

**NESTED MONTE CARLO SEARCH WITH  
MARKOV SAMPLING FOR FIRST SPECIES  
COUNTERPOINT**

Roberto Noorduijn Londono

Master Thesis 00-00

THESIS SUBMITTED IN PARTIAL FULFILMENT  
OF THE REQUIREMENTS FOR THE DEGREE OF  
MASTER OF SCIENCE OF ARTIFICIAL INTELLIGENCE  
AT THE FACULTY OF HUMANITIES AND SCIENCES  
OF MAASTRICHT UNIVERSITY

Thesis committee:

Prof. dr. ????

Prof. dr. ????

Dr. ir. ????

Dr. ?????

Maastricht University  
Department of Data Science & Knowledge Engineering  
Maastricht, The Netherlands  
July 2016



# Preface



# Summary



# Contents





# Chapter 1

## Introduction

### 1.1 AI and Music

As far back as the 18th century, many attempts were made at making so-called *musical games*. The purpose of these games was to be able to automate the compositional process such that novel pieces of music could be created from random events such as the rolling of a die. The most popular of these musical games is the one attributed to Mozart and was published in 1792 **Undefined reference**. The game consisted of using dice to determine the order of various precomposed sections of music. Although the structure of the final piece would be determined through random events, the actual aesthetic was fully determined by the composer by creating musically pleasing sections that would sound good no matter which order they were played in. That said, such musical games, though trivial, demonstrate the first foray into the field of Algorithmic Composition (AC).

The term Algorithm Composition refers to the use of algorithmic procedures in order to create music such that the creative involvement from part of the human is minimized. As the years past, many examples chronicled the development of various AC techniques. For example, the serialist movement, a departure from the traditional stances on harmony and melody, gave rise to the twelve tone method as developed by Arnold Schoenberg (**Undefined reference**). A method which used various rules in order to create entirely *atonal* pieces of music.

Unsurprisingly, as computers have advanced, they have allowed for an incredibly wide range of new capabilities within the field of AC. Initially, these methods were predominantly focused around hard coding rules and functions that would determine the structure and content of various pieces. Although these methods become more and more sophisticated, the creative process underlying the piece was still attributed to the formalisation of the rules and parameters used. The programmer who determined the rules was still essentially the composer of the piece. With the development of various techniques from Artificial Intelligence (AI), the difference between programmer and composer has become more distinct. This distinction arises from the fact that the programmer does not hard code the rules and creative process but instead allows the program to make the stylistic choices on its own. These advancements have allowed the field of AC to become inbedded within that of Computational Creativity (CC), i.e, the use of computation techniques and algorithms for the purpose of emulating creativity. A wide variety of ideas and techniques have already shared many successes within the domain of musical composition.

### 1.2 Combinatorial Optimzation and Counterpoint

A completed composition can in a certain sense be abstracted as a solution to a problem of *combinatorial optimization* (**Undefined reference**). In this regard, the act of composing a piece can be seen as similar to that of writing down a series of *musical events* such that the resulting music maximises some *objective function*. Of course, true composition is not so simple. With music, it is difficult to define such an objective function so that the aesthetic quality of two pieces of music can be compared using these functional values. While some general rules and heuristics do exist for various styles of music, they tend to not be strict rules and as such are generally suggestions based on experience.

Species counterpoint is an example of a musical style with such rigorous and strict rules that an

objective function can in fact be determined. Generally speaking, counterpoint refers to the use of independent vocal melodies, which are combined to form a harmonious whole. Although when discussing the *style of counterpoint* one is instead referring to the set of principles that were amassed and developed during the Renaissance, in relation to the art of voice polyphony. These principles were then formalised into sets of rules, such that the style of counterpoint could easily be taught by following said rules. The most complete collection of these rules was published by J.J. Fux in 1725 in his *Gradus ad Parnassum*. In here, he describes different *species* of counterpoint and corresponding concise rules to help aid the composition thereof.

Due to their rigorous formulation, these rules have been implemented and used by many different optimization algorithms in order to determine good solutions to Fuxian counterpoint. An early example is Schottstaedt's approach which made use of search and heuristics derived from these rules (**Undefined reference**). However, due to the almost infinite amount of possible melodies one could create - without imposing any restrictions, the majority of recent approaches have focused on stochastic methods. One such example is evolutionary algorithms as shown in **Undefined reference** and Jacobs (2012). Such methods have proven to be quite successful in generating novel fugues given a starting melody or *can-tus firmus*. These approaches are examples of the *generate and test* methods (**Undefined reference**). Pieces are generated, and then tested by using either human evaluation or direct evaluation derived from similar rules as the ones explained above. Another approach which seems to have a stronger theoretical foundation, is that of reinforcement learning using SARSA (State-Action-Reward-State-Action) (**Undefined reference**). The idea behind this is to model the act of composing music as a Markov Decision Process. States are represented as pitches and the actions are represented as intervals. The rewards of the models are then derived from evaluation criteria which rewards consonant sounds and pitch intervals and penalises dissonance. These techniques have all had reasonable success, although they still far short of being considered similar to a human composed piece of music. The method proposed by **Undefined reference** combines both evolutionary algorithms and recurrent neural networks in order to generate new melodies. Melodies were then evaluated using tonal and rhythmic rules, such as tonal and rhythmic diversity. In (**Undefined reference**) new music is generated by using Markov Chains analysed from existing compositions. For example, pitch frequencies and their transitions would be recorded and incorporated into a Markov Chain. New pieces would then be made by sampling from this Markov Chain.

### 1.3 Problem Statement and Research Questions

Search based methods such as Monte Carlo Search (MCS) have not generally been applied to the task of music generation. The greater focus of methods have relied on stochastic methods such as Markov Chains and Evolutionary Algorithms. Since the task of composing is generally not a random process, but rather can be seen as a more informed (heuristic) search, it appears as though search based methods should be feasible for this problem. Such approaches have been applied with best-first search methods (**Undefined reference**), although due to the intractability of the problem, they have not been as successful as the stochastic methods. Based on their previous successes in similar domains, MCS based methods are therefore expected to produce pieces of equal if not greater quality than those of the more popular stochastic methods. The problem statement of this thesis is thus:

**Problem Statement.** *Are Monte Carlo Search based approaches viable to the domain of Automated Musical Composition?*

Thus the purpose of the thesis will be to investigate how well MCS based techniques fare compared to the state of the art techniques currently used. In particular, these techniques will be applied in order to create music in the style of First Species Counterpoint. This problem statement will be addressed by answering the following research questions.

