Using an evolutionary algorithm to create human-like techno music

Bachelor Thesis - 7,5 ECTS
15-1-2021

Under supervision of:
Gerard Vreeswijk &
Lasha Abzianidze

Eli Stolwijk
Student number: 6000738

Universiteit Utrecht

Bachelor Artificial Intelligence
Faculty of Humanities
Utrecht University
Netherlands

Contents

1 Introduction

It goes without saying that music is an important part of everyday life. Music can make us feel certain emotions, highlight special occasions or evoke special memories. Listening to, playing and sharing certain types of music can become a big part of someones identity. Besides playing a big role in the personal space of people, music also plays a big part in our social lives. Every song we play in company or share through some other type of medium is a way to express ourselves to others (O'Hara and Brown, 2006).

Artificial intelligence is a field of computer research that aims to use intelligent computer systems to solve all kinds of problems. The range of problems the A.I. field tackles is very wide. This range, for example, includes self-driving cars, job scheduling, speech recognition, classification problems, and creating human like computer entities. Can such a seemingly limitless field of computer science be of use in the seemingly limitless creative space that is music?

1.1 Computational Creativity

An up and coming field in computer science is the field of computational creativity. The last 20 years, computational creativity has grown a lot however not without issues. A big issue that computational creativity faces is the definition of creativity. After all, what would be considered creative and what not? Artificial intelligence has a very similar problem: what would be considered intelligent and what not? Something that would have been considered intelligent 100 years ago would not be today. Because of the similarities, Toivonen and Gross (2015) proposed that computational creativity can be characterized in a manner parallel to artificial intelligence: Where artificial intelligence studies how to perform tasks which would be deemed intelligent if performed by a human, computational creativity studies performances which would be deemed creative if performed by a human. It would therefore not be a surprise that the computational creativity field is dominated by A.I. heavy research. Multiple A.I. systems have been proposed and used in the computational creativity field but for computational music creation most of the research has been done on machine learning.

1.2 Problems of Machine Learning

For an example of machine learning used for music generation I would like to refer to the work of Malik and Ek (2017). In short, a machine learning algorithm analyses a large data set (in this case a data set consisting of existing music) and then uses that data to learn how to play similar kinds of music (Huang and Raymond, 2016). The reason machine learning is so suitable for music generation is because of its generality (Briot and Pachet, 2018). Because of its generality it can automatically adapt itself to every style or genre from an arbitrary corpus. More benefits of machine learning systems are the fact that they can handle applications that are way more complex than traditional methods can and the way they learn from data makes them very adept to widely varying inputs (Fiebrink and Caramiaux, 2016).

The problem with machine learning however is that such a system needs a very large training set in order to produce anything meaningful. This very large training set has to consist of existing real world data, which in the case of computational creativity has to consist of the works created by human artists. Humans who copy works that were originally made by other artists are generally frowned upon, so why should a computer be applauded for it? As reported by Forbes, in 2020 a research group used machine learning to learn the voice of a popular rapper named Jay-Z. With this algorithm they made a little video in which the algorithm spoke some lyrics the researchers had programmed with the voice of Jay-Z. The rapper then sued the researchers for using his voice for the algorithm without having his permission. (Hochberg, 2020). The rapper actually lost the lawsuit, but this example makes it clear that using artist material to use as training data might not be the most ethical way to go. An approach that circumvents potentially stealing other peoples work and intuitively feels more like a creative process instead of a process of mimicking, is the field of evolutionary music.

1.3 Evolutionary Music

Evolutionary music uses evolutionary algorithms rather than machine learning to generate music pieces (Dostál, 2013). An evolutionary algorithm is an algorithm that uses the Darwinian evolutionary principles to efficiently search through large search spaces

(Goldberg and Holland, 1988). There have already been multiple successful projects that have generated music using evolutionary algorithms. These projects mostly produced classical and jazz music. The most notable project is the Lamus project. Lamus is a project created by a research group at the University of Màlaga. Lamus uses an evolutionary algorithm to generate pieces of classical sheet music (Diaz-Jerez, 2011). One of the main performances Lamus created, named "Opus one", was claimed to be "the first major work composed by a computer that was performed by a full orchestra" by the New Scientist (NewScientist, 2012).

