HarmonyMapper: Generating Emotionally Diverse Chord Progressions for Games

Sara Cardinale[1,*,†], Oliver Withington[1,*,†]

[1]Queen Mary University of London, London, UK

Abstract

In this paper, we introduce our system HarmonyMapper, a system for generating diverse musical sequences. By combining the MAP Elites algorithm with insights from music theory, specifically Neo-Riemannian Theory, HarmonyMapper is able to generate sets of chord sequences with a high-level diversity in terms of the emotions they are expected to elicit. Our intention is that this system will form the basis for a mixed-initiative musical composition tool focused on generating diverse music for digital games. While HarmonyMapper is limited in the musical complexity it can produce, our pilot experiments showed that it is capable of generating large numbers of varied sequences with no input data and only limited set up and computational resources required.

Keywords

Procedural Music, Neo-Riemannian Theory, MAP-Elites, Computational Creativity, Quality-Diversity Algorithms

1. Introduction

Generative music has a long history in digital games. From the early dynamic music systems of 1994's System Shock [1] and 1998's Thief: The Dark Project [2] to modern examples with greater sophistication such as Dying Light 2 [3], the capacity of generative music systems to react dynamically to the state of the player has long been prized for their capacity to increase immersion and reduce listener fatigue (see [4] for an overview of these systems in gaming). While generative music for games has typically focused on producing individual musical sequences which are maximally appropriate for a given game-play situation, we argue that there is significant potential benefit in instead generating sets of intentionally diverse musical compositions which are appropriate for diverse moods and events in game.

In this paper, we explore the use of the MAP-Elites algorithm, a genetic search-inspired algorithm that finds sets of diverse solutions rather than fit individuals, and apply it to the challenge of generating diverse chord sequences. To facilitate the generation of these sequences we use a branch of transformational music theory, Neo-Riemannian Theory (NRT) [5], which allows us to generate sequences of chords that smoothly transition between each other without the system needing to directly control or evaluate these transitions. We also make use of the seven modes, scales that have specific characteristics, and harmonic behaviours. To generate chord sequences that are diverse in terms of the moods and emotions they are likely to elicit we use insights from NRT but also music theory more broadly to design heuristics that evaluate chord sequences and aim to quantify these characteristics.

We find that by combining the power of MAP-Elites with insights from music theory and NRT we are able to generate sets of diverse chord sequences without requiring any training data or human guidance during generation, and in a way that is relatively computationally light. Although the current system we present is capable of generating linear chord sequences with some musical complexity limitations, we view it as a promising foundation for a more advanced mixed-initiative system to create game music. Such a system would be useful for generating music that is both harmonious, but also diverse in the moods it elicits, allowing a designer or composer to make music that is appropriate for all of the diverse moods that the game in question wants to elicit.

In Section 2 we discuss the related research that is most directly relevant to that presented here. In Section 3 we describe and justify the current implementation of our music generation system. In Section 5 we discuss the configuration we used for the pilot experiments using this system, and in Section 6 we present and discuss the ramifications of the results of these experiments. Finally, in Section 7 we discuss the future work we intend to carry out to improve and advance this music generation system, and we conclude that while limited, this system is potentially an exciting and useful foundation for future music generation systems focused on games.

AIIDE Workshop on Experimental Artificial Intelligence in Games, October 08, 2023, University of Utah, Utah, USA

*Corresponding author.

†These authors contributed equally.

s.cardinale@qmul.ac.uk (S. Cardinale); o.withington@qmul.ac.uk (O. Withington)

saracardinalemusic.com/ (S. Cardinale); owithington.co.uk (O. Withington)

0000-0002-seven00seven-5193 (O. Withington)

2. Related Work

2.1. Music Generation Systems

The generation of chord progressions has been investigated for the automatic generation of musical harmony to accompany various forms of media. Bernardes et. al. [6] implemented the D'accord harmony generation system which worked over a perceptually motivated tonal interval space. Monteith et al. [7] generated music to induce targeted emotions, using statistical techniques such as HMMs, and applied this in [8] to produce affective music to accompany the audio of fairy tales being read.

Numerous generative systems have been used to create soundtracks for videogames. For instance, the game GhostWriter (1998) [9] uses a generative approach to create the music. The system implements a rule-based algorithm that maps the in-game tension to the music. To create a chord progression, the system implements Schoenberg's Theory of Harmony [10]. While the compositional aspect is conducted by a generative system, the detection of in-game tension is not an automated process: a director identifies and maps the level of suspense in the music to the in-game events.

More recently, Lopez Duarte [11] introduced an algorithmic music generator that makes use of a second or third-order Markov model to manage instrumental range and quantity, RNN for melody and bass, and an Augmented Transition Network (ATN) to provide a tonal grid. Duarte states that a system which combines generative and algorithmic (rule-based systems, deterministic approaches) could provide extensive musical variety in the structure and musicality while supporting the style and offering continuity [11].

