Evolving Cellular Automata for Music Composition with Trainable Fitness Functions

Thesis · March 2012

DOI: 10.13140/RG.2.2.26718.36166

1 author:

Man Yat Lo
University of Essex
3 PUBLICATIONS **30** CITATIONS

SEE PROFILE

Evolving Cellular Automata for Music Composition with Trainable Fitness Functions

Man Yat Lo

A thesis submitted for the degree of Doctor of Philosophy

School of Computer Science and Electronic Engineering

University of Essex

March 2012

Abstract

This thesis focused on the application of evolutionary computational techniques for music composition. Conventionally, the music evaluator in an evolutionary music composition system is either a human operating the system interactively, or a knowledge-based system. The objective of this study was to investigate a novel approach to music composition that combines a machine-learning based evaluator with a music generator. The evaluator is based on a machine-learning technique called N-gram modelling while Cellular Automata (CA) are used as the music generators. Hence Evolutionary Algorithms (EAs) are used to evolve CA capable of generating the style of music that the evaluators have been trained to rate highly.

For the investigation of the N-gram model, the experimental results showed that the discriminative power of the N-gram models were able to correctly classify composers with up to 80% accuracy in a composer classification task. An initial set of experiments used N-gram fitness functions to directly evolve musical sequences. The results showed that in order to evolve interesting music, appropriate musically meaningful genetic operators and constraints must be applied since optimal sequences (rated by the N-gram) tend to be extremely repetitive.

However, some CAs show a natural ability for generating interesting music. The proposed CA-based evolutionary music composition system is able to evolve structured music without pre-defining a musical structure or a separate evolutionary process. Furthermore various types of fitness functions

were proposed that aim to cooperate with N-gram fitness functions and evolve polyphonic music using a multi-objective evolutionary algorithm. Finally in order to evaluate the success of the proposed system and feedback, two online music surveys were conducted. The results showed that although on average the human-composed music was preferred to the evolved music, there is one piece of evolved music was close to indistinguishable from human-composed music.

Acknowledgements

There are a lot of people I would like to thank for making this thesis possible. At the top of the list are my supervisor Professor Simon Lucas and the board members Professor Riccardo Poli and Dr. Amnon Eden. This research project would not have been completed without their guidance and supervision. A lot of my ideas and inspirations came from suggestions and constructive criticism from my supervisor in our regular meeting sessions. Also I would like to thank my family and friends for their untiring support and encouragement.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

This thesis describes empirical research that provides a basis for applying Artificial Intelligence (AI) techniques to the problem domain of automatic music composition. The proposed methodology mainly employs Evolutionary Algorithms (EAs) to perform the optimisation together with statistical natural language processing techniques for the music evaluators. A dynamic model, Cellular Automata (CA), is used for music generation. The EA evolves the parameters of this model. Hence, this research is within the realm of evolutionary music, which can be described as a computer program that operates with a population of machine generated music (usually random) or existing human-composed music, using some kind of evolutionary algorithm to search for the best musical piece that satisfies the music evaluator (either algorithmic or human).

This chapter is organised as follows: section 1.1 provides a brief introduction to the field of computer music. Section 1.2 discusses some issues in the aspect of music creativity. Section 1.3 describes the details of this research

including the motivations, overview, challenges and issues for the current state-of-the-art of evolutionary music, and list of publications. Finally, section 1.4 outlines this thesis.

1.1 Introduction to Computer Music

After more than fifty years of digital computer development, computers have been applied to a vast and ever-increasing range of applications. One such application is music composition. For many years software packages have been developed as efficient tools to aid the human musical composition process. In particular, musicians use music software such as music mixing, digital recording, educational tools and music generators for the benefit of their music creations. An example of similar software that is available commercially is MySong [186] [141] from Microsoft. The aim of the MySong project is to build a system that allows non-musically trained users to create songs by singing into a microphone. The software generates a chord sequence to accompany the vocal melody to produce a full song.

Computer music research is a wide field that comprises several areas with distinct objectives. In general, it can be categorised into four main areas: composition, performance, music theory and digital sound processing [175]. The objectives of each area are summarised as follows:

Composition Build a computer system that is able to generate music or aid a human to compose music. Such a system saves the time of composition and can provide for musical inspiration.

Performance Build a computer system that benefits musical performers throughout their studio recording or stage performance.

Music theory A computer program is used to analyse musical pieces with respect to musical theories. Usually, the program extracts musical elements from a piece, then performs a statistical analysis and finds some general patterns.

Digital sound processing The main objectives of this area are to develop and improve sound processing techniques for accessing and manipulating acoustic sound information.

1.2 Music and Creativity

Questions such as "What is music?", "Where does music come from?", "What is creativity in music?", "How do we evaluate music (including music recognition)?" and "How do we compose music efficiently?" are open-ended questions that multiple disciplines are attempting to answer. Musicologists study the history of music, music notation, music style and physical parameters of playing music e.g. timing, loudness, tempo and articulation [157]. Ethnomusicologists study why and how people make music in a cultural context [43]. Music is often thought of from an anthropological perspective. Music cognition is a study of music perception [57] [158] and the mechanism of music creativity from the cognitive science perspective. Bio-musicology is a study of music from the biological perspective: music is studied in reaction to its meaning in animal communication and evolution [42]. Computer

scientists study the mechanism of musical creativity using simulations [19] [41]. Unfortunately some of the above questions and proposed answers are still being debated amongst researchers, such as Miller who suggests that music is for the evolutionary process of sexual selection [133], while biologist Pinker suggested that music is only a by-product of human evolution and it is unnecessary for humans [164].

For general creativity including music, art, and human thought (idea), the full explanation of such human creativity remains elusive and may never be solved. Some researchers have proposed various definitions of the word 'Creativity': Candy and Edmonds defined creativity as "A Set of activities that give rise to an outcome product that is recognised to be innovative as judged by an external standard" [26]; Boden defined creativity into two categories: 'Personal Creativity' (P-Creativity), a new idea that has never been created by the creator before; and 'Historical Creativity' (H-Creativity), a new idea that has never been created by any creator in public or throughout history [19]. Candy and Edmonds explained that creativity involved external judgment only, while Boden favours both internal and external judgments. However, there is no universal definition of what creativity is yet. In this instance, the author is inclined toward Boden's definition of creativity. We believe that external judgment or evaluation provides feedback and correction on the process of creativity. More detail of the explanation of music creativity in the aspect of cognitive theory is outlined by Pearce and Wiggins in [161].

1.3 Research

1.3.1 Research Motivation

A major motivation for building an automatic music composition system is to reduce the time and cost for rapid music production. Besides its commercial application, an automatic music composition system can be used as a tool in musical education, for example, to instantly generate a particular style of music that can be used for the purpose of demonstration or to act as an aid to help or inspire music students to compose music. Here are two examples of automatic music composition systems that produce commercial music albums: EMI[1] system, which is a knowledge-based system created by Cope [37] [38]; and GenJam[2] , which is an interactive evolutionary system created by Biles [18]. Both systems are being used in music production and some generated music has been recorded in a CD album which is selling worldwide. GenJam is even continuously catching public attention via live performance.

1.3.2 Research Objective, Scope and Approach

The proposed research applies artificial intelligence techniques in music composition. The research objective is to develop a system that is capable of generating music automatically using an evolutionary algorithm. During the evolution, the system has no human interaction (human-in-the-loop approach), the music evaluator is an algorithm that can be trained on existing

[1]http://arts.ucsc.edu/faculty/cope/experiments.htm
[2]http://www.it.rit.edu/ jab/GenJam.html

musical pieces, which in this case are represented in the MIDI format.

This thesis described two approaches for developing an evolutionary music system. First approach (called direct approach) is to use N-gram model to act as the music evaluator in an evolutionary algorithm to evolve musical sequences at the genotype level. Musical sequences are manipulated directly by genetic operators. Musical sequences are not necessarily directly evolved by an EA system. The second approach (called indirect approach) is to evolve a particular generative model such Cellular Automata (CA) which is then used to generate music. The genotype encodes the parameters of the cellular automata models and phenotype is the musical sequence that is rendered by a music mapping algorithm from the space-time pattern, which is generated by the cellular automata model.

Finally, two online music surveys are conducted in order to obtain some idea of how the evolved music sounds to human listeners. The surveys are based on asking human subjects to rate our music samples (composed by human and different types of machines) and to answer whether they believe the music to be composed by a machine or a human. The aim of the surveys is to see whether the music satisfaction of the evolved music is on a par with the human-composed music, and how likely the listeners thought the evolved music to be composed by a human.

1.3.3 Hypotheses

We formulate the central questions of the thesis as five testable hypotheses:

- 1) "The direct approach to music generation with a trained N-gram

fitness function will produce pleasing music."

- 2) "The direct approach to music generation with a trained N-gram fitness function using a "Bag of notes" constraint will produce pleasing music."

- 3) "The indirect approach to music generation using a CA and trained N-gram fitness functions will produce pleasing music."

- 4) "Adding polyphony to evolved CA music results in more pleasing music being generated."

- 5) "The indirect approach to music generation using a CA and trained N-gram fitness functions with added polyphony will produce music which the listeners will mistake for human-composed music."

In above hypotheses, the criterion of "pleasing music" is subjective. In order to objectively define the criterion of "pleasing music", a set of music listening tests was designed and conducted. The aim was to let human listeners determine which pieces of music they considered to be more pleasant than others. In the tests, human listeners listened to a set of music samples that consisted of human-composed, randomly generated and system-evolved (indirectly evolved) music. Each music sample was rated by the human listeners in terms of how pleasant the given music was, in the range of 1 to 5 (1 being the worst and 5 being the best). They also had to answer whether they believed the music to be composed by a machine or a human (two-way choice; the listener specifies human or machine). In this way, each music sample was given mean values to indicate how much the listeners appreciated the

music sample, and how likely they thought it to be composed by a human. It is rational to assume that the randomly generated music will receive the lowest mean rating among other groups in the music tests; therefore in order to judge whether the human listeners were pleased by the system-evolved music, system evolved music must receive a higher mean rating than the random-generated music.

1.3.4 Challenges and Issues for Evolutionary Music

The following outline some major challenges and issues that need to be addressed in evolutionary music systems:

Fitness function Defining a general fitness function for music evaluation is challenging as the outputted music can be subjectively evaluated by an individual listener, unlike other engineering problems which have clear predefined criteria. Therefore, evaluating music objectively is difficult or even impossible. In most existing evolutionary music composition systems, there are three approaches that are commonly used for defining the fitness function:

- Interactive - where a human acts as the fitness function in an EA system. The human is responsible for listening to all or part of the evolved music and then rating their fitness. It is well-known that an interactive approach has led to interesting results but is time consuming [17].
- Knowledge-based - where fitness functions are devised from musical theories or the system designer's knowledge. This approach

requires prior knowledge and the musical rules are subjectively selected by the system designer. Due to the fact that musical rules are explicitly extracted from the expert knowledge, the evolved music is expected follow a certain music style or have certain constraints.

- Machine-learning based - where some kind of machine learning techniques are used to learn the regularities of patterns in the training samples, then the learned model acts as a fitness function to evaluate the music. It could be argued that this approach is close to the process of how humans learn music and create musical ideas by extracting features from samples and then altering the existing musical ideas. The problem for this approach is to fine tune the learning model so it is general enough to produce a variety of music and specific enough to represent the training samples.

Exploration Evolutionary algorithms are used for exploration rather than optimisation in the musical search space. Although, evolutionary music composition systems aim to search for high fitness music with variation, it is not necessary to pick the optimal only, as diversity is important for music creation. However, to define the weights between exploration and optimisation is difficult and the decision might involve trial-and-error.

Representation There is not much research that explores and evaluates music representations specifically for evolutionary music composition systems. A few attempts to investigate the representation for explorative evolutionary computation were made by Bentley [11], who stud-

ied component-based representation and Machwe et al. [119] proposed an object-based representation in agent technology. In the literature, most of the evolutionary music composition systems are implemented with two representations to encode musical melody: absolute pitch and pitch difference (interval between successive pitches). These representations are usually used in Music Information Retrieval (MIR) systems. However, there are no empirical results showing that these musical representations are suitable for evolutionary music composition systems.

Music Extraction A full piece of music contains multi dimensional information such as pitch, volume, duration etc. Effectively extracting the required information from a large music database is a challenge because each piece of music contains various details of information.

Musical structure From traditional to modern western music, a musical piece is described in an hierarchical structure. A full piece of music contains parts (instruments), each part contains phrases (groups of notes), each phrase contains notes (individual with multiple descriptions such as duration, pitch, velocity etc...). The arrangement and parameters for these elements in the hierarchy structure are non-deterministic. The structure of the music is often linked to the composer's preferences or a particular music style. Generating structured music is a major challenge for algorithmic music composition systems. This challenge not only requires the system to generate groups of notes (phrases), it also requires the system to arrange the generated phrases in a musically meaningful way. For instance in the AABA form, which is a well-

known arrangement form used to arrange musical phrases in a melody, the melody is played in order as it plays phrase A twice, then phrase B and then goes back to phrase A. Phrases A and B in this form are often referred to as verse and bridge. In the literature, the issue of generating structural music has been addressed by the following methods: explicitly pre-define the musical structure in advance [98] [148]; musical structure embedded into fitness functions; the music representation in the system has a set of rules/grammar to define the musical structure [49] [173]; use some kind of generative model to generating structural music [168] [125].

.

Research Challenge and Questions

In order to achieve the goals of this thesis, several important questions were raised:

- How should the music be represented as inputs and outputs to the system (polyphony, monophony, representation for note and duration, etc) ?
- What form of model should the music evaluation algorithm be based on?
- What representation should the evolutionary algorithm operate in? And what is the genotype and phenotype mapping?
- What evolutionary variation operators should be used?

- What should the design of the system architectures be in both direct and indirect approaches for the evolutionary music composition systems? And what are the strengths and weaknesses of these approaches?
- How should the success of the approaches be evaluated?

Subsequently, several objective measurements have been used to evaluate the success of this research. These measurements are summarised as follows:

- Evaluate how the discriminative power of trainable music evaluators varies with the major setup parameters. The results are evaluated based on the number of correct classifications.
- Demonstrate how the configurations of the EA affect the abilities of the system to converge to music which has a high fitness evaluated by the fitness function.
- Demonstrate the importance of music representations in affecting the trainable music evaluator.
- Evaluate the proposed music evaluators by providing evidence that shows some existing well-known music shares the same/similar features with the music evaluators.
- Demonstrate the proposed generative model is suitable for evolutionary music.
- Demonstrate how the configurations of music evaluators affect the evolution of a generative model to generate music.

- Evaluate the proposed approaches by conducting music surveys that analyses the musical satisfaction of the evolved music to human listeners.

1.3.5 List of Publications

A number of conference papers have been published relating to this thesis. Most of the experiments discussed in chapter 4 have been published in two papers in the proceedings of IEEE Congress on Evolutionary Computation. These are:

- Lo, M.Y. and Lucas, S.M., Evolving Musical Sequences with N-Gram Based Trainable Fitness Functions, IEEE Congress on Evolutionary Computation, 2006 [113].

- Lo, M.Y. and Lucas, S.M., N-gram Fitness Function with a Constraint in a Musical Evolutionary System. IEEE Congress on Evolutionary Computation, 2007 [114].

1.4 Thesis Outline

The remaining chapters of the thesis are organised as follows: chapter 2 provides all the background information necessary for the reader to understand the underlying technologies used. Chapter 3 describes the research methodologies and the system implementation in detail including the framework, architecture of the proposed evolutionary music composition systems, configurations and parameters of the system components. It then describes the

details of four music evaluators (fitness functions) that are designed based on the N-gram model, Zipf's Law and Information Entropy. Chapter 4 (experiments) focuses on the trainable music evaluator which is implemented with the N-gram model. Firstly, it describes the performance of the N-gram model in terms of its discriminative power for performing a composer classification task. Secondly, it discusses the impact of different sequence-oriented genetic operators on the music evaluator in the evolutionary algorithm. The experiments test the ability of an EA to optimise sequences with respect to the N-gram fitness functions. Finally, it describes a proposed method 'Bag of notes' constraint to solve a prior problem where the Maximum Likelihood Sequence (MLS) generated by the N-gram model is extremely repetitive. Chapter 5 (experiments) comprises a number of experiments to investigate the capability of the proposed indirect approach for evolutionary music composition systems. The experiments also provide an insight into whether the proposed system has an advantage in generating musical sequences with a high level of musical structure. Various configurations of the fitness functions were tested in the experiments. Chapter 6 (experiments) consists of two parts, the first part describes the details of two online music surveys and the results. The surveys are based on asking human subjects to rate our music samples (composed by human and different types of machines) and to answer whether they believe the music to be composed by a machine or a human. The second part describes an experiment that aims to analyse the difficulty of evolving cellular automata using a genetic algorithm method. Finally, chapter 7 provides a summary of this research and discusses improvements and directions for future study. Note that the attached DVD-

rom contains some musical materials taken from the experiments and music surveys described in this thesis.

Chapter 2

Background and Literature Reviews

The aim of this chapter is to introduce the field of evolutionary music and provide all the necessary background information for understanding the concepts described in this thesis and review some existing techniques for automatic music composition. The chapter is organised as follows: section 2.1 briefly introduces evolutionary computation. Section 2.2 introduces algorithmic music composition and its issues. Section 2.3 introduces evolutionary music with a discussion on the relationship between the evolutionary algorithm and the process humans use to compose music. Section 2.4 introduces a framework for music composition systems. Section 2.5 reviews some research projects and discusses their techniques. Section 2.6 describes evolutionary music composition systems from the perspective of genotype-phenotype mapping. Section 2.7 introduces the cellular automata model and reviews some CA based music composition systems and their techniques. Finally sections 2.8, 2.9 and 2.10

are the introduction of N-gram model, Zipf's Law and information entropy.

2.1 Evolutionary Computation

Evolutionary Computation (EC) is a general term that describes problem solving algorithms inspired by natural selection, the survival of the fittest and theories of evolution. The theories of evolution were built upon biological observations discovered by Charles Darwin in the 18th century [47]. Many different evolutionary algorithms have been proposed over the past thirty years. In general, EC has divided into four main directions; Genetic Algorithms [88], Evolution Strategies [3], Evolutionary Programming [67] and Genetic Programming [104] [106] [107]. Evolutionary computation focuses on search and exploration in the solution landscape (known as the search space). Search space is a term to describe a large collection of potential solutions to the problem. Each potential solution is a point located in the search space and its neighbouring solutions tend to be close in terms of how well the solution fulfils the solution criteria. The basic principle of evolutionary computation can be described as a stochastic optimisation method that operates on a set of potential solutions. The subsequent solutions are the modified versions of parent solutions. The modification is performed by two basic principles of evolution: natural selection and variation. Natural selection means that the one that is stronger than the rest is more likely to survive in the environment and is able to pass on its genes to the next generation. Variation means that the offspring in the next generation inherit its parents' genetic information and features. In terms of EC, the process of

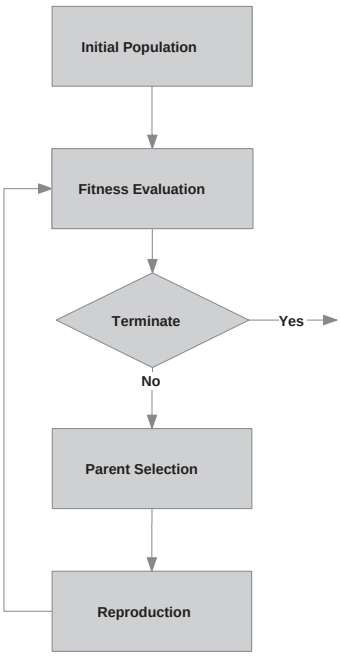


Figure 2.1: Flow chart of evolutionary algorithm.

natural selection is that a potential solution is evaluated by using a set of or a single criteria (known as fitness function), which is then given a fitness. The higher fitness solutions have more chance of being reproduced in their offspring in the next generation. Variation is achieved by recombination and mutation processes (known as reproduction). Recombination represents the offspring solutions that are reproduced by combining its parents' genetic information. Mutation represents the offspring solutions that are modified by a set of predefined genetic operators. These operators modify the genetic information to ensure new solutions are being created. Figure 2.1 shows a general flow chart for an evolutionary algorithm in evolutionary computation and Listing 2.1 shows the detail of each process.

Listing 2.1: Process of evolutionary algorithm

```
Step 1. Initial population: Create a pool of potential solutions
randomly or select from existing sources.

Step 2. Fitness evaluation: If any of the solutions satisfied the
criteria or the maximum generation was reached, then the process
is terminated. The system outputs the fittest solution.

Step 3. Parents selection: Use some kind of method (e.g. Rank
selection: choose some high fitness solutions) to select parents
from the population pool then use them in the reproduction
process.

Step 4. Reproduction: Use a variation of recombination and
mutation genetic operators to reproduce offspring, and then
replace the lower fitness solutions with high fitness offspring.
This step is a means of implementing the survival of the fittest.

Step 5. Go back to step 2.
```

Evolutionary algorithms have been shown to be efficient at guiding random search algorithms for problems with a large search space. The modularity in EA is straightforward; the representation, fitness function, parent selection and operators modules are independently customised according to the needs of the problem. In general, the major issues of designing an EA system are to define fitness functions and the encapsulation of solutions in a suitable genotype for evolution. The representation of the solution has an immediate effect on the algorithmic search and explore solutions in the search space. A typical example of using evolutionary algorithms to solve

real world problems is in the domain of engineering. For example, electric circuit design can be evolved by randomly assigning component connections to a fully functional electric circuit [117].

2.1.1 Genetic Algorithms

Genetic Algorithms (GAs) were invented by John Holland [88] and his students at the University of Michigan. There are two key elements that make GA successful. GA is capable of encoding the search space into an effective representation and domain-specific knowledge can be embedded into the fitness function. Due to GAs being a global, parallel, search and optimisation algorithm, it has been applied in a variety of optimisation problems, such as job-shop scheduling problems and circuit design. Some recent applications of genetic algorithms can be found in [27] [84]. GA works under EC's framework; population, fitness assessment, reproduction and the survival of fittest. Typically GA encodes a pool of solutions presenting it in a binary format (e.g. a string of 0s or 1s) referred to as genotype, and solution is referred to as phenotype in biological term. However solution representation is not limited to using the binary string format, as integer and floating-point string formats are also commonly used. The mapping between genotype and phenotype is problem dependent. Each solution is assigned a fitness, which indicates how well the solution fulfils the problem. The fitness is evaluated by a fitness function which is a set of criteria that are defined by the system designer. The fitness function is heavily dependent on the designer's knowledge about the problem and it should be defined in such a way that represents the solu-

tion which can be evaluated with a scoring system. The following describes the reproduction process in simple GA systems. The reproduction process can be decomposed into three processes:

Parent selection This process ensures individuals with a high fitness being selected for reproduction. Various methods for the selection of parents have been proposed. Two popular methods are 'Rank Selection' and 'Roulette Wheel Selection'. The former works as the best parents are selected according to their fitness ranking. The latter method can be imagined as similar to a roulette wheel where a proportion of each section is assigned to each individual. The higher the fitness of individuals the bigger their surface area in the wheel, thus bettering their chances of being selected.

Crossover This process ensures that a variety of solutions are reproduced in the next generation by recombining selected parents. Crossovers such as one-point, N-point, uniform and order crossover are popular and well studied in the literature.

Mutation This process ensures that new solutions are reproduced in the next generation, usually by applying a small change to the individual. Bitwise mutation is a common method for mutating binary strings, which works by randomly flipping an individual bit in the string. Other popular mutation methods such as random resetting, swap mutation, creep mutation and scramble mutation also appear in some of the literature.

More details of the parent selection, crossover and mutation methods can be found in [48] [137] [64]. Figure 2.2 shows a simple system which uses a genetic algorithm to solve the simple maximum integer problem within three generations. The objective of this problem is to find a maximum decimal number of 8-bit binary string. Figure 2.3 is a typical fitness-generation graph that shows the convergence of an GA system.

2.1.2 Random Mutation Hill Climber

In general, the Random Mutation Hill Climber (RMHC) algorithm [70] is devised from the Hill Climber (HC) algorithm [178]. HC is a type of iterative improvement algorithm. Hill climber works similarly to the idea of hill climbing; it takes a solution at current iteration, then at the next iteration the current solution is modified in some way to create a new solution. If the fitness of the new solution is better than the current solution, then the former replaces the latter, else the current solution remains the same at the next iteration. A simple HC works efficiently when the surface of the search space landscape is smooth and gradually leads to the global optima. However, if the landscape contains a few local optima, a Random-Restart Hill Climber algorithm can overcome this by performing a series of HCs. Each result will be saved until a fixed number of trials end, then the best solution will be returned. For RMHC, it can be described as a simple HC with a genetic mutation operator that is used to modify solutions. In Forrest and Mitchell's paper [70], GA and RMHC are compared by analysing using "Royal Road" function. The result shown was that RMHC significantly outperformed GA.

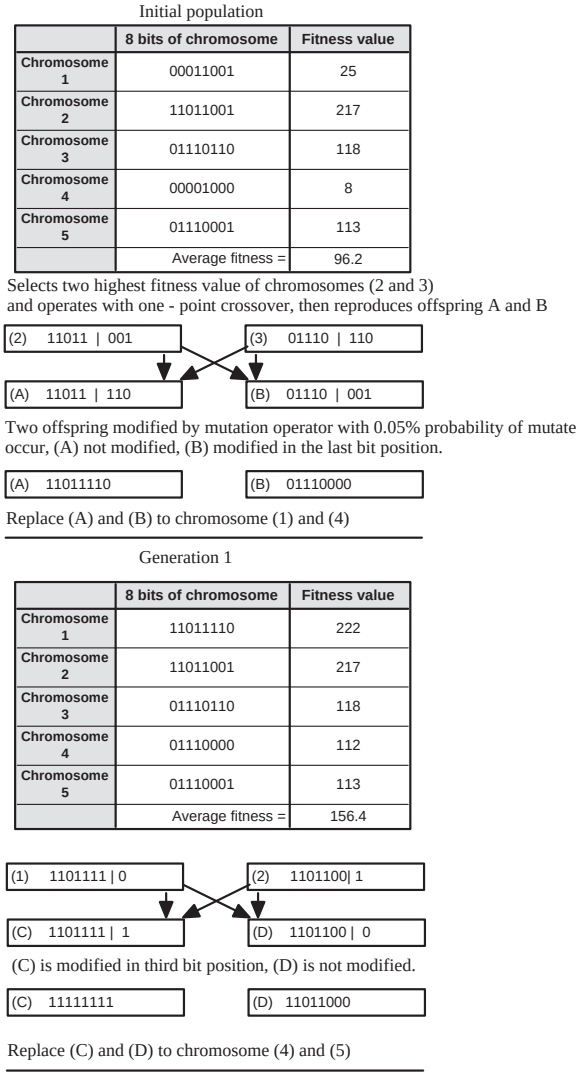


Figure 2.2: GA example.

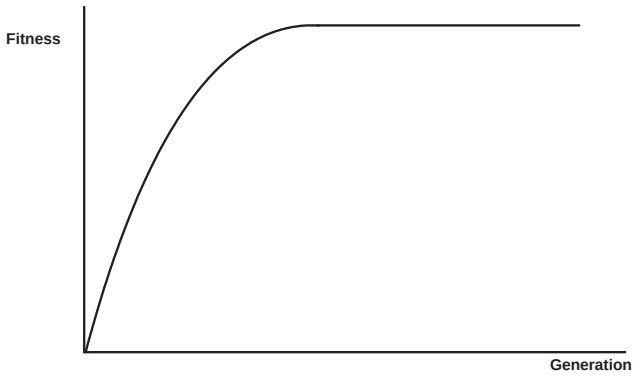


Figure 2.3: Typical fitness curve of a GA evolution.

2.1.3 Multi-Objective Evolutionary Algorithms

In multi-objective optimisation problem, more than one objective needs to be satisfied and these objectives are in conflict with each other. If these objectives do not interfere with each other, it is meaningless to call it a multi-objective problem, because each objective could be optimised separately. Traditionally, the weighted-sum method has been used for multi-objective optimisation in most cases. Weighed-sum is the simplest multi-objective method for transforming a set of objectives into a single objective. Each objective is assigned a weight by the user. Define as Maximum/Minimum $(x) = \sum_{i=1}^{n} w_i \times f_i(x)$, where $f_i(x)$ is the fitness function i, w_i is the weight of fitness function i and n is the total number of fitness functions [50]. This approach has an advantage only when the weights of the fitness functions are known.

