Music Generation with Long-Term Structure Using Constraint Programming and Transformer-Based Decoders

Liliane-Caroline Demers , Gilles Pesant
Polytechnique Montréal
{liliane-caroline.demers, gilles.pesant}@polymtl.ca

Abstract

Successful music generation with AI techniques requires musical consistency, referring to the repetition of identical or similar musical segments. Sequences generated with Machine Learning (ML) models can imitate the dataset quite fruitfully but have difficulty exhibiting long-term structure. Previous work combined constraint programming (CP) with an ML model at inference time to provide structure to the generated sequences. We explore this work further by automatically injecting constraints closely related to the style of the corpus on which the ML model was trained. We first execute pattern detection on our dataset regarding pitches, rhythms and intervals, and then identify trends within the noted patterns that are used to create musically meaningful constraints in the CP models. Our goal is to produce music samples that express the intended long-term structure while still remaining faithful to the style of the corpus.

1 Introduction

Automatic music generation has been anticipated by scientists since Charles Babbage elaborated the first designs for a computer in 1843. The mathematician Ada Lovelace foresaw that the Analytical Engine could transcend numerical calculations and, one day, be capable of composing music: "Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent." [Menabrea *et al.*, 1843] In recent years, research on applied artificial intelligence (AI) techniques in the broad field of music has expanded exponentially [Moysis *et al.*, 2023]. One need only consider the work of the Sony Computer Lab, projects such as Google Magenta, MusicLM [Agostinelli *et al.*, 2023] and even more recent platforms such as Udio and SunoAI to grasp the current avidity of AI researchers for musical composition.

This subject has been broached in many ways and with different types of algorithms such as Machine Learning (ML). ML models have proven to be powerful tools to generate musical content that can remain consistent with the dataset, while still exhibiting creativity. For example, the Chord conditioned Melody Transformer (CMT) [Choi *et al.*, 2021] produces a melody based on a given chord progression with promising results. However, ML models, including CMT, often lack the ability to manifest long-term structure [Herremans *et al.*, 2017; Briot *et al.*, 2019; Lattner *et al.*, 2018; Lee *et al.*, 2019]. Indeed, successful musical composition hinges upon a display of musical consistency; that is, a general sense of organization provided by long-term structure. Several levels can be distinguished. An example of a more fundamental level of structure is the repetition of identical or similar musical segments, called patterns. Previous neuro-symbolic work [Manibod, 2022; Manibod *et al.*, 2025] addressed this challenge by imposing structure with constraint programming (CP) within CMT during inference. However, the author focused on successfully combining these techniques rather than improving the musicality of the generated melodies. We extend this work by using the same framework but with constraints that bear greater musical significance.

1.1 Research Objectives

Our work focuses on adding long-term structure, specifically repeated patterns at the rhythmic and pitch levels, imposed through constraints, while preserving the style of the corpus. We first perform pattern detection on a dataset of songs of a particular style inspired by Briand [2018]. Then, we identify the distinguishing characteristics of the style of the corpus to create constraints that are more impactful from a musical perspective. Finally, we aim to achieve a relevant long-term structure in the generated melodies by incorporating these constraints into CMT. Note that this work inherits certain restrictions from the CMT architecture such as monophony, a fixed 8-bar length with time signature (4/4) and a minimum subdivision of $1/16^{\text{th}}$ note. Relaxing these constraints to support richer forms of music would require adopting a different sequence model.

2 Background

2.1 Transformer

The Transformer model aims to capture long-range relationships between tokens in a sequence. Its structure consists of two main components: an encoder and a decoder. The encoder is made of a stack of layers, each one composed of two

sub-layers: a multi-head self-attention mechanism and a fully connected position-wise feed-forward neural network. The decoder also comprises a stack of layers, each one including a multi-head attention sub-layer in addition to the two sub-layers contained in each encoder layer [Vaswani *et al.*, 2017].

2.2 CP-Based Belief Propagation

Constraint Programming-Based Belief Propagation (CPBP) [Pesant, 2019] transforms the way constraints are propagated, providing a more precise estimation of the suitability of different values that a variable can take. Unlike standard support propagation, this method takes a more sophisticated approach by calculating the probability that each variable-value pair adheres to a specific constraint. The propagation of constraints becomes a process of continuous refinement of knowledge akin to message passing. Indeed, these probabilities are shared between constraints through the variables they have in common allowing each constraint to adjust its own probability estimates. This iterative message transmission process leads to the derivation of marginal probabilities for each variable-value pair, which quantify the level of constraint satisfaction associated with different value assignments to variables.

2.3 Musical Theory

We briefly define a few key concepts. (See e.g. Benward and Saker [2008; 2009].) Music consists of *notes* and *rhythms*. Notes form *chords*, and organized into *scales* based on the key. We focus on the C Major scale, which includes the seven natural notes C, D, E, F, G, A, B. The smallest distance between two notes is a *half-tone*, and there are 12 half-tones in an octave. An *interval* is the distance between two notes and can be ascending or descending. Tonal harmonic analysis identifies the key and chord structure of a piece.

