Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2025)

# Time-based Chart Partitioning: Improving Local Coherency in Rhythm Game Chart Generation

**Jonah Hanzen, Emily Halina, Matthew Guzdial**

Department of Computing Science, Alberta Machine Intelligence Institute (Amii)
University of Alberta, Edmonton, Alberta, Canada
jhanzen@ualberta.ca, ehalina@ualberta.ca, guzdial@ualberta.ca

## Abstract

Creating rhythm game charts can be a time-intensive process involving manually synchronizing gameplay elements with audio. Procedural Content Generation (PCG) techniques have been applied to automate this task, but current methods are limited by imprecise onsets and locally incoherent note patterning. In this paper, we introduce Time-based Chart Partitioning (TCP), a chart generation framework that combines neural network-based onset detection, symbolic beat snapping, and a time partitioning pattern matching algorithm. We evaluate TCP on osu!mania, comparing it against AutoOsu, a state-of-the-art chart generation system, and find that TCP achieves higher onset precision and improved local pattern coherence.

## Introduction

Procedural Content Generation (PCG) refers to generating game content through artificial intelligence and algorithmic techniques (Shaker, Togelius, and Nelson 2016). This content can range from levels (Guzdial and Riedl 2016), to game mechanics (Sumner, Saini, and Guzdial 2024), to narrative (Buongiorno et al. 2024). In this paper, we focus on the genre of rhythm games. Rhythm games are a subgenre of music games in which players perform timed inputs in synchronization with musical elements such as beats, notes, or lyrics, often visualized through on-screen cues that correspond to the structure of a song. Despite prior work in applying PCG to rhythm games, generating high-quality rhythm game levels, also known as charts, remains a challenging task (Lee and Jeong 2023; Halina and Guzdial 2021b).

Generating high-quality rhythm game charts depends on two primary factors: note placement and patterning (Halina and Guzdial 2021b; Lee and Jeong 2023; Donahue, Lipton, and McAuley 2017). Note placement is the timing of notes to match the rhythm of the music, and patterning is the arrangement of notes into sequences to reflect the musical structure. Minor inaccuracies in these areas could negatively affect player experience, resulting in charts that feel artificial or disconnected from the music (Sauvé et al. 2018).

While PCG for rhythm game charts has advanced in recent years, existing approaches struggle to simultaneously model precise note placement and human-like local patterning. For example, Dance Dance Convolution (DDC) (Donahue, Lipton, and McAuley 2017) focuses primarily on rhythmic structure, resulting in patterning that lacks musical context and variation (Halina and Guzdial 2021b). AutoOsu (Lee and Jeong 2023) improves upon DDC by jointly predicting note onsets and patterns using a dual-headed architecture. However, we found that these note placements were typically imprecise and required additional post-processing. If there were a framework that could generate rhythmically precise onsets whose notes were context sensitive, it could allow the automated creation of charts that meet competitive and artistic standards.

In this paper, we introduce Time-based Chart Partitioning (TCP), a chart generation framework designed for arbitrary rhythm games that prioritizes onset precision and human-like patterning. Our approach combines deep learning-based onset detection with symbolic pattern matching techniques. We first detect note placements using a convolutional recurrent network, then apply beat snapping to correct small timing imprecisions. We then employ a binary time-partitioning algorithm with pattern matching to construct the output chart from segments of provided human charts. We evaluate TCP in the game domain of osu!mania and compare it against AutoOsu (Lee and Jeong 2023), a state-of-the-art neural chart generation system. Our results show that TCP achieves more rhythmically precise onset placement and generates patterns more consistent with those found in human-authored charts in comparison to AutoOsu.

The contributions of this paper are as follows:

- We introduce **Time-based Chart Partitioning (TCP)**, a novel chart generation framework that emphasizes rhythmic precision and human-like local patterning by combining neural onset detection with pattern-based note type selection.
- As part of TCP, we propose a beat snapping and time-partitioning algorithm that selects segments of human-authored charts through pattern matching.
- We provide an evaluation of our system in the domain of osu!mania, demonstrating improved onset accuracy and human-like patterning compared to AutoOsu.

## Related Work

In this section, we provide an overview of work related to PCG for rhythm games and partitioning in PCG.

### PCG for Rhythm Games

Researchers have applied PCG techniques to various rhythm games (Wang and Liu 2022; Yi, Lee, and Lee 2023; Lin, Xiao, and Riedl 2019). Much of this work focuses on specific subproblems, such as keysound assignment (Lin, Xiao, and Riedl 2019) or audio-visual synchronization (Hoover et al. 2015). Dance Dance Convolution (DDC) was a seminal contribution in chart generation that introduced a two-stage pipeline. DDC employed a convolutional neural network (CNN) and long short-term memory (LSTM) architecture to predict onset placements, followed by a separate LSTM for predicting note types (Donahue, Lipton, and McAuley 2017). However, by decoupling onset detection from note type selection, the model lacked the ability to handle rhythmic context during the patterning stage. This led to repetitive and incoherent local patterns that diverged from structures seen in human-authored charts (Halina and Guzdial 2021b).