In this paper it will be investigate whether computational creativity can maybe thrive better in a musical field that intuitively would fit it better: electronic music, and more specific techno. An issue with the Lamus algorithm is the fact that it can not play the music it generates. Lamus can generate the sheet music but it lacks the ability to play the sheet music like the instrument players of an orchestra can. Techno music uses computational methods like synthesizers to create its sounds which opens up the possibility of an evolutionary algorithm that can both play and generate its own techno music.

Although the origins of electronic music lie in the late 19th century, it really started growing in the 1960s as synthesizers became more accessible to the average musician. In the 1980s multiple subgenres of electronic music started to form, including the techno subgenre. The techno genre is characterized by a four on the floor beat (Julien and Levaux, 2018). This is a steady beat that occurs in 4/4 time or in other words, the bass drum (also know as the kick) is hit on every beat. The beats per minute (BPM) of a techno song lies between 120 and 150 and artists use electronical instruments like synthesizers, digital audio workstations or a combination of the two to create it. Because techno music is made on a computer, an algorithm that generates techno music does not need an orchestra to perform the music it creates as is the case for projects like Lamus. It can create and play its generated music all on its own.

1.4 Research Question

This paper will research the feasibility of techno music generation with an evolutionary algorithm to lay a foundation for potentially

more elaborate further research. In order to do so the following research question will be investigated: **Can an evolutionary algorithm called the ETM algorithm (Evolutionary Techno Machine) generate an 8-beat loop that when looped creates a small piece of techno music of about 15 seconds that is indistinguishable from man made techno music?**. In order to answer this question the research is split into two parts. The first part consists of generating a small piece of music with the ETM algorithm. The second part consists of performing an alternative form of the Turing test, a Musical Turing test, to determine whether a group of participants can distinguish the generated piece of music from music made by artists.

2 Design

This chapter will cover the design of the algorithm and all its components. The ETM algorithm can be split into two sub-algorithms: the algorithm that is responsible for the evolutionary processes and the algorithm that is responsible for decoding the chromosomes into sound. From now on the term chromosome will be used to describe the internal representation of a piece of music. In other words, a chromosome in an evolutionary algorithm is a string of characters that represents one specific combination of properties out of all possible combinations, just like the chromosomes of DNA do in biology. How the chromosomes used in the ETM algorithm are build up and how they are decoded into sound is explained further in section 2.3 but for now it suffices to know that the chromosome is the internal representation for one 8-beat loop that the algorithm uses to execute the evolutionary processes on.

Both the sub-algorithms were programmed in NetLogo and were linked to each other by an intuitive interface that is optimized for efficient evaluation. NetLogo is a programming language that was created for educational purposes. It involves agents called turtles that are able to manipulate their environment. Despite being for educational purposes, NetLogo is a Turing complete language that has an easy to use build in sound library. The library consists of around 200 sound samples of many different instruments that can be individually manipulated in duration, pitch and velocity. This library and the way the sounds can easily be manipulated without

additional sound manipulation algorithms makes NetLogo an ideal programming language for decoding the chromosomes. The decision to also make the evolutionary part of the algorithm in NetLogo was made due to the fact that NetLogo also provides an easy to use interface that can link the two algorithms seamlessly. From this sound library the algorithm will choose two hat instruments (Hat1 Hat2), two percussion instruments (Percussion1 Percussion2) and one melody instrument (Melody). When these five instruments are played and at what pitch is determined by the chromosomes present in the population.

Alongside these 5 instruments there will be two fixed instruments playing at fixed intervals: a kick every one beat and a clap every two beats. As described in the first chapter, the kick every beat is a requirement of the techno genre. Because of the inability to use custom sound files in the algorithm, the bass drum instrument from NetLogo is used as a dummy to emulate a real kick sound during the evaluation process. A good kick sound consists of a short relatively high pitch slap sound and a longer much lower pitched bass sound. Unfortunately, there are no sounds available in the sound library that posses these properties. The reason the kick is so important is that it does not only carry the rhythm, it also sets the overall mood and tone of the song. The kick is a defining feature of every techno song which is why it is essential to use a good kick sound in this experiment. The base drum instrument that the NetLogo sound library provides is not a suitable sound for a techno kick but it is the sound that comes closest. For this reason it is used to emulate a kick sound during the evaluation process. When the final winning chromosome is chosen the base drum sound is replaced with a custom sound file of a custom made basic kick sound. The clap every other beat is not a requirement of the genre but is a very common addition which is why it is chosen to be a fixed instrument as well. As described in chapter 1 the range of beats per minute for the techno genre lies between 120 and 150. For this experiment a beats per minute of 130 is used because it lies within the range and it is considered as an average beats per minute for the techno genre.