2.2. Neo-Riemannian Theory

Neo-Riemannian Theory (NRT) is a branch of transformational theory that provides analytical methods to study chromatic chord progressions that deviate from traditional tonality. Conventional music theory, which relies on keys and modes to analyze harmonic developments, often struggles to effectively analyze these types of progressions. NRT introduces Neo-Riemannian Operators (NROs) to examine chord changes. These operators describe the transitions between chords in a sequence, allowing for the analysis of chord progressions that go beyond traditional tonal frameworks. These NROs can be applied individually to a chord to produce a new chord, or several can be applied at once to produce a Compound NRO (CNRO) which is more distant from the initial chord.

NRT does not just allow for the description of non-traditional progressions; it also allows them to be analysed. Through the analysis of diverse film scores using NRT, Lehman observed a link between the chosen NROs and compound NROS employed by composers and the specific events or emotions they aimed to portray [12]. By identifying the NROs and CNROs used in a progression we can to an extent infer the likely emotional response of a potential listener.

NRT has been used as the basis of generative music for games before in the work of Cardinale and Colton [13]. They introduced procedural NRT and suggested that it could be used in conjunction with Lehman's observations to write event-based music for media, including games. This led to the development of their GENRT system which used NRT to procedurally generate musical sequences which matched with a given specification such as matching a specified tempo, or by incorporating changes in the music's mood using Lehman's descriptors at specific time stamps. We expand on these ideas by generating sets of diverse sequences rather than individual sequences, though the current version of our system produces simpler music than GENRT which is capable of generating sequences with a melody, harmony, bassline, and percussion rather than just a chord sequence.

2.3. MAP-Elites and Music Generation

The algorithm at the centre of this work's music generation system is the 'Multi-dimensional Archive of Phenotypic Elites' algorithm, more commonly referred to as MAP-Elites. MAP-Elites is the most prominent algorithm within a family of relatively modern algorithms referred to as 'Quality-Diversity' (QD) algorithms. These algorithms work similarly to traditional genetic search methods, which aim to find high-performing solutions to problems by combining and mutating the best solutions found so far, in a way heavily influenced by real-world evolution. Where QD algorithms distinguish themselves is by generating populations of solutions rather than fit individuals, and by aiming to maximise the phenotypic diversity of this population.

MAP-Elites achieve quality and diversity by discretising the search space for solutions into an evenly spaced grid, with each cell in the grid defined by a range of calculable phenotypic traits (Phenotype/phenotypic in this context meaning characteristics of the actual generated artefact, in our case a musical sequence. This is opposed to genotype/genotypic which refers to the encoded form of the artefact). Classically this grid is defined by two dimensions, though there are implementations that work in higher dimensions [14]. Each cell in the grid is only allowed to contain a fixed number of solutions, and the membership of the cell is determined by a fitness function. If a new solution is generated and its phenotypic traits mean it should be added to a cell that is already at capacity, then the fitness of all solutions is calculated and the least fit solution is removed. Each loop of the algorithm previously generated solutions are selected

from the archive, new solutions are generated from them and then they are added to the archive. By repeating this process until a stopping condition is met such as a certain number of loops, the intention is to arrive at a final grid populated with a set of solutions that are both high performing and diverse in terms of designer-selected traits.

MAP Elites is a relatively new algorithm, having been introduced by Mouret and Clune in 2015 [15]. However it and its variants have already seen significant success in domains as varied as generating robot designs [16], generating digital artworks [17], controlling swarms of micro-robots [18], and many others. It has also seen significant popularity in the generation of content for digital games, especially game levels [19, 20, 21].

A domain that has seen little use of QD algorithms is the domain of music generation (that the authors are aware of). However, they have seen success in similarly creative tasks such as digital drawing. McCormack et al.[22] employ quality-diversity search techniques to investigate a creative generative system based on an agent-based line drawing model. The application of quality-diversity search allowed for the discovery of multiple high aesthetic value phenotypes within the system.

Berker and Colton have recently argued that QD algorithms for music generation could be a good fit for the field [23]. We agree that it is a potentially good fit for multiple reasons. There is a substantial amount of music theory that can be leveraged to produce interesting phenotypic metrics with which to evaluate generated music. Music can also be encoded genotypically in a form that makes it easy to manipulate using conventional genetic operators, especially when using NRT as we do in this paper.

3. Methodology

3.1. Musical Encoding

To generate our musical sequence we need a genotypic form for them which can be easily manipulated by genetic operators, while also being convertible into midi audio with limited effort and compute. This is where NRT works well. NRT describes a series of Neo-Riemannian Operators (NROs), conventionally labeled with a letter, which can operate on a trichord to produce a new trichord. These NROs can be combined together and applied in sequence to a starting chord to produce a Compound NRO (CNRO). The musical encoding we use in this version of our system is to store tracks as an ordered array of CNROs along with a starting chord. From this we can generate the series of chords produced by applying these CNROs in order, forming a musical sequence that can be converted to midi at any point.