3 Framework

3.1 Chord Conditioned Melody Transformer

The Chord Conditioned Melody Transformer (CMT) [Choi *et al.*, 2021] produces a melody based on a sequence of chords. The generation of a melody unfolds in two steps. The rhythm is first generated based on a given chord sequence, then the pitch is produced using the results for the rhythms from the first step. In other words, from a given chord progression $c_{1:T} = \{c_1, ..., c_T\}$, CMT's objective is to create the rhythm sequence $r_{1:T} = \{r_1, ..., r_T\}$ and pitch sequence $p_{1:T} = \{p_1, ..., p_T\}$, where each sequence is composed of T tokens and where T represents the total number of time steps.

The CMT's architecture, shown in Fig. 1, consists of three main components: the chord encoder (CE), the rhythm decoder (RD) and the pitch decoder (PD). CE uses a BLSTM to encode the chords, instead of the self-attention mechanism used in the Transformer's encoder. For their part, RD and PD incorporate a stack of N self-attention blocks and autoregressively generate rhythm and pitch tokens, respectively. Output of both decoders passes through an added output layer composed of a fully-connected layer and a softmax layer in order to obtain probability distributions over rhythm and pitch tokens. Rhythm tokens can take three possible values, *onset*, *hold* and *rest*. *Onset* translates to playing a note and *hold* indicates that the previous note is still being played. Possible pitch tokens are 48 MIDI notes, *hold*, and *rest*. Hence, at a particular time step, if the rhythm token is either *hold* or *rest*, the corresponding pitch token must also have the same value. The unit of a time step is a sixteenth note.

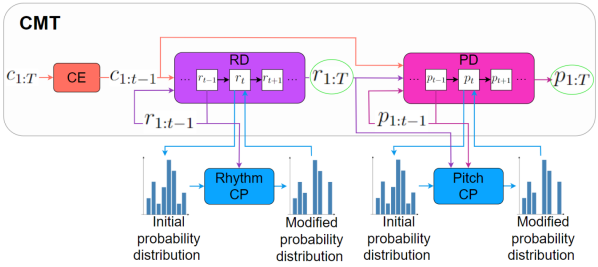


Figure 1: Overview of original CMT architecture during token generation (gray rectangle) combined with CPBP. The CP models modify the probability distribution from which the next token is sampled at each time step (adapted from [Manibod, 2022]).

The generation phase creates the sequences autoregressively, token by token. A probability distribution is calculated to represent the likelihood of all token values for the subsequent step and the next token is thus sampled from this distribution. In the case of the pitch sequence, the sampling is restricted to consider only the top 5 most probable tokens.

3.2 CMT with CP-Based Belief Propagation

CMT with CP-Based Belief Propagation (CMT-CPBP) is a neuro-symbolic framework that attempts to impose long-term structure to the generated melodies with CMT by integrating CPBP at inference time [Manibod, 2022; Manibod *et al.*, 2025]. Fig. 1 illustrates the resulting architecture of CMT-CPBP. There are two CP models, one for the notes and one for the rhythms. As explained previously, at each time step of the generation phase, the decoders' softmax layer produces a probability distribution from which the next token is sampled. However, the probability distributions simply consider the style of the dataset. The CP models with BP modify the probability distribution from which the token is sampled to take into account the satisfaction of constraints. We may add that CP is not used to construct and execute a search tree to find a feasible solution but rather only to restrict the token's domain. The sequence of tokens is represented in the CP models as a sequence of variables whose domain corresponds to the possible token values. The models receive as input the previously generated tokens and the probability distribution from CMT. They use the `oracle` constraint, expressed as:

$$\texttt{oracle}(x_t, p)$$

where p is a fixed probability mass function over the domain of variable x_t. This is a unary constraint that does not impose a relation between variables but rather contributes messages from CMT about the next token represented by x_t during BP. Subsequently, the token is sampled from the resulting marginal probabilities. Thus, it is the `oracle` constraint that permits the successful combination of CMT and CPBP since

it allows the models to respect both the style of the corpus and the constraints.

3.3 Pattern Detection

We also draw inspiration from the pattern detection method of Briand [2018], who decomposes melodies in sequences of notes, rhythms and intervals. Suffix trees were then applied to these sequences to detect patterns that were later used in the melody generation system. Three levels of relevance of patterns were also established by combining patterns from different sequences together.

