Orchidea

intelligent assisted orchestration

Jinwoong Kim
Tokyo University of the Arts

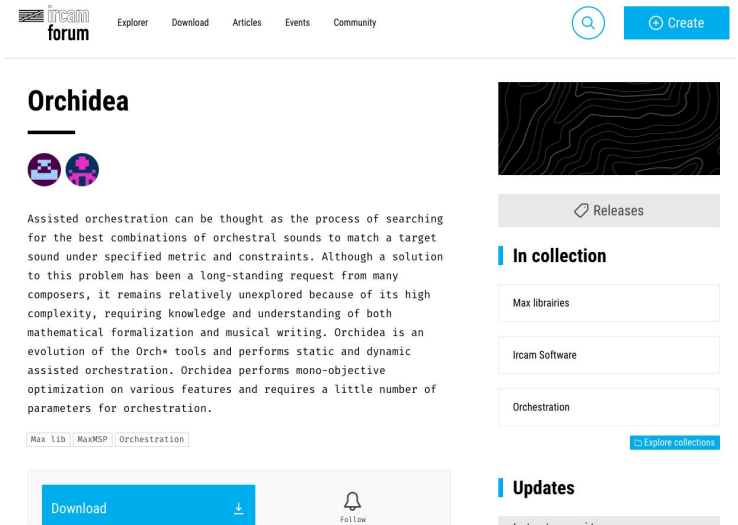
Content

- What is Orchidea?
- What is assisted orchestration?
- Static orchestration
- Dynamic orchestration

What is Orchidea?

Orchidea

- intelligent assisted orchestration tool developed in IRCAM. Works in **Max/MSP.**
- Orchidea performs mono-objective optimization on various features and requires a little number of parameters for orchestration.
- website: https://forum.ircam.fr/projects/detail/orchidea/



A bit of History

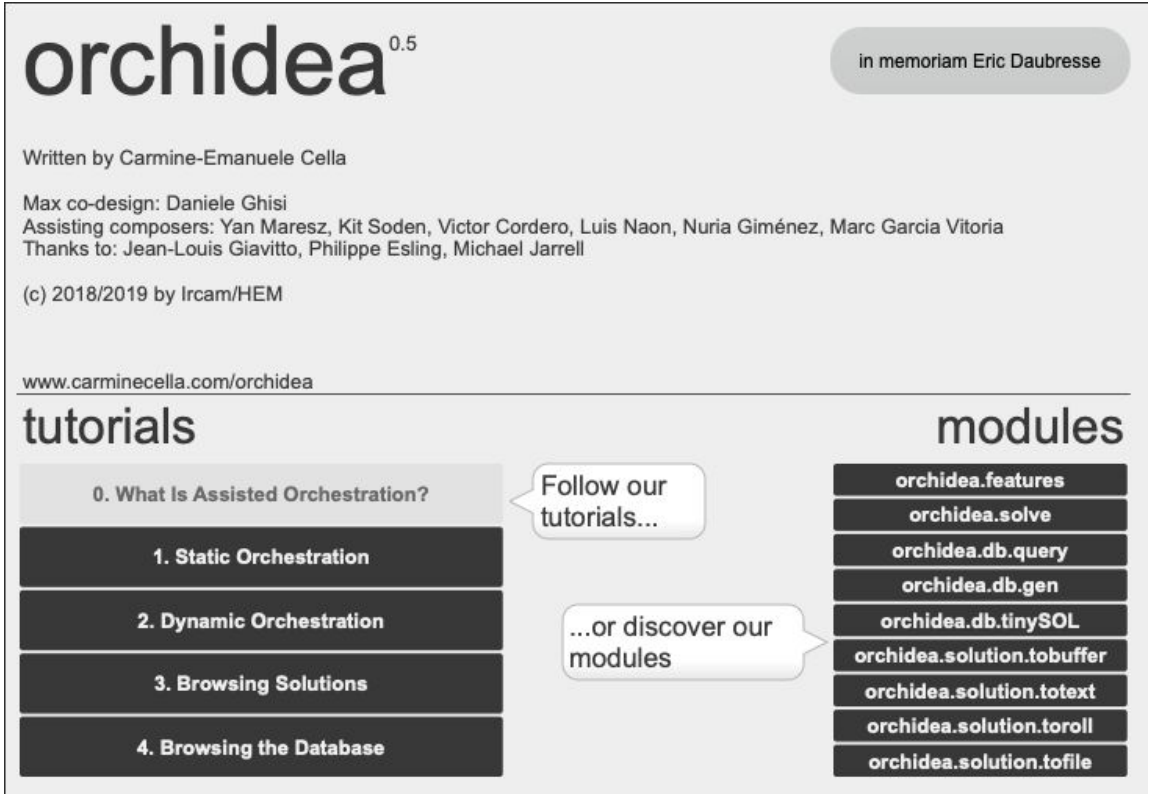
- English composer **Jonathan Harvey**, "Mortuos plango, vivos voco" (1981)
  - The first ante-litteram version of this problem has been investigated by Jonathan Harvey
  - In his work "Mortuos plango, vivos voco" (1981), where he tried to simulate the sound of a bell by a specific model of sound synthesis.
  - It is not by chance that this problem found fertile terrain at Ircam, in Paris, where the influences of the **spectralism music** were very strong around that period.