**Research Question 1.** *How do Monte Carlo based approaches compare to Genetic Algorithms within this domain?*

**Research Question 2.** *What is a suitable representation for music within the context of counterpoint?*

In searching the space of counterpoint efficiently, it is necessary to have a concise representation in order to represent music as points within a specific state-space. Such a representation will effectively

require a formalisation of important musical concepts. In representing music, there exists a spectrum between direct or explicit representations and more indirect or implicit representations. An example of a direct representation would be a collection of MIDI information. That is, the pitch, duration, velocity (loudness) and attack time of a particular music event (e.g. a note played on a piano). A slightly more implicit approach would be to only model aspects such as relative pitches between notes, consonance and chord progression. An example of a representation in the more extreme implicit end of the spectrum would be that described by the *Generative Theory of Tonal Music*, a grammatical hierarchical approach to music representation. This representation deals more with the cognitive aspects of music theory and thus is generally more difficult to implement. (Undefined reference) describe a method for automatically analysing music according to this theory. It is expected that an approach that lies somewhere in between would be the most appropriate as it is necessary to have a more direct representation in order to simplify the task of searching, but also it would be necessary to encode more information in order to represent more complex relationships inherent in the music.

**Research Question 3.** *How can the contrapunctal pieces be evaluated?*

As discussed earlier, the evaluation of music is difficult problem. The main approach in this thesis will be to take the qualitative rules of first species counterpoint and turn them in quantitative measures. Other approaches to determining a feasible evaluation will also be researched. These include using features learned from a corpus of pre-existing music and seeing how closely the generated music resembles music from the corpus.

Since the actual fitness of a peice of music tends to be a subjective matter, the results will have to be evaluated using human participants. To this extent the various results from the different techniques and evaluation functions will be used in a survey carried out on a number of people with varying musical maturity. The purpose of such a survey will thus be to truly see how well the different pieces sound. The results of this survey will also be able to answer another research question:

**Research Question 4.** *Using just the rules derived from first species counterpoint, how good are the pieces generated?*

Due to species counterpoint being mainly a pedagogical tool that was created in order to aid students in the understanding of contrapunctal composition, it has never been a standard under which music is actually composed. The main reason for this is the fact it is very difficult to ensure that all the rules have been satisfied. It is also especially difficult for even an expert, to be able to listen to a piece of species counterpoint and determine whether it satisfies all the rules. Given that an appropriate formalisation of the rules have been determined, it is expected that a piece of music that obtains the highest fitness value from the evaluation function should represent the best piece of species counterpoint. Therefore, under the previous assumption, the survey test will serve to reveal whether or not these rules do in fact create aesthetically pleasing music.

**Research Question 5.** *Can Markov Chains be used in order to improve the MCS methods?*

The purpose of this thesis is to investigate different techniques from AI applied to the problem of *first species counterpoint*. In particular, a new method based on Nested Monte Carlo Search (NCMS) will be proposed and compared to existing techniques from the literature. This problem will be addressed by describing the exact rules that will be used for evaluation, discussing the representation and then setting up the proposed algorithm

## 1.4 Outline



## Chapter 2

# Music Theory

Music theory is based on the analysis of music. This typically reflects an analysis of different fundamental aspects of music. These include melody, harmony, rhythm and form. The following sections aim to introduce these concepts. Though music theory has developed to encompass many different forms of music, for the purpose of this research, the concepts introduced will be those that are in relation to the traditions of Western music.

### 2.1 Pitch

Pitch is a perceived property of sound directly related to frequency. It is the quality of sounds that allows us to distinguish between higher and lower sounds. When a string vibrates at a frequency of 440hz this is referred to as the pitch A. Though pitch is related to frequency, human perception of pitch depends not linearly on frequency but rather logarithmically. Therefore, the difference between 220hz and 440hz sounds is perceived as the same difference between 130.81hz and 261.63hz. Such a difference between pitches is called an *interval*. In particular, the intervals just described are denoted as an *octave*. Octaves are special intervals as they can be used to partition pitches into equivalence classes. The following demonstrates how to do this. Let  $\sim$  be a binary relation such that:

$$a \sim b \equiv \frac{a}{b} = 2^k \quad \text{for some } k \in \mathbb{Z} \quad (2.1)$$

Clearly,  $\sim$  is an equivalence relation for the space of frequencies. Equivalence classes thus represent similar sounding tones and are therefore denoted by the same letter. When one refers to A, one is actually referring to a collection of different pitches  $[A] = \{\dots, 55, 110, 220, 440, 880, 1760 \dots\}$ . These equivalence classes are typically called *pitch classes*. Although the space of possible frequencies is a continuum of different values, typical Western music theory only deals with a finite subset of frequencies represented by 12 different pitch classes. The pitch classes are referred to as A, A#, B, C, C#, D, D#, E, F, F#, G, G#. Since pitches are on a logarithmic scale, they can be transformed into a linear scale with the following function:

$$M(x) = 69 + 12 \log_2 \frac{x}{440} \quad (2.2)$$

$M(440) = 69$  and  $M(220) = 57$  (see Table ??). In this scale, the difference between two adjacent pitches is called a *semitone*. In order to distinguish between different tones of the same pitch class, i.e., A at 440hz and A at 220hz, we use the *Scientific Pitch Notation*. In this notation we use a number suffix to represent whether a tone is higher or lower, see the table ??.

The value of a pitch can be altered by using *accidentals*. These are shown by the symbols  $\flat$  and  $\sharp$ . Writing  $\flat$  next to a pitch lowers the pitch by a semi-tone, whereas  $\sharp$  raises it by a semitone.

### 2.2 Scales

Let the set of all pitch classes be given by  $\Omega = \{A, A\#, B, C, C\#, D, D\#, E, F, F\#, G, G\#\}$ . A *scale* is any non-empty ordered subset  $S \subseteq \Omega$ . The scale  $S = \Omega$  is called the *chromatic* scale. Scales are typically

Table 2.1: Relations between Pitch, Pitch class and Frequency

Pitch	Pitch Class	Freq (hz)	$M(x)$	Freq. diff.	$\log_2(\text{Freq. diff.})$
<b>A3</b>	<b>A</b>	220	57	-	-
<b>A#3</b>	<b>A#</b>	233.08	58	13.08	$\frac{1}{12}$
<b>B3</b>	<b>B</b>	246.94	59	13.86	$\frac{1}{12}$
<b>C4</b>	<b>C</b>	261.63	60	14.69	$\frac{1}{12}$
<b>C#4</b>	<b>C#</b>	277.18	61	15.55	$\frac{1}{12}$
<b>D4</b>	<b>D</b>	293.66	62	16.48	$\frac{1}{12}$
<b>D#4</b>	<b>D#</b>	311.13	63	17.47	$\frac{1}{12}$
<b>E4</b>	<b>E</b>	329.63	64	18.50	$\frac{1}{12}$
<b>F4</b>	<b>F</b>	349.23	65	19.60	$\frac{1}{12}$
<b>F#4</b>	<b>F#</b>	369.99	66	20.76	$\frac{1}{12}$
<b>G4</b>	<b>G#</b>	392.00	67	22.01	$\frac{1}{12}$
<b>G#4</b>	<b>G#</b>	415.00	68	23.00	$\frac{1}{12}$
<b>A4</b>	<b>A</b>	440.00	69	24.00	$\frac{1}{12}$

Table 2.2: Scale Degrees for the C major key

Pitch	Scale Degree	
C	Tonic	$\hat{1}$
D	Supertonic	$\hat{2}$
E	Mediant	$\hat{3}$
F	Sub-dominant	$\hat{4}$
G	Dominant	$\hat{5}$
A	Sub-median	$\hat{6}$
B	Leading note	$\hat{7}$

used to represent the types of pitches used in a piece of music. Although a musical piece may use one scale, one can use other pitches through the use of accidentals mentioned above.