TaikoNation introduced an end-to-end supervised learning pipeline that predicted onsets and patterning in a single step (Halina and Guzdial 2021b). This design improved local pattern coherence over DDC by simultaneously considering short-term audio features and recent note history. However, TaikoNation struggled to produce precise note placements and treated patterning as a byproduct of note placement, rather than modelling patterning independently.

AutoOsu employed a dual-headed neural architecture for chart generation that jointly predicts note placements and patterning (Lee and Jeong 2023). While this architecture yields high-quality onset detection on a high level, its individual note placements are often imprecise. This limitation is significant in a rhythm game context where slight timing deviations can disrupt the user's experience. Patterning was not a focus of this approach, and thus the generated charts lack human-like patterns as we demonstrate below. In contrast, our approach improves note placement precision through a beat-snapping post-processing step and promotes local pattern coherence through a time partitioning algorithm drawing on human data.

### Partitioning in PCG

Partitioning is a major component of our approach. Previously, researchers have explored partitioning methods in PCG, particularly in spatial domains for level generation (Snodgrass 2019; Halina and Guzdial 2023).

Snodgrass proposed Example-Driven Binary Space Partitioning, an algorithm that generates game levels by partitioning a sketch into sections and fills them with matching pieces from existing levels (Snodgrass 2019). Tree-Based Reconstructive Partitioning built upon this approach by using Monte Carlo Tree Search playthrough data with binary space partitioning to generate game levels (Halina and Guzdial 2023). Our partitioning approach can be understood as an extension of this line of research.

Constraint-based generators, such as WaveFunctionCollapse (WFC) and Sturgeon, use partitioning techniques to generate game levels. These approaches divide content into overlapping regions and fills them in based on adjacency constraints learned from example data (Karth and Smith 2017; Cooper 2022). Hierarchical WFC is an iterative approach that first performs broad partitions then refines these partitions through successive passes (Beukman et al. 2023). Similarly, TCP first partitions the chart into broad temporal segments, then uses pattern-matching to populate each segment with note types. Another variation, Space-Time WFC, learns constraints across temporal and spatial dimensions (Facey and Cooper 2024). TCP similarly operates in the temporal dimension, as it partitions charts into segments of time. However, due to their focus on traditional 2D levels, these techniques have not been applied to chart generation and would require additional research work to adapt effectively. Thus, despite the use of partitioning, we do not include them as baselines in this work.

## Domain

Rhythm games challenge players to perform precise inputs in time with music. Each level, or chart, specifies a sequence of actions tied to specific points in a song, forming the basis of interaction. We focus on osu!mania 4k, a mode in the rhythm game osu! where notes fall in four vertical lanes, each mapped to a keyboard key. Players must press the correct key as the note falls from the top of the screen toward the judgment line at the bottom. The player's goal is to press the key corresponding to each note as it overlaps with the judgment line. These notes include short notes, which require a single key press, and long notes, which must be held for a duration. Charts are typically designed by humans and represented as text files that specify each note's timestamp, type, and position, enabling synchronization between gameplay and music. A chart's difficulty is measured using a star rating calculated by the game's internal logic. This rating accounts for factors such as note density, timing complexity, and pattern structure. Figure 2 depicts multiple human-authored and generated osu!mania 4k charts.

## System Overview

In this section we provide a high-level overview of Time-based Chart Partitioning (TCP). Figure 1 depicts our chart generation pipeline. The system takes raw audio, associated metadata, and a difficulty value as input, encoding them into a tensor representation. The system also requires a dataset of human-authored charts for pattern matching. We first pass the input tensor through an existing convolutional recurrent network to predict note placements. These note placements are then adjusted through beat-snapping to precise positions within measures. Finally, the system determines note types by matching onsets to similar local patterns in the database of human-authored charts, resulting in a playable chart.

### Input and Requirements

Our system takes three pieces of input: an audio file, corresponding metadata information, and a difficulty value.