2.1 An Evolutionary Algorithms

For this research a fairly standard Evolutionary algorithm is used. As mentioned before, an evolutionary algorithm is an algorithm that uses Darwinian principles to achieve its results. Darwinian principles are the principles of biological evolution as described by Darwin (1859). This set of principles consists of how the fitness of individuals determines which properties are preserved in a population, how new individuals of that population are generated and how new properties can be introduced into a population through mutation.

The fitness value of an individual is a value determined by some fitness function. In biology the fitness function is an extremely complex function between the properties the individual possess and the properties its environment possesses. In evolutionary computing this fitness function can be a function of arbitrary complexity. The general idea is that an individual with a high fitness thrives better in its environment than one with low fitness. Therefore, an individual with high fitness has a better chance of producing offspring (generating a new individual) and thus has a higher chance of passing its properties along to the next generation.

Because individuals with beneficial properties (high fitness) are more likely to generate more new individuals than individuals that have harmful properties, the idea is that every generation will consist of individuals with more beneficial properties then the last generation and will therefore be better adept to the environment than the previous one. In evolutionary computing this is an important concept because a population will become better every generation with respect to some predetermined fitness function. In biology, individuals can have multiple offspring and can live on after reproducing. Evolutionary computing does this different and works per generation. A generation works as follows: parents with beneficial fitness are chosen from the population, then the whole population gets deleted, then the chosen parents create their offspring to fill the population back up and with this new population a new generation has been generated and the cycle is closed. This cycle can be repeated an arbitrary amount or until a predetermined fitness value of one or more individuals is reached. Lastly, the mutation concept enables the possibility for new properties to enter a population. A mutation means that some, usually very small, part of the chromosome is changed so that it now decodes for a different property than

before the mutation. Whether this new property is more beneficial than the previous one is not of concern to the mutation principle since it works at random.

2.2 The ETM Algorithm

For the basics of the inner workings of evolutionary algorithms the work of Goldberg and Holland (1988) can be consulted. The basics of the ETM algorithm follow the guidelines set by Goldberg and his colleague with two alterations; the selection process and because of that also the crossover operator.

Goldberg describes multiple methods to select what chromosomes will get to produce offspring for the next generation that usually use some form of probability, tournament selection for example. A selection method like tournament selection results in a new population that consists of the winners from randomly chosen fitness comparisons. A method like tournament selection preserves a lot more variation than the method used in this research which is why the crossover operator was changed slightly in order to compensate for this. Tournament selection leaves the possibility open to have chromosomes with lower fitness to still produce offspring if they randomly get matched against chromosomes that have even lower fitness values. Lower fitness values do not mean that all their properties necessarily have to be bad, they can posses one very important property and still have a low fitness if all the rest of its properties are bad. This is why evolution generally wants to have at least some level of variation in the chromosomes of its population. Because our selection method only picks the two chromosomes with the highest fitness, a lot of variation in the population gets lost. This is why the crossover operator is changed slightly to create a little more variation than the standard crossover operator. In this research only two chromosomes are chosen to produce offspring for the next generation. In order to maintain a population size of 10, both these chromosomes are cloned five times. It is clear that this method includes a lot less variation than a method like tournament selection. To prevent the crossover operator from crossing over two clones with each other and thus resulting in no change (and thus less variation) a clone from chromosome 1 can only be crossed over with a clone of chromosome 2. Otherwise the one-point and two-

point crossover operators used in this research work as described by Goldberg. On initialisation (so at generation 0), 10 chromosomes are filled randomly. The 10 chromosomes then become available to be decoded into sound by the decoding algorithm with the click of a button. After listening to all 10 chromosomes the evaluator can select the two chromosomes he values the highest and decide to let the algorithm generate the next generation. The two chosen chromosomes are then duplicated five times and the crossover operator is applied to all the (parent 1 clone, parent 2 clone) pairs. After the crossover operator has been applied to all five pairs, every bit in every chromosome is looked at by the mutation operator and with a certain chance it randomly assigns that bit with a new random value. Note that this value can be the same as the previous assignment allowing the possibility for a bit to stay the same through a mutation.

