Sondre Søberg

Lyric-based melody generation

Making use of multi-objective optimisation

Master's thesis in Informatics
Supervisor: Björn Gambäck
June 2020

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Sondre Søberg

Lyric-based melody generation:

Making use of multi-objective optimisation

Master's thesis in Informatics
Supervisor: Björn Gambäck
Spring 2020

Department of Computer Science
Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology

Abstract

The work presented in this master's thesis revolves around the field of computational music generation, which is a prominent sub-field of Computational Creativity. The main motivation for the research conducted throughout this project has been to discover various computational solutions for music generation, with an emphasis on evolutionary approaches, and as such provide contributions within the field of Computational Creativity. The most substantial part of the work, presented in this thesis, is the implementation of a system capable of generating melodies based on the emotional expression and structure of input lyrics. The core element of said system is a multi-objective evolutionary algorithm that handles the generation of melodies, also making use of sentiment analysis and syllabification algorithms for the input lyrics. The implementation was quantitatively evaluated through a questionnaire, and the most interesting finding was that the system was deemed useful as an assisting resource for music composition. The system output can also be considered encouraging, as the aggregated questionnaire scores suggested that the quality of the generated melodies was on the favourable side of neutral.

The main contributions of this master's thesis are the implemented system, as well as explorations within evolutionary concepts of computational music generation, such as the testing of various musically grounded fitness approaches. The insights gathered from the questionnaire can also be deemed a valuable contribution, as it could provide clarity of people's expectations regarding the behaviour and quality of automatically generated music.

Sammendrag

Arbeidet som blir presentert i denne masteroppgaven dreier seg hovedsakelig om datamaskinstyrt musikkgenerering, som er en fremtredende gren innenfor datamaskinstyrt kreativitet. Den viktigste motiverende faktoren for å gjennomføre dette forskningsprosjektet har vært å oppdage muligheter for datamaskinstyrt generering av musikk, med et hovedfokus på evolusjonære løsninger, og på denne måten bidra til videreutviklingen av feltet datamaskinstyrt kreativitet. Den viktigste delen av det gjennomførte arbeidet i forbindelse med denne masteroppgaven er implementasjonen av et system som er i stand til å automatisk generere melodier basert på oppbyggingen av gitte inputtekster, samt tekstenes følelsesmessige kvaliteter. Hoveddelen av systemet er en evolusjonær algoritme med fokus på å optimere flere delmål. Denne algoritmen håndterer selve melodigenereringen, mens det i tillegg er brukt andre algoritmer til sentimentanalyse og detektering av stavelser i inputtekster. Systemimplementasjonen ble kvantitativt evaluert gjennom en spørreundersøkelse, og det viktigste funnet fra denne var at systemet ble ansett som potensielt nyttig som en assisterende ressurs ved komponering av musikk. Kvaliteten til systemets genererte melodier kan også anses som lovende, ettersom de aggregerte resultatene fra spørreundersøkelsen antyder at de evaluerte melodiene har en kvalitet noe over middels.

De viktigste forskningsbidragene til denne masteroppgaven er det implementerte systemet, i tillegg til utforskning av flere evolusjonære konsepter innen datamaskinstyrt musikkgenerering, slik som testing av diverse fitnessfunksjoner av musikalsk art. Viktige funn fra spørreundersøkelsen kan også regnes som viktige bidrag, ettersom de kan hjelpe med å kartlegge folks forventninger til kvaliteten på automatisk generert musikk.

Preface

This master's thesis has been written as the main part of completing a two-year master programme in Informatics, with the specialisation interaction design, game and learning technology, at the Norwegian University of Science and Technology (NTNU). The thesis work was conducted at the Department of Computer Science, NTNU, Trondheim, and the supervisor for the project was Björn Gambäck.

I would especially like to thank my supervisor Björn Gambäck for his impeccable guidance, hours of discussion and valuable feedback provided throughout the course of this project. I would also like to thank Marinos Koutsomichalis for his feedback during two joint meetings with Björn Gambäck's master's students within fields of computational creativity. In addition I would very much like to thank all participants of the questionnaire for providing answers valuable to the research.

Sondre Søberg
Trondheim, 1st June 2020

Contents

List of Figures

List of Tables

1. Introduction

This master's thesis presents research conducted within the field of Computational Creativity, namely by doing an in-depth study of automatic music composition, with an emphasis on multi-objective evolutionary algorithm approaches. The thesis provides a literature review of state-of-the-art solutions for automatic music generation, as well as the development and implementation of a system that generates melodies based on input lyrics, with a core focus on having the melody be fitting of the lyrics. The research includes a questionnaire for evaluating the system output, as well as a review of different architectural solutions for the evolutionary design and sentiment analysis.

The following chapter is intended to give the reader a clear and thorough overview of the project scope of this thesis, as well as introduce the fundamental goal of the project, give a declaration of the main motivating factors resulting in the given thesis, and present the research questions and contributions.

1.1. Background and Motivation

The field of Computational Creativity is highly viable and continuously evolving. It combines Computer Science, mainly through the use of Artificial Intelligence technologies, with different understandings of creativity, aiming to develop software that can emulate human creativity. This includes systems concerned with generating inherently creative ideas, as well as systems intended to understand and evaluate creativity. Possibilities in the development of creative software include all branches of creativity, such as art, music and poetry, and takes many forms, such as autonomous algorithms for generating stand-alone creative pieces of art, or supportive programs functioning as collaboration tools for the human creative process (Colton et al., 2009).

Automatic music generation is a sub-topic within Computational Creativity, that has been researched for several decades, and numerous solutions have been presented to solve a variety of sub-problems (Nierhaus, 2009; Rader, 1974), where the first fully computer generated composition was famously done by Hiller Jr and Isaacson (1957), when they generated their "Illiac Suite".

As music generation is a highly relevant topic of Computational Creativity, filled with endless possibilities, only limited by creative imagination and computer software, this thesis focuses on research within the topic of computational music generation. The main goal of the thesis has been to develop a system for generating harmonising melodies, based on lyrical input, having a strong focus on applying a correct sentiment value fitting both the lyrical input and the generated melodies. The quality of the system was then to be evaluated by human users.

With a basis in these topics, the main motivation of this thesis has been to contribute to the field of Computational Creativity through exploring different possibilities of music generation, as well as developing a system that could be used as an idea generator for musicians and songwriters, and as such be a collaboration tool for human creativity.

1.2. Goals and Research Questions

Goal *To provide a novel solution for automatically generating melodies fitting the structure and sentiment of user defined input lyrics*

Research question 1 - *(RQ1)* *By using an evolutionary algorithm approach, in what ways could a melody be automatically generated, as to fit given lyrics, as well as capture the lyrics' sentiment?*

To provide a satisfying answer to *RQ1*, it is important to conduct a study of state-of-the-art techniques within music generation and sentiment analysis. To be able to capture the sentiment of given lyrics and translate it to a melody, key musical components associated with given sentiments must be asserted. For *RQ1* to be relevant to the main goal, key elements from the study must be analysed and considered, when developing the algorithm concerned with melody generation. As the research question specifies an evolutionary algorithm approach, it is necessary to determine the most prosperous evolutionary approaches to music generation.

Research question 2 - *(RQ2)* *How good will the quality of the melodies generated by the system prove to be, when judged in a human context?*

Evaluation is an invaluable aspect of Computational Creativity, as large scale testing and questioning can be highly valuable in determining the quality of generated creativity. To answer *RQ2* it is necessary to establish how a set of humans evaluates the generated results from the separate parts of the system, such as results from sentiment analysis, syllabification and the generated melodies. A main component within this research question is also to determine to what extent the system could be of assistance in human songwriting and composition tasks.

1.3. Research Method

The main research methods adopted in this project could be said to be both exploratory and empirical in nature. The initial research into state-of-the-art methods of computational music generation, covered in the literature review of chapter 3, could be considered exploratory research.

Experiments conducted in the system development phase, when making architectural decisions, could be considered a combination of exploratory and empirical research, as testing was conducted on multiple setups, where selected experiments were conducted

as part of exploring previous solutions and problems. The architectural decisions that were made based on the test results can be described as empirical decisions.

The main evaluation criteria for establishing to what degree the project goal has been successfully reached are found through empirical research. To accurately evaluate the finished system implementation, the quality of a set of system generated melodies was quantitatively evaluated through a questionnaire. Human judges were set to evaluate the generated melodies based on specific evaluation criteria, described in chapter 5.

1.4. Contributions

This section serves as a brief overview of the main contributions of this master's thesis. A further description of the contributions is presented in section 7.2.

1. *An implemented system for lyric-based melody generation, following a multi-objective evolutionary approach, that can serve as an assisting tool in human composition*

2. *A review of interesting approaches to evolutionary music generation*

3. *The results gathered from experiments conducted on different architectural approaches in the development phase*

4. *The results gathered from a questionnaire regarding the quality of melodies generated by the system*

1.5. Thesis Structure

The structure of this thesis follows a division into 7 chapters. In addition, the main matter is succeeded by a bibliography, presenting all cited literature throughout the thesis, and an appendix providing valuable extra information regarding the melodies generated as part of this project, as well as additional questionnaire information and results.

Chapter 1 provides a core overview of the background, motivation and the main goals of the master's thesis.

Chapter 2 is intended to give a thorough overview of the most important background information necessary to understand key concepts of the system design and conducted research throughout the project.

Chapter 3 presents a literature review of state-of-the-art approaches within computational music generation and lyrical sentiment analysis.

Chapter 4 presents the system architecture of the designed and implemented system for lyric based melody generation, as part of this project.

Chapter 5 presents the experiments conducted on the system implementation presented in the previous chapter and the results gathered from the experiments.

Chapter 6 presents a thorough evaluation of the results gathered from the experiments and the questionnaire, as well as a discussion of the degree of accomplishment regarding the research questions and main goal of the thesis.

Chapter 7 is intended to conclude the results in light of the discussion presented in the previous chapter, as well as give an overview of the main contributions of the research and provide an insight into possible future work.

2. Background Theory

This chapter is intended to familiarise the reader with the most important terms and concepts adopted throughout this master's thesis. Firstly general musical concepts are covered in section 2.1, then theories in evolutionary algorithms, as well as the main algorithm used as a baseline for the system architecture in chapter 4, is presented in section 2.2. The following section 2.3 covers the most important concepts within natural language processing. Lastly an overview of the main developmental resources is presented in section 2.4.

2.1. Musical Theory

As to have a useful point of reference for the evolutionary algorithm and melody generation, it is essential to have a basic understanding of key principles in music theory. The music theory will be presented with a main focus on conventions in western music. The main sources of information when authoring this section was the chapter "Introduction to Music Transcription" from the book *Signal Processing Methods for Music Transcription* (Klapuri, 2006), and the e-book *The Joy of Harmony and Composition* (Tobey, 2012).

Pitches and Notes A musical *pitch* is a reference point for a perceptual attribute, that places a sound on a frequency scale, determined by the sine wave of the sound's fundamental frequency. *Notes* are pitches with predefined values for their fundamental frequencies, e.g. the frequency of the note A4 is set to 440Hz, and two notes in the same scale have a set relationship between them, e.g. calculating the frequency of a corresponding note an octave higher simply implies doubling the frequency (A4 = 440Hz, A5 = 880Hz). This does not imply that all instruments have discrete pitches, i.e. strictly follow note values, and in the use of many instruments, as well as singing, pitches between notes must also be considered as having musical value.

Scale In western music notation, notes are arranged in different scales within an octave range, i.e. 12 notes (including sharps/flats). The main notes range from A to G-sharp/A-flat, with all notes, except E/F and B/C, having a sharp/flat note between them. The special marking of notes as either flat (♭) or sharp (♯), varies depending on musical *key* and *scale*. The distance between a note and its immediate neighbour (possibly sharp/flat) is of a semitone distance, which is the smallest note distance in regular notation. Different scales are usually a sub-set of the 12 possible notes

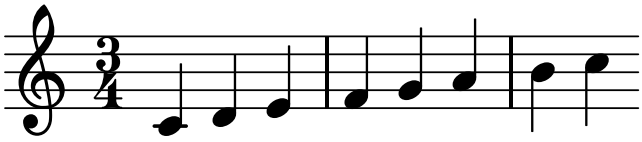

Figure 2.1.: A C scale with a 3/4 time signature. Generated using LilyPond (described in subsection 2.4.4)

of an octave, e.g. in a regular C major scale, the term octave is used to define the 8 notes between and including C to C. The C major scale is often used as a reference point for a scale, as it employs no use of sharp/flat semitones. Figure 2.1 shows a C scale in regular western musical notation. Scales are always looked at with regards to their starting note, i.e. their tonic note. The scales applied in this thesis are the *major scale*, the *natural minor scale*, and the *harmonic minor scale*. These scales all include 8 notes (when the tonic is counted twice), and the naming of the notes, based on their position in the scale, follows the naming convention in Table 2.1. The naming convention is derived from Tobey (2012).

Table 2.1.: Scale degrees and naming convention

Scale positions and naming						
1st	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**
Tonic	Supertonic	Mediant	Subdominant	Dominant	Submediant	Leading tone

Key The concept of musical keys extends the description of scales in the sense that a musical *key* is a rule set that defines which notes a specific scale should include. This depends on the tonic note, which is the fundamental note the scale is based upon, and whether the scale is fundamentally major or minor.

Rhythm and time signature A simple *rhythm* consists of relationships between time-stamps for consecutive pitches (inter-onset intervals) and their duration. A rhythm is generated by establishing the duration of a *beat*, i.e. periodic pulses in a given time-space, which are often counted in beats per minute (*bpm*) and by deciding upon the musical *measures* of a piece. The number of beats between two bar lines establishes the musical measure and is determined by the top number (divisor) of the fractions found in musical notation. Figure 2.1 shows the C major scale, divided into measures, where the measure is 3 beats to a measure. The lower number (dividend) specifies the length of each beat in fractions of a whole note (𝅝), i.e. if the number is 4, each beat is a quarter note (♩) long ($\frac{1}{4}$ of a whole note), and because this number is 4 in Figure 2.1, there is room for 3 quarter notes in each of its measures. The most commonly used time signature is $\frac{4}{4}$, which in some cases, such as in *LilyPond* (see subsection 2.4.4) notation, is denoted by a *C*, rather than the actual time signature. The *C* specifies that the time signature is common time.

The most basic method of defining a note's duration is by defining its duration as a fraction relative to a whole note, most often by having a dividend being a power of 2. In addition, more complicated duration markers may be applied, when needed. *Dotted notes*, for example, are notes with an appended dot (.), that have a duration equal to the un-dotted note duration multiplied by 1.5. *Syncopation* is a rhythmical phenomena that occurs when durations are introduced, which moves the music off-beat, e.g. when notes with durations shorter than the size of the beat are introduced and as such syncopates the melody, or when a note's duration is carried over to a succeeding measure/beat.

Intervals An interval is comprised of two neighbouring notes in a melody and describes the distance between them.

Consonance & Dissonance Consonance is a term used to describe note distances that sound pleasant to the human ear. It could be consonance between two simultaneously played notes or between notes in an interval. Dissonance is the somewhat opposite of consonance, meaning that the note distances sound less pleasant, and could easily be noticed within a melody.

Melody A melody is, in its most basic form, comprised of a series of pitches (notes), that have a musical value, and follow a rhythmical pattern with inter-onset relationships, as described in the previous paragraph. A melody may often contain musical phrases, which are shorter repeatable note patterns, that might help give the melody character. In a song or composition it might be hard to define where one melody ends and another one starts, as this is up to interpretation, but melodic patterns often repeat themselves throughout a piece.

Structure The term *structure* aims to explain a musical piece from the highest point in the hierarchy, i.e. divide it into large sub-pieces, such as verse, chorus, bridge, etc. The different sub-pieces are identified by studying differences in repeating patterns in the music, and when generating new pieces of music, depending on the musical genre, it is important to keep song structuring in mind.

Melisma In cases where notes are aligned to lyrics, in e.g. a song or a rhyme, *melismas*[1] are groups of notes that are sung/played over a single syllable. Unless marked as a melisma, only one note is generally mapped to each syllable in the lyrics. In notation a melisma is added either by spreading a syllable over multiple notes using slurs, which are curved lines connecting the notes, or by extending the syllable in the lyrics with continuous underscores.

Staffs and Clefs A staff is a common, five lined representation of music. Notes are placed horizontally with respect to time, i.e. notes are sequentially placed from left to right on the staff, with their vertical placing specifying the note's pitch. The clef is the first notation in a staff and specifies how notes are to be placed on the

[1] `https://www.dictionary.com/browse/melisma`

five lines, with regards to their pitch. The clefs used in this project is the *treble clef* for melody, and the *bass clef* for chord notation. For the treble clef, the first note to touch the lower staff line is *D4* (*d'* in LilyPond notation), i.e. the second note from the left in Figure 2.1. For the bass clef, the first note to touch the lower staff line is *F2* (*f,* in LilyPond notation). An example of bass clef usage can be seen in Figure 4.3.

Table 2.2.: Chord triad types

Type	**Example**	**Description**	**Integer distances**
Upper-case letter	A	Signifies a major triad	{0, 4, 7}
Upper-case letter with added plus	A+	Signifies an augmented major triad	{0, 4, 8}
Lover-case letter	a	Signifies a minor triad	{0, 3, 7}
Lower-case letter with added degree sign	a°	Signifies a diminished minor triad	{0, 3, 6}

Chords and triads When any number of notes are played at the same time, they constitute a chord, i.e. a minimal chord is two notes played simultaneously. A triad is a three-note chord, usually following a specified set of rules. The main part of which a chord is made up of is a root (tonic) note, and notes revolving around said note. The musical chord notations considered in this thesis are *major triads*, *augmented major triads*, *minor triads* and *diminished minor triads*. The notation, and examples are presented in Table 2.2. The main source for the notation is Tobey (2012). It is worth noting that the integer notation follows absolute note distances, and not scale distances. In other words the integer distances describe the semitone distance from a chord's root note in relation to the notes in the chord, i.e. when the first number is 0 it signifies that the first note in the chord is the root note, the 4 signifies that the next note is a major third, etc. The first note describes the lowest note in the chord, and for the sake of understanding the integer distance notation, it is important to note that the triad does not have to start on its root note. The concept behind this is called *chord inversions*.

Chords and scale degrees Scale degrees are the names of the notes in a given scale and how they are correlated. By taking major inspiration from concepts mentioned by Tobey (2012), the defined chord triad types for the given major/minor scale degrees used in thesis are presented in Table 2.3. The table follows the notation from Table 2.2, but, as is normal when describing rules for chords with a disregard for specific notes, have substituted note names with roman numerals, describing the scale degree of a chord's root note. The minor scale chord harmonisation rules are not based on the natural minor scale, but rather the harmonic minor scale. Tobey (2012) describes this as a commonly occurring concept in harmonisation.

Table 2.3.: Chord triad types in different scale degrees

Chords in different scale degrees							
	1st	2nd	3rd	4th	5th	6th	7th
Major	I	ii	iii	IV	V	vi	vii°
Minor	i	ii°	III+	iv	V	VI	vii°

2.2. Evolutionary Algorithms

Evolutionary algorithms (*EAs*) is a set of algorithms grounded by the general idea of mimicking natural evolution processes. The main concept of *EAs* is to emulate how species adapt to their environment over time, how the stronger individuals in a population are able to reproduce and as such shape the evolution of their species (Yu and Gen, 2010, p. 6).

This section gives a brief introduction of key concepts in evolutionary algorithms, with a main focus on *Genetic Algorithms (GAs)*, which are *EAs* that follow a rigorous set of rules/steps to determine the optimal solution to a problem. The optimal solution is found iteratively, by continually evaluating the individuals in a population. The main loop of a genetic algorithm is shown in Figure 2.2. The main sources applied when authoring this section were Yu and Gen (2010); Whitley (1994); Floreano and Mattiussi (2008) for the general description, and Branke et al. (2008); Deb et al. (2002) for multi-objective approaches. The need for much variety in source material spun out from a tendency in articles on *EAs* to apply diverse terms and descriptions for similar concepts.

Genotype and phenotype

The genotypes (genes) of *GAs* are the building stones of the individuals in the *GA* population. Genotypes, which can be compared to biological chromosomes, define all the information and qualities of a phenotype (individual). This information is most often represented as binary strings. Phenotypes are the individuals that exist within a given population, and as such, one can say that a population is defined by its phenotypes, which in turn is defined by their genotypes.

As the first step of a genetic algorithm an initial population must be generated. This is done by specifying the population size N, and then generate N individuals, most commonly with randomised genotypes.

Fitness and selection

As shown in Figure 2.2, the second step after a population is generated, the phenotypes (individuals) must be individually evaluated based on their fitness. The fitness of a phenotype defines the quality of the phenotype, based on its genes. This is done by applying at least one fitness function to the phenotypes.

The phenotypes of the population are then ranked by the results given by the fitness functions, and a set amount of top ranking phenotypes are then selected for reproduction

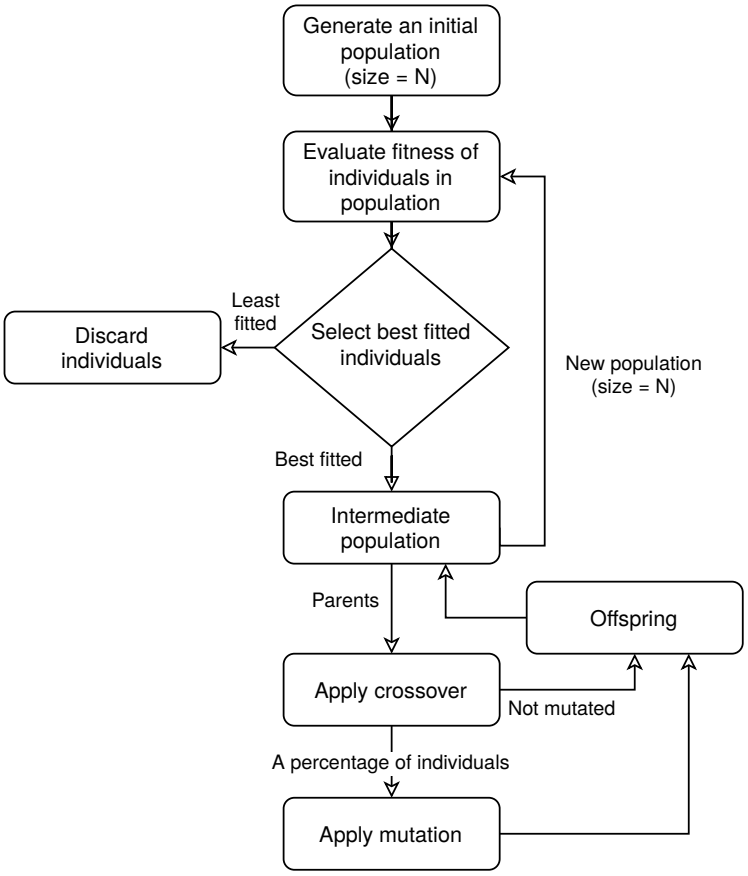

Figure 2.2.: The main loop of a general Genetic Algorithm

in the next steps. The selected phenotypes are put in an intermediate population. The least fit individuals are discarded. Offspring is generated by applying *Genetic operators* to the intermediate population until the population has a size = N. The intermediate population is then considered the new population for the next generation.

Genetic operators

The first step in creating offspring is to recombine the genotypes of two parent phenotypes, selected for breeding, into a new phenotype. This process is called crossover, and one or multiple crossover functions decide the rules of how the parent genes are combined into an offspring. Crossover is the first genetic operator applied in offspring generation. The chance of a crossover function to be applied is set beforehand. If no crossover operator is chosen in this step, the offspring will be a clone of one of its parents.

A set, but arbitrary percentage of the offspring generated by the crossover operator are chosen for mutation. Mutation is, as well as a phenomenon that happens naturally in evolution, used as a tool for avoiding local maximum solutions to problems by introducing

random mutations to some of the offspring. The chance of offspring mutating is set by manual evaluation, and is usually a low number, e.g. 1-10%.