- A systematic approach developed in 2003, when the French composer **Yan Maresz** proposed a more formalized statement of the problem

- Since then, the problem has been researched at Ircam for more than 15 years and several outcomes have been produced: different tools (Matlab frameworks, Max/MSP interfaces, C++ standalone command line tools, etc.), several PhD thesis including G. Carpentier, D. Tardieu and P. Esling and several journal papers.

Orchidea Tutorial in Max/MSP

Contents of today's presentation is based on orchidea official tutorial



What is assisted orchestration?

Introduction

- The relation between **signals** and **symbols** is a central problem for acoustic signal processing.

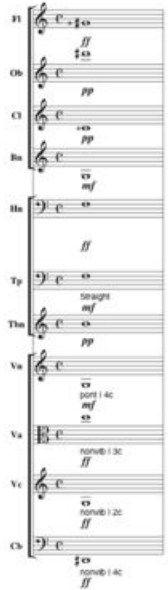
- Which connections can we make between the **symbolic space** and the **signal space**?
  - Symbolic Space(score) <---> Signal Space(timbre)

- **Assisted orchestration** finds its root in this context and can be considered as a possible answer to the previous question.

- **Target-based assisted orchestration** can be thought of as the process of
  - searching for the best combinations of orchestral sounds to match a target sound
  - under specified metric and constraints.

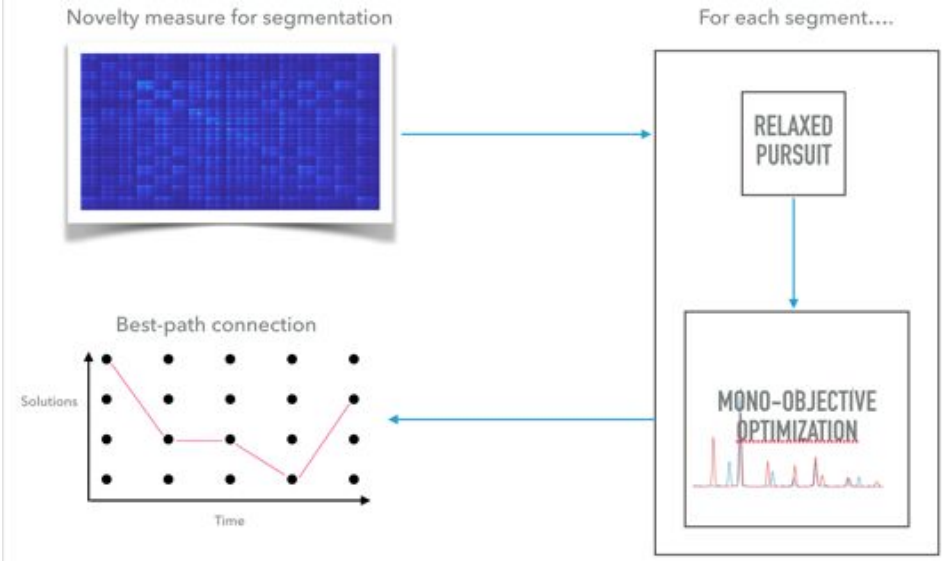
Problem statement

- By playing a score, you can convert (project) a representation existing in the symbolic space into a representation in the signal space. Is it possible to perform the **inverse operation**?