2.3 Operators and Parameters

The mutation operator used for the experiment will be the random mutation operator as described by Goldberg. This operator uses one parameter called the mutation factor. This parameter determines the chance of mutation. A mutation factor of 0,01 means that every gene has a 0,01% chance to be mutated. Which crossover operator and what mutation factor used for the experiment will be determined by an optimisation test. Prior to the experiment the algorithm will be tested with two crossover operators and multiple mutation factors to determine which parameter combination works best. Because the actual fitness function of the experiment is too complex to efficiently perform multiple optimization tests with, these optimization tests will be conducted with a less complex fitness function: the sum of all gene values. Every parameter combination will be tested 10 times, and the average will be calculated and analysed. Every test will be conducted on the same fixed two parents, both with a starting fitness of 328 and an optimal fitness of 659. The two crossover operators that will be tested are the two most basic crossover operators described by Goldberg: the one-point crossover operator and the two-point crossover operator. The mutation factors that will be tested begin with 0,01 as suggested by Goldberg and then in increasing increments of 0,01 to 0,15. The combination

Table 1: Overview of the functions per gene of the chromosome.

Bit number	Alphabet	Description
1 - 2	0 - 1	Codes for the number between 0 - 3 that determines which instrument is used for Hat1
3 - 5	0 - 1	Codes for the number between 0 - 7 that determines which instrument is used for Hat2
6 - 8	0 - 1	Codes for the number between 0 - 7 that determines which instrument is used for Percussion1
9 - 12	0 - 1	Codes for the number between 0 - 15 that determines which instrument is used for the melody
13 - 15	0 - 1	Codes for the number between 0 - 7 that determines which instrument is used for Percussion2
16 - 23	0 - 1	Determines whether Hat1 is played for every 1/2 beat (1 = yes, 0 = no)
24 - 39	0 - 1	Determines whether Hat2 is played for every 1/4 beat (1 = yes, 0 = no)
40 - 43	0 - 1	Codes for the number between 0 - 16 that determines when Percussion2 is played every 4 beats
44 - 51	0 - 1	Determines whether Percussion2 is played every 1/2 beat (1 = yes, 0 = no)
52 - 83	0 - 1	Determines whether the melody instrument is played every 1/4 (1 = yes, 0 = no)
84 - 147	0 - 9	Determines the pitch of the corresponding melody note in pairs with the next bit number

of cross-over operator and mutation factor that performs the best will be used for the actual experiment.

2.4 Decoding the Chromosome

The chromosomes used by the ETM algorithm are composed of 147 individual numbers (genes). The first 83 bits have a binary alphabet (values of either 0 or 1). The last 64 bits have a value coded alphabet (values between 0 and 9). Value coding is a method that is used to encode more complex information into one gene compared to using a binary alphabet (Bessaou and Siarry, 2001). The reason value coding is used is because, as discussed later in this sub-chapter, each melody note must be assigned its own pitch value. This is a value between 0 and 99. Encoding numbers up to 99 would mean we need 7 bits per melody note. With 32 notes it would mean we had to triple the size of the chromosome. Value coding this part makes

it possible to reduce the number of bits coding for the melody pitch significantly from around 2/3 of the chromosome to close to $\frac{1}{4}$ making the chromosome a lot more balanced. Earlier work suggests that this can lead to a better overall performance (Janikow and Michalewicz, 1991). Table 1 gives an overview of the individual functions of every individual gene of the chromosome.

The first 15 bits encode for the specific instruments used for the general instrument groups. As mentioned before the sound library that was available consisted of many different instruments. From this library a set of possible candidates is selected for every instrument group, so the algorithm can choose one instrument for every instrument group out of the corresponding set of instruments. In other words, a small set of instruments that are suitable for the specific role that instrument group should play is selected for every instrument group. The different combinations of instruments the algorithm can choose are encoded by the first 15 bits. Instruments that are suitable for the role the Hat instrument groups should play are short high pitched instruments like a high hat or a cymbal. Instruments that are suitable for the role that the percussion instruments should play are more general and more low pitched instruments like a low tom or a cow bell. The set of instruments in the melody instrument group consists of both acoustic (piano, violin, etc.) and electronic (synthesizers) instruments.

The Percussion1 instrument only plays once every 4 beats which is why only 4 bits are sufficient to determine when to play the sound. Hat1 can be played every 1/2 beat and is looped every 4 beats meaning it only needs 8 bits to determine when to play. Hat2 and Percussion2 can be played every 1/4 beat and are looped every 4 beats meaning they both need 16 bits. The melody can be played every 1/4 beat and is looped every 8 beats meaning it needs 32 bits to determine when it is played. The value coded bits 84 to 147 work in pairs. Bit 84 and 85 determine the pitch of the first melody note together, if bit 84 has value 4 and bit 85 has value 8 than the first melody note plays at pitch 48. Bit 86 and 87 determine the pitch of the second melody note, bit 88 and 89 the third, etc.