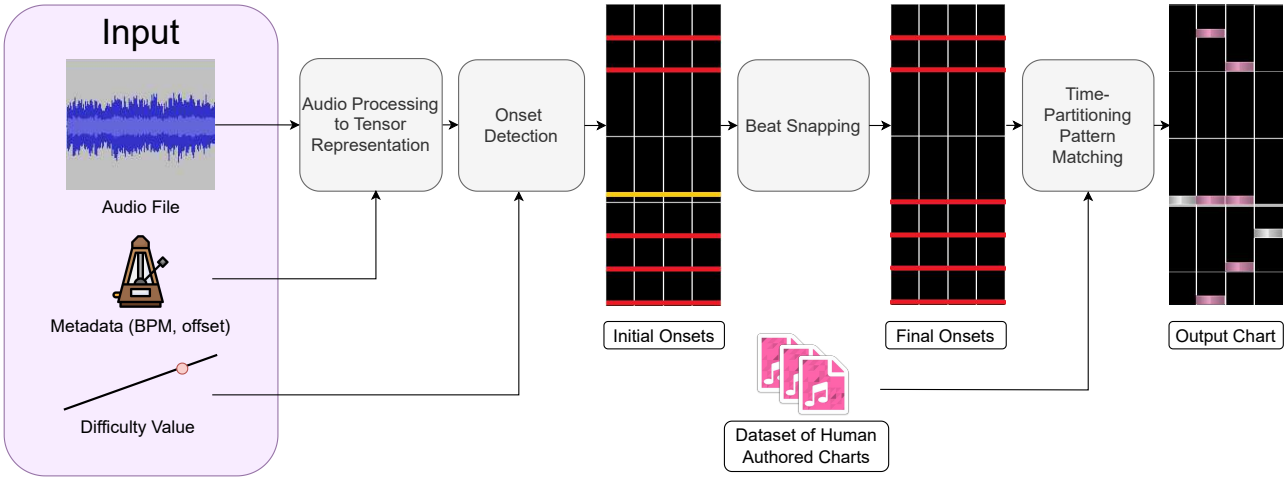


Figure 1: Visualization of the TCP chart generation pipeline. As input, TCP requires an audio file representing the song to chart, associated metadata, a desired difficulty value, and a dataset of human-authored charts from which to draw patterns. After processing the audio input into a tensor representation, we perform onset detection using a neural model. Each onset snaps to the closest measure subdivision via beat snapping before we perform pattern matching to fill in the note types of the output chart.

We process audio files from raw waveforms into a time-frequency representation for onset detection as described below. The metadata information includes required information such as the song's BPM and offset in milliseconds, as well as the song's title and artist. The difficulty value corresponds to the desired difficulty of the generated chart. For the difficulty rating, we follow osu!mania's star rating, which employs a formula authored by the game's developers. Our system also requires a dataset of human-authored charts to use for pattern matching. This dataset is discussed in the Dataset subsection.

## Onset Detection

The first stage in our pipeline involves detecting musical onsets—the moments when osu!mania notes should appear during gameplay. We employ the pretrained onset prediction head from AutoOsu's dual-headed convolutional recurrent neural network (CRNN) architecture (Lee and Jeong 2023). We use the onset detection head exactly as specified in the AutoOsu paper, omitting the note type selection head. Notably, TCP is a modular framework, and any onset detection model could be substituted in this stage. We select AutoOsu's onset prediction head because it is the current state-of-the-art neural model, and using the same model enables a clear comparison between TCP and AutoOsu. First, we process the audio waveforms via the short-time Fourier transform (STFT) with a 10 ms hop size, consistent with prior work (Böck and Schedl 2011; Lee and Jeong 2023). Then we extract log-magnitude spectrograms at 23 ms, 46 ms, and 93 ms resolutions to capture fine, mid, and coarse temporal detail, as in AutoOsu. This processes the raw audio waveform into a workable tensor representation for neural network inference. In addition to this spectrogram data, we compute beat number and beat phase embeddings at each timestep and concatenate these with the spectrogram features. Beat number embeddings represent the current beat position within a measure, and beat phase embeddings represent the subdivision within an individual beat. We pass these features through four convolutional layers with $5 \times 3$ kernels and then into a bidirectional Gated Recurrent Unit (GRU). Its hidden states go through a GELU-activated linear layer and a final linear layer to produce logits over note placements. At inference we select each timestep's prediction by taking the argmax of these logits. For more details on the architecture of our onset detection approach, see the original AutoOsu paper (Lee and Jeong 2023).

## Beat Snapping

The predicted note placements typically fall near the intended beat positions, but often deviate by a few to several milliseconds. This imprecision arises because neural onset detection operates over continuous audio features, which do not conform to the discrete subdivisions found in human-authored charts. In a musical context, these small timing imprecisions can make a chart feel off-beat and unintuitive. To correct this, we snap each onset to the nearest expected timing point within a measure. We define a set of common subdivisions $\mathcal{S}$, composed of quarter notes, triplets, sixteenth notes, and other subdivisions that are frequently used in rhythm game charts. We construct this set as:

$$\mathcal{S} = \bigcup_{d \in \{2,3,4,6,8,12,16,24,32\}} \left\{ \frac{k}{d} \,\middle|\, 0 \le k < d \right\}$$

This approach captures the most common subdivisions found in our dataset. For each predicted note placement, we choose the rhythmic position $\frac{k}{d} \in \mathcal{S}$ closest in time to the

Algorithm 1: Time-Partitioning Pattern Matching

**Input:** snapped onset chart $C$, dataset of human charts $H$
**Params:** partition bound $b$, tolerance $\tau$, min section length $\mu$
**Output:** chart $O$ with note types and holds assigned

initialize $O$ as empty chart
initialize sections with Binary Time Partitioning
initialize section queue $Q$ with sections

**while** $Q \neq \emptyset$ **do**
  $X \leftarrow$ dequeue section from $Q$
  $matches \leftarrow \emptyset$

  **for** each chart $H_i \in H$ **do**
    **for** each window $W$ of size $|X|$ in $H_i$ **do**
      **if** $|X - W| \leq \tau$ **then**
        $matches \leftarrow matches \cup \{W\}$
      **end if**
    **end for**
  **end for**

