MidiComb

A CP-based Approach to Music Generation

Francesco Ballerini

Università di Bologna

Motivation

Hyun et al., NeurIPS 2022

Hyun et al. introduce ComMU: a dataset of short musical samples built for the task of combinatorial music generation.

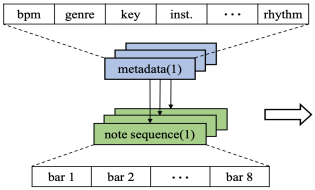
Combinatorial Music Generation

Given a set of short musical samples together with specific metadata describing their properties, combine them to create a complete piece of music.

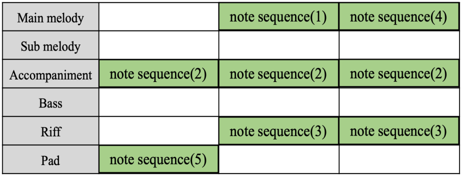
If samples share a subset of the metadata — bpm, key, time signature, number of measures, genre, rhythm, and chord progression — they are guaranteed to sound harmonious when combined together.

Combinatorial Music Generation

- Stage 1: train a machine learning model on ComMU to generate new samples given the desired metadata
- Stage 2: combine the generated samples into a complete piece of music



(a) stage1



(b) stage2

Our contribution: MidiComb

Hyun et al. only focus on stage 1: they train a Transformer-XL on ComMU and show the quality of the generated samples, leaving stage 2 for future work.

MidiComb addresses stage 2 of the combinatorial music generation task by:

1. Taking in input the desired shared metadata values
2. Querying ComMU for a set of samples satisfying those requirements
3. Generating a complete musical composition by combining those samples

Implementation

Music Generation as a Scheduling Problem

The sample arrangement problem is modeled as a scheduling problem, analogous to the job shop problem:

- Each track role (main melody, sub melody, etc.) represents a machine
- Each sample represent a task

Main melody		note sequence(1)	note sequence(4)
Sub melody			
Accompaniment	note sequence(2)	note sequence(2)	note sequence(2)
Bass			
Riff		note sequence(3)	note sequence(3)
Pad	note sequence(5)		

Music Generation as a Scheduling Problem

- Constraints: *how* samples are combined
- Sampling strategy: *which* samples are combined

Constraints

Samples with the same track role do not overlap

$$trackRole_i = trackRole_j \implies \neg\textsc{Overlap}(sample_i, sample_j) \quad \forall i \neq j$$

where

$$\textsc{Overlap}(sample_i, sample_j) \iff end_i > start_j \land end_j > start_i$$

Constraints

Overlapping samples must start at the same time

$$\textsc{Overlap}(sample_i, sample_j) \implies start_i = start_j \quad \forall i \neq j$$

Rationale: without this constraint overlapping samples might be out-of-sync.

Constraints

The sum of track role "weights" at each step must be $\leq$ a threshold

$\textsc{Comulative}(\textit{samples}, \textit{demands}, \textit{capacity})$

where $\textit{capacity} = 6$ and

$$\textit{demand}_i = \begin{cases} 3 & \text{if } \textit{trackRole}_i \in \{\text{mainMelody}, \text{riff}\} \\ 2 & \text{if } \textit{trackRole}_i \in \{\text{subMelody}, \text{accompaniment}\} \\ 1 & \text{if } \textit{trackRole}_i \in \{\text{bass}, \text{pad}\} \end{cases}$$

Rationale: we want "heavier" track roles to overlap with "lighter" ones.

Objective

Minimize makespan

minimize $\max\{end_i\}$

Rationale: this objective encourages the solver to produce more "compact" solutions — without it, a trivial solution would be to arrange the samples sequentially.

Solver

The program is implemented with the CP-SAT interface of the Google OR-Tools framework.

Sampling strategy

- All selected samples must share the input metadata.
- ComMU is queried for one sample for each track role.
- If there is no sample with input metadata + that track role, one of the available track roles is chosen at random and a sample with the new track role is picked.
- When 6 samples (as many as there are track roles) have been selected, the sampling process is completed.

Of these 6 samples, 3 will be repeated throughout the composition: this is achieved by defining an *OptionalIntervalVar* for each sample.

Extension: Sample Generation

Pros and cons

- MidiComb can also work with samples generated by Hyun et al.'s trained Transformer model.
- The CP model does not change, only the sample retrieving process does.
- Although sample generation leads to more diverse musical compositions, their quality is not always on par with their ComMU counterparts.
- A direction for future research might be going beyond the Transformer baseline and devise architectures tailored to music structure, instead of relying on preexisting methodologies developed for natural language.

Demo

frallebini.github.io/midicomb-demo