In general, Multi-Objective Evolutionary Algorithms (MOEAs) aim is to search for Pareto front solutions know as non-dominated solutions, because these solutions cannot dominate each other, so they are the optimal.

The final selected solution for solving the problem is usually evaluated by other predefined criteria or is left up to the end user to choose among these non-dominated solutions. The MOEA approach has the advantage over the weighted-sum method when the weighting of each fitness function is not given or is unknown to the designer. For more information regarding different kinds of MOEA is refer to [228] [226] [229] [22].

2.1.4 Fitness Distance Correlation (FDC)

Fitness Distance Correlation is a tool for predicting the performance of a GA on a problem with a known global optimal solution. FDC is proposed by Jones and Forrest [95]. It measures the correlation coefficient of fitness and the distance between the optimal solution and sampled solutions. Equation 2.1 is the definition of FDC,

$$FDC = \frac{cFD}{sF \times sD} \tag{2.1}$$

Where $cFD = \frac{1}{n}\sum_{i=0}^{n}\left(f_i - \overline{f}\right)\left(d_i - \overline{d}\right)$ is the covariance of F, a set of individual finesses $F = \{f_1, f_2 \ldots f_n\}$ and a corresponding set of distances $D = \{d_1, d_2 \ldots d_n\}$. sF,sD, $\overline{f}$ and $\overline{d}$ are the standard deviations and means of F and D respectively.

In Jones and Forrest's paper [95], they suggest that FDC is able to indicate which class a problem belongs to. For a minimisation problem, fitness is the error away from the global optimum.

Class 1 Misleading ($FDC \leq -0.15$), the fitness tends to decrease (better) when an individual is moving away from the global optimum.

Class 2 Difficult ($-0.15 < FDC < 0.15$), there is a weak correlation between fitness and distance from the global optimum i.e. when two individuals have similar distances, their fitness can be very different. The problem becomes difficult for an evolutionary algorithm.

Class 3 Straightforward ($FDC \geq 0.15$), the fitness tends to decrease (better) when an individual is approaching the global optimum.

2.2 Algorithmic Music Composition

Algorithmic music composition can be described as a set of well-defined rules used for generating music. In other words, once the system is built then composing music is fast and cheap. Researchers Hiller and Isaacson [87] were possibly the first to study algorithmic music composition scientifically. They based their research on the Markov Chain model [131] [79]. Since then many researchers have attempted to address different problems of algorithmic composition using deterministic and stochastic techniques.

Deterministic technique This technique is based on generating music according to predefined rules and users provide inputs to the system. A simple deterministic system has a one-to-one mapping between input and output. Cope's Experiments in Musical Intelligence system (EMI) [37] [38] is an example of a deterministic music composition system. EMI is implemented based on an expert system technique. In an expert system, domain specific knowledge is encapsulated and represented in a set of rules. When new inputs feed into the system, it then performs

inference [172]. The EMI system uses a set of rules to decompose the entire music samples into different pieces and then reconstructs new music by recombining pieces. Other examples of deterministic music composition systems are: the PGA-system by Biles [18], where musical sequences are generated by a Fibonacci sequence [53]; and Beyls's cellular automata based music generator [14].

Stochastic technique Using this technique, music is generated based on probability. Markov models are popularly used in stochastic music composition systems. A well-known real life example of using a stochastic technique to generate music is Mozart's dice game. This game uses a dice roll as a random generator to select pre-written bars of music and then these bars are combined to form a full music piece. All pre-written bars were composed in such a way that every bar harmonises with the others, hence, the combined music satisfies certain musical requirements. From the last decade, stochastic search algorithms such as evolutionary algorithms are also being used for music generation and are increasingly receiving great attention [25] [73].

2.2.1 Music Information Extraction

A musical piece might consist of multiple instruments or voices that are playing simultaneously. Non-tonal instruments such as percussion might also appear in the piece. Each full piece of music in the samples may contain rich information, depending upon the aim of the composition system, as some musical information is not necessary for the system. Music information

extraction is a process that is commonly used to extract the required information out of the musical piece. Depending on the system requirements, the extracted musical information can be melody, rhythm, polyphonic music or even percussion. A typical example is that a perceived melody can move across multiple instruments simultaneously in polyphonic music, therefore it is necessary to use some kind of algorithm to extract monophonic melody from a polyphonic music. However, choosing a melody extraction algorithm can be difficult. Melody extraction algorithms usually originate from music perception research and are tested by the music information retrieval community [208]. Most common melody extraction algorithms are 'All-Mono' and 'Top-Channel'

All-Mono "Combine all channels and keep the highest note from all simultaneous note events." [210] Musicologists suggest that when humans listen to polyphonic music, the highest pitches are more perceivable than the lower pitches.

Top-Channel "Keep only the channel with highest average pitch, then keep only the highest notes." [210]

Uitdenbogerd and Zobel [210] [211] reported that 'All-Mono' is the most successful extraction algorithm in their experiments for music retrieval. Doraisamy and Ruger [55] also investigate the fault-tolerance of using the N-gram approach to index music, where music information is extracted by an 'All-Mono' algorithm from the samples. The result was promising when compared to other tested techniques.

2.2.2 Music Representation

Music representation is a critical part of any musical system. First, it determines how much information in a piece of music is represented. Second, music information can be shared between systems via the same representation. It is important to understand the characteristics of music representations when designing a suitable encoding method for genotype-phenotype mapping in evolutionary music composition systems. Below describes four categories of music representations and its associated research areas in computer music:

Audio The natural way of music representation in recording and audio forms. Digital audio scheme, such as ISO-MPEG Audio Layer-3 (MP3) and Pulse Code Modulation (PCM) are the common music representations. The main research areas included automatic music transcription [9], music classification [6] [209] [13], music indexing for music retrieval [33] [220] and music analysis [68] [46] [69] [86].

Visual Traditional musical manuscripts are written in music notation. In the research of optical music recognition, musical manuscripts are transformed into digital images, and then the recognition system transforms these images into digital music representations such as audio. Music in digital form is easy to access and manipulate by computer for performing music analysis [112] [32] [129].

Symbolic Music information is represented in a symbolic form such as event-based recordings MIDI (Musical Instrument Digital Interface: MIDI Manufacturers Association 1996); text-based music language:

GUIDO [89], XML [176] and MusicXML [77]; Hybrid representation such as MPEG-4 which allows the synchronisation of symbolic music elements with audiovisual events [8]. Music analysis and manipulation in symbolic data are more practicable for data processing on both playback and notation applications. The research areas include music matching [210], theme or melody extraction [130], voice separation, music information retrieval and music analysis [163] [40] [5].

Metadata Musical pieces are represented at the abstract information level such as description [152] and, cataloguing and bibliography [182].

In the literature, it seems that the MIDI protocol is too rich and complex for automatic music composition systems. A few abstract music representation techniques have been proposed, and each of them has its own strengths and weaknesses to match individual system requirements. The following are the common abstract music representations that are used for representing musical pitch but note duration is ignored:

Absolute Pitch This representation is performed by mapping each integer value to a specific musical pitch. The pitch assignments of an integer usually follow the standard MIDI protocol, e.g. middle C is assigned to an integer value of 60. To raise middle C to C#, the pitch is raised by one semi-tone, hence, the integer for that is 61. +1 equals raising one semi-tone, -1 equals dropping down one semi-tone. Using this representation, if any mistake has been made in a melody, the mistake is only at individual note.

Pitch Difference This representation is sometimes referred to as pitch interval. Each integer value represents an interval between two successive pitches, "+" represents a rising interval and "-" represents a falling interval. Musical key transposition is easily done in pitch difference representation, by only changing the starting pitch to a desired pitch then the rest of the pitches are changed. When a melody is transposed to a different key, the generality of the sound will still hold. A major problem of using pitch difference representation is called "Peculiar" coined by Cruz-Alcazar and Vidal [44]: if any mistake is made on a single pitch or on more than one interval, then all the succeeding pitches are affected. This problem becomes critical for a fitness function that is sensitive to each pitch in the sequence, the fitness of the musical sequence might be totally different.

Scale Degree This is one of the traditional music representations in western music theory. Scale degree refers to a note's position in the scale, for instance, C is the 1st degree in C Major Scale; D is the 2nd degree; and so on. Typically the number of notes in a scale within one octave for a particular musical key is less than 8. The advantages and disadvantages of using this representation is similar to pitch difference. Music transposition is easily achieved by changing the root (first) position of the scale. However, this creates a problem for MIDI files, as the MIDI protocol does not support scale information. Also the musical scale and key signature information must be known in advance.

Contour This representation is developed by Parsons [158]. Parsons's idea

provided an easy but effective solution for non-musical people to recognise and search for a melody or theme. Contour representation is based on the measure of whether the note goes up or down or remains the same between two successive notes. A melody is represented in a string of contour symbols, "U" representing pitch up, "D" representing pitch down and "R" representing pitch repeat. Contour representation is popular in query-by-humming systems, which can be described as music retrieval systems that take a user-hummed melody and performs a query in the music database. Since the capability of humans to present music is weak, the user often makes some errors like not humming the correct absolute pitch, interval or omission. The contour representation addresses such mistakes and this makes it more fault-tolerant than other representations for query-by-humming systems [102] [167].

2.3 Evolutionary Computation for Creativity

Gero and Kazakov [74] stated that creativity is the generation of "surprising and innovative solutions" and creating "novel solutions that are qualitatively better than previous solutions". We inferred from the statements that when a human is creating ideas, there are some mechanisms in the human brain responsible for searching for and exploring new ideas and also performing an evaluation of these ideas. For the process of searching, it involves some criteria for evaluating ideas in the search space, and finding the best idea among all ideas visited. For the process of exploration, it involves some kind of mechanism that allows the human brain to explore new ideas in different

areas within the search space or even transform ideas between different domains. Unfortunately, the cognitive science community still cannot find the best explanation for the above mechanisms in human brains and it remains a 'Black-Box'.

Natural evolution is very good at solving creative problems. For example, all living creatures evolved generation by generation, the more adaptive their body structures are to different environments the more chances they have of surviving. In the literature, evolutionary computation techniques are shown to be efficient search methods, when faced with problems in the creative domain and where there is a large search space. For design problems, Bentley [10] used his generic evolutionary design system to evolve twenty coffee tables from a set of criteria. Many of the designs are reviewed as creative, original and unusual ideas. For artistic painting, Sims [187] developed an evolutionary system with a genetic algorithm that evolves a series of computer images. His system is built based on an interactive approach, where the fitness function is replaced by a human who evaluates each evolved image. Bentley's and Sims's experiments fully satisfy Gero and Kazakov statements as the experimental results showed that an evolutionary algorithm evolves creative solutions for the domain of engineering and artistic problems. From another perspective, Goldberg [76] stated that creativity requires "Transferring useful information from other domains". Clearly, an evolutionary algorithm does not satisfy this statement because currently there is no such technique to identify and transfer knowledge between domains to aid searching solutions.

For the purpose of explorative evolutionary computation (creative design), Bentley [11] focuses on the component-based representation for evolu-

tionary computation. The principle of explorative evolutionary computation is that a solution is constructed from a set of low-level components. EA acts as an exploration mechanism to arrange and link them in various dimensional spaces. The system is expected to search for new and interesting solutions instead of optimal solutions. Bentley used a simple application to illustrate proof-of-concept of his framework, the task being to evolve a paper shape that is able to fall as slowly as possible to the ground from a specific height. The application is implemented based on a GA, where chromosomes are sets of 16-bit binary strings, and the mapping of genotype-phenotype is called "Embryogeny". It works by having eight vertices with (x, y) parameters from each chromosome. Each vector is joined to its successor by an edge (i.e. join the last vector to the first with an edge, and fill the resulting shape). The paper will be cut out and tested manually. The fitness is calculated by 1/(time 1 + time 2 + time 3). The result showed that most shapes evolved "using the same technique of having a smaller flap which causes the shapes to rotate as they fall. The main benefit of this solution slipping sideways in the air and plummeting to the ground at tremendous speed".

Machwe et al. [119] proposed an object-based representation to overcome the issues of aesthetic criteria. The system was implemented based on interactive evolutionary programming with agent technology. The idea behind using agent technology is to reduce the workload of human interaction to the EA system. These agents are rule-based agents, responsible for developing initial solutions and later for maintaining and repairing solutions during the evolution. In the problem tested, such as in the domain of bridge construction, a bridge is constructed based on the number of span beams and

support parts required. The agents ensure the sizes and shapes of objects are constructed within the given constraints. They also analyse the user's preferences in ranking the individual designs. If the user disagrees with the agents ranking of an individual design, the agents will perform an adjustment of the agents' weighting to match the user's preference. The object-based representation has an advantage over string and tree-based representations while the problem domain is solving the components arrangement. For more comprehensive tutorials and surveys in the aspect of explorative evolutionary computation, refer to [11] [12] [28].

Evolutionary Music Composition

From the perspective of human musical creation, musical ideas are often thought to be generated in one of two ways: 1) Inspiration, a composer has musical ideas that appear instantly. For example, Mozart had the ability to compose an entire piece of music in an instant. 2) Perspiration, a composer develops his musical ideas in parts or by reworking existing musical ideas. For example, Beethoven composed his music using this technique."In short, creativity comes in two flavours: genius and hard work" (Jacob [92]). Modelling inspiration is not an appropriate method due to the lack of understanding of the psychological mechanism of inspiration. Alternatively, modelling perspiration could be more feasible. Modelling perspiration involves understanding how musical ideas are developed by observation and exploration. When looking at the evolutionary algorithm there are similarities between it and the perspiration process of how humans compose music, see Listing 2.1 and 2.2.

Listing 2.2: Perspiration process of human music composition.

```
Step 1. Initial population: Creating or selecting the initial
musical ideas either randomly or from existing sources.

Step 2. Fitness evaluation: If the composer is satisfied with the
musical ideas created, then the composition process finishes.

Step 3. Parents selection: Selecting the composer's preferred
musical ideas in the population.

Step 4. Reproduction: Change or modify the selected musical ideas
according to the composer's predefined rules. Replace the
non-satisfied musical ideas with the newly created musical ideas.

Step 5. Go back to step 2.
```

Evolutionary music composition is a relatively new paradigm that uses stochastic algorithms compared to other paradigms that are using deterministic algorithms to generate music. Evolutionary algorithms are strong at exploring and searching multiple solutions in a parallel fashion to satisfy the solution criteria. These abilities make it suitable for a music composition system. For more details of how evolutionary algorithms are suitable for music composition refer to the excellent papers by Burton and Vladimirova [25], and Gartland-Jones and Copley [73]. After a decade of development, evolutionary music is divided into two paradigms for defining fitness functions that are used for evaluating music. These paradigms are known as the interactive and automated music evaluator paradigms. The idea of the interactive paradigm is that a human serves as a music evaluator in the EA system. On

the other hand, the idea of the automated music evaluator paradigm is to use some kind of defined or learning algorithms to evaluate music.

2.4 Framework for Music Composition Systems

The framework consists of the conceptual stages and components for building systems. Each of them is well-defined and responsible for a specific function within the system. Most importantly, designers can use the framework to build systems stage by stage, while understanding the conceptual aims of each component. Designers can have a direct comparison of systems that are built using the same framework. In the literature, frameworks for machine composition are seldom explored. Pearce and Wiggins [160] were probably the first to investigate this, and proposed a general framework for the evaluation of machine compositions. They introduced a methodology that attempts to evaluate composition systems objectively through empirical experimentation. The success of the machine composition system must be measured both internally and externally with no bias from the system developer. Pearce and Wiggins's framework consists of four stages:

Specification of composition aims Composition goals and motivation must be specific and clearly defined before any development. This is because the designs of subsequent components depend on the composition aims. When using machine learning techniques for music composition, there are generally two categories: a music composition system that is either

modelling a particular musical genre or a style of particular composer. The designer also needs to define what kind of composition the system is processing. For example, polyphonic music, monophonic music, rhythm, percussion and so on. Any additional issues must be stated at this phase.

Inducing the critic A critic is induced by using some kind of machine learning technique to capture the patterns from music samples. The critic is the internal evaluator of the generated music. The generated music is required to satisfy this critic. The choice of the machine learning technique should be justified on the basis of the composition aim.

Composition Using any computational method to generate music, the output compositions must satisfy the critic. This phase implements the algorithm for music composition.

Evaluation In order to objectively evaluate the success of a machine composition system, external evaluation is required, which removes the bias from the system designer. If the composition aim is to compose music that satisfies music theory, then it make sense to use some criteria that are derived from music theory to evaluate the generated music. For example, Phon-Amnuaisuk et al. [162] evaluate the generated music by using criteria that are used for examining first year undergraduate students' harmonies. If the composition aim is not to satisfy any music theory or there is no expert knowledge available to evaluate the generated music then an empirical experiment is necessary. For example, the generated music can be evaluated by asking human subjects to identify

whether the music is generated by machine or a human. If the results show that the mean proportion of system generated music that was correctly classified is the same or lower than that expected if the subjects answered randomly, then we can conclude that the system is a success since the generated music is indistinguishable from human-composed music.

Pearce and Wiggins claimed that there are three attractive features in their framework. First, the critic is inducted from the given samples using a machine learning technique. Second, the generated music is internally evaluated by the critic. Third, the success of the system is externally evaluated through empirical experimentation, removing any bias from the system designers.

2.5 Literature Review

This section reviews some projects and applications that are under the paradigms of interactive and automated music evaluator. Note that in the section below, some of the reviewed music evaluators are not implemented in evolutionary algorithm systems at all. These evaluators are used solely or incorporated with other search algorithms for music composition. The reason we included them here is that these automated music evaluators have a great potential to become fitness functions in evolutionary music composition systems. However, here it is impossible to review all the existing works, [25] and [135] are the survey papers for the reader who would like to cover more details in the realm of evolutionary music.

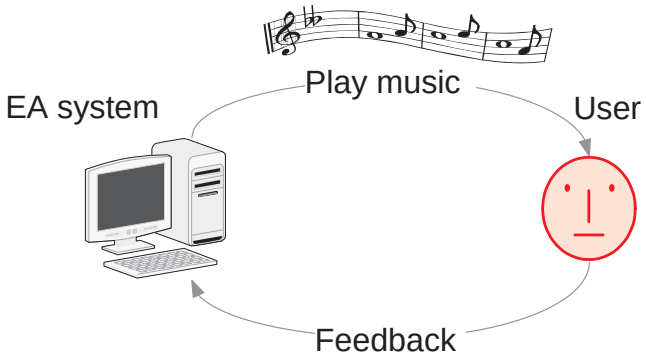


Figure 2.4: Interactive-based evolutionary music composition system.

2.5.1 Interactive Paradigm

In this paradigm, a human serves as a music tutor who gives feedback on the evolving music in the EA system. Figure 2.4 illustrates a typical interactive-based EA music composition system. In the system, at each generation a human is required to assign a fitness to each evolved music. Then, the system attempts to generate 'better' music in the next generation. This type of human interactive system (sometimes referred to as 'human-in-the-loop') promises to produce an acceptable result only when the tutor provides rational ratings. Inefficiency, subjectivity and inconsistency are the issues that commonly arise when using interactive-based fitness functions. It is well-known that the music evaluation process in this type of system is time consuming and the tutor might change his personal music criteria over time. This could lead the system to behave inconsistently and be too user dependent.

Numerous interactive-based evolutionary systems have been proposed in the past ten years. The main differences among them are that they use

different types of evolutionary algorithm, representation of genotype and phonotype, mapping process and genetic operators. For a general survey of interactive-based evolutionary systems refer to [196]. The following highlights some of the relevant interactive-based evolutionary music composition systems.

GenJam [17] is probably one of the most famous interactive-based evolutionary music composition systems developed by Biles. Biles's system can be compared to a novice jazz musician that is constantly learning to improvise. GenJam plays its solos over the accompaniment of standard jazz rhythms in a session. Biles implemented GenJam using a genetic algorithm with an interactive fitness function. Each individual is evaluated by a user who listens to it and gives it a fitness. The music population in GenJam is comprised of two hierarchically interrelated populations, 'Measure' and 'Phrase', which mirror the hierarchical nature of musical scores. In the 'Measure' population, each individual consists of a number of MIDI musical events (short melody). Each melody is randomly generated but the choice of notes is determined by some rules. For example, given a chord, the choice of pitches for generating the melody is within a particular musical scale, ensuring that the generated melody harmonises with the given chord, so that no 'wrong notes' are played. In the 'Phrase' population, each individual consists of a number of indices of measures in the 'Measure' population. During in the evolution, GenJam used a GA with musically meaningful genetic operators to evolve phrases (solo melodies) through the interactive fitness function. These operators are designed based on melody composition techniques such as notes transposition, sort notes ascending/descending and reverse. Biles concluded that

"Genjam showed that a GA can be a useful tool for searching a constrained melodic space". Later, Unehara and Onisawa [212] implemented a similar system which evolves 16-bar melodies. The generation of initial population and genetic operators were designed based on music theory.

Thywissen [199] developed a hybrid evolutionary music system named GeNotator. In the system, rewriting rules are used to define the genotype structure of each chromosome and the music (phenotype) is rendered by the defined grammar and gene settings. The user is able to define rewriting rules using a music grammar scripting language, which is a flexible scripting language able to create some rules such as key change, transposition and so on. An interactive fitness function is implemented in the system to evaluate the evolving music. GeNotator is flexible in that the user has a complete control of the underlying mechanism of the system from the music representation through to music evaluation. In other words, the system's parameters are customised for individual user is needs and reflect his/her preferences.

The following is a summary of the advantages and disadvantages of interactive paradigm:

Advantages:

Searching without evaluative criteria System designer does not need to design a general algorithm to represent the subjective criteria. Instead, the evaluation process is performed by a human judge.

Supervisory searching The human judge guides and forces the evolution process to search in his preferred areas in the search space, which is useful when the evolution is trapped in local optimal solutions.

Exploring different solutions The user has a close look at individual solutions, the longer the user operates the system, the more solutions he/she will see.

Artistic reflection The interactive fitness function approach allows musician to evolve creations using their senses, hence, reflecting their true artistic tastes. This approach helps musically unskilled users compose music to reflect the user's musical taste.

Disadvantages:

Inefficiency Since a human judge is required to evaluate each potential solution, it is time consuming. Interactive-based systems are more favourable for evolutionary art painting, as the speed of evaluating paintings is much faster than for evaluating music. Users are able to evaluate a series of evolved images on the screen for a parallel comparison [187] [36] [109] [63]. It is difficult for human to perceive and evaluate multiple musical pieces in a parallel fashion.

Subjectivity A human judge might be biased as their decisions will be influenced by their experience, culture and knowledge.

Inconsistency A human judge might change his/her mind during the evaluation process. Factors such as the emotional state, tiredness, boredom and being influenced by other solutions can result in inconsistent judgment.

'Black-Box' Both system designer and user do not understand the evaluative criteria, in other words the trajectory of searching and exploring

the musical ideas are unknown and the search landscape is difficult to determine and predict.

2.5.2 Automated Music Evaluator Paradigm

In this paradigm, music is objectively evaluated by well defined criteria which are present in computing algorithms. This section reviews and discusses various types of music evaluators and summarises them into different categories. Figure 2.5 shows the general hierarchical structure of the automated music evaluator paradigm, where each node represents a type of technique to implement a music evaluator.

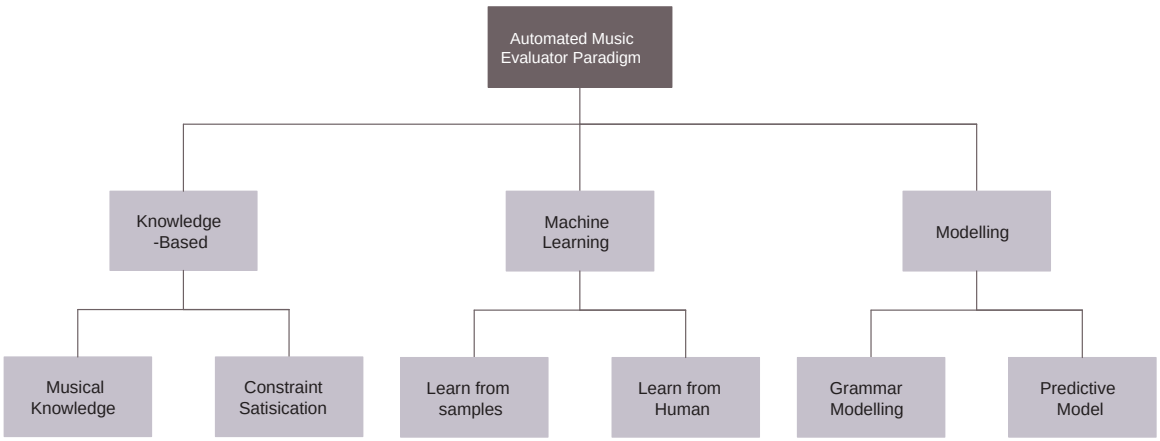


Figure 2.5: Hierarchical structure of automated music evaluator paradigm.

Knowledge-Based

A knowledge-based system is a general term that is given to a system that uses domain-specific knowledge (from a designer/expert) to solve a problem. It searches solutions based on the guidance of built-in knowledge, similar to

an expert system. When the solution criteria are clearly formalised, using this approach is effective when based on executing a set of logical rules. In other words, the constraints for the solution are precisely defined. Such systems are expected to produce a particular type of solution and these solutions satisfy the given rules.

Musical Knowledge

Under this category, musical knowledge is embedded into the components of the EA systems. Musical knowledge is not necessarily only embedded into fitness functions, it could be embedded into the representation and genetic operators as well. Typically, if a set of fitness functions is devised from the music theory, then the outputs are expected to reflect the supplied knowledge. This type of embedding becomes a transformation from rules to constraints for the problem. The transformation is dependent on the problem, each solution will be scoped by the transformed constraints. In general, transformation is a process that transforms a Constrained Optimisation Problem (COP) to a Free Optimisation Problem (FOP) [64]. COP consists of objective functions, free search space and constraints (boolean function). FOP consists of objective functions and free search space.

Typically, in the domain of music composition, melodies satisfy some sort of musical constraints that could be predefined by the composer or derived from music theory. Tonality and musical harmonisation are the most common musical constraints in western music.

Tonality Traditional western tonal music is described in terms of scale of

notes. (E.g. in the C Major Scale: C D E F G A B. C is the tonic note). Usually a piece of music consists of notes and chords which are within a particular chosen scale. Notes or chords within the same scale are more important than others that are outside the scale. Often the notes that are outside the scale sound odd relative to those corresponding to the scale or chord. A typical tonal melody contains many tonic notes. The chosen notes or chord progression will be based on the relationship between the tonic notes. The main differences between melodies within the same scale are the note arrangements and frequency distribution of the notes.

Musical Harmonisation Harmony means playing more than one note simultaneously, where the combined sound is considered pleasant for the human listener. This type of harmony is referred to as vertical harmonisation, e.g. four voice music: Soprano, Alto, Tenor and Bass. Music interval theory is one of the most common rules for vertical harmonisation such as unison, perfect fourth, perfect fifth, perfect octave. Another type of harmonisation is called horizontal harmonisation, where the harmonising notes are played sequentially.

It is not intended as part of this thesis to describe the complete list of tonality and harmony rules. Treatises on tonality and harmony are often studied by musicologists. These studies are subjective, there are different interpretations of the different composers leading to a variety of musical styles. For more details of music theory refer to [195] [197] [198]. The following section describes some of the evolutionary music composition systems that have

embedded musical knowledge into their fitness functions, representation and genetic operators.

Tonality and Musical Harmony Constraints

Biles's GenJam [17], and Papadopoulos and Wiggins's musical GA system [155] both evolve jazz melodies. These systems use a similar approach to handle the tonality constraints within the chromosome representation. The chromosomes (melody) in their GA systems are represented in the degree of scale, which is relative to the current selected chord. Degree of scale representation ensures that each melody consists of scale notes only, hence, the melody satisfies the tonality constraint and its notes are harmonised with the corresponding chord.