3 Method

This chapter will explain how the process of the actual experiment will be executed and how the outcome will be tested on a small group of around 40 participants.

3.1 The Experiment

The experiment is the part where the algorithm is supposed to traverse the space of possible chromosomes helped by the feedback of the evaluator. The process of evaluating the individuals of every generation is done in 30 minute intervals for two reasons. The first reason is described by Spector et al. (2005) as the long-term garbage exposure problem. Spector and his colleagues describe how the evolutionary operators will mostly produce garbage because they are random operators. Sorting through all this garbage to find the treasures is a difficult task that becomes progressively harder as the ear gets more numbed with every piece of garbage it hears.

The second reason is the problem of similarity in later generations. As the generations progress the population will start to become more and more similar to each other. This means that especially in the later generations each chromosome will sound very similar to the previous one you listened to and the next one you will listen to. This leads to two problems. One is that when you hear one sound over and over again a change in this repetition will automatically sound strange regardless of whether it sounds better or worse, this can negatively impact the quality of the evaluation. A second problem is that it can become annoying to hear the same piece of music over and over again. This can cause a decrease in motivation for the evaluator and therefore decrease the evaluation quality. To minimize the fore mentioned problems the evaluator will be allowed to evaluate for 30 minutes after which he has to take a break of at least 30 minutes. The time needed for the evaluation of one generation is expected to be around 2 minutes. Meaning that the evaluator can generate about 15 generations before he has to take a break.

For this research the number of generations before the final result will be 100 generations. The evaluator stays the same throughout all generations to make sure evaluation criteria stay the same throughout all the generations. The evaluation criteria are the following:

- The rhythm of percussive elements is pleasant.
- The melody is pleasant.
- Overall listening pleasure.

The evaluator uses these 3 criteria to create a mental ranking of the 10 chromosomes in order to pick the two he thinks sounds the best.

3.2 Performing a Musical Turing test

When making the Turing test, Alan Turing intended the test to be an assessment of intelligence in computers, not for the assessment of computational creativity. But as described in chapter one, computational creativity can be characterized in a manner parallel to computational intelligence. This is why inspiration was taken from the classical Turing test to create a musical variation, the Musical Turing test. A classical Turing test consists of two entities (a computer and a human) and an interrogator. The interrogator can interact with both entities and has the task to distinguish which entity is which (Turing, 1950). The main difference of the Musical Turing test compared to the classical Turing test is the interaction part. Where the interrogator is able to interact with the entities through own formulated questions in the classical Turing test, the interrogator is only able to interact with the entities through consuming the art pieces produced by them in the Musical Turing test. In other words, instead of classifying the answers to some queries as human or computer, the interrogator is now asked to classify art works as either made by a human or made by a computer (Chamberlain et al., 2015). To eliminate the 50% correct guess chance the classic Turing test has the decision was made to let the participants compare the computer generated music fragment to three artist-created music fragments instead of one.

In order to determine whether the piece of music generated by the ETM algorithm can be distinguished from man made music, 39 participants between the ages of 18 and 65 from the Netherlands will be questioned through an online survey. In the survey the participants are first given a short explanation about the aim of the survey. It is made clear to the participants that they are contributing to a bachelor thesis and that they are going to have to distinguish between

artist-created music and computer generated music. It is also mentioned that the three other comparison songs are respected songs released around 15 years ago to establish a baseline for what they can expect from the songs and to prevent participants from thinking the comparison songs were deliberately chosen to be bad or unlikeable as will be explained further in this sub-chapter, they were not. They are then shown a playlist of four music fragments that are about 15 seconds long. They can play all of the songs separately as many times as they want. While they are able to play the songs, they are first asked whether they are already familiar with one or more of the options. If they indicate that they are familiar with one or more options the result will not be counted. They then are asked to choose which fragment they think was made by a computer.