To apply these CNROs to a chord, we need to store the chord in the form of an array of midi note numbers. A midi note number is a widely used encoding form within music in which each note is assigned a number from 0 to 127, with middle C found at number 60. For example, a C Major chord in the 4th octave composed of the notes C4, E4, and G4 can be encoded as [60, 64, 67]. Each NRO in traditional NRT can be described as a set of additions and subtractions to the notes of a trichord in midi note number form (see Table 1). This makes applying an NRO to a trichord a mathematically trivial operation. However, after applying an NRO to the midi note number, its form may have to be reordered to place the trichord in its root position. This is also complicated by the fact that each NRO letter operates on note numbers differently depending on whether it is applied to a Major or a Minor triad.

Table 1
List of Every NRO in their Forms for Acting on Trichords in Note Number Form

NRO Letter	Major Form			Minor Form		
R	0	0	2	-2	0	0
P	0	-1	0	0	1	0
L	-1	0	0	0	0	1
N	0	1	1	-1	1	0
M	-2	-2	0	0	2	2
S	1	0	1	-1	0	-1

3.2. Phenotypic Metrics and Fitness Function

As discussed in Section 2.3, a central decision when implementing MAP-Elites is on the choice of phenotypic metrics to use to determine where generated artifacts are placed within the grid archive. One of the contributions of this work is the introduction of metrics that can be calculated for a musical sequence. These metrics are informed by musical theory and are intended to capture specific characteristics of musical pieces which are meaningful to listeners, and therefore characteristics that a game designer or composer might like to have diversity in to fit diverse game settings.

3.2.1. Fitness Function: Key Prediction

To quantify the fitness of the generated chord sequences we experimented with several alternatives before arriving at the option that gave the most appealing results: the strength of fit between a chord sequence and its predicted key. To analyse the predicted key of a piece we use the Krumhansl-Schmuckler Key-Finding Algorithm [24]. By comparing the distributions of pitches in a musical

Algorithm 1 Illustration of the Process for Generating a List of Chords from a Start Chord and a List of Compound NRO Objects

Require: $Start\_Cord, CNRO\_List$
  $Chord\_List = []$
  $Chord\_List \leftarrow Chord\_List + Start\_Chord$
  for CNRO IN CNRO_List **do**
    for NRO IN CNRO **do**
      $Current\_Chord \leftarrow GetLastChord(Chords)$
      $Is\_Major \leftarrow GetIsMajor(Current\_Chord)$
      if Is_Major **then**
        $New\_Chord \leftarrow ApplyMajorNRO(NRO, Current\_Chord)$
      else
        $New\_Chord \leftarrow ApplyMinorNRO(NRO, Current\_Chord)$
      end if
      $New\_Chord \leftarrow MoveToRootPosition(New\_Chord)$
      $Chord\_List \leftarrow Chord\_List + New\_Chord$
    end for
  end for
  return $Chord\_List$

sequence with those of the fixed key profiles which they developed, one can predict the key of any piece of music.

In this work, we use *Music21* [25], a popular musicology Python Library, and their implementation of Krumhansl-Schmuckler's Algorithm. This not only gives us a possible key of a piece, which is used later for predicting the mode of our generated music, but it also gives us the correlation coefficient of the confidence level of the algorithm. The higher the coefficient, the better the distribution of pitches matches the predicted key. This is important to know, as it is more than possible for a piece to fit best with a certain key, but to still only weakly match its distribution. It is this correlation coefficient that we use as our fitness function, with the goal of pushing the system to produce music that is strongly within one key and therefore that is likely to sound more cohesive to a listener. However, this does significantly limit the variety of music that will be generated, something we discuss further in Section 7 on our future plans with this system.

4. Implementation

4.0.1. Average CNRO Shift

The first of the two metrics that we use is what we term Average NRT Shift. This metric is based on the observations of musicologist Frank Lehman [12] that there is a relationship between how many NROs are used between two chords in a sequence and how unsettling a piece is. This could be attributed to the fact that when using only one NRO, the resulting chord is more likely to be in the same key or a closely-related key, whereas as you increase the size of the CNROs used, the resulting chord tends to be further away from the original key. In a series of experiments carried out by Krumhansl [26], it was shown that tones less closely related to the tonality are less stable than tones closely related to the tonality. As tonality is inferred through harmonic context and the interplay of tension and release among surrounding chords in relation to a specific tone or chord base [27], the absence of a tonal centre can be said to create a sense of uncertainty.