Of great importance to early Western music are the *Diatonic* scales. These are scales which consists of 7 different pitches and can be split into the *major* and *minor* scales. For each of the pitch classes, there is a distinct diatonic scale defined which uses that pitch class as its starting point, also called the *tonic*.

An invariant feature among all different diatonic scales are the amount of semi-tones - or intervals - between consecutive pitches. For example, the diatonic C major scale is given by the pitches {C, D, E, F, G, A, B} while the G major scale is given by {G, A, B, C, D, E, F#}. Examining these scales, one can see that the number of semi-tones between consecutive pitches are {2, 2, 1, 2, 2, 2, 1} where the final semi-tone is the transition from the final pitch to the starting note, one octave up. The interval differences for the diatonic minor scales are given by {2, 1, 2, 2, 1, 2, 2}. Using these differences one can see that the A minor scale is given by {A, B, C, D, E, F, G}. Since C major and A minor share the same pitches, A minor is called the *relative minor* of C major. Similarly, C major is the *relative major* of A minor.

Within these diatonic scales, each pitch is given a particular name referred to as the *scale degree*. These scale degrees reflect the different qualities and functions of the pitches within a certain scale. Table ?? shows the scale degrees corresponding to the pitches in the C major scale. The tonic, sub-dominant and dominant are particularly important scale degrees and their functions will be discussed in more detail later.













## 2.3 Intervals

As discussed above, intervals measure the distance between two pitches. While intervals can be measured using the semi-tone distance between pitches, it is clearer to refer to intervals relative to their positions in a specific scale. The following shows the intervals relative to C in the C major scale.

- C  $\rightarrow$  D, 2 semitones, **major second**
- C  $\rightarrow$  E, 4 semitones, **major third**
- C  $\rightarrow$  F, 5 semitones, **perfect fourth**
- C  $\rightarrow$  G, 7 semitones, **perfect fifth**
- C  $\rightarrow$  A, 9 semitones, **major sixth**
- C  $\rightarrow$  B, 11 semitones, **major seventh**
- C  $\rightarrow$  C, 12 semitones, **octave**

Other intervals are possible. Minor intervals are those which are a semi-tone less than major intervals, such as the minor third C  $\rightarrow$  Eb. Augmented intervals raise the value of perfect intervals by one semitone while diminished intervals lower the value by one semitone.

Table 2.3: Durations of Notes and Rests

Symbol	Name	Duration	Equivalent Rest
	whole note	1	
	half note	$\frac{1}{2}$	
	quarter note	$\frac{1}{4}$	
	eighth note	$\frac{1}{8}$	
	sixteenth note	$\frac{1}{16}$	
	thirtysecond note	$\frac{1}{32}$	


## 2.4 Chords





A chord is a combination of 3 or more pitches sounding simultaneously. The most basic type of chord is the *triad*, composed of 3 pitches. Each triad is built off a base pitch or *root* and two intervals above the root. There are 4 different types of triads.

- **Major Triad** - root, major third and perfect fifth, e.g., a C maj. = (C, E, G).
- **Minor Triad** - root, minor third and perfect fifth, e.g., a C min. = (C, Eb, G)
- **Augmented Triad** - root, major third and augmented fifth, e.g., a C aug. = (C, E, G $\sharp$ )
- **Diminished Triad** - root, minor third and diminished fifth, e.g., a C dim = (C, Eb, Gb)



Another type of chord are the seventh chords. These are essentially the same as the triads, except with added augmented or diminished seventh interval. Chords do not strictly need to be played with the closest pitches, for example the C major key need not consist of the pitches (C4, E4 G4), but they can be (E2, G3, C4). The process of separating pitches over wider registers is called *voicing*. Voicing corresponds to the act of assigning separate voices or instruments to different parts of a chord.

## 2.5 Duration

Another important property of perceived sound, is that of duration. As the name implies, duration refers to the amount of time for which a particular sound persists. A pitch with a specified duration is referred to as a *note*. In music, duration is measured in terms relative to the whole note  which has a duration of 1 (see Table ??). Lack of notes are represented in music using rests.

The length of notes can be extended through two principle ways. The first way is by use of the  $\cdot$  operator. When placed next to a note, the dot operator increases the duration of that note by half its original duration. Thus  has a duration equal to  $\frac{3}{4}$ . Similarly the duration of  has a duration of  $\frac{3}{32}$ . The other way of increasing the duration of a note is by *tying* them together. Tying notes has the effect of adding the duration all of tied notes together. For example the duration of   is  $\frac{1}{2} + \frac{1}{8} = \frac{5}{8}$ . Patterns of varying durations and rests give rise to the phenomenon known as *rhythm*.

## 2.6 Music Notation

Music is typically represented using *Staff Notation*. It provides an organizational method that allows music to be read from left to right. A Staff consists of 5 lines and 4 spaces. Each line and space corresponds to a different pitch. Clefs are used to assign specific values to each position. The treble clef  assigns the value of G4 to the second line from the bottom, while the bass clef  assigns the value of F3 to fourth line from the top. Ledger lines are used to represent pitches that do not fit on the staff lines. See the following figures below (??, ??).