3.4 Our Contribution

As the literature has highlighted, the lack of long-term structure is a central challenge in the field of automatic music generation. This work is primarily concerned with addressing that problem. First, the note representation we use is formulated relative to its musical context, drawing on concepts from music theory. Inspired by Briand [2018], this work focuses on identifying patterns across an entire corpus, as in Shan and Chiu [2010], rather than within a single piece, which contrasts with the approaches taken by [Briand, 2018; Herremans and Chew, 2017; Conklin, 2021] and others. The types of patterns considered include exact repetitions of note sequences, rhythms, or intervals. Closed patterns, a concept developed by Conklin [2021], or non-trivial, as used by Shan and Chiu [2010], are also included, with an effort to further explore these notions. We carry out pattern detection using regular expressions, a method which, according to the literature review, has not yet been explored in the context of music. Once the patterns are captured, the next step is to filter out overlapping patterns, similar to what Shan and Chiu [2010] did . Next, we perform a statistical analysis of identified patterns, rather than focusing on the nature of any individual one. Much like the MorpheuS system [Herremans and Chew, 2017], which used the COSIATEC algorithm to detect and locate patterns [Meredith, 2013], we emphasize higher-level characteristics of a piece's patterns, such as their positions and lengths. Finally, we use the information extracted from this analysis to design patterns that are then turned into constraints. These constraints are incorporated into CMT-CPBP to guide the generation of melodies. It is worth noting that although the pattern-based constraints enforce certain long-range repetitions, they govern only specific aspects of the musical structure. The Transformer architecture ensures both local and global musical consistency, and captures stylistic dependencies beyond what constraints impose.

4 Music Generation with Long-Term Structure

The overview of our approach is illustrated at Fig. 2. The contribution of this paper is illustrated in the blue rectangle while the work done by Manibod [2022] is presented in the red rectangle. The author addressed the challenge that ML models, such as CMT, face by imposing structure with CPBP within CMT. The goal of this work is to create constraints that hold greater musical significance and that are learned from

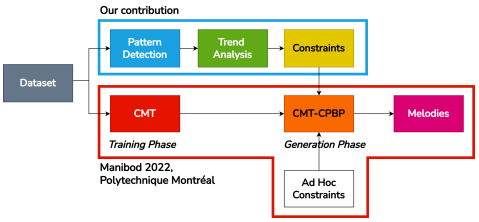


Figure 2: Method overview

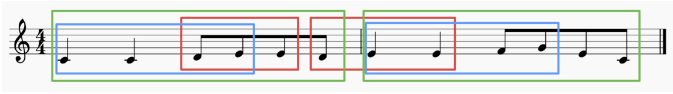


Figure 3: Note (red), rhythm (green), interval (blue) patterns

the dataset. The first step consists in performing pattern detection on a dataset of songs of a particular style. The second step is to execute a trends analysis on the collected patterns in order to identify the characteristics that distinguish the style of the corpus. These characteristics are then used to create constraints.

4.1 Data Representation

The choice of data representation is critical, as it directly influences the patterns detected. A common approach assigns each note a fixed value (e.g., MIDI or semitone 1–12), but this fails to capture modulated patterns. For instance, the pattern $\{C, G, E, C\}$ in C Major, later transposed to F Major as $\{F, C, A, F\}$, would not be matched. To address this, we perform tonal harmonic analysis to identify chords and normalize notes relative to key. In both cases, the pattern becomes $\{1, 5, 3, 1\}$, enabling its detection. Rhythms are encoded as *r* (played) or *s* (rest), followed by a float indicating duration (e.g., *r1.0* for a quarter note). Intervals are calculated as the difference between consecutive notes: positive for ascending and negative for descending. Consider the example melody in Fig. 3. In C Major, the notes are represented as $\{1, 1, 2, 3, 3, 2, 3, 3, 4, 5, 3, 1\}$, rhythms as $\{r1.0, r1.0, r0.5, r0.5, r0.5, r0.5, r1.0, r1.0, r0.5, r0.5, r0.5, r0.5\}$, and intervals as $\{0, 1, 1, 0, -1, 1, 0, 1, 1, -2, -2\}$.

4.2 Pattern Detection

In order to detect relevant patterns within the dataset, we first proceed by capturing all patterns and then continue by filtering them to keep only the relevant ones.

Pattern Capture

The lists of notes, rhythms and intervals are first converted into strings where each element is separated by a comma. The requirements our regular expression must meet are described in Table 1 along with the expression that conveys each of them. After combining the equivalent regex for each requirement, we obtain the following final expression:

```
(?=((?P<pattern>([^,]+\,){3,})
    (([^,]+\,)+)?(?P=pattern)))
```
(1)

Requirement	Regular Expression
1. An element may consist of one or more characters.	`+`
2. A pattern needs to be repeated twice or more.	`(?P<pattern>=...)(?P=pattern)`
3. A pattern has to contain three elements or more.	`3,`
4. Each element is separated by a comma.	`\,`
5. An element can be any character but a comma.	`[^,]+`
6. Patterns may be separated by a # of other elements.	`(([^,]+\,)+)?`
7. An element can be included in more than one pattern.	`(?=...)`

