

Symbolic The sample name, its instrument, family, and database source are displayed on the left panel

Spectral The temporal shape of the selected spectral descriptor as well as two other features are displayed

Bach This control is very useful to add single elements to orchestration (and especially to the timeline). It is in fact a mini-roll that contains the related match. The user has to click once on this part and then use the common Copy action. This element can then be pasted inside the timeline.

Play Plays the corresponding sample of the match

Chapter 4

Other features

4.1 Timeline

When accessed through the toolbar, an extended score representation is opened. This timeline can be seen as a full sketch of potential orchestrations. As it is based on the *Bach* system, it embeds all the functionnalities offered by this system.

Copy	The solutions found in an orchestration search represented in the results panel (cf. Section 2.9) can be completely or partly copied and then pasted in this timeline. Matches found by the temporal query system (cf. Section 3.2.3) can also be copied and pasted.
Play	The whole score can be played in the same way as the results section (spacebar key).
Export	The export system from Bach is available
Slots	A slot system allows to embed synthesis effects

For more information on all the functionalities for the timeline, we refer the interested reader to the *Bach* documentation.

4.2 Saving and loading sessions

Because of the heavy load of settings and parameters involved in the orchestration search process, it may come handy to save and load complete sessions. This functionnality can be found directly in the toolbar. When saving a session, a compressed *.orch* file is created which will contain the orchestra, filters, target and solutions at the save point. The user can later load this file and the client will automatically re-update its interface to show all saved parameters.

4.3 Report system

In order to facilitate the exchange of bug reports, exchange of idea and collaborative thinking, the *Ato-ms* system embeds an automatic report system (cf. Figure 4.1). As we encourage any kind of idea exchange, the report can be on Errors, Bad / Good results, Desired features, Idea, Remarks and any other topic. The reports from every users can be seen at the address *:*
http://groups.google.com/group/ircam-orchestration