Implementation

- The target-based assisted orchestration problem can be split into two joint problems:
  - A **combinatorial optimization problem** defined on a multidimensional timbre description.
  - A **constraint solving problem** on the variables of musical writing.



Static orchestration

Calculation, not musical decision

- This toolbox is meant to give you some help in orchestrating: this help is more about making **calculations** at your place than taking musical decisions. These **calculations**, of course, are driven by some pre-compositional choices that either you or the designer of the tool that you are using must take.

Mimesis and katharsis

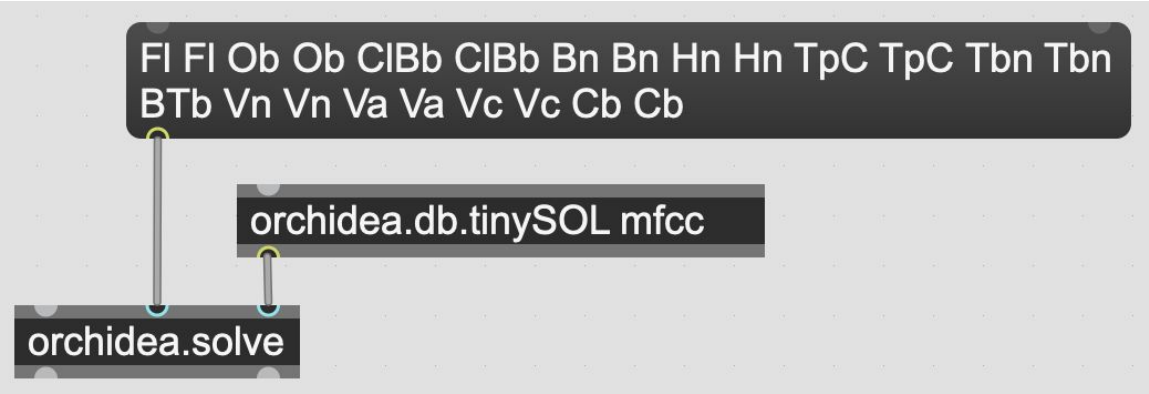
- It is hard to determine which assisted orchestration approach is preferable or not. Orchidea follows the path which mimics the input sound character.

- However, we have several parameters that control the mimesis and catharsis.
  - Feature selection: Pitch vs. Timbre
  - Pitch -> **Fourier spectrum** as the feature to be optimized
  - Timbre -> **MFCC** (Mel frequency cepstral coefficients) as the feature to be optimized

- Orchidea uses **database** as its features for problem solving.

Orchestra Setting

- The first important parameter to choose is the orchestra.
- A **message** specifies the instruments of the orchestra to be used; the names depend on the **database** used.
- Even you can set a specific subset of **articulations** and **dynamics**