We operationalise this metric by summing all of the NROs in all CNROs that define a track, and then dividing this number by the number of CNROs, giving us the mean number of NROs per chord transition. We also argue that this is a natural characteristic to want to find diversity in when generating video game music, as varying levels of tension during the course of play is common to so many genres of game.

4.0.2. Predicted Mode

The second metric we use is the predicted mode of a piece. As discussed in Section 1, music theory describes seven musical modes. These are highly useful for our purposes as prior research has found that they can be ordered from brightest-sounding, Lydian, to darkest-sounding, Locrian, as shown in Figure 1 [27].

There are several steps involved in HarmonyMapper's process for predicting the mode of a generated piece of music. First, we take the predicted key of the piece using the same Krumhansl-Schmuckler's Algorithm [24] which we use in calculating fitness. For this key, we then generate the scale of that key for each of the seven modes, giving us seven scales, each containing 7 notes, to

Table 2
Modes and their character notes.

Mode	Character Note
Ionian	Natural 4th
Lydian	Raised 4th
Mixolydian	Lowered 7th
Dorian	Raised 6th
Aeolian	Natural 6th
Phrygian	Lowered 2nd
Locrian	Lowered 5th

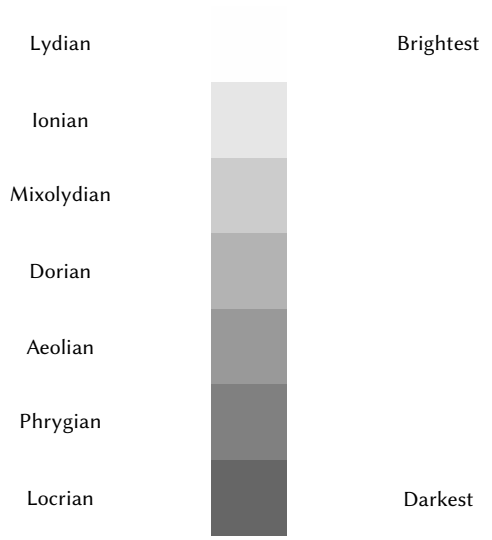

Figure 1: The seven modes ordered in terms of brightness [27].

check. We then loop through every chord in the piece of music and count each chord whose notes are all contained within each mode's scale. At the end of this process, we then select the mode for which the most chords fit within. In the case that there the chords fit within two or more scales equally, we then check the frequency rate of the appearance of a *character note* of each scale within the chords. This note can be considered to be the signature note of a scale, and the one that most represents that scale (see Table 2) [28]. For instance, the character note of the Lydian mode is the raised fourth degree.

We again argue that this mode prediction would be potentially very useful to a game designer or composer, as similarly to NRO shift, being able to target a variety of moods would be very useful for many games and game genres. While some games may not change their intended mood much at all during play, especially traditional arcade titles, these are in the minority in contemporary video gaming. The composer or game designer would be able to pick a mode and NRO shift that most represent the in-game situation or the situation they want the player to perceive. For example, in the song At Doom's Gate [29] from the game Doom [30] the main guitar riff is in Locrian, giving a tense, dark mood to the level, whereas the soundtrack frequently switches to brighter moods for more triumphant moments.

4.1. Genetic Operators

In this system, we use two genetic operators to produce child solutions from pairs of parent tracks, namely a mutation operator and a crossover operator both designed to work on a sequence of CNROs. This parent pair are selected at random from the MAP-Elites archive archive population as in conventional MAP-Elites [15].

The mutation operator works by replacing a CNRO in a sequence with a new generated CNRO. This CNRO is generated by first stochastically selecting a CNRO length with a range between 1 and the max length for a CNRO which is specified for the MAP-Elites run as a whole. We then generate a sequence of NROs by selecting NROs at random until we have enough to produce a CNRO of the specified length. This new CNRO then replaces the selected CNRO in the original sequence. This process has a chance of being applied to every CNRO in a sequence, so a mutation rate of X% means that every CNRO in a selected sequence has an X% chance of being replaced with a newly generated one.

The crossover operator used is a one-point crossover. First, we generate a random crossover point for a selected pair of parents which can be anywhere in the range 1 to the fixed track length minus 1. Two child CNRO sequences are then generated by taking the portion of parent 1s CNRO sequence up until the crossover point and concatenating it with the portion of parent 2s following the crossover point, and vice versa for child 2.

4.2. Duplicate Chord Fitness Adjustment

An issue with our method for generating CNROs stochastically is that there is a chance that when a series of NROs are applied to a chord it can produce a loop that produces the same chord as the initial chord. This can be especially problematic in conjunction with the NRT shift metric. As discussed, the goal for this phenotypic metric is as a heuristic for how discomforting the track might be to a listener based on the magnitude of the shifts between sequential chords. Due to this looping issue, however, it is possible to have long NRO sequences which would cause higher scores on the NRT shift metric, while producing duplicate chords which do not produce the desired discomfort.