  **if** $matches \neq \emptyset$ **then**
    $W^* \leftarrow$ random choice from $matches$
    copy note types from $W^*$ to $O$ at positions of $X$
  **else if** time span of $X \geq \mu$ **then**
    split $X$ at random time point into $X_1, X_2$
    enqueue $X_1, X_2$ into $Q$
  **end if**
**end while**

**Return:** chart $O$

Algorithm 2: Binary Time Partitioning

**Input:** section $S = (t_{start}, t_{end}, notes)$
**Params:** partition bound $b$
**Output:** list of sections $sections$

initialize $sections \leftarrow \emptyset$

**procedure** BTP($s, b$)
  **if** $s.t_{end} - s.t_{start} < 2 * b$ **then**
    $sections \leftarrow sections \cup \{s\}$
  **else**
    $t_{split} \leftarrow$ random point in $[s.t_{start}, s.t_{end}]$
    $left\_notes \leftarrow \{n \in s.notes : n.time < t_{split}\}$
    $right\_notes \leftarrow \{n \in s.notes : n.time \geq t_{split}\}$
    $s_1 \leftarrow (s.t_{start}, t_{split}, left\_notes)$
    $s_2 \leftarrow (t_{split}, s.t_{end}, right\_notes)$
    BTP($s_1, b$)
    BTP($s_2, b$)
  **end if**
**end procedure**

BTP($S, b$)
**return** $sections$

model's output. Formally, we find the value in $\mathcal{S}$ that minimizes the absolute distance to the predicted onset within the current measure.

.

## Time Partitioning

Algorithm 1 describes the recursive time partitioning algorithm which assigns note types to onsets. The algorithm performs localized pattern matching using a provided dataset of human-authored charts. We model each chart as a sequence of tuples $(m, s, t, d)$, where $m$ is the measure, $s$ is the subdivision, $t$ is the note type, and $d$ is the hold duration. We first divide the provided list of onsets into sections bounded by a size parameter $b$ as described in Algorithm 2. If a section's length exceeds $b$, the algorithm selects a split point within the section uniformly at random. The tuples are partitioned based on whether they fall before or after the split point. This process acts recursively on both resulting halves of $b$, progressively subdividing the chart into a set of non-overlapping note tuple sections of size $b$ or less that span the full chart. We then attempt to match each resulting section against human authored chart sections of the same length $n$ by sliding a context window of size $n$ across all reference charts. We accept a match if all note tuples within the section perfectly align with the reference chart. If a match is accepted, the system transfers the human-authored reference chart's note types to the corresponding generated onsets, leaving timing and rhythm structure unchanged. If no match is found, the section is further subdivided until either a match is identified or the section is reduced to a single note. In the latter case, matching is trivial: the single note is compared and matched to all single notes in the dataset, and one is chosen uniformly at random.

## Dataset

We use the same dataset as AutoOsu (Lee and Jeong 2023), which contains 1,126 human-authored charts corresponding to 400 distinct songs, and approximately 16.4 hours of audio. The average chart contains 676 notes and is drawn from osu!mania's 4k mode, where each chart has a developer-authored difficulty rating called its star rating. We categorize the charts into three difficulty bins based on star ratings: easy (0.0–2.0), medium (2.0–3.0), and hard (3.0–4.0). We selected this binning scheme to produce a balanced distribution of chart difficulties while remaining consistent with the AutoOsu dataset, which only includes charts rated up to 4.0 stars. This binning resulted in 355 charts labelled as easy, 380 as medium, and 342 as hard. This left us with a total of 1077 charts after processing. The excluded 49 charts had a star rating over 4.0 after recent changes made to the star rating formula. These charts serve as the reference human-authored charts we use for our pattern-matching algorithm to match note types to onsets.

# Evaluation

In this section, we describe our evaluation framework, including the baseline system we compare against, the chart

quality metrics, and the experimental setup for generating and analyzing charts.

## Baseline

We select AutoOsu, a recent state-of-the-art chart generation model, as our baseline. AutoOsu employs a dual-headed convolutional recurrent neural network (CRNN) that jointly predicts onsets and note patterns from audio, conditioned on chart star ratings (Lee and Jeong 2023). We chose AutoOsu as our baseline due to its demonstrated effectiveness at producing reasonable charts and its similar domain focus, making it a strong comparison point for evaluating Time-based Chart Partitioning (TCP). Although TCP uses AutoOsu's onset detection head to approximate note placements, our approach applies an additional post-processing step to align these onsets more precisely and assigns note types through symbolic pattern matching rather than a neural system. Thus, TCP's nature as a hybrid variant of AutoOsu means the comparison will allow us to determine the effect of these variations over a strictly neural model.

Other possible baselines were considered but ultimately excluded. Earlier neural approaches such as Dance Dance Convolution (DDC) (Donahue, Lipton, and McAuley 2017) or TaikoNation (Halina and Guzdial 2021b) are substantially outdated relative to AutoOsu and were designed for different rhythm game formats, making them unsuitable for direct comparison. Ideally, we would also compare against a symbolic or pattern-based chart generator; however, to our knowledge, no such systems currently exist in the academic literature. Community-authored symbolic tools do exist, but they are primarily designed for chart editing or visualization rather than autonomous generation, and thus do not constitute appropriate baselines. For these reasons, we focus our evaluation exclusively on AutoOsu.