Table 1: Requirements for pattern capture with corresponding regex

Pattern Filtering

The second step of pattern detection is to exclude all patterns that are Pareto-dominated by other patterns. After this filtering step, we use the function `finditer` to identify all the occurrences of the remaining patterns. Formally, a pattern x is said to overlap with other patterns of P if it satisfies the condition $C(x, P)$ where $o(x)$ denotes the number of occurrences of x. This relationship is defined as follows:

$$C(x,P) \Leftrightarrow \exists\ p \in P \mid x \subset p\ \wedge o(x) = o(p) \\ \vee\ \forall\ y \subset x, C(y, P \setminus x) \tag{2}$$

The first clause of relation 2 states that a pattern is considered an overlapping sub-pattern if all of its occurrences are entirely contained within another, longer pattern. For example, in the note sequence $\{G, A, B, G, A, B\}$, the patterns $\{G, A\}$ or $\{A, B\}$ are not retained because they are already included in $\{G, A, B\}$ and occur just as many times. Moreover, the second clause of relation 2 reflects a deliberate choice for a stricter notion of redundancy. It states that a pattern may also be considered overlapping within the set P when all of its sub-patterns are themselves overlapping with other patterns in P. For instance, as illustrated in Fig. 4, in the sequence $\{C, D, E, F, C, D, E, F, C, D, E, F, C, D, E, F\}$, the pattern $x = \{C, D, E, F, C, D, E, F\}$ is overlapping because it contains the sub-pattern $y = \{C, D, E, F\}$, which, according to the first part of relation 2, is already overlapping in $P \setminus x$, given that the pattern exists independently.

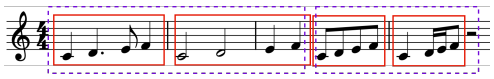


Figure 4: Second clause of relation 2: dotted pattern is removed.

Fig. 3 illustrates examples of note, rhythm, and interval patterns in a melody, detected using regular expression 1 after filtering. The note pattern is $\{2, 3, 3\}$, the rhythm pattern is $\{r1.0, r1.0, r0.5, r0.5, r0.5, r0.5\}$ and the interval pattern is $\{0, 1, 1\}$. Each pattern occurs twice.

4.3 Trend Analysis

Based on the patterns extracted from a set of pieces, the next phase involves gathering information to identify the trends that characterize the style of the data.

Pattern Occurrences

The first type of analysis we perform on the patterns is portrayed in Fig. 7. These heatmaps represent the relative rate of patterns found according to the progression of a piece's temporal progression and the pattern length, for note, rhythm, and interval patterns. The horizontal axis is divided into 16 slices of 6.25%, indicating the time at which the pattern occurs within the melodic progression. The vertical axis is also divided into 6.25% increments, up to 50%, and represents the pattern length. The color indicates the trend of a section x to contain patterns of length y. The closer the color is to yellow, the higher the number of patterns. The heatmap can be interpreted as follows: in section x, there is a $(x, y)\%$ probability of finding a pattern of length y.

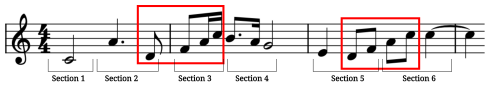


Figure 5: Note pattern occurrences (red) distributed across sections

Thus, this type of heatmap answers the following question: what is the likely length of a pattern that begins in section x?

Construction of the Occurrence Matrix

For each pattern occurrence, we compute the pattern's length and its position in terms of sections. The value of each matrix element is incremented by the proportion of the section that is covered by the occurrence. The matrix is then normalized by dividing each value by the sum of its column. In this matrix, it is the starting position of the pattern that matters, rather than the entire span of its occurrence, because the goal of the resulting heatmap is to determine the length (y) of a pattern that begins in section x. Moreover, the occurrence rate of a pattern of length y in section x must faithfully reflect the importance of that pattern within section x. For this reason, each element of the matrix is incremented proportionally to the coverage of the pattern. Although the heatmap of occurrences provides valuable information, it does not establish a link between the locations of the pattern occurrences. This motivates the need for a second type of analysis.

Pattern Correlation

The second type of analysis conducted on the patterns is illustrated in Fig. 9, and concerns the correlation between different sections of the melody across all pieces in the corpus. Both the horizontal and vertical axes represent 16 segments, each corresponding to 6.25% of a typical piece in the corpus. The colors indicate the degree of similarity between two sections. These heatmaps can thus be interpreted as follows: patterns occurring at $x\%$ of a piece have a $(x, y)\%$ tendency to resemble patterns at $y\%$ of the same piece. The relevance of this correlation analysis lies in its ability to reveal where patterns are likely to emerge across the dataset. This information is essential for applying patterns in a meaningful and strategic way when defining constraints in the next stage.

Construction of the Correlation Matrix

Each matrix entry $correlation[a][b]$ represents the degree of correlation between sections a and b, based on shared pattern occurrences. For each occurrence, we compute its relative coverage within the sections it spans. For example, if a note pattern covers half of section 2 and a full beat in section 3, it contributes 0.25 and 0.5 respectively (see Fig. 5).