To address this issue we opted to check each newly generated track for the existence of duplicate chords in

sequence. Where a duplicate chord is found we force the fitness of the track to be 0. This means the track can still be stored in the MAP Elites grid, but it will always be discarded when a sufficient number of tracks without duplicates are generated within the same cell location. While this approach is commonly used to handle infeasible individuals in genetic search and it undoubtedly achieves the goal of avoiding them being present in the final generated track set, it has also long been argued against as an approach due to the loss of genetic information involved in effectively discarding these individuals which may have made for useful parents [31]. As a result, we consider it to be an non-ideal solution to the problem and something we will look to improve in future iterations of this system.

5. Experimental Configuration

To conduct our initial experiments to explore this system and its capabilities we first needed to decide on how the system would be parameterised. Where possible we based these decisions on quantitative data from pilot experiments to indicate parameterisations that appeared to lead to the best system performance, but where this was not possible a more arbitrary decision was made.

5.0.1. Musical Parameterisation

In terms of the parameterisation of the music tracks themselves, each track starts with a tri-chord in the 4th octave. This tri-chord is randomly selected for each track when generating the starting population, and then inherited from one of the two parents for tracks generated during a run. Each track is composed of 20 CNROs giving a total of 21 chords including the initial chord. The maximum CNRO length was capped at 5. Decisions on track and max CNRO length were made arbitrarily, and it is here where we expect that designers and composers using future iterations of this system would have the most control. However, it is more than possible there is a performance impact induced by these choices, something we aim to examine in future work.

5.0.2. MAP Elites and Genetic Operator Parameters

The most important performance-affecting parameters are the mutation and crossover rates, as well as the number of elites that are stored per cell in the MAP Elites grid. Initial experiments using a grid search suggested that a mutation rate of 0.2, a crossover rate of 0.5, and storing a maximum of 3 elites per cell led to the best overall performance. We note that in future iterations of this system that are designed to meet the needs of game designers and composers, we may have to consider not just the performance of the system when it comes to the number of elites stored, but also the impact on system usability. One can imagine circumstances in which having a large range of similar compositions with similar traits would be useful to present a user with a sufficient choice, but also circumstances in which a choice of any kind could be redundant. However, for these initial experiments, this decision was made purely based on system performance.

For the dimensions of the map, the predicted mode dimension was split into 7 increments, one for each possible mode, whereas the NRT Shift dimension was split into 10 increments evenly divided between 1 at the bottom end of the theoretical range if every CNRO in a track was of the minimum length and 5 at the top of the range.

5.1. Performance Measures

To evaluate the performance of a generative run we calculate two features: Average Grid Fitness and Coverage.

To calculate the Average Grid Fitness we first sum up the fitness scores of the fittest tracks found in each grid cell, with empty cells receiving a fitness of 0. This number is then divided by the total number of cells. To calculate the Coverage we count the number of cells which have any solution at all, and divide this count by the total number of cells. Coverage is intended to measure how effectively we are exploring the overall space, whereas Average Grid Fitness looks at performance more generally and requires both exploration but also the finding of high-performing tracks. Coverage is commonly calculated in MAP Elites implementations including in the paper which introduced the algorithm [15], and Average Grid Fitness is similar to the Reliability score which appears in that paper and others, except that we do not have an alternative algorithm to compare to.

5.1.1. Run Length

In terms of experiment length, we let the process run for 5,000 loops. This cutoff point was selected after initial experimentation suggested that we were unlikely to see Average Grid Fitness gains after this point. Each loop involved the selection and generation of two tracks to support the use of the crossover operator. Therefore 10,000 tracks were generated in total. To seed the initial map a starting population of size 500 was generated for each run. This starting population was generated through purely stochastic generation and concatenation of CNROs, as well as the selection of a random 4th-octave start chord for each starting track.

5.1.2. Computational Resources Used

All experiments were run on a Dell laptop with an i5-10310U CPU with 16.0 GB of RAM. With the above con-

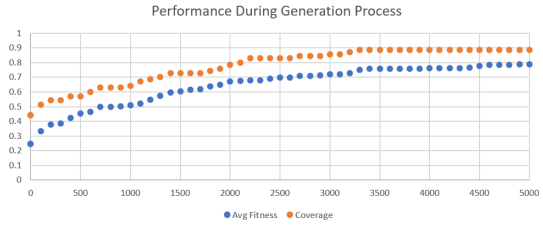

Figure 2: Figure showing the performance of the generative process in terms of average grid fitness and coverage as loop count increases. Measurements recorded and visualised for every 100th loop