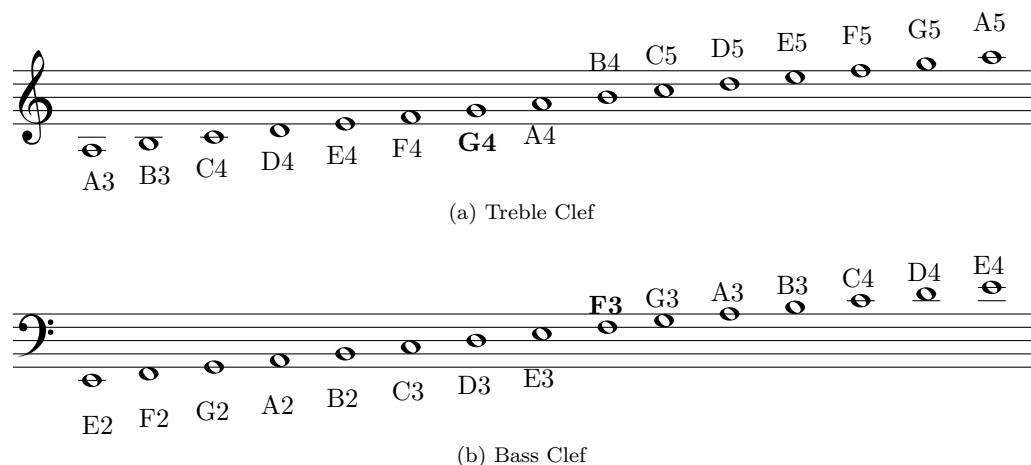
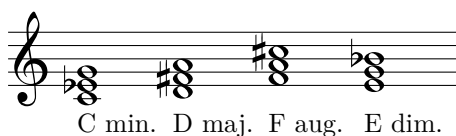


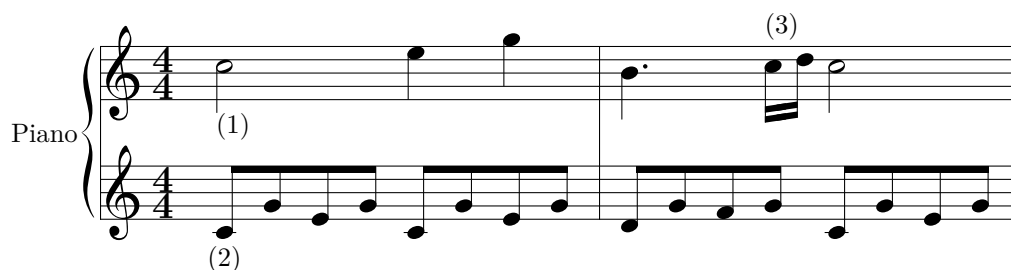
Figure 2.1: Staff notes with treble and bass clef

Chords are represented by writing the notes of which it consists of on top of one another. Accidentals ( , ) are used to denote other pitches than the ones shown in figures ?? and ??. For an example of accidentals, see figure ??.

Figure 2.2: Examples of chords and accidentals in staff notation



Vertical lines divide up the Staff into bars or measures. The length of these measures is determined by the two numbers next to the clefs. These two numbers represent a fraction that determine the length of the bar in terms of whole notes. These two numbers are called the *meter* of a piece. Thus, if the meter is  $\frac{4}{4}$ , then each bar lasts for the same duration as a whole note. It should be noted that, although there is a difference between similar meters such as  $\frac{4}{4}$  and  $\frac{2}{2}$ , such a discussion is beyond the scope of this chapter. Figure ?? shows an extract from a piece of music in  $\frac{4}{4}$ .

Figure 2.3: First two bars of W.A. Mozart's *Piano Sonata No. 16 in C Major*

Inspecting figure ?? one can see that the two staves are linked by use of vertical bar lines. This implies that the music contained in both bars is to be played simultaneously. The note at position (1) is the same as the standard  $\downarrow$  although it is written upside down for aesthetic reasons. At (2) eighth notes  $\uparrow$  are linked together by use of a *beam*. This is done to aid legibility of notes and to link together groups of notes that form a natural *phrase*.

## 2.7 Melody

A phrase or melody is a collection of successive notes that are perceived as a whole within a piece of music. Melodies are typically described by the intervals between successive notes as well as by its rhythm. A way of characterizing the overall structure of a melody is through its *contour*. The contour of a melody refers to the movement of the melody as well as the proximity between consecutive notes. An *ascending* melody is one in which pitches between consecutive notes is increasing, *descending* when the pitches decrease and *undulating* refers to when there is equal movement upwards and downwards. The proximity of a phrase can be described as either *conjunct* or *disjunct*. If a phrase is conjunct that means that intervals between notes are only seconds (major or minor, see the section on intervals) also called *steps*. Conversely, disjunct means that the interval between two consecutive notes is anything but a second, also called *skips*.

In Western music, melodies are usually played within one specific scale. When analysing a melody, one also pays attention to the role of the various pitches within a particular scale according to their diatonic function. For example, the tonic of a scale is considered the tonal home and thus usually provides a good starting and ending point for a melody. The dominant of a scale usually represents an increase in tension which can be resolved by tending back towards the tonic. Typically, notes with a particular function are played on a beat to emphasize their role. Notes whose purpose are only transitional between these functional notes are called *passing notes*.

## 2.8 Harmony

Harmony refers to the quality of music that is perceived when multiple pitches or voices are sounding. Harmonic analysis is generally built on the analysis of chords through *chord progressions* and *cadences*.

A chord progression is a sequence of chords within a particular piece. These sequences are typically understood in terms of diatonic chords and triads. Diatonic triads are special triads that are built within a given diatonic scale (major or minor). For each scale degree, a different triad is built on it, using that pitch as its root, the third pitch above the root in the scale as the middle pitch and the fifth pitch above the root as the top pitch. These triads are identified using roman numerals, corresponding to the scale degree of the root note. A capital roman numeral indicates that the triad is major, while a lower case one indicates that it is minor. A plus sign is used to indicate that the triad is an augmented one while a circle (°) indicates that it is diminished.

(a) C major Scale

(b) C Minor Scale

Figure 2.4: Diatonic Chords with Roman Numbering

A chord progression is thus identified by a sequence of identifiers, eg,  $I \rightarrow IV \rightarrow V \rightarrow I$ . Of particular interest are chord progressions known as *cadences*. These are chord progressions that sound to create a sense of *resolution* and, as such, are usually used to signify the end of a phrase or a piece of music. There are four main types of cadences:

- **Authentic Cadence** ( $V \rightarrow I$ ): This cadence has a strong sense of resolution as it alternates between the dominant (tension) and the tonic (home).

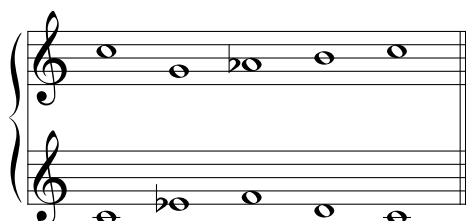
- **Half Cadence:** A half cadence refers to any cadence that ends on V. Since it ends on the dominant, it has the weakest sense of resolution of all cadences and thus has a strong desire for continuation (towards a resolution). It is called a half cadence as its ending chord is half of the authentic cadence.
- **Plagal Cadence (IV → I):** A cadence which involves the subdominant and the tonic. Since the subdominant is tonally less tense than the dominant, such a cadence has a weaker resolution than the authentic cadence.
- **Deceptive or Interrupted Cadence (V → vi):** The deceptive cadence is one which gives the listener a suspended feeling as it is very close to the authentic cadence but does not ultimately resolve to the tonic.