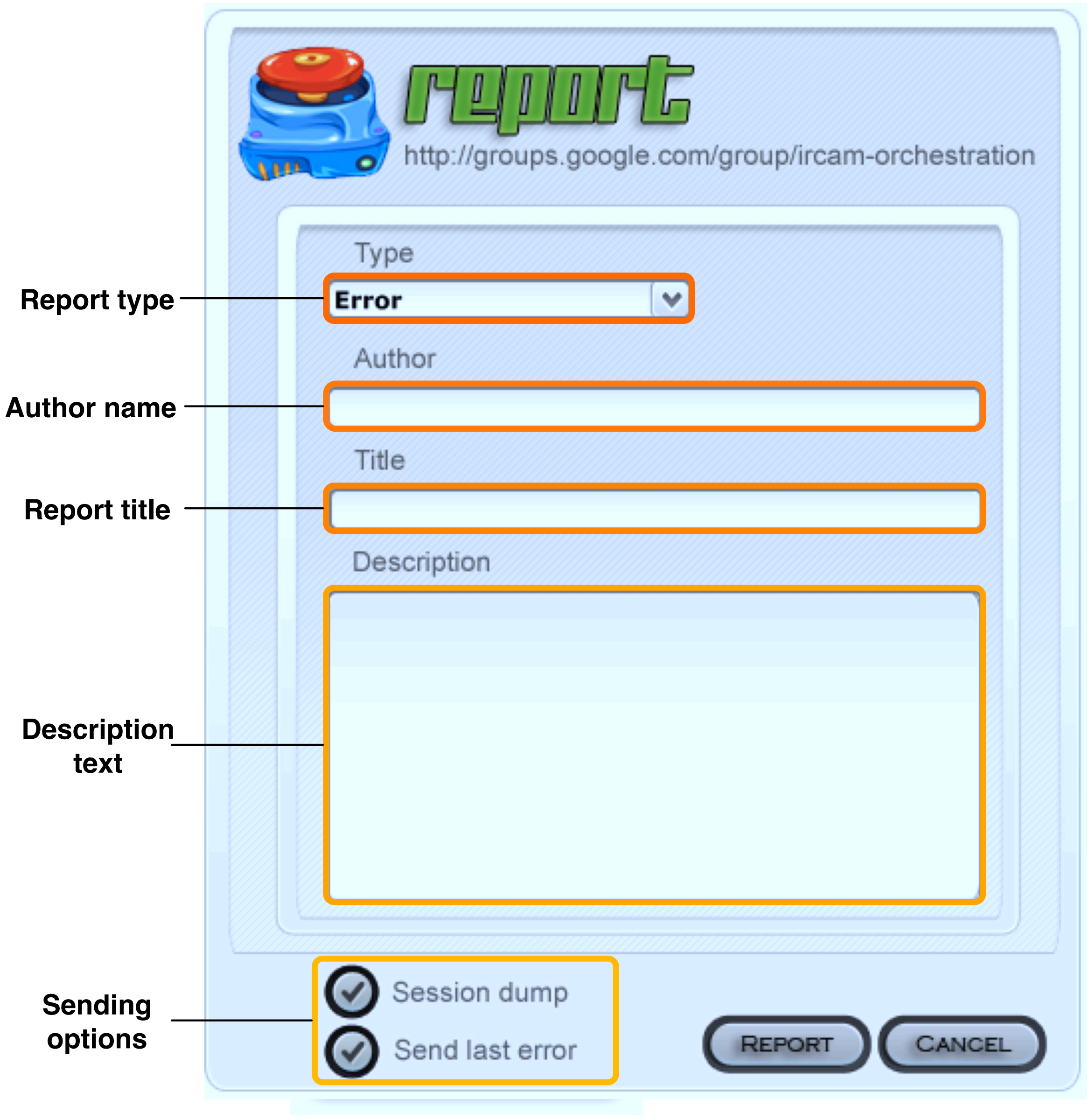


Figure 4.1: The report system allows to exhibit bugs, share ideas or point out good or bad results.

Type	Generic categories allow to quickly state the type of report
Author	The reports can be anonymous or use a pseudonym or mail adress
Title	Please use a descriptive title
Description	Free text box to describe the report
Dump	If this option is checked, the report system will automatically save a session (.orch) file and send it along with the report. That way, anyone interested by a report can immediately open the session save and by opening it, will have the orchestration system in exactly the same state as the author of the report when he created it.
Error	Add to the description text the last error found by the system (this option is automatically performed when "report" is clicked in an error box).

Chapter 5

Matlab source code

5.1 Overview

ATO-MS is a multi-objective optimization system which allows to use heuristics to find relevant sound combinations that match a target. It has been re-thinked to feature completely new interaction principles and has been completely coded from scratch. Some module are intentionally left blank to open up future works on the topic. Finally, it can be seen that any module from the server can be completely extended in a novel way to include new interaction principles.

One of the main limitation rising from previous orchestration systems was the lack of temporal modelization even at the smallest scales, which is unfortunate as temporal structures are at the core of our auditory perception. This was the first motiviation to create a new orchestration system that could account for the temporal evolution of spectral properties. The solutions are now time-aware and instruments can have different onsets to better match the temporal evolution of sound descriptors.

Second, the notion of sound target was somehow too limiting as this target is sometimes unavailable to users. The abstract target part of this new system allows to directly input temporal shapes as well as mean and deviation values for sound descriptors to be optimized. However, the soundfile version of the target is still available and has now been greatly extended by the possible use of *multiple* sound targets. It is therefore possible to optimize jointly some descriptors of a first target with other descriptors of a second. The number of target is unlimited (given the possible number of descriptors).

Finally previous implementations were based on a linear modelization which didn't allowed any flexibility for further developments. The new system has been architectured around an *Object-Oriented Programming* conceptualization. Therefore, the system in itself is an extensible and modular structure which can allow modification, extension, removal and even complete introduction of tasks and resolution methods. Furthermore, this new implementation is based on a SQL database which allows to perform temporal shape matching and also allow a potentially infinite knowledge source. Figure 5.1 summarizes the modular structure and components of the new *ATO-MS* system.

5.2 Modules

5.2.1 Session

The *Session* object centralizes every information about a current orchestration problem. It can be seen as the mother object which contains current instances of every sub-part of the problem.