Spector and Alpern [191] proposed an evolutionary music framework based on the idea of separating the music criticism and cultural context in a genetic programming system. Their system is able to generate jazz melodies in a 'Call and Response[1]' form, four measures per melody. The function set consists of thirteen musically meaningful operators such as 8VA which transposes an input melody up one octave. RETROGRADE: reverses an input melody and so on. The terminal set is a single melody, which is provided by the user. The evolved melodies are evaluated using five predefined critics to measure the similarity between the input and output melodies and give fitness. Multiple melodies can be input in turn into the GP program, using the so called 'Case-based fitness' evaluation process. An example of criti-

[1]It is a turn-based jam session between two jazz musicians. One plays a short melody, the other responds with a similar melody (which is usually a modification of the other melody played).

cism is TONAL-RESPONSE-BALANCE: note by note comparing the input and output melodies. They reported that in the experiment the best individual (GP program) produced melodies satisfied the critic, but it did not satisfy the system designers. Grachten [78] developed a system called Jazz Improvisation Generator (JIG), which generates jazz melodies by combining musical constraints with the distribution probability of notes. Three major constraints were used: Tonality, the improvisation must be tonal to the key of the music; Continuity, the melodic contour of the improvisation must be smooth and not convoluted; Structure, interrelated groups of notes are identified and used. During the evolution, these constraints are used to evaluate music. Later, Spector and Alpern [192] further enhanced their system by replacing the music critic with a neural network classifier, which evaluates music and indicates whether it is good or bad. They reported that the output melodies evolved using combined case-based and neural network music critics sound more pleasing than just using a stand-alone neural network critic. The neural network failed to capture deep musical structure from the music used for the training. It is worth mentioning another work by Spector et al. [193]. Their paper gives a brief introduction to a possible way of implementing an evolutionary music system with respect to ecological notion. A musical note is produced by an artificial creature in a 3D virtual world, for example, when the creature's legs strike the ground, a piano tone is played. Music is rendered by music mapping algorithms between behaviours and musical notes. The idea proposed is that music is evolved based on the interactions between the creature's behaviours, the world and other creatures's behaviours. The music itself is a by-product of the creature's behaviour in the environment.

They claimed that the music is evolved from a complex and adaptive process, physical constraints (such as creature's body movement) are applied implicitly to the music. For a survey of A-life evolutionary music refer to [136].

Khalifa and Al-Mourad [98] developed an evolutionary music composer system with two tonal fitness functions, the system consists of two stages of evolution processes. At stage one, short motifs (short segments of primary music pattern) are evolved using a note interval fitness function. If the interval of any two successive notes in the motif is not deemed acceptable by the predefined threshold, then the motif will be rejected, otherwise accepted. At stage two, 'best keep' motifs from stage one are used for evolving musical phrases. The fitness function at this stage is a tonal ratio function. The ideal ratios of the notes are predefined by the user. In their paper they have chosen 60% for the notes that make up the chords within a key, 35% for the notes remaining in the key and 5% for the notes which are outside the key. In Khalifa et al.'s paper [99], similar fitness functions are used in cooperation with some music theory driven fitness functions integrated in formal grammar in a Chomsky Normal Form e.g. B $\rightarrow$ beaA where each character is a musical note, lower and upper case characters are the non-terminal and terminal symbols respectively. The idea of two stages of composition is to use a set of motifs to form full pieces by arranging the motifs in an appropriate way based on some rules. This has the advantage of breaking down the full piece into a number of reusable short motifs. So the high level structure of the evolved music is more easy to control. This kind of technique is often used by human composers who compose new music by rearranging existing short

phrases. Some researchers have begun to investigate the potential of using Grammatical Evolution (GE) [179] for music composition. One of the motivations is that there is already a strong research background on generative grammar in music theory [111]. Musical genre can be generally represented in generative grammar, when considered as knowledge represented in a form of grammar rather than a set of rules. More details of how grammatical evolution can be applied on music composition is described in [49].

Phon-Amnuaisuk et al. [162] developed a GA system that evolves four-parts harmonised melodies. The harmony constraints are handled by fitness functions and reproduction operators, giving rich domain specific knowledge to the system. For example, for fitness functions, solutions are penalised for parallel unison, parallel perfect 5th and parallel octave between melodies. For reproduction operators, ReChord: mutates the different chord types based on the melody data; PhraseEnd: mutates the end of each phrase to end with a chord in root position. Some of the evolved melodies were assessed and marked by a music lecturer according to the criteria which he used for a first year undergraduate students' harmony test. Most of the melodies earned a mark of around 30%. This low mark was due to the lack of coherent large-scale musical progression. However, they claimed that their system was better than the student harmonisers at getting the basic rules right. McIntyre [127] developed a similar system 'Bach in a box', a genetic algorithm is used to evolve a four-part baroque harmony.

Wiggins et al. [140] developed a music composition system named Vox Populi, which allows the user to adjust the fitness functions to evolve chord-based music. Three fitness functions are implemented based on musical fea-

tures: melodic, harmonic and vocal range. Allowing the user to interact with the fitness function to change its parameters, enables composers to use their own settings to evolve music according to their musical preferences. The system has the advantage of the user's subjective evaluation being embedded into the fitness function via the interactive control, which somehow overcomes the evaluation speed issue of using humans as music critics.

Towsey et al. [205] developed twenty-one musical features (melodic statistics) evaluation algorithms based on the suggestions from music composition text books by [194] [195] [90]. These proposed algorithms can be implemented as fitness functions in an EA system for music composition. To support the idea that these features are important for music composition, they conducted a list of experiments to evaluate a set of existing musical pieces (classical, traditional nursery rhymes and popular tunes) by these features. They reported that some of the features have statistically high means with lower standard deviations in the samples. Later, Reddin et al. [173] implemented six fitness functions based on Towsey et al.'s musical features in a grammatical evolution system to evolve short music composition. He reported that a simple fitness function with grammatical evolution is able to produce satisfactory results. The idea behind his work is to use grammar to act as constraints that cooperate with the fitness functions, instead of embedding the constraints into fitness functions. The system is flexible in the way that generative grammar can be custom-defined by the programmer or experienced musician.

Ozcan and Ercal [151] developed an evolutionary music composition system called AMUSE. It aims to generate melodies for improving the given mu-

sic. The system contains a single fitness function that consists of ten melodic and rhythmic feature algorithms for objectively evaluating the evolved melodies. These features included chord note, the relationship between successive notes and drastic duration change and so on. Some of the evolved music was evaluated by thirty-six students in two tests. In the first test, a small scale musical 'Turing Test' was performed. There were two solos over two musical pieces played one after the other; one solo was composed by a musician, the other was evolved by AMUSE. For each solo, students were asked to select whether it was composed by a human or a machine. The result of the test was that 52% of them provided wrong answers. In the second test, students were asked to rate five pairs of solos, each pair consisted of an initial solo and an evolved solo (after 1000 generations), using the rating from 1 (worst) to 10 (best). The result showed that for each pair, the average rating of the initial solo was lower than the evolved solo. Two-tail T-tests were performed, and 4 out of 5 pairs showed a statistically significant difference.

Dahlstedt [45] developed a real-time evolutionary music composition system for real performance. Evolved music was played using a computer-controlled grand piano. The initial music was either generated by machine or a musician who played music on the piano, then the computer captured and translated the music into the music representation that was used in the system. The type of fitness functions in the EA was knowledge-based, based on his previous experience of designing an interactive system. The genetic operators are the standard GP genetic operators. The system uses a recursive binary tree to represent music. Each sub-tree can be assigned one or more special operators, such as multiplying all amplitude/duration in the tree

by a constant value. Using a recursive representation enables gradual/constant change effects to be made on the music, including 'accelerando' (play gradually faster), 'ritardando' (play gradually slower), 'diminuendo' (play gradually softer) and so on. Such a representation avoids the evolved music containing sudden changes of pitch and note volume, making the melodic contour of the evolved music smoother. Usually, an unexpected sudden change in pitch and note volume in a musical sequence is considered to be irritating for the listener.

Oliwa [148] used GA with 'abc' music language (a type of music representation) to evolve four instruments (lead guitar, rhythm guitar, rock organ and drum) of structured rock music. The representation of individual is separated into k (the number of instruments) number of sets of genomes (each instrument has its own set of genomes). There are five different genomes with different integer genome configuration. For each run, the initial structure of the genomes set is randomly generated. Each type of genome is designed for a specific instrument with a specific set of fitness functions. For example, genome type I is for lead guitar and the choice of fitness functions are 'ascending tone scale', 'descending tone scale' and so on. Oliwa's system is a heavy knowledge-based evolutionary music system. The knowledge is not only embedded into the fitness function, but also into the structure of the evolved music. This method addresses the arrangement problems for musical phrases and instruments.

In general, a knowledge-based approach is efficient when the musical structure and knowledge are built into the system, as the output music is more meaningful in terms of musical structure, but the trade-off is less novelty

and the domain-specific knowledge must be well-defined and implemented explicitly and correctly into the system. A major drawback for knowledge-based music composition systems is that the embedded knowledge is limited to the designer's subjective knowledge and experiences. It is hard to maintain the system especially if part of the knowledge has been implemented incorrectly or needs updating. All the above reviews are focused on knowledge-based EA music composition systems. More general knowledge-based music composition systems are reviewed in [37] [38] [116] [61].

Constraint Satisfaction

This section briefly introduces how musical problems are being solved using a Constraint Satisfaction (CS) [207] technique rather than EA and provides some reference to research in this area. In this approach, musical problems are converted to a Constraint Satisfaction Problems (CSPs) format, and then constraint satisfaction algorithms are used to search for solutions. The rules of the problem are converted into a cost function that evaluates the fitness of the solutions that satisfy the constraints.

An excellent paper by Truchet [206] proposes a visual programming language for composers to define their own musical constraint problems, and the system comes with an adaptive search algorithm [34] that is used for searching solutions. A number of researchers used Truchet's visual programming language to solve musical problems such as sorting chord and rhythmic patterns [206]; Nzakara harp and Ligeti's textures [29]; asynchronous rhythms, gestures and harmonisation of a rhythmical canon [35]; rhythmic oddity property [30]. Other similar works are in [149] [150] [153] [215] and a survey paper

of constraint satisfaction for music composition is [154].

Machine Learning

Machine Learning (ML) [138] [1] is a sub-field under the umbrella of AI. The discipline of machine learning is to design and develop algorithms to allow computer programs to learn from training samples or its history to adapt a new environment, on the basis of predictions and reactions or both. Programmers do not need to foresee all the possible solutions for designing a general algorithm for adapting new environments. The following section reviews some of the works that used machine learning techniques for music composition. For a general survey, refer to [156].

Learning from Music Samples

Using music samples to train a machine learning model which then uses it to generate music is efficient, since this method reduces the amount of musical knowledge built into the system. Among the ML models, Markov model and Artificial Neural Network (ANN) [85] [65] are strong prediction and generative models that are frequently used in the domain of music composition.

The Markov model is a probabilistic model that is widely use for pattern capturing in Natural Language Processing (NLP), but it received a great deal of attention for music composition. Researchers Hiller and Isaacson [87], and Pinkerton [165] were possibly the pioneers of the use of the Markov model for music composition. Since then numerous Markov model-based music composition systems have been proposed in the literature. A survey paper to refer to is [2]. It gives an excellent comprehensive tutorial that demonstrates how

to employ the Markov model for music composition and a survey of relevant research in the past three decades before 1989. For the recent development of a Markov model-based music composition system, Simon et al. [186] [141] developed commercial software MySong that automatically generates a set of chords to accompany a vocal melody (end-user sings into a microphone). The technology underlying this software is a type of Markov model called Hidden Markov Model [7] [171], where chords are the hidden states and a histogram of the notes represents the observation states. One remark on the system is that it pre-processes the entire training samples into a particular musical Key (Key of C) by transposition, this makes the model avoid any odd note (outsider notes) in the transition matrix, but the Key signature must be given or identified in advance.

When using the Markov model to capture patterns from musical sequences, it learns a short scale sequence structure rather than a long scale. Realistically, the Markov model is only able to generate short meaningful musical sequences. For long musical sequence generation, sequences tend to lack musical structure and there is a great potential that these musical sequences have low probabilities and introduce greater accumulative mistakes in terms of the music.

ANN is an efficient machine learning algorithm that is capable of training itself from samples or by a human. It is a widely used technique for solving problems such as pattern recognition, robot controller and financial forecasting. ANN optimises the associations (connection weights) between inputs and outputs. The trained network produces the best guessed output according to the inputs. Among different types of neural networks, Recurrent

Neural Network (RNN) is a sequential network that feeds parts of the outputs back into the network as inputs. The network has the ability to produce the next output based on the memory of the past inputs, where each current output influences the next output. This ability makes RNN able to perform sequential prediction and become a favourable type of neural network for the music composition task. One of the pioneer works is by Todd [201], who used an RNN with back-propagation to train on a set of music samples, then used the trained model to generate new pieces. Todd's system also encountered a similar problem to the Markov model-based music composition, in that RNN is only able to generate a short scale of structured musical sequences, as it lacks global structure for longer sequence. However, Todd claimed that using an hierarchically organised and connected set of RNN can address this problem. Furthermore, Mozer's CONCERT [142] showed a little success at addressing the global structure problem. But still, Mozer concluded that CONCERT cannot realistically be expected to construct a piece of music with a musical form like AABA. For more detail of the ANN approach for music composition, refer to [202] [75] [31] [62] [71].

Learning from Humans

The idea underlying this approach is to model individual user's music preferences by observing the history of their decisions, then the trained model acts as the fitness function in the EA system for music composition.

Johanson and Poli [94] developed an interactive GP system with automatic fitness raters. This system generates short musical sequences which are then evaluated using a set of fitness raters. These raters are implemented

based on neural networks with shared weights trained by the back propagation algorithm. A user rates a list of melodies while the raters use the results to learn to rate other melodies in a similar fashion to the user.

Tokui and Iba [204] developed a drum pattern generative system called Conga. Conga was implemented with both GA and GP. GA's individuals represent short pieces of drum patterns, and GP's individuals represent the arrangement of these patterns. Both populations were evolved interactively through the user's evaluation. In the latest version of his system, he used a neural network to model the user's musical preferences and then replace the GA's fitness function with the trained ANN. The results showed that the neural network based fitness function increases the speed of the evaluation. However, Todd and Werner [203] identified a problem when using ANN as the fitness function in an EA system is that if the musical search space has steep surfaces (thus, if changing one note can make a melody change from good to bad), a neural network may fail to model this adequately.

Modelling

This technique is based on modelling the regularity patterns in the musical samples into appropriate rules or grammars and then new music is generated by these rules or grammars.

Prediction Model

Predictive models such as Zipf's Law [224] [225] and Fibonacci number [53] are well-known methods for describing distribution properties for some natural phenomena. It seems that researchers using Zipf's Law and Fibonacci

number suggest it is better for analysing musical pieces than for evolutionary music composition [170] [132] [120]. For an example of using Fibonacci number for music composition can be found in [18]. For Zipf's Law, Manaris et al. [123] proposed various Zipf's Law based fitness functions for music evaluation, each of them evaluates Zipf's Law property with different degrees, such as pitch, duration, volume and so on.

Grammar Modelling

L-system [169] has been widely used to model the structure of plants for computer graphics. It has been shown it is capable of generating complex patterns with a certain degree of fractal and self-replication. A simple type of L-system consists of a set of alphabets and a set of rewriting rules (where each rule has possibly been assigned a probability), the system is given an initial string and a number of iterations. On each iteration, rewriting rules are applied to the current string and then a new string is generated replacing the current string. For example, if the rewriting rules are F $\rightarrow$ FB and B $\rightarrow$ FFB and the initial string is FB, then after the first iteration, the string becomes (FB)(FFB), which is equivalent to string=FBFFB, the brackets indicate the rewritten part. The second iteration will become (FB)(FFB)(FB)(FB)(FFB), and so on. L-system is one of the most popular grammar models for music generation. The earliest work that used L-system to generate music was probably by Prusinkiewicz [168], where the outputted string is interpreted by a turtle interpretation method, each alphabet is interpreted according to the direction of the turtle's movement i.e. F = move forward 1 step, R=turn right 90 degrees. After a number of iterations the

output string will be represented as a 2D graph, which is then interpreted into a musical form by a music interpretation algorithm for music generation. The music interpretation algorithm is similar to placing a turtle generated graph on top of the piano-roll view. The algorithm can be described as follows: the Hilbert curve is traversed from the defined position, the horizontal line segment is mapped to a note, the pitch of this note is defined by its Y coordinate where each unit is assigned to a particular pitch, and the duration of the note is the length of the line segment, a set of duration values is assigned to the X-axis. Prusinkiewicz demonstrated how L-system generated 2D patterns can be interpreted into music by a simple music interpretation algorithm, however, the mapping algorithm is dependent upon the designers. Some recently developed music interpretation algorithms for L-system can be found in [125] [218] [139] [118].

Overall, L-system is able to produce interesting music, but using L-system to generate music without a clear objective function or constraints, it is difficult to control the system to generate music in a musically meaningful way, and hard to find interrelated phrases within the generated musical sequence. One major challenge for L-system is that designing suitable rewriting rules and music interpretation algorithms involves trial-and-error. McCormack [124] attempted to address this problem by using an interactive EA system to evolve L-system's rewriting rules for generating computer graphics. Two major issues to consider under this approach: firstly, there is a chance that some rules (rule set A) are not called if the rest of the rules (rule set B) do not contain the symbol that has occurred in the rule set A. Therefore rule set A has no effect on the system except to exist. Secondly, to inference the

relationship between rewriting rules and the generated pattern is difficult from the observation. More details of the interactive evolution of L-system is described in [126].

Finally, the following is a summary of the advantages and disadvantages of the automated music evaluator paradigm:

Advantages:

Efficiency Using a computational algorithm to evaluate music is far more efficient than a human listener in terms of speed and stability for music evaluation.

'Open-box' The algorithm is exposed to the end-user and music is evaluated objectively. The more the user listens to the output songs, the more he will recognise the evaluation algorithm.

Fine tuning If the developer created an interface for the end-user to interact with the evaluation algorithm, subsequently, the system would be capable of adapting new musical tasks and implicitly reflecting on the user's preference e.g. Morris et al.'s system [141].

Disadvantages:

Hard-wire composition Miranda [134] said "However, that these AI Systems are good at mimicking well-known musical styles. They are either hard wired to compose in a certain style or are able to learn how to imitate a style by looking at patterns in a bulk of training examples. Such systems are therefore good for imitating composers of musical

styles that are well established, such as medieval baroque or jazz. Conversely, issues such as whether computers can create new kinds of music are much harder to study, because in such cases the computer should neither be embedded with particular models at the outset nor learn from carefully selected examples...". Miranda's statement is somehow arguable; if a system is supposed to learn a particular musical style, then it will never be able to create a new musical style because the system is designed to follow particular musical style. Consequently, the question arises "Can humans create a new musical style without learning musical knowledge?", and on the other hand, "Can a non-musical person compose pleasant music accepted by the majority ?"

Trial-and-error Designing music evaluators is dependent on the designers, the only way to evaluate the performance of the music evaluator is by trial-and-error.

2.6 Toward An Indirect Approach for Evolutionary Music

In EA systems, the genotype-phenotype mapping algorithm is the critical part of the system, but highly depends on the problem and how well the designer knows the domain. The principle of the genotype-phenotype design is that the genotype encapsulation must be compact but represent the solution well so that the EA system can perform an efficient search. Given an example of inefficient representation, if a small change (e.g. 1 bit flipped)

occurs in the chromosome and its fitness dramatically changes, then the EA system finds global optimal searching difficult and gets easily trapped in local optimal. This situation is pointed out in Jones and Forrest's paper [95]. As a minimisation problem it belongs to Class 2: Difficult (-0.15<FDC<0.15), there is weak correlation between fitness and distance from global optimum i.e. when two individuals have similar distances but their fitness is very different, the problem becomes difficult for an evolutionary algorithm.

In section 2.5, we generally categorised techniques for evolutionary music into interactive and automated music evaluator paradigms. In this section we look at evolutionary music systems from the perspective of genotype-phenotype mapping. We categorised them into two approaches, direct and indirect approaches to evolutionary music.

Direct Approach to Evolutionary Music

This approach is straightforward, the genotype-phenotype mapping algorithm is normally a one-to-one mapping, where the genotype is the phenotype. A musical sequence is directly represented at the genotype level. A typical example is to represent a musical sequence as a sequence of integers stored in a chromosome in an EA system. Figure 2.6 shows the general architecture of the direct approach to evolutionary music. This approach has an advantage because each musical sequence can be directly manipulated by using genetic operators (musically meaningful operators are preferable). This allows the developer to understand the immediate effects of these operators and makes it easier for debugging and performing analyses. For a user, it makes it easier to choose the operators that reflect his/her musical taste, and

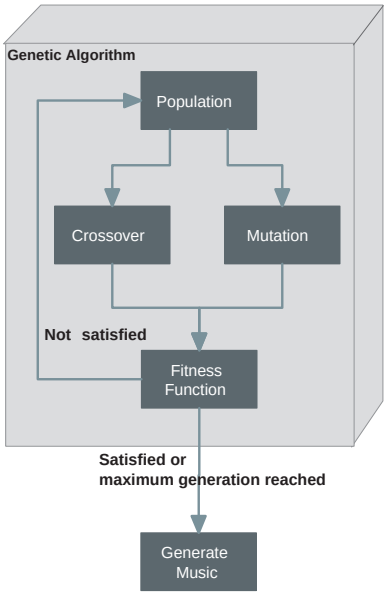


Figure 2.6: Architecture of direct approach to evolutionary music.

this leads to the evolved music being more controllable and predictable. Examples of applications that are developed using the direct approach are [17] [191] [155] [162] [148]. It seems that most of the existing knowledge-based evolutionary music systems developed under this approach are favourable.

Indirect Approach to Evolutionary Music

The indirect approach consists of two components; an EA system and a generative model. The generative model is used for generating musical sequences corresponding to the inputted model parameters. Figure 2.7 shows the general architecture of the indirect approach to evolutionary music. The EA system is responsible for evaluating the generated music and searching for the optimal parameters of the generative model. At each generation, each set of model parameters (chromosomes) is passed to the generative model, and

then the model generates music and sends it back to the EA's fitness function for music evaluation. This process is repeated until the generated music satisfies the fitness function. This approach has the advantage of modulation, and a clear separation of the processes between the EA system and generative model. The designer must know the structure of the model parameters in advance to design a suitable genotype that represents these parameters. However, choosing an effective generative model for music composition is difficult and can be vague, depending on the system requirements and problem domain. But the suggestion is to choose a model that is capable of generating a wide range of patterns from static to chaotic, where the exploration and selection of generated patterns are handled by the EA system.

2.7 Introduction of Cellular Automata

In the sixties, Von Neumann [145] introduced Cellular Automata (CA) as a model of a self-reproductive machine. One of the basic CAs was introduced by Wolfram [216] [217] called Elementary Cellular Automata (also known as one-dimensional CA). Wolfram's elementary cellular automata, an infinite one-dimensional array of cells where only two states are considered (0=dead and 1=alive). At discrete time intervals, every cell spontaneously changes its state based on its transitional rule (also referred to as CA rule interchangeable in this thesis). A transitional rule defines the cell's state at t + 1, given its state and the state of its neighbours to its left and right at time t. Figure 2.8 shows a transitional rule for a one-dimensional CA. Under the Wolfram's rule scheme, rules are often encoded from a binary string (black grid = 1 and

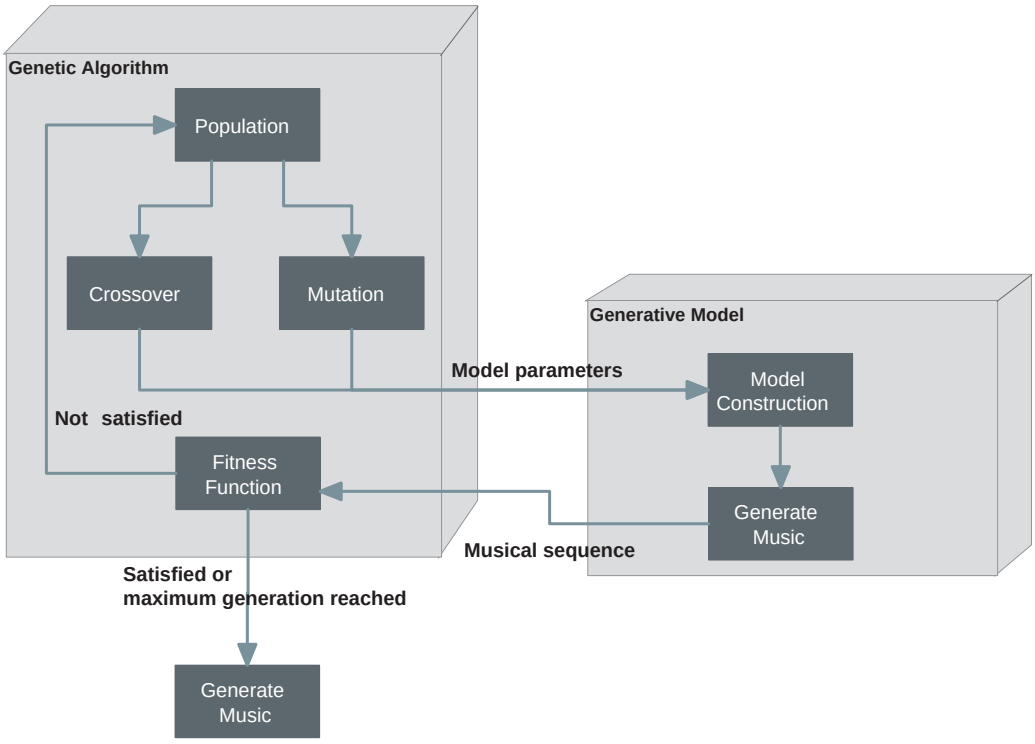


Figure 2.7: Architecture of indirect approach to evolutionary music.

white grid = 0) into a decimal number i.e. rule [0 0 0 1 1 1 1 0] = rule 30. This rule is particularly well-known because it produces repetitive, chaotic and unpredictable emergent patterns. The total number of rules for the binary state of the CA rule can be computed by 2^{2^n}, where n is the number of neighbouring cells (including target cell); i.e. for 3 and 5 neighbourhoods of a one-dimensional CA, the total number of rules is 2^{2^3}= 256 and 2^{2^5}= 4,294,967,296 respectively.

One-dimensional CA is often represented in 2D, called a space-time pattern with CA cells running across (referred to as lattice) and time step running down. Figure 2.9 shows a space-time pattern that is generated by an elementary CA using rule 30, approximately 1000 time steps. However, CA can be designed to be more than one-dimensional and there are infinite ways of designing its rule; i.e. Two-dimensional CA also known as Conway's Game of Life [72].

Overall, CA model can be described as a dynamic model that is able to generate patterns between homogeneous fixed points (class I), periodic (class II), chaotic (class III) and complex (class IV) [217]. In the class of complex patterns, patterns with self-replicating structures are found within in this class as proven by Lohn and Reggia [115] who used GA to evolve two-dimensional CA's rules that govern self-replicating structures within the defined iteration step. The type of CA they used is called Oriental-Insensitive CA, where its cell is insensitive to the orientation of the states of its neighbouring cells. Their experimental results were promising and showed that CA is able to produce self-replication structure-like patterns.

In the 90s, Langton [108] introduced a single parameter called 'Lambda'

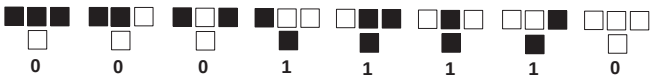


Figure 2.8: CA transitional rule (Rule 30).