The avoidance of *local maximum* solutions is a key purpose for genetic operators. With regards to evolutionary algorithms, a maximum is reached when the individuals in the population reach a seemingly maximal score for their fitness evaluation. The main goal for the genetic operators is to strive for the global maximum solution to be found, i.e. evolving the individuals in the population to reach the maximum fitness scores possible. A local maximum solution is found when further evolving cannot seem to improve the fitness scores, although the absolute maximum score has not been reached.

Termination

A genetic algorithm terminates either when one or more optimal phenotypes are found in the population or after a set number of generations has passed. One could also set the algorithm to terminate, if there has not been sufficient change in phenotype genes over the course of multiple generations.

2.2.1. Multi-objective evolutionary algorithms

Evolutionary multi-objective optimisation (*EMO*) is a well established field of research, that focuses on evolutionary algorithms solving problems by finding optimal solutions where the problem space has multiple objectives. This stands in contrast to basic evolutionary approaches, which consider one optimised numerical value representing the fitness of the phenotypes in a population. *EMO* takes multiple fitness evaluations, arranged as objectives, into account, and selects the optimal phenotypes by sorting them into Pareto-optimal fronts. Such fronts are determined by a domination principle, that iteratively sorts phenotypes into fronts with individuals dominating, i.e. phenotypes that are more optimal than phenotypes in succeeding fronts (Branke et al., 2008). Another main principle in sorting the individuals is finding solutions that are diverse enough to represent the entirety of each front. In addition to the main operators of evolutionary algorithms *selection, crossover and mutation EMO* approaches implement an elite preservation operator that prunes the worst solutions of a population at the end of each iteration.

2.2.2. Non-dominated sorting genetic algorithm II

The non-dominated sorting genetic algorithm II (*NSGAII*) presented by Deb et al. (2002), is a widely used genetic algorithm for solving multi-objective problems (Yu and Gen, 2010; Branke et al., 2008). The algorithm follows the principles presented in subsection 2.2.1, and the main flow of the algorithm is displayed in Figure 2.3.

After an initial population (size=2N) is generated, non-dominated sorting based on multiple objective fitness values is imposed on the population. This follows the dominating principle described in subsection 2.2.1, as the population is sorted into multiple fronts, where individuals in front 1 dominate individuals in front 2, and so on. From the

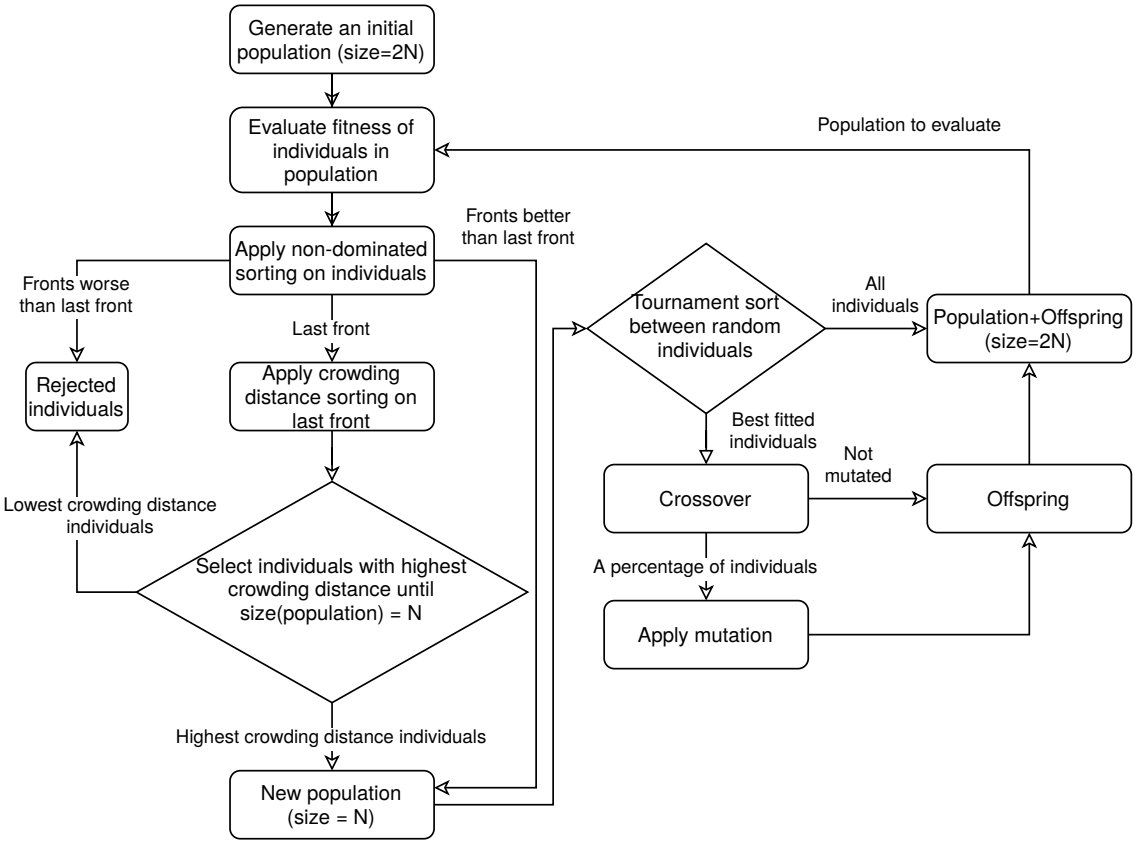

Figure 2.3.: The main loop of the NSGAII algorithm

highest ranking fronts a new population (size=N) is selected, and since the accumulated size of the fronts is 2N, fronts are chosen for the new population until the size is less than or equal to N. If the size is less than N, crowding distance sorting is imposed on the last front not implemented in the new population. The crowding distance sorting selects the individuals that have the highest distance to other individuals in the front, and iteratively includes them into the new population until the population size = N. This is done to ensure that the diversity principle described in subsection 2.2.1 is met.

If a termination criterion for the algorithm is met, the algorithm will terminate at this point, when the population size = N. Otherwise individuals from the population are chosen as parents for generating offspring. This is done by iteratively using tournament sort between two random individuals from the population, until the the offspring population is of size = N. A mutation operator function is applied to a predefined percentage of offspring. To satisfy the elitism principle mentioned in subsection 2.2.1, the offspring is combined with the parent population, such that the combined population is of size = 2N. This makes sure that elite individuals from earlier generations can survive multiple algorithm iterations (generations), and as such the best problem solutions are never lost. As the first step in generating the next generation, the newly combined population is

sent back to the top of the main loop of the algorithm for evaluation (fitness ranking). The algorithm follows this pattern iteratively until termination.

2.3. Natural Language Processing

Natural Language Processing (*NLP*) is a field of Computer Science that focuses on how computers may analyse, process and/or generate natural language. In *NLP* the main focus of research is discovering ways of increasing a computer's ability to understand and handle natural language, as well as map it to machine code. As an example, one could say that the development of database query languages and high-level programming languages both are early forms of *NLP*, as they differ from machine code in that their vocabulary is highly based on simple English language.

Natural Language Processing is also much used in linguistic research, and is an invaluable resource when it comes to tasks such as translating between natural languages (e.g. between English and Norwegian), and in researching historical language development (Matthews, 2016).

2.3.1. Syllabification

Syllabification is one of many sub-tasks in *NLP*, and is the task of identifying which syllables different words in a vocabulary is comprised of. Syllabification, in general, is an important task in linguistics, but it is also a main component of e.g. speech synthesis and speech recognition systems (Bartlett et al., 2009). Syllabification is a hard task to solve accurately, as the lettering for different *phonemes* might be the same. Phonemes are the different pronunciation sounds that exist within a language. Even though there is some debate regarding how a syllable is defined, there is a wide agreement that a syllable is constructed around a vowel sound (Bartlett et al., 2009). The vowel sound is in vocal music the main component of a syllable, over which the tone of a note is sung/held. The main computational forms of syllabification is done either by *rule-based* or *lexicon-based* methods, where rule-based systems focus on general rules as how many consonants should be applied to a vowel and how words are allowed to start/end. Lexicon-based systems presume that the word to look up exists in a lexicon, where the syllables or phonemes of a language are manually annotated.

2.3.2. Sentiment Analysis

Sentiment Analysis (*SA*) is a sub-concept of *NLP* that focuses on the computational study of human emotions towards different entities. SA may be viewed as a classification problem, where different levels of complexity exist, both in the size of language data to be classified, as well as the complexity of the language of the data (Medhat et al., 2014). As an example it would be more straightforward to classify the sentiment of a single sentence, rather than a poem, which could have a more complicated language structure, as well as sentiments which cannot explicitly be deduced from the words.

The two main approaches to sentiment analysis are considered to be machine learning approaches and lexicon-based approaches. There also exist methods combining features from both approaches. Machine learning approaches combine an understanding of linguistic features and different machine learning structures and algorithms. Lexicon-based methods employ either a dictionary or a corpus of words and word-combinations annotated with either single or contextual sentiment values (Medhat et al., 2014).

2.4. Resources

This section introduces the main external resources and libraries used in the development of the system architecture (chapter 4) of the project.

2.4.1. Natural Language Toolkit (NLTK)

The Natural Language Toolkit (*NLTK*)[2] is a widely used platform for Natural Language Processing of data in Python[3] applications. The platform is a collection of many *NLP* resources, such as corpora and lexical resources, as well as scripts for tokenising sentences. As concrete examples, this project makes use of included resources such as sentence tokenising by sonority principle, sentiment analysis and a pronunciation dictionary.

2.4.2. CMU Pronouncing Dictionary

The Carnegie Mellon University Pronouncing Dictionary (*CMUdict*)[4] is an open source pronunciation dictionary containing more than 134,000 American-English words, and their corresponding pronunciations represented by a set of 39 phonemes. Vowels in the dictionary are marked with stress-markers (0 - No stress, 1 - Primary stress, 2 - Secondary stress), that define where the stress of different words lie. *CMUdict* is easily accessible through the Natural Language Toolkit. The resource has been used both in syllabification tasks and stress identification in this project.

2.4.3. Vader

Valence Aware Dictionary and sEntiment Reasoner (*VADER*) is a sentiment analysis tool, mainly designed to determine the sentiment of social media posts, presented by Hutto and Gilbert (2014). The tool makes use of rules and a comprehensive lexicon in its sentiment analysis. Vader is continuously updated, and has been incorporated into the Natural Language Toolkit. In spite of being tuned to a social media domain, Vader performs well in multiple domains (Hutto and Gilbert, 2014), and because it is readily available and in no need of training data, it has been a valuable resource throughout this project.

[2] `https://www.nltk.org/`
[3] `https://www.python.org/`
[4] `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`

The lexicon incorporated by Vader was built from the results of evaluations by ten human raters. The raters used values ranging from *Extremely Negative* to *Extremely Positive* to rate 9000 token features, where 7500 were later kept as lexical features, tagged as positive/negative on an intensity scale ranging from -4 to 4. In combination with the lexical approach, Vader follows five rules, based on word-order sensitive relationships between terms. The first three rules determine word intensity based on *punctuation* (e.g. exclamation points), *capitalisation* (e.g. in all-caps words) and *degree modifiers* (e.g. adjective detection and handling). The last two rules handle negation and contradiction, such as sentences having a "but" that neutralises the sentence or an "isn't" negating token values.

Vader scores sentences and texts based on the lexicon and the rules, and summarises the word values it considers negative, neutral or positive into an accumulated *compound* value, which is normalised between -1 and 1. The suggested thresholds for the compound value and the general sentiment of a sentence is that $compound \geq 0.05$ implies a positive sentiment and that $compound \leq -0.05$ implies a negative sentiment. Otherwise the sentiment is considered neutral.

2.4.4. LilyPond

LilyPond[5] is a music engraving language for computational creation of sheet music, with the possibility to compile the code into multiple file formats, including PDF, SVG and MIDI. LilyPond follows a classical musical engraving method, but regardless of this, it is highly customisable. The output files of the project system are generated as LilyPond-files, and then compiled and cleaned using the open source graphical user-interface Frescobaldi[6].

All sheet music presented throughout this thesis is generated through the use of LilyPond and Frescobaldi.

[5] `http://lilypond.org/`
[6] `https://frescobaldi.org/`

3. Related Work

This chapter is intended to give the reader an overview of related work and common state-of-the-art approaches within the topic of music in computational creativity. Section 3.1 is intended to give a general overview of the most common state-of-the-art approaches within computational music generation. Section 3.1.3 explores state-of-the-art methods in combining textual and musical concepts in generating music.

3.1. Computational music generation

3.1.1. Evolutionary Algorithms

For musical generation, there has been a significant number of solutions applying different approaches to evolutionary algorithms. One of the most notable approaches can be found in the works of John Biles' GenJam project, which he has been continuously developing over the course of close to three decades (Biles, 1994, 2013). He found that an evolutionary approach, described thoroughly in section 2.2, would be highly suitable for musical generation, given its natural ability to search through strange problem spaces and having few requirements of use (Biles, 1994). While developing the first GenJam system, Biles found that generating suitable fitness functions for musical genetic algorithms was a major bottleneck, and after failing to implement an automatic fitness function, he decided to use a human fitness approach. He would then use himself as a mentor for the system, and for each new evolution of the population he would rate the solutions with either "good" or "bad" (Biles, 1994).

Biles continued working on eliminating the fitness bottleneck, and after failing in his attempt to develop neural-network-based fitness functions, he succeeded in creating a fitness free *AutoGenJam* (Biles, 2001). He achieved this goal by having an initial population that was randomly gathered from a database of licks, i.e. musical phrases, from human performances, implementing an intelligent crossover operator for breeding and functionality for mutating repeated licks to ensure freshness. He argued that AutoGenJam should still be considered a genitive algorithm, even though it did not follow the norm of using fitness functions (Biles, 2001). As for the state-of-the-art version of GenJam, both of the previous solutions can be used to interactively improvise jazz music along with a human counterpart (Biles, 2013).

The concept of fitness in evolutionary art and music is a heavily researched topic, so much in fact, that Johnson (2012) attempted to classify implementations of fitness functions into a taxonomy of fitness scope and fitness basis by conducting a survey considering a substantial number of previous papers in the area. Johnson (2012) also

suggests future possibilities for bases of fitness functions that have been little considered in previous works in evolutionary art and music. Their results suggest that even though *Human Interaction* is a popular approach for fitness bases, the majority of the approaches to fitness bases are autonomous. As to circumvent the problem of the fitness bottleneck, autonomous fitness functions are often grounded in music theory, such as in Wu et al. (2014), or heuristic rules or machine learning techniques, such as in (Jeong et al., 2017). Some approaches still involve some human interaction, such as the *Corpus or Example* fitness basis. Biles is an advocate of using human interaction as fitness basis. Manaris et al. (2003) used this technique when evaluating the possibilities of using the *Zipf-Mandelbrot law*, which defines a concept of how the occurrence of "words" in various natural phenomena can be described by a probabilistic model of rapidly decreasing frequencies, on a 220-piece corpus of various music styles to generate fitness functions for music generation. Bell (2011) resorted to human interaction as the fitness function, and used a set of *Markov Chains* (Gagniuc, 2017) to represent genomes, where the Markov chains represented the probability of a chord change based on a previous chord. The findings by Johnson (2012) also suggest that Biles' approach of completely removing fitness functions has not, to any broad extent, been adopted by state-of-the-art research.

In recent years one of the most favoured techniques in evolutionary music generation has been to implement multi-objective optimisation in the algorithms, which is a concept of obtaining the optimal solution of a problem, when multiple fitness objectives are simultaneously optimised, and it as a consequence exists a set of equally qualified solutions and trade-offs (Abraham and Jain, 2005). Jeong and Ahn (2015) developed a multi-objective generative algorithm, using two fitness functions to deal with a trade-off between tension and stability in a melody, given some chord progression. To rank the solutions, an implementation of the *Non-dominated Sorting Genetic Algorithm (NSGA-II)*, described in subsection 2.2.2 was used, as the approach is stated as a known standard for multi-objective optimisation problems. The work is further described and extended by Jeong et al. (2017). Their work served as a baseline for the system implemented in the Master's Thesis of Olseng (2016). In his thesis Olseng presents an extensive solution for co-evolving melodies and harmonies, incorporating a sum of 43 different fitness measures. The thesis suggests using a four-objective approach, dividing generational sub-tasks for the melodies and harmonisation into four separate parts to be simultaneously optimised. His results, supported by a quantitative evaluation, suggest that the approach could not consistently generate well perceived melodies, and that they were too rhythmically imperfect. His implementation did though indicate some promise, as it presented the possibility for co-generation of melodies and harmonisation, as well as having produced some musical pieces, that were favourably evaluated. In addition to the presentation found Olseng's thesis, a rundown version of the system was presented by Olseng and Gambäck (2018).

To increase the quality of the result, it is not uncommon to adopt other techniques, besides modifying fitness functions, in algorithms for evolutionary music generation. Khalifa and Foster (2006) present a two-stage solution, i.e. a sequential generation process, where the algorithm initially generates motifs, and in stage II generates phrases based on

the given motifs. Kramann (2013) proposed to implement a competitive concept in evolutionary music generation, by having two separate evolutionary algorithms cooperate in evolving a piece of piano music, generated over a repeating pattern.

3.1.2. Artificial Neural Networks

The use of *Artificial Neural Networks* (ANNs), which is a technology in machine learning concerned with computationally emulating the neural networks found in living organisms, in regards to computational generation of music, has been a relevant topic in research since the late 1980s, when Todd (1989) implemented a system for generating simple melodies using an early model of *Recurrent Neural Networks* (RNNs). The problem of music being a temporal concept, i.e. the succession of notes in a melody changes with the passing of time, was solved by implementing a sequential network model, where the network had a memory of the previously generated parts of the melody (context), and updated its melody generation plan continuously based on the previous notes. The system only possessed the ability to generate melodies with a high similarity to the limited amount of melodies used for its training.

In recent years the usage of ANNs in music generation has had an upswing, as newer models and increased computer performance has made ANNs increasingly suitable for generating more complex music. The networks used usually follow an approach implementing RNNs, as they have proved more fitting, when considering the temporal nature of music and their long-term dependencies (Colombo et al., 2016; Liang et al., 2017; Castro, 2019).

Colombo et al. (2017) implement two RNNs, using *Gated Recurrent Units* (GRUs) as memory cells, in a system for generating melodies, extending their earlier work (Colombo et al., 2016). Here one network is trained to concern itself with the probability distribution of upcoming durations and one with finding the probability distribution of upcoming pitches. Both networks draw their conclusions based on current durations and notes, as well as their respective internal states. The model was trained using a corpus of melodies from two different genres, where fixed partitions of 80% and 20% of the corpus were used as a training set and a validation set, respectively. During each training epoch, the model was evaluated using the validation set. Melodies were composed by sampling from the networks. First a duration was sampled, and then used as an additional input to the pitch network. The previously sampled notes were used as an input for the networks. Colombo et al. found that their model produced melodies that were close to melodies in their data set, and that even though the data set contained two distinct musical styles, the generated melodies were consistent with one of the styles at a time, i.e. the melodies were not a merger of both styles.

Liang et al. (2017) took the idea of using RNNs in music generation one step further, and developed a model that generated harmonising melodies with Bach's chorales as the data set for training. As opposed to Colombo et al. (2017), Liang et al. (2017) used *Long-Short-Term Memory* (LSTM) cells instead of GRUs as their memory cells. In the article by Chung et al. (2014), they found that LSTMs and GRUs are much preferable to older recurrent units, but they found no conclusive answers regarding which of them

was the superior one.

The problem of processing time for deep learning models was explored by Castro (2019), where a system that could improvise a rhythm and melody in a "call and response", fashion along with a human counterpart was developed and explored, i.e. when provided with a melody, it could respond with a fitting continuation. The system relied on two RNN models with LSTM memory cells, previously developed by Google Magenta[1]. It was found that their model was able to perform without any considerable delay, and that an audience could not tell that there was a computer model involved in the improvisations. Some of their problems included some timing issues, as well as problems with more complex improvisations. They also omitted a previously implemented rhythm generation network, when it gave dissatisfying results.

3.1.3. Text based melody generation

Generating music based on textual input is not a groundbreaking concept, and multiple articles have been written concerning the topic. This section will give a brief overview of some implementations seen within research on keywords and lyrics as input for music generation.

Harmon (2017) implemented a method concerned with discovering the semantic meaning of an input sequence of keywords and generate ambient music fitting the topics and content of the keyword input. The music was generated by analysing word relations, and important concepts, which were then translated into queries, that would return ambient music piece annotations within web-based sound libraries. The system combined pieces deemed fit for the given ambience, through several self-critique steps to generate new pieces of music.

The concept of computational generation of melodies to accompany a lyrical input have been explored both by Monteith et al. (2012) and Bao et al. (2019). A novel solution was implemented by Monteith et al. (2012) to generate a melody that would fit given input lyrics. Their system was grounded in the usage of *n-gram*[2]-models to assign rhythm and notes to the input lyric. Firstly a set of MIDI-files was compiled to detect useful rhythmical and pitch patterns for desired musical styles, as well as determining what rhythms and pitches fit with given syllables. An n-gram model was generated to give an assumption of which notes were most likely to succeed other notes, in different musical styles. The system firstly generated a rhythm fitting the lyric. Their system used stress values found for each phoneme of a word, based on their ratings in the *CMUdict* (see subsection 2.4.2), and by using an approach similar to one iteration of an evolutionary algorithm, the best evaluated rhythmical sequence was chosen among 100 randomly generated down-beat annotated rhythms. When the down-beats was assigned, fitting rhythmical values, such as time signature and note durations, were found by taking inspiration from the MIDI-files. The pitch values for each line in the generated songs are chosen by using an n-gram model that assigns note values by a probabilistic

[1] `https://magenta.tensorflow.org/`

[2] `https://web.stanford.edu/~jurafsky/slp3/3.pdf`

approach, based on the defined style. Three randomly chosen input notes for the n-gram model was common for each line of the song, to assure that the melody for each line was related. Monteith et al. (2012) state that: "the system was able to generate melodies that received the same average ratings for pleasing melodic lines as the original melodies", and that the original melodies only scored slightly better than the generated, in regards to how well they fitted with the given lyrics.

Bao et al. (2019) implemented an encoder-decoder RNN model, that would take lyrics as input, as well as a melodic input to provide the model with a context, and generate a fitting melody for the lyrics. The previously predicted melodies would be used as context for succeeding melody line generation, as the melody was generated for one line in the lyrics at a time. Three RNNs were implemented, with GRU memory cells, where one of the models was concerned with encoding the input lyrics to represent the syllable values of the lyric, one model was concerned with encoding the context melody note values, and the last model was concerned with decoding new melody, i.e. predict the next note values in the melody. Their results suggested that their model performs well, and produce melodies of high quality, with respect to the given input lyrics.

Another machine learning approach was introduced by Ackerman and Loker (2017), where a system for generating melodies based on song lyrics was implemented with the goal of it being a collaborative tool in songwriting. Their implementation was based on using random forests[3] rather than Markov chains for predicting the following notes using two prediction based models, without any previous musical knowledge. Similar to the division of the composition task of Bao et al. (2019), the prediction of coming note durations and pitches was separated into two models. Ackerman and Loker (2017) states that their model can generate melodies without human input. By the use of multiple input parameters and the choice of melody corpus, it is though highly tunable by human interaction. Their results suggested that they were satisfied with the system's potential in co-composition.

3.2. Sentiment Analysis

The concept of sentiment analysis, covered in section 2.3, has been subject to much research. As stated by Xia et al. (2008) the demand of access to both lyrics and songs has increased, e.g. with the popularity of song access through portable devices. Subsequently sentiment analysis of song lyrics has become a hot topic in research. Xia et al. mention song recommendations based on a user's current mood, as a concrete example of a technology sprung out of such research. Hu et al. (2009) mention the wish of having such technology work on small devices as a motivation for their research. Both Xia et al. and Hu et al. address that previous algorithms in song sentiment detection have mostly worked by analysing the audio of the song, rather than the lyrics, and argue that analysis of song lyrics would contribute effectively to the sentiment detection.