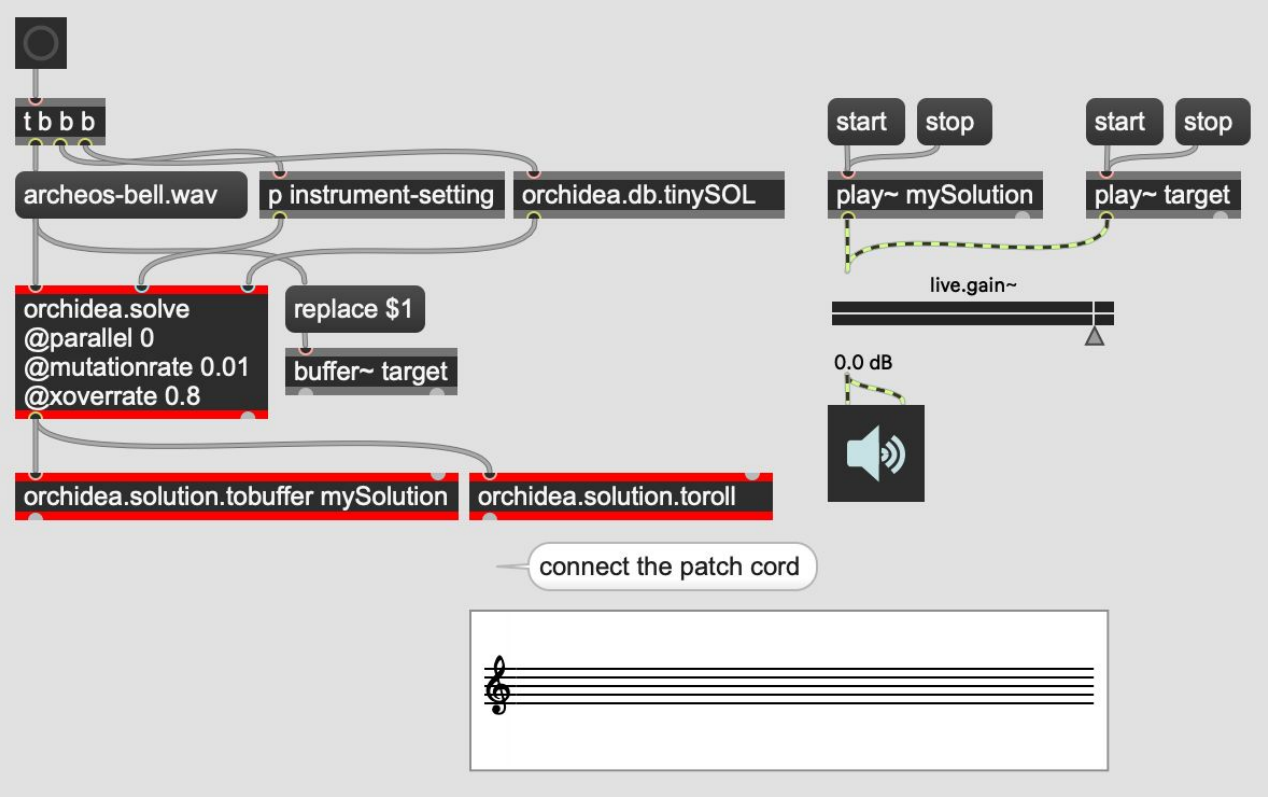


Optimization Setting (technical)

- Computation mode selection for using separate thread or not.
- The **population size** and **the number of epochs** determines the width of the search.
- **Initial population** for the optimization: random, relaxed pursuit, etc.
- **Crossover** and **mutation rate** for genetic optimization.
- The parameters for the optimization are really important, but normally you don't need to change them unless the solutions are not satisfactory.

```
orchidea.solve @parallel 0 @onsetthreshold 0.1
@onsettimegate 100 @partialswindow 32768
@partialsfiltering 0.1 @popsize 200 @maxepochs 200
@exportsolutions 0 @negativepenalization 10
@positivepenalization 0.5
```

1st Example


t b b b
archeos-bell.wav
p instrument-setting
orchidea.db.tinySOL
orchidea.solve
@parallel 0
@mutationrate 0.01
@xoverrate 0.8
replace $1
buffer~ target
orchidea.solution.tobuffer mySolution
orchidea.solution.toroll
start
stop
play~ mySolution
start
stop
play~ target
live.gain~
0.0 dB
connect the patch cord

More Parameters

- **Sparsity** determines the freedom that the algorithm has to remove instruments of the given orchestra from the solution. If sparsity is 0, each solution will use all instruments; if this value increases, the probability of dropping an instrument increases correspondingly.