The three other songs that the participants will have to compare the ETM created piece of music to are the following and will from now on be referred to as Option x for the corresponding song (with Option 1 being the ETM generated music fragment):

- Option 2 - THEO PARRISH Falling up - CARL CRAIG remix (2005)
- Option 3 - Ricardo Villalobos – Hireklon (2004)
- Option 4 - Audion - Mouth to Mouth (Original Mix) (2006)

These songs were chosen via the most wanted record list of their respective release year in the techno category on Discogs.com. Discogs.com is the worlds leading online marketplace for vinyl records with a userbase of over half a million users as of 2021. A record being in the most wanted list of its genre should be a good indication that it is a well-respected song. As mentioned before, the sound library the algorithm uses was made in 2004 so the comparison songs have to be produced around the same time. Besides the ranking on Discogs.com and the release year, the instruments, beats per minute and overall sound have to somewhat be comparable with each other and the sound library. With these constraints in mind Option 2 was chosen because it was the second most wanted record in its release year, Option 3 was the second most wanted and Option 4 was the fifth most wanted in its release year.

4 Results

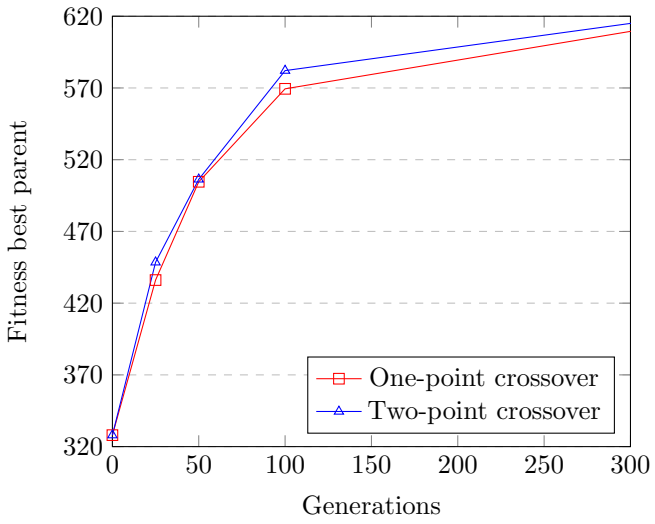

Figure 1: Crossover operator comparison with mutation factor 0,12.

For both crossover operators a mutation factor of 0,12 turned out to perform the best. Both crossover operators had the steepest slope in the first 150 generations with that parameter. Lower mutation factors sometimes lead to a faster ultimate solution, but not within a realistic number of generations for the purpose of this research (the fastest still only found a solution in an average of 1000+ generations). A mutation factor of 0,12 performed the best in the range of realistic numbers of generations which is why this parameter was chosen. In Figure 1 the comparison between the two crossover operators is shown. Both crossover operators performed similarly well but because the two-point crossover operator performed slightly better, the decision was made to use the two-point crossover with a mutation factor of 0,12 for the ETM algorithm for the rest of the experiment.

After the generation of the piece of music, 39 participants were asked to fill in the online survey. Because three of them were already familiar with at least one of the three comparison songs their data was removed from further analysis. The data set analysed from here consists of 36 entries. The percentage of votes for computer generated per option are displayed in Figure 2. Apparently Option 4

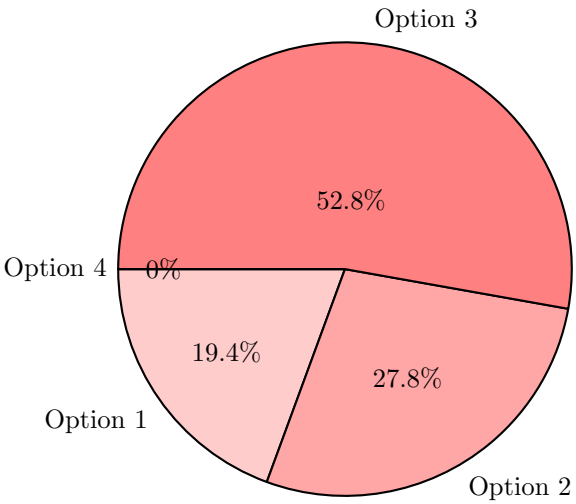

Figure 2: Percentage of participants that choose the option to be the computer generated one.

was so easily distinguishable as man made that it did not get a single vote but with 19.4% of the votes the ETM generated music fragment, Option 1, came in second to last. Meaning that the participants considered Option 2 and 3 to be more likely to be computer made than option 1.