[3] `https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm`

Xia et al. (2008) based their research on the unsatisfactory results of the *Vector Space Model*[4] (VSM) in classifying the sentiment of songs. They therefore implemented a model called *Sentiment Vector Space Model* (s-VSM), outperforming the VSM. They also found that s-VSM outperformed both Audio-based and Knowledge-based approaches in classifying the sentiments of lyrics. The s-VSM extracted sentiment features from the lyrics and an implementation of the *SVM-light* algorithm (Joachims, 2002) was used for the task of classifying the lyrics' sentiment labels. The s-VSM model was grounded in a set of rules which only considered the terms of lyrics that had sentiment related value, as well as the term's neighbouring modifiers and negations. The strengths of terms and modifiers were gathered from a dictionary of semantics, and in combination the sentiments was settled on.

Hu et al. (2009) suggests that the sentiment of a song should not only be considered by a one-dimensional model, i.e. a negative/positive valence graph, but rather a two-dimensional model, considering the *arousal* (i.e. the energy) of a song as well. Similar to Xia et al. (2008), Hu et al. also used a lexicon of terms with corresponding sentiments to find the sentiment of a song's lyrics by evaluating the lyrics sentence by sentence. The sentence sentiments were then clustered using a *Fuzzy-logic*[5] method. The lexicon also included words in regards to arousal for the sentence evaluation to consider. Important findings from Hu et al. (2009) include that lyrics are not necessarily expressing much in the sense of arousal, as they found that there was more confusion in the arousal dimension than the valence dimension, and that the sentiment in lyrics is not always expressed explicitly, meaning that some information of the sentiment of lyrics can only be found by reading between the lines. Inaccuracies in regards to the workings of their Natural Language Processing tool may also have affected their results.

[4] https://link.springer.com/10.1007/978-0-387-39940-9_918
[5] https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-30440-3_234

4. System Architecture

The following chapter is intended to give the reader a thorough description of the complete architecture of the system developed as a part of this project. The section structure follows the main data flow of the system from input to output closely. Source code for the author's implementation of the system architecture can be found on *GitHub*[1].

4.1. General overview

The architecture presented in this chapter shows the flow and core functionality of an algorithm implemented to generate a melody with included harmonisation for arbitrary input lyrics and their corresponding output files, i.e. a PDF-file with the associated sheet music and a MIDI-file with the associated audio. The main flow of the architecture follows the five main steps *Syllabification* (section 4.3) and *Sentiment analysis* (section 4.2) of the lyrics, *Time signature identification* (section 4.4), *Music generation* (section 4.5) and *Output generation* (section 4.6), which are all described in detail in this chapter. As a supplementing resource, a flow chart for the main architecture is presented in Figure 4.1. As to significantly improve algorithm run-time, and generate output of questionnaire friendly sizes, only the first verse of the lyrics is considered in the syllable and music generating steps of the algorithm. The whole lyrics is considered in the sentiment analysis step.

4.2. Sentiment analysis

The sentiment analysis part of the architecture is heavily dependent on *VADER*, described in subsection 2.4.3. The first step in the sentiment analysis is to calculate the valence scores of each line in the lyrics, which is done through storing Vader's *compound* value for each line. Absolute *neutral* lines, i.e. with a compound value of 0.0, are ignored in further processing. This was done as absolute neutral values were deemed destructive in determining valuable lyrical sentiments for the music generation, as it could impair decisive sentiment results, if many lines are deemed to be absolute neutral.

The thresholds for *positive* and *negative* compound values, as described in subsection 2.4.3, were kept at their existing values, but two new thresholds for *extra positive* ($compound >= 0.8$) and *extra negative* ($compound <= -0.4$) were implemented to apply increased importance to extreme values. For each original compound sentence value x, Equation 4.1 was applied. Manual evaluation of selected lyrics and their associated

[1] `https://github.com/Barnemat/Masteroppgave`

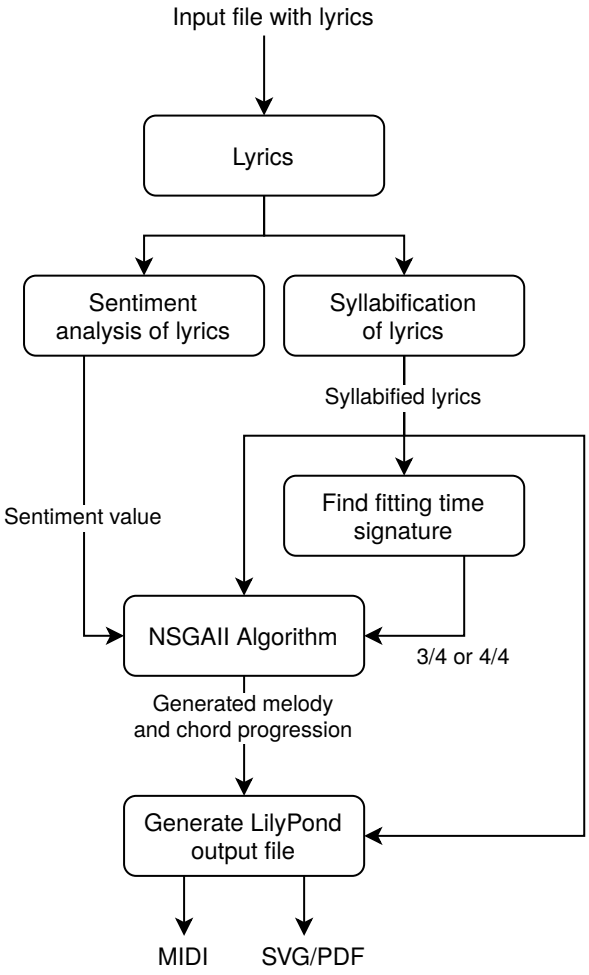

Figure 4.1.: An overview of the main flow of the system architecture

valence scores, derived from the results in the generated data set from Malheiro et al. (2018) was conducted by the author. The results suggested that the disproportionate increase for negative values compared with positive values, given by Equation 4.1, was necessary. The normalised average of the sentence values is then calculated, and the resulting value is passed on to the main algorithm described in section 4.5.

$$f(x) = \begin{cases} x * 2 & \text{if } x \leq -0.4 \\ x * 1.5 & \text{if } -0.4 < x \leq -0.05 \\ x & \text{if } 0.05 < x \leq 0.8 \\ x * 1.25 & otherwise \end{cases} \tag{4.1}$$

4.3. Syllable handling

The syllabification processing of lyrics is done in two possible manners, the first is a hybrid lexical and rule-based approach and the last approach is entirely rule-based. The first step in the syllabification process is to clean the lyrics, which means that all punctuation except apostrophes in contracted words, such as "don't", are removed, and all words are converted to lowercase letters.

Lexical analysis and phonemes

The first solution is based on the availability of the word in *CMUdict* (subsection 2.4.2). If the word is found in the dictionary, the word phonemes are retrieved. The letters of the word are then associated to their corresponding phonemes. This is done by firstly detecting the vowels of the word, as syllables mainly revolve around vowel sounds. The phoneme markings detect if neighbouring vowels should be considered part of the same syllable, or if they should be separated. When vowels are labeled, the consonants are associated to a consonant phoneme, and consonants directly neighbouring a vowel are appended to this vowel's syllable. If there is a doubt between which syllable a consonant should be added to, a neighbouring syllable is chosen at random.

Sonority sequencing principle

The second solution is a more basic rule-based backup method that is imposed on words that have thrown an error in the *CMUdict*-based solution. This solution is based on the Sonority Sequencing Principle (*SSP*), as described by Bartlett et al. (2009). SSP assumes that words are divided into syllables by firstly assigning "sound loudness" values to phonemes of the word, then defining different syllables based on the main rule that sonority values should increase from a syllable's *onset* phoneme through to the *nucleus*, and subsequently fall off to the *coda*. Here *onset* is the first part of a syllable, *nucleus* is the middle (vowel) part and the *coda* is an optional extra part succeeding the nucleus. Based on the sonority principle, multiple solutions could exist for each word, and it does not specify how to choose between legal onsets (Bartlett et al., 2009).

The first step of syllabification in this solution is to syllabify a word in accordance to the *SSP*, which is done using the *SyllableTokenizer* tool implemented in *NLTK* (subsection 2.4.1). Secondly a set of rules is imposed on the syllable tokenised word, to help in mitigating errors. Such errors occur as a consequence of SSP not being able to correctly differ between seemingly equally valid syllabification solutions, such as described in the previous paragraph. Examples of problems that are mitigated are special cases where vowels should not have any consonants added to them, word endings which are to be considered part of a previous syllable, and word endings that should not be considered a part of the previous syllable. This solution is prone to minor errors, which consequently is the reason it is considered a backup solution.

4.4. Time signature identification

The time signature for the algorithm is decided by a simple evaluation of the syllables generated by the syllabification algorithm, described in section 4.3. The possible time signatures that may be passed on to the music generation algorithm, presented in section 4.5, are limited to either $\frac{3}{4}$ or $\frac{4}{4}$. The numerator is the only variable part of the time signature.

The first step in deciding the time signature is to count the syllables in each line of the lyrics, adding numbers to ensure that the syllable number is either divisible by 3 or 4. The time signature is then decided by counting the number of lines having a syllable count divisible by 3 or 4 respectively. The higher number decides the time signature.

4.5. Genetic Algorithm Implementation (NSGAII)

This section covers the main part of the system, i.e. the music generating part. The implemented algorithm is a multi-objective genetic algorithm closely following the description of the non-dominated sorting algorithm (*NSGAII*), described in subsection 2.2.2.

4.5.1. Genotype and phenotype design

The genotype design for phenotypes (individuals) of the algorithm is based on the fact that music for given lyrics might be of variable length and composition. Some inspiration in the genotype design has been taken from Olseng (2016), such as the decision to divide the genes into separate parts for chords and melody. One for storing the chord sequence for the music and one for storing the melody. To make the genes easily understandable for debugging purposes, as well as simplify the output generating process, the note representation in the music closely follows the syntax of LilyPond (subsection 2.4.4).

As to establish control of the gene pool and its individuals, each individual in a population is handled by a phenotype class, which holds information about its genotypes, as well as handling the initialisation of said genotypes.

Representation of melodic genes

The melody genes are represented as a multi-dimensional array of musical beats. Each beat holds a number of notes, specified by the note's timing values. Notes could also be stored in a deeper array dimension within a beat, signifying melismas (multiple notes that should be sung/played over one syllable). Although notes are stored in beats, the main handling of notes is done with respect to measures, which are collections of either three or four beats, depending on the time signature.

Notes are represented as letters corresponding to note pitches, a corresponding octave marking and a timing value. The note octave is specified by an empty string (for small octave notes), commas (for octaves lower than the small octave) or apostrophes (for octaves higher than the small octave). c' is middle C or C4 in LilyPond notation (2.4.4). The possible span of notes in the melodic genes are within three octaves, but with g

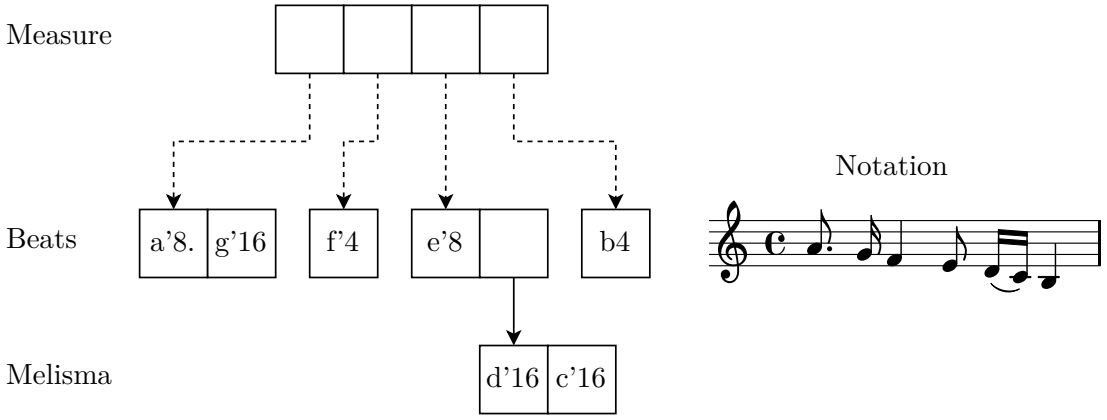

Figure 4.2.: An example of a melody genotype, representing a descending melody in the A-minor scale that spans one measure.

defined as the minimum note value, hence notes span from *g* to *b"*, i.e. the number of possible notes $N = (7 + 17 + 17) + 1 = 42$, when both values for sharp/flat notes are included. The numbers in parentheses correspond to how many notes are included from each octave, and the succeeding 1 represents the rest value *r*. The apostrophes appended to the note *b"* signify the note's octave, i.e. a high B value.

A note's timing is defined by an appended number that may be dotted, i.e. have a succeeding dot (.). The number defines what fraction of a whole note the note's duration is. The possible fractions consist of multiples and divisors of $4 \in [1, 16]$. This means that the longest note duration is $\frac{1}{1}$, i.e. a whole note, and that the shortest possible note duration is $\frac{1}{16}$, i.e. a 16th note. For fractions with a denominator larger than 1 and lower than 16, notes may have a dot appended to the number. The dot value specifies that the note timing should be the fraction specified by the number in addition to the fraction divided by 2. As an example, this would mean that the note *c4.* has a duration of $\frac{1}{4} + \frac{1}{8}$.

The total number of possible variations a note could be represented by is consequently $n = 42 * 8 = 336$.

Notes could technically be distributed at random within the given genetic space, but the algorithm makes sure that beats are grouped into measures and that a beat has the size specified by the denominator of the time signature, i.e. 4. This means that a beat should not have a duration longer than a quarter note ($\frac{1}{4}$). In cases where fractions exceed the quarter note limit, possible durations from other beats in the given measure are either removed or reduced. A measure holds an amount of beats equal to the nominator of the time signature. The only exception is if in the last measure of the melody, the maximum number of notes is reached. The measure could then be cut short. The maximum number of notes in a melody genotype varies based on number of input syllables given to the algorithm, as well as the number of generated melismas. In an initial population there is a 15% chance for melisma generation in each beat.

An example of a simple melody genotype is displayed in Figure 4.2. Each layer of

the figure is meant to symbolise different arrays, i.e. the top layer (measure) contains sub-arrays (beats), and each beat could have one sub-array containing a melisma. The melody genotype is set to span one measure, and the dashed arrows illustrate the different beats needed to fill the measure. The reasoning behind using dashed lines is to signify that the genes are not represented by measures in the phenotype, and that the concept of measures is only applied when needed. For the sake of simplicity all beats in the example are "perfect" beats, i.e. there is no reductions in consecutive beats, as a result of note durations longer than a quarter note. In the third beat, an example of a melisma, and how it is divided into a separate array from the rest of the beat, is displayed. The corresponding musical notation is shown at the far right of the figure. The time signature is set to be $\frac{4}{4}$.

Representation of chord genes

Chord genes are represented as a one-dimensional array of chord values. One chord maps to a single measure in the melody, and the number of generated chords is directly proportional to the number of measures of melody genes generated.

Single chords are represented as a collection of up to four notes, following the same notation as in melody, but with a combined timing value, as well as a different octave space. The allowed octave space is two, but the minimum chord note value is set to a, (great A - great is one octave lower than small), which is in the middle of its octave. The comma signifies that the note octave is one lower than a. Further the maximum chord root value is set to a, to ensure that all notes may be chord roots. The main reason for setting a maximum root note value was to prevent chords interfering to much with the melody. The absolute maximum note value the last note of a chord may have is f', which happens if an augmented major triad is generated on an a note.

Chords are generated into specific note patterns, designed to help chords conform to a given key signature. Following a set of triad types, presented in Table 2.2 in section 2.1, the chord notes are chosen. This means that all chords follow one of the four patterns defined in Table 2.2, i.e. generated chord triads could be either *major triads*, *augmented major triads*, *minor triads* and *diminished minor triads*. Chords are constructed by following the set integer distances. To generate a C major triad the root note is found by adding the distance $d = 0$ to C, the second note is found by adding $d = 4$ to C and the last note by adding $d = 7$ to C. This yields a C major triad (C, E and G). The chord patterns are chosen to fit with scale degrees and harmonisation concepts in regards to notes in major/minor scales, and their harmonic value, as previously described in section 2.1. It is worth noting that even though the melody follows the natural minor scale, the harmonisation in chords follows the harmonic minor scale.

All chord notes have a fixed position, meaning that the first note is the root, the second note is regarded as the third, and the third note regarded as the fifth. Chords also have the possibility to have a fourth note added to it for flavour. This makes it possible to generate more complicated chords. The flavour note does not follow any strict patterns, such as for the three first notes, other than relative note type/octave. This means that all notes are added in ascending order, where each note must be higher

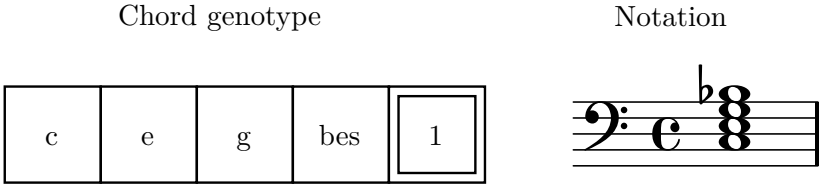

Figure 4.3.: An example of a chord genotype, representing a C7 chord that spans one measure.

than the previous.

The chord timing is decided solely based on the time signature. As a chord is mapped to a measure, the timing is set to fit within the measure. For $\frac{4}{4}$ time, this means that all chords have a duration of a whole note. For $\frac{3}{4}$ time, all chords have a duration of a dotted half note.

An example of a minimal chord genotype is shown in Figure 4.3. The chord progression is set to hold only one chord. For multiple chords, the general pattern would stay the same. The first four cells are reserved for the chord notes, and the last cell is reserved for the genotype duration. The fourth cell is omitted for chords without a flavour note. The corresponding musical notation is shown at the far right of the figure. The time signature is set to be $\frac{4}{4}$. Notice how the staff clef notation is different from the melody genotype from Figure 4.2, i.e. the melody use a treble clef and the chords use a bass clef.

4.5.2. Initialisation

When the algorithm is initialised, the number of syllables in the lyrics and the lyrics' sentiment value are analysed. The first step taken is to generate a musical key for the population. The key is chosen semi-randomly, where the letter is random, but the flavour, i.e. the choice between major and minor, is chosen based on the input sentiment value. For a sentiment value $v > 0$, the key will be major, otherwise it will be minor. The reasoning behind the strict choice of key is that it helps for consistency during testing. If the system was to be applied to real world problems, it would probably be wise to choose the key based on a probabilistic approach, especially for values close to zero.

Due to a problem with double flat and double sharp notes, no flat/sharp keys are generated. This problem occurred as a consequence of scale handling in fitness functions (subsection 4.5.3), where the algorithm tried to build and handle scales that would need double flat/sharp notes, for sharp/flat key signatures. As double flat/sharp key signatures and scales are not covered by the *Circle of Fifths*, described by Tobey (2012), there was no initial implementation supporting the concept. The double flat/sharp problem was deemed of little importance to the main functionality of the algorithm, and it was consequently bypassed.

An initial population of phenotypes is generated based on the set population size (N). The melody genes are generated at random. The only constraint for the melody size is

the number of syllables (s) given by the input. Even though a melisma is a collection of notes, it is counted as note one in regards to the number of notes in the melody. When s notes have been generated, the chords are initialised semi-randomly. The root note and a possible fourth flavour note is generated at random, while the other notes follow a randomly chosen triad pattern from Table 2.2. The number of generated chords is proportional to the number of measures generated by the melody.

4.5.3. Objectives and fitness

As mentioned in subsections 2.2.1 and 2.2.2, *NSGAII* is a multi-objective genetic algorithm, which means that its fitness functions and evaluation are grounded in the strive for finding an optimal solution that satisfies all objectives. The decision taken regarding what objectives to implement and which fitness functions to incorporate into each objective, was arguably the most decisive factor for shaping the melodies. It was chosen to divide the optimisation task for the three most distinct different parts of the melody generation, i.e. melodies, chords and lyrics, into different objectives. As such two completely separate objectives for melody and chord shaping, *objective 2* for optimising the melody and *objective 3* for handling the optimisation of the chord progression, was implemented. Objective 1 is mainly meant as an optimisation of the melody, but with a serious focus on having the melody revolve around its corresponding chords. Objective 4 handles the optimisation of the melody with a basis in it fitting well with the given input lyrics.

The main inspiration for objective implementation in this system was Olseng (2016), and both objective 1 and 2 are heavily influenced by his implementation.

Objective 1 - Melodic measures

The first objective handles the melody on a measure to measure basis, with respect to the melody line and the corresponding chord. As mentioned in subsection 4.5.1, chords always last for exactly one measure. This measure approach is heavily inspired by Olseng (2016) and Wu et al. (2014), which was Olseng's main inspiration for his corresponding objective. The main purpose of the objective is to make sure that the melody revolves around its chords, as well as being well rooted in the key signature. The objective also ensures that notes are closely related in pitch, i.e. it does not benefit major pitch distances between notes.

As done by Olseng (2016), notes are classified into three main categories, namely **chord notes**, **scale notes** and **non-scale notes**. *Chord notes* are notes in the melody that also appear in the chord, when the octave is disregarded. *Scale notes* are the notes in the melody that are found in the scale specified by the key signature, but are not already classified as *chord notes*. The remaining notes are classified as *non-scale notes*.

Scale notes and *non-scale notes* are also classified as **ornament notes**, if they are directly neighboured by *chord notes*. This is a simplified solution to the one of Olseng (2016), where he sub-classified *ornament notes* into two sub-categories, i.e. passing and

neighbouring tones. Olseng's fitness functions concerning the sub-categories were instead combined into one fitness function (no. 2) in this implementation.

Note intervals are defined as two neighbouring notes $[x, y]$ in the measure, which are used in Equation 4.2. The equation takes in a number n of intervals and, depending on the distance in semitones between the notes calculated by $|x - y|$, where x and y denote the absolute indices of the notes, yields the negative value of the sum of $f(x_i, y_i)$ results. The equation is based on similar equations used by Olseng (2016) and Wu et al. (2014), in their fitness evaluations.

$$-\sum_{i=1}^{n} f(x_i, y_i) \quad \text{where} \quad f(x, y) = \begin{cases} |x - y| - 7 & \text{if } |x - y| > 7 \\ 0 & otherwise \end{cases} \tag{4.2}$$

The fitness functions of the objective are shown in Table 4.1. They are presented numerically, with their testing conditions, fitness value impact for when the conditions are true and a brief description of the fitness function's purpose. The fitness functions were either taken directly from Olseng (2016) (no. 5 and 6), modified adaptations of Olseng's fitness functions (1 through 4) or novel (no. 7 and 8). The pseudo-function name $num(x)$ is used to define the number of x.

Table 4.1.: Objective 1 - Fitness functions

No.	**Condition**	**Cond. is true**	**Description**
1	num(non-chord notes) $\leq num$(chord notes)	$+1$	Stabilises melody to revolve around chord
2	num(ornament notes) $< num$(scale pitches)	$+2$	Makes sure ornament notes most often appear in scale
3	num(non-scale-notes) $< num$(ornament notes)	$+1$	Rewards ornament notes being in scale
4	First note is root or fifth of chord	$+1$	Stabilises melody to start on most important chord notes
5	num(unres. non-scale notes)	$-n$	Punishes measures that have a number of non-scale notes not resolving in chord notes
6	num(intervals) where distance > 7	Eq. 4.2	Stabilises melody to not include large jumps between notes
7	Measure contains only one note	-1	Increase probability of measures containing more than one note, i.e. contribute to melody progression
8	Flavour note has higher pitch than first note	-1	If a chord has a 4th flavour note, it is discouraged for it to have a higher pitch than the first melody note.

The fitness functions are imposed on each measure in the melody, and the total fitness

score of a melody is the sum of all values from each measure divided by the number of measures. The division step is a necessity done to stabilise the fitness values, based on the ability of phenotypes to have genes that vary in length, as specified in subsection 4.5.1.

Objective 2 - Melodic composition

The second objective is concerned with a global evaluation of the melody, i.e. the melodic composition, rather than the measure to chord evaluation presented objective 1. The objective is rooted in 21 fitness functions that normalise numerical values regarding the melody to fit in the interval $[0, 1]$. In addition to said fitness functions, the objective makes use of two *melodic punishment* functions, that are implemented to punish violations of important concepts.

