FuX, an Android app that generates counterpoint

D. Herremans, K. Sørensen

IEEE SSCI, April 16-19th, 2013, Singapore
CI for Creativity and Affective Computing
Overview

Computer Aided Composing

Quantifying Counterpoint Quality

VNS

Android Implementation

Conclusion
Computer aided composing (CAC)

Composing music = combinatorial optimization problem

- Music → combination of notes
- “Good” music → fits a style as well as possible
- Formalized and quantified “rules” of a style → objective function
Counterpoint

- Polyphonic baroque music
- Inspired Bach, Haydn, ...
- One of the most formally defined musical styles
  → Rules written by Fux in 1725
5th species counterpoint

- Counterpoint & Cantus firmus

Represented as a vector of note objects, each with:
- Pitch: midi value
- Duration
- Beat number
- Measure number
- Tied?
Examples of rules:

- Each large leap should be followed by stepwise motion in the opposite direction
- Half notes should always be consonant on the first beat, unless they are suspended and continued stepwise and downward
- All perfect intervals should be approached by contrary or oblique motion

→ 19 vertical and 19 horizontal subscores between 0 and 1
Quantifying musical quality

- Eight notes (8ths) must move in step.

\[
subscre^H_1(s) = \frac{\#\text{8ths not preceded by step} + \#\text{8ths not left by step}}{\#\text{8ths} \times 2}
\]  

(1)

- Whole notes should always be vertically consonant.

\[
subscre^V_1(s) = \frac{\#\text{dissonant whole notes}}{\#\text{whole notes}}
\]  

(2)
Quantifying musical quality

\[
f_{cf}(s) = \sum_{i=0}^{19} a_i \cdot \text{subscore}_{cf_i}^H(s) \quad (3)
\]

\[
f_{cp}(s) = \sum_{i=0}^{19} a_i \cdot \text{subscore}_{cp_i}^H(s) + \sum_{j=0}^{19} b_j \cdot \text{subscore}_j^V(s) \quad (4)
\]
Variable Neighborhood Search

- Local search with 3 neighborhoods
- Selection: steepest descent

<table>
<thead>
<tr>
<th>$N_i$</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{sw}$</td>
<td>Swap</td>
<td>Swap two notes</td>
</tr>
<tr>
<td>$N_{c1}$</td>
<td>Change1</td>
<td>Change one note</td>
</tr>
<tr>
<td>$N_{c2}$</td>
<td>Change2</td>
<td>Change two notes</td>
</tr>
</tbody>
</table>
Variable Neighborhood Search

- Excluded fragments
  - Tabu list
  - Infeasible
- Perturbation
  - Change r% of the notes randomly
- Adaptive weights mechanism
  - Increase weight of subscore with highest value
  - Keeps the search in the right direction
### Experiments & Results

- Full factorial experiment, n=2304

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
<th>Nr. of levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{sw}$ - Swap</td>
<td>on with $tt_{sw}=0$, $tt_{sw}=rac{1}{16}$, $tt_{sw}=rac{1}{8}$, off</td>
<td>4</td>
</tr>
<tr>
<td>$N_{c1}$ - Change1</td>
<td>on with $tt_{c1}=0$, $tt_{c1}=rac{1}{16}$, $tt_{c1}=rac{1}{8}$, off</td>
<td>4</td>
</tr>
<tr>
<td>$N_{c2}$ - Change2</td>
<td>on with $tt_{c2}=0$, $tt_{c2}=rac{1}{16}$, $tt_{c2}=rac{1}{8}$, off</td>
<td>4</td>
</tr>
<tr>
<td>Random move</td>
<td>$\frac{1}{4}$ changed, $\frac{1}{8}$ changed, off</td>
<td>3</td>
</tr>
<tr>
<td>Adaptive weights</td>
<td>on, off</td>
<td>2</td>
</tr>
<tr>
<td>Max. iterations</td>
<td>5, 20, 50</td>
<td>3</td>
</tr>
<tr>
<td>Length of music</td>
<td>16, 32 measures</td>
<td>2</td>
</tr>
</tbody>
</table>
Experiments & Results

