Inferring Tonality from Note Distributions: Why Models Matter

Fabian C. Moss*, **Martin Rohrmeier**

Digital and Cognitive Musicology Lab, École Polytechnique Fédérale de Lausanne

Background

Pitch-class statistics in pieces are assumed to correspond to mental representations of tonality [2].

References

[1] D. Harasim, F. C. Moss, M. Ramirez, and M. Rohrmeier. "Cognitive modeling reveals history of major and minor in Western classical music". Submitted.

[2] D. Huron. *Sweet Anticipation. Music and the Psychology of Expectation.* Cambridge, MA: MIT Press, 2006.

Acknowledgements

The research presented on this poster is generously supported by EPFL through the Latour Chair in Digital Musicology.

Improvement 1

Use models of tonal pitch space to reveal further regularities in pitch-class distributions [1]

Improvement 2

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Basic Algorithm

The algorithm for enumerating skipgrams takes as input:

- a list L of objects to generate skipgrams over
- a skip function f on pairs of objects
- the skipgram length n
- the skip limit k

The list L must be ordered with respect to the skip function:

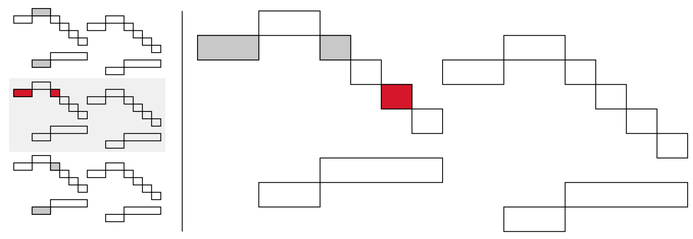
$$\forall i < j < k : f(L_i, L_j) \leq f(L_i, L_z).$$

The algorithm returns all sublists x of L with $|x| = n$ and $\sum_i f(x_i, x_{i+1}) \leq k$.

L is traversed, building up a **list of prefixes** that eventually become complete skipgrams.

For each element e in L:

1. remove old prefixes that cannot be extended with e
2. extend all remaining prefixes with e
3. output all completed prefixes (length n)
4. add all incomplete prefixes to the prefix list
5. add a new prefix $[e]$.



Extensions

Efficient **filtering** can be implemented by **test-**

Historical Development