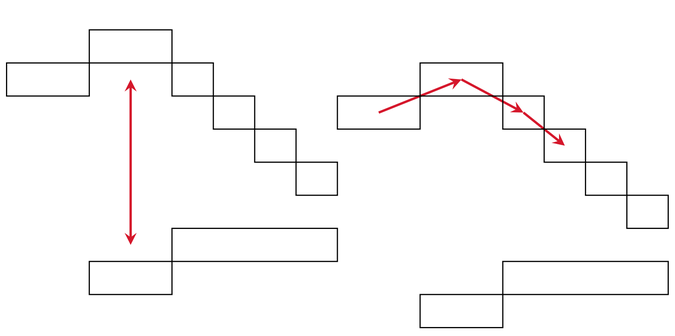
Inferring Tonality from Note Distributions: Why Models Matter

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

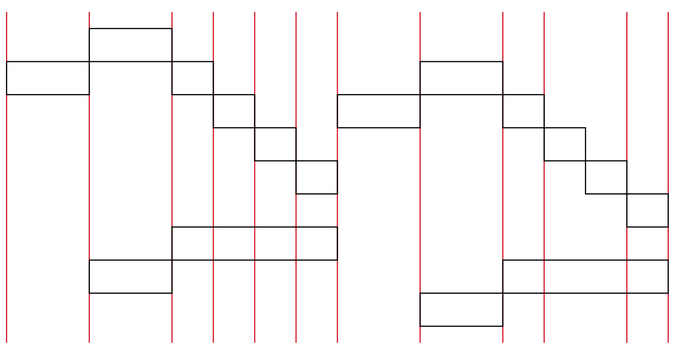
Digital and Cognitive Musicology Lab, École Polytechnique Fédérale de Lausanne

Background

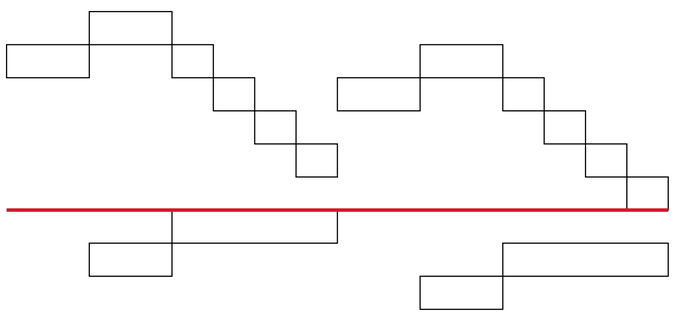
Many aspects of music are oriented **vertically** (harmony), **horizontally** (melody), or both (voice leading). However, the structure of music is neither cleanly vertical nor horizontal, but rather irregular in both dimensions.



Vertical structure is usually enforced by **slicing**. This can be problematic for cutting through notes and in cases where vertically related notes do not overlap (and thus do not have a common slice).



Horizontal structure is often ensured by working on **monophonic voices** or just a single **melody**. This is impractical in the general case, e.g., in piano music.



Improvement 1

We generalize the **skipgram technique**, well known in linguistics [1] and recently introduced to music research [2]. Skipgrams are similar to n-grams (fixed length subsequences of a sequence) but allow a limited amount of **gaps** between their elements.

If instead of counting gaps the "amount of skip" is defined by a **skip function**, the technique can be applied to non-sequential structures. The skip function is applied to consecutive elements in the skipgram. The summed skip must not exceed some parameter k.

Vertical structure can now be described by **skipgrams over notes** in a piece. An appropriate skip function could be the difference between the notes' onsets. This generates fixed-size groups of notes with **limited** (but possible) **non-simultaneity**.

Horizontal structure can be extracted much like vertical structure. Additionally requiring the notes not to overlap enforces **sequentiality**.

Horizontal and vertical structure can be combined by **recursive application of skipgrams**. A first pass generates vertical structure as skipgrams over notes ("stages"). A second step generates horizontal structure as skipgrams over stages, resulting in a nested, two-dimensional pattern.

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.

Improvement 3

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 ali-

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:

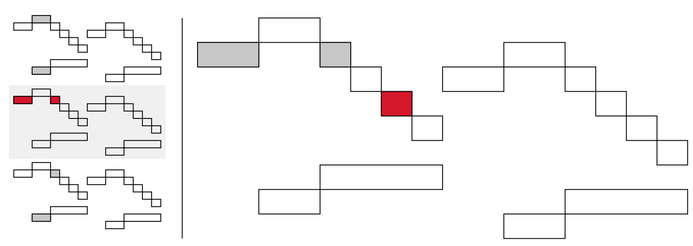
$$\forall i < j < k : f(L_i, L_j) \leq f(L_i, L_z).$$

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 **testing a predicate** on every extension of a prefix. If the new prefix does not satisfy the predicate, it is discarded.

Sampling can be implemented efficiently by **flipping a coin** on every prefix extension, deciding whether to keep or to discard the prefix. A prefix is extended $n - 1$ times, so keeping each prefix with probability $\sqrt[n-1]{p}$ means keeping the skipgram with probability p.

The output order depends on the last element of each skipgram, because the algorithm outputs skipgrams when they are completed. If the **order of the initial elements** should be retained, completed skipgrams are first entered in a **priority queue**. In each iteration, only those skipgrams are taken from the queue that cannot be preceded by currently active prefixes anymore.

Acknowledgements

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

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

References

[1] D. Guthrie, B. Allison, W. Liu, L. Guthrie, and Y. Wilks. "A Closer Look at Skip-Gram Modelling". In: *Proc. of the 5th International Conference on Language Resources and Evaluation (LREC-2006)*. European Language Ressources Association, 2006, pp. 1222–1225.

[2] D. R. W. Sears, A. Arzt, H. Frostel, R. Sonnleitner, and G. Widmer. "Modeling Harmony with Skip-Grams". In: *Proc. of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*. Ed. by S. J. Cunningham, Z. Duan, X. Hu, and D. Turnbull. 2017, pp. 332–338.