Another important aspect of harmony is that of *consonance* and *dissonance*. The difference between consonance and dissonance can vaguely be described as that which sounds "good" versus that which doesn't and, as such, is highly context dependant. Something that may sound dissonant in one culture could be consonant in another. One usually refers to consonance in relation to notes being played together (although one could talk about consonant rhythms and melodies). In western music, all perfect intervals (the octave, fifth and fourth) as well as the thirds and sixths (major and minor), all other intervals are considered dissonant. Though dissonance implies that something may not sound "good", it plays a pivotal role in tension building within music with dissonances typically being resolved to consonances.

## 2.9 Counterpoint

Counterpoint refers to the relation between different voices that are independent melodically yet come together harmonically to form a whole. The process of writing counterpoint is generally considered a difficult one as, for each melody, each note serves a dual purposes; melodically and harmonically. Species counterpoint is a form of *strict* counterpoint, in which counterpoint on a given melody, or *cantus firmus*, is created by following a series of rules.

Figure 2.5: Counterpoint



When discussing counterpoint, one usually refers to the resulting motion between two different voices. This motion can be characterized in 4 ways.

- **Parallel motion:** The voices move in a similar direction, mainting the same interval between notes.
- **Similar motion:** The voices is move in a similar direction, although the interval between notes changes.
- **Contrary motion:** The voices move in exactly opposite directions. When one voice moves up, the other one moves down.
- **Oblique motion:** One voice moves while the other remains on the same note.

The following describe some of the most important rules used within species counterpoint:

- Avoid unisons (except at beginning or end of the piece)

- Prioritise contrary motion
- Begin and end on perfect consonance (except for a fourth)
- Do not move in parallel fourths
- Do not move too much in parallel thirds and sixths
- Approach perfect consonances by oblique or contrary motion
- Intervals should not exceed more than a tenth

More rules are described for different *species* of counterpoint. The above rules are applied for when the two melodies fall exactly one on top of the other. Other rules need to be taken into account when the rhythmic structure of the two voices differ.

## 2.10 Imitation

In music, imitation refers to the repetition of a melody in a different voice in close succession. Typically a voice will start with the melody, shortly followed by another voice playing the same melody and so on. The melody may be repeated as is, resulting in *real* imitation, or it may be altered according to one or more transformations. These are as follows:

- **Inversion:** The intervals between notes are inverted. The inversion  $i'$  of an interval  $i$  can be calculated as follows:  $i' = 12 - i$ . Therefore if an interval consists of a perfect fifth  $i = 7$ , then the inverted interval is  $i' = 5$ , a perfect fourth.
- **Retrograde:** The melody is played backwards. The rhythmic structure of the original melody can be maintained or it can also be transformed so that each original note maintains their duration.
- **Augmentation:** Increasing the value of the duration of all the notes in original melody by a constant factor.
- **Diminution:** Reducing the value of the duration of notes by a constant factor.

Many different musical compositional techniques exist which make extensive use of imitative counterpoint. These include the *ricercar*, the *canon* and the *fugue*.

# Chapter 3

## Search

### 3.1 Monte Carlo Search

Monte Carlo Search (MCS) is a family of search techniques that have had much success, particularly within the fields of games, planning and optimization. Generally speaking, MCS is a best-first search strategy that combines random sampling in order to determine an optimal action given a specific domain. In order to describe the ideas of MCS it is useful to be able to describe the domain concretely. To this extent, it is necessary to define the *search domain* (**Undefined reference**), this notation will be used throughout the remainder of this thesis.

The search domain  $\mathbb{S}$  is defined as a 5 - tuple  $=S, S_T, A, f, R$  where:

- $S$  - the set of states.
- $S_T \subseteq S$  - the terminal states.
- $A$  - the possible actions.
- $f : S \times A \rightarrow S$  - the state transition function.
- $R : S \rightarrow \mathbb{R}^k$  - the utility function.

Within the domain, an *agent* progresses between states by performing an action. The state  $s_0 \in S$  is the unique starting state. At each state  $s_i$ , an action  $a \in A$  is chosen and the agent progresses to the next state  $s_{i+1} = f(s_i, a)$ . After performing an action, the agent receives a reward determined by the utility function  $R$ . Though an agent can obtain a reward for entering any state, rewards are usually only defined for terminal states. These rewards reflect the utility of a certain state and are generally normalized to values between  $[0, 1]$ . Since it is typically the case that not all actions are valid within a particular state, one can define a function  $\alpha : S \rightarrow 2^A$  which given a state  $s$  returns all *legal* actions within that state.

Within the context of search, the idea is to find a sequence of actions  $(a_0, a_1, \dots, a_t)$  that leads to a final state  $s \in S_t$  such that  $R(s)$  is maximized. Such a sequence is thus called an *optimal strategy* or *policy*. Algorithms such as MCS thus aim to provide approximations for an optimal strategy by estimating the true value of an action through random simulation and then using the actions with the best values in the strategy. The simplest example of MCS called *Flat Monte Carlo* is illustrated in Algorithm ???. Given a starting state, the optimal action to play is determined by, for each possible actions, making random playouts until a terminal state is reached, evaluating those states, and consequently propagating that value back through the tree. The optimal action is then the one which has the largest propagated value.

The algorithm described in ??? is highly dependant on the random rollouts and thus it is highly likely that the action obtained is not the optimal action. MCS techniques, in general, thus take advantage of repeated simulations in order to obtain more valid approximations to the values of the various actions.

#### 3.1.1 Nested Monte Carlo Search

Nested Monte Carlo Search (NMCS) is a variant of MCS that uses recursive layers of random simulations in order to determine an optimal move(**Undefined reference**). This algorithm (see Algorithm ??) is

**Algorithm 1** Flat MCS

---

```

1: procedure FLATMCS( $s$ )
2:    $max \leftarrow -\infty$ 
3:   for  $a \in \alpha(s)$  do
4:      $val \leftarrow R(\text{RANDOMROLLOUT}(f(s, a)))$ 
5:     if  $val > max$  then
6:        $a_{best} \leftarrow a$ 
7:        $max \leftarrow val$ 
8:   return  $a_{best}$ 
9: procedure RANDOMROLLOUT( $s$ )
10:   $s_{cur} \leftarrow s$ 
11:  while  $s_{cur} \notin S_T$  do
12:     $s_{cur} \leftarrow f(s_{cur}, \text{GETRANDOM}(\alpha(s_{cur})))$ 
13:  return  $s_{cur}$ 