The fitness functions of objective 2 are presented in Table 4.2. The main objective of the fitness functions is to evaluate the entire melodic composition, based on important musical concepts and boundaries. The functions yield normalised values regarding the melody, and the values are examined and evaluated to score the phenotypes, based on a complete analysis of their melodic genotypes. As opposed to the approach in objective 1, the final fitness score is not re-normalised, based on melody size, as all return values are already normalised. The fitness functions have deep foundations in the melodic analysis methods introduced by Towsey et al. (2001), which was also Olseng (2016)'s main inspiration for his implementation of the corresponding objective. Some of the concepts from Towsey et al. that were discarded by Olseng are reintroduced in this implementation. Fitness functions 1 through 16 are all based on Towsey et al. (2001), whereas only fitness functions 1 to 2 and 5 to 12 were implemented by Olseng (2016). Some fitness functions have been renamed, but keep similar functionality. Fitness functions 17 through 21 can be deemed novel fitness functions that were implemented to discourage the generation of melodies the author deemed unsuited for vocals. The reasoning behind fitness decisions is further elaborated in subsection 5.2.3.

For the sake of simplicity, some important terms and concepts to better understand the fitness functions are explained and/or repeated in the following description.

Tonic & Dominant Tonic is a term specifying the first note of the key signature scale. Dominant is the fifth note of said scale.

Intervals and their dissonance ratings An interval is defined as two neighbouring note values. Their dissonance ratings are considered in fitness function 4, and follows the values specified by Towsey et al. (2001). Intervals with a distance of 0, 1, 2, 3, 4, 5, 7, 8, 9 and 12 are defined as having no dissonance, i.e. a value of 0.0. Intervals with a distance of 10 are defined as having a dissonance value of 0.5 and intervals with a distance of 6, 11, 13 or higher are defined as having a dissonance rating of 1.0.

Quanta Quanta is a term applied by Towsey et al. (2001) that describes the shortest possible note duration in the melody, i.e. a quanta is the minimal fraction a note's

duration can be. As described in subsection 4.5.1, the minimal duration fraction is $\frac{1}{16}$ in this implementation.

Formula specific information The formulas described in Table 4.2 are presented in a simplified notation, where *num(x)* specifies a function counting the number of x elements and $min - max$ functions specify the min-max values of different data types, depending on context. The function *sum(x)* is used instead of the $\sum$ symbol.

Table 4.2.: Fitness functions for objective 2

No.	Simplified formula	Target value	Description
1	$\frac{\text{num(distinct notes)}}{\text{num(notes)}}$	Table 4.3	**Note variety** specifies the note variety based on the fraction of distinct notes in the melody.
2	$\frac{max(\text{notes}) - min(\text{notes})}{24}$	0.50	**Note range** takes the index of the highest and the lowest note in the melody, and divides their difference by the desired range. If max range is exceeded, 1.0 is returned. 24 is the total distance of two octaves.
3	$\frac{num(\text{tonic or dominant quantas})}{num(\text{quantas})}$	Table 4.3	**Key focus** specifies the fraction of notes that are either tonic or dominant.
4	$\frac{sum(\text{dissonance ratings})}{num(\text{intervals})}$	Table 4.3	**Dissonant intervals** looks at the fraction of intervals that are regarded dissonant, based on their dissonance values.
5	$\frac{num(\text{rising intervals})}{num(\text{intervals})}$	Table 4.3	**Contour direction** defines the contour direction of most intervals, i.e. if most intervals are ascending or descending. A value of 0.5 means that the amount is equal.
6	$\frac{num(\text{same direction intervals})}{num(\text{intervals}) - 1}$	Table 4.3	**Contour stability** finds the fraction of intervals, which following intervals move in the same direction.

The table is continued on the next page

No.	Simplified formula	Target value	Description
7	$\frac{num(\text{diatonic distance intervals})}{num(\text{intervals})}$	Table 4.3	**Diatonic step movement** specifies the fraction of intervals that have a diatonic step distance. I.e. a distance = 2 between all notes, except between (B-C) and (E-F), where the distance = 1.
8	$\frac{num(\text{notes})}{num(\text{quantas})}$	0.30	**Note density** is the fraction of notes in regards to the number of quantas.
9	$\frac{num(\text{rests})}{num(\text{quantas})}$	0.15	**Rest density** is the fraction of rests in regards to the number of quantas.
10	$\frac{num(\text{distinct note durations})}{num(\text{possible note durations})}$	0.70	**Rhythmic variety** specifies the fraction of distinct timings that are used, based on the total number of possible timings.
11	$\frac{num(\text{equal notes intervals})}{num(\text{intervals})}$	Table 4.3	**Repeated notes** specifies the fraction of intervals that contain the same two notes.
12	$\frac{num(\text{equal duration intervals})}{num(\text{intervals})}$	0.20	**Repeated rhythms** specifies the fraction of intervals that contain notes with the same duration.
13	$\frac{num(3 \text{ notes rep. patterns})}{num(\text{intervals}) - 4}$	0.10	**Repeated notes (patterns of 3)** specifies the fraction of times three consecutive notes are repeated by the next three consecutive notes, based on the number of possible three note patterns in the melody.

The table is continued on the next page

No.	Simplified formula	Target value	Description
14	$\frac{num(\text{3 durations rep. patterns})}{num(\text{intervals}) - 4}$	0.10	**Repeated rhythms (patterns of 3)** specifies the fraction of times three consecutive note durations are repeated in the next three consecutive notes, based on the number of possible three note patterns in the melody.
15	$\frac{num(\text{4 notes rep. patterns})}{num(\text{intervals}) - 5}$	0.05	**Repeated notes (patterns of 4)** specifies the fraction of times four consecutive notes are repeated by the next four consecutive notes, based on the number of possible four note patterns in the melody.
16	$\frac{num(\text{4 durations rep. patterns})}{num(\text{intervals}) - 5}$	0.10	**Repeated rhythms (patterns of 4)** specifies the fraction of times four consecutive note durations are repeated in the next four consecutive notes, based on the number of possible four note patterns in the melody.
17	$\frac{num(\text{semitone distance intervals})}{num(\text{intervals})}$	Table 4.3	**Semitone steps** specifies the fraction of intervals where the note distance = 1.
18	$\frac{num(\text{16th notes})}{num(\text{notes})}$	0.05	**16th notes** specifies the fraction of 16th notes in the melody.
19	$\frac{num(\text{whole notes})}{num(\text{notes})}$	Table 4.3	**Whole notes** specifies the fraction of whole notes in the melody.
20	$\frac{num(\text{four repeated notes})}{num(\text{notes}) - 4}$	0.00	**Heavily repeated notes** specifies the fraction of notes in patterns of four, where the notes have a distance of 0.

The table is continued on the next page

No.	Simplified formula	Target value	Description
21	$\begin{cases} 1.00 & \text{if last note is tonic} \\ 0.00 & \textit{otherwise} \end{cases}$	1.00	**Last note is tonic** specifies whether the melodic composition ends on a tonic.

Target values are values specifying the most desired outcome of the fitness functions, described in Table 4.2. The normalised values generated from each fitness function are all compared with a target value. The comparing method is described in Equation 4.3, and is the same as used by Olseng (2016) in his *Melodic Global Objective*. The equation takes in a fitness evaluation value x that is generated by a fitness function and the associated target value y, and adds the result to the aggregated sum for all fitness functions. This means that if a fitness evaluation gives a value equal to the target value, the returned value is 1. The maximum achievable fitness score for this objective is therefore equal to the number of fitness functions, listed in Table 4.2, i.e. 21.

$$f(x, y) = 1 - |x - y| \quad \text{where} \quad \text{x, y} \in [0, 1] \tag{4.3}$$

An important part of the sentiment based melody generation is handled by dynamically choosing target values for the fitness functions, based on the input sentiment value. All fitness functions are given an initial target value, but in some special cases the initial values are overwritten by a sentiment based target value. For fitness functions that may have their target value overwritten, their different values are specified in Table 4.3. The number in the first column specifies the fitness function index in relation to its index in Table 4.2. The target values are then represented as a one-dimensional array y, which specifies return values based on the value of a sentiment value x. To get the precise target value, firstly Equation 4.4 is applied to the values, i.e. $g(x, y)$. The equation returns the corresponding element from the array y, based on the value of x.

$$g(x, y) = \begin{cases} y_1 & \text{if } x < -0.75 \\ y_2 & \text{if } -0.75 \leq x < -0.5 \\ y_3 & \text{if } -0.5 \leq x < -0.1 \\ -1 & \text{if } -0.1 \leq x < 0.1 \\ y_4 & \text{if } 0.1 \leq x < 0.5 \\ y_5 & \text{if } 0.5 \leq x < 0.75 \\ y_6 & \textit{otherwise} \end{cases} \tag{4.4}$$

Equation 4.5 is then applied to the Equation 4.4 output, to get the correct target value. For Equation 4.5 the input value r is the return value from $g(x, y)$. The input value v is the initial target value, specified in Table 4.3. The output from Equation 4.5 is considered the actual target value. The equation is meant to return the original target

value v, if no specific target value is added for the given sentiment value, i.e. v is used when the corresponding index $y_i = -1$ or the sentiment value is not substantial enough to invoke special sentiment based values. If the result from Equation 4.4 is -1, then the original target value is returned.

It is important to note that Equation 4.4 and Equation 4.5 are not exact representations of the implemented functions, but rather meant to be abstractions to aid in understanding how the initial target values are only overwritten in specific cases, and only for specific fitness functions.

$$f(r, v) = \begin{cases} r & \text{if } 0.0 \leq r \leq 1.0 \\ v & otherwise \end{cases} \tag{4.5}$$

Table 4.3.: Sentiment based target values for selected fitness functions - objective 2

No.	**Target values** $f(g(x, y), v)$ where $x =$ sentiment value
1	$y = (0.25\ 0.30\ 0.40\ -1\ -1\ -1),\quad v = 0.50$
3	$y = (-1\ -1\ -1\ -1\ 0.45\ 0.50),\quad v = 0.35$
4	$y = (0.20\ 0.20\ 0.15\ -1\ -1\ -1),\quad v = 0.00$
5	$y = (0.25\ 0.30\ 0.40\ 0.60\ 0.70\ 0.75),\quad v = 0.55$
6	$y = (0.65\ -1\ 0.55\ -1\ 0.45\ 0.40),\quad v = 0.60$
7	$y = (0.50\ 0.45\ -1\ -1\ -1\ -1),\quad v = 0.40$
11	$y = (0.20\ 0.20\ -1\ -1\ -1\ -1),\quad v = 0.15$
17	$y = (0.30\ 0.25\ 0.20\ -1\ -1\ -1),\quad v = 0.10$
19	$y = (0.15\ 0.10\ -1\ -1\ -1\ -1),\quad v = 0.05$

Melodic punishment functions are fitness functions exempt from the target value based approach used by the functions listed in Table 4.2. The main purpose of the melodic punishment functions is to more severely punish melodies that are violating important concepts, not thoroughly covered by the minimal impact of each target value based fitness function. The impact of each target based function decreases based on total amount of functions, and the two implemented melodic punishment functions cover two concepts that were deemed to be of importance for generating a melodies intended for vocals. Reasoning for this is covered in subsection 5.2.3. The specific melodic punishment functions are listed in Table 4.4. This table follows the same simplified notation used in Table 4.2.

Objective 3 - Harmony optimisation

The third objective is concerned with evaluating the chord progression of the phenotypes. The objective is grounded in several fitness functions aiming to generate a varied, interesting, but key focused chord progression. Similar to the approach described in

Table 4.4.: Melodic punishment functions - objective 2

No.	Simplified formula	Description
1	$\begin{cases} -\left(\frac{num(\text{fitness functions})}{4}\right) & \text{if } num(\text{16th notes}) > \frac{1}{4} \\ -\left(\frac{num(\text{fitness functions})}{4}\right) & \text{if } num(\text{whole notes}) > \frac{1}{4} \\ 0 & otherwise \end{cases}$	**Large number of extreme notes** punishes melodies where more than a quarter of the notes have a duration of either 1 or $\frac{1}{16}$.
2	$-\left(\frac{num(\text{extreme intervals})}{\frac{num(\text{fitness functions}) * 0.01}{2}}\right)$	**Extreme note intervals** punishes melodies based on the fraction of intervals containing two whole notes or two 16th notes.

objective 2, the fitness functions return normalised values in the interval $[0, 1]$ for evaluation. The concept of post-fitness punishment as used in the melodic punishment functions, presented in Table 4.4, is carried over to this objective as well, in the form of harmonic punishment functions, presented in Table 4.8. For objective 3 a total of 10 normalising fitness functions and 5 harmonic punishment functions were implemented.

This objective is loosely based on concepts such as chord roots not being in key, semitone dissonances, unresolved chords and repetition used by Olseng (2016) in his *Harmonization Objective* and *Harmonic Progression Objective*. This objective however follows a different approach than Olseng, in having normalised values as a basis for the objective, as well as other structural changes, such as having increased musical knowledge in the fitness functions. Whereas Olseng used punishment functions as a basis for his corresponding objectives, this objective uses a combination of normalise-based and punishment-based functions, where the punishment functions are in minority.

Fitness functions that revolve around normalised observations of phenotypes' chord progressions are presented in Table 4.6. The fitness ranking process is highly similar to the one used in objective 2, where all normalised scores from each fitness function are summed, before being handled by the punishment functions. This objective has no regard of the melody genes of phenotypes, and is exclusively handling the evaluation of chord genes.

As done for objective 2, some important terms and concepts that should be explained and/or repeated for the sake of better understanding the fitness functions are presented below. Some tonic-dominant and formula specific information is skipped, as these concepts were presented in section 4.5.3.

Table 4.5.: Chord triad types in different scale degrees - copy of Table 2.3

Chords in different scale degrees							
	1st	2nd	3rd	4th	5th	6th	7th
Major	I	ii	iii	IV	V	vi	vii°
Minor	i	ii°	III+	iv	V	VI	vii°

Triad A triad is a combination of three specific notes, played simultaneously as a chord. As described in subsection 4.5.1, all chords in this implementation follow specific patterns. In the fitness ranking in Table 4.6 the word triad specifies only the triad portion of the chord. Chords might have a fourth flavour note in combination with the triad. For the ranking and handling of specific root note and triad relationships, the scale degrees presented in Table 4.5 are used.

Tonic & Dominant Tonic and dominant chords are specified with regards to their root notes being either tonic or dominant, as well has having their triad qualities matching the scale degrees defined in Table 4.5.

Table 4.6.: Fitness functions for objective 3

No.	Simplified formula	Target value	Description
1	$\sum f(x_i)$ where $f(x) = \begin{cases} 0.5 & \text{if chord is tonic} \\ 0 & \text{otherwise} \end{cases}$	1.00	**Starts/ends on tonic** specifies whether the chord progression start/ends with chords containing tonic triads. For the formula, x is an array containing the progression's starting chord and ending chord.
2	$\frac{num(\text{dominant triad chords})}{num(\text{chords})}$	Table 4.7	**Dominant triads** specifies the fraction of chords that contains dominant triads.
3	$\frac{num(\text{resolved dominant chords})}{num(\text{dominant chords})}$	1.00	**Resolved dominants** specifies the fraction of chords containing dominant triads that resolve into a tonic triad chord within two measures.

The table is continued on the next page

No.	Simplified formula	Target value	Description
4	$\frac{num(\text{repeated chord intervals})}{num(\text{chord intervals})}$	Table 4.7	**Repeated chords** specifies the fraction of chord intervals containing the same chord.
5	$\frac{num(\text{tonic triad chords})}{num(\text{chords})}$	Table 4.7	**Tonic triads** specifies the fraction of chords containing tonic triads.
6	$\frac{num(\text{distinct chords})}{num(\text{chords})}$	Table 4.7	**Distinct chords** specifies the fraction of chords that are distinct.
7	$\frac{num(\text{4th flavour note chords})}{num(\text{chords})}$	0.30	**4th flavour note** specifies the fraction of chords that have an extra 4th flavour note.
8	$\sum f(x_i)$ where $f(x) = \begin{cases} 0.5 & \text{if triad is dominant} \\ 0 & \text{otherwise} \end{cases}$	0.00	**Starts/ends on dominant** specifies whether the chord progression start/ends with chords containing dominant triads. For the formula, x is an array containing the progression's starting chord and ending chord.
9	$\frac{num(\text{tonics with 4th flavour note})}{num(\text{tonic triad chords})}$	0.30	**Tonics with flavour** specifies the fraction of tonic triad chords containing a 4th flavour note.
10	$\frac{num(\text{semitone flavour note dist.})}{num(\text{4th flavour note chords})}$	0.00	**Semitone flavour dissonance** specifies the fraction of chords containing a 4th flavour note, where the flavour note has a semitone distance to another note in the chord.

Target values are values specifying the most desired outcome of the fitness functions, described in Table 4.6. The target value ranking follows the same process as for objective 2. The normalised output from the evaluation functions are compared to target values by using Equation 4.3.

As for objective 2, the harmonisation objective also follows a dynamic sentiment based target value approach, where some fitness functions have variable target values. For said fitness functions, the same approach as in objective 2 is followed, by first applying Equation 4.4 and then Equation 4.5 to obtain a target value. All fitness functions with sentiment based target values for this objective are presented in Table 4.7 in the same manner as for objective 2.

Table 4.7.: Sentiment based target values for selected fitness functions - objective 3

No.	**Target values** $f(g(x,y),v)$ where $x =$ sentiment value
2	$y = (0.20\ 0.20\ -1\ -1\ 0.40\ 0.40), \quad v = 0.30$
4	$y = (0.30\ 0.20\ 0.15\ -1\ 0.20\ 0.20), \quad v = 0.10$
5	$y = (0.60\ 0.50\ 0.45\ 0.50\ 0.55\ 0.55), \quad v = 0.40$
6	$y = (0.35\ 0.40\ 0.40\ -1\ 0.40\ -1), \quad v = 0.50$

Harmonic punishment functions are implemented as an addition to the target value based fitness functions presented in Table 4.6, and follow an approach similar to the one of melodic punishment functions presented in Table 4.4. The main purpose of the harmonic punishment functions is to punish chord progressions more heavily than a normalised valued approach does. This is done for the most important concepts, such as verifying that chords follow correct scale degrees, chord root notes are found in the given key signature, etc. The harmonic punishment functions and their simplified formulas are presented in Table 4.8. The formulas were designed as to punish chord progressions based on the severity of the violated concepts, but with an intention of not directly disqualifying phenotypes for minor violations. In other words, punish phenotypes gradually, and somewhat linearly.

Objective 4 - Lyrics optimisation

The *Lyrics optimisation* objective is designed to evaluate the extent of which the lyrics fit the generated melodies of different phenotypes. The fitness functions are implemented to promote phenotype solutions where the words of the lyrics land on sensible places in the melody. The reasoning behind the addition of fitness functions 2 through 5 were subjective evaluations by the author, by evaluating generated melodies and corresponding lyrics, as well as studying lyrical layouts of various song books. All fitness functions for this objective are presented in Table 4.9.

$$\sum_{i=1}^{n} f(x_i, y_i, y) \quad \text{where} \quad f(x,y,z) = \begin{cases} g_1(y,z) & \text{if } x = 1 \\ g_2(y,z) & \text{if } x = 2 \\ g_3(y,z) & otherwise \end{cases} \tag{4.6}$$

The first fitness function is rooted in the lexical stress markers found as parts of phonemes returned by looking up lyrical words in *CMUdict* (subsection 2.4.2). The

Table 4.8.: Harmonic punishment functions - objective 3

No.	**Simplified formula**, $y = num$(fitness functions)	**Description**
1	$f(x,y) = \begin{cases} -\left(\frac{y}{3}\right) & \text{if } x < 0.30 \\ -\left(\frac{y}{1.5}\right) & \text{if } x < 0.15 \\ 0 & otherwise \end{cases}$	**Minimum tonic amount** punishes chord progressions where less than 30% of the progression contains tonic triad chords. For the formula, x is the portion of tonic triad chords.
2	$f(x,y) = \begin{cases} -(x*y) & \text{if } x < 0.30 \\ 0 & otherwise \end{cases}$	**Exceeding chord repetition** punishes chord progressions based on the product of the number of fitness functions and portion of repeated chords in the progression. For the formula, x is the portion of repeated chords.
3	$f(x,y) = -(1.5*y - (x*y*1.5))$	**Correct scale degrees** punishes chord progressions where chords do not follow scale degrees specified by Table 4.5. For the formula, x is the portion of chords following set scale degrees.
4	$f(x,y) = -(2*y - (x*y*2))$	**Roots in key** punishes chord progressions where chord roots are not in the given key. For the formula, x is the portion of chord roots not in key.
5	$f(x,y) = -\left(x*10*\frac{x}{y}\right)$	**Flavour note in chord key** punishes chord progressions based on the number of flavour notes that are not in the key of its associated chord. For the formula, x is the number of chords with flavour notes not in its key.

intent of the function is to increase fitness scores for phenotypes where syllables have a correct rhythmical placement in the melody. This means that based on how correctly note durations fit with stress markers in their corresponding syllable phonemes, the phenotype fitness score increases. The concept of using phoneme stress markers to generate suitable syllable rhythms and relationships was also explored by Monteith et al. (2012), in their lyric to melody implementation.

As the main equation for this fitness function is of a complicated nature, it is mostly

Table 4.9.: Fitness functions for objective 4

No.	Simplified formula	Target value	Description
1	$\frac{sum(\text{Equation 4.6) for each word}}{num(words)}$	1.00	**Stress and duration satisfaction** returns a normalised value describing how well syllables of the lyrics and their stress values correspond to appropriate note durations. E.g. primary stress values should have longer note duration than a secondary or no stress value in the same word.
2	$\frac{num(\text{lines ending measure})}{num(\text{lines})}$	1.00	**Lines ending measure** specifies the fraction of lines where the last syllable also ends a measure.
3	$\frac{num(\text{longer note ending lines})}{num(\text{lines})}$	1.00	**Longer note line ending** specifies the fraction of lines where the last syllable of the line has a longer corresponding note duration than the previous syllable.
4	$\frac{num(\text{tonic/dominant ending lines})}{num(\text{lines})}$	0.60	**Tonic/dominant line endings** specifies the fraction of lines where the last syllable ends on a tonic or dominant note.
5	$\frac{num(\text{multi-measure syllables})}{num(\text{syllables})}$	0.10	**Multi-measure syllables** specifies the fraction of syllables that span multiple measures.

presented outside of the table view in Table 4.9. The result of the stress and duration evaluation is calculated by Equation 4.6, where x is an array of stress values associated to a word in the lyrics, y is an array of the note durations corresponding to each stress value and n is the size of the arrays. Equation 4.6 makes use of three different supporting functions g_1, g_2, g_3, presented in Equation 4.7 and Equation 4.8. For all g functions, y is the note duration corresponding to a phoneme, and z is the total list of durations for a given phoneme in a word. For each word, the output value is generated by normalising the aggregated word value by the number of stress values in the word. The total fitness score generated by this fitness function is the normalised sum of values generated for each

word and its corresponding note durations in the melody/song. It is important to note that the functions are only considered to aid in understanding the fitness function, and therefore have a simplified representation. The handling of occurrences where the word is not found in the dictionary is for example disregarded in the formula representation.

$$g_1(y,z) = \begin{cases} 1.00 & \text{if } y = max(z) \\ 0.25 & \text{if } y > min(z) \\ 0 & otherwise \end{cases} \quad \text{and} \quad g_2(y,z) = \begin{cases} 0.50 & \text{if } y = max(z) \\ 0.25 & \text{if } y \geq min(z) \\ 0 & otherwise \end{cases} \tag{4.7}$$

$$g_3(y,z) = \begin{cases} 0.75 & \text{if } y < max(z) \wedge y = min(z) \\ 0.50 & \text{if } y < max(z) \\ 0 & otherwise \end{cases} \tag{4.8}$$

The objective fitness functions are presented in Table 4.9. The fitness ranking principle is the same as in objective 1 and objective 2, where scores from each fitness function evaluation are compared to a target value, using Equation 4.3, and aggregated into a total fitness score. As opposed to the approaches in objective 1 and objective 2, no sentiment based target values are added, as the sentiment value was deemed irrelevant for the low-level lyrical evaluation proposed in this objective. It was also deemed unnecessary to implement punishment functions, such as done in the previous objectives.

