



University of Antwerp
Operations Research Group

ANT/OR

Composing Counterpoint Music With Variable Neighborhood Search

D. Herremans & K. Sörensen

EU/ME, May 10-11, 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



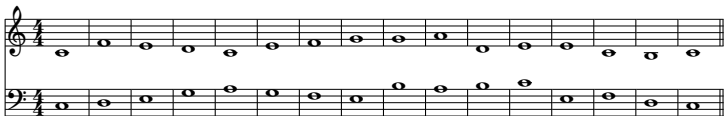
Counterpoint

- ▶ 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

$$f(s) = \underbrace{\sum_{i=0}^{19} a_i \cdot \text{subscore}_i^H(s)}_{\text{horizontal aspect}} + \underbrace{\sum_{j=0}^{19} b_j \cdot \text{subscore}_j^V(s)}_{\text{vertical aspect}} \quad (1)$$



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

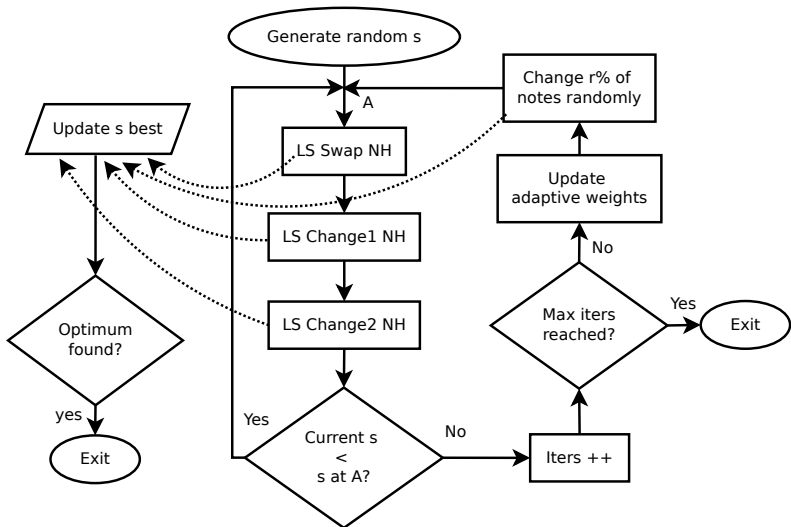
- ▶ Local search with 3 neighborhoods
- ▶ Selection
 - ▶ Steepest descent
 - ▶ Based on adaptive score $f^a(s)$

N_i	Name	Description
N_{sw}	Swap	Swap two notes
N_{c1}	Change1	Change one note
N_{c2}	Change2	Change two notes



Variable Neighborhood Search

- ▶ Excluded fragments
 - ▶ Tabu list
 - ▶ Infeasible
- ▶ Perturbation
 - ▶ Change $r\%$ of the notes randomly
- ▶ Adaptive weights mechanism
- ▶ Update best solution s_{best} , based on original score $f(s_{\text{best}})$





Experiments & Results

- Full factorial experiment, $n=2304$

Parameter	Values	Nr. of levels
N_{sw} - Swap	on with $tt_{sw}=0$, $tt_{sw}=\frac{1}{16}$, $tt_{sw}=\frac{1}{8}$, off	4
N_{c1} - Change1	on with $tt_{c1}=0$, $tt_{c1}=\frac{1}{16}$, $tt_{c1}=\frac{1}{8}$, off	4
N_{c2} - Change2	on with $tt_{c2}=0$, $tt_{c2}=\frac{1}{16}$, $tt_{c2}=\frac{1}{8}$, off	4
Random move	$\frac{1}{4}$ changed, $\frac{1}{8}$ changed, off	3
Adaptive weights	on, off	2
Max. iterations	5, 20, 50	3
Length of music	16, 32 measures	2



Experiments & Results

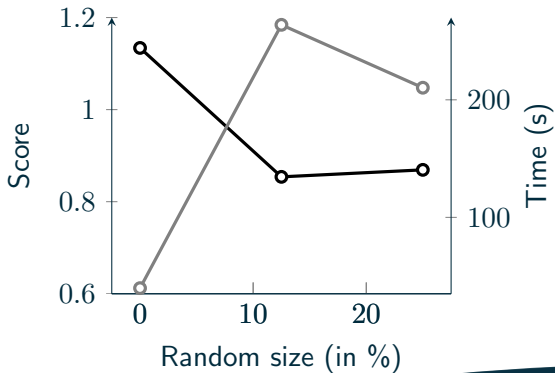
- ▶ Multi-Way ANOVA model with interaction effects, using R
- ▶ $R^2 = 0.9642$

Parameter	Df	F value	Prob ($> F$)
N_{c1}	1	4857.6450	$< 2.2e^{-16}$
N_{c2}	1	7436.5417	$< 2.2e^{-16}$
N_{sw}	1	2156.2797	$< 2.2e^{-16}$
randsize	2	594.1391	$< 2.2e^{-16}$
maxiters	2	145.0207	$< 2.2e^{-16}$
tt_{c1}	2	0.7588	0.468333
tt_{c2}	2	2.3595	0.094707
tt_{sw}	2	1.3150	0.268681
adj. weights	1	9.3497	0.002257



Experiments & Results

- Mean plot for the size of the random jump





Optimal parameter settings

Parameter	Value
N_{sw} - Swap	on with $tt_{sw}=0$
N_{c1} - Change1	on with $tt_{c1}=\frac{1}{4}$
N_{c2} - Change2	on with $tt_{c2}=\frac{1}{2}$
Random move	$\frac{1}{8}$ changed
Adaptive weights	on
Max. number of iterations	50



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

The screenshot displays the MuseScore software interface. The title bar reads "MuseScore: endmusic". The menu bar includes "File", "Edit", "Create", "Notes", "Layout", "Style", "Display", "Plugins", "Optimize", and "Help". The "Optimize" menu is open, showing three options: "Generate 5th Species Counterpoint", "Generate 1st Species Counterpoint", and "Generate Cantus Firmus".

On the left, a "Palettes" panel is visible, listing various musical elements such as Grace Notes, Drums, Clefs, Key Signatures, Time Signatures, Bar Lines, Lines, Arpeggio & Glissando, Breath & Pauses, Brackets, Articulations & Ornaments, Accidentals, Dynamics, Fingering, Note Heads, Tremolo, Repeats, Breaks & Spacer, Beam Properties, and Symbols.

The main workspace shows a score titled "endmusic" with a yellow background. A box labeled "Generated Music" is centered at the top of the workspace. Below it, the score is displayed in two systems. The first system is labeled "Part 1" and "Part 2". Part 1 is in treble clef and Part 2 is in bass clef. The second system is labeled "11" and also shows two staves. The word "Optimize" is visible in the bottom right corner of the workspace.



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
 - ▶ Theme
- ▶ Analyse DB of existing music and extract composer characteristics



University of Antwerp
Operations Research Group

ANT/OR

Composing Counterpoint Music With Variable Neighborhood Search

D. Herremans & K. Sörensen

EU/ME, May 10-11, 2012

dorien.herremans@ua.ac.be