- Multi-Way ANOVA model with interaction effects, using R
- $R^2 = 0.98$

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Df</th>
<th>$F$ value</th>
<th>Prob ($&gt; F$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{c1}$</td>
<td>1</td>
<td>9886.2323</td>
<td>$&lt; 2.2e^{-16}$</td>
</tr>
<tr>
<td>$N_{c2}$</td>
<td>1</td>
<td>15690.7234</td>
<td>$&lt; 2.2e^{-16}$</td>
</tr>
<tr>
<td>$N_{sw}$</td>
<td>1</td>
<td>3909.2959</td>
<td>$&lt; 2.2e^{-16}$</td>
</tr>
<tr>
<td>randsize</td>
<td>2</td>
<td>1110.1724</td>
<td>$&lt; 2.2e^{-16}$</td>
</tr>
<tr>
<td>maxiters</td>
<td>2</td>
<td>322.6488</td>
<td>$&lt; 2.2e^{-16}$</td>
</tr>
<tr>
<td>length</td>
<td>1</td>
<td>165.6053</td>
<td>$&lt; 2.2e^{-16}$</td>
</tr>
<tr>
<td>adj. weights</td>
<td>1</td>
<td>4.0298</td>
<td>0.0448367</td>
</tr>
<tr>
<td>$tt_{c1}$</td>
<td>2</td>
<td>2.2575</td>
<td>0.1048791</td>
</tr>
<tr>
<td>$tt_{c2}$</td>
<td>2</td>
<td>8.271</td>
<td>0.0002646</td>
</tr>
<tr>
<td>$tt_{sw}$</td>
<td>2</td>
<td>3.2447</td>
<td>0.0391833</td>
</tr>
</tbody>
</table>
Experiments & Results

- Mean plot for the size of the random jump
**Optimal parameter settings**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{sw}$</td>
<td>on with $tt_{sw} = \frac{1}{16}$</td>
</tr>
<tr>
<td>$N_{c1}$</td>
<td>on with $tt_{c1} = \frac{1}{16}$</td>
</tr>
<tr>
<td>$N_{c2}$</td>
<td>on with $tt_{c2} = \frac{1}{16}$</td>
</tr>
<tr>
<td>Random move</td>
<td>$\frac{1}{8}$ changed</td>
</tr>
<tr>
<td>Adaptive weights</td>
<td>on</td>
</tr>
<tr>
<td>Max. number of iterations</td>
<td>50</td>
</tr>
</tbody>
</table>
C++ Implementation

Generated Music

Part 1

Part 2
Android App - FuX

- Software toolkit for mobile devices
- Dalvik Virtual Machine → Java
- VNS is computationally expensive
  → Native C++ (jni) with java NDK
- Midi files (Android-Midi-Lib & Mediaplayer)
  ⇒ Continuous stream of music (multitreading)
<table>
<thead>
<tr>
<th>Time</th>
<th>Generate</th>
<th>Playback</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>16 measures (file 1)</td>
<td></td>
</tr>
<tr>
<td>0s</td>
<td>8 measures (file 2)</td>
<td>file 1</td>
</tr>
<tr>
<td>16s</td>
<td>8 measures (file 3)</td>
<td>file 2</td>
</tr>
<tr>
<td>24s</td>
<td>8 measures (file 4)</td>
<td>file 3</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

→ Evaluation based on last 16 measures
Results over time

Evolution of the objective function over time

(a) Cantus firmus

(b) Counterpoint
You +1'd this

Description

Are you a composer looking for ideas? Or do you just enjoy listening to music? FuX lets you listen to an endless stream of continuously generated counterpoint music.

FuX uses a Variable Neighbourhood Search algorithm (VNS) to generate a continuous stream of fifth species counterpoint.

Strict counterpoint is a formally defined musical style that originates in the 16th century. The rules of this style were written down by Fux in his book "Gradus Ad Parnassum" in 1725. This app generates a continuous stream of music that adheres to these rules as well as possible.

The VNS algorithm used in FuX takes into account 19 melodic and 19 harmonic rules when generating counterpoint music.

Visit Developer’s Website  Email Developer

App Screenshots
Composer specific music - FuX 2.0

- Analyzed existing music (Bach, Beethoven, Haydn)
- 3 composer classification models
- Incorporated in objective function
Conclusion

An efficient VNS has been developed and implemented as an Android app. The resulting app can play a continuous stream of counterpoint music on any Android phone or tablet.

Future research:

- More complex music:
  - Different styles
  - More parts
  - Theme
- Improved sound quality
- Bugfixes for multitude of devices
- Hit song prediction
FuX, an Android app that generates counterpoint

D. Herremans, K. Sörensen
IEEE SSCI, April 16-19th, 2013, Singapore
dorien.herremans@ua.ac.be