4.5.4. Genetic operators

Multiple genetic operators were implemented as to generate and diversify offspring based on the current population. The reasoning behind the presented approaches, such as operator probabilities is presented in subsection 5.2.2.

Crossover

The first step in offspring generation is to combine the genes of two phenotypes from the previous generation. This is handled by the function **random measure crossover**. The function combines genes from two phentoypes by firstly separating each phentoype's melody genes into measures. Genes for the recombination are chosen with a semi-random approach. Until the maximum length of one of the phenotypes is reached, a random number of measures from each phenotype is added to the offspring with a positional approach. This means that if measure 1 and 2 from the first phenotype is added to the offspring, then the random number of measures added from the second phenotype starts off from the third measure index of its melody genes. The next phenotype to add measures from is chosen at random for each combination iteration, meaning that even though measures have been added from one phenotype already, the same phenotype could be chosen for combination in following iterations.

When the maximum length of one of the phenotypes is reached, remaining measures are added in random increments from one of the two phenotypes at a time, but with a disregard for the previously defined index positioning approach.

Chords are added based on their corresponding measure, i.e. chords are, by design, associated with a specific measure.

As an extra input variable for the crossover operator, a possibility for shuffling the notes in the measures was implemented. This is done as to further diversify the gene pool, as the combination and recombination of measures is done in large portions, which could increase the chances of the optimisation algorithm getting stuck at local maximum solutions. The shuffle function shuffles the notes in a measure by finding random notes and combinations of notes in beats that have the same duration, and swapping their places. The probability of the crossover operator utilising the shuffle method is set to a 50% chance.

The function continues the recombination until the melody length $l \geq n$ number of syllables in the lyrics. If $l > n$, the exceeding number of notes are pruned from the offspring melody. The necessity for the second round of measure recombination and the following melody pruning is a consequence of the variable genotype sizes of different phenotypes, as described in subsection 4.5.1.

Mutation

Of the offspring phenotypes generated by the crossover operator, 30% are chosen for mutation. The reasoning behind such a high mutation possibility was a necessity for diversifying the offspring, as the offspring tended to have high gene similarity already at early generations.

The different mutation operators are presented below. For the offspring chosen for mutation, one mutation operator is applied, but in cases where one operator fails to apply a mutation, another mutation operator is applied, following the same probabilities. This happens in special cases, such as e.g. when the operator tries to mutate durations for a beat that is empty. The probabilities of which operator to apply to an offspring is specified in the parentheses next to the operator name in the description below.

Mutate random note (5%) The operator finds a random note in the melody and overrides it with a new, randomly chosen one within legal bounds. The initial note duration is kept equal.

Mutate random scale note (25%) This operator finds a random note in the melody and overrides it with a new, randomly chosen one within legal bounds and within the scale of the global key signature. The initial note duration is kept equal.

Mutate beat durations (15%) This operator selects a randomly chosen beat, with a length $l > 1$, and shuffles the note durations in the beat, i.e. notes stay the same place in the beat, but swap durations.

Swap random notes (15%) The note swap operator chooses two random notes in the melody, having an equal duration, and swaps their places.

Mutate random scale chord (20%) This operator chooses a random chord in the chord progression and overrides it with a new chord. The new chord is generated by

finding a random root note from the global key signature, and generating its valid triad chord based on the chord scale degree relationship, described in Table 4.5. No possibility for a fourth flavour note is implemented by this operator. The main reasoning for keeping the totally random scale note mutation operator, and not having a corresponding randomness in chord mutation, was that such chord changes would drastically punish the phenotype in fitness, which could disqualify an otherwise well ranked phenotype.

Swap random chords (10%) This operator selects two random chords from the phenotype, and swaps their places.

Mutate chord flavour note (10%) This operator adds a fourth flavour note for a chord of length $l < 4$, or overrides the flavour note of a chord with a length $l = 4$. The chord to modify is randomly chosen from the phenotype chord genes, and the flavour note is generated at random from the scale of the chord root note. The root note scale follows a major pattern or a natural minor pattern, which is decided at random.

4.5.5. Termination

The algorithm is set to terminate after a given number N generations is reached. The number is set in advance. A possibility for the algorithm to terminate if the number of *Pareto fronts* have been of size = 1 for 100 generations was also implemented. This happens when none of the phenotypes in the population dominates other phenotypes, i.e. they are all pareto optimal. The concept of pareto fronts is covered in subsection 2.2.1.

4.6. Algorithm output

The algorithm output is generated based on the genes of the phenotypes chosen for output. In the main implementation, five phenotypes are chosen for output after each 100 generations, as well as when the algorithm terminates. The phenotypes are chosen at random from the winners of the tournament sort used in choosing parents for offspring, i.e. they are randomly chosen from the parent population.

For each phenotype a *LilyPond* (subsection 2.4.4) file is generated. The file contains all information about the melody and chord genes, as well as the lyrics. The file may be compiled into multiple formats, but for the main usage of this algorithm, the most relevant file formats are *PDF, SVG and MIDI* file compilation. File types are chosen manually.

5. Experiments and Results

This chapter is intended to give the reader an insight into the conducted experiments and the results generated as part of the project research. Firstly the experimental plan of the project is presented in section 5.1, then a description of each conducted experiment and their corresponding results are presented in the following sections, arranged by experiment topics. Testing of architectural decisions and results are presented in section 5.2. A questionnaire and the generated results are presented in section 5.3 and section 5.4.

5.1. Experimental plan

The first part of the experimental plan of the project is concerned with evaluating different architectural approaches that were considered during the developmental phase of the project. This includes reasoning behind architectural decisions such as fitness function design, genetic operator design and probabilities, as well as an overview of the reasoning behind the general data structures. The plan also includes testing conducted in trying to amend the fitness ranking scheme collapse, mentioned by Olseng (2016), that also was encountered during this project. The plan of testing various architectural approaches is an important aspect of providing results for *RQ1*, presented in section 1.2.

The second part of the experimental plan includes a questionnaire sent out to evaluate the quality of the output generated by the system, as well as give an indication of the system's music composition potential. The reasoning for using a questionnaire based evaluation is to get a more extensive view of the musicality of the generated output, when evaluated objectively, rather than with subjective evaluations of the system. The questionnaire evaluation is directly related to generating results for *RQ2*, presented in section 1.2.

5.2. Testing of architectural approaches

This section is intended to present testing conducted on various architectural approaches and problems encountered during the developmental phase of the project, as well as give an insight into the test results and the decisions made as a consequence of the results. The testing conducted on architectural approaches could be considered as having generated qualitative results, as the tests were entirely performed through subjective observations by the author. A thorough quantitative evaluation of the implemented system architecture, as a whole, is presented subsequently in section 5.3 and section 5.4.

5.2.1. General setups

The testing of general setups consists of the evaluation of architectural approaches not directly concerned with the evolving nature of the system, and is as such comprised of the testing of initial setups.

Beats vs. measures

Two setups regarding the melodic genotype representation were tested as part of the project. Firstly the melodic genotype was designed to be initiated and recombined in the crossover function with a beat to beat approach. This means that the duration of the combined beats of each measure was not considered in the melody generation, and that duration limitations were only put on beats, i.e. a beat should have a duration of a quarter note. When a duration longer than a beat was generated, empty beats were appended to the given beat, to signify beats covered by the duration. The biggest problems with this approach became clear as dotted notes were introduced and when evaluating the placement of whole notes, when notes regularly got irregular durations, with regards to the regular beat concept. Though generating subjectively interesting melodies, it introduced major syncopation occurrences in the melodies, i.e. notes did not consistently appear at rhythmically expected positions in the melody. An example of said syncopation issues can be seen in Figure 5.1, where introduced whole notes offset the melody in regards to the chords.

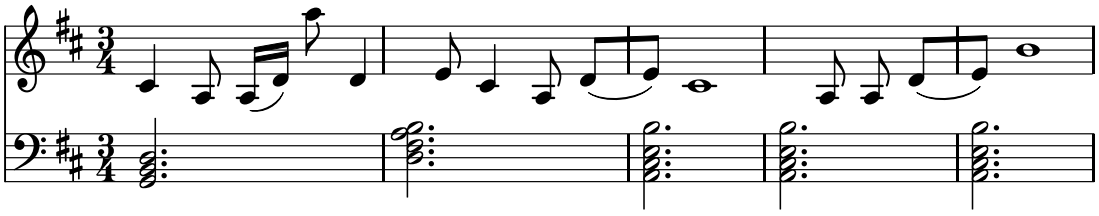

Figure 5.1.: 5 measures generated with beat representation.

Another setup was considered, where the melody genotype, though still represented as an array of beats, would be more strictly treated with regards to measures. The initialisation of random durations for each melody genotype would, rather than generating durations for each beat, generate durations that would fill entire measures. The crossover operator was also changed to recombining the genes from parent phenotypes by combining beats of measure size. The desired outcome of these changes was that the melodies would be less syncopated in nature, as well as increasing the melodic stability. An example of a melody line generated with a measure based representation is shown in Figure 5.2, where one can observe that even though there are dotted notes, and some notes with a longer duration than one beat, the melody is not offset.

The results from the change from beat handling to measure handling showed that the melodies were much more grounded in the time signature, and that notes appeared at expected places in the melody to a much larger degree. The change did though present

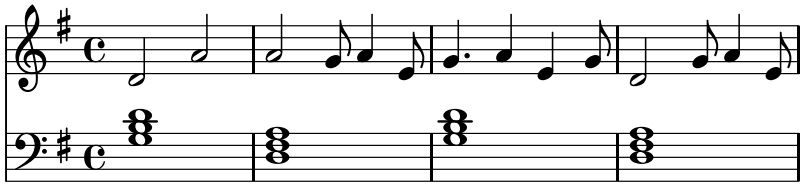

Figure 5.2.: 4 measures generated with measure representation.

a new problem in which the crossover operator combined large gene selections from each parent, such that a large degree of measure repetition occurred regularly. This did impair the generation of varied music to some extent, and introduced a possibility for the algorithm to converge on local maximum solutions. This problem is further addressed in subsection 5.2.2.

Sentiment analysis

To arrive at useful sentiment values for generated melodies, much testing was involved to improve the sentiment analysis of the input lyrics. As a basis for lyric testing and having some reference points of appropriate sentiment values, the valence score values from a data set presented by Malheiro et al. (2018) were used. The main testing method was manual evaluation of a set of lyrics and their corresponding data set values.

The first approach to sentiment analysis was to test how the unmodified Vader (Hutto and Gilbert, 2014) algorithm evaluated a set of lyrics. Two approaches were tested, where the first approach used Vader to get a valence score for the complete lyrics, and the other used a normalised aggregate of valence scores found for each line in the lyrics. The solutions showed similarities in their results, but the aggregated sentence value approach was ultimately decided to be the best of the two, with regards to values from the data set. An important note in the comparison between data set values and Vader values is that Vader presents the valence score $v \in [-1, 1]$, whereas the data set presents it as $v \in [-4, 4]$.

The results from the first approaches yielded inconsistent results, and attempts were made to find a solution that could improve the consistency. The approach of having two extra thresholds for valence score boosts, previously presented in section 4.2, was implemented. When increasing the importance of the most extreme valence scores, with an emphasis on the negative scores, the algorithm yielded more promising results.

An example set of results is presented in Table 5.1, where the three solutions are compared, and the distance to the data set values are calculated by Equation 5.1. The minimum distance values are marked with **bold** text for each row.

$$d = \left| \text{algorithm valence} - \frac{\text{data set valence}}{4} \right| \tag{5.1}$$

The division by 4 is done to get the relative placement of the data score $v \in [-1, 1]$. The songs chosen for the example are in sequential order *"Moon Song"* by Norah Jones, *"Mother"* by John Lennon and *"Ode to my family"* by The Cranberries. Note that

different songs might provide differing results, but manual reviews conducted by the author suggest that the modified sentence value solution provides the most consistent results.

Table 5.1.: Sentiment result examples

Song	**Whole text value**	**Sentence value**	**Modified sentence value**	**Data set value**, v = $\frac{\text{value}}{4}$
1	-0.026, d = 0.451	0.007, d = 0.418	0.1, d = **0.325**	1.7, v = 0.425
2	-0.057, d = 0.518	-0.001, d = 0.576	-0.1, d = **0.475**	-2.3, v = -0.575
3	0.982, d = 1.357	0.200, d = **0.575**	0.3, d = 0.675	-1.5, v = -0.375
AVG(d)	0.775	0.523	**0.475**	

5.2.2. Genetic operators

Several approaches to genetic operators were tested, with the main evaluation criteria being how they avoided generating generic, i.e. overly similar, phenotypes, as well as avoiding local maximum solutions.

Crossover

As mentioned in the paragraph concerning beats vs. measures in subsection 5.2.1, two main crossover solutions were implemented, one for recombining genes in beat divided chunks and one for recombining genes in measure divided chunks. Only the latter will be considered in this paragraph, as measure recombination was the solution used in the system architecture, with the reasoning being presented in subsection 5.2.1.

The local maximum problem, described in subsection 5.2.1, was the main concern when experimenting with the crossover operator. Initial testing of the operator showed that after an increasing number of generations, the melodic genotype of the phenotypes suffered from many repeating note patterns, and an alarming number of phenotypes contained highly similar genes, i.e. a local maximum was likely. As to circumvent this problem a solution was tested that would attempt to locate two beats in each measure of the melody, and swap their places. The swap was only to be done if the beats had an exactly equal duration, which decreased the chances of a swap occurring in all measures.

The probability of a swap occurring during recombination was set by a parameter passed into the function, before execution. The parameter was set to 50%, which immediately yielded promising results, and the parameter value was hence not changed. With a 50% chance of beat shuffling within measures the phenotype gene variation showed an increase, and the exact repetition of measures in a melody decreased.

Mutation

The different mutation operators were chosen based on a subjective evaluation of reasonable possibilities of melody and chord progression changes, with some inspiration taken

from Olseng (2016). The operator probabilities were set based on their importance in avoiding local maximum solutions and a subjective evaluation of their profitability in regards to music generation.

The tests conducted on the main mutation operator were done as a consequence of a discovered risk where the genes, most importantly chord genes, showed little evolution in the succeeding generations after approx. 100 generations, often indicating that a local maximum solution had been reached. These findings were supported by e.g. discovering that a large amount of solutions had a significant lack of chords with tonic root notes, which is inherently undesirable behaviour.

Experiments were then conducted on increasing the probability of offspring mutating. A higher mutation probability made sure that when a higher number of generations was reached, phenotypes continued to evolve to e.g. increase their amount of tonic triads, as the introduction of tonic triad chords to the genes immediately increases the fitness evaluation score of objective 3. The initial mutation probability was set to 5%, which was then step-wise upped to the 30% probability used by the system architecture based on evaluations by the author.

5.2.3. Fitness approaches

The development of suitable fitness functions was a crucial part of the system architecture design, presented in chapter 4. Multiple fitness approaches were considered for the architecture, and this subsection presents some of the most important approaches that were tested.

Punishment vs. normalisation

From the chapter describing the system architecture, one can see that objective 1 follows a simple punishment/reward system, i.e. the fitness conditions either punish or reward a phenotype with numerical values, while objective 2 and 3 both follow a ranking scheme where most of the fitness functions are based on normalised observations of the phenotype qualities rewarding phenotypes based on a set of target values, but also having extra *punishment* functions punishing seemingly bad qualities, which can be considered a hybrid approach. Objective 4 follows a strictly normalisation based approach, where all fitness rewards are given based on the target value based quality evaluation.

Objective 1 and 4 were never changed from their setups presented in chapter 4, and the only testing conducted on these objectives was in regards to which fitness functions to include, and in objective 4's case tweaking of the target values. These decisions were fully based on qualitative evaluations by the author on fitness function importance, in addition to how well different melodic aspects were covered in other objectives. For example a normalising fitness function from objective 2 regarding the number of non-scale notes in the melody was removed, as the concept was deemed well covered by objective 1.

For objective 2 and 3, two major different fitness approaches were tested. For objective 2 an approach without the melodic punishment approach, where the number of

repeated notes and the number of whole and 16th notes was first handled by the target value based functions, was initially tested. The testing conducted on this approach yielded unsatisfying results, where the author's subjective evaluation suggested that the probability of melodies being generated with unnatural note duration combinations was too high, and that the number of notes having extreme durations was too high in general, i.e. the number of whole and 16th notes that could impair the ability to sing along with the melody was problematic. The proposed change was to implement and move some of the functionality to *punishment* functions that would more severely punish melodies with an irregular number of extreme note durations, and especially punish the repetition of said notes. Subjective evaluation by the author suggested that the implementation of the punishment functions showed significant improvements in the melodies, although some voluntary comments from respondents of the questionnaire described further in section 5.3 and section 5.4, specifically suggest that the system generated melodies with rhythmical problems related to extreme note durations.

For objective 3, the first tested fitness approach was a more punishment based system, similar to the one used by Olseng (2016) in his *Harmonization* and *Harmonic Progression* objectives. This approach also included another objective, i.e. the lyric based objective 4 was not fully implemented and the harmonisation was separated into two separate objectives, where one followed a one chord at a time evaluation approach and the other evaluated the progression as a whole. The initial lyric testing was included into the progression objective. The progression objective followed a hybrid target based normalising and punishment approach to the one mentioned for objective 2, whereas the one chord at a time approach was fully punishment based. As for objective 2, the use of a hybrid approach was added based on subjective evaluations done by the author, suggesting that the chord progressions had a high likelihood of being generated without being thoroughly grounded by the tonic of the key signature. The change into using punishment functions to punish progressions with a too small number of tonic triad chords, as well as punish progressions with a large degree of repetition, subjectively yielded better results.

The reason for later combining the chord based fitness objectives into one objective and the separation of the lyric based functions into a separate objective was mainly based on amending a problem regarding the main ranking scheme, described in detail in subsection 5.2.4, although it was also desirable to test whether separating the fitness functions regarding optimisation for lyrics would add to the melodies' lyrical suitability. The results from subjective evaluations on separating the lyrical fitness evaluation into a separate objective indicated that the fitness of lyrics was given more importance in the evolution process, and that there was a slight increase in melodies evolving to be better fitting to the input lyrics.

Sentiment based target values

A key element in implementing the target value based fitness functions, as well as having them react correctly to the given sentiment, was to provide the algorithm with a set of profitable target values. The target values, as well as what target values to alter based on the given input sentiment value, were decided entirely based on subjective evaluations by

the author by multiple testing iterations. Reasoning for the chosen sentiment changeable target values was general assumptions, such as "sad" music being more dissonant, as well as containing a higher number of decreasing intervals, and "happy" music being more consonant. The main sentimental observation done by the author was that the biggest sentimental difference was seen when changing between major/minor keys, and that the variable target value approach only served as a minor supplement to the key choice.

5.2.4. Ranking scheme collapse

Olseng (2016) mentioned a problem with the ranking scheme of his system collapsing when the number of fitness functions increased. The collapse was caused by having no phenotypes dominating the others, i.e. they were all non-dominated, in regards to the non-dominated sorting approach adopted by the *NSGAII* algorithm described in subsection 2.2.2. This meant that all phenotypes were considered pareto-optimal, although not necessarily being fully evolved solutions.

A similar collapse was observed in the initial tested implementations of this project, where the number of pareto fronts would decrease to 1 within approx. 150-250 generations. The population had not yet evolved into subjectively good melodies, but all individuals had evolved into melodies following general musical rules, such as revolving around chords and the key. The main problem observed with the collapse was that only the crowding distance sorting would choose the parent population, meaning that the non-dominated sorting had no effect. This effect indicated that it was no guarantee that individuals chosen for reproduction were the best, and it seemed that further evolving, at this point, was up to the mutation operators.

Two particular solutions were tested to prevent the algorithm from converging into one pareto optimal front too soon. Instead of implementing a solution like the one used by Olseng (2016), where fitness duplicates were eliminated, a restructuring of the fitness objectives was tested and applied. As such, a solution of removing separate fitness objectives for chords, mentioned briefly in subsection 5.2.3, was tested. The core change proposed was to combine two separate chord objectives into one, where the mainly punishment based objective of handling chords individually in the evaluation was added into the chord progression objective. The most important fitness functions, such as the check of valid scale degrees in chords, were added to the harmonic punishment functions of objective 3, as well as the ones of less importance, such as semi-tone distance in flavour notes, being added as target value based normalising functions. In addition to this change, all lyric optimisation functions, previously found in objective 3 (harmonic progression), were separated into the now used objective 4, as described in chapter 4. The separation was intended to introduce a more clear aspect of domination within the phenotypes' lyrical optimisation, as well as attempting to reduce the number of fitness functions within each objective.

The results from the change indicated that the algorithm would run for longer until converging on one pareto optimal front. The generation count would increase to approx. 300-500 generations until the number of pareto fronts first reached 1. Another improvement included that even when converging on one pareto front for a number of

consecutive generations, the algorithm could re-evolve into having multiple pareto fronts within few generations. This was opposed to the results found before the change, where the algorithm would rarely recover from converging on one pareto optimal front. During testing it was also observed that the length of the input lyrics, and consequently the size of the genotypes, was a key factor in determining the number of generations needed to converge on pareto optimal solutions.

To further increase converging time, an attempt was made at increasing the population size N from $N = 1000$ to $N = 1500$. The results gathered from this change indicated that it would increase the converging time to some degree, but the results were not consistent for all cases (generations). The change was anyhow kept when generating melodies for the questionnaire, described in section 5.3, as is could be of some help in increasing the converging time, as well as improve phenotype diversity.

5.3. Questionnaire design

A questionnaire was designed with an intent of generating a quantitative evaluation of the system's ability to generate lyric-based melodies. The main goal of the evaluation was to achieve valuable insights into the quality of the generated output, as well as getting an indication of the system's usability in real world music composition tasks.

5.3.1. Algorithm setup and generated melodies

As a preliminary step in the questionnaire design, five melodies for five different input lyrics were generated by the system described in chapter 4. For each of the generated melodies, the same input parameters for the system were used. The generation count G was set to 1000 iterations, and the population size P was set to 1500 individuals. The input parameters were based on observations done by the author in relation to how the melodies seemed to evolve over time, as well as results based on the ranking scheme converging observations mentioned in subsection 5.2.4.

The generated melodies for the questionnaire were chosen from the subjectively best evolved phenotypes, i.e. the best of five phenotypes was chosen by the author for each of the input lyrics, when the algorithm terminated. The melodies, as well as the lyric based input values for the system are presented in Table 5.2. The lyrics are listed in the same order as they appeared in the questionnaire with their corresponding title. All generated melodies, their corresponding lyrics and sheet music, as well as selected questionnaire results are presented in Appendix B. A specific appendix marker is appended to the corresponding title in Table 5.2. The table also presents the sentiment values, musical keys and time signatures for each of the different melodies generated by the system. The question template presented in subsection 5.3.2 was then used as a basis for the questions regarding each of the generated melodies in the questionnaire.

Table 5.2.: Melodies generated for questionnaire

No.	Title	Sentiment value	Key	Time signature
1	Nellie Dean (B.1)	0.3	G-major	$\frac{4}{4}$
2	Bridget O'Malley (B.2)	0.1	E-major	$\frac{3}{4}$
3	Early one morning (B.3)	0.0	G-minor	$\frac{3}{4}$
4	Henry Martyn (B.4)	0.0	E-minor	$\frac{4}{4}$
5	Billy Lyons and Stack O'Lee (B.5)	-0.4	G-minor	$\frac{3}{4}$

5.3.2. Questions

The questionnaire was designed with an intent of being readily understandable, without being designed for a specific target group. The intention was that a participant's age and computational or musical background should be of no constraint, when providing answers for the questionnaire. The questionnaire was inherently anonymous, and only insignificant personal questions regarding the participants background were asked, namely questions with respect to the participants' musical interests and abilities.

