Composing Counterpoint Music With Variable Neighborhood Search

D. Herremans & K. Sørensen
ORBEL26, February 2-3, 2012
Overview

Computer aided composing (CAC)

Variable Neighborhood Search

Experiments & Results

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 classical music
- Inspired Bach, Haydn, ...
- One of the most formally defined musical styles
  → Rules written by Fux in 1725
1st species counterpoint

- Counterpoint & Cantus firmus

- Represented as 2 vectors with midi values

\[ [60 \ 65 \ 64 \ 62 \ 60 \ 64 \ 65 \ 67 \ 67 \ 69 \ 62 \ 64 \ 64 \ 60 \ 59 \ 60] \]
5th species counterpoint

- Counterpoint & Cantus firmus

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

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

\[ f(s) = \sum_i a_i \cdot \text{subscore}^H_i(s) + \sum_j b_j \cdot \text{subscore}^V_j(s) \]  

horizontal aspect  
vertical aspect
Quantifying musical quality

- Weights $a_i$ and $b_j$
- Specified at input
  - Emphasize subscore from start
- Adaptive weights mechanism
  - Increase weight of subscore with highest value
  - Keeps the search in the right direction
Variable Neighborhood Search

- Local search with 3 neighborhoods
- Selection
  - Steepest descent
  - Based on adaptive score \( f^a(s) \)

<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
- Update best solution $s_{\text{best}}$, based on original score $f(s_{\text{best}})$
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
- $R^2 = 0.9642$

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>$F$ value</th>
<th>Prob ($&gt; F'$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{c1}$</td>
<td>1</td>
<td>155.73</td>
<td>155.73</td>
<td>4857.6450</td>
<td>$&lt; 2.2e^{-16}$</td>
</tr>
<tr>
<td>$N_{c2}$</td>
<td>1</td>
<td>238.40</td>
<td>238.40</td>
<td>7436.5417</td>
<td>$&lt; 2.2e^{-16}$</td>
</tr>
<tr>
<td>$N_{sw}$</td>
<td>1</td>
<td>69.13</td>
<td>69.13</td>
<td>2156.2797</td>
<td>$&lt; 2.2e^{-16}$</td>
</tr>
<tr>
<td>randsize</td>
<td>2</td>
<td>38.09</td>
<td>19.05</td>
<td>594.1391</td>
<td>$&lt; 2.2e^{-16}$</td>
</tr>
<tr>
<td>maxiters</td>
<td>2</td>
<td>9.30</td>
<td>4.65</td>
<td>145.0207</td>
<td>$&lt; 2.2e^{-16}$</td>
</tr>
<tr>
<td>$tt_{c1}$</td>
<td>2</td>
<td>0.05</td>
<td>0.02</td>
<td>0.7588</td>
<td>0.468333</td>
</tr>
<tr>
<td>$tt_{c2}$</td>
<td>2</td>
<td>0.15</td>
<td>0.08</td>
<td>2.3595</td>
<td>0.094707</td>
</tr>
<tr>
<td>$tt_{sw}$</td>
<td>2</td>
<td>0.08</td>
<td>0.04</td>
<td>1.3150</td>
<td>0.268681</td>
</tr>
<tr>
<td>adj. weights</td>
<td>1</td>
<td>0.30</td>
<td>0.30</td>
<td>9.3497</td>
<td>0.002257</td>
</tr>
</tbody>
</table>
Experiments & Results

- Mean plot for the size of the random jump

![Graph showing the relationship between score and time against random size (in %)]
Optimal parameter settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_{sw}$ - Swap</td>
<td>on with $tt_{sw}=0$</td>
</tr>
<tr>
<td>$N_{c1}$ - Change1</td>
<td>on with $tt_{c1} = \frac{1}{4}$</td>
</tr>
<tr>
<td>$N_{c2}$ - Change2</td>
<td>on with $tt_{c2} = \frac{1}{2}$</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>
Implementation

- C++ → VNS
- JavaScript using the QtScript engine → MuseScore plugin
- Input:
  - Key (i.e., G♯ minor)
  - Weights for each subscores
  - VNS parameters
- Result: MusicXML
Implementation

Generated Music

Part 1

Part 2
Example of a generated fragment with score 0.556776.
Conclusion

The fifth species counterpoint rules have been quantified and an efficient algorithm has been implemented to compose this style of music.

Future research:

- More complex music:
  - Different styles
  - More parts
- Analyse DB of existing music and extract composer characteristics
- Compare the VNS to other algorithms, e.g. genetic algorithm
Composing Counterpoint Music With Variable Neighborhood Search

D. Herremans & K. Sörensen
ORBEL26, February 2-3, 2012