```

---

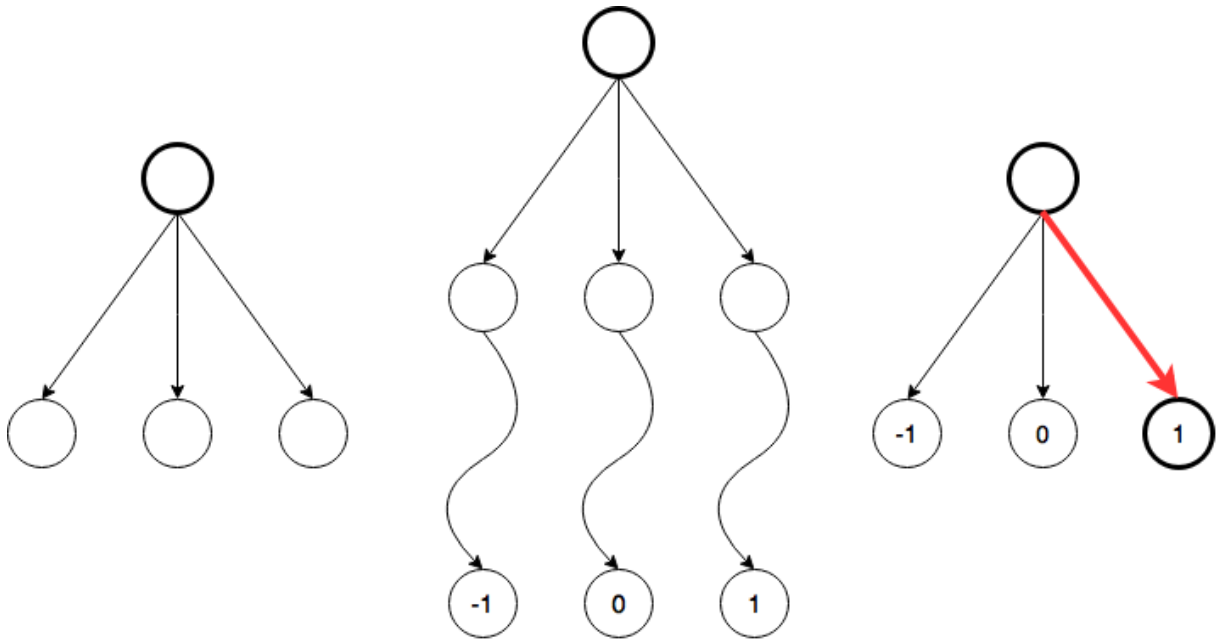


Figure 3.1: *Example of flat MCS. The right-most action is rated as the best and thus the player chooses to play it.*

typically used for 1-player games and thus instead of searching for the best action at a given state, it searches for the whole policy  $\pi_{nmcs} = \{s_0, \dots, s_t\}$ <sup>1</sup> where  $s_0$  is the initial state and  $s_t$  is a possible terminal state. The level  $n$  of NMCS defines the depth of recursion that is used in the search. For example a level-0 NMCS algorithm obtains the policy just by performing random rollouts from the initial state until it reaches a terminal state, lines 6 - 10 in the algorithm. NCMS defines utility in terms of policies  $R(\pi_{ncms}) := R(s_t)$ , i.e, the utility of a policy is the utility of its terminal state. Therefore, NCMS effectively searches to directly find the most optimal terminal state. At higher levels, a level  $n$  NCMS algorithm searches for the optimal policy by, for each possible successor state, determining the optimal policy through a level  $n - 1$  NCMS, lines 14 - 19. The state corresponding to the best optimal policy is chosen as the next state, and the algorithm continues. This idea is illustrated in Figure ??.

Nested Monte Carlo Search has proven to be an effective algorithm within the domain of single player games such as Morpion, Same Game and Crossword puzzle generation(Undefined reference). It is because of this success that it is believed that NMCS would be suitable for the problem of first species

---

<sup>1</sup>Here, some liberty is taken in deviating from the definition of the policy  $\pi$  as a function. Since NCMS deals with single-player games, it is sufficient to describe  $\pi$  as a set of states

counterpoint generation.

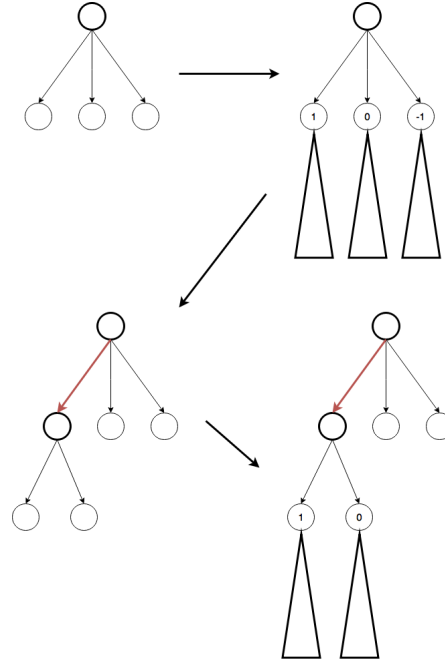


Figure 3.2: *Illustration of the  $n$ -level NMCS algorithm. The triangles represent an  $n-1$  level NMCS search.*

---

**Algorithm 2** Level- $n$  NMCS

---

```

1: procedure NCMS( $n, s$ )
2:    $ply \leftarrow 0$ 
3:    $\pi \leftarrow \emptyset$ 
4:   if  $n = 0$  then
5:      $\pi \leftarrow \{s\}$ 
6:     while  $s \notin S_T$  do
7:        $s \leftarrow f(s_{cur}, \text{GETRANDOM}(\alpha(s)))$ 
8:        $\pi.\text{ADD}(s)$ 
9:        $ply \leftarrow ply + 1$ 
10:    return  $(\pi, R(\pi))$ 
11:  else
12:     $max \leftarrow -\infty$ 
13:    while  $s \notin S_T$  do
14:      for  $a \in \alpha(s)$  do
15:         $(\pi_{temp}, score) \leftarrow \text{NCMS}(n-1, f(s, a))$ 
16:        if  $score > max$  then
17:           $max \leftarrow R(\pi_{temp})$ 
18:           $\pi.\text{ADD}(s, ply)$ 
19:           $\pi.\text{ADD}(\pi_{temp}, ply + 1)$ 
20:     $s \leftarrow \pi.\text{GET}(ply)$ 
21:    return  $(\pi, R(\pi))$ 

```

---

### 3.1.2 Nested Rollout Policy Adaptation

Nested Rollout Policy Adaptation (NRPA) is a method based on NMCS created by Rosin (**Undefined reference**). The algorithm is similar in the sense that it performs nested searches and at level - 0

performs a full playout. The main difference is that, apart from directly traversing the search tree based on the results of previous nested calls, NRPA adapts the rollout policy based on the scores obtained from the nested calls. The idea is that, as search progresses, the policy is updated such that good or promising moves are given increasing probabilities. The algorithm is described in ??.