The questions for the questionnaire are listed below, accompanied by general descriptions of the questions, as well as a description of the associated evaluation criteria. Aside from specifically annotated questions, all questions were to be given an answer on a scale of integers in the interval $[-3, 3]$ with correlating descriptions, presented in Table 5.3. The questionnaire included no mandatory questions, to lessen the likelihood of participants getting stuck or quitting, when finding a question to hard or incomprehensible to answer. All questions were originally presented in Norwegian, and as such, the list of questions and score ratings are translations done by the author. The original questions and score ratings can be found in Appendix A.

Table 5.3.: Score rating scale

Score:	Correlated wording
-3	To a much lesser degree
-2	To a lesser degree
-1	To a minor degree
0	Neutral
1	To a certain degree
2	To a greater degree
3	To a much greater degree

Introduction

The initial questioning, presented below, was intended to get an overview of the participants' previous experience with music and composition, as well as obtain an overview of their ability in reading the sheet music accompanying each melody. The ability to differentiate between experienced and less experienced participants was regarded as an important aspect of evaluating the questionnaire responses, as answers could be highly dependent on the participants' experience levels.

1. Do you listen to music often?
2. Do you ever compose music?
3. Do you know how to read sheet music?

Questions regarding melodies

The below presented question template was used for each melody generated as part of the questionnaire, and was arguably the most integral part of the quantitative evaluation process. Along with the question set, the participants were presented with a video containing the melody gathered from a MIDI-file, as well as the corresponding sheet music for the generated melodies. The first question in the template was intended to determine to what degree the participants were able to be objective in their melody evaluation. If they had heard the actual song and corresponding melody previously, it could have impacted their evaluation of the generated melody. For each melody the participants were able to leave an extra textual comment about the melody. This ability was added as to bring clarity into the participants' reasoning behind their answers, and hence improve the author's ability to analyse and understand the data provided by the questionnaire. A description was linked to each of the questions regarding melody and lyrics, shown for each question below. The main evaluation criteria covered by the questions are presented in the description following the question template.

1. Objectivity
    a) Have you heard the original song before? (yes - no - don't know)
2. Melody:
    a) Is the melody musically pleasing?
        - Does the music follow musical rules/norms?
        - Do you think the melody is pleasant to listen to?
    b) Is the melody interesting?
        - Do you think the melody demonstrates interesting qualities, or is it boring/meaningless?
    c) Does the musical structure make sense?

- An overall impression of the melody.
- Does the melody have a natural progression, or does the melody e.g. show sudden/unnatural changes?

3. Lyrics and melody:
   a) Does the melody leave a fitting emotional impression (sentiment), based on the lyrics?
      - Are the lyrics sad or happy? Does the melody reflect this?
   b) Do the words and syllables fit with their appointed notes?
      - Are there any unnatural differences between the rhythm and the notes in regards to words and syllables?
4. Voluntary comment

Musically pleasing This metric is intended to clarify to what extent the generated melody is musically pleasing. It is concerned with determining whether the melodies follows musical norms/rules, such as keeping well within the defined key signature. The metric of musical pleasantness is also used to cover whether the melody sounds pleasing to the listener in general, i.e. not having large interval spans, as well as a minimal number of meaningless dissonant notes.

Interesting melody The interesting melody metric is concerned with the evaluation of the melody with regards to its interesting qualities. The metric is intended to detect whether the generated melodies are considered boring and uninteresting, and differs from the above general musicality metric, as a melody could be pleasing, or rather not displeasing, but simultaneously be less interesting. The metric could give valuable input with regards to the general quality of the system, as well as give an indication about the melodies' inherent creative qualities.

Structural integrity This metric is concerned with detecting whether the melodies are assembled by consistent parts, i.e. detecting whether the different parts of a melody have a correlation and do not seem to appear at random. The metric could be valuable in determining the degree to which the system is able to generate coherent melodies.

Sentiment correctness The sentiment correctness metric is solely concerned with evaluating the extent of it being a clear correlation between the lyrics provided for the melody generation and the generated melody. It gives an indication of how well the system is able to impose sentiment analysis on the lyrics, and then properly benefit from the analysis in the generation phase.

Syllable correctness The syllable correctness metric is adopted to clarify whether the input lyrics properly and naturally aligns with the generated melody. The metric could return valuable input with regards to the rhythm generation, as well as

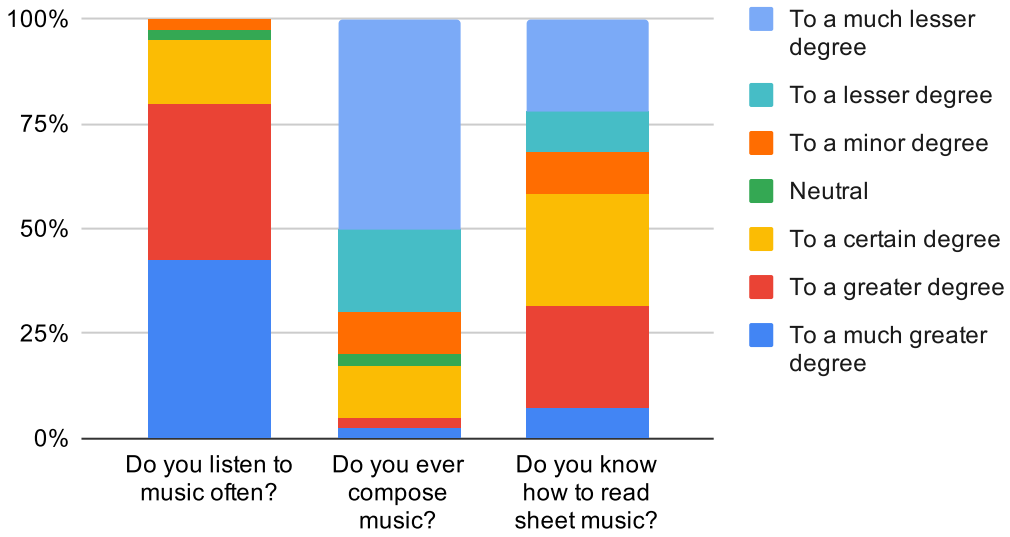

Figure 5.3.: A chart showing the distribution of respondent music experience

serve as an evaluation metric of the impact the implementation of fitness objective 4 (Lyrics optimisation) has had on the system.

Overall impression

The last set of questions was intended to bring clarity into the system's overall melody generation potential. The questions are concerned with determining to what extent the system could be used in composition tasks. The participants received no specific information regarding the run-time of the algorithm when generating the melodies, and hence the below questions were designed to be answered with only the music/lyric aspect of the melodies in mind.

1. Could the system work well for composing on its own?

2. Could the system work well as assistance during composing?

5.4. Questionnaire results

The questionnaire was answered by a total of 42 participants, where most of them answered all the questions listed in the questionnaire, except for question 4, "Voluntary comment", which were answered by 3-12 participants for each melody. The extreme minimum number of answers for each of the remaining questions was 37.

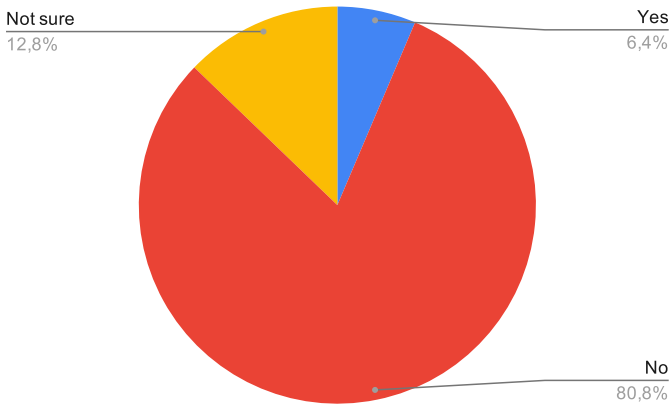

Figure 5.4.: A chart showing the distribution of respondents having heard the original songs of the lyrics (blue: yes, red: no, orange: not sure)

Introduction

The distribution of previous musical experience for the questionnaire participants is shown in Figure 5.3. The distribution shows that a vast majority of the participants listen to music often, but only a minor fraction of the participants answered that they regularly compose music on their own. More than 50% of the participants states that they have some ability in reading sheet music, with a third of the participants stating a high sheet music reading ability.

Questions regarding melodies

The results from the questions regarding the generated melodies are presented in Figure 5.4, Table 5.4 and Table 5.5.

The aggregated average value of the distribution of the participants previous knowledge of the original songs, of which the input lyrics are sourced from, is presented in Figure 5.4. The chart was generated by summing the answers of question 1 a), regarding objectivity, for each melody and compute the answer fractions. A percentage description of the combined answer fractions for each specific melody is presented in Appendix B.

In Table 5.4 results regarding the different aspects of the melodies are presented. The different aspects are *Melody*, which is comprised of the three questions regarding the melody and *Lyrics and melody*, which is comprised of the two questions regarding the lyrics and melody, presented in the questionnaire template. The *Melody* aspect is related to the evaluation metrics *Musically pleasing, Interesting melody* and *Structural integrity*. The *Lyrics and melody* aspect is related to the evaluation metrics *Sentiment correctness* and *Syllable correctness*. The total values from all melody questions are shown in the last column of the table.

For each column the *AVG* value is the average scoring of the questions of the given aspect, and *SD* is the average standard deviation result for the corresponding questions.

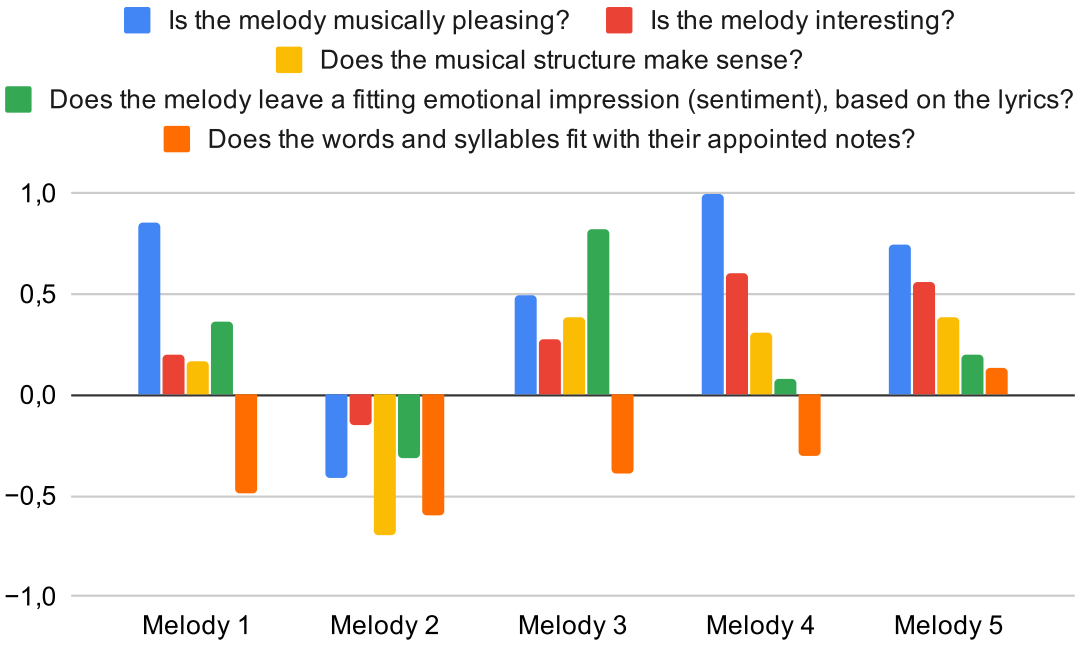

Figure 5.5.: A chart showing the average response for each question by melody

The melody order is the same as shown in Table 5.2. The value of each number can be decoded by using the value sheet presented in Table 5.3. A more detailed overview of the *AVG* and *SD* results for each question is presented for each melody in Appendix B.

Table 5.4.: Questionnaire results - melodies and lyrics

	Melody (AVG/SD)	**Lyrics and melody (AVG/SD)**	**Total (AVG/SD)**
Melody 1	0.4 / 1.2	-0.1 / 1.3	0.2 / 1.3
Melody 2	-0.4 / 1.4	-0.5 / 1.3	-0.4 / 1.4
Melody 3	0.4 / 1.4	0.2 / 1.5	0.3 / 1.4
Melody 4	0.6 / 1.2	-0.1 / 1.5	0.3 / 1.3
Melody 5	0.6 / 1.4	0.2 / 1.5	0.4 / 1.4

As a supplement to the values presented in Table 5.4, the average result value for each melody specific question is presented for each melody in Figure 5.5. The standard deviation metric is ignored for this chart, but can be found for each question and corresponding melody in Appendix B. The chart presents the results for the given melodies in the interval $[-1, 1]$, where the value follows the rating scale presented in Table 5.3.

The aggregated average results (AVG) for each question regarding the generated

melodies in the questionnaire, along with the standard deviations (SD), are presented in Table 5.5. The table presents the overall tendencies in the participants' answering, and the overall system scoring within the given question aspects. The questions are presented in row 1 by their questionnaire template question symbol. The total value, presented in the last column, is the total average value for all melody questions for all melodies, giving an overall score for all system generated melodies. The value of each number can be decoded by using the value sheet presented in Table 5.3.

Table 5.5.: Questionnaire results - aggregated for all generated melodies

	2. a)	**2. b)**	**2. c)**	**3 a)**	**3 b)**	**Total**
AVG	0.5	0.3	0.1	0.2	-0.3	0.2
SD	1.3	1.4	1.4	1.4	1.4	1.4

The results presented in Table 5.4 and Table 5.5 show that the aggregated scoring of the melodies tend to revolve around the value *Neutral*, as defined in Table 5.3. For all melodies except *Melody 2 - Bridget O'Malley*, the values are on the favourable, i.e. positive, side of neutral. The question based, rather than category based, representation of the results in subsection 5.3.2 supports the claim that *Melody 2* is an anomaly compared with the results for the other melodies. The results presented in the chart also show how the last question regarding syllable correctness consistently is the lowest scoring metric, except for *Melody 2*, where the structural integrity metric scores lowest. The results also suggest that the musical pleasantness and the degree of how interesting the melody is, are not highly correlated. Except for the results regarding *Melody 2*, the structural integrity is the most consistent metric.

Table 5.6.: Questionnaire results - Composition potential

	Standalone composition tool	**Assistance composition tool**
AVG	-0.1	1.1
SD	1.4	1.2

Overall impression

The distribution of the responses to the questions regarding the overall impression of the system's potential as a composition tool is shown in Figure 5.6. From the chart there is a clear increase in respondents seeing a potential in the use of the system as a composition assistance tool, rather than a standalone composition tool. Table 5.6 presents the average value and standard deviation for the overall impression questions. The average respondent results show that the average response is *Neutral*, but on the negative side, for the systems potential as a standalone composition tool, and that the average response in regards to the assisting potential of the system is *To a certain degree*, following the values presented in Table 5.3.

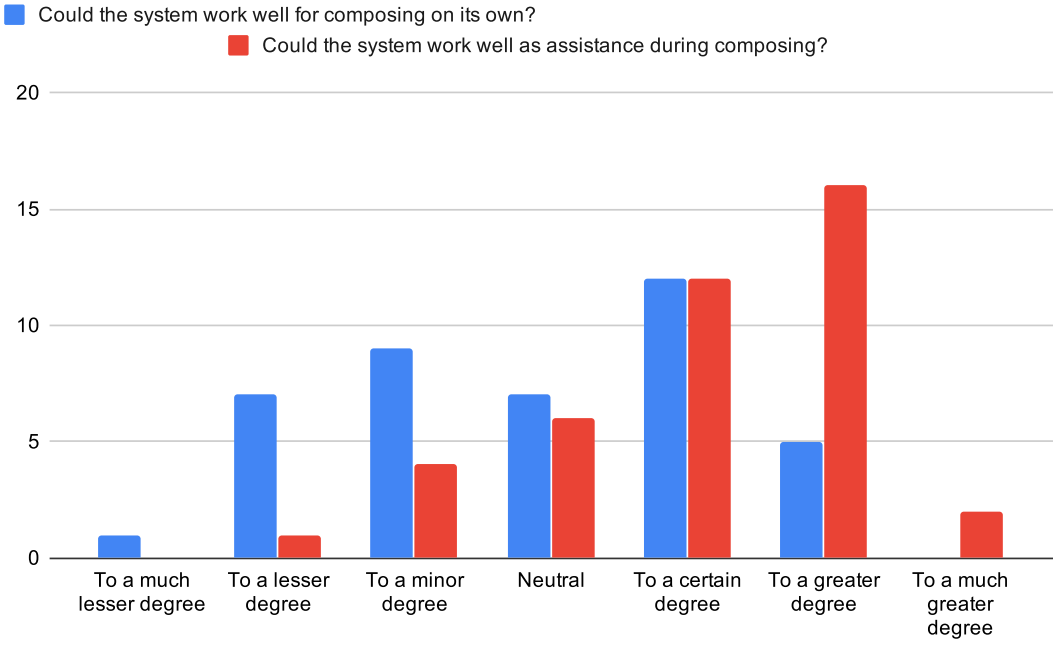

Figure 5.6.: A chart showing how respondents deem the composing qualities of the system

6. Evaluation and Discussion

The following chapter is intended to give the reader an overview of the conducted research, and presents a comprehensive evaluation (section 6.1) of the implemented system architecture, presented in chapter 4, in light of the experiments and results provided by chapter 5. Following the evaluation of conducted experiments, a thorough discussion of how the results and evaluation relate to the research questions and the main goal of the project, presented in section 1.2, is performed. The discussion describes the extent to which the main goal has been met, as well as the extent to which the research questions can be answered, by drawing lines between the results and evaluation of the implemented system and the literature review presented in chapter 3.

6.1. Evaluation

This section is intended to give the reader an insight into the evaluation conducted on the implemented system, based on the results and experiments presented in chapter 5.

6.1.1. Evaluating genotype design and genetic operators

As mentioned in subsection 4.5.1 and subsection 5.2.1, the genotype design was, after a subjective evaluation, grounded in a beat representation for note genes, where the beats were handled as measures during initialisation and crossover phases, as well as in fitness objective 1. The transition from having a strict beat representation to the hybrid method of abstracting beats into measures clearly showed that more stable melodies were generated, but since it established problems with not having a varied recombination (crossover), it can not be said to have produced unambiguously profitable results. The implementation of the swap possibility in the crossover functions improved the gene variation, but as mentioned in subsection 5.2.2, there was a clear similarity in the genes of many of the phenotypes in later generations, suggesting that the crossover operators could have been more diverse in their recombination. The inclusion of a global beat swap crossover operator, recombining single beats from the whole melody, rather than within measures, as is the case with the current swap parameter, could perhaps have generated increasingly better results.

As the number of mutation operators has been high, it has been difficult to properly evaluate their separate impacts, with regards to avoiding local maximum solutions, but from the gathered results, it is though reasonable to claim that their combined effect has been positive. The increase from a 5% chance of offspring mutation to 30%, proved a valuable change, as it increased the diversification of genes. It is a clear possibility that

some of the observed effects of the less diversifying crossover operator were mitigated by the increase in mutation probability.

The complicated nature of the genotype and genetic operator design may have increased the chance of the occurrence of developmental errors. One such discovered, but unmitigated error is connected to the fact that even though it should be no possibility of major syncopation, such as having no note on the first beat of a measure, with the current design, the phenomenon is still observable in some phenotypes for some runs of the algorithm. The source of the error has not been determined, but could have a relation to the pruning of notes at the end of genotypes where notes exceed the syllable count during recombination, or another process of beat/measure handling. A possible reason for not uncovering the error could be that the genotype and crossover design are too complex. The handling of phenotypes with variable genotype lengths, such as in this implementation, is unusual in general genetic algorithm approaches. For a general genetic algorithm, genotypes usually have a fixed length and a binary representation to circumvent such problems (Whitley, 1994; Yu and Gen, 2010). A less complicated genotype design could be to reintroduce a numerical approach, such as done by Olseng (2016), or to circumvent the implemented beat representation, and rather employ a measure by measure representation.

6.1.2. Evaluating fitness approaches

The most important findings from the fitness function experiments presented in subsection 5.2.3 suggest that the combination of using target value based descriptive fitness functions, in tandem with a set of punishment functions for the most decisive factors, functioned better than a purely punishment based or target value based approach. Even though the separation of lyric specific optimisation into a fourth objective did subjectively yield better results, the results and comments uncovered in the questionnaire, further described in subsection 6.1.4, suggest that the system still struggles to correctly optimise the melody for the input lyrics. A more strict focus on stress value to rhythm satisfaction, in the objective regarding lyrical optimisation could possibly have helped amending this problem. The nature of the stress satisfaction fitness function might have been too forgiving in its rhythm/stress evaluation, hence being too insignificant in the evolution. A strict initialisation of the melody rhythms, following a similar approach to the one used by Monteith et al. (2012), for each phenotype in the initial population could have improved the relationship between lyrics and melodies from an early generation stage, but it could be argued that such a solution would counteract the evolutionary principle of the system design.

A possible problem with the fitness function implementation is that the total number of fitness functions (including punishment functions) in the system architecture is 51, which for the largest objectives could have decreased the impact of the various musical concepts. The observation of this effect was the main reason for the introduction of melodic and harmonic punishment functions for the subjectively most important concepts. Further evaluation of which target based fitness functions to convert into punishment functions, as well as a consecutive reduction in the total number of fitness functions, could have

increased the quality of generated melodies.

The introduction of variable target values in the system design, with regards to the input sentiment value, generated some subjectively interesting results, but the results from the sentiment based target value testing are far from conclusive, as it is reasonable to assume that the most decisive factor in the melodic sentiment was whether the key was major or minor, rather than the sentiment based target values. To properly measure the effect of the sentiment based target value implementation is troublesome, as there is no specific way to differentiate between regularly generated melodies and melodies generated with sentiment based target values, as there is an inherently high randomness factor involved in the evolutionary nature of the system. There is though a high possibility that the implementation of variable target values has had a positive effect, as the values are specifically tuned for corresponding sentiment values. Further evaluation of the system's sentiment approaches is presented in subsection 6.1.4 and subsection 6.1.5.

6.1.3. Evaluating ranking scheme collapse solution

From the results presented in subsection 5.2.4 there is a clear indication that the change to a single objective for optimising harmonisation, as well as separating the optimisation of the melodies' fitness with regards to input lyrics into a separate objective, served as a solution to the mentioned ranking scheme collapse. From Olseng (2016) it is unclear exactly what is meant by collapse, i.e. at what number of generations the ranking scheme started to converge to one pareto optimal front. The assumption used in this thesis is that the collapse is meant to be a quick (i.e. within 100-200 generations) and unrecoverable converging of the algorithm into one pareto front.

With the solution presented by this implementation, the *NSGAII* algorithm can run for 1000 iterations, i.e. the main number of iterations supplied to the algorithm for the melodies generated for the questionnaire, without definitively converging to one pareto optimal front. In other words, the algorithm quickly (within approx. 100-200 iterations) begin to rank phenotypes into two separate pareto fronts, rather than a higher number of fronts, and after a varying amount of time it might converge to one pareto optimal front. An important aspect here is that even when having converged to one pareto optimal front, the algorithm may re-converge back into having two pareto fronts within a few iterations, indicating that the ranking scheme has not collapsed.

The solution can indeed be deemed successful, though having only two pareto fronts might also be considered sub-optimal, as a high number of phenotypes in the parent population would still have to be chosen based on crowding distance sorting.

6.1.4. Evaluating questionnaire results

This subsection is intended to evaluate the questionnaire results presented in section 5.4, with regards to the different aspects of system and questions, as well as discuss the quality of the generated results.

Participants

The wast majority of participants in the questionnaire stated that they often listen to music, but that they rarely compose music themselves. About half of the participants stated that they had some ability in reading sheet music. This gives the indication that most participants have a strong interest in music in general, but that few have been involved in the technical aspects behind musical composing. The sheet music metric gives an indication that at most half of the participants have been adequately able to follow the melody and the accompanied lyrics.