The matrix is populated by summing the product of these proportions across all pattern pairs. Each column is then normalized so that it sums to 1, yielding an asymmetric matrix

where column x reflects the influence of all other sections y on section x. The diagonal values quantify self-correlation and are nonzero when a pattern overlaps a section boundary.

Unlike the occurrence matrix in Section 4.3, which only considers pattern start points, this matrix captures patterns' full spans. Ignoring continuation would misrepresent importance, as in Fig. 5, where much of a pattern lies in section 3, despite beginning in section 2. Accurate correlation thus requires proportional attribution across all affected sections.

4.4 Constraint Creation

We now turn to the central question: how can the information extracted from the heatmaps in the previous section be transformed into constraints? The process begins by sampling patterns, which involves selecting the most relevant pairs of sections from the correlation graph and associating them with the most frequent pattern lengths from the occurrence graph. Constraints are then derived from these sampled patterns.

Pattern Sampling

First, the correlation graph provides insight into the degree of similarity between different sections. Pairs of section indices (i, j) are randomly selected from the data in this graph. However, not all section pairs are equally relevant. For this reason, only those above a certain threshold are retained, ensuring that the selected pairs meet a minimum level of relevance. This threshold is defined during experimentation (see Section 6.1). Next, the occurrence graph provides information about the most frequent pattern lengths. The length L of each selected section pair is sampled proportionally to the frequency of occurrence values in column i of the occurrence graph. The combination of a section pair and a sampled length constitutes a **pattern**. Fig. 6 shows a simplified example of how a **pattern** is applied to a melody as a constraint: it occurs twice at positions 25% and 75% with a length of 25%.

Pattern selection follows the rule that only a limited proportion of the composition can be constrained by patterns of a specific type (notes, rhythms, or intervals), to avoid overconstraining the piece. This proportion is also defined during experimentation (see Section 6.1). In the case of an 8-bar composition in 4/4 time, such as those generated by CMT-CPBP, each 6.25% section corresponds to 2 beats. Thus, if the allowed proportion is 50%, constraints can be applied to up to 8 sections, or 4 bars, for each pattern type. Patterns are randomly selected from the pool of eligible candidates we produced. For each selected pattern, the total number of constrained sections is updated by summing the lengths of all selected patterns and doubling that sum (since each pattern spans two sections). New patterns continue to be selected until the maximum allowable number of constrained sections is reached.

Modeling Patterns as Constraints

The sampled patterns are modeled as constraints within a CP framework adapted from Manibod [2022], which incorporates the `oracle` constraint to reshape the probability distribution used by CMT when sampling the next token. All additional constraints described below are original contributions of this work. In both CP models, a vector of 128 variables $x = \{x_1, \ldots, x_{128}\}$ represents the melody to be generated.

Figure 6: Pattern applied to melody as an `equal` constraint

The rhythm model defines $CSP_{rhythm}(X \cup o, D, C)$, where $D(x_i) = \{0, 1, 2\}$. Constraints include `equal`(x_{i+k}, x_{j+k}) for all $0 \leq k < L$, over section pairs (S_1, S_2) of length L (see Fig. 6). To ensure equal numbers of *onset* tokens in both sections, necessary for note and interval constraints, `among` constraints are added:

$$\texttt{among}(S_1, onset, o) \tag{3}$$

$$\texttt{among}(S_2, onset, o) \tag{4}$$

The `oracle` constraint is applied at each time step t:

$$\texttt{oracle}(x_t, p) \tag{5}$$

where p is the marginal distribution over $D(x_t)$.

The note model defines $CSP_{notes}(X \cup X_c, D, C)$, where $D(x_i) = \{0, \ldots, 49\}$. A converted variable set X_c represents notes relative to the current chord (see Section 4.1). The link between X and X_c is enforced via:

$$\texttt{element}(conversion[i], x_i, x_{c,i}) \tag{6}$$

Equality constraints are applied over each section pair:

$$\texttt{equal}(x_{c,i+k}, x_{c,j+k}) \quad \forall\, 0 \leq k < L \tag{7}$$

To enforce octave consistency, we constrain pitch differences in X to be within one octave:

$$\texttt{lessOrEqual}((x_{i+k} - x_{j+k}), 11) \tag{8}$$

$$\texttt{largerOrEqual}((x_{i+k} - x_{j+k}), -11) \tag{9}$$

Interval constraints require matching relative intervals between adjacent notes:

$$\texttt{equal}((x_{c,i+k} - x_{c,i-1+k}), (x_{c,j+k} - x_{c,j-1+k})) \tag{10}$$

To ensure octave consistency, we reuse the pitch constraints 8-9 and additionally apply them to the preceding notes in the interval:

$$\texttt{lessOrEqual}((x_{(i-1)+k} - x_{(j-1)+k}), 11) \tag{11}$$

$$\texttt{largerOrEqual}((x_{(i-1)+k} - x_{(j-1)+k}), -11) \tag{12}$$

The `oracle` constraint is also applied on each note token:

$$\texttt{oracle}(x_t, p) \tag{13}$$

5 Experiments

5.1 Experimental Configuration

The model was trained on a corpus of 457 Irish reels from the Nottingham Dataset[1], selected for its compatibility with CMT's architecture (binary meter, 4/4). Preprocessing followed the procedure in Choi et al. [2021], including data augmentation by key transposition and segmentation into 8-bar excerpts (128 time steps). CMT was trained using negative log-likelihood for the rhythm decoder (RD) and focal loss for the pitch decoder (PD), with hyperparameters matching the original implementation[2]. The dataset was split 80/10/10 for training, validation, and testing.