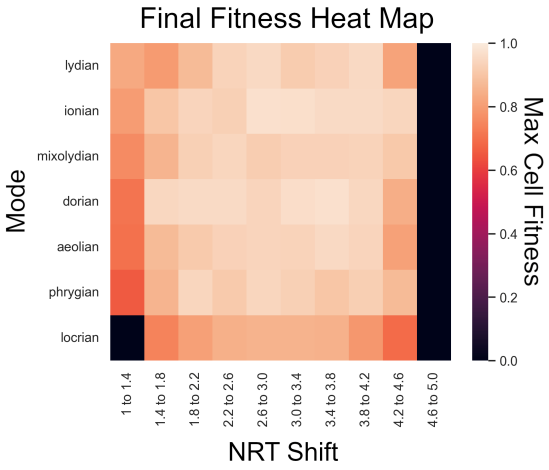

Figure 3: Figure showing the final MAP Elites archive in the form of a heat map. The color of each cell is determined by the fittest track found for the cell

figuration on this hardware, a full generative run took 1 hour and 32 minutes.

The system including all of its code, data for the results, as well as the generated chord progressions from this paper's experiment in both textual and .midi form is available at `github.com/KrellFace/harmony-mapper/`.

6. Results and Discussion

Here we discuss the overall performance of the system including its strengths and weaknesses of it which are highlighted by the results of our initial experiments.

In terms of systems performance, we are cautiously optimistic. It is able to produce sets of tracks with significant amounts of diversity present in terms of our two metrics. The relatively high final average fitness score of 0.786 indicates that the system is able to produce series of chords with a high level of tonal consistency.

The system has the advantage of being fairly computationally light and it was able to generate and evaluate over 10 thousand chord sequences on a modestly powered laptop in only 92 minutes. However, the process would be too slow to be used in a generative system which required quick iteration and generation such as a mixed-initiative generative system with a human in the loop (see [32] for an overview of mixed-initiative content generation systems). Additionally, though we were able to achieve a robust average fitness during our experimental runs, Figure 6 suggests that a longer run time would be required to achieve the maximum possible fitness as average fitness and coverage were still increasing at the termination point.

As is evident in Figure 3 there were combinations of metric values which the system struggled to find tracks for. At the extreme low and high ends of the possible NRT Shift values the system visibly struggles to further populate the MAP Elites grid, especially in the 4.6 to 5 range in which it found 0 tracks. While one would expect that the outlying locations of the possibility space would naturally be harder to reach, we expect that this could also be the result of the high mutation rate used and the nature of our mutation operator. With the mutation rate used of 20% for each of a track's 20 CNROs, and with the 80% chance that a generated CNRO will have less than 5 NROs there is significant pressure pulling newly generated tracks back to the centre of the grid.

By inspecting the rules of our system, we can see that it cannot produce chord progressions that are fully in the Locrian mode. This is because it is impossible to produce its signature half-diminished interval tonic using only conventional NRO shifts. The best tracks found in this part of the archive are comprised of chord sequences that do not fit comfortably into any key when analysed with Krumhansl-Schmuckler's Algorithm but happen to contain chords that conform best to the Locrian scale for the best-fitting key. However, this does not mean they are actually in the Locrian mode. This highlights the need for a fitness cutoff before any generated chord sequences are actually selected for use, as the lower the fitness of a track, it both conforms less to a key, but also less likely to be in the correct mode. The inability of the system to produce sequences in the Locrian mode is a limitation we aim to address on as discussed later in Section 7. This limitation could be improved by using the extended version of NRT by Colton and Cardinale [33] which is able to produce multiple chord types, including diminished.

The music produced by the system consists of chords played on a piano. Each chord has an equal duration and the instrumentation does not change throughout the chord progression. This is a simplified version of the system as proof of concept. In Section 7 we discuss the

future addition of melody components and improving the complexity of the music generated by HarmonyMapper.

In terms of the metrics themselves, future listening studies are required to explore whether or not we are successfully producing emotionally diverse chord sequences. However, our initial impressions are that the NRT shift metric did not perform as expected. We were aware that there were limitations to it as a heuristic due to the way NROs operate. To take an example from one of the reviews of this paper, R and SSLL are equivalent in terms of the chord change they produce but they differ massively in length. We were aware of this limitation but hoped that larger shifts would still tend to indicate tracks with chord transitions which would be perceived as more jarring. To the authors' ears, however, this was not always the case. As a result, improving on this distance metric is going to be a focus of our future work.

7. Future Work

The most important area in which to improve and expand HarmonyMapper is the musical complexity it is able to generate. Specifically, we aim to extend the system with the generation of melodies, bass lines and percussion to pair with the diverse harmonies that are currently generated. One of the potential end goals for this system is for non-musicians to be able to use HarmonyMapper musical sequences in their games without the need for any alteration. In terms of how these additions are generated, we could take inspiration from Cardinale and Colton's GENRT and generate them stochastically based on a set of specifications [13]. Naturally, these musical additions are also another facet of the music in which we can try and promote interesting diversity. For instance, we plan to produce deliberate diversity during the melody generation process, by using similar heuristics to the ones introduced in this paper to make the melody more or less voice-led.