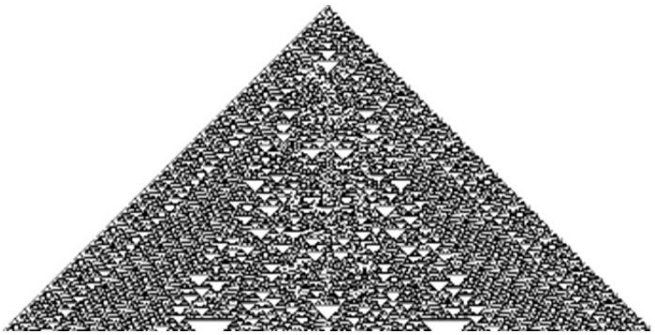


Figure 2.9: An space-time pattern that is generated by an elementary CA using rule 30.

that is used for predicting when a given CA will fall in the ordered or chaotic realms. Since the space-time pattern only represents the states' trajectory of a particular given rule, Wuensche and Lesser [219] proposed 'Basin of Attraction Field' to represent the global behaviour of CA for a synoptic and qualitative view. For the a brief history of CA, the reader is referred to [181]. For more advanced aspects of CA such as rule cluster, rule transformation (i.e. mirror image of space-time pattern) and basin of attraction field, the reader is referred to Wuensche's book [219].

2.7.1 Cellular Automata and Music

Using cellular automata to generate music is not a new idea in automatic music composition, the earliest work can be traced back to [14].

The simplest way of using CA for music composition is to use some kind

of music mapping algorithm to render/interpret the CA generated space-time patterns into music. The design of the music mapping algorithm is highly dependent on the designer. The fundamental idea is to keep the mapping process as simple as possible, but generalised enough to represent the desired range of musical elements. The arrangement of musical elements in music is handled by the CA via altering its rules and parameters. Most of the proposed music mapping algorithms in the literature use similar techniques such as mapping a single cell to an individual musical note and they then go through the space-time pattern from top to bottom to render a full piece of music. Using such mapping the user can roughly foresee the output music by seeing the generated space-time pattern.

2.7.2 Review of Cellular Automata Based Music Composition System

This section reviews some CA-based music composition systems and their music mapping algorithms.

CA-based Music Composition System

Hunt et al. [91] developed a CA workstation that generates music based on user interaction with inputs and CA parameters. Users are allowed to change these parameters during the run-time. Hence, users can have their own settings for generating their preferred music style. The type of CA is a one-dimensional CA with a space-time representation. Music is rendered by one of three music mapping algorithms. Each music event is mapped to

each horizontal lattice: 1. Each cell is mapped to a particular pitch from a set of pitches, so when there more than one cell is activated in the array, two or more notes will be played simultaneously. 2. The user defines part of the space-time pattern as a block which is then interpreted into parameters used in a music synthesiser. 3. Each cell is mapped to a particular music grid, which is an existing musical material.

Reiners [174] demonstrated how to use some simple music mapping algorithms with one-dimensional CA to generate interesting MIDI music. The mapping algorithms he used in his system are: (Note that the mapping algorithms below ignored note duration.)

Keyboard mapping This is the same as Hunt et al.'s music mapping algorithm 1 [91].

Row to binary number It is similar to keyboard mapping, instead of producing polyphonic music, each cell array is represented in a binary string and maps to a particular note. A 7-bit binary string is long enough to represent pitches from 0 to 127.

Cumulative row to binary number This mapping is similar to the 'Row to binary number' mapping, but the row value is cumulative over each generation.

Reiners suggested that simple mappings with CA are able to produce pleasing music and there is a potential musical bias when using some complex mapping algorithms that encode predefined musical knowledge.

Burraston [23] developed a CA based interactive music composition system for actual live performance, named CA Simplistic Selector (CASS) which

is an extended version of Bill Vorn's 1D CA Max external program [213]. Burraston investigated his system from the perspective of CA global dynamic. In his initial experiment, CASS consists of a one-dimensional CA (3 neighbourhood) which has 8 cells in the lattice and each cell is mapped to a particular adjustable MIDI event, named Cellular MIDI Event (CME). CME is controllable in real-time by the performer, with changeable parameters such as note, note velocity, duration, MIDI channel and so on. The CMEs onsets and tempo are controlled by the CA or performer in real-time.

Finally, for more reviews of CA-based music composition in MIDI refer to [24].

Evolutionary CA-Based Music Composition System

Nelson [143] [144] provided a brief introduction to using interactive GA to evolve dynamic systems to generate music. These dynamic systems include L-System, cellular automata and chaos models, which are able to produce one or more dimensional patterns. Then these patterns are interpreted into music. For example, the X and Y coordinates on the 2D pattern represent time and pitch respectively. During the evolution, the user is able see a visual representation of each individual pattern and listen to the rendered music, and then the user selects the parents for reproduction based on subjective evaluation.

Beyls [15] implemented an interactive evolutionary system to evolve CA for music generation. The type of CA is one-dimensional and the size of its neighborhood (3, 5, or 7) is determined by the user. Music is rendered by a music mapping algorithm that interprets the CA space-time pattern.

For the EA system, using the CA rule as the genotype, in the reproduction process, the user is responsible for selecting a pair of CA rules for the parents which apply crossover and mutation operators to create offspring. The user evaluates the CA rule either by listening to the CA generated music or by visually inspecting the space-time pattern. In his conclusion, Beyls claimed that using his proposed approach provides the ability to combine exploration and exploitation searches for a creative solution in the composer's personal search space. Beyls's CA music system is particularly relevant to our proposed system, but the major difference is that our fitness function is implemented with a machine learning technique rather than an interactive fitness function. This addresses the problem of interactive fitness functions being time consuming, as this is a particularly stressful method for evolving CA rules as the combination of the rules is huge when the CA neighborhood size is greater than 3.

2.8 N-gram Model

In the field of Natural Language Processing (NLP), the N-gram model is a widely used statistical language modelling technique. Also it is a machine learning technique because it involves a training process. Statistical methods tend to be more robust in learning from data samples than purely symbolic techniques. Using a statistical approach has the advantage of minimising embedding domain specific knowledge into the system and is flexible with respect to the learned grammar that represents the training samples (corpus in statistical terminology). The learned grammar can be used for generating

similar data or to evaluate the probability of given data.

N-gram is a type of Markov Model (MM). The Markov assumption is that the probability of a symbol depends only on the probability of the limited history. In other words, the N-gram model is based on the assumption that the next symbol in a string is largely dependent on a small number of immediately preceding symbols in the string. N is a model parameter which represents the total number of preceding symbols plus the current symbol. N could be any number ≥ 1, e.g. N=1 is called a Uni-gram which looks zero (N-1) symbols into the past. N=2 is called a Bi-gram which looks one symbol into the past. N=3 is called Tri-gram (3-gram) which looks two symbols into the past, and so on. The maximum combination of symbols is $\mid A \mid^N$ in the model, where A is the size of the alphabet. While larger N results capture more contexts, it creates exponential growth the size of the maximum number of combinations. Generally, working with N-grams, the user chooses N preceding symbols on which the next symbol is assumed to depend. The recommendations of N for capturing natural languages are 2 or 3. That because when N is too high, the captured sequence is long and too exact so the model is not generalised enough. The captured long sequence might not appear in the testing sequences and for a sequence generation, exact long sequences will appear often. Hence, it is well-known that N-gram is not capable of capturing long-distance dependencies.

Equation 2.2 is the equation for a Bi-gram model (N=2); this generalises in straightforward ways different values of N. Equation 2.3 is the equation for estimating the general case of N-gram probability.

w_n: Individual symbol, where n is its order.

w_{n-1}: Previous symbol to w_n.

$w_1, w_2, w_3 \ldots, w_n$ Or w_1^n: Representing the complete string (sequence) of symbols.

$C(w_n)$: is the frequency number for symbol w_n.

N: is the N-grams model's parameter, number of $N-1$ preceding symbols.

$P(w_1^n)$: The probability of a complete string.

$$P(w_1^n) = P(w_1) \prod_{k=2}^{n} P(w_k | w_{k-1}) \tag{2.2}$$

$$P(w_n | w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1} \cdot w_n)}{C(w_{n-N+1}^{n-1})} \tag{2.3}$$

The implementation of the N-gram model is straightforward as the representation of the model is a transitional matrix. Each transitional probability is calculated by dividing the observed frequency of a particular sequence by the observed frequency of a prefix, as shown in Equation 2.3. Additionally, the programmer is able to edit the matrix directly, unlike in other techniques such as ANN which is difficult for programmers to manually adjust the weights of node associations. Figure 2.10 shows an example of using a Bi-gram model for a string prediction task. To make it simple, in the example, the alphabet only consists of symbols a, b, c and d, the training corpus is a string = "a b c a b c a c d a d d" and the test strings are "a d a b" and "a d a c". The prediction task is to find out which given string is the most likely to be generated by the Bi-gram model. The results showed that string = "a d a b" is more likely to be generated than string = "a d a c", because the

Alphabet: a, b, c, d
Corpus string: "a b c a b c a c d a d d"

Bi-gram: Transition matrix

Next symbol

Previous symbol	a	b	c	d
a	0.0	0.5	0.25	0.25
b	0.0	0.0	1.0	0.0
c	0.666	0.0	0.0	0.333
d	0.5	0.0	0.0	0.5

Use Equation 2.2 to calculate the probabilities of String="a d a b" and String="a d a c". The highest probability of a particular string occurring is the one the system is most likely to generate. Logarithm is applied in the calculation to avoid small number and zero probability.

Given that the probability of generating that the first symbol is either a, b, c or d is 0.25.

P("a d a b") = P(a) * P(d|a) * P(a|d) * P(b|a)
Log P("a d a b") = Log 0.25 + Log 0.25 + Log 0.5 + Log 0.5
Log P("a d a b") =-1.8061
P("a d a b") = Antilog -1.8061 or 0.015625

P("a d a c") = P(a) * P(d|a) * P(a|d) * P(c|a)
Log P("a d a c") = Log 0.25 + Log 0.25 + Log 0.5 + Log 0.25
Log P("a d a c") =-2.1072
P("a d a c") = Antilog -2.1072 or 0.0078125

Thus, String = "a d a b" is more likely to be generated by the system than String = "a d a c".

Figure 2.10: String prediction using Bi-gram model.

probability $0.015625 > 0.0078125$. To avoid the probability being too small, a logarithm should be applied in the Equation 2.2 as 0.015625 = Antilog -1.8061.

N-gram is often used in three aspects which are sequence generation; sequence evaluation; and sequence classification. For sequence generation, a sequence of symbols is generated randomly according to the transition matrix. For sequence evaluation, a single N-gram model can be used for

calculating the probability of the given string, see Figure 2.10 for an example. For music classification, the N-gram classifier is given a name when multiple N-gram models are used for a classification task. Each class is associated with a trained N-gram model, and the given sequence is evaluated by each model. The test sequence belongs to the class that its N-gram model returned, with the highest probability among other N-gram models.

Although the N-gram model is simple in nature, it works well in practice, providing sufficiently large samples of training data are available. N-grams have been mostly used for natural language problems in text prediction, text retrieval (i.e. indexing[2]) in different languages [110] [146] [128] [100], language grammar checking, speech-understanding [97] [96] and pattern recognition [93]. N-gram also has been widely used for music indexing in music information retrieval systems [180] [221] [222]. Doraisamy and Ruger [56] [54] studied the performance of using N-gram for music indexing in a large music database containing polyphonic MIDI music. The issues regarding fault-tolerance (i.e. music query with probability of error level) of the N-gram approach for a music retrieval system is studied in [55] [58] [59]. For audio music indexing, Patel and Mundur [159] used the N-gram approach to determine repeating patterns.

The N-gram model is receptive to the training data, since any symbol in the corpus is counted. The corpus must be carefully selected to ensure that it relates to the problem domain. If the training corpus is overly narrow then the probabilities are not generalised enough. On the other hand if the corpus

[2]A simple form of N-gram indexing for querying a text database is done by searching a string in the collection that contains the most common distinct N-gram windows that also occurred in the query string.

is overly general, then the probabilities do not reflect the samples. In practice any particular training corpus is finite. There will be some events that have not occurred in the corpus. Naive estimation techniques based solely on a frequency count would give these zero probability. Any test sequence that contains an event that has not occurred would be given zero probability. The simplest solution to avoid this problem is to adjust the transition matrix, changing the zero probabilities to a small floating number such as 0.000001 and then normalise the matrix to ensure the sum of probabilities is equal to 1. Another alternative technique is the 'Add-one' discounting method where all events are given an initial occurrence count of 1. These types of method are called 'smoothing', which avoids the large impact caused by unseen data and performs well over sparse data. For other smoothing methods refer to [96].

Maximum Likelihood Sequence

One feature of Markov models is that it is possible to compute the Maximum Likelihood Sequence (MLS), that is, having chosen the length of the sequence, a dynamic programming algorithm can be used to compute the sequence of symbols that is most likely to be generated by the given model. When using a trained N-gram model as a fitness function in an evolutionary algorithm, the optimal sequence can be compare to the MLS that is computed in advance, to see how close the EA gets to the true optimal sequence. As we shall see in chapter 4, the best sounding sequences tend to have middle range fitness, while low fitness sequences sound randomised and MLS tends to be very repetitive and sound dull.

N-gram Model for Music Composition

Music and language share some common features, both consist of a set of symbols and grammars. For natural language such as English, letters (e.g. a to z) are the symbols. For music, musical pitches (note or absolute pitch) are the symbols. An individual style of music can be seen as an individual musical language with a particular grammar. If a statistical approach works well in natural language, then it should work well in music with appropriate adjustments.

The earliest investigation of using an N-gram model to generate musical sequences can be traced back to mid-1950s, when Pinkerton [165] analysed 39 nursery tunes using Uni-gram and Bi-gram models. He manually constructed a simple binary decision network that was derived from the transition matrixes to generate melodies. He claimed that one of the generated melodies reminded him of a nursery tune called 'Ride a Cock Horse to Banbury Cross'.

Brooks et al. [21] used N-gram (up to 8-gram) to generate music sequences in the hymn genre. The N-gram model is trained on 37 common hymn tunes. Musical sequences are generated based on the N-gram transition matrix and some constraints are applied. If a generated sequence does not satisfy the constraints then it is discarded. They commented that in their system Uni-gram and Bi-gram would have yielded sequences unacceptable as hymns and 8-gram would have yielded sample members.

Ponsford et al. [166] used 3-gram and 4-gram models to capture harmonic movement (similar to chord transition) from a training corpus of seventeenth-century dance music, and then generate music based on a sequential predic-

tion of the next note from given previous notes, taking probabilities from the transition matrix of the trained N-gram model. They attempted to generate a structural musical piece by using templates with a predefined arrangement of a musical phrase. The generated harmonies are constrained to follow these templates. The finally generated pieces were subjectively judged by the system author. They claim that their approach is promising.

Although the literature shows the success of using an N-gram model for music composition, there are some issues we have to consider when using an N-gram model to capture the pattern of musical sequences. For learning sequence-structure, N-gram learns a short scale rather than a long scale. For sequence generation, generated sequences are not guaranteed to have high probabilities that are obtained from the given N-gram model. Therefore, the N-gram model is often used for short music generation. It could be argued that because of these characteristics, the N-gram model is able to generate various musical sequences which have reasonably high probabilities. The other issue is that since a naive N-gram composition system is not guaranteed to generate high probabilities of sequences, longer sequences tend to have lower probabilities than shorter sequences, because the probability calculation is based on multiplication. In fact, it is not true that longer musical sequences receive lower artistic values than shorter sequences. However, the MLS is not guaranteed to be musically pleasant, but statistically it is the optimised sequence that has the highest probability that is obtained from the given N-gram model.

2.9 Zipf's Law

Zipf's Law [224] [225] is probably one of the most well-known laws that describes a frequency property for some natural phenomena. These include earthquake magnitudes [190], city sizes [189], word frequencies [185] [82] [4] and so on. For instance, in natural language processing of English words, Zipf's Law states that given some corpus, the frequency of any word is inversely proportional to its rank in the frequency table. A simple form of Zipf's Law for estimating the frequency of a given word is $f(n) \backsim \frac{1}{r(n)}k$, where $f(n)$ is the frequency of word n, $r(n)$ is the ranking of word n and k is a constant number representing the frequency of the word with the highest ranking. If the data set has a perfect Zipf's Law property and applies a logarithm to both the rank and frequency in a scatter graph (referred to as log-log plot in this thesis), the slope of the regression line will be exactly -1. Hence, log-log slope and its R-squared values are metrics for measuring how fit the data is following Zipf's Law. R-squared is the CorrelationCoefficient^2 that indicates how well a regression line approximates real data points; the value range is from 0.0 (0%) to 1.0 (100%).

Zipf's Law and Music

Using Zipf's Law to analyse and evaluate music objectively in the context of statistics, allows us to understand the construction of the tested music from a higher perspective. An example of using Zipf's Law to analyse a preprocessed music sample is now given. Table 2.1 shows a summary of pitch statistics and Figure 2.11 shows the log-log plot after it applied a Logarithm

Rank	Absolute pitch	Frequency
1	74	48
2	Rest note	45
3	78	28
4	76	27
5	81	23
6	79	22
7	73	20
8	69	19
9	71	12
10	62	10
11	67	10
12	80	6
13	83	6
14	66	5
15	64	3
16	72	3

Table 2.1: Pitch frequency table for K133 1772 Salzburg Symphony No. 20 in D Mvt3.

(base 10) on both ranks and frequencies. The slope of its regression line is -1.017 and the R-squared value is 0.819. This result shows that the music has a strong Zipf's Law property with over 80% data fits in the regression line.

One issue that we encountered when sorting the rank order in the pitch frequency table is that, for example, rank 10 and 11 frequencies tied, so a decision was made not to allow more than one entry at the same rank. To our knowledge, there are no universal guidelines for this situation. However, the Zipf's Law model does not predict frequencies to tie with one another.

Here is provided a belief survey of experimental works that aim to verify music with some degree of Zipf's Law properties:

Manaris et al. [123] [121] used more than twenty Zipf's Law based met-

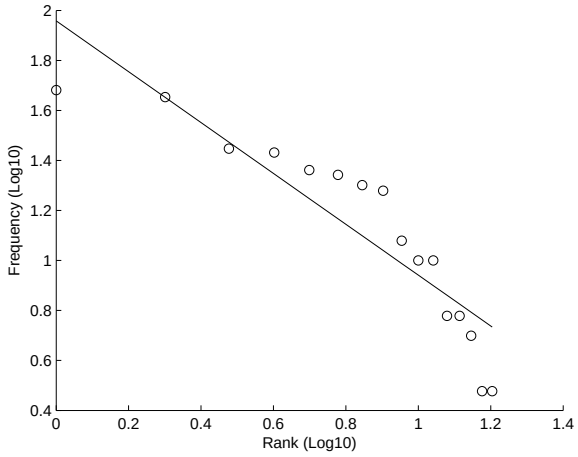


Figure 2.11: log-log plot: K133 1772 Salzburg Symphony No. 20 in D Mvt3. Slope = -1.017 and R-squared = 0.819.

rics to analyse various genres of music (Baroque, Classical, Rock, Jazz etc) and DNA sequences, providing a total of up to 196 samples. These metrics were devised from music theory, which includes pitch, duration, pitch and duration, harmonic interval, and so on. The experimental results were promising, the average Zipf's Law log-log slope was 1.2023 with an average value of 0.8233 for R-squared. These results indicated that there are strong Zipf's Law properties within the samples. Later, they used these metrics to train a neural network to classify musical pieces into different composer styles (Bach vs. Beethoven, Chopin vs. Debussy etc.). In the experiments, the success rates across five composers are in range of 93.6 to 95 percent. However, he suggested that "We may have discovered certain necessary but not sufficient conditions for aesthetically pleasing music". Furthermore Manaris et al. [122] extended their earlier works in the following ways: first, they evaluated their Zipf's Law based metric by conducting a list of music

classification tasks i.e. between popular and unpopular music with up to 87.85% accuracy. The classifier is implemented on the basis of training a neural network using a list of extracted features from music samples in order to make a note prediction. Second, they used a trained ANN and dynamic ANN ('bootstrapped' by feeding the last population into a random corpus) as music evaluators embedded in a genetic programming system for music composition. The whole system is called NEvMuse (Neuro Evolutionary Music environment). Finally, they put the aesthetic judgments of both evolved and human-composed music to the test by asking twenty-three human subjects to respond to the music by indicating how each piece made them feel with a choice of emotional label (i.e. happy, sad, calm, lethargic etc...) They found that their music evaluator was able to strongly predict both the pleasantness and activation ratings of human listeners.

Zanette [223] analysed four musical piano pieces in the context of musical pitch. He demonstrated that there were strong Zipf's Law properties in the samples in terms of note usage. He suggested that for natural languages, words are the 'units of context', they are the perceptual elements whose collective function yields coherence and comprehensibility to a message. For music, musical notes are the 'units of context'. Notes are the fundamental building block of a musical phrase.

Yip and Kao [221] studied four musical features with an N-gram model for a content-based music retrieval system. These features are: interval sequence, profile (the same as Contour representation), coarse interval sequence and note duration ratio sequence. They found that for Bi-gram and Tri-gram models, the log-log slope curve is not straight, instead it is an upward convex.

This indicates that the occurrence probability falls more slowly when rank r is high, and falls faster when r is low. They suggested that the N parameter in the N-gram model should be $N \geq 2$ for interval sequence, $N \geq 9$ for profile, $N \geq 3$ for coarse interval sequence and $N = 1$ for note duration ratio sequence.

Voss and Clarke [214] studied a wide range of music, such as classical, jazz, blues and rock. They measured several fluctuating physical variables, including output voltage of an audio amplifier, loudness fluctuations of music, and pitch fluctuations of music. They discovered that pitch and loudness fluctuation in the music sample followed Zipf's Law.

2.10 Information Entropy

In brief, Information Entropy is a measure of the average information transmitted over communication. Information Entropy is also known as Shannon Entropy [183] [184]. The best way of illustrating entropy is to refer to the horse betting example described in [39] and [96]. For natural language, Shannon [184] described how entropy can be used to measures how much average information is transmitted (encoded in binary i.e. 0001 = 'a') for each letter of a text message in the English language. Equation 2.4 is the definition of information entropy with a Log base = 2. Equation 2.5 is the definition of Maximum Entropy.

$$Entropy(x) = -\sum_{i=1}^{n} p(x_i) log_2 P(x_i) \tag{2.4}$$

Where X is a discrete random variable $\{x_1, x_2 \dots x_n\}$. $p(x_i)$ is the probability mass function of outcome x_i, and n is the number of possible events.

$$Maximum\ Entropy(x) = -log_2 \frac{1}{n} \tag{2.5}$$

In the domain of music, entropy can be used to measure how much average information is transmitted for each note of a music piece. For melodic analysis, low entropy indicates that fewer pitches are occurring more often than the rest in the melody and high entropy indicates that the frequency of pitches that are occurring in the melody is similar. Maximum entropy indicates that the frequency of each pitch is equal (all pitches have an equal probability of occurring). For example, if a melody has an entropy of 0.4 and the maximum entropy value is 1.2, the entropy indicates that the melody consists of regular phrases and has 66.6% redundant pitches. On the other hand, randomly generated music tends to have a high entropy. Pinkerton [165] suggested that "Thus the composer of a melody must make the entropy of his music low enough to give it an apparent pattern and at the same time high enough so that it has sufficient complexity to be interesting."

Chapter 3

System Framework, Implementation and Music Evaluators

This chapter is divided into three parts to describe the methodology of how we built various evolutionary music composition systems. These systems will be tested and evaluated using a number of experiments described in chapters 4, 5 and 6.

Part 1 describes an extended version of Pearce and Wiggins's music composition system framework [160] that is described in section 2.4. The extended framework aims to fulfil our needs in the domain of evolutionary music. Part 2 describes the implementation of the proposed evolutionary music composition systems which embody the extended framework and the details of its components. Finally, part 3 describes four music evaluators (fitness functions) that were used in the proposed systems, explaining in detail

their aims, algorithms and expectations.

3.1 Framework for Evolutionary Music Composition Systems

For our research, we extended the Pearce and Wiggins's framework to make it more suitable for the proposed evolutionary music composition systems. Four components are added into the framework; music information extraction; music representation; evolutionary computation; and genotype-phenotype mapping. Figure 3.1 shows the extended version of the framework. The following describes each component in detail:

Music Information Extraction In section 2.2.1 where we discussed the detail of music information extraction, we emphasised that music is complex as it might contain multiple instruments or non-tonal instruments (Some DJ music contains sounds recorded from nature). Raw musical pieces contain too much information and this form of music might not be suitable for the composition aim and the machine learning techniques. For example, using monophonic music represented in a sequence of integers to train the Markov model is more straightforward than polyphonic music. Doraisamy [54] used an exhaustive approach to construct N-gram windows from a piece of polyphonic music by obtaining all possible paths of monophonic melodic string from the given polyphonic music. This approach is computationally expensive for the higher N order of N-gram. For example a short piece of music consisting

of ten instruments, each onset contains 10 notes, for 5-gram window extraction, the total different monophonic paths is 10^5=100,000. He suggested restricting some of the possible paths to reduce the size of the combinations, such as by only allowing the event of the top two and bottom two notes in a piece of polyphonic music. In the stage of music information extraction, music samples will be pre-processed for the preparation of training the critic. The extraction process aims to extract the minimum required information out of music samples. The method or algorithm of this process must be carefully chosen depending upon the composition aim and the machine learning techniques.

Music Representation The music representation must be specific at this stage, such as the music representation for the training samples and output music. There are three common categories of music representation: audio, symbolic music information and musical notation. For the symbolic music representation, there is range of representations in this category, such as absolute pitch, pitch difference (interval), contour, scale degree and so on, see section 2.2.2. The decision of music representation made at this stage will influence the later stage of designing the genotype representation and the genotype-phenotype mapping in the evolutionary computational system.

Evolutionary Music Composition Using any evolutionary computational technique to evolve music, the music must satisfy the critic (fitness function). In this stage, the designer must specify the evolutionary algorithm, representation, genetic operators, fitness function and genotype-

phenotype mapping. How these components are specified is largely dependent on the composition aim, but is also influenced by the music representation selected. For example, music can be represented as a tree structure directly evolved in the GP system [45].

Genotype-Phenotype Mapping This component is a critical part within any evolutionary system, efficient genotype-phenotype mapping leads the EA system to search effectively in the search space. The designer must clearly define the genotype-phenotype mapping process before any computation, and specify how to represent music in the genotype which operates in the EA system. In the domain of music composition, in general, the mapping process can be categorised into direct and indirect approaches for evolutionary music, the details of these approaches are described in section 2.6.

There are three more attractive features in the extended framework. First, is the ability to specify an algorithm for pre-processing the music samples at the stage of music information extraction, truncating the unnecessary information and extracting specific information from raw musical piece. Second, the music representation of the phenotype is specified during the stage of music representation. Third, in the genotype-phenotype mapping component, it provides two approaches to choose from for designing the genotype-phenotype mapping process.

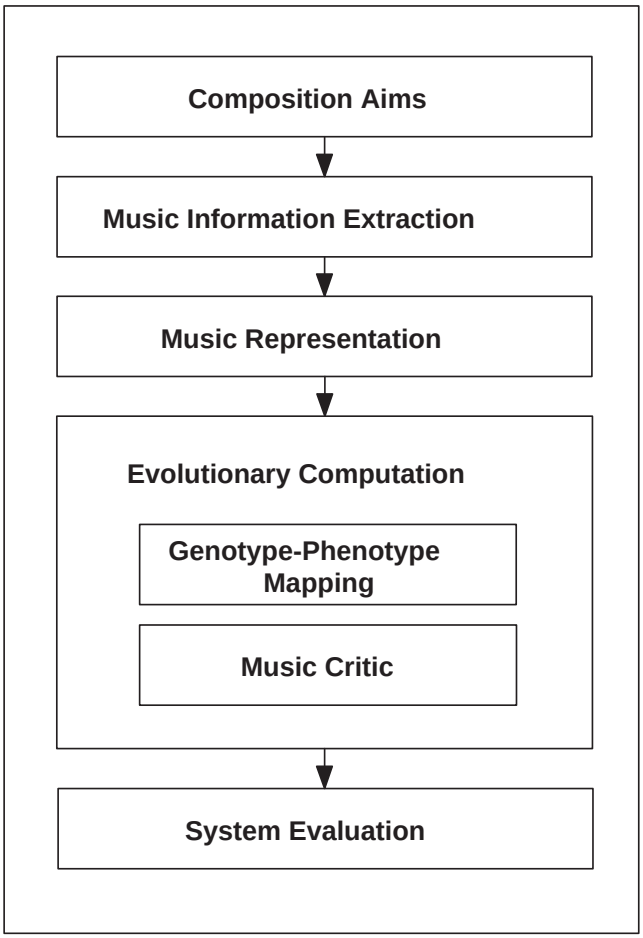


Figure 3.1: Framework for evolutionary music composition system.