## Metrics

High-quality rhythm game charts require both precise note placement and coherent patterning. To evaluate these dimensions, we define four metrics grouped into two categories: note placement precision and patterning.

**Note Placement Precision** To measure the precision of placed notes within a generated chart in comparison to an associated human-authored chart, we employ two metrics: onset precision and onset & note type precision. We note that while these human-authored charts serve as a reference point for evaluation, they are not ground truths in the traditional sense. Charting is a creative process, and multiple valid charts can exist for the same song depending on the designer's style and interpretation. As such, our evaluation treats human charts as a proxy for ground truth.

The first metric, onset precision, compares the set of predicted note onsets to the reference human-authored chart onsets, ignoring note type. This isolates the note placement accuracy without conflating it with pattern structure. For AutoOsu, which does not apply beat snapping, we introduce a small tolerance window $\delta \in \{0.001, 0.003, 0.005\}$ measured in song measures to accommodate the imprecisions in the model's note placement predictions. Since TCP performs beat-snapping, its note placements remain invariant across these tolerance windows, so we omit them. We define a match as a generated onset falling within the tolerance window of a reference human-authored onset. We calculate the F1 score, a standard evaluation metric in onset detection tasks (Böck, Krebs, and Schedl 2012), which balances two factors: precision, the proportion of generated onsets that are correct, and recall, the proportion of successfully recovered ground-truth onsets. Let $M$ be the number of matched onsets, $G$ be the total number of generated onsets, and $T$ be the total number of ground-truth onsets. We compute the F1 score as

$$F1 = \frac{2 \cdot \frac{M}{G} \cdot \frac{M}{T}}{\frac{M}{G} + \frac{M}{T}}$$

The second metric, onset & note type precision, extends onset precision by requiring that matched onsets share the same note type—the column in which the note appears. A match occurs when a predicted onset falls within the tolerance window $\delta$, and has the same note type as a reference human-authored onset. This stricter criterion makes the metric more sensitive to human-like structures and design choices, which notably can vary from author to author. We again report the F1 score computed as described above.

**Patterning** To measure the similarity of patterning between generated and human-authored charts, we employ two metrics: local pattern recall and note type patterns.

Local pattern recall assesses the extent to which generated charts reproduce short-timescale patterns found in human-authored charts. We extract patterns by sliding windows of measure length $m \in \{0.25, 0.5, 1.0, 2.0\}$ across the chart, and record the note placements within these windows. This process yields a set of localized rhythmic patterns over varying timescales. We compute pattern recall as the fraction of unique patterns from the human-authored chart that match patterns in the generated chart, defined as

$$PatternRecall = \frac{|patterns_{gen} \cap patterns_{ref}|}{|patterns_{ref}|}.$$

A match occurs when all note placements within a window align. As with prior metrics, we make use of a tolerance of $\delta \in \{0.001, 0.003, 0.005\}$ measures for AutoOsu to accommodate note placement imprecision.

Note type patterns measure similarity in the sequences of note types from generated charts to human-authored charts, independent of note placement. We extract the note types of all contiguous sequences of $n$ notes from each chart, where $n \in \{2, 3, ..., 8\}$. We then compare the sets of note type sequences and compute the proportion of human-authored note type sequences also found in the generated chart. While local pattern recall captures rhythmic patterns over timescales, note type patterns emphasize the stylistic and game-mechanics aspects of chart patterning. Together, we argue these metrics offer a comprehensive evaluation of patterning, accounting for both rhythmic coherence and the expressive use of note types in human chart design.

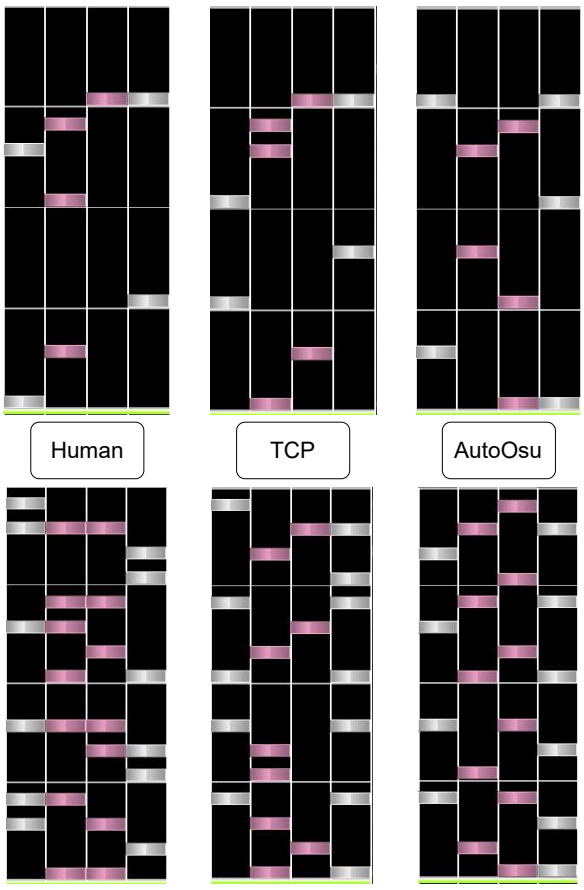


Figure 2: Comparison of generated charts to a human-authored approximated ground truth. Human-authored charts are on the left, our approach's charts are in the middle, and AutoOsu's charts are on the right. We chose charts automatically based on TCP's performance, with the top chart having the best onset + note type accuracy, and the bottom chart having the best onset accuracy across the dataset.