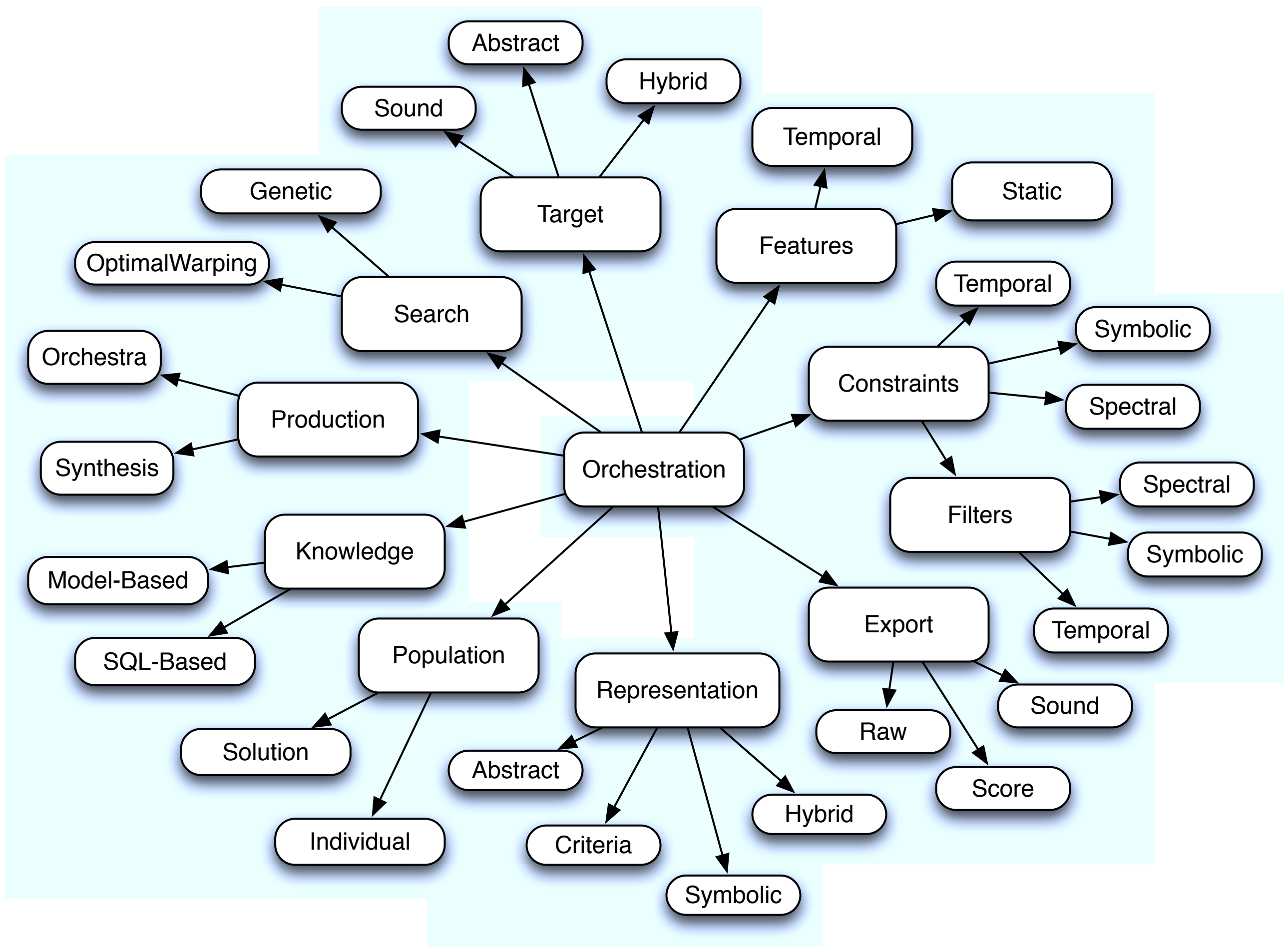


Figure 5.1: The current prototype for Abstract Temporal Orchestration with Modular Structure (ATO-MS) features an extensible architecture of modules to tackle the problem of Computer-Aided Orchestration.

5.2.2 Production

The *Production* object informs the system on the current means of sound generation that can be used. It is an informative class that contains definitions on the capacities and set of instruments that are allowed for a specific orchestration. The production therefore contains

- Domains of variables and allowed instruments (these are indices taken from the *Knowledge* object).
- Duration and onsets informations on different individuals
- Resolution to be used in generation (for example the orchestra can have a microtonic resolution).
- Set of filters to be applied on the production capabilities

This version of *Ato-ms* contains a complete implementation of the *ProductionOrchestra* object, which allows to use classical orchestra instruments in orchestration. However, an example blank class *ProductionSynthesis* (that could be linked to *KnowledgeModel*) shows that a possible extension would be to incorporate synthesis instruments in the search.