- **Partials filtering**: if it is 0 there is no filtering and all the sounds in the database are used. If it is between 0 < x < 1 than the database is reduced by removing the sounds that does not correspond to the partials in the target. Increasing this value produces a stricter selection and only strongest partials (dynamically) are retained.

2st Example

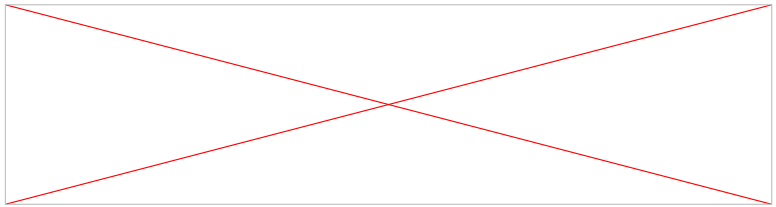

0.
0.01
0.1
0.
0.2
0.4
sparsity $1
partialsfiltering $1
t b b b
archeos-bell.wav
p instrument-setting
orchidea.db.tinySOL
orchidea.solve
@parallel 0
@mutationrate 0.01
@xoverrate 0.8
replace $1
buffer~ target
orchidea.solution.tobuffer mySolution
orchidea.solution.toroll
connect the patch cord
start
stop
play~ mySolution
start
stop
play~ target
live.gain~
0.0 dB

Dynamic orchestration

More Parameters

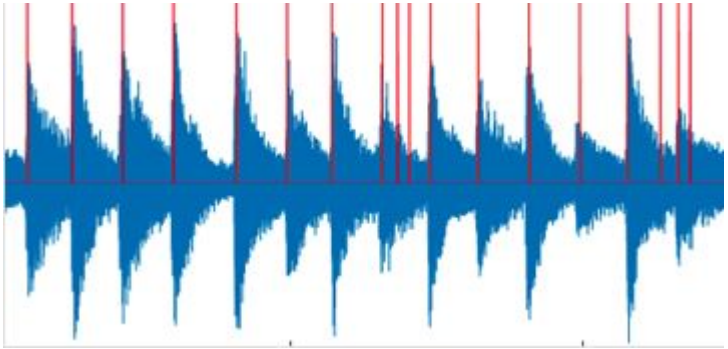
- The basic principle is to **cut** a dynamic target into static segments and **reconnect** them after the orchestration.

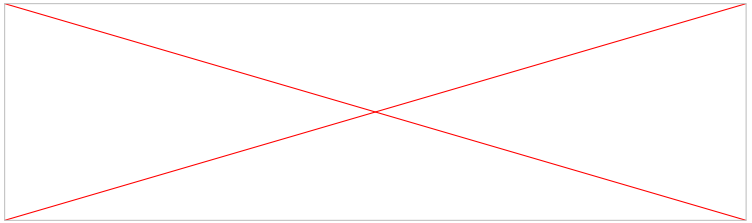
- The current segmentation model uses **thresholds** and **timegates** on specific spectral features to determine onsets. Timegates are specified in seconds, thresholds are absolute values between 0 and 1.



Segmentation policies: frames vs. flux

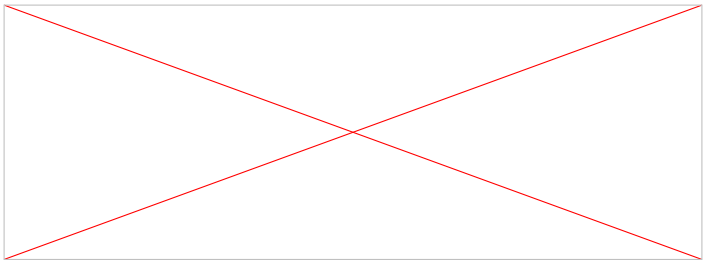
- There are two different policies to segment the target (attribute @segmentation): frames and flux.
- Fames will produce segments of equal length (called frames)
- Flux will use a specific 'novelty' measure to cut the target in significant chunks. In most of the cases you may want to use the second policy, the flux.