## Experimental Setup

To compare the performance of TCP and AutoOsu, we curated a test dataset of 100 osu!mania charts released between 2024 and 2025. This time range was chosen to avoid any potential training data overlap with AutoOsu, whose model was trained on charts from 2023 and earlier. We split the charts into the same three difficulty bins based on star ratings: easy (0.0–2.0 stars, 63 charts), medium (2.0–3.0 stars, 67 charts), and hard (3.0–4.0 stars, 62 charts). For each chart in the dataset, we used the official AutoOsu codebase on GitHub to generate a corresponding chart using their pipeline. For TCP, we generated five charts per song to account for the stochasticity in its pattern-matching and time-partitioning phases. We tested four different sizes in measures for the time partitioning bound parameter $b \in \{0.25, 0.5, 1.0, 2.0\}$ and observed no substantial difference in performance. We selected $b = 0.25$ for our evaluation since it produced charts with the least self-similarity. We then computed all evaluation metrics described above. We evaluated AutoOsu using tolerance thresholds $\delta \in \{0.001, 0.003, 0.005\}$ to account for lack of beat snapping, while we evaluated TCP using exact matches. For TCP, we scored the five chart variants generated for each song individually, and averaged their metric values to yield a final result per chart.

## Results

Table 1 presents the results of our evaluation across six metrics: Onset Precision, Onset & Note Type, and Local Pattern Recall at four measure lengths $(0.25, 0.5, 1.0, 2.0)$. For each metric, we report the mean and standard deviation across all charts in the test set. Figure 3 complements these results by showing plotted values for the Note Type Patterns metric.

As shown in Table 1, TCP achieves substantially higher onset precision compared to AutoOsu across all tolerance settings. While AutoOsu's performance improves as $\delta$ increases, its scores drop sharply as the threshold approaches zero. This decline highlights the limitations of using a purely neural model for note placement prediction. To validate our results, we conducted a Wilcoxon signed-rank test on the $\delta = 0.005$ threshold setting where AutoOsu performs best. A preceding Shapiro-Wilk test confirmed non-normality in the paired F1 differences ($p < 0.005$), motivating the use of a non-parametric test. The Wilcoxon test found the improvement by TCP to be statistically significant ($p < 0.005$). These findings demonstrate the effectiveness of beat-snapping in enabling TCP to achieve more rhythmically precise note placement compared to the raw onset predictions of a neural model.

For onset & note type precision, we observe that TCP performs comparably to AutoOsu under its most permissive tolerance setting. Although AutoOsu achieves slightly higher average performance under $\delta = 0.005$, this difference was small. Further we found from our own observation that $\delta = 0.005$ represented too high of a tolerance, with clearly incorrect notes considered correct. The similarity in the results is encouraging for TCP, as it achieves comparable performance to a neural model that jointly learns onset timing and note type, despite using a symbolic pattern-matching strategy for note type assignment. We again ran statistical tests for this data but were unable to reject the null hypothesis that the metric values fell within the same distribution.

Additionally, TCP substantially outperforms AutoOsu in local pattern recall, the metric that captures similarity to local rhythmic patterning in human charts. Both systems exhibit a decline in local pattern recall as the pattern length increases, which we expected given that a match requires every note within the pattern length to align with the human-authored chart. As such, longer pattern lengths, such as two measures, are substantially harder to match in full. Nevertheless, TCP exhibits a statistically significant advantage over AutoOsu across all pattern lengths, as determined using the same Wilcoxon signed-rank test procedure with $p < 0.005$. This supports our claim that TCP is more effective at pro-

| | TCP | AO ($\delta = 0.001$) | AO ($\delta = 0.003$) | AO ($\delta = 0.005$) |
|---|---|---|---|---|
| **Onset Precision F1** | $\mathbf{0.765 \pm 0.147}$ | $0.208 \pm 0.129$ | $0.547 \pm 0.205$ | $0.742 \pm 0.191$ |
| **Onset & Note Type Precision F1** | $0.134 \pm 0.041$ | $0.038 \pm 0.024$ | $0.010 \pm 0.044$ | $0.136 \pm 0.047$ |
| **Local Pattern Recall (0.25)** | $\mathbf{0.829 \pm 0.187}$ | $0.499 \pm 0.319$ | $0.720 \pm 0.248$ | $0.809 \pm 0.194$ |
| **Local Pattern Recall (0.5)** | $\mathbf{0.648 \pm 0.255}$ | $0.255 \pm 0.297$ | $0.499 \pm 0.282$ | $0.587 \pm 0.259$ |
| **Local Pattern Recall (1.0)** | $\mathbf{0.313 \pm 0.230}$ | $0.077 \pm 0.156$ | $0.171 \pm 0.188$ | $0.256 \pm 0.210$ |
| **Local Pattern Recall (2.0)** | $\mathbf{0.093 \pm 0.138}$ | $0.024 \pm 0.061$ | $0.029 \pm 0.065$ | $0.069 \pm 0.111$ |

Table 1: Comparison of TCP and AutoOsu (AO) across our metrics. For each metric, we provide the mean value across all generated charts and the standard deviation. Statistically significant results are bolded. $\delta$ represents the threshold value used for AutoOsu, given in measures. TCP charts were generated with a section size of 0.25.

ducing local rhythmic patterning similar to human-authored charts over varying short timescales.