5.2.3 Knowledge

The *Knowledge* object is used by the system to retrieve the features and spectral attributes for individuals used in the search process. Therefore the knowledge must contain and implement the following interaction

- Description of the knowledge on attributes and domains
- Set of allowed features to be used as optimization criteria
- Values of features for the neutral element
- Compute domains for a specific attribute
- Building, updating, adding and removing knowledge
- Retrieving informations about a knowledge field for any element such as symbolic values or spectral descriptors
- Perform high-level queries over the knowledge source

The current implementation is based on a SQL database architecture and therefore the *KnowledgeSQL* contains the complete code to allow such interaction. However, an example blank class *KnowledgeModel* shows that a possible extension to the system would be to incorporate model-based knowledge. This means that instead of refering to sound samples example, the values of spectral descriptors could be derived directly from computational models.

5.2.4 Search

The *Search* modules is the core of the orchestration system. These objects represent heuristics and algorithms that are able to provide fast and orderly proposals to a given orchestration problem. It therefore contains

- Set of current proposed solutions
- Parameters that can be tuned by the user

- Heuristics to combine sound features and evaluate how well they approximate the target

There are currently two search algorithms provided with the *Ato-ms* system. The *SearchGenetic* provides a multi-objective genetic algorithm which has been enhanced by using the hypervolume domination criterion instead of simple weighting. The *SearchOptimalWarping* is largely based on the time series indexing research. It is lightly slower than the genetic algorithm but should provide more complex and structurally interesting proposals.

5.2.5 Target

The *Target* is a core element of the orchestration system as it defines the objectives to attain and therefore which values to approximate for each feature. The target object therefore contains

- Set of spectral features (mean or temporal values) to approximate
- Eventually a set of symbolic parameters
- Parameters used for the spectral analysis
- Function to compute the set if the target is a sound file

There are currently two main types of target. The *TargetSound* is simply a container for a sound file target and will contain its features analysis. The *TargetAbstract* object allows to define a set of features to approximate without the need to provide a sound file. Therefore the user can directly input its shapes and values to be approximated. It is interesting to note that a *TargetAbstract* can always be derived from a *TargetSound* object.

5.2.6 Features

The *Features* objects allows to define for each spectral feature its computation method, its addition function and how to compute the distance between the same feature of two different mixture. Therefore each object must contain

- Weights (unimplemented yet) represent the relative importance of each feature.
- A list of dependancies if the feature requires computation or retrieval of other features to work correctly.
- An analysis function to compute the value for this feature given a signal or a sound file.
- An addition function which allows to predict the value of this feature for a mixture of sounds whose features are already known.
- A comparison function to compute the distance between the features of two different mixtures.
- An optional transposition function to predict the feature values given a particular resolution
- A neutral element function which returns the neutral value for this feature.

Currently there are several features that are already available for the user. It should be noted that two kind of generic features classes *FeaturesGenericTemporal* and *FeaturesGenericStatic* allows to take into account any undefined features if they are either temporal shapes or static values.

5.2.7 Population

The *Population* defines the set of proposals computed by an algorithm to a particular orchestration problem. Therefore populations are composed of a set of *Solution*s each of which is a mixture of several *Individuals*. Populations must therefore contain

- The set of criteria (distances) for each solution
- The set of features for each solution
- Operators to combine and modify populations.

Chapter 6

OSC Protocol

6.1 Server state

→ /isready

This message is sent by the client in order to probe the disponibility of the server. If the latter is free, it automatically answers with a `/ready` message.

```
/isready
```

Data
∅

← /ready

The server indicates that it is ready to receive messages. This message may be send after receiving a `/isready` message or after the execution of a particuliar task.

```
/ready
```

Data
∅

← /busy

This message is sent by the server to notify that it is starting a task. It will thus be busy until it sends a `/ready` message. This message may optionnaly contain a float `progress` (in the interval $[0, 1]$) in order to be used in a waitbar, as well as a message indicating to the client the current work in progress.

```
/busy <progress> <message>
```

Data		
progress	Float	Task progress status in $[0, 1]$
Data		

Data		
message	String	Current work in progress

Examples :

```
/busy 0.5 'Querying temporal shapes'
/busy 0.2 'Adding directory /sol/db/Bassoons/ to database'
```

← /acknowledge

This message is sent by the server to acknowledge the good reception of an incoming message for which the `data` field contains an identification `<id>`. Such an input message corresponds to a task pending to be performed. The `/acknowledge` message is automatically followed by a `/busy` message indicating the start of execution for the task.

/acknowledge <id>

Data		
id	Integer	ID of the message to acknowledge

→ /version

This message allows the client to ask for the current version of the OSC database server.