We also plan to explore using different evaluation metrics for our system. One substantial change we plan to make is to use a different distance metric, such as Lerdhal and Krumhansl's model for tonal tension [34]. Lerdahl suggests that a multidimensional representation of a pitch space allows for a more nuanced understanding of tonal relationships and can capture the hierarchical organization of tonal music [35]. This could allow us to understand more in-depth the link between NRT chord progressions, modes, and moods. This would potentially also allow us to avoid the use of our duplicate chord fitness adjustment and the subsequent loss of genetic information.

Another potential extension to HarmonyMapper which we are keen to explore is the introduction of the concept of leitmotifs. Leitmotifs are repeated musical phrases that often represent a concept or character in soundtracks. Firstly we aim to explore the generation of diverse harmonies that all include the same leitmotif, mirroring how they are used in traditional composition to evoke different moods while still referencing the same character or concept. More excitingly perhaps, we aim to investigate whether we can use HarmonyMapper to produce tracks that are blends of two leitmotifs by developing a heuristic for evaluating how much a composition confirms to either or both of two specified leitmotifs. This could allow us to generate musical sequences that both evoke different moods but also evoke two contrasting characters or concepts to a lesser or greater degree, similar to how composers write music to unite two or more leitmotifs.

8. Conclusion

In this paper, we introduced HarmonyMapper, a novel system for generating diverse chord sequences using MAP Elites, and have argued that a Quality-Diversity approach for music generation with music theory-inspired metrics is a valuable basis for game music generation systems. While the current output is limited to simple chord progressions, their intentional diversity in terms of the mood they convey to a listener makes them valuable as a musical foundation for generating more complex compositions. We look forward to expanding on this approach to develop a controllable system for generating diverse and appropriate music for games.

Acknowledgements

We thank the reviewers for providing insightful feedback. This work has been funded by UKRI and EPSRC as part of the "UKRI Centre for Doctoral Training in Artificial Intelligence and Music", under grant *EP/S022694/1*, as well as the EPSRC Centre for Doctoral Training in Intelligent Games & Games Intelligence (IGGI) [EP/S022325/1].

References

[1] LookingGlass Technologies, System Shock, 1994.
[2] Looking Glass Studios, Thief: The Dark Project, 1998.
[3] Techland, Dying Light 2, 2022.
[4] C. Plut, P. Pasquier, Generative music in video games: State of the art, challenges, and prospects, Entertainment Computing 33 (2020) 100337. URL: https://linkinghub.elsevier.com/retrieve/pii/S1875952119300795. doi:10.1016/j.entcom.2019.100337.

[5] D. Lewin, A formal theory of generalized tonal functions, Journal of Music Theory 26 (1982) 23–60.

[6] G. Bernardes, D. Cocharro, C. Guedes, M. Davies, Harmony generation driven by a perceptually motivated tonal interval space, Computers in Entertainment 14(2) (2016).

[7] K. Monteith, T. Martinez, D. Ventura, Automatic generation of music for inducing emotive response, in: Proceedings of the International Conference on Computational Creativity, 2010.

[8] K. Monteith, V. Francisco, T. Martinez, P. Gervás, D. Ventura, Automatic generation of emotionally-targeted soundtracks, in: Proceedings of the International Conference on Computational Creativity, 2011.

[9] J. Robertson, A. de Quincey, T. Stapleford, G. Wiggins, Real-time music generation for a virtual environment, in: PROCEEDINGS OF ECAI-98 WORKSHOP ON AI/ALIFE AND ENTERTAINMENT, 1998.

[10] S. A., Theory of Harmony, University of California Press, 1983.

[11] L. Duarte, A. E., Algorithmic interactive music generation in videogames: A modular design for adaptive automatic music scoring, SoundEffects - An Interdisciplinary Journal of Sound and Sound Experience 9 (2020) 38–59. doi:10.7146/se.v9i1.118245.

[12] F. Lehman, Film music and Neo-Riemannian Theory, Oxford Handbook (2014).

[13] S. Cardinale, S. Colton, Neo-riemannian theory for generative film and videogame music, in: Proceedings of the Int. Conference on Computational Creativity, 2022.

[14] V. Vassiliades, K. Chatzilygeroudis, J.-B. Mouret, Using Centroidal Voronoi Tessellations to Scale Up the Multi-dimensional Archive of Phenotypic Elites Algorithm (2016). URL: https://arxiv.org/abs/1610.05729. doi:10.48550/ARXIV.1610.05729, publisher: arXiv Version Number: 2.