3.2 Implementation

This section describes the implementation details of the proposed evolutionary music composition systems, which are embodied in the extended framework for evolutionary music system described in the previous section.

3.2.1 Composition Aim

There are three aims that we expect our proposed systems to fulfil. They are:

- The approach for developing the music composition systems is evolutionary computation. The systems do not require a human to evaluate all potential music. Instead the user selects a set of music samples for training a machine learning model that is then used as a fitness function in the EA system. The training samples are in monophonic, since polyphonic music involves multiple dimensional features it is much harder for pattern capturing.

- The system is able to generate pleasing music[1]. At best, would be for the generated music to be indistinguishable from human-composed music and receive a higher rating than the human-composed music.

- The system is capable of generating both monophonic and polyphonic music.

[1]The term "pleasing music" is defined in section 1.3.3

3.2.2 Music Information Extraction

The proposed trainable fitness functions are implemented based on the N-gram model, which is an efficient model for capturing natural language in a monophonic manner. To maximising the capability of the N-gram model, we used a set of monophonic music (referred as musical sequence) that is represented in a symbolic form as sequence of symbols to train the N-gram model. All our music samples are raw musical pieces in the style of classical music, composed by Mozart, Beethoven and Chopin. Samples are preprocessed using an 'All-Mono' extraction algorithm, which has the advantage of extracting perceivable monophonic melody over polyphonic music.

3.2.3 Music Representations

Musical sequences are represented as sequences of integers in the systems. Two representations are used:

Absolute Pitch (AP) Each integer value represents a single absolute pitch, except for 128 which represents the 'Rest' note to indicate silence/rest in the music. The integer value of pitch follows the standard MIDI protocol. The range of integer values is from 0 to 127, which is enough for representing a standard grand piano which has a total of 88 keys.

Pitch Difference (PD) Each integer value represents an interval between two successive pitches (an interval can be + or -). Each pitch difference from the previous pitch to the next pitch is equal to next pitch minus previous pitch. In musical terms, a +1 interval is equal to one raised

semi-tone, and a -1 interval to one dropped semi-tone. A special symbol +128 is used to represents a Rest note.

Figure 3.2 is an example of music shown in integer string formats from both encoding methods.

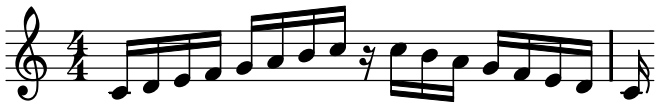


Figure 3.2: C Major Scale.
(Absolute pitch) Integer string = [60, 62, 64, 65, 67, 69, 71, 72, 128, 72, 71, 69, 67, 65, 64, 62, 60].
(Pitch difference) Integer string = [+2, +2 , +1, +2, +2, +2, +1, +128, +0, -1, -2, -2, -2, -1, -2, -2].

For music samples, musical sequences are represented in absolute pitch or pitch difference; note duration is ignored and not taken into account by the music evaluators. For the system-evolved music, the duration of each note is generated by either of the following methods:

Fixed Note Duration (FND) All the pitches are played at the same note duration. The same pitch being played twice is perceived as being played double duration. The Rest note can be used to create notes of different perceived duration. This method is simple and each note is played equally.

Probabilistic Note Duration (PND) Each note duration is randomly generated with probabilities that are assigned to duration values. The probabilities of duration values are predefined by the user. This method generates various durations for the sequence and gives the sound of the sequence more texture.

Grouping Notes Duration (GND) In this method, first, notes are separated into a number of groups. In each group, notes are assigned to the same value of note duration that is randomly generated by the PND method. GND method's aim is to create a group effect in the melody and give the sound of the sequence more texture. This method is inspired by one of the laws in the 'principles of perceptual organisation' from Gestalt psychology [103]. According to the 'Law of Similarity', objects that are similar in some way seem to be grouped together (for both visual and audio objects). A similar idea is found in Bregman's Auditory Scene Analysis [20]. Bregman's experimental results showed that the law of similarity influences our sound perception in stream melody. The theory explains that when a sequence of pitches is played in a fast tempo with close intervals between pitches, listeners are likely to hear this sequence of pitches as consecutive pitches.

Figure 3.3 shows an example to demonstrate the grouping procedure used in the GND method. The user is required choose a musical scale and set a threshold which is used for determining whether the next notes belong to the current group or next group. The threshold is an unsigned integer that represents a scale degree interval. In the example, the threshold is set to 2. The top row shows the chosen C Major scale, the bottom is the grouped sequence determined after the grouping procedure. The arc symbol between notes indicates the interval of scale degree between them. From the beginning of the grouping procedure, the first note is the reference note. In the sequence, the first three notes are in the same group, because the intervals of notes E

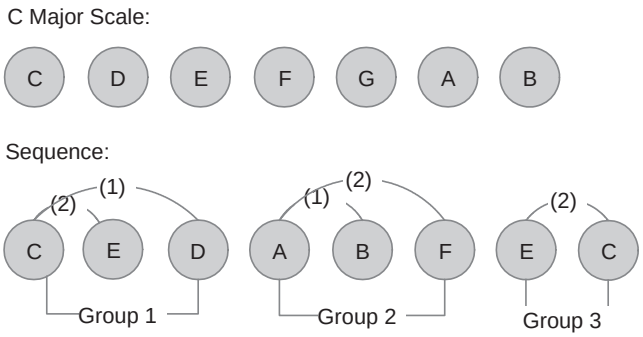


Figure 3.3: Grouping note.

and D away from the reference note C are equal to and less than the threshold. The 4th note A is 5 intervals away from the reference note C; it is more than the threshold, so it becomes the reference note of the second group. Then the intervals of note B and F away from the second reference note A are less than and equal to threshold, so these notes are assigned to group 2. The same procedure is then applied to the rest of the notes in group 3.

3.2.4 Evolutionary Computation

Several evolutionary algorithms are used in the experiments: Random Mutation Hill Climber (RMHC), genetic algorithm, NSGA-II and SPEA2. The choice of algorithms and their parameters, genotype representation, genotype-phenotype mapping and genetic operators will depend on the purposes of the experiments. The full details of their implementation are described in the experiments chapters 4, 5 and 6.

3.2.5 Fitness Functions

Four fitness functions are designed based on the N-gram model, Zipf's Law and Information Entropy for objectively evaluating musical sequences in the context of pitch. There is no domain specific knowledge embedded into these fitness functions. To better organize this thesis, the long descriptions of these fitness functions are to be found in section 3.3: Music Evaluators.

3.2.6 Genetic Operators

The main purposes of genetic operators are to serve as a tool for exploring the search space, locating the solution points and speeding up the search process. If the nature of the problem is in the domain of music composition, some human composition techniques can be embedded into the genetic operators. Various genetic operators are used in the experiments, including standard genetic operators (one-point crossover and flip bit mutation) and various musically meaningful operators (also called Sequence-Oriented Variation Operators). The sequence-oriented variation operators are derived from several melody composition techniques that are commonly used by human music composers:

Transposition Changing a note or collection of notes, by increments or decrements in pitch by a constant interval.

Retrograde Collection of notes is rearranged in reverse order.

Swapping Selecting two collections of notes and then swapping their positions.

Sorting Rearranging the melodic contour with an ascending or descending melodic line.

Listing 3.1 shows the detail of each sequence-oriented variation operator and describes how it works by providing an example. Table 3.1 is used to illustrate each sequence oriented variation operator applied on the segment that is taken from the famous piece 'Canon in D' composed by Johann Pachelbel. The segment of 'Canon in D' is shown in Figure 3.4 using standard music notation. In Table 3.1, the first row shows the segment in absolute pitch representation, ( ) indicates the changed pitch and $< >$ indicates the selected segment.

Listing 3.1: Sequence-oriented variation operators.

```
Note: the segment ranges from 2 - 10 notes, based on our intuition
of what would be reasonable.

0) Random single note modification: randomly select a location in
a sequence and randomly change the pitch value. During each
operation a small change is applied in the sequence to ensure the
operator searches solutions narrowly. Table 3.1 - row 2 shows the
operator 0 selected the 2nd location in the sequence, and then
replaced a random generated pitch value 76.

1) Random segment note modification: this is similar to operator
0, but it modifies each pitch value in the selected segment. This
operator modifies a group of pitches per operation to ensure that
the search process is fast. Table 3.1 - row 3 shows the operator 1
randomly selected a segment which was located from the 4th to 7th
```

positions, then randomly generated a new segment to replace the selected segment.

2) Random segment copy and paste: randomly selects two segments, copies the first segment and replaces it into the second segment. This operator allows segments with high fitness to be replicated. Table 3.1 – row 4 shows the operator 2 randomly selected two segments: segment A: <a to a> located from the 1st to 6th positions and segment B: <b to b> located from the 4th to 9th. Segment A is copied and replaced into segment B.

3) Random segment reversion: randomly selects a segment, then pitches are sorted into a reverse ordering. This operator simulates the retrograde composition technique. Table 3.1 – row 5 shows the operator 3 randomly selected a segment which is located from the 5th to 9th positions and then reversed the pitch ordering in the segment.

4) Random segment transpose: randomly selects a segment, and then transposes each pitch value by a constant interval which is randomly generated, the interval range from −5 to +5. This operator simulates the transposition composition technique whereby it tries to increase the overall fitness of the melody by raising or lowering each pitch in the selected segment. Table 3.1 – row 6 shows the operator 4 randomly selected a segment which was located from 10th to 14th positions, and then raised each pitch by 2 semi−tones (add 2).

5) Random segment swap: randomly selects two segments and then swaps them. This technique is commonly used in music composition.

This operator does allow segments to be overlapped. Table 3.1 – row 7 shows the operator 5 randomly selected two segments: segment A: <a to a> located from 1st to 3rd positions and segment B: <b to b> located from 12th to 14th, then the two segments were swapped.

6) Random segment ascending: randomly selects a segment and then the pitches in the segment are sorted into an ascending order. This is a common composition technique used for creating notes in ascending order. Table 3.1 – row 8 shows the operator 6 randomly selected a segment which was located from 7th to 11th positions and then sorted the pitch contour (pitch value) in an ascending order starting from pitch 69 to 81.

7) Random segment descending: randomly selects a segment and then the pitches in the segment are sorted into a descending order. The effect is similar to operator 6 but it is in a descending order. Table 3.1 – row 9 shows the operator 7 randomly selected a segment which was located from 7th to 11th positions, and then sorted the note contour in a descending order starting from pitch 81 to 69.

8) Random segment copy from training samples: randomly selects a segment from training samples and then copies and replaces it into a random selected segment. This operator guarantees that the high fitness segments are copied into the evolving sequence. Table 3.1 – row 10 shows the operator 8 randomly selected a segment which was located from 1st to 6th positions and replaced it with a same length of music segment that was randomly taken from the training sample that is shown in Figure 3.5.

9) Observed distribution single note modification: randomly

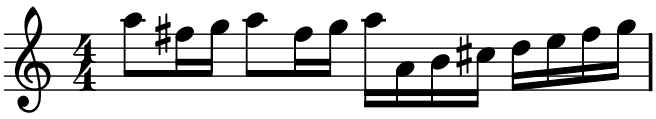


Figure 3.4: A segment taken from Canon in D.

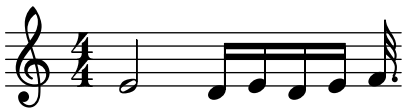
selects a location of sequence and modifies the pitch by sampling from observed note distribution in the training data. The pitch is determined by a roulette−wheel selection method. The frequently occurring high pitches have a greater chance of being selected. This operator ensures that the chosen pitch is various but there is a high chance of selecting the high fitness pitch. Table 3.1 − row 11 shows the operator 9 selected the 12th location in the sequence and replaced by pitch 64. The training data is the segment shown in Figure 3.5, the probability of pitch 64 being select is $3/6 = 0.5$, which is the highest probability over the others pitches.

10) Random single point swap: randomly selects two pitches, and then swaps their locations. This operator ensures the sequence is changing slowly to enable the greater exploration. Table 3.1 − row 12 shows the operator 10 randomly selects two pitches, 1st and 3rd positions in the sequence, then the two pitches are swapped.

11) Random segment shuffle: randomly selects a segment and then shuffles the pitch ordering. Table 3.1 − row 13 shows the operator 11 randomly selected a segment which was located from 6th to 10th positions, and then randomly shuffled its pitches.

Original	81	78	79	81	78	79	81	69	71	73	74	76	78	79
Operator 0	81	(76)	79	81	78	79	81	69	71	73	74	76	78	79
Operator 1	81	78	79	<(80)	(76)	(80)	(78)>	69	71	73	74	76	78	79
Operator 2	<a81	78	79	<b(81)	(78)	(79)a>	(81)	(78)	(79)b>	73	74	76	78	79
Operator 3	81	78	79	81	<(71)	(69)	(81)	(79)	(78)>	73	74	76	78	79
Operator 4	81	78	79	81	78	79	81	6	71	(75)	(76)	(78)	(80)	(81)
Operator 5	<a(76)	(78)	(79)a>	81	78	79	81	69	71	73	74	<b(81)	(78)	(79)b>
Operator 6	81	78	79	81	78	79	<(69)	(71)	(73)	(74)	(81)>	76	78	79
Operator 7	81	78	79	81	78	79	<(81)	(74)	(73)	(71)	(69)>	76	78	79
Operator 8	<(64)	(62)	(64)	(62)	(64)	(65)>	81	69	71	73	74	76	78	79
Operator 9	81	78	79	81	78	79	81	69	71	73	74	(64)	78	79
Operator 10	(79)	78	(81)	81	78	79	81	69	71	73	74	76	78	79
Operator 11	81	78	79	81	78	<(73)	(79)	(81)	(69)	(71)>	74	76	78	79

Table 3.1: Example of sequence-oriented variation operator.



(Absolute pitch) Integer string = [64, 62, 64, 62, 64, 65]

Figure 3.5: A segment taken from Beethoven Sonata No. 3 in C Major, op.2, no. 3.

3.2.7 Genotype-phenotype Mapping

In section 2.6 we have described direct and indirect approaches for designing genotype-phenotype mapping for evolutionary music systems. In the experiments, two types of evolutionary music composition systems are implemented, these are: 1) Musical Sequence Evolver (direct approach). 2) Cellular Automata Music Evolver (indirect approach). The following describes the general architectures of these systems, the specific details and settings of individual system are described in chapters 4 and 5.

Musical Sequence Evolver (Direct Approach)

Musical Sequence Evolver uses an evolutionary algorithm to evolve melodies which are represented as sequences of integers for evolution. The genotype is same as the phenotype. The evolved integer strings are then converted to a standard MIDI format for playback. Figure 3.6 shows the system architecture of the Musical Sequences Evolver. To operate the system, the user provides a set of music samples to train the N-gram classifier which becomes the fitness function used in the Random Mutation Hill Climber, which is similar to a (1+1) Evolutionary Strategy (ES). The trained N-gram model has discriminative power to assess melodies and give fitness to each melody. The

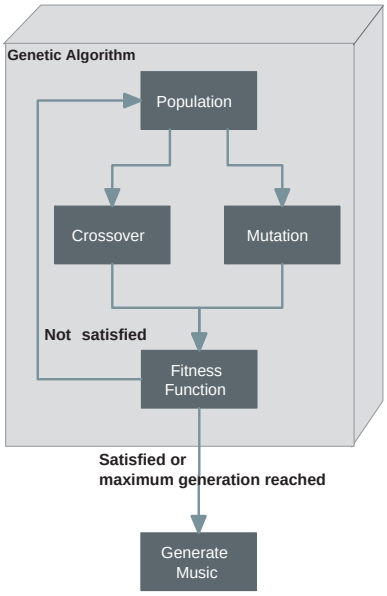


Figure 3.6: System architecture of Musical Sequences Evolver.

mutation operators are governed by a set of musically meaningful genetic operators (Sequence-Oriented Variation Operators, see Listing 3.1) which are defined by the author. The reason we used single chromosome and offspring in the population is because we aimed to investigate the immediate effect of each operator on the evolving melody.

Cellular Automata Music Evolver (Indirect Approach)

Cellular Automata Music Evolver is developed using an indirect approach to evolve music. It uses an evolutionary algorithm to evolve cellular automata that act as a generative model for music generation. Figure 3.7 shows the system architecture of the Cellular Automata Music Evolver. The system consists of three core parts, the EA system, CA system and a music mapping algorithm. The system uses an evolutionary algorithm to evolve CA's

transitional rules. Each rule will be passed to the CA system for generating the space-time pattern, then music will be rendered using a music mapping algorithm that interprets the space-time pattern. The rendered music will then be passed to the fitness function for an evaluation and assigned a fitness. This process is repeated until the rendered music satisfies the fitness function or reaches the maximum number of generations. The settings of each component in the experiments is described in chapter 5.

There are two reasons for using CA as the generative model in our indirect approach. First, CA is a dynamic model that is capable of generating from random to complex patterns including self-replication structure-like patterns. This character is well observed and supported by researchers [108] [219] [115] [181] [217]. Our approach relies on the CA model generating structure-like music. Second, using our method to render music allows the user to foresee and evaluate the generated music by visually seeing the generated space-time pattern.

Music Mapping Algorithms

We have innovated two music mapping algorithms inspired by Reiners's mapping algorithms [174]. These two music mapping algorithms are designed to operate on sets of horizontal cells running down the time step in the space-time pattern. These sets of horizontal cells are called 'Strip' in this thesis. Strip is the only pattern considered for rendering music by the music mapping algorithm. In order to use the mapping algorithms, the location of the strip within the evolved space-time pattern must be determined. We have designed two methods for determining the location of the strip:

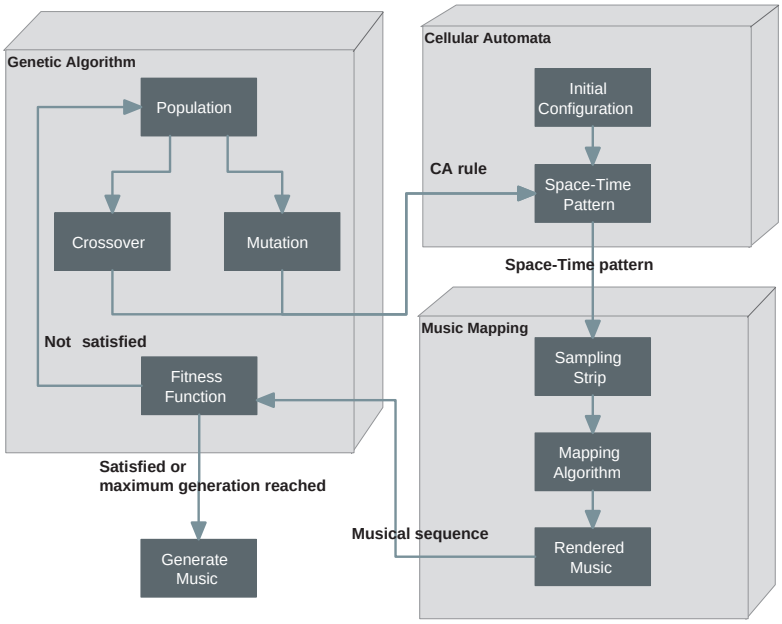


Figure 3.7: System architecture of Cellular Automata Music Evolver.

Middle 7-bit Taking the 7-bit strip at the location of the middle of the space-time pattern. This method is simple and uses less evaluation time. Figure 3.8 shows an example of strip that is sampled using Middle 7-bit sampling method.

Scan them all Scan all the possible 7-bit strips in the space-time pattern, render them into music using a music mapping algorithm, then evaluate all the rendered music using the fitness function and return the strip that corresponds to the music with the highest fitness. This method takes longer to perform an evaluation than the Middle 7-bit method, but it ensures that the highest fitness strip is found in the pattern.

The following describes the music mapping algorithms:

Scale Based Row to Binary (SBRB) (Monophonic) This music map-

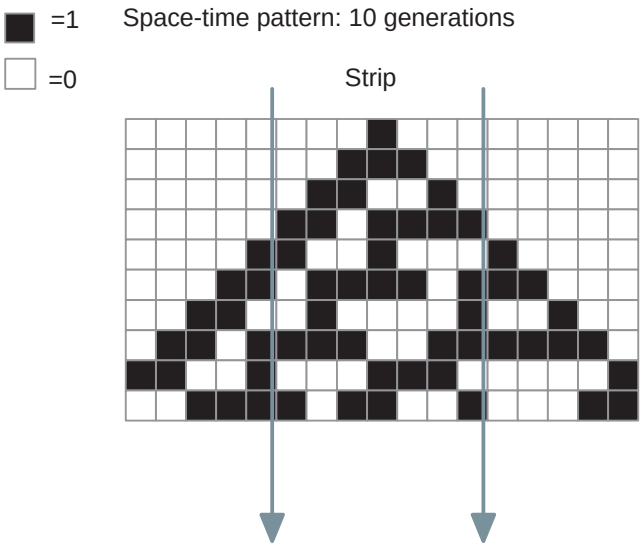


Figure 3.8: Example of sampling strip using a Middle 7-bit sampling method.

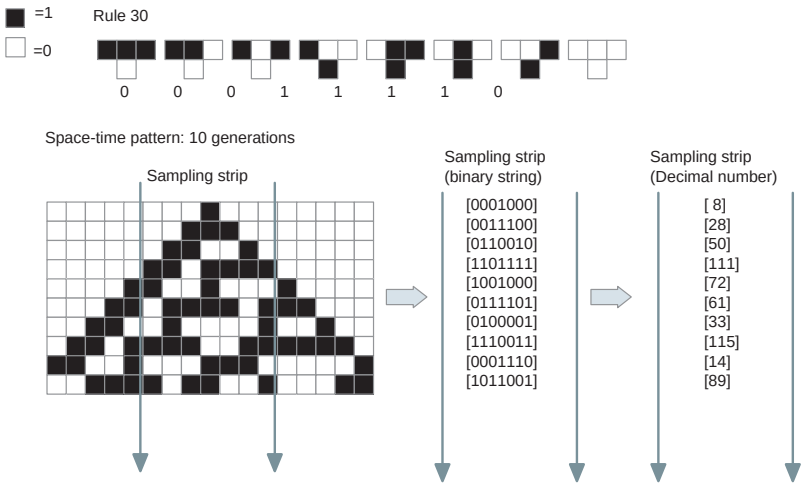
ping algorithm is similar to Reiners's 'Row to binary number' algorithm, the difference being that the rendered musical sequence is constrained in such a way that pitches are forced to make a selection from a set of notes within a particular scale. Therefore, the SBRB algorithm ensured all the rendered notes are horizontally harmonised, so there are no odd notes which are outside the given musical scale. There are three parameters in the SBRB algorithm; type of musical scale, first pitch in the scale and scale's octave range. For example, if the given musical scale is C major, the first pitch is C4 (Middle C in MIDI code) and the octave range is 2. Then the scale notes are C4, D4, E4, F4, G4, A4, B4, C5, D5, E5, F5, G5, A5, B5, C6. The SBRB algorithm maps each set of horizontal cells in the strip to a particular note in the scale notes, the mapping runs through the strip from top to bottom to form a mu-

sical sequence. In the process of individual note mapping, each set of horizontal cells is represented in a binary string format then converted into a decimal number. This decimal number is used for determining the note position in the scale of notes using the Equation 3.1. Note that the length of given a binary string must be 7-bit. A 7-bit binary string is enough to represent any note in MIDI, which only has pitch in the range of 0 to 127.

$$Note\ Position = \frac{BSD * TSN}{MNP}, \tag{3.1}$$

where BSD is the decimal number of the given binary string. TSN is the total number of the notes in the given scale notes. MNP is the maximum number of pitches that can be represented in the given binary string, which in this case is 128. Given here is an example showing how to determine note position using Equation 3.1 given the above musical scale and a binary string [0 1 0 0 1 0 0] that is converted into a decimal number 36. Working out Equation 3.1, the note position of this binary is $(36 * 15)/128 = 4.2$. We get the value 4 with the remainder of the result ignored. Therefore the binary string is mapped to the 4th position in the given scale, which is note F4. Figure 3.9 illustrates this music mapping algorithm when given a CA transitional rule 30.

Multiple Scale Based Row to Binary (MSBRB)(Polyphonic) This mapping simply combines multiple SBRB mapped sequences into polyphonic music. All the sequences of pitches are played simultaneously. Because the MSBRB algorithm is inherited from the SBRB algorithm,



Music mapping algorithm = Scale Based Row to Binary (SBRB) .
Chosen scale = C Major, Start note = C4 and Octave = 1.

Hence, Scale notes= {C4, D4, E4, F4, G4, A4, B4, C5}.

After applied SBRB mapping algorithm:

Sampling strip sequence =[8, 28, 50, 111, 72, 61, 33, 115, 14, 89] .

Sampling strip sequence (Position in Scale notes) = [0, 1, 3, 6, 4, 3, 2, 7, 0, 5].

Final musical sequence = [C4, C4, E4, A4, F4, E4, D4, B4, C4, G4].

*Note that if note position < 1, the note position is consider as 1.

Figure 3.9: An example of music rendering using SBRB music mapping algorithm.

all the played notes are within the same scale, so multiple notes played simultaneously are vertically harmonised. This feature is useful for creating musical chord.

3.2.8 System Evaluation

The success of the system is evaluated by using empirical experiments. For the internal evaluation, first, experiments are conducted to evaluate the discriminative power of the proposed music evaluators and to demonstrate how some of the existing well-known music shares the features of the music evaluators. Second, is a demonstration of how the configurations of the EA affect the abilities of the system to converge to a highly fit piece of music. For the external evaluation, we carried out two online music surveys, which were based on asking human subjects to rate our music samples (composed by a human and different types of machines) and to answer whether the music is composed by a machine or human. If evolved music received a higher mean rating than the baseline randomly-generated music, then the evolved music is considered as pleasing music. If evolved music received more than 50% of human listeners answered it is composed by human, then the evolved music is considered as indistinguishable from human-composed music. Finally, the surveys included asking each candidate to comment on our music samples, providing a qualitative analysis to obtain some idea of how the human subjects perceive the evolved music.

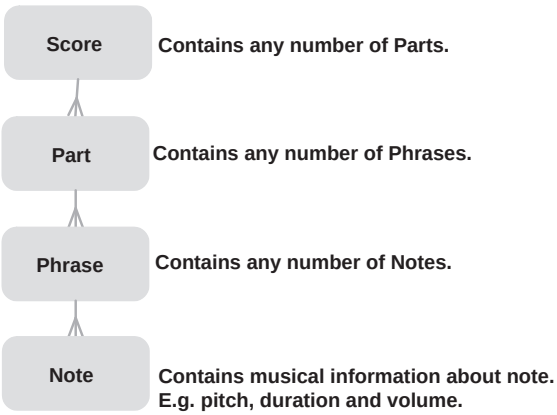


Figure 3.10: Music representation in JMusic.

3.2.9 Research Resources and Training Data

The proposed systems are written solely in Java programming language. For the sound manipulation, we used an open source Java package called JMusic[2] , which is a MIDI sound manipulation package developed by Andrew Sorensen and Andrew Brown at Queensland University of Technology. For music representation, MIDI files can be automatically translated into an hierarchical data structure in Java, and also reversed. Figure 3.10 shows the music representation in JMusic.

For training data, a total of 282 classical music samples in MIDI format were used in the experiments. These samples are chosen based on the author's preference. The samples consist of 110 by Mozart (Wolfgang Amadeus Mozart), 72 by Beethoven (Ludwig Van Beethoven) and 100 by Chopin (Frederic Chopin), approximately 470,000 notes in total. Each music sample is around 4 - 5 minutes long and most of them are polyphonic.

[2]http://jmusic.ci.qut.edu.au/

All music samples are downloaded without being quantised from the online music database[3].

3.3 Music Evaluators (Fitness Functions)

In our study, the primary focus is to design fitness functions that aim to evaluate music in the context of pitch, so other musical information such note duration, note volume, tempo are ignored. Four fitness functions were defined for evaluating musical sequences in the evolutionary music composition systems, these fitness functions only take into account the pitch statistics and pitch arrangement in the musical sequences. The following describes the definitions of four fitness functions[4] and discusses their motivations, expectations and related issues.