/version

Data
∅

← /version

This message is sent by the server after receiving a `/version` message and returns a formated message containing the current running version.

/version <message>

Data		
message	String	Current version message

→ /quit

This message sent by the client ends the OSC communication with the server, before the latter stops automatically. It will therefore be necessary to launch the server once again in order to open a new communication.

/quit

Data
∅

← /quit

This message is the last one sent by the server before shuting down.

/quit

Data
∅

6.2 Session handling

→ /newsession

This message is sent by the client and allows to start a new session with the server. It connects to a database which is placed either on a local or distant host. The database name, username and password for the session should be provided as well as the adress and port to use for connecting to the distant host. The client must also specify its IP address with parameter `ipOut`. This allows the server to send replies. This message should be sent first as it opens the connection and allows to retrieve a handler that will be used by all subsequent functions.

/newsession <host> <user> <pass> <dbName> <ipOut>

Data		
host	String	Adress of host for the database
user	String	Username for the connection
pass	String	Password for the connection
dbName	String	Name of the database
ipOut	String	IP adress of the client for reply

Examples :

```
/newsession 'localhost' 'root' 'root' 'iSDB' '129.102.12.25'
/newsession '65.76.1.23' 'esling' 'y,kH' 'iSDB' '129.102.6.92'
```

→ /closesession

This message is sent by the client in order to end the session with the server. It should always be sent when the client quits as it allows to properly close the connection with the database. It should be noted that after sending this message it is not anymore possible for the client to interact with the server. If the client wishes to do so, it must send a `/newsession` message.

/closession

Data
∅

6.3 Database

6.3.1 Serialization

→ /databasefileexport

This message allows to export the database to a file. Using a dumping procedure, the whole database is exported to a single serialized file which contains table structures as well as all elements that were present in the original database. Parameters for host adress, username, password and name of the database are implicitely set from the session parameters.

/databasefileexport <filename>

Data		
filename	String	Name of the exported file

Example :

```
/databasefileexport '/tmp/database/ircamSpectralDB.db'
```

→ /databasefileimport

This message allows to import the database from a serialized file generated by the `/databasefile-export` message. It should be noted that all existing content in the host database will be overwritten by the new database content encapsulated in the serialized file (elements and table structures). Parameters for host, user, pass, dbName are implicitely set from the session parameters.

/databasefileimport <filename>

Data		
filename	String	Name of the file to import

Example :

```
/databasefileimport '/tmp/database/ircamSpectralDB.db'
```

→ /getdescriptorvalue

This message allows to retrieve the value of a specific descriptor for a single sound specified by its ID.

/getdescriptorvalue <soundID> <descriptor>

Data		
soundID	Integer	ID of sound to query
descriptor	String	Name of descriptor to query

Example :

```
/getdescriptorvalue 573 'SpectralCentroid'
```

← /getdescriptorvalue

This message is sent by the server in response to a `/getdescriptorvalue` message. It contains the value of a single descriptor for a specific ID.

/getdescriptorvalue <descValue>

Data		
descValue	Variable	Value of the required descriptor

Example :

If the client asks for the value of spectral centroid for sound number 573 :

```
/getdescriptorvalue 573 'SpectralCentroid'
```

The server will reply :

```
/getdescriptorvalue '(276.9533,559.7832,...,492.68,490.24)'
```

6.3.2 Complex research

→ /getsoundsquery

This message is a complex query builder on the database that embeds all research paradigms. It is such possible to make queries based on symbolic and spectral criteria as well as temporal shape of descriptors. The queries parameter are passed as a list which contains organized structure elements. It is possible for the client to put a potentially infinite number of constraints on the query. Each element of the list must follow the structure :

- type = type of relation used

- descriptor = name of the descriptor

- connector = logical connector (and / or)

- value = values to use

In the case of temporal queries the array contains :

value{1} : array of values representing the shape

value{2} : weight of the descriptor

/getsoundsquery <maxN> (<desc> <type> <v1> <v2> <connect>)+

Data		
maxN	Integer	Max number of sounds desired
desc	String	Name of the descriptor
type	String	Relation to impose on the descriptor
v1	Variable	First value to use for query
v2	Variable	Second value to use for query
connect	String	Logical connector ("AND" / "OR")

Example :

Simple queries for 5 sounds depending on the family name. It should be noted that the second value and the logical connector must always be present in the parameters even if they are unused.

```
/getsoundsquery 5 'family' 'is' 'Bassoons' '' 'AND'
/getsoundsquery 5 'family' 'contains' 's' '' 'AND'
```