In addition to local pattern recall, we analyze note type pattern similarity across varying sequence lengths. Figure 3 visualizes the proportion of $n$-note type sequences shared between generated and human-authored charts, ignoring onsets entirely. The graph shows that TCP outperforms AutoOsu at shorter pattern lengths of two to four note sequences, indicating stronger local patterning coherence. At longer note sequences, AutoOsu holds a slight edge, indicating that a fully neural approach may have a broader notion of context over our locally focused approach. This trade-off reflects the contrast between TCP's localized pattern matching and AutoOsu's learned global structure.

Figure 2 depicts a comparison between two sets of human-authored and generated charts for each approach. When reading the figure, it may be helpful to consider each row of the generated charts in comparison to the human chart both in terms of note placement and type.

Overall, our results demonstrate that TCP improves in note placement precision and coherent local patterning. Across all metrics, TCP either outperforms or matches AutoOsu, with statistically significant performance in onset precision and local pattern recall. Importantly, TCP's use of beat-snapping enables evaluation using exact matches, whereas AutoOsu requires a tolerance window to compensate for imprecise predictions. This tolerance acts as a corrective buffer rather than a principled solution, highlighting a core advantage of our hybrid symbolic-neural system.

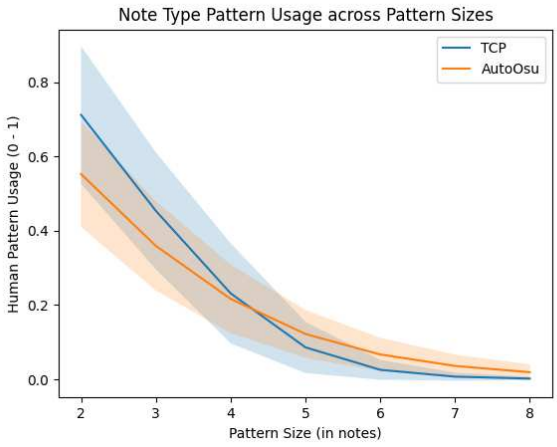


Figure 3: Line graph comparing note type pattern usage by TCP and AutoOsu. Standard deviation is highligthed around each line.

## Discussion

In this section, we examine the limitations of our approach and outline potential directions for future work.

### Limitations

Currently, we have only adapted TCP for the osu!mania domain, though we argue that the TCP framework is extendable to other rhythm games. However, this extension may require changes to the algorithm and onset detection model depending on domain-specific information. Additionally, since TCP uses human-authored charts, it inherits any limitations in the quality of those charts. This constraint is also relevant in practical applications where TCP might be used with charting tools that assist with pattern placement. In such contexts, replicating poor design choices from the training data could compromise the quality of the generated content. However, the reliance on human-authored charts can also be viewed as a strength: TCP gives users full control over the dataset used to guide generation, allowing them to vet for high-quality examples and deliberately select charts of a particular style they wish to capture. A further limitation is the absence of human evaluation in this work. While our quantitative metrics demonstrate significant improvements over the baseline, playtesting with human participants remains the gold standard for assessing chart quality and player experience. Future work could address this gap through user studies and co-creative evaluations. Finally, TCP's note placements are limited by the quality of its neural onset detection system. While our beat-snapping procedure improves note placement precision, it cannot fully compensate for inaccurate onset predictions. That said, because TCP is modular with respect to onset detection, future improvements in onset models can be seamlessly integrated into the framework.

### Future Work

We see several promising directions for extending the TCP framework. One avenue is to adapt TCP to additional rhythm game domains beyond osu!mania. However, for particular

rhythm games, this may introduce unique challenges. For example, games such as Sound Voltex incorporate twistable knobs as input, which require modeling continuous trajectories rather than discrete notes. Additionally, dance-based games such as StepMania and Dance Dance Revolution introduce embodied constraints, where certain sequences are physically difficult or even impossible to execute due to the limitations of human movement (Franks et al. 2023). Similarly, games that use hardware-specific input devices, such as Guitar Hero, require consideration of controller ergonomics (Yuan and Folmer 2008). Another avenue of exploration involves incorporating global patterning into the chart generation framework. While TCP emphasizes local pattern coherence, rhythm game charts often rely on recurring motifs. Enhancing TCP with mechanisms for identifying and reusing such global patterns could further improve the quality of generated charts. We also envision integrating TCP into co-creative charting tools, such as KiaiTime (Halina and Guzdial 2021a), where it could serve as an intelligent assistant to human designers. In this setting, TCP's reliance on human-authored chart data could be leveraged as a strength, supporting collaborative workflows where designers retain control while benefiting from automated suggestions. Beyond rhythm games, the hybrid symbolic-neural structure of TCP may hold promise for related domains such as sound engineering or other forms of temporally structured procedural content generation. Exploring this broader applicability represents another fruitful direction for future work. In addition, human evaluation remains an important next step. While our quantitative evaluation demonstrates statistically significant improvements, playtesting with human participants represents the gold standard for assessing chart quality and player experience. Finally, TCP's modular design allows future advances in onset detection to be easily incorporated. Although we currently use the neural onset head from AutoOsu, any improved detector can replace it without changes to TCP's core beat-snapping and time-partitioning algorithms.