3.3.1 N-gram Fitness Function

In our systems, first, the user provides a set of training music samples to train the N-gram model, then during the evolution, musical sequences are evaluated by the trained N-gram that returns probabilities of the sequences by working out the transition matrix. Theoretically, the expected optimal sequence found by the EA is the MLS which has the highest probability given by the N-gram model. Since, we know that the MLS tends to contain repetitive patterns and it has less chance of being musically pleasant, we have proposed two ways to resolve this problem: 1. Use a constraint embedded

[3]http://www.classicalarchives.com/

[4]Fitness is normalised in the range between 0 to 1, where 0 is the worst and 1 is the best.

into the EA system to avoid the evolved sequences converging to the MLS, the constraint is named as 'Bag of notes', see chapter 4 for more detail; 2. Design various fitness functions to cooperate with the N-gram fitness function to tackle the MLS repetitive problem by guiding the evolved sequences away from MLS. These fitness functions are described as follows from section 3.3.2 to 3.3.4.

3.3.2 N-gram Absolute Pitch Zipf's Law (NAP-ZLaw) Fitness Function

Since Zipf's Law properties are evidently found within well-known musical pieces [123] [121] and as we shall see in chapter 5, we analysed more than 280 musical pieces using various Zipf's Law metrics and the results showed that there were strong Zipf's Law properties in the sample music. We assumed that if music has Zipf's Law properties in some way (e.g. pitches, duration, etc...), it tends to be pleasant to listen to.

A Zipf's Law based fitness function is implemented in the proposed systems. This fitness function aims to guide the frequency of groups of pitches (named the N-gram window in this thesis) to converge to Zipf's Law property.

Using the NAP-ZLaw fitness function to evaluate a musical sequence, the sequence's fitness is calculated by the following procedure: first, the musical sequence is converted into a sequence of N-gram windows, and then the frequency of each unique N-gram window is counted and sorted by its frequency. Figure 3.11 illustrates how to count N-gram (Tri-gram, N=3) windows given a musical sequence. Using a gliding window, the sequence is fragmented into

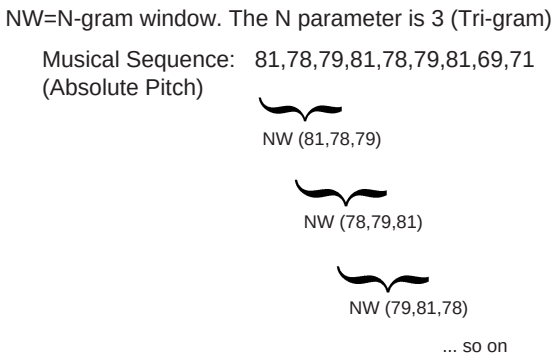


Figure 3.11: Counting N-gram windows.

overlapping N-1 subsections. A sequence of pitches is converted into a sequence of the N-gram windows. The size of N-gram window is dependent on the N parameter.

Second, using Zipf's Law metric, log-log slope and R-squared values will then be calculated. The weights for both normalised log-log slope and R-squared are 0.5, which is determined by observation. Equation 3.2 defines the NAP-ZLaw fitness function. Since the perfect log-log slope is -1 for Zipf's Law and the measured slope can be any number, in order to normalise the fitness of the slope it falls within the range of 0 to 1, maximum allowable error must be set and if the measured slope is greater than the maximum allowable error, then the NAP-ZLaw fitness is 0. Equation 3.3 defines the normalised fitness of a log-log slope. Note that the optimal fitness for the NAP-ZLaw fitness function is 1 that is equivalent to the log-log slope being equal to -1 and the R-squared value being equal to 1.

$$NAP - ZLaw\ fitness = (NSF \times W_{NSF}) + (RS \times W_{RS}) \tag{3.2}$$

Where NSF is the normalised fitness for the log-log slope, W_{NSF} is the weight of NSF, RS is the R-squared value and W_{RS} is the weight of RS.

$$\begin{gathered} NSF = 1 - \frac{\mid PZ - ZS \mid}{Max.Err}, \\ if \ \mid PZ - ZS \mid > Max.Err, \ then \ NAP - ZLaw\ fitness = 0. \end{gathered} \tag{3.3}$$

Where PZ is the perfect Zipf's Law slope -1, ZS is the actual log-log slope from the data and $Max.Err$ is the maximum allowable error, which in our case is set to 0.5

For example, in Table 2.1 the music sample Mozart K133 has a log-log slope of -1.1017 and an R-squared value of 0.819. Its NAP-ZLaw fitness is calculated by working out Equation 3.2 and Equation 3.3 giving:

$$\begin{gathered} NAP - ZLaw\ fitness = \left(\left(1 - \frac{(-1)-(-1.017)}{0.5}\right) \times 0.5\right) + (0.819 \times 0.5) \\ NAP - ZLaw\ fitness = 0.892 \end{gathered}$$

In practice, when working with the NAP-ZLaw fitness function, there might be a situation where the musical sequence[5] contains a few repetitive N-gram windows with a perfect Zipf's Law slope. To overcome this problem,

[5]For example, a melody consists of 2 different notes with frequencies of 100 and 50. This melody is not interesting to listen to even though it has a perfect -1 Zipf's Law slope. Therefore, a pleasant melody should contain a greater variety of notes with Zipf's Law properties.

we used 'Add-one' discounting method, when all possible N-gram windows that have not occurred are given an initial occurrence count of 1.

However, we observed some results and suggest that the 'Add-one' discounting method is advantageous only when applied to Uni-gram. That is because when the N parameter is more than 1, the total number of possible N-gram windows is large where most of the N-gram windows never occur in the sequence. Therefore, if 'Add-one' discounting method is applied, many tied frequencies of 1 will be appeared. It will skew the calculation of linear regression and affect the fitness calculation.

3.3.3 N-gram Absolute Pitch Entropy (NAP-Entropy) Fitness Function

Using N-gram Absolute Pitch Entropy Fitness Function, music is objectively evaluated using an information-theoretic approach which is simple, easy to implement and understand. NAP-Entropy fitness function works similar to the NAP-ZLaw, but instead of using Zipf's Law to evaluate musical sequences, information entropy is used. N is the parameter of the fitness function used to determine the size of the N-gram windows. NAP-Entropy aims to guide the frequency of N-gram windows to converge to a uniform distribution, which has a maximum entropy. The fitness assignment for the NAP-Entropy fitness function is defined in Equation 3.4.

$$NAP - Entropy\ fitness(x) = \frac{Entropy(x)}{MaximumEntropy(x)} \tag{3.4}$$

We use the Mozart sample K133 again to demonstrate the fitness as-

signment of the NAP-Entropy fitness function, where the N parameter is 1 (Uni-gram). Table 3.2 is the pitch-frequency-probability table for the Mozart sample K133. To obtain the maximum entropy, we need to give the number of possible events to the Equation 2.5. In this case the events are the possible pitches, which is 20. Working out the Equations 2.4, 2.5 and 3.4 gives:

$$Entropy(Pitch) = 3.530591$$

$$Maximum\ Entropy(Pitch) = 4.321928$$

$$NAP - Entropy\ fitness = 0.816902$$

Absolute pitch	Frequency (Freq)	Probability (Freq/Total Freq)
74	48	0.198
78	28	0.116
76	27	0.112
81	23	0.095
79	22	0.091
73	20	0.083
69	19	0.079
71	12	0.050
62	10	0.041
67	10	0.041
80	6	0.025
83	6	0.025
66	5	0.021
64	3	0.012
72	3	0.012
	Total =242	

Table 3.2: Pitch-Frequency-Probability table from K133 1772 Salzburg Symphony No. 20 in D Mvt3.

Note that for Uni-gram NAP-Entropy, it only evaluates the pitch distribution of sequences, and does not take into account the pitch arrangement.

It means that even if two melodies sound different, if their pitch distributions are the same then so is their fitness.

3.3.4 N-gram Histogram (NH) Fitness Function

The N-gram histogram (NH) fitness function aims to capture the global statistic of pitch frequencies in the samples (target) and guide the evolved sequences toward the target pitch statistic. Therefore, each evolved sequence has its N-gram histogram and compares it to a target N-gram histogram that is trained from the samples. In this fitness function, instead of precisely counting the frequency of each unique N-gram window, it counts the numbers of N-gram windows that are occurring at specific frequencies. This fitness function is specifically designed for cooperating with the N-gram fitness function to reduce the repetitive pitches in the evolved sequences and to be generalised enough to allow the evolved sequences to have certain numbers of repetitive N-gram windows, which are not specific to any particular N-gram window. The N-gram histogram fitness function has two parameters, N and B. Where N is the N parameter of the N-gram window and B is the number of histogram bins, which normalise the histogram using a fixed number of weighted bins. Equation 3.5 shows the normalisation process:

$$Weighted\,Bin(x) = \frac{Freq(x)}{MFreq} \times B \tag{3.5}$$

Where x is a particular number of N-gram windows, $Freq(x)$ is the frequency of x , $MFreq$ is the maximum frequency, B is the number of histogram bins.

Figure 3.12 demonstrates how to evaluate a test sequence using an NH fitness function. In the figure, first, the target N-gram histogram is trained from 110 Mozart samples. Both the target and test sequence of N-gram histograms are constructed using Equation 3.5. The results are shown from (a) to (b), and (d) to (e) respectively. Then these histograms are converted to probability distributions using Equation 3.6. The results are shown in (c) and (f) respectively. Second, the similarity between the target and test probability distributions is measured using Kullback-Leibler divergence (KL divergence or Relative Entropy) [105]. From the aspect of information theory, the KL divergence indicates how much information is lost when model P is used to represent reality Q. In the other words, KL divergence is a measure of the difference between two probability distributions. It is not symmetric, therefore it is divergent. Equation 3.7 is the definition of KL divergence (D_{KL}), where P is the optimal (reality) probability distribution, Q is the probability distribution (model) that is being evaluated. KL divergence is a value that represents the distance of Q from P. In this case, we used symmetrised KL divergence, which is given in Equation 3.8. Finally, the resulting symmetrised KL divergence is normalised into a fitness value using Equation 3.9.

$$Probability\,of\,Weighted\,Bin(x) = TFreq(x)/n \tag{3.6}$$

Where x is the weighted bin number, $TFreq(x)$ is the total frequency of N-gram windows occurring in weighted bin x and n is the total frequency of N-gram windows in all weighted bins.

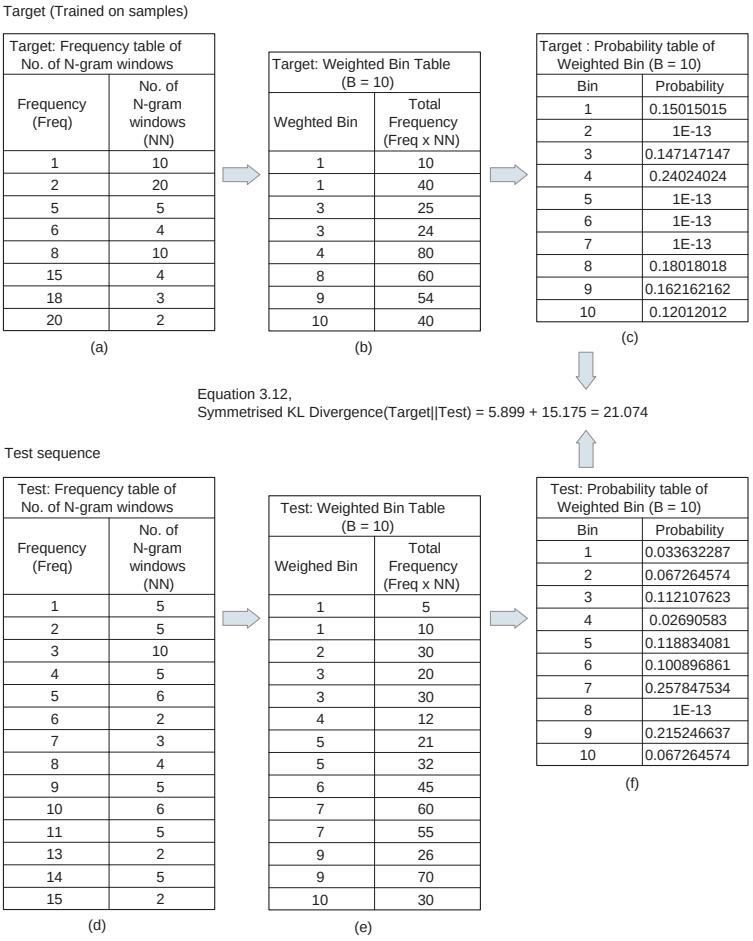


Figure 3.12: Example of fitness assignment using N-gram Histogram fitness function.

$$D_{KL}(P||Q) = \sum_i P(i) log \frac{P(i)}{Q(i)} \tag{3.7}$$

$$Symmetrised D_{KL}(P||Q) = D_{KL}(P||Q) + D_{KL}(Q||P) \tag{3.8}$$

$$N-gram\ Histogram\ fitnesss = 1 - \left(\frac{SymmetrisedD_{KL}(P||Q)}{Max.Err}\right),$$
$$if\ SymmetrisedD_{KL}(P||Q) > Max.Err,\ N-gram\ Histogram\ fitness = 0. \tag{3.9}$$

Where $SymmetrisedD_{KL}(P||Q)$ is the symmetrised KL divergence between target and test probability distributions, $Max.Err$ is the maximum allowable symmetrised KL divergence.

3.4 Chapter Summary

This chapter aims to provide the complete implementation details of the proposed evolutionary music composition systems, including the framework, system architectures and details of the four fitness functions that are used for objectively evaluating musical sequences in the EA systems. However, minor alterations to the systems settings and parameters were made in the individual experiments, depending on the aim of the experiment, as we shall see in the next chapters 4, 5 and 6.

Chapter 4

Evolving Musical Sequences Using N-gram Model

In this chapter we focus on the evaluation of the ability of using the N-gram model as a fitness function in an evolutionary system for music composition. The evolutionary music system is developed using a direct approach to evolutionary music, which is described in section 3.2.7 Musical Sequences Evolver. When using the N-gram model to generate a long sequence of musical notes, musical sequences tend not to have a high level of structure in terms of phrase arrangement and accumulate mistakes with a great potential of having low probabilities that are obtained from the given N-gram model. Perhaps by employing evolutionary computational techniques, the N-gram fitness function is able to maintain the evolving sequence so it has a high probability and converges to an MLS.

The present chapter is organised as follows: Section 4.1 describes experiment 1 that aims to analyse the performance of using the N-gram model for

a composer classification task. This experiment demonstrated the discriminative power of the N-gram model. We assumed that if an N-gram model has a strong discriminative power to classify musical sequences, then it also has a strong evaluative power to assess the likelihood of the given musical sequences being generated by the N-gram model. In section 4.2 experiment 2, demonstrated that an optimal musical sequence evolved by the N-gram fitness function does not necessary sound good as the pattern of the music is repetitive. In section 4.3, experiment 3 investigated and discussed the impacts of different sequence-oriented genetic operators in an evolutionary algorithm with an N-gram fitness function. In section 4.4, experiment 4 investigated the difference of the evolved sequences between the musical representations: absolute pitch and pitch difference. In section 4.5, experiment 5, we proposed a method called the 'Bag of notes' constraint that solves a prior problem where the expected optimal sequence MLS evolved by using the N-gram fitness function contains repetitive notes, since any musical sequence that contains highly repetitive notes is not interesting to listen to. The 'Bag of notes' constraint results in the evolution of more pleasing musical sequences and the melodic shape of the melodies are more controllable and predictable with appropriate operators. Note that to make the experiments in this chapter simple and focus on pitch analysis, the duration generation method for all the evolved music is Fixed Note Duration, the duration of each note in the sequence is the same length. Section 4.6 describes the hypothesis claim made in section 1.3.3 (Hypothesis 1). Finally section 4.7 is a summary of this chapter.

4.1 Experiment 1: Composer Classification

This is an experiment that analyses the discriminative power of the N-gram model using a standard pattern classification evaluation technique - 'Leave-one-out' cross-validation (see Appendix A.1). The motivation behind this experiment was the idea that a classifier with the discriminative power to classify the compositions might also be good as a fitness function for music evolution. In this case, we set the N-gram classifier a task to classify the compositions of three composers of the same genre classical music. There is a reason for choosing samples that are categorised by composer rather than by musical genre. It is commonly stated that each genre of music follows particular musical rules, for example, a particular set of notes fits into a particular chord progression. But personal composition style is different as composers are free to compose any style of music by combining musical rules taken from both music genres and their own preferences. Hence, a composer classification task is much harder than a musical genre classification task.

A total of 282 classical music samples were used for training the N-grams, those were 110 Mozart, 72 Beethoven and 100 Chopin, the music representation is absolute pitch. All the samples are pre-processed by 'All-Mono' music extraction algorithm (see section 2.2.1) and the detail of how to train N-gram model is described in section 3.3.1. For a composer style classification task, the classifier consists of three individual N-gram models and each one is assigned to a particular composer of Mozart, Beethoven and Chopin. The composers are assumed to have equal prior probabilities, and each test melody is assigned to a composer whose N-gram model rated the sequence as

having the highest likelihood. In the experiment, two N-gram classifiers were tested, Bi-gram (N=2) and Uni-gram (N=1) for each classification task, and both applied the 'Add-one' discounting method.

Experimental Results

Table 4.1 is a summary of the experimental results of a Bi-gram classifier and Table 4.2 shows its individual result. The results show that the average rate of correctly predicting the composer is 231/282=81.9%. In addition we calculated the confidence intervals, the actual average correct rate lies within the interval of 77.8% - 85.3% with a 90% confidence. The Uni-gram results are shown in Table 4.3 and 4.4. It achieved an average correct prediction rate of 218/282=77.3%, with a confidence interval of 72.9% - 81.1%. Inspecting the result for each individual composer, we found that when both Bi-gram and Uni-gram models failed to predict Mozart and Beethoven music, they mis-classified most of the classes that fall between Mozart and Beethoven. That means the styles of Mozart and Beethoven are more similar relative to the style of Chopin. When both models failed to predict the Chopin music, most of the failures fell into the Beethoven class. That means the style of Chopin is closer to Beethoven than Mozart. To illustrate this interpretation, pitch frequency presented in a histogram for each composer is provided in Figures 4.1, 4.2 and 4.3. The overall statistics of pitch frequency for Mozart and Beethoven are similar, while Chopin's music has a wider pitch range. Note that Rest note occurred the most in the samples. Therefore we expected the MLS for a Uni-gram to be a string that repeats the Rest note only. In the results, we were surprised to see that the Uni-gram performed nearly as well

		Predicted		
		Mozart	Beethoven	Chopin
Actual	Mozart	93	16	1
	Beethoven	10	60	2
	Chopin	0	22	78

Table 4.1: Confusion Matrix: Bi-gram classifier (Absolute Pitch).

Mozart	Beethoven	Chopin
84.5%	83.3%	78%

Table 4.2: Individual correct prediction rate: Bi-gram classifier (Absolute Pitch)

as the Bi-gram. When these models fail, they fail in different ways. Bi-gram classifiers fail because they over-fit the data, while Uni-gram classifiers fail because they under-fit it.

4.2 Experiment 2: Maximum Likelihood Sequence

In this experiment, given an N-gram model, we computed the maximum likelihood sequence and used a simple Random Mutation Hill Climber (RMHC) to evolve a single musical sequence with the N-gram fitness function. Both used a Bi-gram model that trained on 110 Mozart samples in absolute pitch

		Predicted		
		Mozart	Beethoven	Chopin
Actual	Mozart	92	17	1
	Beethoven	17	50	5
	Chopin	1	23	76

Table 4.3: Confusion Matrix: Uni-gram classifier (Absolute Pitch).

Mozart	Beethoven	Chopin
83.6%	69.4%	76%

Table 4.4: Individual correct prediction rate: Uni-gram classifier (Absolute Pitch).

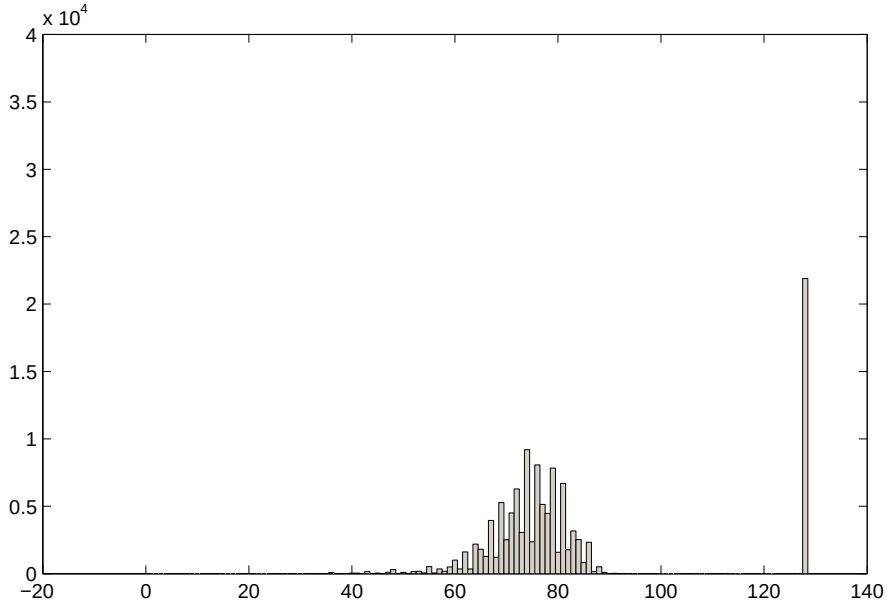


Figure 4.1: Mozart samples - absolute pitch histogram.

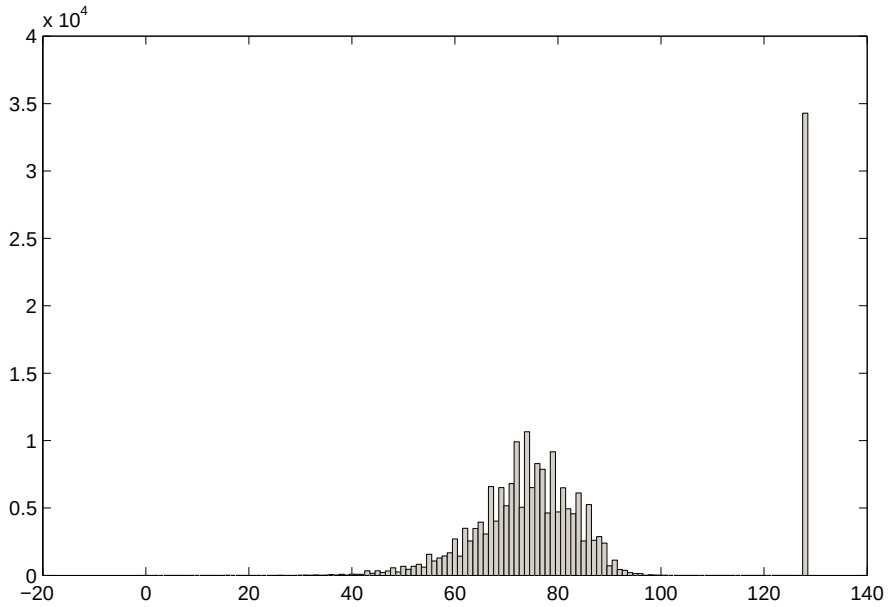


Figure 4.2: Beethoven samples - absolute pitch histogram.

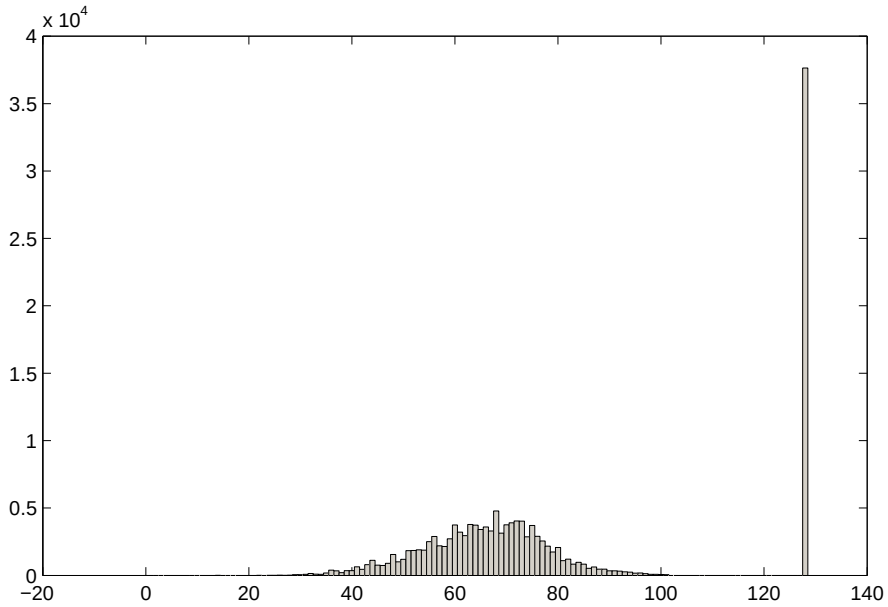


Figure 4.3: Chopin samples - absolute pitch histogram.

representation. The MLS was computed for two reasons: because it gives a value for the best possible fitness given the trained N-gram model and in order to listen to this sequence. The Viterbi dynamic programming algorithm was used to generate the MLS [96]. Then, we investigated the convergent ability of a simple RMHC by recording the fitness trajectory of the evolving musical sequence. The optimistic hope is that RMHC is able to find the global optimal which is the MLS.

The RMHC used in this experiment is different from Forrest and Mitchell's RMHC [70], as the chromosome is encoded as a vector of real-value. From an EA perspective, the RMHC works similarly to the (1+1) evolution strategy, which is an evolutionary algorithm for real-value parameter optimisation. The parameter settings of the RMHC are: 600,000 generations, the length of chromosome is 250, the size of population and offspring are both set to 1. The mutation operator for the RMHC is operator 0 which is taken from the sequence-oriented variation operator in section 3.2.6. Operator 0 works by randomly selecting and modifying a single note in the sequence.

In the result of MLS generation, the MLS is ([77] [79] [81] [79] [81] [79] [81] [79]...(repeat)...[77] [76] [74] [128]). The fitness of MLS = Antilog -177.26 given the Bi-gram model. The pattern of MLS is based on two repetitive notes [81][79]. This is because the transition probability from note [81] to [79] is high, in our case it is Antilog -0.5918. Therefore a sequence containing these two state transitions in repeating order would be the most probable sequence that the system could generate. The reason that the last four notes in the MLS were not repeated is because the fitness of sequence ([77] [76] [74] [128] [77]) = Antilog -3.1286 is lower than the fitness of sequence ([79]

[81] [79] [81] [79]) = Antilog -2.8683. But when both sequences had the last note removed the fitness of sequence ([77] [76] [74] [128]) = Antilog -1.8814 is higher than the fitness of sequence ([79] [81] [79] [81]) = Antilog -2.2764, therefore the ([77] [76] [74] [128]) would be the optimal last four notes.

For the RMHC, the experiment result is shown in Figure 4.4. The fitness of MLS = Antilog -177.26 and the fitness of the initial random sequence = Antilog -576.19. The random sequence started to converge after generation 100,000. After 350,000 generations its fitness reduced to Antilog -255.98 and remained the same until 600,000 generations. The optimal sequence fitness is lower than MLS, it was trapped in a local optimum. Hence, the RMHC failed to search for the MLS. When listening to both sequences, the sound of the MLS is boring, as it keeps repeating two notes only. The evolved sequence sounds more interesting than the MLS, as it contains a reasonable variety of notes.

4.3 Experiment 3: Sequence-Oriented Variation Operators

The aim of experiment 3 is to analyse the performance of each musically meaningful operator in a EA system and given the Bi-gram and Uni-gram fitness functions. These operators are the sequence-oriented variation operators that we described in section 3.2.6. This experiment has the same setup as experiment 2, except that 10,000 generations were used. Note that at each run, a single fitness function is used and nine operators are individually run.