[1] https://github.com/jukedeck/nottingham-dataset

[2] https://github.com/ckycky3/CMT-pytorch

5.2 Data Preprocessing for Pattern Detection

Following Briand [2018], we converted and preprocessed MIDI files to extract note onsets, durations, and chord information. We reconstructed chords via harmonic analysis so notes could be represented by scale degree relative to the current chord. We computed relative intervals as differences between consecutive notes, and inferred rhythmic values by correcting for MIDI-inserted silences. Since note durations are not proportional to musical time, we expressed pattern positions in absolute time rather than note indices. We then applied pattern detection (Section 4.2) to the processed data.

5.3 Trends

Pattern Occurrences

Fig. 7 shows that 25%-length patterns dominate across all types, especially for notes and intervals. Shorter patterns (6.25%) are also frequent, which is expected as they often recur within longer patterns (see Fig. 8). Moreover, an ending diagonal reflects the last possible occurrence for patterns of a given length near the piece's end.

Pattern Correlation

The heatmaps in Fig. 9 reveal strong repetition within the quarters of the first and last halves of pieces, especially for notes and intervals. For instance, patterns at 0% often reappear at 25% and 40%. In contrast, rhythmic patterns are more evenly spread. The square (notes) and rectangular (rhythm and interval) clusters suggest consistent but asymmetric repetition, i.e. 25%-length rhythm and interval patterns are more likely to diverge near the end of their spans.

Comparison with Another Musical Style

To validate the trend analysis, we applied it to a second corpus, the hornpipes, drawn from the same repository. Fig. 10 illustrates that hornpipes display distinct trends compared to reels: they favour longer patterns (50%) across all types, whereas reels lean toward 25%. Correlation heatmaps in Fig. 11 also show that hornpipes repeat 50% segments more frequently and symmetrically, especially for notes and intervals. Nonetheless, some similarities persist. Both styles show light repetition of 25% segments, and rhythmic patterns tend to be evenly distributed, centered at 25% in reels and 50% in hornpipes. This may reflect a general structural trait of Irish folk music.

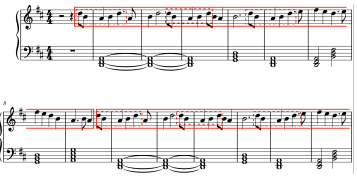
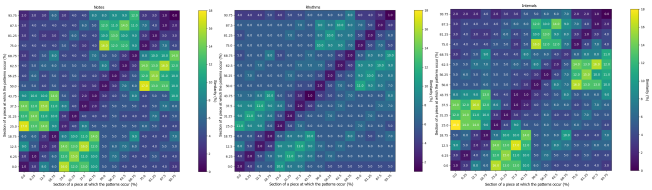


Figure 8: Short note pattern (dotted) repeated within long (solid)



(a) Note patterns (b) Rhythm patterns (c) Interval patterns

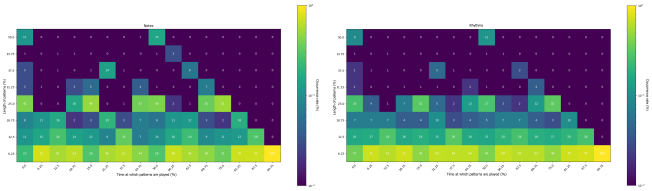
Figure 9: Correlation of pattern occurrences across sections within a piece in the reels corpus

5.4 Constraints

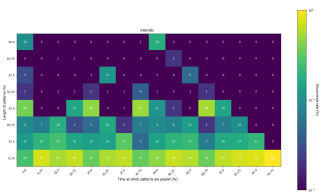
We implemented the constraints in the MiniCPBP solver[3] [Pesant, 2019]. The CP models were integrated into CMT-CPBP architecture[4]. Belief propagation is run using the `vanillaBP` function from MiniCPBP, with 5 iterations. In the occasional event the constraints become unsatisfiable, generation is aborted, and then restarted.