5 Discussion

This research set out to investigate the feasibility of techno music creation by an evolutionary algorithm. We have seen that the small piece of music the ETM algorithm has generated got 7 out of 36 votes e.g. it was indicated as being made by a computer as opposed to made by a human by 7 out of the 36 participants. We have the following null hypothesis: most participants recognise Option 1 as being made by a computer e.g. 18 or more out of 36 participants indicated Option 1 is computer made. The probability that 7 out of 36 participants indicate Option 1 as computer made given the null hypothesis is $P = 0$ because not more than 18 participants correctly identified Option 1 as computer made, in fact only 7 did. Because of this the null hypothesis can not be true. With a $P = 0$ is smaller than 0,05 the null hypothesis gets rejected and the alternative hypothesis

gets accepted: Most participants can not recognise Option 1 as being made by a computer.

In conclusion, the results suggest the ETM algorithm has successfully created a music fragment that is indistinguishable from artist-created music fragments. This means it is feasible to use an evolutionary algorithm to generated short pieces of techno music. One of the benefits of generating techno music compared to classical sheet music like we have seen established projects like Lamus do is that the algorithm that generates the piece of techno music can also be equipped to play that music. The purpose of this paper was to investigate the feasibility and in doing so lay a foundation for further research on techno music generation with evolutionary algorithms. For this purpose there are multiple points that can be improved on to further validate the capability of music creation by evolutionary algorithms.

5.1 The limits set by NetLogo

NetLogo has been a good programming language to conduct this research in, but its limited capabilities put strong constraints on the sort of music fragments you can make. This research focused on small pieces of music between 10 and 15 seconds. This short amount can be achieved by looping the same 8-beat loop for about three or four times. In techno, the looping of the same loop is common but it goes without saying that an actual song has more dynamic elements than just one loop that loops over and over for 6 minutes straight. The lack of dynamic pitch manipulation (changing the pitch of a long note while its playing), dynamic velocity manipulation (changing the loudness of a note while its playing) along with other problems like the inability to add custom sound samples or the inability to apply common sound filters like reverb or echos make NetLogo unsuitable for the creation of any music pieces larger than one loop.

The inability to add custom sound samples is a problem for larger pieces of music but already posed its challenges when creating the 8-beat loop. The standard sound library provides all the basic sounds a techno song needs but nothing more than the absolute basics. When making songs, artist generally have huge sound libraries that they can choose from and if they don't find that specific thing they want they can make their own samples. The very basic sound library

therefore limits the musical expressiveness of the songs generated. The research tried to compensate this by using comparison songs that originated from a similar age as the library. The limited library however also lead to the use of a dummy sound for the kick in order to emulate the end result as best as possible. This may have had a negative influence on the evaluation process.

5.2 The limits set by the Evaluator

As described in chapter 3.1 the process of chromosome evaluation brings a lot of challenges. For this research a limit of 100 generations was set to accommodate for these problems that especially the later generations bring. From the tests that were performed to determine the operators and parameters described in chapter 2.2 we know that after 100 generations the optimal solution has not been reached. At generation 100 the 2-point crossover operator with mutation factor 0,12 had achieved an average fitness of 582,1 out of a possible 659. This means that at generation 100 there is still a lot of progress to be made. The average time that optimal solution was found however took a little over 2000 generations. With 30 generations evaluated per hour, it is obvious that this is not a realistic goal in the setting of this research and should be a consideration for further research.

5.3 ETM Parameters and Operators

As described in chapter 2 the ETM operator used a standard two-point crossover and a standard random mutation operator. It is possible that more complex operators or different parameters would have worked better. The reason these operators were chosen was by testing them on a less complex fitness function (the sum of all bit values). There is no guarantee that the performances on this less complex fitness function transfer over to a more complex one. In other words, the fact that the operators worked well for summing up the bit values does not mean they will perform well for producing good sounding pieces of music.

5.4 Validity of a Turing test

Despite the influence it has had, the Turing test has been a much debated topic for many years now. Most A.I. researchers see the Tur-

ing test as a distraction from the main goal of A.I. They see passing the Turing test as using trickery to pretend to achieve a goal instead of actually achieving that goal (Pease and Colton, 2011). Whether this is true or not is beyond this paper but it is worth mentioning that many of the philosophical arguments against the Turing test do not apply to the Musical Turing test that was performed in this research. According to Turing a computer has achieved intelligence when it is indistinguishable from a human. It is the use of the concept intelligence where most of the critique is about (Saygin et al., 2003). However, in this research this concept of intelligence is removed from the equation and the Turing test was only used as inspiration to establish whether the ETM algorithm could generate a short piece of music that was indistinguishable from man made music.