[15] J.-B. Mouret, J. Clune, Illuminating search spaces by mapping elites, arXiv:1504.04909 [cs, q-bio] (2015). URL: http://arxiv.org/abs/1504.04909, arXiv: 1504.04909.

[16] J. Nordmoen, F. Veenstra, K. O. Ellefsen, K. Glette, MAP-Elites Enables Powerful Stepping Stones and Diversity for Modular Robotics, Frontiers in Robotics and AI 8 (2021) 639173. URL: https://www.frontiersin.org/articles/10.3389/frobt.2021.639173/full. doi:10.3389/frobt.2021.639173.

[17] J. McCormack, C. Cruz Gambardella, Quality-diversity for aesthetic evolution, in: T. Martins, N. Rodríguez-Fernández, S. M. Rebelo (Eds.), Artificial Intelligence in Music, Sound, Art and Design, Springer International Publishing, Cham, 2022, pp. 369–384.

[18] L. Cazenille, N. Bredeche, N. Aubert-Kato, Exploring Self-Assembling Behaviors in a Swarm of Bio-micro-robots using Surrogate-Assisted MAP-Elites, in: 2019 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, Xiamen, China, 2019, pp. 238–246. URL: https://ieeexplore.ieee.org/document/9003047/. doi:10.1109/SSCI44817.2019.9003047.

[19] A. Khalifa, S. Lee, A. Nealen, J. Togelius, Talakat: bullet hell generation through constrained map-elites, in: Proceedings of the Genetic and Evolutionary Computation Conference, ACM, Kyoto Japan, 2018, pp. 1047–1054. URL: https://dl.acm.org/doi/10.1145/3205455.3205470. doi:10.1145/3205455.3205470.

[20] A. Khalifa, M. C. Green, G. Barros, J. Togelius, Intentional Computational Level Design, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 796–803. URL: https://doi.org/10.1145/3321707.3321849. doi:10.1145/3321707.3321849, event-place: Prague, Czech Republic.

[21] M. C. Fontaine, R. Liu, A. Khalifa, J. Modi, J. Togelius, A. K. Hoover, S. Nikolaidis, Illuminating Mario Scenes in the Latent Space of a Generative Adversarial Network, in: 35th AAAI Conference on Artificial Intelligence, AAAI 2021, 35th AAAI Conference on Artificial Intelligence, AAAI 2021, Association for the Advancement of Artificial Intelligence, 2021, pp. 5922–5930.

[22] J. McCormack, C. C. Gambardella, S. J. Krol, Creative discovery using qd search, arXiv preprint arXiv:2305.04462 (2023).

[23] B. Banar, S. Colton, A Quality-Diversity-Based Evaluation Straetgy for Symbolic Music Generation, 2022.

[24] C. L. Krumhansl, M. A. Schmuckler, Key-finding in music: An algorithm based on pattern matching to tonal hierarchies, in: 19th Annual Meeting of the Society of Mathematical Psychology, Cambridge, MA, 1986.

[25] M. S. Cuthbert, C. Ariza, Music21: A toolkit for computer-aided musicology and symbolic music data., in: J. S. Downie, R. C. Veltkamp (Eds.), ISMIR, International Society for Music Information Retrieval, 2010, pp. 637–642. URL: http://dblp.uni-trier.de/db/conf/ismir/ismir2010.html#CuthbertA10.

[26] C. L. Krumhansl, The psychological representation of musical pitch in a tonal context, Cognitive Psychology 11 (1979) 346–374. URL: https://www.sciencedirect.com/science/

article/pii/0010028579900161. doi:https://doi.org/10.1016/0010-0285(79)90016-1.
[27] V. Persichetti, Twentieth century harmony: creative aspects and practice, W. W. Norton & Company, 1978.
[28] R. Rawlins, N. E. Bahha, Jazzology: the encyclopedia of jazz theory for all musicians, Hal Leonard Corporation, 2005.
[29] B. Prince, At doom's gate, 2012.
[30] id Software, Doom, 1993.
[31] Z. Michalewicz, Do not kill unfeasible individuals, Proceedings of the Fourth Intelligent Information Systems Workshop (IIS'95) (1995).
[32] G. Lai, F. F. Leymarie, W. Latham, On mixed-initiative content creation for video games, IEEE Transactions on Games 14 (2022) 543–557.
[33] S. Colton, S. Cardinale, Extending generative neo-riemannian theory for event-based soundtrack production, in: International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar), Springer, 2023, pp. 67–83.
[34] F. Lerdahl, C. L. Krumhansl, Modeling Tonal Tension, Music Perception 24 (2007) 329–366. doi:10.1525/mp.2007.24.4.329.
[35] F. Lerdahl, Tonal pitch space, Music Perception: An Interdisciplinary Journal 5 (1988) 315–349. URL: http://www.jstor.org/stable/40285402.