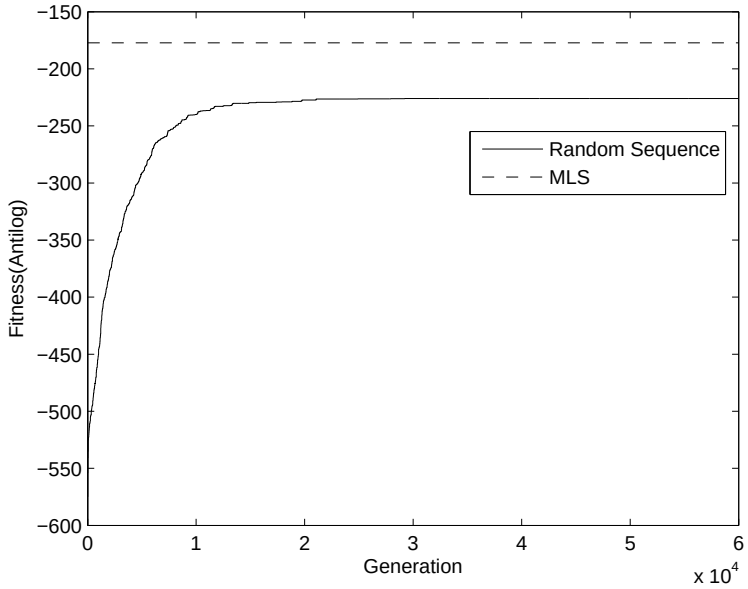


Figure 4.4: RMHC: random sequence evolved by operator 0 (random single note modification operator).

Figure 4.5 and 4.6 show the experimental results of Bi-gram and Uni-gram. The summary of experimental results is shown in Table 4.5 and the performance ranking summary is shown in Table 4.6. The following section is the analysis and describes the convergent power of each operator when applied to the either Bi-gram or Uni-gram fitness functions.

Operators Analysis

The analysis of operators 8, 5 and 0 is presented first, as these operators are the top ranked in the result. Then rest of the operators are presented in their numeric order. Note that some of the analysis compared the computed MLSs that are generated by Bi-gram and Uni-gram models. The Bi-gram MLS is taken from the experiment 2, it has the fitness of Antilog -177.26.

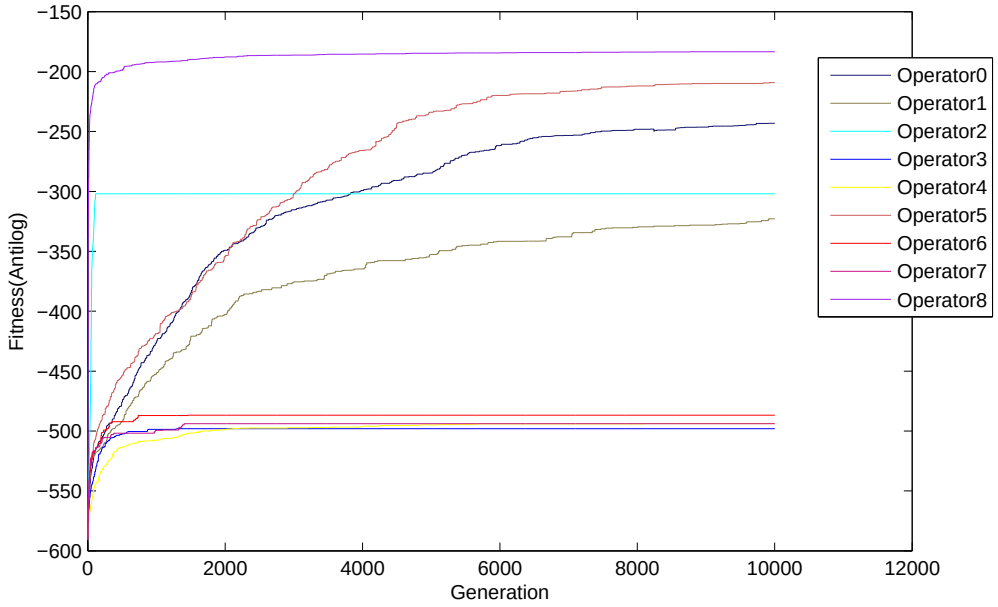


Figure 4.5: Performance of various operators given the Bi-gram fitness function.

Operator number	O.S.F in Bi-gram (Antilog)	O.S.F in Uni-gram (Antilog)
0 (Note change)	-243.12	-187.51
1 (Notes change)	-322.91	-322.99
2 (Copy & paste)	-302.03	-182.18
3 (Reversal)	-498.08	-965.75
4 (Transpose)	-493.75	-404.89
5 (Swap)	-209.22	-182.18
6 (Ascending)	-486.76	-965.75
7 (Descending)	-493.93	-965.70
8 (Copy from samples)	-183.40	-199.56

Table 4.5: Experiment 3 - the summary of operators results.
O.S.F = Optimal sequence fitness.

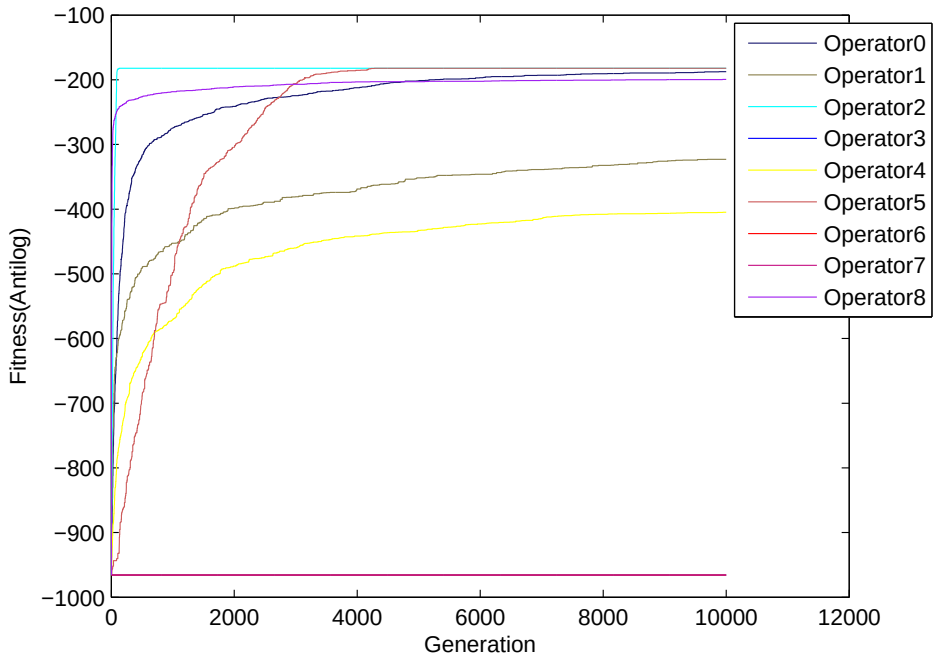


Figure 4.6: Performance of various operators given the Uni-gram fitness function.

Ranking	Bi-gram	Uni-gram
1	Operator 8	Operator 5 (converged to MLS)
2	Operator 5	Operator 2 (converged to MLS)
3	Operator 0	Operator 0
4	Operator 2	Operator 8
5	Operator 1	Operator 1
6	Operator 6	Operator 4
7	Operator 4	Operator 3
8	Operator 7	Operator 6
9	Operator 3	Operator 7

Table 4.6: Experiment 3 - performance ranking summary.

The Uni-gram MLS is computed beforehand, the sequence contains [128] [128] [128]...(repeat) and its fitness is Antilog -182.18. Integer value 128 represents a Rest note.

Operator 8 (Copy from samples) - This operator performs well on both models because this operator copied segments from the training samples to the evolving sequences. Therefore any segment that is copied is seen by the N-gram beforehand and the fitness of each copied segment is guaranteed to have a high fitness. For Bi-gram, this operator evolved the highest fitness sequence amongst all the other operators. Figure 4.7 shows the optimal sequence (Melody A) evolved by operator 8 with Bi-gram, the sequence has not converged into MLS. Overall, the evolved sequence is similar to the characteristic of MLS, as some parts contain repetitive notes. There is no large interval between successive notes, some of the paired notes are being repeated. For Uni-gram, the evolved sequence contains mostly Rest notes and a few different pitches. This is because the Rest note is occurred the most in the samples.

Operator 5 (Swap) - The swap operator swaps two segment locations, in additional to that it allows overlapping segments. This operator copies the existing high fitness notes and overwrites the lower fitness notes. For Bi-gram, this operator has a high convergent power. The evolved sequence is similar to the sequence evolved by operator 8. For Uni-gram, this operator has a high convergent power as well. The evolved sequence converged into MLS, which contains repeated Rest notes only.

Operator 0 (Note change) - This operator performs well on both Bi-gram and Uni-gram. The Uni-gram model is well suited to this operator, because

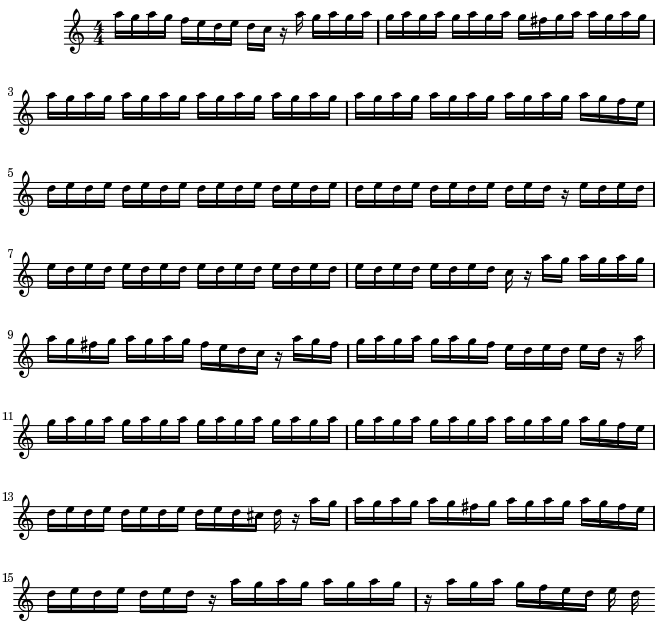


Figure 4.7: A sequence (Melody A) evolved by operator 8 with Bi-gram fitness function.

it only modifies a single note per operation. The Uni-gram model takes an individual symbol into account only, with no reference to the previous or next symbol. The fitness of the evolved optimal sequence is Antilog -187.5, very close to the MLS fitness Antilog -182.18. It has more Rest notes and the fitness is higher. Using this operator on a Uni-gram model will converge to the MLS given enough generations. But in this case it failed, the sequence is trapped in the local optimal, the Uni-gram fitness landscape is convex and similar to the well studied one-max problem. For Bi-gram, this operator gave a poorer performance and the fitness landscape has many local optima.

Operator 1 (Notes change) - This operator modifies a group of notes. The convergent power is lower than the operator 0 which modifies a single note value per operation. This is because each time when the operator randomly

modifies a group of notes, it is very hard to pick up the higher fitness notes. If the fitness of the modified sequence is less than the fitness of the unmodified sequence, then the modification is discarded, and the sequence remains unchanged.

Operator 2 (Copy & paste) - The convergent power of this operator is very high when applied on Uni-gram, as the evolved sequence converged into MLS. However, this operator gives lower convergent power when applied on Bi-gram, because for Bi-gram the note ordering is sensitive. Hence, it is hard to randomly copy a group of notes that has a higher fitness to paste into the target location of the sequence. However, Uni-gram does not have this problem, as note ordering does not affect the fitness of a sequence on Uni-gram.

Operator 3 (Reversal) - This operator is the worst performing on Bi-gram experiments because this operator does not change any single value of a note and it reverses the note ordering, nor does it modify the note ordering in a different way, it slows down the evolving sequence to converge to the MLS. This operator is even worse when it performs on the Uni-gram model, as it takes no account of the sequence fitness.

Operator 4 (Transpose) - The transpose operator did not perform well on both models. This is a little surprising because this operator modifies each note value in the segment. The reason for this is because the sequence allows overwriting only when the new modified sequence has a higher fitness; if it is lower, the sequence remains the same. For example the operator only allows +5 and -5 transpose, so if a chosen segment never gets the higher fitness within +5 and -5 transpose, then this segment would never change its note's

values.

Operator 6 (Ascending) and 7 (Descending) - These sorting operators did not perform well on both Bi-gram and Uni-gram. They sort the sequence in one way in either ascending or descending order only. Although these operators did not guide the sequence to reach a high fitness, they did guide the melodic contour of the sequence to make it more interesting, as the note ordering is ascending or descending. It is commonly observed that if a melody contains ascending or descending note ordering in the instrument solo section, it sounds interesting. Since these operators do not change the value of any note in the sequence, these operators do not affect anything on the Uni-gram model. Although these operators have low convergent power compared to the others, interesting patterns are produced. Figures 4.8 and 4.9 show the sequences that were evolved by operator 6 with Bi-gram and Uni-gram respectively. In Figure 4.8, the sequence started from a low pitch then continued to raise the pitch value up to a certain position followed by a sudden drop to a low pitch again before another raise. This pattern is repeated until the end. The sudden drop occurred because it allows the next ascending process to maximise the number of ascending notes. However, for Uni-gram, the pattern of the sequence is slightly different. The length of the ascending process is longer, because anything changed by the operator has no effect on the fitness, therefore the operator continues sorting the sequence through to the end of the generation. We reasoned that if N increases in the N-gram model, the length of the ascending process would decrease. Since operator 7 is a reversed version of operator 6, the sequences evolved by operator 7 are similar to these sequences but in a descending order.

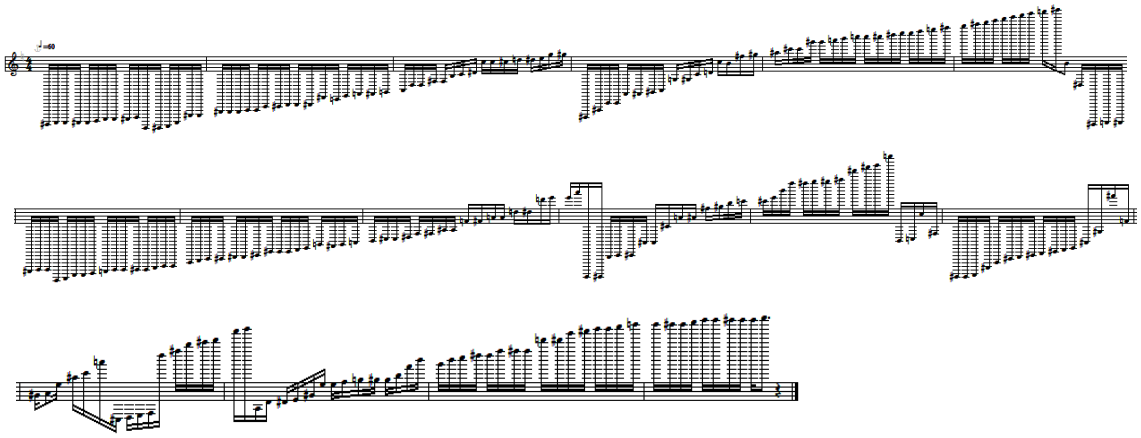


Figure 4.8: A sequence (Melody B) evolved by operator 6 with Bi-gram fitness function.

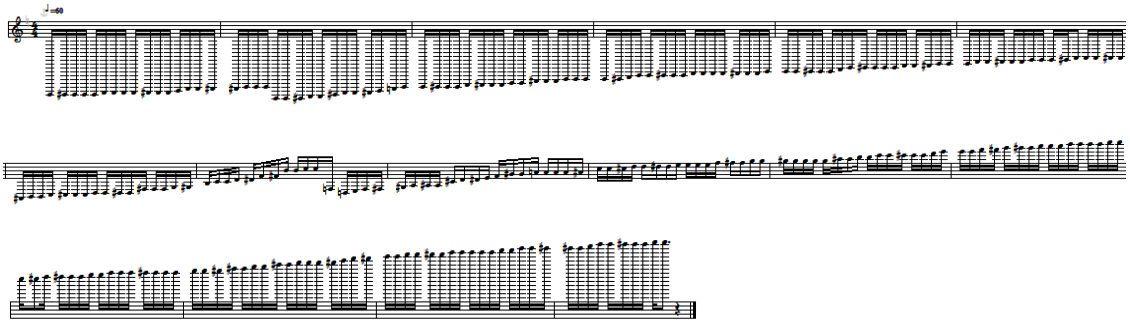


Figure 4.9: A sequence (Melody C) evolved by operator 6 with Uni-gram fitness function.

4.4 Experiment 4: Interval Statistics

The aim of this experiment is to investigate the impact of using a pitch difference representation in the N-gram model for the same system. There are two advantages when using a pitch difference N-gram model. First, it can be used to discriminate melodies which are in different musical key/scale. Second, this model is general enough to generate a variety of notes with a high fitness value. This is because it does not capture specific notes but instead the interval between them. For example, suppose there are two melodies (C-major and D-major scales) containing notes: C D E F G A B C and D E F# G A B C# D. Their fitness values are completely different under the absolute pitch N-gram model, but they are exactly the same under the pitch difference N-gram model, all the major scales are constructed by the same interval arrangement (+2,+2,+1,+2,+2,+2,+1).

This experiment consisted of two parts. Part 1 comprised the classification test, conducted in the same way as experiment 1 but using a pitch difference Bi-gram model. For part 2, an experiment used the pitch difference Mozart Bi-gram to generated ten melodies, and then evolved them using operator 9 (Observed distribution of single note modification). Using the model generated sequences for the initial sequences could speed up the evolutionary process. The rest of the experiment settings are the same as in the experiment 3.

		Predicted		
		Mozart	Beethoven	Chopin
Actual	Mozart	81	27	2
	Beethoven	15	55	2
	Chopin	11	22	67

Table 4.7: Confusion Matrix: Bi-gram classifier (Pitch Difference).

Mozart	Beethoven	Chopin
73.6%	76.3%	67%

Table 4.8: Individual correct prediction rate: Bi-gram classifier (Pitch Difference).

Experimental Results

Part 1: Composer Classification

The result summary of the composer classification task is shown in Table 4.7, the Bi-gram model achieved an average correct rate of $203/282 = 71.9\%$, the confidence interval is 67.3% to 76% with 90% confidence. Table 4.8 shows the individual result. The average correct classification rate of the pitch difference Bi-gram model is lower than the absolute pitch Bi-gram model, which achieved 81.9%. This result indicates that in our samples, composers tend to use a similar key, scale and pattern to compose music. This is common in the genre of classical music. The classifier fails to classify some music with similar patterns, and again, the result showed that the styles of Mozart and Beethoven are relatively more similar than to the style of Chopin. This is the same interpretation as in experiment 1.

Melody	Fitness before evolution (Antilog)	Fitness after evolution (Antilog)
Melody 1	-285.41	-166.79
Melody 2	-282.93	-158.09
Melody 3	-281.81	-162.21
Melody 4	-277.16	-163.05
Melody 5	-284.65	-157.93
Melody 6	-254.74	-155.20
Melody 7	-271.24	-161.43
Melody 8	-279.08	-157.54
Melody 9	-273.99	-160.00
Melody 10	-273.77	-165.90

Table 4.9: Result summary of evolving musical sequences by pitch difference Mozart Bi-gram.

Part 2: Evolving Musical Sequences

Ten musical sequences were generated by the pitch difference Mozart Bi-gram model, then evolved by operator 9 with the same Bi-gram model. The results summary is shown in Table 4.9. The fitness of the initial generated sequences was in the range of between Antilog -290 to Antilog -250, after the evolution process, the fitness values of the evolved sequences were in the range from Antilog -170 to Antilog -150.

Figures 4.10 and 4.11 show an example of a melody before and after the evolution process. Melody 5 was chosen as it has a high fitness and the sound of this melody is pleasant to us. Before the evolution, melody 5 contains a variety of notes, exhibits more (perhaps excessive) variation in pitch and some large intervals occurred between successive notes but not as much as occurred in the randomly generated sequence. After the evolution, the sequence of notes in Melody 5 is constant and smooth, there are no larger intervals between successive pitches. However, it contained as much variety

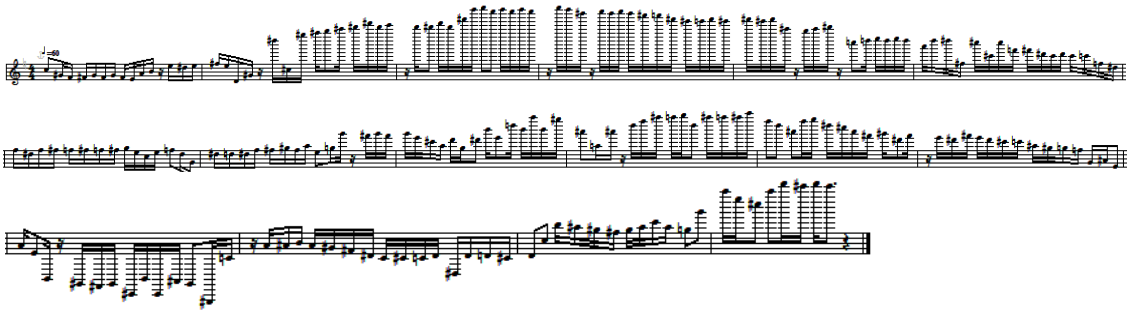


Figure 4.10: Melody 5 generated by pitch difference Bi-gram model.

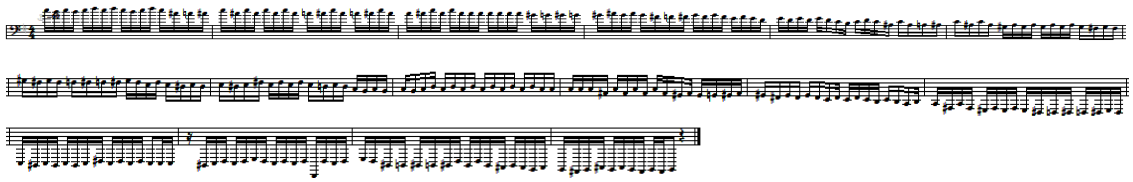


Figure 4.11: Melody 5 after evolution, evolved by operator 9 with pitch difference Bi-gram fitness function.

in the notes as it did before evolution, but the notes arrangement is more constant, like going up and down a stairway. For the rest of the evolved melodies, all of them contain a variety of notes and their fitness was higher after the evolution process. In this case the melodies evolved using the pitch difference model are more interesting than the melodies evolved using the absolute pitch model.

4.5 Experiment 5: 'Bag of notes' Constraint

In experiments 2, 3 and 4 we have observed that the optimal melodies that were evolved by the Mozart Bi-gram and Uni-gram fitness functions consisting of a single or group of repetitive pitches, such melodies are not interesting

to listen to. This problem is intrinsic in the N-gram model, as the maximum likelihood sequence generated by the N-gram model is repetitive. In this experiment, we proposed a simple constraint called 'Bag of notes' that is able to prevent the MLS repetitive sequence. The 'Bag of notes' constraint is a means of simply limiting the resource of pitches when evolving a melody. The optimal melody must have the same pitch distribution to the initial melody, both sequences contain exactly the same given notes. Therefore the variation between the initial and evolved melodies is only in the pitch arrangement. To operate with this constraint, the role of the human within the system is to select a set of melodies to train an N-gram model, and choose a set of notes, where notes can be repeated. The system then evolves melodies by permutating the order of the notes selected from the bag.

It is interesting to note that, the 'Bag of notes' constraint is able to avoid evolving melodies converging to MLSs, on the other hand, this constraint is able to tackle the problem of finding the closest pattern (pitch arrangement) to a particular composer's style when given a set of pitches e.g. to evolve a Beethoven melody in the style of Mozart composition.

In experiment 5, two experiments 5a and 5b were conducted with the aim of analysing the Musical Sequence Evolver that embeds the 'Bag of notes' constraint to solve the MLS repetitive sequence problem that arises from the MLS over N-gram models. The other aimed to investigate the effectiveness of each operator in enabling the evolving melody to converge to the optimal sequence (though in general this may be hard to find). As far as we are aware there is no existing algorithm that is guaranteed to find the N-gram rated MLS given the 'Bag of notes' constraint; the problem seems similar to

the travelling salesman problem when N is greater than one. Note that for a Uni-gram model (N=1) with a 'Bag of notes' constraint, all orderings have equal fitness, hence this constraint is not suitable for the Uni-gram model.

Experiment Settings

Table 4.10 shows the experiment settings for both experiments. The only difference between the two experiments is the representation in the N-gram model: absolute pitch representation is used in experiment 5a. Pitch difference representation is used in experiment 5b. For the 'Bag of notes' constraint, it is handled by the sequence-oriented variation operators: operators 3, 6, 7, 10 and 11 taken from the section 3.2.6. These operators are derived from three common melody composition techniques, ascending, descending and reversal. Each operator is guaranteed to only rearrange the pitch/pitch difference ordering within a melody. Hence, these operators satisfy the requirement of the 'Bag of notes' constraint. Note that due to the constraint that we applied here, the traditional crossover operator is omitted.

At each run, the initial sequence is generated by randomly shuffling the original Mozart melody. Figure 4.12 shows the original Mozart melody and Figure 4.13 shows an example of the initial sequence, which is a shuffled Mozart melody. The initial melody contains more pairs of successive notes with large intervals, it sounds irritating and not pleasant to hear.

Experimental Results

Figures 4.14 and 4.15 show the experimental results of the absolute pitch Mozart Bi-gram and the pitch difference Mozart Bi-gram respectively. The

GA parameters	
Population size	1
Offspring size	1
Chromosome length	250
Generation	10,000
Fitness Functions	Either absolute pitch Mozart Bi-gram or pitch difference Mozart Bi-gram
Genetic operators	Operator 3, 6, 7, 10 and 11. Each evolution only used a single operator. The probability of execution is 1.0, which ensures the operator execute at each generation.
Initial sequence	Randomly shuffle a sequence that is a melody composed by Mozart, named K310 Sonata in A minor Mvt.3, which has a fitness Antilog -306 that is assessed by the absolute pitch Mozart Bi-gram and Antilog -316 that is assessed by the pitch difference Mozart Bi-gram.

Table 4.10: Experiment 5 - experiment settings.

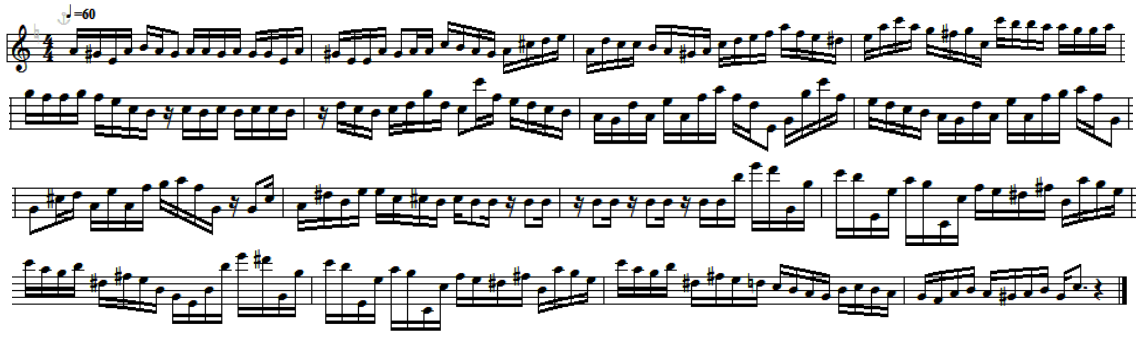


Figure 4.12: Mozart's K310 Sonata in A min Mvt.3 (extracted monophonic melody).

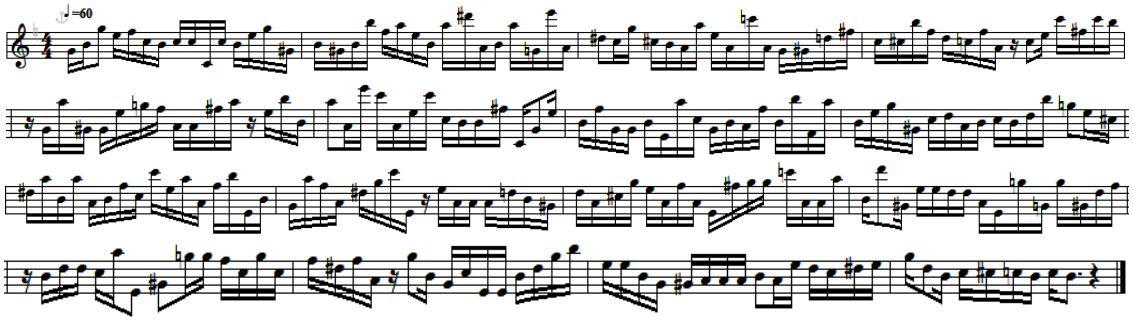


Figure 4.13: Randomly Shuffled Mozart's K310 Sonata in A min Mvt.3.