5.5 The Participants

Lastly, it is worth mentioning that because of the way the participants were reached out to, most of the participants were younger than 25 years old (28 out of 36) and most of the participants reported listening to techno music at least once a month (22 out of 36). However, the data suggests these characteristics have little to no influence on the results. The percentages between participants that never listen to techno and listen to it at least once a month only differed 2 % at most. Though beyond the scope of this paper this is an interesting observation for future research.

6 Conclusion

We have seen how the ETM algorithm was able to make an 8-beat loop that when looped a couple times made a small piece of techno music. When shown to a group of participants and asked to be distinguished from man made music we saw that the participants were unable to do so. This confirms the feasibility of techno music generation by the ETM algorithm opening up the route for further research on the generation of longer techno music fragments by evolutionary algorithms.

For these longer fragments however, it is important to incorporate some form of dynamic sound manipulation and the ability to

use custom sounds into the algorithm. Another problem this research ran into was the extremely limited evaluation capabilities one evaluator brings, and this problem will only becoming bigger when researching longer music fragments. There are some efforts being made on the field of automatic music evaluation to solve this issue, but this is still very much in its infancy Ren et al. (2020). A maybe more accessible solution to the evaluation problem at this time would be a system of multiple evaluators that all have to follow the same specific evaluation criteria in a very disciplined manner to optimize evaluation speed and minimize the evaluator differences.

References

Bessaou, M. and Siarry, P. (2001). A genetic algorithm with real-value coding to optimize multimodal continuous functions. *Structural and Multidisciplinary Optimization*, 23:63–74.

Briot, J.-P. and Pachet, F. (2018). Deep learning for music generation: challenges and directions. *Neural Computing and Applications*, 32(4):981–993.

Chamberlain, R., Mullin, C., and Wagemans, J. (2015). The artistic turing test: An exploration of perceptions of computer-generated and man-made art. *Journal of vision*, 15:112.

Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection*. Murray, London. or the Preservation of Favored Races in the Struggle for Life.

Diaz-Jerez, G. (2011). Composing with melomics: Delving into the computational world for musical inspiration. *Leonardo Music Journal*, 21:13–14.

Dostál, M. (2013). *Evolutionary Music Composition*, pages 935–964. Springer Berlin Heidelberg, Berlin, Heidelberg.

Fiebrink, R. and Caramiaux, B. (2016). The machine learning algorithm as creative musical tool.

Goldberg, D. E. and Holland, J. H. (1988). *Genetic Algorithms and machine learning*.

Hochberg, B. (2020). Youtube won't take down a deepfake of jay-z reading hamlet — to sue, or not to sue. https://www.forbes.com/sites/williamhochberg/2020/05/18/to-sue-or-not-to-sue—that-is-the-jay-zs-deepfake-question/?sh=103225c8128b retrieved on 18-12-2020.

Huang, A. and Raymond, W. (2016). Deep learning for music. *arXiv preprint arXiv:1606.04930.*

Janikow, C. Z. and Michalewicz, Z. (1991). An experimental comparison of binary and floating point represenation in genetic algorithms. *ICGA*, 1991:31–36.

Julien, O. and Levaux, C. (2018). play it again (and again), sam. *Bloomsbury Academic*, pages 1 – 10.

Malik, I. and Ek, C. H. (2017). Neural translation of musical style.

NewScientist (2012). Computer composer honours turing's centenary. https://www.newscientist.com/article/mg21528724-300-computer-composer-honours-turings-centenary/ Retrieved on 18-12-2020.

O'Hara, K. and Brown, B. (2006). *Consuming Music Together: Social and Collaborative Aspects of Music Consumption Technologies*, volume 35.

Pease, A. and Colton, S. (2011). On impact and evaluation in computational creativity: A discussion of the turing test and an alternative proposal.

Ren, I., Volk, A., Swierstra, W., and Veltkamp, R. C. (2020). A computational evaluation of musical pattern discovery algorithms.

Saygin, A. P., Cicekli, I., and Akman, V. (2003). *Turing Test: 50 Years Later*, pages 23–78. Springer Netherlands, Dordrecht.

Spector, L., Klein, J., and Harrington, K. (2005). Selection songs: Evolutionary music computation. *YLEM Journal*, 25:24–26.

Toivonen, H. and Gross, O. (2015). Data mining and machine learning in computational creativity. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(6).

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59(236):433–460.