Simple queries for 5 sounds depending on a float value

```
/getsoundsquery 5 'duration' 'over' 0.5 '' 'AND'
/getsoundsquery 5 'duration' 'under' 0.3 '' 'AND'
/getsoundsquery 5 'duration' 'between' 0.12 0.59 'AND'
```

The last query can equivalently be written

```
/getsoundsquery 5 'duration' 'over' 0.12 '' 'AND' 'duration' 'under' 0.59 '' 'AND'
```

For temporal shapes querying, the first value correspond to an array of values representing the desired function and the second value represents the weight of this criteria in the final result

```
/getsoundsquery 5 'FundamentalFrequency' 'follows' '0.0 0.5 1.0' 1.0 'AND'
```

Thanks to the weight parameter it is possible to make multiple temporal queries with different user-defined preferences

```
/getsoundsquery 5 'FundamentalFrequency' 'follows' '0.0 0.5 1.0' 0.7 'AND' 'Loudness...
    ' 'follows' '1.0 0.5 0.0' 0.3 'AND'
```

Finally it is possible to make a mixed query with symbolic, spectral or temporal constraints :

```
/getsoundsquery 5 'family' 'contains' 's' ' ' 'OR' 'note' 'is' 'C5' '' 'AND' '...
    FundamentalFrequency' 'follows' '0.0 0.5 1.0' 0.7 'OR' 'Loudness' 'follows' '1.0 ...
    0.0' 0.3 'AND'
```

← /getsoundsquery

This message is sent by the server in response to a /getsoundsquery message. It contains a simple list of ID matching the query in the original message. The list is in ascending order if only static criterias were used in the query. However, if the query contained temporal constraints, the resulting list will be ordered by the weighted distance measure.

/getsoundsquery (<ID>)+

Data		
ID	Integer	Value of a matching sound ID

Example :

If the client asks for 5 sounds with a query containing only static criterias

```
/getsoundsquery 5 'family' 'is' 'Bassoons' '' 'AND'
```

The resulting response will be ordered in an ascending manner

```
/getsoundsquery 133 134 142 151 168
```

However if the original query contains temporal constraints

```
/getsoundsquery 5 'FundamentalFrequency' 'follows' '0.0 0.5 1.0' 1.0 'AND'
```

The server will respond with a list of ID ordered by the weighted distance measure

```
/getsoundsquery 85 292 186 289 81
```

6.4 Target

6.5 Orchestra

6.6 Filters

6.7 Algorithms

→ /getsearchalgorithms

This message is sent by a client in order to retrieve the name of the algorithms which are currently implemented in the server and that the user can use.

/getsearchalgorithms

Data

Example :

```
/getsearchalgorithms
```

← /getsearchalgorithms

This message is sent by the server to the client to specify the name of the algorithms that can be used for orchestration searches.

/getsearchalgorithms

Data		
id	Integer	ID of the incoming message
error	String	Text of encountered error

Example :

```
/getsearchalgorithms
```

→ /getsearchparameters

This message is sent by a client in order to retrieve the parameters of a specific search algorithm that can be tuned by the user

/getsearchparameters

Data

Example :

```
/getsearchparameters
```

← /getsearchparameters

This message is sent by the server to the client to specify the name and the default values of parameters for a specific search algorithm.

/getsearchparameters

Data

Example :

```
/getsearchparameters
```

→ /setsearchparameters

This message is sent by a client in order to set one of the parameters of the currently selected algorithm to a specific value

/setsearchparameters

Data

Example :

```
/setsearchparameters
```

6.8 Saving and loading

6.9 Settings

→ /setoutport

This message is sent by the client to the server in order to set which port will be used by the server for outgouing communication.

/setoutport <id> <port>

Data		
id	Integer	Id of the incoming message
port	Integer	Port for outgoing communication

6.10 Report

→ /sendreport

This message is sent by the client to the server in order to use the report system embedded in it. That way, the client can specify the values to fill in the report.

/sendreport

Data

6.11 Errors

← /error

This message is sent by the server to the client whenever an error occurs after the execution of a task or calculus following an incoming message with identification index `<id>`.