[3]https://github.com/PesantGilles/MiniCPBP

[4]https://github.com/Manibod/CMT_CPBP

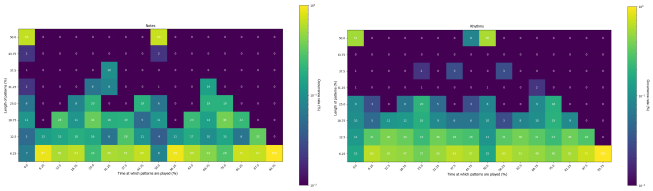


(a) Note patterns (b) Rhythm patterns

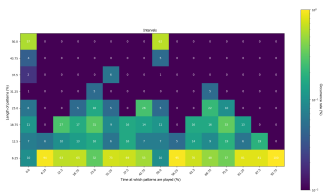


(c) Interval patterns

Figure 7: Pattern occurrences according to piece progression and pattern length in the reels corpus

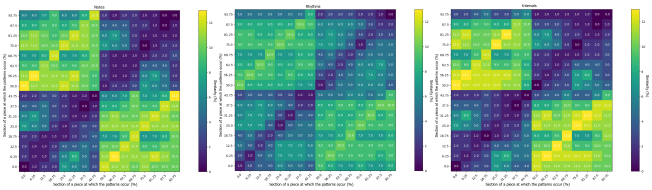


(a) Note patterns (b) Rhythm patterns



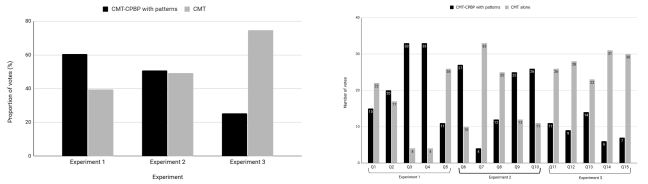
(c) Interval patterns

Figure 10: Pattern occurrences according to piece progression and pattern length in the hornpipes corpus



(a) Note patterns (b) Rhythm patterns (c) Interval patterns

Figure 11: Correlation between pattern occurrences across sections within a piece in the hornpipes corpus



(a) Mean participant preference by model across experiments (b) Participant vote distribution by question and model

Figure 12: Results of subjective evaluation

6 Evaluation

6.1 Experiment Configuration

Pattern construction and selection are governed by two parameters: (1) the section selection threshold, which controls pattern relevance, and (2) the desired coverage, which determines how many patterns to include (see Section 4.4). Three experiments are conducted by varying the threshold (95%, 80%, 0%) while keeping coverage fixed at 100%. These thresholds correspond to selecting sections from the top 5%, top 20%, or with no filtering, respectively. In each case, pattern selection continues until the target coverage is reached. Furthermore, for each experiment, we generate forty musical samples, and we select the top five to form three survey sets. In the first experiment, we use only note and interval patterns; rhythm is handled solely by CMT. This setup proves advantageous: as shown in the trend analysis, rhythmic patterns in reels are uniformly distributed. This suggests that (1) CMT alone can adequately model rhythmic structure, and (2) additional rhythmic constraints are unnecessary, as no distinctive rhythmic features dominate. Samples from this experiment are available for listening here. We generated the samples from the second experiment using patterns selected with greater flexibility, but still drawn from the most significant regions of the graphs. The selected patterns therefore include note, rhythm, and interval patterns. As for the third experiment, the patterns are selected randomly in order to observe the impact of the threshold used to choose sections.

6.2 Subjective Evaluation of Results

We conducted a subjective evaluation with 37 participants via an online survey consisting of 15 identical questions, 5 per experiment. In each question, participants compared two excerpts: one generated by CMT alone, and the other by CMT-CPBP with patterns. The order of excerpts was randomized, and both used the same chord progression. Participants could listen as many times as they wished. Survey results are shown in Figures 12a and 12b, which respectively display vote percentages per experiment and vote counts per question. As shown in Fig. 12a, excerpts generated by CMT-CPBP with patterns outperformed CMT alone in the first experiment (95% threshold), receiving 60.5% of the votes. This preference declined as the threshold decreased: 50.8% in the second experiment (80% threshold), and just 25.4% in the third (0% threshold), where 74.6% of votes favored CMT alone.

6.3 Statistical Testing of Results

We assessed significance using one-tailed Z-tests ($n = 37$). CMT-CPBP was significantly preferred in experiments 1 and 2, while CMT was favored in experiment 3. We compared against critical values $Z_c = 2.58$ (99% confidence) and $Z_c = 2.06$ (98% confidence). For example, CMT-CPBP was significantly preferred in Questions 3, 4, and 6 with $Z = 4.74$, 4.74, and 2.78 respectively (99%), and also in Questions 9 and 10 at 98%. Conversely, CMT was significantly preferred in Questions 5, 7, 8, 11, 12, 14, and 15 (with Z reaching -4.76), confirming that stylistically-guided constraints can produce statistically meaningful shifts in listener preference.