There was no intended main target group for the questionnaire participants, but as a consequence of the questionnaire being distributed through various social media platforms, having a direct connection to the author, there is good reason to suppose that the main demographic of the participants is people within the age range of 20-35 years, with a higher education level. It is also worth mentioning that since the questionnaire was distributed with a personal connection to the author, there is a possible bias within the participant demographic. People with a closer relationship to the author have probably had a greater motivation for participating in the questionnaire, rather than people with a more peripheral relationship. The anonymous nature of the questionnaire can though not guarantee such a distribution of the participants.

The main impact the suggested participant demographic might have had on the questionnaire results could be that the participant's responses might have been influenced by them having a more sympathetic bias towards the questions. It is though worth noting that the participants were given an explicit notice to answer each question as honestly and elaborately as they could.

As the total number of participants for the questionnaire was 42, one could argue that the number of participants has been too low to draw definitive conclusions from the data. Regardless, the results have provided valuable insights, and given indications of the system's potential.

Melodic quality

Looking at the average results for each melody, the first main observation is that the metrics regarding melody have scores revolving around *Neutral*. For all melodies, except melody 2, the *Musically pleasing* metric scores highest, and could be considered the only metric touching the *To a certain degree* scoring value presented in Table A.1. The melodies could be said to rank slightly lower regarding the metric *Interesting melody*, where only melody 4 and 5 have values slightly closer to the *To a certain degree* value than *Neutral*. The metric *Structural integrity* is for all melodies except melody 3, the lowest scoring metric, but for all melodies, except melody 2, still revolving around *Neutral*. For the metrics regarding melodic quality, the standard deviation varies from 1.2 to 1.4, which indicates that the responses, for the most part, have varied from *To a minor degree* to the value *To a certain degree*.

In general, there is a clear indication that the generated melodies are more musically pleasant than interesting, and that the melodies are more interesting than the degree of

maintaining structural integrity. Melody 2 clearly deviates from the remaining melodies, in having all metrics on the negative side of *Neutral*. The main conclusion one can draw from this is that the system cannot guarantee steady melodic qualities, but that the most likely outcome is generating melodies with melodic qualities on the positive side of *Neutral*. Having the melodic qualities revolve around *Neutral* indicates that the system, though not able to generate consistent, high quality music, can generate average ranking melodies. The fact that the generated music's melodic quality is average can definitely be deemed as a good result, as it shows that the system can generate melodies without drastic violations of musical concepts.

Lyrical relation to melodies

The average results from the questions regarding the metrics *Sentiment correctness* and *Syllable correctness*, both describing the relation between the melody and the given input lyrics, revolves around *Neutral*, but the metric *Syllable correctness* consistently scores lower than *Sentiment correctness*. For all melodies *Syllable correctness* is the lowest scoring metric, whereas the *Sentiment correctness* metric is more volatile, but has a score rating close to *Neutral*, except for melody 3, where the rating is closer to the value *To a certain degree*. The only melody achieving a positive value for *Syllable correctness* is melody 5, and for all other melodies the value moves towards *To a minor degree*, though keeping within the *Neutral* bounds for all melodies, except melody 2. For the metrics regarding the melodic relationship to the input lyrics, the standard deviation varies from 1.3 to 1.5, which is slightly larger than the standard deviation for the melodic quality metrics. The indication is still that most responses have varied from *To a minor degree* to *To a certain degree*.

There is a clear indication that the generated melodies have more properly matched the lyrical sentiment, rather than correctly fitting the lyrics' words and syllables. As for the metrics regarding melodic qualities, melody 2 deviates from the other melodies with regards to the lyrics/melody relationship. It is though interesting to note that the *Syllable correctness* metric has the smallest deviation from the results of melody 2 compared with the other melodies. Another important note with regards to the *Syllable correctness* metric is that a substantially higher number of respondents answered the questions regarding how well the lyrics fitted with the appointed notes than the number of participants specifying having a high competence in reading sheet music. This may have impaired their ability to provide high quality answers to the melodies' relation to the lyrics, and could therefore be considered a source of inconsistencies in the results.

The main conclusions to draw from the results is that there is a variation in how well the sentiment analysis has impacted the melody generation, and that the impact has neither been strong nor weak, as the values revolve around *Neutral*. This indicates that the system to some extent has been able to generate melodies with a correct emotional expression. The conclusion to draw from the results, with regards to *Syllable correctness*, is that the system is generally worse in generating well fitting melodies for the input lyrics than for providing melodies that satisfy the other metrics. The system does though not generate bad melodies with regards to the melody/lyrics relation, as the result values

still fit within *Neutral* bounds.

Insights from voluntary comments

Many participants chose to leave voluntary comments as part of their responses, but with a decreasing answer frequency for later melodies. The comments could give valuable insights into the participants' reasoning behind their responses, as well as give an indication of possible margins of error within the different question categories (melodic quality and melodic/lyrical relation). This could give valuable input in regards to uncovering possible misunderstandings, as well as indicate whether the posed questions were ambiguous to some degree.

As for the general elaboration the participants have given through the voluntary comments, most state problems with a correct connection between the lyrics and the melody as the biggest topic of concern, with a varying degree of severity for different melodies, which supports the findings discussed in section 6.1.4. One participant did e.g. state that a repeating pattern in the lyrics, probably would have some sort of repeating melodic connection. Multiple responses specifically stated that the words and syllables did not fit well with the note pitches and rhythms, as well as them having difficulties with singing along with the melodies. In addition the most prominent melodic problem had to do with the structural integrity. It was e.g. stated that the rhythmical complexity was too high, and that the note duration span was too large. For the last melody though, one participant stated that he was almost able to sing along with the melody, indicating some disagreements between respondents with regards to the severity of the problems with the rhythm/lyrics relation.

One respondent stated that it was hard to decide to what extent the melody was fitting to the lyrics, as there was no vocal track added for the melody, which could be a possible source of errors with regards to the *Syllable correctness* metric, since many respondents stated having less of an ability in following sheet music. The lack of a vocal track, and a possibly too vague problem description, may have been a source of misunderstandings of the main intention of the questionnaire. One respondent stated that the melody might have functioned better with a warmer piano sound, which is a musical concept not intended to be of any significance for either the questionnaire or the system scope. Another respondent indicated difficulties in determining what sentiment the lyrics posed, which consequently made it difficult to answer whether the melody fitted the lyrics' sentiment or not.

The only clear conclusion one can draw from the voluntary comments is that the system has had some issues with regards to making the melody fit well structurally with the lyrics, as it was mentioned by multiple participants. The remaining discussed concepts, derived from the voluntary comments, are too vague in nature, and have been mentioned by too few participants to give unambiguous results. The comments do though indicate that some misinterpretations of the questionnaire and the posed questions have occurred.

Overall impression

The results from the questions regarding the system's potential as a musical composing tool show that the average response in regards to the system's potential as a standalone tool in music generation is *Neutral*, but with a standard deviation of 1.4 and the spread seen in Figure 5.6, there is a clear indication that the respondents have been divided in their opinions regarding the potential. The standard deviation decreases to 1.2 for the question regarding the system potential as an assisting tool in musical composing. In addition the average goes up to the value *To a certain degree*, with the largest response group of $\frac{16}{41}$ stating that the system *To a greater degree* could be of assistance in music composing.

From the neutral average and large spread in responses, no definitive conclusion regarding the system's standalone composing potential can be drawn, though having the value revolve around neutral can be regarded as promising. On the other hand there is a more clear conclusion to be drawn regarding the system's potential as an assisting tool. There is a clear indication from the results that, in the eyes of the respondents, the system can be of assistance in musical composing tasks.

6.1.5. Evaluating sentiment analysis approach

The results gathered, regarding the tested sentiment analysis approaches from subsection 5.2.1, show that the modified sentence value approach generated the best results, but with an average distance of 0.475 in an interval [-1, 1] it has a close to 25% deviation from the valence score provided by the test data set. This deviation clearly indicates that the sentiment values were not highly accurate with regards to the data set values.

As Vader is a tool inherently designed to decide the sentiment in a social media context (Hutto and Gilbert, 2014), there is a clear possibility that the decision to use the tool has had less of an accurate impact on the sentiment analysis of lyrics, than could have been the case with different approaches. The decision of not including more lyrics for the initial testing and tweaking of parameters for the modified sentence value solution of the implemented approach might have negatively impacted the possible results, i.e. it might have been a contributing factor to the 25% deviation from the data set valence score evaluation.

Supposing a perfect translation of the sentiment value found by Vader into the generated melodies, the results from the questionnaire do however indicate that sentiment value has been helpful with regards to generating melodies with correct emotional expressions.

6.2. Discussion

The following section provides a discussion of the research questions, presented in section 1.2, and discusses the answers found in regards to each question. Lastly a discussion of the degree to which the main goal of the project has been achieved is provided.

6.2.1. Research question 1 (RQ1)

By using an evolutionary algorithm approach, in what ways could a melody be automatically generated, as to fit given lyrics, as well as capture the lyrics' sentiment?

The most important aspect in providing a satisfactory answer to research question 1 has been to explore multiple solutions and technologies to problems regarding computational music generation, which has been done by conducting a preliminary literature review, presented in chapter 3. The chapter presented two of the most prominent technologies within state-of-the-art research of computational music generation in detail, as well as providing an in-depth overview of research within music generation based on textual inputs. A brief overview of previous solutions of sentiment analysis of music, both in audio and lyrics analysing, was also provided.

As the research question states that the approach should be evolutionary, more weight has been applied to evolutionary approaches of music generation, but other solutions, such as artificial neural network approaches and lyrical approaches not relying on evolutionary methods, were explored as to provide further ideas, as well as give a thorough overview of the field of research.

Regarding evolutionary algorithm approaches, the most discussed topics in the research revolve around how to handle the "fitness bottleneck", mentioned by Biles (2001). As stated by Johnson (2012), the most prominent approaches within this area are human based and autonomous fitness functions, where human fitness functions, though having a higher quality of the evaluation, are much more ineffective. With regards to fitness functions in music generation, a solution proposed by Biles (2001) was to remove fitness functions, and rather focus on well tuned genetic operators. This seems to have had little effect on further research, and most research within evolutionary approaches to music generation are based on the use of autonomous fitness functions. One commonly used approach to autonomous fitness functions for music generation revolves around incorporating musical concepts, which was a highly adopted approach by the system design of this thesis, as all implemented fitness functions have a direct musical evaluation basis, except for the fitness functions regarding lyrics. Approaches regarding implementing artificial neural network (ANN) designs for fitness functions were also considered for the system design, but were deemed too complicated for the task at hand. The concept of using ANNs as bases for fitness is further discussed in section 7.3.

A clear indication from the literature review was that a highly popular approach in state-of-the-art evolutionary algorithms designs has been to make use of multi-objective optimisation techniques, where many of the most often adopted designs are based on the *NSGAII* algorithm proposed by Deb et al. (2002). Both Jeong and Ahn (2015) and Olseng (2016) have used this algorithm in music generation tasks. It did also serve as the core design of the evolutionary approach in this project.

The concept of including lyrics in melody generation had been explored by some earlier research, but no similar research combining it with an evolutionary approach, as well as including the sentiment was discovered by the author in the literature review

phase of the project, and therefore the exploration of possibly usable techniques for lyric-based melody generation includes various solutions and problems within both sentiment analysis of lyrics and generating melodies fitting of input lyrics.

The most profitable concept discovered by the author, with regards to implementing lyric-based fitness functions for an evolutionary approach, was to use an approach close to the one used by Monteith et al. (2012), which focused on determining stress values of syllables in words. Their results were described as scoring highly in pleasantness for melodies, as well as scoring nearly as well as the actual melody of the input lyrics with the generated melodies. The solution implemented by Monteith et al. (2012) was though, more strict regarding the text to rhythm translation, and based on the results from the system implementation of lyrical fitness, described in subsection 6.1.2, it is clear that the rhythm of the melody could have been evaluated more strictly with regards to the lyrics.

Multiple approaches to sentiment analysis of song lyrics were reviewed and considered as possible solutions that could be adopted for ensuring that the generated melodies were fitting of the lyrical sentiment. Even though the data set used for testing included scores for an arousal metric for sentiment (Malheiro et al., 2018), the observations made by Hu et al. (2009) state that lyrics are not necessarily expressing enough in relation to arousal for it to be a valuable metric. As such there was a higher motivation for testing the tool Vader, described in subsection 6.1.5, which only considers the valence sentiment scores of input text.

For the translation of a given sentiment value to the melody, key concepts would be to determine the musical components most influenced by sentiment. A review of musical background theory, as well as subjective testing found that key changes between major and minor had the biggest impact, but that concepts such as the relationship between dissonance and consonance in the melody, as well as the contour, i.e. number of rising v.s. falling intervals are also valuable in determining the sentiment of a melody. The fact that the implemented sentiment analysis methods and the translation of the sentiment scores into the melody provided neutral results, described in subsection 6.1.5 and subsection 6.1.4, could indicate that other measures, both with regard to the sentiment analysis tool and what melodic aspects influence the emotional expression of a melody, could have been utilised.

The evaluations provided in subsection 6.1.1, subsection 6.1.2 and subsection 6.1.3 all present different tests, concepts and solutions for the implemented multi-objective evolutionary algorithm approach in this project. The evaluations all provide valuable insights with regards to research question 1.

To conclude on one distinct answer to research question 1 is hard, and likely not desirable, as the possibilities are seemingly endless within the topic, and as such this discussion can only serve as a set of proposals and insights with regards to lyric-based evolutionary music generation.

6.2.2. Research question 2 (RQ2)

How good will the quality of the melodies generated by the system prove to be, when judged in a human context?

It was seen as a necessity for thoroughly evaluating the system quality, and as such give a satisfactory answer to research question 2, to design a questionnaire where participants could subjectively evaluate the quality of the generated melodies, based on metrics described in section 5.3. A thorough evaluation of the questionnaire results can be found in subsection 6.1.4, and as such this section will serve as a concluding factor for the different aspects uncovered by the evaluation by bringing the answers within the different categories together.

The general melodic evaluation done by the human judges show that the system is able to generate melodies that are both pleasant and interesting, but that there are no definitive high quality musical aspects for all generated melodies. The overall structure of the melodies is judged to have made sense to the degree of being slightly more favourable than neutral. The sentiment of the lyrics and their translation into melodies have similarly been judged to score slightly higher than neutral, and even being considered as to a certain degree leaving a fitting emotional impression for one of the melodies. As the mentioned values, i.e. *Neutral* and *To a certain degree*, are both average values, the results are definitely encouraging. The most clearly lowest ranking metric of the generated melodies has to do with the degree of how well they fit their given input lyrics. Even though the rankings could still be considered neutral, there is a clear indication that the biggest struggle of the system has been to properly align rhythms (note durations) to syllables with various stress values.

The arguably most interesting aspect from the quantitative results, gathered from the questionnaire, has to do with the fact that a majority of the respondents deem the system as a possibly valuable resource in musical composing. There is a clear tendency among the respondents that the system's ability to function as a standalone musical composition tool is lower than that of it being an assisting resource, but the average value of its standalone potential still revolves around *Neutral*.

As to give a brief conclusion for research question 2, the quality of the generated melodies has proven to revolve around *Neutral*, with the quality being high enough for the system to be a potentially useful resource in music composition.

6.2.3. Goal

To provide a novel solution for automatically generating melodies fitting the structure and sentiment of user defined input lyrics

The main goal of this project can be said to have been reached, as a novel solution has been proposed for generating melodies based on input lyrics. The system implemented, by taking the findings of the literature review from chapter 3 into account, has definitively been able to generate melodies with a certain quality to them, but with the average values for most musical metrics revolving around *Neutral*, there has been some disagreement

between respondents regarding the quality of the output.

Backed by the findings of research question 2 and the evaluation presented in subsection 6.1.4, there are clear indications that the goal of generating melodies fitting the structure of the lyrics is the aspect the system has struggled the most with. Results for other evaluation metrics suggest that there has not been a bad quality over generated melodies, and that the melodies in some cases have been fitting of the emotional sentiment of the lyrics. In general, the highest scoring metric for the melodies has been in regards to its pleasantness, which has the only average value that can be considered as being *To a certain degree*, rather than *Neutral*.

Even though the results show that the average answer values mostly revolve around neutral, the system can be regarded as promising, especially because of being deemed suitable as a resource for musical composition, but also because it has generated melodies on the favourable side of neutral. The wish of providing a tool that could collaborate in human creativity was specifically listed as an important motivation for the project in section 1.1, and the results show a clear indication that said goal has been reached.

7. Conclusion and Future Work

This chapter is intended to bring the work conducted as part of this master's thesis to a conclusion, as well as present the main contributions of the research in section 7.2. The main conclusion is presented in section 7.1. Possibilities of future work are presented in section 7.3.

7.1. Conclusion

The core element of this master's thesis has been to present the design and implementation of a system capable of automatically generating melodies based on given input lyrics, as well as research conducted on the system and computational music generation in general. A key aspect within the mapping of a melody to lyrics has been to analyse the inherent sentiment of the input lyrics, as well as providing fitness functions for improving the lyrical suitability for the melody based on stress values of words and their placement. Much of the design decisions were grounded in a literature review conducted within the field of computational music generation, with a main focus on evolutionary approaches.

The main developmental phase consisted of testing of various architectural solutions, and the results from these experiments were important factors in deciding upon the most profitable fitness functions, design of genetic operators and system flow. The experiments also included testing within sentiment analysis, as to find a suitable method in evaluating the emotional expression of input lyrics.

The most important part of the system evaluation has been to distribute a questionnaire concerned with evaluating the musical potential of the system. A total of five melodies with different input lyrics, and hence different sentiment values, time signatures and musical keys, was used a basis for the questionnaire, and from the opinions of 42 participants, valuable information was gathered about the system's abilities within automatic music generation.

The quantitative results from the questionnaire suggested that the system was able to generate melodies of average (neutral) quality. In general, the highest scoring metrics for the melodies were within pleasantness and the degree of how interesting the melodies were. The system was able to give proper emotional impressions to the melody based on the input lyrics to some extent, but struggled the most with aligning a well functioning rhythm to words and syllables in the lyrics. The most important finding from the questionnaire was that a majority of the participants saw the system as a potentially valuable assisting resource in musical composition tasks.

As the generated melodies have been deemed as having a slightly better than neutral quality, the system can definitely be regarded as having generated encouraging results, and as the system has been regarded as having potential as a collaboration tool for musical composition, the goal of thesis has been reached.

7.2. Contributions

From the research results, and the findings from the evaluation and discussion of the conducted work, the most important contributions for further research provided by this master's thesis are presented below.

A system for lyric-based melody generation The perhaps most important contribution of this master's thesis is the implementation of a multi-objective evolutionary algorithm based system capable of generating melodies based on input lyrics and capture the sentiment of said lyrics. The developed system has been deemed capable to be of assistance in human composition of music. Further research within evolutionary melody generation, especially with lyrical input, could profit from a study of the implemented system.

Review of approaches to evolutionary music generation The review of interesting approaches within evolutionary music generation can be deemed an important contribution, as the results could help with further research. This contribution includes the literature review presented in chapter 3, but most importantly the discussion of the findings in regard to *RQ1* (subsection 6.2.1), which provide an overview of the most important aspects of the considered and adopted approaches for this project.

Results from architectural experiments An important contribution of this thesis is the results gathered from the experiments conducted during the development phase of the project. These results include the research conducted within the design of musically grounded genotypes, genetic operators, fitness functions and how sentiment values can be used in variable fitness evaluation, as well as the proposed solution to the problem with a non-dominated sorting ranking scheme collapse. The results can all contribute as valuable background material when developing or researching similar systems.

Results from the questionnaire The quantitative evaluation of the system, through the questionnaire, can also be considered a significant contribution, as it provides insight into the quality of the output generated by the system, as well as providing insight into people's expectations of automatically generated music. The insights could serve as a valuable input for the design of a similar system, or further improvement on the system design presented in this master's thesis.

7.3. Future Work

The arguably most important aspect of further work, in regards to the results gathered from the system evaluation, would be to improve on how the melodies are evolving to fit the given input lyrics. Possible solutions to this include to adopt fitness functions stricter in their evaluation than the ones in the current system design or to separate the rhythm generation from the main music evolution loop entirely. A rhythm generation process separated from the main loop could e.g. follow the *n-gram* approach used by Monteith et al. (2012). Another possibility could be to implement a novel two-stage solution comprised of having a lyric-based rhythm generation evolutionary algorithm as the first generation stage, and then generating a melody based on the rhythm generated from stage 1 with an architecture based on the system implemented in this project, but with updates to fitness objectives and functions. A two-stage evolutionary approach for music generation has been explored earlier by e.g. Khalifa and Foster (2006), mentioned in subsection 3.1.1.

An interesting possibility for further work could also be to incorporate artificial neural network techniques for different aspects of the evolutionary algorithm, and possibly within sentiment analysis. As explored by Mitrano et al. (2017), where the fitness of a genetic algorithm was judged by a recurrent neural network, the idea of replacing fitness functions with a neural network approach could provide interesting results. This could be tested for several of the implemented fitness objectives, but a total redesign of the system would probably be more effective than attempting to replace the fitness functions of all objectives in the current design. Another interesting possibility for future work could be to replace Vader as the basis for sentiment analysis with a neural network approach. A model such as BERT, proposed by Devlin et al. (2019), which is a state-of-the-art model for natural language processing, could prove useful for the task of sentiment classification.

Another interesting aspect of further work could be to extend the system into also automatically generating the input lyrics for the system. Lyrics could e.g. be generated using an artificial neural network approach. Navarro et al. (2020) did for example combine their novel melody generating Markov model based system with a previously developed system, called Tra-La-Lyrics (Oliveira, 2015), for automatically generating lyrics, such as to get a complete song writing pipeline. They stated that their overall results were positive.

Bibliography

Ajith Abraham and Lakhmi Jain. Evolutionary multiobjective optimization. In Ajith Abraham, Lakhmi Jain, and Robert Goldberg, editors, *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, chapter 1, pages 1–6. Springer, London, 2005.

Margareta Ackerman and David Loker. Algorithmic songwriting with ALYSIA. In João Correia, Vic Ciesielski, and Antonios Liapis, editors, *Computational Intelligence in Music, Sound, Art and Design*, pages 1–16, Cham, 2017. Springer International Publishing.

Hangbo Bao, Shaohan Huang, Furu Wei, Lei Cui, Yu Wu, Chuanqi Tan, Songhao Piao, and Ming Zhou. Neural melody composition from lyrics. In Jie Tang, Min-Yen Kan, Dongyan Zhao, Sujian Li, and Hongying Zan, editors, *Natural Language Processing and Chinese Computing - 8th CCF International Conference, NLPCC 2019, Proceedings, Part I*, volume 11838 of *Lecture Notes in Computer Science*, pages 499–511. Springer, 2019.

Susan Bartlett, Grzegorz Kondrak, and Colin Cherry. On the syllabification of phonemes. In *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics*, pages 308–316, 2009.

Chip Bell. Algorithmic music composition using dynamic Markov chains and genetic algorithms. *Journal of Computing Sciences in Colleges*, 27(2):99–107, 2011.

John A. Biles. GenJam: A genetic algorithm for generating jazz solos. In *Proceedings of the 1994 International Computer Music Conference, ICMC 1994*, pages 131–137. Michigan Publishing, 1994.

John A Biles. Autonomous GenJam: eliminating the fitness bottleneck by eliminating fitness. In *Proceedings of the 2001 Genetic and Evolutionary Computation Conference Workshop Program, San Francisco*, 2001.

John A Biles. Straight-ahead jazz with GenJam: A quick demonstration. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2013.

Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Slowiński. *Multiobjective optimization: Interactive and evolutionary approaches*, volume 5252. Springer Science & Business Media, 2008.

Pablo Samuel Castro. Performing structured improvisations with pre-trained deep learning models. In *Proceedings of the 10th International Conference on Computational Creativity 2019*, pages 306–310, 2019.

Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *ArXiv*, abs/1412.3555, 2014.

Florian Colombo, Samuel P. Muscinelli, Alexander Seeholzer, Johanni Brea, and Wulfram Gerstner. Algorithmic composition of melodies with deep recurrent neural networks. *ArXiv*, abs/1606.07251, 2016.