---

**Algorithm 3** Level-n NRPA

---

```

1: procedure NRPA( $n, p$ )
2:   if  $n = 0$  then
3:      $s \leftarrow \text{root}()$ ,  $ply \leftarrow 0$ ,  $\pi \leftarrow \emptyset$ 
4:
5:     while  $s \notin S_T$  do
6:        $s \leftarrow f(s, \text{GETRANDOM}(\alpha(s), p))$ 
7:        $\pi.\text{ADD}(s)$ 
8:        $ply \leftarrow ply + 1$ 
9:     return  $(\pi, R(\pi))$ 
10:  else
11:     $max \leftarrow -\infty$ 
12:    for  $N$  iterations do
13:       $(\pi_{temp}, score) \leftarrow \text{NRPA}(n - 1, p)$ 
14:
15:      if  $score \geq max$  then
16:         $max \leftarrow score$ 
17:         $\pi \leftarrow \pi_{temp}$ 
18:
19:     $p \leftarrow \text{ADAPT}(p, \pi)$ 
20:    return  $(\pi, R(\pi))$ 
21:
22: procedure ADAPT( $p, \pi$ )
23:    $s \leftarrow \text{root}()$ ,  $ply \leftarrow 0$ ,  $p' \leftarrow p$ 
24:
25:   while  $ply < \pi.\text{LENGTH}()$  do
26:      $s' \leftarrow \pi.\text{GET}(ply)$ 
27:      $p'[\text{CODE}(s, s')] \leftarrow p'[\text{CODE}(s, s')] + \beta$ 
28:
29:      $z \leftarrow \sum_{a \in \alpha(s)} \exp(p[\text{code}(s, f(s, a))])$ 
30:
31:     for  $a \in \alpha(s)$  do
32:        $p'[\text{code}(s, f(s, a))] \leftarrow p'[\text{code}(s, f(s, a))] - \frac{\beta}{z} \exp(p[\text{code}(s, f(s, a))])$ 
33:
34:      $s \leftarrow s'$ 
35:   return  $p'$ 

```

---

The algorithm is called with two parameters,  $n$  and  $p$ , the search level and the initial *policy vector*. The policy vector represents the likelihood of an action being selected at a specific state. Therefore, the policy vector must contain a value for each action at each state. These state action pairs are encoded using the function  $\text{CODE}(s, s')$ , where  $s$  represents the current state and  $s'$  is the resultant state after applying a specific action. At level 0, NRPA performs a random playout where each action is chosen proportional to the values in the policy vector, lines 5 - 9. At higher levels, the algorithm searches for  $N$  results from  $n - 1$  level searches, choosing the best sequence based on its evaluation, lines 12 - 17. The current best sequence is then used to update the values in the policy vector using the  $\text{ADAPT}(p, \pi)$  procedure. For all states in the sequence, their weight in the policy vector is increased by a set amount  $\beta$ .



## 3.2 Markov Chains

Markov Chains (MC) are used to model specific random processes in which transitions between various states occur. The most important property that these processes must possess, is that the transition between one state to another must be *memoryless*. This means that the probability of going from one state to another is independent of the previous states. This chapter introduces the theory behind MCs and also describes how they can be applied to the domain of algorithmic music composition.

Let  $S$  be a set of states and assume that there is some process by which random states are selected at discrete time intervals. The sequence of random variables  $X_1, X_2, \dots$  represent the random state that is chosen at time interval  $1, 2, \dots$ , respectively. A realization of all these random variables  $s = (s_1, s_2, \dots)$  is called a *history* which thus represents the states chosen at each time index. The sequence of random variables is called a *Markov Chain* if they satisfy the *Markov Property*:

$$\Pr(X_{n+1} = s | X_1 = s_1, X_2 = s_2, \dots, X_n = s_n) = \Pr(X_{n+1} = s | X_n = s_n) \quad (3.1)$$

This property states that the probability of going from one state  $s_n$  at time  $n$  to another  $s_{n+1}$  at time  $n + 1$  is dependent on only the previous state  $s_n$  and not the history  $s = (s_1, s_2, \dots, s_n)$ . Further, if the transitional probability from a state  $y \in S$  to  $x \in S$  is independent of  $n$ , then the MC is said to be *stationary*. These stationary markov chains can be described using a single matrix  $P$  whose entries correspond to transitional probabilities, i.e,  $P_{ij} = \Pr(X_2 = j | X_1 = i)$ .

Markov Chains have been used to model the process of music composition by capturing information about transitions between note events in order to generate random pieces with similar structures. The states of the model can be defined using the pitch class values, i.e,  $S = (C, C\#, D, \dots, B)$ . The transitional probabilities of a piece of music would then be determined by calculating the frequencies between various pitch transitions. A new piece is then composed by performing a *random walk* using the probabilities from the transition matrix.

As an example, the melody *Mary had a Little Lamb* is represented by the note sequence  $M = \{E, D, C, D, E, E, E, D, D, D, E, E, E, E, D, C, D, E, E, E, E, D, D, E, D, C\}$ . This melody only has 3 distinct pitches so it is sufficient to describe the state space with just  $S = (C, D, E)$ . An MC built with this melody would then be represented by the following transition matrix.

$$P = \begin{matrix} & \begin{matrix} C & D & E \end{matrix} \\ \begin{matrix} C \\ D \\ E \end{matrix} & \begin{pmatrix} 0 & 1 & 0 \\ \frac{3}{11} & \frac{4}{11} & \frac{4}{11} \\ 0 & \frac{5}{13} & \frac{8}{13} \end{pmatrix} \end{matrix} \quad (3.2)$$

A caveat to this approach is the fact that most, if not all, music does not possess the Markov Property. The structure of note transitions is highly dependant on the larger scale structure of music as notes play different roles depending on whether they are part of a chord progression, a cadence or the climactic note of phrase. One way to address this issue is to increase the *order* of the MC. The term order refers loosely to the amount of memory an MC has. In an  $m$ th order MC, the probability distribution for the next state depends on the previous  $m$  states. If  $m = 1$  we have the MC defined above. Higher order MCs can be described in a similar fashion to first order MCs by increasing the state-space.

In the previous example, in order to use a second order MC, one would use the state space  $S' = (CC, CD, CE, DC, DD, DE, EC, ED, EE)$ . As such, given an original state space of size  $|S|$ , an  $n$ th order MC would have a refined state space of size  $|S|^n$ . With higher order MCs, sampled musical pieces will tend to exhibit larger scale structures such as concrete phrases or themes.

