# 2 - Literature Review

## 2.1 - Generative Algorithms - Overview

WHY ARE GENERATIVE ALGORITHMS NEEDED

Generative algorithms appear in many forms, but at their core, they work by pseudo-randomly generating the next step, based on the current step. How the algorithm generates this next step can be based on precomposed music, which the program analyses and works out the musical structure, it can also use a rule-based system, created from music theory, to generate this structure or a combination of these in a hybrid system. In the past decades, there has been a number of variants that have been utilised to produce music, some of which are more appropriate for in-game music generation than others. This section will outline some examples of these and their uses.

### 2.1.1 - Neural Networks

Neural networks were originally inspired by biological networks (Bishop, 1996), mainly the human and the way in which it can processes information with a series of interconnected processing units and can also modify the connection between these units, thus allowing it to learn (van Gerven, 2017). Neural networks have been long been used for pattern recognition, for example, given pictures of letters it would be able to learn to identify, or produce, pictures it was not trained on.

This suggest that neural networks would be an appropriate algorithm for algorithmic music production. For example, Colombo et al. (2017) developed a system which produced music in the style of Irish and Klezmer folk, as this is the style they chose to train it on. Or Johnson (2017), who intentionally didn’t give information about the musical domain of the pieces used so the trained network produced more ‘rounded’ music. However, in both of these experiments their chosen analysis method was to look at the distribution of notes compare to the training music, while they found that the produced music was mathematically similar they do not get human experts to evaluate how ‘good’ the music sounds, a common trend in many papers ADD MORE HERE. This is not always the case, for example, Prisco et al. (2017) utilises an algorithmic evaluation and a group of musical expert, whom all had more than ten years in the music field. Who found the music produced was of high quality and stylistically coherent, the expert were also able to point out flaws in the rhythmic elements of the music, giving Prisco et al. avenues for future work. However, as many of these paper’s goals are not focused on producing the music in real-time they make no mention of the time it takes for their systems to be trained, produce the music, or their computational cost. This would make their use in a video-game scenario questionable.

[ANY MORE REASONS WHY I DIDN’T USE THEM?]

### 2.1.2 - Genetic Algorithms

Like neural networks, genetic algorithms are also inspired by nature, however these are based on evolution. This works by generating several options for the output, the best of which is chosen and then generating several more option based on this best option, this can then be repeated until a satisfactory outcome is reached.

An example of this is a system created by Ostermann, Vatolkin, and Rudolph (2017) which can generate a drum beat, in real-time, to go allow with a live band. This generated a MIDI file of what the players were playing and used this to choose the drum beat which fit best with the current musical solo. The musicians reported that this system was mostly successful when improvising along with a band, however this would then require some sort of pre-generated music to be present before the drums tracks could be altered, whether this is precomposed or procedurally generated. The purpose of this system was as a practice aid for the musicians, rather than create finished pieces.

Another way genetic algorithms have been used is to generate a fitness functions, which can then numerically evaluate musical pieces, this technique was utilised by Loughran and O’Neill (2017). This would then allow for the music and the fitness functions to ‘evolve’ together. However the authors state that the music produces was not overly impressive, coupled with the fact that this generation techniques requires many melodies to be generated at a time, this would drastically increase the amount of processing power given over to the music production, which could have an impact on the performance of a game as a whole.

### 2.1.3 - Markov Chains

Markov chains are used for modelling a finite number of states and the probability of transitioning between them (Snodgrass and Ontañón, 2014). For example, if a small musical melody is ‘D, D, F#, D, D, E, ’, if the current note being generated is ‘D’, then the next note will have a 50% chance to be ‘D’, a 25% to be ‘F#’ and the same chance for ‘E’. This type of Markov chain is defined as 1st order, as each new generated state is based only on one step beforehand. As the order increases the generated sequences have an increased level of similarity to the training data making the generated music sound less random, although this can also mean that the only possible sequence that can be generated is the trained data, which is not ideal when the purpose of the application is to generate new music.