## Conclusions

In this paper, we presented Time-based Chart Partitioning (TCP), a novel chart generation framework for osu!mania. We evaluated TCP against a neural baseline and found that it achieves more precise note placements and generates local patterns that more closely resemble those authored by humans. In the future, we hope TCP can support a broader range of rhythm game formats and assist human designers in generating higher-quality charts.

## Acknowledgements

This work was funded by the Canada CIFAR AI Chairs Program. We acknowledge the support of the Alberta Machine Intelligence Institute (Amii).

## References

Beukman, M.; Ingram, B.; Liu, I.; and Rosman, B. 2023. Hierarchical wavefunction collapse. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 19, 23–33.

Böck, S.; Krebs, F.; and Schedl, M. 2012. Evaluating the Online Capabilities of Onset Detection Methods. In *ISMIR*, 49–54.

Böck, S.; and Schedl, M. 2011. Enhanced beat tracking with context-aware neural networks. In *Proc. Int. Conf. Digital Audio Effects*, 135–139.

Buongiorno, S.; Klinkert, L. J.; Chawla, T.; Zhuang, Z.; and Clark, C. 2024. Pangea: Procedural artificial narrative using generative ai for turn-based video games. *arXiv preprint arXiv:2404.19721*.

Cooper, S. 2022. Sturgeon: tile-based procedural level generation via learned and designed constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 18, 26–36.

Donahue, C.; Lipton, Z. C.; and McAuley, J. 2017. Dance dance convolution. In *International conference on machine learning*, 1039–1048. PMLR.

Facey, K.; and Cooper, S. 2024. Toward space-time WaveFunctionCollapse for level and solution generation. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 20, 25–34.

Franks, B. J.; Dinkelmann, B.; Fellenz, S.; and Kloft, M. 2023. Ordinal Regression for Difficulty Estimation of StepMania Levels. *arXiv preprint arXiv:2301.09485*.

Guzdial, M.; and Riedl, M. 2016. Game level generation from gameplay videos. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 12, 44–50.

Halina, E.; and Guzdial, M. 2021a. A demonstration of kiaitime: A mixed-initiative pcgml rhythm game editor. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 17, 240–242.

Halina, E.; and Guzdial, M. 2021b. TaikoNation: Patterning-focused chart generation for rhythm action games. In *Proceedings of the 16th International Conference on the Foundations of Digital Games*, 1–10.

Halina, E.; and Guzdial, M. 2023. Tree-based reconstructive partitioning: a novel low-data level generation approach. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 19, 244–254.

Hoover, A. K.; Cachia, W.; Liapis, A.; and Yannakakis, G. N. 2015. Audioinspace: Exploring the creative fusion of generative audio, visuals and gameplay. In *International Conference on Evolutionary and Biologically Inspired Music and Art*, 101–112. Springer.

Karth, I.; and Smith, A. M. 2017. WaveFunctionCollapse is constraint solving in the wild. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, 1–10.

Lee, S.; and Jeong, D. 2023. AutoOsu: Audio-Aware Action Generation for Rhythm Games. In *Ismir 2023 Hybrid Conference*.

Lin, Z.; Xiao, K.; and Riedl, M. 2019. Generationmania: Learning to semantically choreograph. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 15, 52–58.

Sauvé, S. A.; Sayed, A.; Dean, R. T.; and Pearce, M. T. 2018. Effects of pitch and timing expectancy on musical emotion. *Psychomusicology: Music, Mind, and Brain*, 28(1): 17.

Shaker, N.; Togelius, J.; and Nelson, M. J. 2016. Procedural content generation in games.

Snodgrass, S. 2019. Levels from sketches with example-driven binary space partition. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 15, 73–79.

Sumner, M.; Saini, V.; and Guzdial, M. 2024. Mechanic maker: accessible game development via symbolic learning program synthesis. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 20, 235–244.

Wang, Z.; and Liu, J. 2022. Online game level generation from music. In *2022 IEEE Conference on Games (CoG)*, 119–126. IEEE.

Yi, J.; Lee, S.; and Lee, K. 2023. Beat-Aligned Spectrogram-to-Sequence Generation of Rhythm-Game Charts. *arXiv preprint arXiv:2311.13687*.

Yuan, B.; and Folmer, E. 2008. Blind hero: enabling guitar hero for the visually impaired. In *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, 169–176.