Another issue with modeling music composition with MCs is one that has to do with the state-representation. Using the states as defined above would only be sufficient to model *monophonic* music, music with just one voice. In order to model polyphonic music or harmonies, one needs a different representation. One possible approach would be to use sets of notes as the different states, i.e, one state  $s$  could be given by  $s = \{F, A, C\}$  to represent that a F major triad was played. If the state space  $S$  in such an approach would be limited to a maximum of three pitches per state and only using pitch classes, then  $|S| = 12^3 + 12^2 + 12 = 1884$  different states. Creating a transition matrix from a piece of music using

such a large state space would result in a very sparse matrix. Sampled music from such a model would thus be very similar if not identical to the original piece. Therefore, in determining a suitable representation, it is necessary to extract features that sufficiently describe the music from a melodic and harmonic point of view. This problem remains an open question, although some attempts have been made to address the issue. In the work of Conklin (**Undefined reference**) a *vertical viewpoint method* is described. For the purposes of the domain of first species counter point this method would entail representing two pairs of simultaneous notes using three values: two values for the melodic interval between the previous notes and a value for the harmonic interval between the two notes.

$$IV(a, b) = |b - a| \quad (3.3)$$

In the formula above,  $a$  and  $b$  represent the (MIDI) pitch values of the notes and thus the interval just represents the amount of semitones between two notes. Using such an approach one could model, for example, which types of intervals typically follow a perfect fifth. Obviously, relying solely on this representation would result in a loss of information about the absolute pitches used in the piece.

Markov Chains are used in this thesis in order to influence the playout policy of the Monte Carlo search. Since composing music is such a complex problem, guiding the search towards transitions that have been sampled from existing works is expected to increase the efficiency of the search algorithm.

### 3.3 Genetic Algorithms

Genetic Algorithms (GA) are a class a meta-heuristic search algorithms used for many different types of optimization problems. The main idea behind these algorithms is that they employ a search strategy similar to that of biological evolution and natural selection. This chapter firstly introduces the theory behind GAs and consequently describes how they have been used within the domain of counterpoint generation.

Being an algorithm inspired by biological phenomena, the theory of GAs has adopted terms from the field of genetics. When defining a GA one first needs to specify what the *genotype* and *phenotype* are. Within biology the genotype refers to the genetic makeup which effectively *encodes* a physical trait, or phenotype. In the case of GA, the distinction is analogous. The genotype is an effective encoding of what the algorithm is optimizing. For the problem of counterpoint generation, one could view the genotype as a sequence of pitches, which represents the phenotype, a phrase of music. Therefore, given a genotype representation, the space of possible genotypes is the search space. An objective or fitness function is a function that takes a genotype and outputs a genotype's fitness, i.e, a measure of how good it is. The idea is thus to find a genotype that maximises the objective function. Genetic Algorithms attempt to find these by using the following procedure.

- **Initialization**

A set of  $N$  random genotypes are generated, representing the current *population*. This population represents the first *generation* of candidates.

- **Evaluation**

The current population is evaluated according to the objective function.

- **Selection**

A selection procedure is employed in order to select  $k$  individuals from the current population. Typically the selection procedure randomly selects an individual proportional to their fitness value. Another possible method is to just select the  $k$  best individuals.

- **Cross-over**

$N$  new individuals are created by recombining the genotypes of two random individuals selected from the  $k$  best. The type of recombination is highly domain dependant, although the idea is to be able to retain and combine the information of the two individuals so that the new genotype has an equal or better fitness.

- **Mutation**

For each of the  $N$  new individuals, their genotype is mutated by some probability  $\alpha$ . This mutation

usually entails changing a small part of the genotype. For example, if a binary representation is used, mutation would change the value of a bit with probability  $\alpha$ . Mutations are used in order to try to prevent the algorithm from prematurely converging on local maxima(**Undefined reference**). Higher values of  $\alpha$  will result in more exploration of the search space although it will take longer for the problem to converge at a optimal solution (**Undefined reference**).

- **Repeat**

The algorithm returns to the evaluation phase and the process continues until some stopping criteria is reached. Again, this criteria is highly domain dependant. As an example, one could stop searching when there is no change between the maximum fitness value of two generations.

Genetic Algorithms have seen a number of sucesses in various applications. In particular, they have become a popular go to method for various problems algorithmic composition ().

**TODO: add more information about literature of GA successes within the domain of music composition.**

In order to determine how well the Monte Carlo based approach fares within the domain, it was tested against a Genetic Algorithms within the domain of first species counterpoint generation. To this end a Genetic Algorithm similar to ones found in the literature was defined. The specifics of the algorithm are found in a subsequent chapter.



## Chapter 4

### Related work



## Chapter 5

# Experiments





## Chapter 6

# Results



## Chapter 7

## Conclusions



# References

- Acevedo, Andres Garay (2004). Fugue composition with counterpoint melody generation using genetic algorithms. *Computer Music Modeling and Retrieval*, pp. 96–106. Springer. [-]
- Browne, Cameron B, Powley, Edward, Whitehouse, Daniel, Lucas, Simon M, Cowling, Peter I, Rohlfshagen, Philipp, Tavener, Stephen, Perez, Diego, Samothrakis, Spyridon, and Colton, Simon (2012). A survey of monte carlo tree search methods. *Computational Intelligence and AI in Games, IEEE Transactions on*, Vol. 4, No. 1, pp. 1–43. [-]
- Bruhn, Siglind (1993). *JS Bach's well-tempered clavier: In-depth analysis and interpretation*, Vol. 4. Siglind Bruhn. [-]
- Cazenave, Tristan (2009). Nested Monte-Carlo Search. *IJCAI*, Vol. 9, pp. 456–461. [-]
- Chen, Chun-Chi J and Miikkulainen, Risto (2001). Creating melodies with evolving recurrent neural networks. *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*, Vol. 3, pp. 2241–2246, IEEE. [-]
- Collins, Tom, Laney, Robin, Willis, Alistair, and Garthwaite, Paul H (2011). Chopin, mazurkas and Markov. *Significance*, Vol. 8, No. 4, pp. 154–159. [-]
- Fernández, Jose D and Vico, Francisco (2013). AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, pp. 513–582. [-]
- Fux, Johann Joseph and Mann, Alfred (1944). *The Study of Counterpoint*. JSTOR. [-]
- Hofstadter, Douglas R. (1979). *Godel, Escher, Bach: An Eternal Golden Braid*. Basic Books, Inc., New York, NY, USA. ISBN 0465026850. [-]
- Phon-Amnuaisuk, Somnuk (2009). Generating tonal counterpoint using reinforcement learning. *Neural Information Processing*, pp. 580–589, Springer. [-]
- Shottstaedt, William (1989). Automatic counterpoint. *Current directions in computer music research*, pp. 199–214, MIT Press. [-]
- Togelius, Julian, Yannakakis, Georgios N, Stanley, Kenneth O, and Browne, Cameron (2011). Search-based procedural content generation: A taxonomy and survey. *Computational Intelligence and AI in Games, IEEE Transactions on*, Vol. 3, No. 3, pp. 172–186. [-]
- Tojo, Satoshi, Hirata, Keiji, and Hamanaka, Masatoshi (2013). Computational Reconstruction of Cognitive Music Theory. *New Generation Computing*, Vol. 31, No. 2, pp. 89–113. [-]



# Appendix A

## Algorithms





## Appendix B

### Detailed results



# Samenvatting