Florian Colombo, Alexander Seeholzer, and Wulfram Gerstner. Deep artificial composer: A creative neural network model for automated melody generation. In João Correia, Vic Ciesielski, and Antonios Liapis, editors, *Computational Intelligence in Music, Sound, Art and Design*, pages 81–96, Cham, 2017. Springer International Publishing.

Simon Colton, Ramón López de Mántaras, and Oliviero Stock. Computational creativity: Coming of age. *AI Magazine*, 30:11–14, 2009.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

Dario Floreano and Claudio Mattiussi. *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. The MIT Press, 2008.

Paul A Gagniuc. *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.

Sarah Harmon. Narrative-inspired generation of ambient music. In Ashok K. Goel, Anna Jordanous, and Alison Pease, editors, *Proceedings of the Eighth International Conference on Computational Creativity, Atlanta, Georgia, USA, June 19-23, 2017*, pages 136–142. Association for Computational Creativity (ACC), 2017.

Lejaren A Hiller Jr and Leonard M Isaacson. Musical composition with a high speed digital computer. In *Audio Engineering Society Convention 9*. Audio Engineering Society, 1957.

Yajie Hu, Xiaoou Chen, and Deshun Yang. Lyric-based song emotion detection with affective lexicon and fuzzy clustering method. In Keiji Hirata, George Tzanetakis, and Kazuyoshi Yoshii, editors, *Proceedings of the 10th International Society for Music*

Information Retrieval Conference, ISMIR 2009, pages 123–128. International Society for Music Information Retrieval, 2009.

C. Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. 2014. URL `https://www.aaai.org/ocs/index.php/ICWSM/ICWSM14/paper/view/8109`.

Jae Hun Jeong and Chang Wook Ahn. Automatic evolutionary music composition based on multi-objective genetic algorithm. In *18th Asia Pacific Symposium on Intelligent and Evolutionary Systems*, volume 2, pages 105–115. Springer, Cham, 2015.

Jae Hun Jeong, Yusung Kim, and Chang Wook Ahn. A multi-objective evolutionary approach to automatic melody generation. *Expert Systems with Applications*, 90:50–61, 2017.

Thorsten Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Kluwer Academic Publishers, USA, 2002. ISBN 079237679X.

Colin G. Johnson. Fitness in evolutionary art and music: What has been used and what could be used? In *Proceedings of the First International Conference on Evolutionary and Biologically Inspired Music, Sound, Art and Design*, EvoMUSART'12, page 129–140, Berlin, Heidelberg, 2012. Springer-Verlag.

Yaser Khalifa and Robert Foster. A two-stage autonomous evolutionary music composer. In Franz Rothlauf, Jürgen Branke, Stefano Cagnoni, Ernesto Costa, Carlos Cotta, Rolf Drechsler, Evelyne Lutton, Penousal Machado, Jason H. Moore, Juan Romero, George D. Smith, Giovanni Squillero, and Hideyuki Takagi, editors, *Applications of Evolutionary Computing*, pages 717–721, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

Anssi Klapuri. Introduction to music transcription. In *Signal processing methods for music transcription*, pages 3–20. Springer, 2006.

Guido Kramann. Darwinian pianos: realtime composition based on competitive evolutionary process. In *International Conference on Evolutionary and Biologically Inspired Music and Art*, pages 37–46. Springer, 2013.

Feynman T Liang, Mark Gotham, Matthew Johnson, and Jamie Shotton. Automatic stylistic composition of Bach chorales with deep LSTM. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017*, pages 449–456. ISMIR, 2017.

Ricardo Malheiro, Renato Panda, Paulo Gomes, and Rui Pedro Paiva. Emotionally-relevant features for classification and regression of music lyrics. *IEEE Transactions on Affective Computing*, 9(2):240–254, 2018.

Bill Manaris, Dallas Vaughan, Christopher Wagner, Juan Romero, and Robert B Davis. Evolutionary music and the Zipf-Mandelbrot law: Developing fitness functions for pleasant music. In *Workshops on Applications of Evolutionary Computation*, pages 522–534. Springer, 2003.

Clive Matthews. *An introduction to natural language processing through Prolog*. Routledge, 2016.

Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 5(4):1093 – 1113, 2014.

Peter Mitrano, Arthur Lockman, James Honicker, and Scott Barton. Using recurrent neural networks to judge fitness in musical genetic algorithms. In *Proceedings of the 5th International Workshop on Musical Metacreation (MUME) at the 8th International Conference on Computational Creativity (ICCC)*, 06 2017.

Kristine Monteith, Tony R Martinez, and Dan Ventura. Automatic generation of melodic accompaniments for lyrics. In *Proceedings of the Third International Conference on Computational Creativity*, pages 87–94, 2012.

Maria Navarro, Hugo Gonçalo Oliveira, Pedro Martins, and Amílcar Cardoso. Integration of a music generator and a song lyrics generator to create Spanish popular songs. *Journal of Ambient Intelligence and Humanized Computing*, 03 2020.

Gerhard Nierhaus. *Algorithmic composition: paradigms of automated music generation*. Springer Science & Business Media, 2009.

Hugo Gonçalo Oliveira. Tra-la-lyrics 2.0: Automatic generation of song lyrics on a semantic domain. *Journal of Artificial General Intelligence*, 6(1):87–110, 2015.

Olav Olseng. An application of evolutionary algorithms to music: Co-evolving melodies and harmonization. Master's thesis, Norwegian University of Science and Technology, June 2016.

Olav Olseng and Björn Gambäck. Co-evolving melodies and harmonization in evolutionary music composition. In Antonios Liapis, Juan Jesús Romero Cardalda, and Anikó Ekárt, editors, *Computational Intelligence in Music, Sound, Art and Design*, pages 239–255, Cham, 2018. Springer International Publishing.

Gary M Rader. A method for composing simple traditional music by computer. *Communications of the ACM*, 17(11):631–638, 1974.

Forrest Tobey. *A Feeling for Harmony - Book 1: The Joy of Harmony and Composition*. Earlham College, 2012. URL: `http://legacy.earlham.edu/~tobeyfo/musictheory/Book1/home1.html` Retrieved: 2020-05-12.

Peter M Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 13(4):27–43, 1989.

Michael W Towsey, Andrew R Brown, Susan K Wright, and Joachim Diederich. Towards melodic extension using genetic algorithms. *Educational Technology & Society*, 4(2): 54–65, 2001.

Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

Chia-Lin Wu, Chien-Hung Liu, and Chuan-Kang Ting. A novel genetic algorithm considering measures and phrases for generating melody. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2101–2107. IEEE, 2014.

Yunqing Xia, Linlin Wang, Kam-Fai Wong, and Mingxing Xu. Sentiment vector space model for lyric-based song sentiment classification. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, HLT-Short '08, page 133–136, USA, 2008. Association for Computational Linguistics.

Xinjie Yu and Mitsuo Gen. *Introduction to evolutionary algorithms*. Springer Science & Business Media, 2010.

A. Questionnaire - Norwegian

This appendix presents the untranslated versions of the scoring scale and the questions used in the questionnaire.

A.1. Untranslated scoring scale

The participants of the questionnaire rated most of the questions using the scoring scale presented in Table A.1.

Table A.1.: Score rating scale in Norwegian

Score:	**Correlated wording**
-3	I svært liten grad
-2	I liten grad
-1	I ganske liten grad
0	Nøytral
1	Til en viss grad
2	I større grad
3	I svært stor grad

A.2. Untranslated questions

The Norwegian, untranslated questions and the included question descriptions used in the questionnaire is presented in the description below, which is separated following the same pattern as used in section 5.3.

- Musikkerfaring
    1. Hører du ofte på musikk?
    2. Lager du egen musikk?
    3. Kan du å lese noter?
- Melodier
    1. Har du hørt den originale sangen før? (ja - nei - vet ikke)
    2. Melodi
        a) Er melodien musikalsk?

   b) Er melodien interresant?
   c) Henger musikken sammen (gir oppbyggingen mening)?
3. Tekst og melodi
   a) Gir melodien et følelsesmessig riktig intrykk med tanke på teksten?
   b) Passer ordene og stavelsene til sine gitte noter?
4. Ekstra kommentarer

- Helhetsinntrykk - Anser du et slikt system som potensielt nyttig i forbindelse med sangkomponering?
  1. Kunne det fungert godt til å komponere på egenhånd?
  2. Kunne det fungert godt som støtte innen komponering?

Beskrivelse for spørsmål om melodi

- 2. a) Er melodien musikalsk? - (Synes du den følger musikalske regler/normer? Synes du den er behagelig å høre på?)
- 2. b) Er melodien interessant? - (Synes du melodien har interessante trekk, eller er den kjedelig/innholdsløs?)
- 2. c) Henger musikken sammen? - (Helhetlig inntrykk av melodien. Har melodien en naturlig progresjon, eller har melodien for eks. brå/unaturlige forandringer?)

Beskrivelse for spørsmål om tekst og melodi

- 3. a) Gir melodien et følelsesmessig riktig intrykk med tanke på teksten? - (Er teksten trist eller glad? Bærer melodien preg av dette?)
- 3. b) Passer ordene og stavelsene til sine gitte noter? - (Er det unaturlige takt- eller noteforskjeller mellom ord eller stavelser? Kan hoppes over, dersom man har vansker med å følge med på tekst/noter)

B. Generated melodies

The source of each melody's lyrics is added as footnotes to the section headings. For all melodies a figure containing the corresponding sheet music is presented. Following the sheet music is a table containing questionnaire results regarding the melodic qualities of the given melody. For all questionnaire results, *AVG* is the average of all answer values for a given question and *SD* is the standard deviation of the answer population of each given question. The questions are organised by their question annotations from subsection 5.3.2 and section A.2. For each melody, a few of the voluntary comments have been selected from each melody question. All comments have been translated from Norwegian to English by the author, and some have been cropped because of a large size. Note that the translations might not be subjectively accurate. The percentage of participants having previous knowledge of the song the different lyrics have been taken from is also listed for each melody.

The melody presentation starts on the next page.

B.1. Melody 1 - Nellie Dean[1]

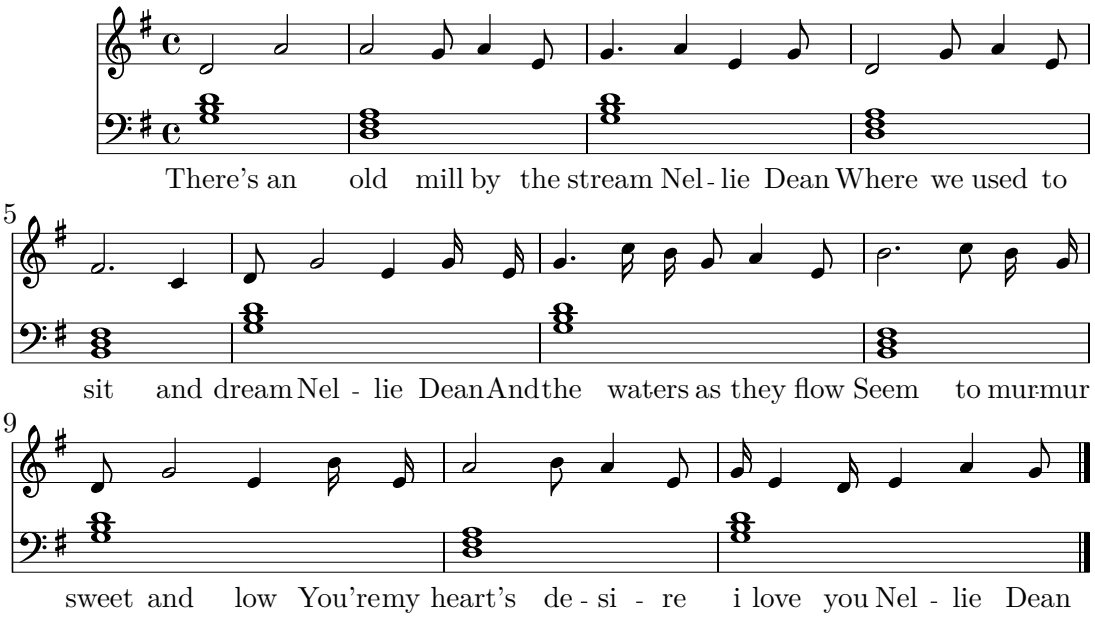

Figure B.1.: Generated melody for Nellie Dean

Table B.1.: Questionnaire results - melody 1

	2. a)	**2. b)**	**2. c)**	**3. a)**	**3. b)**	**Total**
AVG	0.9	0.2	0.2	0.4	-0.5	0.2
SD	1.1	1.3	1.3	1.3	1.3	1.3

Previous knowledge of song: **Yes:** 2.33%, **No:** 81%, **Not sure:** 16.66%

Selected comments from questionnaire

- The melody works OK, and sounds good. Works well on sentiment. The places where the lyrics are dense, the notes in the melody becomes unnatural. (...)
- (...) The long notes can be in the middle of a sentence, and then a new sentence might jump to next too abruptly, without pause or that the note is held.
- The lyrics follows the notes badly, such that a sentence is concluded and another begins between two notes with a short duration. It seems somewhat untidy, and it happens on multiple occasions throughout the song.

[1] `https://en.wikisource.org/wiki/Nellie_Dean`

B.2. Melody 2 - Bridget O'Malley[2]

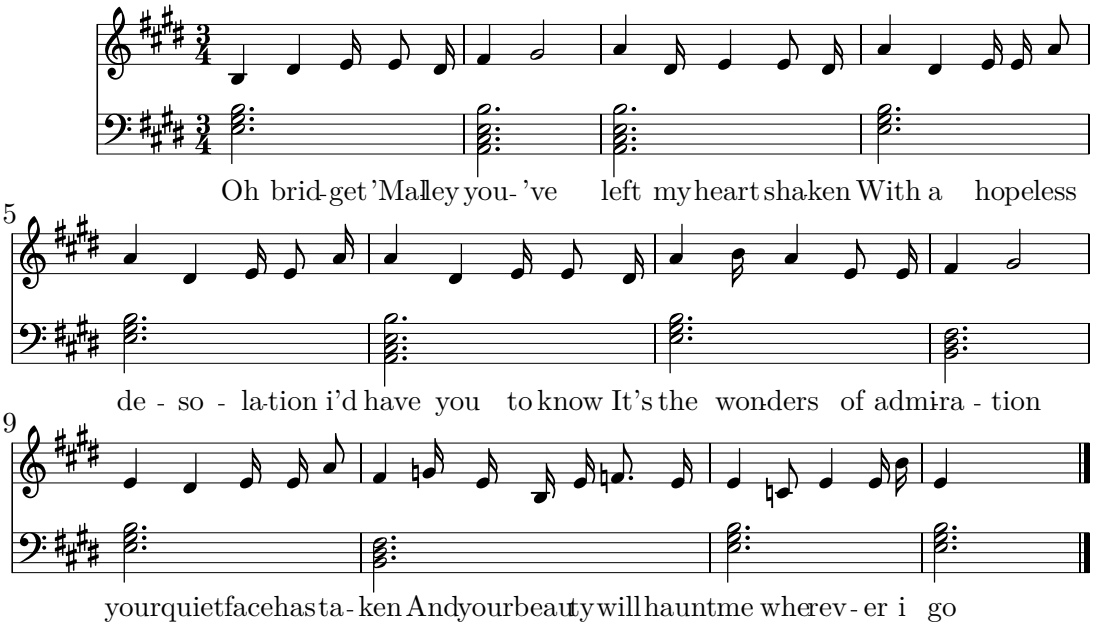

Figure B.2.: Generated melody for Bridget O'Malley

Table B.2.: Questionnaire results - melody 2

	2. a)	**2. b)**	**2. c)**	**3. a)**	**3. b)**	**Total**
AVG	-0.4	-0.1	-0.7	-0.3	-0.6	-0.4
SD	1.4	1.6	1.4	1.4	1.3	1.4

Previous knowledge of song: **Yes:** 9.5%, **No:** 78.6%, **Not sure:** 11.9%

Selected comments from questionnaire

- (...) I can't quite sing the lyrics to the melody
- This one is less tidy, especially in the ending. Many misses with regards to syllables. The sentiment works relatively well (...).
- The notes fit with the syllables on nearly everything (...) I regard it as almost catchy until the last three measures. There everything gets strange (...)

[2] `https://en.wikisource.org/wiki/Bridget_O%27Malley`

B.3. Melody 3 - Early one morning[3]

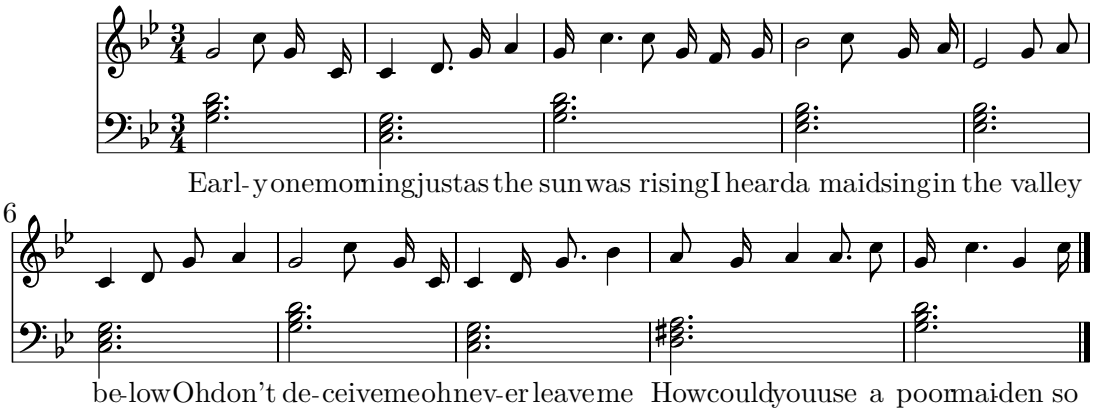

Figure B.3.: Generated melody for Early one morning

Table B.3.: Questionnaire results - melody 3

	2. a)	**2. b)**	**2. c)**	**3. a)**	**3. b)**	**Total**
AVG	0.5	0.3	0.4	0.8	-0.4	0.3
SD	1.4	1.3	1.4	1.4	1.5	1.4

Previous knowledge of song: **Yes:** 7.5%, **No:** 80%, **Not sure:** 12.5%

Selected comments from questionnaire

- Naturally weighted syllables in the lyrics occurs less frequently on weighted beats in the measures in this song than the previous. (...)

- The biggest problems are that less important words in the middle of phrases gets long note durations, and the last word does not get it. (...) The melody is generally hard to follow, although the note order makes some sense. Would have to practice hard to be able to sing this myself.

[3] https://en.wikisource.org/wiki/Early_One_Morning_(Anonymous)

B.4. Melody 4 - Henry Martyn[4]

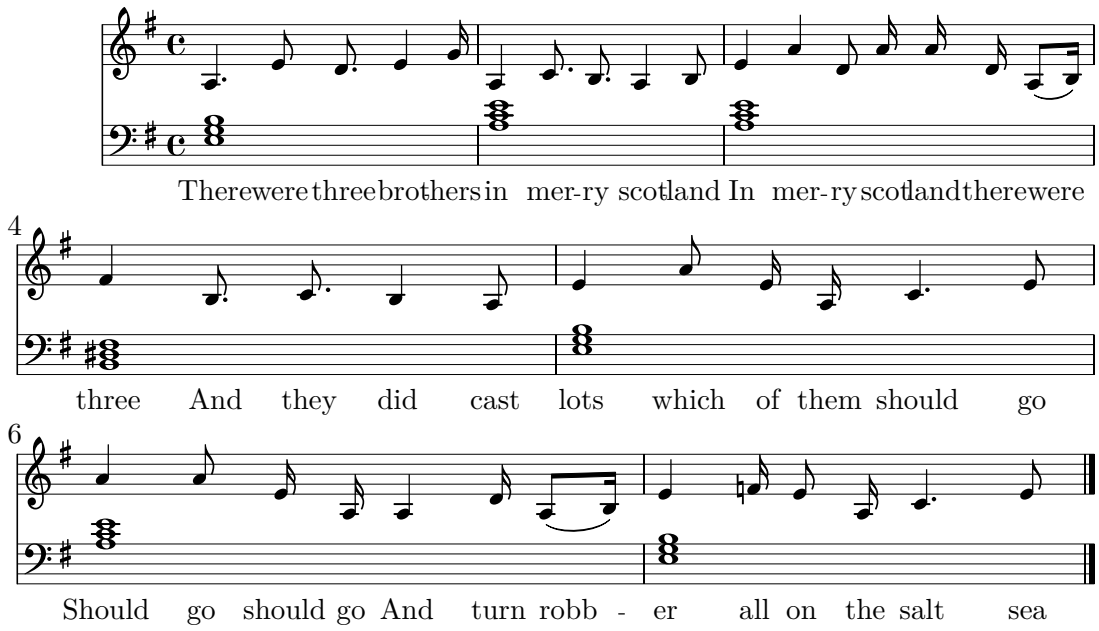

Figure B.4.: Generated melody for Henry Martyn

Table B.4.: Questionnaire results - melody 4

	2. a)	**2. b)**	**2. c)**	**3. a)**	**3. b)**	**Total**
AVG	1.0	0.6	0.3	0.1	-0.3	0.3
SD	1.1	1.2	1.3	1.5	1.4	1.3

Previous knowledge of song: **Yes:** 5%, **No:** 82.5%, **Not sure:** 12.5%

Selected comments from questionnaire

- This one is quite good on syllables, but does not fit with regards to sentiment (...)
- (...) There is a somewhat better relation between lyrics and syllables (...) From a human I would expect that the repetition in the lyrics was highlighted to a larger extent in the rhythm and melody. All in all one of the more singable melodies.

[4] https://en.wikisource.org/wiki/Child%27s_Ballads/250

B.5. Melody 5 - Billy Lyons and Stack O'Lee[5]

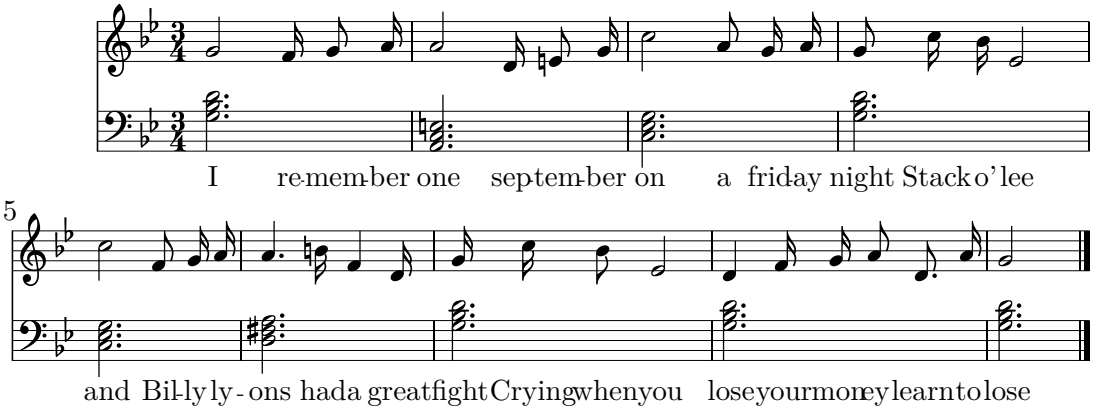

Figure B.5.: Generated melody for Billy Lyons and Stack O'Lee

Table B.5.: Questionnaire results - melody 5

	2. a)	**2. b)**	**2. c)**	**3. a)**	**3. b)**	**Total**
AVG	0.8	0.6	0.4	0.2	0.1	0.4
SD	1.4	1.4	1.3	1.5	1.6	1.4

Previous knowledge of song: **Yes:** 7.7%, **No:** 82%, **Not sure:** 10.3%

Selected comments from questionnaire

- I can almost sing along. Only a couple of places could I have needed a breather.
- Like the other songs, a somewhat unnatural rhythm. A bit complicated syncopation (...)
- Mostly good, but some notes feel a bit "off", mostly because of cancelling the flats. (...) Seems like it has been hard to understand which words were related, when the melody was generated (...)

[5] https://en.wikisource.org/wiki/Billy_Lyons_and_Stack_O%27_Lee

Sondre Søberg

Lyric-based melody generation

NTNU
Norwegian University of
Science and Technology