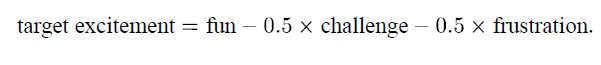
Conversely to be too similar to the training data, a higher order chain can also produce states that were not even present in the training data, if this happens the application would require a method of overcoming this problem, such as stepping back through the data and then generating the failed sequence again (Snodgrass and Ontañón, 2013). Higher order chains may not even be worth the extra computing power that they require, as Schulze and van der Merwe (2011) found that when presented with a human composed piece, music generated using a low Markov, and a one generated with a high order, the high order music was preferred the least. In regards to the randomness of the 1st order chains, Collins et al. (2011) found that generated music, based off of Chopin’s Mazurkas, was difficult to discern from the real thing.

A recent example Markov generated music was present in the game ‘Rise of the Tomb Raider’ (Crystal Dynamics, 2015) utilising the ‘dynamic percussion system’ (Intelligent Music System), this was originally developed by Brown (2012) for his PhD and utilises a combination of 1st order Markov chains, Genetic algorithms, and music theory to produce music at run-time. Following the other research on the topic, Brown found that the genetic algorithms took too long to generate music at run time, so this was done as the application loaded. While the original application was created to produce scored music for piano, it was utilised in ‘Rise of the Tomb Raider’ exclusively for the dynamic percussion, which would react to the various states that the player could find themselves, from being completely hidden and the enemy not being aware of the player, to being engaged in combat. This dynamic drums was then set against precomposed music (Lamperski and Tahouri, 2016).

Drums are a very common subject of PGM as they have a much lower level of variability in the number of note types that can be played (a standard drum kit has about around 10 noises, while a standard piano has around 80), the length of drum also matters less as a drummer has much less control over the length of a note than a pianist. Drum tracks are generally much more repetitive.

### 2.1.4 – Game Play as the Composer

Regardless of the method chosen to generate music, there needs to be some way that a game actively control this process, an example of this is experience driven procedural content generation (EDPCG). This is where the many different events in a game are used to calculate an ‘excitement’ level, which is then used to control how the music is generated. Plans and Morelli (2012) used this for an infinite Mario Bros level generator, where they associated various actions in the game with frustration (time standing still and dying), challenge (alive time, time ducked), and fun (running time, coins collected, and monsters killed). These three variables were then combined using the following formula:



This excitement value was then used to control aspects of the music generation, such as the tempo, scale, sparseness, and novelty. While the authors found that the use of this system increased player’s enjoyment, they admit that they did not get enough testers to come to a significant conclusions.

A similar experiment by Chan et al (2017) looked into how EDPCG can be used to inform the player on how well a player is doing in a game. The game the created to test this consisted of the player trying to find specific objects while avoiding hidden enemies. For this they used significantly less in game events to control the music, however this was to make the cause of the changes in the music more apparent to the player. They used; Tension (enemy proximity, low number of items collected, and low player health) and Progress (collectable proximity and high number of items collected). While this experiment also suffered from a low number of tester, they results found suggest that this method of procedural music control does indeed inform the player in how they are playing a game.

Another way that gameplay can affect the music generation is shown in an experiment by Mauceri and Majercik (2017). In this they created a music system based up a swarm algorithm reacting with a live performer playing a traditional musical instrument. The swarm would react to aspects of the performance such as; the amplitude would regulate the number of swarm members, the pitch would affect the general location of the swarm, and the noisiness would affect the cohesion. The swarm would in turn interact with a granular synthesiser to produce the procedural music, such as the difference in swarm member locations would determine the length of notes produced, and the mean Z location would affect the amplitude. While this method may not be appropriate in some game scenarios it would be interesting to have a similar system for a game’s antagonists or to be affecting a game’s wildlife.

## 2.2 - Music Theory - Overview

This sub-chapter will discuss the various musical rules that can be utilised in narrative elements of video games, and the ways in which music can be used to directly affect the player’s experience of a game.

### 2.2.X - Brightness

When composing a piece of music with a particular emotional theme, a common technique is to use a major scale if the song is to have a positive feel and a minor scale if it is to be negative. This is actually an example of the concept of musical ‘Brightness’, in regards to major and minor scales they are actually part of a larger group of seven modes, in decreases brightness these are;

* Lydian
* Ionian (or major)
* Mixolydian
* Dorian
* Aeolian (or minor)
* Phyrigian
* Locrian

These mode can then be used

Musical brightness can also be changed by other factors

* Increase in pitch
* Tempo
* Rhythmic density

(Jayden Chan *et al.,* Oct 2017)

### 2.2.X - Motific Structure

Period / Sentence