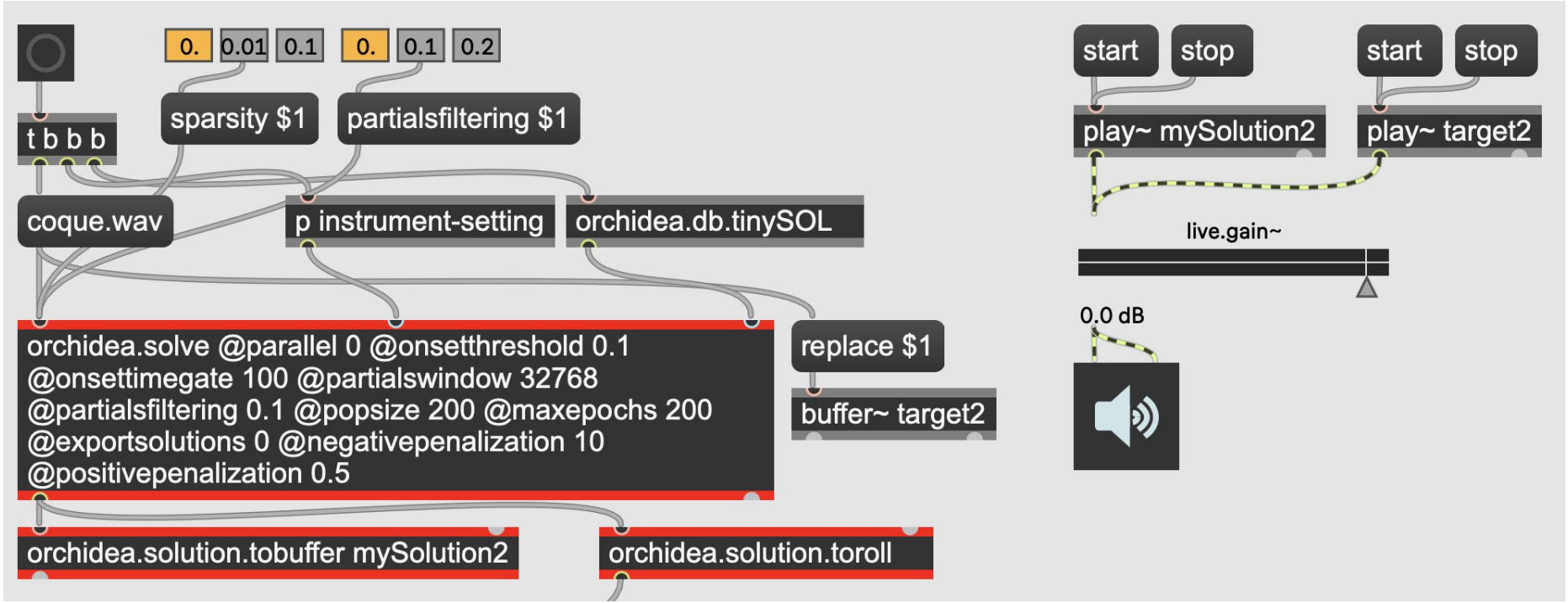


Connection policies: best vs. closest

- There are two policies to connect the solutions in time (attribute @connection): best and closest.
- “best” will connect the best solution of each segment (the solution that best matches the target).
- “closest”, instead, will connect the solutions that minimize movements for each instrument (for example by keeping the same instrument and the same note across segments).



3rd Example


0.
0.01
0.1
0.
0.1
0.2
sparsity $1
partialsfiltering $1
t b b b
coque.wav
p instrument-setting
orchidea.db.tinySOL
orchidea.solve @parallel 0 @onsetthreshold 0.1
@onsettimegate 100 @partialswindow 32768
@partialsfiltering 0.1 @popsize 200 @maxepochs 200
@exportsolutions 0 @negativepenalization 10
@positivepenalization 0.5
replace $1
buffer~ target2
orchidea.solution.tobuffer mySolution2
orchidea.solution.toroll
start
stop
play~ mySolution2
start
stop
play~ target2
live.gain~
0.0 dB

Thank you