/error <id> <error>

Data		
id	Integer	ID of the incoming message
error	String	Text of encountered error

Message	Action
/isready	Ask if the server is ready
/version	Get the current version of server
/quit	Quit the OSC server
/newsession	Start a new session with the database
/closesession	Ends a previously started session
/addsound	Add a new sound to the database
/addsounddirectory	Add a directory to the database
/addsoundxml	Add a sound from an XML description
/addsoundxmldirectory	Add an XML directory
/removefamily	Remove a complete family from the database
/removesound	Remove a specific sound from the database
/setdescriptorvalue	Change the value of descriptor for a sound
/databasefileexport	Serialize the whole database to a single file
/databasefileimport	Import a database from a serialized file
/getdescriptorlist	Get all descriptors names and types available
/getdescriptorrelation	Get query relations available for a descriptor
/getfamilylist	Retrieve all families available
/getvalueslist	Retrive all values possible for a descriptor
/getalldescriptorsvalues	Get all descriptors values for a sound
/getalldescriptorsvaluesfast	Get all atomic descriptors values for a sound
/getdescriptormultipleid	Get multiple descriptors for multiple sounds
/getdescriptormultiplevalues	Get multiple descriptors for a single sound
/getdescriptorvalue	Get a descriptor value for a sound
/getsoundsquery	Complex queries builder over the database
/gettemporalquery	Temporal shape query on a single descriptor
/gettemporalquerydtw	Temporal query (Dynamic Time Warping)

Table 6.26: List of OSC messages available for the client

Message	Action
/ready	Server is ready to receive queries
/version	Current version message of the server
/quit	Server is closing
/busy	Server is currently busy
/acknowledge	Acknowledging the receive of a message
/getdescriptorlist	Get all descriptors names and types available
/getdescriptorrelation	Get query relations available for a descriptor
/getfamilylist	Retrieve all families available
/getvalueslist	Retrive all values possible for a descriptor
/getalldescriptorsvalues	Get all descriptors values for a sound
/getalldescriptorsvaluesfast	Get all atomic descriptors values for a sound
/getdescriptormultipleid	Get multiple descriptors for multiple sounds
/getdescriptormultiplevalues	Get multiple descriptors for a single sound
/getdescriptorvalue	Get a descriptor value for a sound
/getsoundsquery	Complex queries builder over the database
/gettemporalquery	Temporal shape query on a single descriptor
/gettemporalquerydtw	Temporal query (Dynamic Time Warping)

Table 6.28: List of possible OSC replies from the server

Chapter 7

Database

We present in this section an overview of the current content of the database. We first summarize the different taxonomy used for the instruments. Table 7.2 contains the abbreviations for all instrumental families currently present in the database. Table 7.4 contains the same for all available mutes. Finally, Table 7.6 contains the abbreviations for all playing styles in the database. We subsequently present in Table 7.8 the list of all currently available symbolic descriptors, followed by Table 7.10 which presents the list of all spectral descriptors obtained from IRCAMDescriptor and their respective temporal modelisations.

Name	
ASax	Alto-Sax
BFl	Bass-Flute
BTb	Bass-Tuba
BTbn	Bass-Trombone
BClBb	Bass-Clarinet-Bb
Bn	Bassoon
ClBb	Clarinet-Bb
ClEb	Clarinet-Eb
Cb	Contrabass
CbClBb	Contrabass-Clarinet-Bb
CbFl	Contrabass-Flute
CbTb	Contrabass-Tuba
EH	English-Horn
Fl	Flute
Hn	Horn
Ob	Oboe
Picc	Piccolo
TpC	Trumpet-C
TTbn	Tenor-Trombone
Va	Viola
Vc	Violoncello
Vn	Violin

Table 7.2: List of instrumental families currently available in the database

Name		
C	Cup	TpC, TTbn
H	Harmon	TpC, TTbn
P	Plunger	TTbn
So	Sordina	BTb, Hn, Va, Vn, Vc, Cb
St	Straight	TpC, TTbn
Sp	Sordina piombo	Va, Vn, Vc
W	Wah	TpC, TTbn
Whsp	Whisper	TpC