6.4 Observations

The results show that our approach using stylistically-informed constraints improves the musical quality of generated musical excerpts. The higher the threshold, the better CMT-CPBP with patterns performs, often producing excerpts that outperform CMT alone. Conversely, Experiment 3 (threshold of 0%) reveals that imposing constraints without taking into account the characteristics of the dataset introduces arbitrary structure that interferes with CMT's learned musicality, and leads to less natural-sounding outputs.

7 Conclusion

This work addressed the challenge of long-term structure in AI-generated music, specifically, the recurrence of musical segments. Building on prior work integrating CP with CMT [Manibod, 2022; Manibod *et al.*, 2025], we enhance the CMT-CPBP model by introducing style-specific constraints derived from the training corpus. Our results show that pattern-based constraints significantly improve musical quality, and that higher pattern relevance leads to stronger listener preference for CMT-CPBP over the baseline.

Because the chord progression used by CMT is fixed and sampled randomly from the dataset, preventing any control over musical closure. As a result, generated excerpts often end unresolved. Enabling control over chord progressions and analyzing their internal patterns, would be a valuable extension. Another direction of future work is to account for meaningful pattern variations—i.e., perceptually similar sequences with melodic embellishments—by adapting the pattern detection method and allowing partial equality in constraints.

References

[Agostinelli *et al.*, 2023] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. Musiclm: Generating music from text, 2023.

[Benward and Saker, 2008] Bruce Benward and Marilyn Saker. *Music in Theory and Practice*. McGraw-Hill Education, 8 edition, 2008.

[Benward and Saker, 2009] Bruce Benward and Marilyn Saker. *Music in Theory and Practice, Volume 2*. McGraw-Hill Education, 9 edition, 2009.

[Briand, 2018] Alexandre Briand. Génération automatique de mélodie par la programmation par contraintes. Master's thesis, Polytechnique Montréal, 2018.

[Briot *et al.*, 2019] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. Deep learning techniques for music generation – a survey, 2019.

[Choi *et al.*, 2021] Kyoyun Choi, Jonggwon Park, Wan Heo, Sungwook Jeon, and Jonghun Park. Chord conditioned melody generation with transformer based decoders. *IEEE Access*, 9:42071–42080, 2021.

[Conklin, 2021] Darrell Conklin. Mining contour sequences for significant closed patterns. *Journal of Mathematics and Music*, 15(2):112–124, 2021.

[Herremans and Chew, 2017] Dorien Herremans and Elaine Chew. Morpheus: generating structured music with constrained patterns and tension. *IEEE Transactions on Affective Computing*, 10(4):510–523, 2017.

[Herremans *et al.*, 2017] Dorien Herremans, Ching-Hua Chuan, and Elaine Chew. A functional taxonomy of music generation systems. *ACM Computing Surveys*, 50(5):1–30, September 2017.

[Lattner *et al.*, 2018] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. Imposing higher-level structure in polyphonic music generation using convolutional restricted boltzmann machines and constraints. *Journal of Creative Music Systems*, 2(2), March 2018.

[Lee *et al.*, 2019] Jay Yoon Lee, Sanket Vaibhav Mehta, Michael Wick, Jean-Baptiste Tristan, and Jaime Carbonell. Gradient-based inference for networks with output constraints, 2019.

[Manibod *et al.*, 2025] Virasone Manibod, David Saikali, and Gilles Pesant. Constrained sequential inference in machine learning using constraint programming. In *IJCAI*, 2025.

[Manibod, 2022] Virasone Manibod. Ajout de structure aux modèles génératifs de séquences avec la programmation par contraintes. Master's thesis, Polytechnique Montréal, août 2022.

[Menabrea *et al.*, 1843] L.F. Menabrea, C. Babbage, A.K.C. Lovelace, and A.A. L. *Sketch of the Analytical Engine invented by Charles Babbage ... with notes by the translator. Extracted from the 'Scientific Memoirs,' etc. [The translator's notes signed: A.L.L. ie. Augusta Ada King, Countess Lovelace.]*. R. & J. E. Taylor, 1843.

[Meredith, 2013] David Meredith. Cosiatec and siateccompress: Pattern discovery by geometric compression. In *International society for music information retrieval conference*. International Society for Music Information Retrieval, 2013.

[Moysis *et al.*, 2023] Lazaros Moysis, Lazaros Alexios Iliadis, Sotirios P Sotiroudis, Achilles D Boursianis, Maria S Papadopoulou, Konstantinos-Iraklis D Kokkinidis, Christos Volos, Panagiotis Sarigiannidis, Spiridon Nikolaidis, and Sotirios K Goudos. Music deep learning: Deep learning methods for music signal processing-a review of the state-of-the-art. *IEEE Access*, 2023.

[Pesant, 2019] Gilles Pesant. From support propagation to belief propagation in constraint programming. *Journal of Artificial Intelligence Research*, 66:123–150, 2019.

[Shan and Chiu, 2010] Man-Kwan Shan and Shih-Chuan Chiu. Algorithmic compositions based on discovered musical patterns. *Multimedia Tools and Applications*, 46:1–23, 2010.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.