Table 7.4: List of mutes currently available in the database

Name	
aeol	aeolian
aeol+ord	aeolian-and-ordinario
aeol-flatt	aeolian-flatterzunge
art-harm	artificial-harmonic
art-harm-trem	artificial-harmonic-tremolo
brassy	brassy
flatt	flatterzunge
flatt-closed	flatterzunge-closed
flatt-hi-reg	flatterzunge-high-register
flatt-open	flatterzunge-open
flatt-stopped	flatterzunge-stopped
fp	fortepiano
harm-fngr	harmonic-fingering
key-cl	key-click
legno-batt	col-legno-battuto
legno-tratto	col-legno-tratto
legno-tratto-pont	col-legno-tratto-ponticello
legno-tratto-tasto	col-legno-tratto-tasto
nonvib	non-vibrato
ord	ordinario
ord-closed	ordinario-closed
ord-hi-reg	ordinario-high-register
ord-open	ordinario-open
pdl-tone	pedal-tone
pizz	pizzicato
pizz-bartok	pizzicato-bartok
pizz-lv	pizzicato-l-vib
pizz-sec	pizzicato-secco
pont	sul-ponticello
pont-trem	sul-ponticello-tremolo
sfz	sforzando
stacc	staccato
stopped	stopped
tasto	sul-tasto
tasto-trem	sul-tasto-tremolo
tng-ram	tongue-ram
vib	vibrato

Table 7.6: List of playing modes currently available in the database

Name	Example
name	Vn-trill-min2-A4-mf-3c
file	/db/Strings/Violin/trills/Vn-trill-min2-A4-mf-3c.aif
source	ViennaSymphonic
instrument	Vn
family	Strings
playingStyle	trill-min2
dynamics	mf
note	A4
pitchClass	A
octave	4
stringMute	No
brassMute	NA
duration	7.29868

Table 7.8: List of symbolic descriptors currently available in the database

Name		
PartialsFrequency	NoiseSpectralCentroidBands	RelativeLoudnessBands
PartialsAmplitude	NoiseSpectralCentroid	RelativeLoudness
TotalEnergy	NoiseSpectralSpreadBands	Sharpness
HarmonicEnergy	NoiseSpectralSpread	Spread
NoiseEnergy	NoiseSpectralSkewnessBands	SpectralFlatnessBands
EnergyEnvelope	NoiseSpectralSkewness	SpectralFlatness
BandFluctuationStrengthBands	NoiseSpectralKurtosisBands	SpectralCrestBands
BandFluctuationStrength	NoiseSpectralKurtosis	SpectralCrest
BandRoughnessBands	NoiseSpectralSlopeBands	MFCCBands
BandRoughness	NoiseSpectralSlope	MFCC
FluctuationStrength	NoiseSpectralDecrease	DMFCCBands
Roughness	NoiseSpectralRollOff	DMFCC
FundamentalFrequency	NoiseSpectralVariationBands	DDMFCCBands
Noisiness	NoiseSpectralVariation	DDMFCC
Inharmonicity	PerceptualSpectralDeviationBands	SpectralCentroidBands
HarmonicSpectralDeviationBands	PerceptualSpectralDeviation	SpectralCentroid
HarmonicSpectralDeviation	PerceptualOddToEvenRatioBands	SpectralSpreadBands
HarmonicOddToEvenRatioBands	PerceptualOddToEvenRatio	SpectralSpread
HarmonicOddToEvenRatio	PerceptualTristimulusBands	SpectralSkewnessBands
HarmonicTristimulusBands	PerceptualTristimulus	SpectralSkewness
HarmonicTristimulus	PerceptualSpectralCentroidBands	SpectralKurtosisBands
HarmonicSpectralCentroidBands	PerceptualSpectralCentroid	SpectralKurtosis
HarmonicSpectralCentroid	PerceptualSpectralSpreadBands	SpectralSlopeBands
HarmonicSpectralSpreadBands	PerceptualSpectralSpread	SpectralSlope
HarmonicSpectralSpread	PerceptualSpectralSkewnessBands	SpectralDecrease
HarmonicSpectralSkewnessBands	PerceptualSpectralSkewness	SpectralRollOff
HarmonicSpectralSkewness	PerceptualSpectralKurtosisBands	SpectralVariationBands
HarmonicSpectralKurtosisBands	PerceptualSpectralKurtosis	SpectralVariation
HarmonicSpectralKurtosis	PerceptualSpectralSlopeBands	SignalAutoCorrelationBands
HarmonicSpectralSlopeBands	PerceptualSpectralSlope	SignalAutoCorrelation
HarmonicSpectralSlope	PerceptualSpectralDecrease	SignalZeroCrossingRate
HarmonicSpectralDecrease	PerceptualSpectralRollOff	TimeFrame
HarmonicSpectralRollOff	PerceptualSpectralVariationBands	soundID
HarmonicSpectralVariationBands	Loudness	

Table 7.10: List of spectral descriptors currently available in the database

List of Figures

List of Tables