Operator Number	O.S.F in absolute pitch Mozart Bi-gram (Antilog)	O.S.F in pitch difference Mozart Bi-gram (Antilog)
6 (Ascending)	-282.03	-313.24
7 (Descending)	-271.15	-312.16
3 (reversal)	-243.16	-267.21
10 (Point Swap)	-236.78	-248.79
11 (Notes Shuffle)	-268.56	-278.35

Table 4.11: Experiment 5 - results summary.
O.S.F = Optimal Sequence Fitness (Average of 500 sequences).
Note: The average fitness of initial sequences is Antilog -402.53

summary of results is shown in Table 4.11. The following section is the analysis of each operator:

Operator 6 (Ascending) - This operator has the lowest convergence among other operators in both experiments. It forced the evolved melody into a shape of ascending pitch arrangement (see Figure 4.16). Even though the fitness of this melody is low when compared to other melodies that were evolved by other operators, the perceived sound of this melody is quite interesting to hear. This is because its pattern is similar to arpeggio[1] and sounds like climbing up a stair way, then dropping down immediately and

[1]In music, notes in a chord are played sequentially.

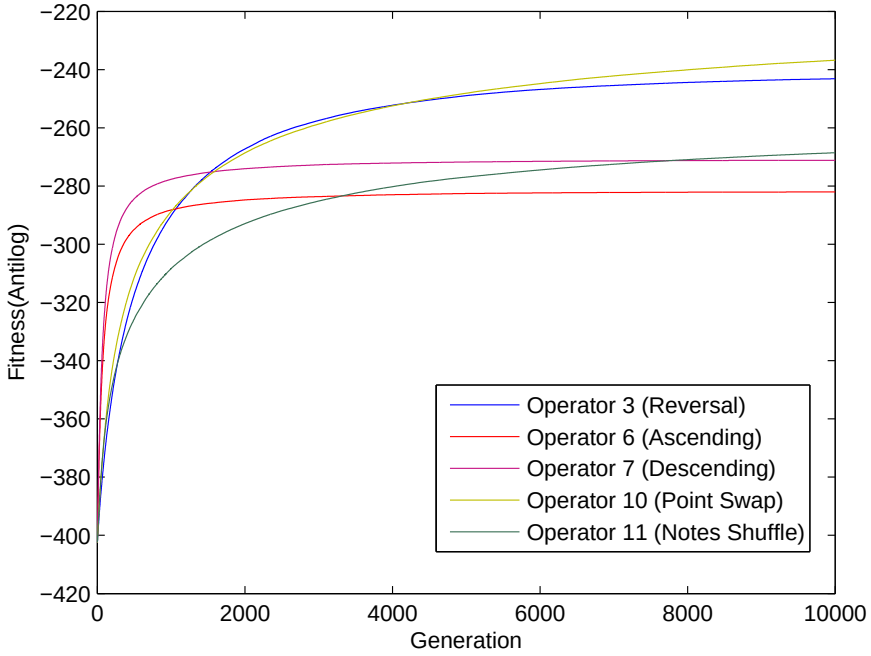


Figure 4.14: Experiment 5a - performance of various operators given the absolute pitch Mozart Bi-gram fitness function.

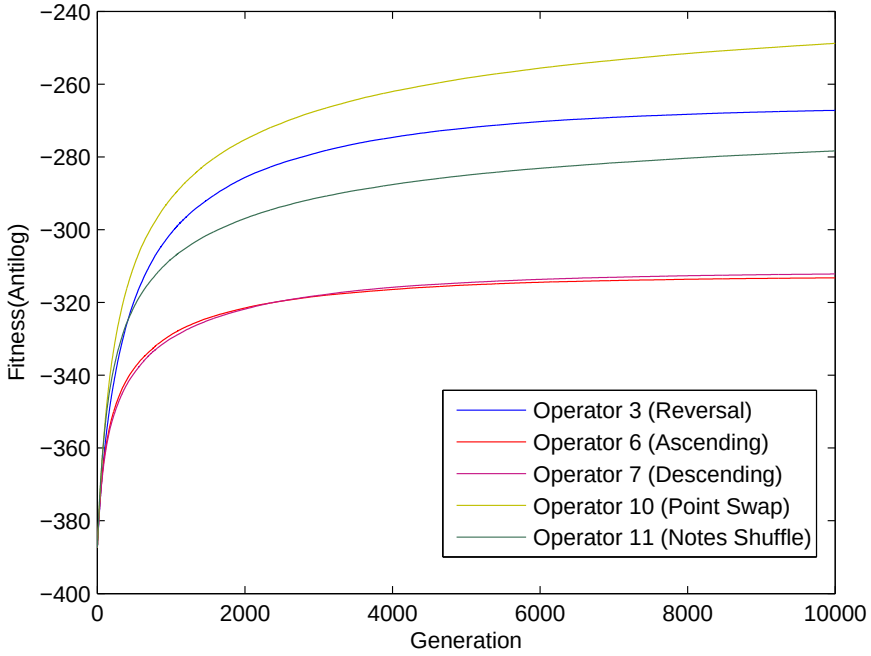


Figure 4.15: Experiment 5b - performance of various operators given the pitch difference Mozart Bi-gram fitness function.

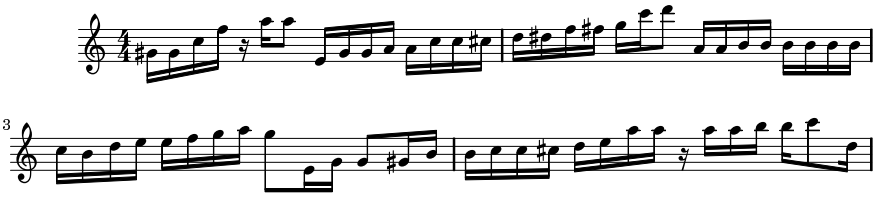


Figure 4.16: A melody (Bar 1-4) evolved by absolute pitch Mozart Bi-gram fitness function and ascending operator.

then climbing up again. For the pitch difference Mozart Bi-gram in experiment 5b, the melody shape of the evolved melody is more balanced with ascending and descending pitch arrangement (see Figure 4.17). The reason is that within the pitch difference representation, the maximum rise and drop intervals are lower than the absolute pitch representation. For example, consider a melody containing pitches [72,74,76,77,79,77,74,73], the pitch difference representation of this melody is [+2,+2,+1,+2,-2,-3,-1]. The maximum rise and drop intervals are +2 and -3. But using the absolute pitch representation, the maximum rise and drop intervals are +7 (from pitch 72 to 79) and -7 (from pitch 79 to 72) when considering any possible ordering of the sequence. Therefore by using absolute pitch representation, the melody can have a large drop interval between successive pitches. By using pitch difference representation, the melody is forced to use all the pitch difference in the pitch difference 'Bag of notes' so the evolved melodies will not contain as many large immediate drop pitches as the evolved melodies that are represented in absolute pitch representation.

Operator 7 (Descending) - This operator has the second lowest convergence amongst the others. It has a similar effect to operator 6. The only difference is that this operator forces the melodic shape of the evolved melody

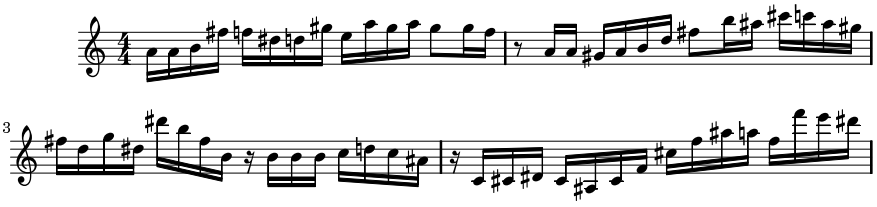


Figure 4.17: A melody (Bar 1-4) evolved by pitch difference Mozart Bi-gram fitness function and ascending operator.

into descending order. In both experiments, the descending operator outperforms the ascending operator although this is not statistically significant. Based on observation, we suggest that possibly most of the sample melodies are in descending pitch arrangement rather than in ascending pitch arrangement.

Operator 3 (Reversal) - This operator has the second highest convergence amongst the others. When examining all the high fitness melodies, they share the same feature as the melodic shape is balanced with ascending and descending pitch arrangement. During the evolution, when an evolving melody reached a certain level of high fitness, it was very hard to evolve it to have a higher fitness by making large changes to the melody using this operator. However, this operator is able to reverse pitch ordering of a segment melody that might cause the melody to hit a high fitness value. For example, if a segment contains ascending pitch ordering, it becomes descending after the change. The descending version of this segment might hit a higher fitness value.

Operator 10 (Point Swap) - This operator has the highest convergence in both experiments. The melodic shapes of evolved melodies in both ex-

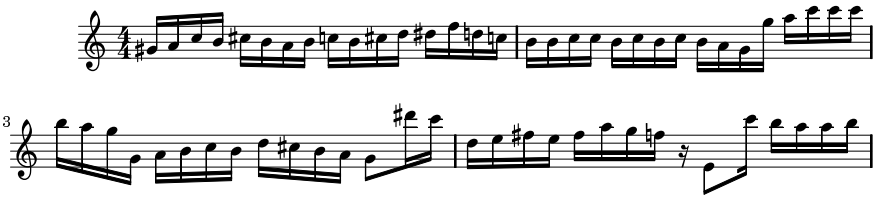


Figure 4.18: Example melody (Bar 1-4) evolved by absolute pitch Mozart Bi-gram fitness function and single point swap operator.

periments are smooth when balanced by ascending and descending pitch arrangement. See Figure 4.18 and 4.19. In the result summary that was shown in Figure 4.14, the convergence of operator 10 is in generation 0 to 1000, it takes longer than the operators 6 and 7. This is because operator 10 modifies the melody only two pitches at a time (swapping), operators 6 and 7 modify from a range of 2 to 10 pitches at a time while sorting the segment in a musical way. Operator 10 has more potential of searching out higher fitness melodies than other operators. Adding a small change to the high fitness melody may hit an even higher fitness more easily than the melody with a large change added, but the trade off is slow convergence. However, there is one potential problem of using this operator over an evolutionary system that uses single chromosome in the population. For example, if the optimal melody is X = [37,32,34,22,73] and the second highest fitness melody is Y = [73,22,34,32,37], then there is no way to evolve melody Y to melody X by using the single point swap operator. Because in order to evolve melody Y to X, two simultaneous applications of the swap operator are required as [73] swaps [37] and [22] swaps [32].

Operator 11 (Notes Shuffle) - This operator slightly outperforms operators 6 and 7 in experiment 5a, but outstandingly outperforms operators 6 and 7

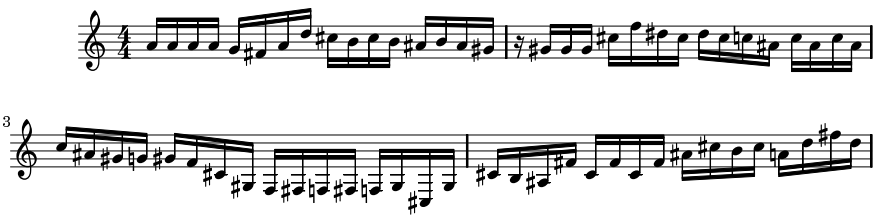


Figure 4.19: Example melody (Bar 1-4) evolved by pitch difference Mozart Bi-gram fitness function and single point swap operator.

in experiment 5b. The evolved melodies produced using this operator have smooth ascending and descending pitch arrangements of melodic shape. The convergence of this operator is lower than operators 3 and 10 because when an evolving melody reached a high fitness, it randomly shuffles the pitches within the melody and is very hard to evolve it to hit an even higher fitness value, especially when the chosen segment length is large.

Finally we conclude that the overall performance of all operators is good in terms of smoothing the shape of a melody. Each operator is able to smooth out the evolved melody and rearrang the pitch ordering in such a way that it contains fewer large interval gaps between two successive pitches. The overall shape of the melody is balanced by ascending and descending pitch arrangement, which is more structured and patterned, hence, more joyful to hear.

Limitation of N-gram fitness function

A major issue when using the N-gram fitness function to directly evolve a musical sequence is the MLS repetitive problem. This is probably the main reason N-gram model is often used as a generator for melody generation. As

we have seen, in experiments 2 to 4, evolved sequences which are not optimal are much more interesting and sound more pleasant than the MLS. In order to evolve sequences that have a middle range of note variations and a variety of patterns, the EA system must avoid the evolving sequences converging to MLS, instead the neighbours of MLS are the target sequences. According to our experimental results, we attempted to estimate the characteristic of musically interesting sequences in terms of notes variation. We suggest that if a sequence has too few or too many varied notes, then it is too extreme at either end, such a sequence is more likely to be less interesting to hear. A musically interesting sequence should be somewhere in the middle of the two extremes, with enough varied notes to create various sequence patterns with few repetitions.

4.6 Hypotheses Verification

Recap the hypothesis 1 described in section 1.3.3 is "The direct approach to music generation with a trained N-gram fitness function will produce pleasing music." The hypothesis 1 is refuted, the claim is based on the experimental results of experiments 2, 3 and 4. The results show that when using a singular N-gram model as a fitness function to directly evolve musical sequences and its evolved sequence reaching the optimal fitness, the sequence will contain highly repetitive notes. The MLS generated by N-gram model is repetitive as its contains a sequence of notes with the highest probability. The music evolved by the direct approach is clearly not pleasing to ear, therefore no further experiment and survey are conducted.

4.7 Chapter Summary

This chapter provided a fundamental investigation of using N-gram models as trainable music evaluators (fitness functions) in evolutionary algorithms. Experiment 1 and experiment 4 (part 1) demonstrated that N-gram classifiers were able to correctly identify the composer around 80% of the time. The most optimistic hope was that N-gram classifiers would enable the evolution of new pleasant sounding melodies, similar to those composed by some of the great composers.

In experiment 2, an interesting result here is that while both randomly generated sequences and the maximum likelihood sequence are uninteresting, evolutionary algorithms can be used to explore the search space between those extremes and find melodies that are reasonably interesting to listen to.

For experiment 3 we analysed each musical operator in detail, showed their performance on both Bi-gram and Uni-gram models. There was no single operator that could evolve a sequence that reaches the MLS with Bi-gram model. Most of the evolved sequences are in the range of fitness Antilog -200 to Antilog -500. As mentioned before, given the N-gram model, the creation of interesting melodies relies on the inability of the algorithms to converge to the optimum.

In experiment 4, the experimental results showed that using the pitch difference Bi-gram model for melody generation and evolution subjectively outperformed the absolute pitch Bi-gram model, even though it is no better at classification. We suggest the reason for this is that pitch difference statistics are closer to a perceptual model than using absolute pitch statistics. This is

intuitively obvious, since melodies played in a different key sound essentially the same, and have identical likelihoods in the pitch difference model, but completely different ones in the absolute pitch model. Also we found that training an absolute pitch N-gram model on melodies from different musical scales or keys can result in evolved melodies that are not well harmonised. This is because some notes outside a particular scale sound odd in the melody.

In experiment 5, the 'Bag of notes' constraint is imposed by starting with a bag of notes that are randomly ordered, and then applying permutation-based mutation operators that work by re-ordering the sequence to maximise fitness. The experimental results showed that the 'Bag of notes' constraint is not only able to effectively solve the excessive notes repetition problem; it also provides a new opportunity for evolutionary music. It is possible to re-generate the pitch arrangement of an existing melody towards a particular musical style e.g. To re-generate a Beethoven melody in the style of Mozart. This can be done by inserting a Beethoven melody as an initial sequence in the system and then evolve it by using a Mozart N-gram fitness function.

At the end of the experiment 4, we suggested that pitch difference representation of the Bi-gram model for melody evolution subjectively outperforms the absolute pitch representation. It is general enough to generate variety in the pitches with high fitness. However, in order to control the overall melodic shape of the evolved melodies with the 'Bag of notes' constraint applied, we suggest that when using absolute pitch representation with an appropriate operator (e.g. ascending or descending operator) to evolve melodies, the output melodies are more controllable and predicable than using pitch difference representation. The reason for this suggestion is based on the analysis

in experiment 5 and our observation.

Chapter 5

Evolving Music using Cellular Automata as A Generative Model

One major challenge for machine-learning based music composition systems is to generate music that has musically meaningful structures both locally and globally. In the literature, most research is focused on building systems that use some kind of machine learning techniques to capture the patterns of the given samples, and then the system generates similar or new music using the trained model. None of them have reported that their systems are able to generate music that has both local and global musical structures. It is often reported that the generated music has local structure only. As we mentioned in section 2.8, using the N-gram model to capture sequence, it is only able to capture the sequence structure in short scale rather than long scale. This makes the N-gram model difficult to generate a musical structure such as

musical form AABA that was described in section 1.3.4. We encountered this problem in experiments 2 to 5 in chapter 4. The experimental results showed that Musical Sequence Evolver is unable to achieve the goal of generating a global structure of music pattern.

This chapter explores indirectly evolved musical sequences with cellular automata and investigates using this approach for evolutionary music to achieve the goal of generating some musical sequences with some degrees of global structure. The system used in the experiments is 'Cellular Automata Music Evolver' that was described in section 3.2.7. The cellular automata music evolver uses a genetic algorithm to evolve a population of cellular automata transitional rules. At each evaluation, a space-time pattern is constructed by using the evolving transitional rule feeds into a cellular automata model. Then a music mapping algorithm is used to render the space-time pattern into music. The fitness of transitional rule is assigned by the fitness function that evaluates the rendered music.

This chapter is organised as follows, four experiments were conducted: First, we validate whether Zipf's Law based metrics are suitable to act as evaluators for objectively evaluating musical sequences. Towards this aim, we conducted experiment 6 (section 5.1) to investigate and analyse whether there are any Zipf's Law properties within the well-known musical pieces. In experiment 7 (section 5.2), we observed the impact of using NAP-ZLaw fitness function to evolve a simple CA that has 3 neighbourhood (1 neighbour radius), which would then generate monophonic melodies. After, in experiment 8 (section 5.3) we investigated how the proposed four fitness functions

[1] perform in the system with a complex CA that has 5 neighbourhood (2 neighbour radius). The results in experiment 8 showed that N-gram and N-gram Histogram fitness functions are in conflict with each other. Therefore, in experiment 9 (section 5.4), we further investigate these fitness functions with a multi-objective evolutionary algorithm (MOEA) techniques for music generation. Finally 5.5 is the chapter summary.

5.1 Experiment 6: Zipf's Law Music Analysis

In this experiment a total of 282 classical music samples [2] are analysed and investigated to see whether there are any Zipf's Law properties within each sample by using a set of Zipf's Law based metrics. Since the focus of this thesis is on pitch representation, these metrics are designed with more of an emphasis on the pitch statistic of the sequences. The definitions of these metrics are shown in Listing 5.1.

Listing 5.1: Zipf's Law metrics

```
Absolute Pitch – An integer that represents pitch, range from 0 to
127.

Absolute Duration – A double value that represents note duration
i.e. 0.5 represents a eighth note duration.

Pitch Difference – An integer that represents the interval between
two successive absolute pitches.
```

[1] Full details of the fitness functions are described in section 3.3
[2] Use the same set of pre-processed samples from experiment 1

Absolute Pitch and Duration – Combined absolute pitch and absolute duration i.e. (60,0.5) represents middle C with eighth note duration.

Pitch Difference and Duration – Combined pitch difference and absolute duration.

Twelve Chromatic Notes – There are 12 musical notes (Twelve Chromatic Notes: C, C#, D, D#, E, F, F#, G, G#, A, A#, B). Pitch is represented in a musical note.

Twelve Chromatic Notes and Duration – Combined twelve chromatic notes and absolute duration.

Bi–gram Absolute Pitch – Sequence is extracted into sequence of Bi–gram windows in absolute pitch representation.

3–gram Absolute Pitch – Sequence is extracted into sequence of 3–gram windows in absolute pitch representation.

4–gram Absolute Pitch – Sequence is extracted into sequence of 4–gram windows in absolute pitch representation.

Bi–gram Absolute Duration – Sequence is extracted into sequence of Bi–gram windows in absolute duration representation.

3–gram Absolute Duration – Sequence is extracted into sequence of 3–gram windows in absolute duration representation.

Remembering that, for a perfect Zipf's Law, the ideal log-log slope and R-squared values are -1 and 1. There is no critical region defined for the decision to accept or reject the data that is considered to have a Zipf's Law property when the given log-log slope and R-squared values are not equal to -1 and 1. Thus, in here we set the critical region for Zipf's Law property, if the given log-log slope value is between -0.8 to -1.2 and R-Squared values are higher than or equal to 0.7, then the given data is considered to have a Zipf's Law property.

Experimental Results

Table 5.1 shows the summary of the experimental results. The percentage value represents the percentage of musical pieces that are considered to have Zipf's Law property. Metrics marked with '*' in the table indicate that there are strong Zipf's Law properties in the samples.

The results showed that for each composer, more than 75% of their musical pieces have the Zipf's Law property of Bi-gram absolute pitch and it is the strongest property among others. Also Mozart samples have high average pass rates on Zipf's Law metrics 'Absolute Pitch and Duration', 'Pitch Difference And Duration', and 'Twelve Chromatic Notes and Duration', but the rates are low for Beethoven and Chopin samples. The Zipf's Law properties of the '3-gram Absolute Pitch', '4-gram Absolute Pitch' and '3-gram Absolute Pitch And Duration' metrics have a very low percentage because if N parameter increases, it will introduce many symbols with a low frequency and some of these frequencies will be ties. When most of the N-gram windows have low frequencies, it skews the log-log slope far away from -1.

Zipf's Law Metrics	Average pass rate		
	Mozart	Beethoven	Chopin
Absolute Pitch	11%	4%	19%
Absolute Duration	10%	4%	17%
Pitch Difference	5%	0%	8%
* Absolute Pitch and Duration	72%	70%	22%
* Pitch Difference And Duration	68%	30%	28%
Twelve Chromatic Notes	10%	32%	35%
* Twelve Chromatic Notes and Duration	65%	18%	28%
* Bi-gram Absolute Pitch	78%	93%	81%
3-gram Absolute Pitch	14%	16%	4%
4-gram Absolute Pitch	0%	0%	0%
Bi-gram Absolute Pitch And Duration	0.9%	9%	5%
3-gram Absolute Pitch And Duration	0%	0%	0%

Table 5.1: Summary of Zipf's Law experimental results

According to the Zipf's Law analysis results, using the Bi-gram absolute pitch metric as a music evaluator is convincing. We suggest that a Zipf's Law based metric provides an objective way to evaluate the level of pleasantness of piece of music. It indicates whether the music is too monotonic or too chaotic. However, the metrics only evaluate the global statistic of the musical elements. If the metric does not involve any note ordering capture e.g. for Uni-gram absolute pitch, the arrangements of the musical elements do not affect its Zipf's Law property. In other words, even if the notes in a melody are arranged similarly in sequence[3], which makes the melody sound monotonous, the plotted log-log slope would still be -1 and the frequency of pitches would follow Zipf's Law perfectly. This means that a melody that has a Zipf's Law property may not sound pleasant if the notes are not arranged in an interesting way.

[3]A sequence where the C note is played 10 times, the D note is played 5 times, and the E note is played 3 times sequentially.

5.2 Experiment 7 (Preliminary): Evolutionary Elementary Cellular Automata for Monophonic Music

This experiment provides an insight into the cellular automata model used in an evolutionary music system. It demonstrated how the Cellular Automata Music Evolver works in practice. Also we investigated how the evolved sequences sound. For the CA model, the type of CA model is one-dimensional CA with 3 neighbourhood (1 neighbour radius). For the EA part, a standard GA is used and the fitness function is similar to the NAP-ZLaw with N=1(Uni-gram), the fitness function evaluates the error distance between the sequence's Zipf's Law slope and the perfect Zipf's Law slope -1. Note that the R-squared value is ignored here. The EA is solving a minimising problem, searching the best solution with minimal error.

This simple experiment setting allows us to observe the system without altering the parameters too much. The summary of system settings is shown in Table 5.2. The music mapping algorithm used here is SBRB as described in section 3.2.7. The settings for this algorithm are based on the author's experience and preferences; we have chosen a C Minor pentatonic scale for this algorithm, this scale is mostly to be used in modern rock and Asian music.

The solution search space in this problem is small, given an one-dimensional CA and its initial configuration is fixed. It has total 256 rules with 256 patterns. Due to this reason, the optimal transitional rule appears in the initial

GA parameters	
Chromosome	8 bits binary string
Population size	100
Generation size	100
Crossover rate	0.8
Mutation rate	0.1
Fitness function	NAP-ZLaw with N=1 (Uni-gram)
CA parameters	
Neighbour radius	1
Lattice size	149
Generation size	500
Initial configuration	Middle cell is set to 1 (alive) and the rests are set to 0 (dead)
Music mapping algorithm	• Scale Based Row to Binary (SBRB): Settings is shown in Table 5.3. • Fixed Note Duration (FND): Each duration is set to a Eighth note.
Sampling strip	'Middle 7-bit', see section 3.2.7 for more details.

Table 5.2: Experiment 7 - experiment settings

SBRB settings	
Musical scale	Minor Pentatonic
Maximum octave	3
Start pitch	48 (C3 in MIDI)
Total number of pitches	16

Table 5.3: SBRB settings in experiment 7

population. The optimal transitional rule is "10010110" or called Rule 150 under the Wolfram's rule scheme. It has a Zipf's Law log-log slope value of -1.12 and an R-Squared value of 0.711. The fitness is 0.76, given the maximum allowable error is 0.5. However, most of the initial individuals have log-log slope ranging from -4 to -6, which gives the fitness of 0. Additionally, we computed all 256 transitional rules and found that the highest fitness of an individual is the Rule 150. Figure 5.1 shows a space-time pattern that was generated by the Rule 150. After the music mapping process, the melody is rendered. The melody contains 500 notes across 63 bars [4] of music. There are particular groups of notes repeated in the melody. Figures 5.2 - 5.4 show some of the interesting segments that were found in the melody. Segment A (Figure 5.2) is repeated 10 times. Segment B (Figure 5.3) is repeated 5 times. Segment C (Figure 5.4) is played only once. The melody also contains other segments, but all these segments only occurred once or twice in the melody. The opening part of the melody begins by being mixed with segments A and B, and then near the middle part segment C is played once. Then the rest of the melody is mixed with segments A, B and other segments. We emphasise that the arrangement of these segments is well organised in a way that is similar to the structure of musical form. This makes the sequence very interesting to hear. The overall structure of the sequence is achieved by the recursion mechanism of the CA model given Rule 150. Arranging the structure of the melody without specific instructions to the system is not a trivial task. In chapter 4 experiments 2 to 5, we used the N-gram model as a fitness function to evolve musical sequence directly. Hardly any structural

[4]The time signature is 4 crotchets per bar

sequences were seen to have evolved there. The N-gram model has a very limited memory, e.g. Bi-gram only takes the last event into account. Subsequently, the evolved sequences have no musical structure and it sounds like there are no breaks.

5.3 Experiment 8: Evolutionary 5 Neighbourhood Cellular Automata with Various Fitness Functions

The aim of this experiment is to analyse the performance of each defined fitness function given the cellular automata music evolver with a 5 neighbourhood CA model (2 neighbours radius). The given CA model is capable of generating a total of $2^{2^5} = 4,294,967,296$ patterns for a single initial lattice configuration. Thus, the search space is huge. This experiment consists of two parts: Part 1 describes the analysis of how each fitness function and the weighted sum of two fitness functions affect the evolved sequences. Part 2 demonstrates the system is capable of evolving polyphonic music by using more than one fitness function.

Experiment Settings

Overall, the settings of the experiment parts 1 and 2 are similar to experiment 7, but some changes have been made: the CA model is replaced by a 5 neighbourhood CA; various fitness functions are used; the evaluation and fitness assignment procedure of each CA rule is changed; both monophonic

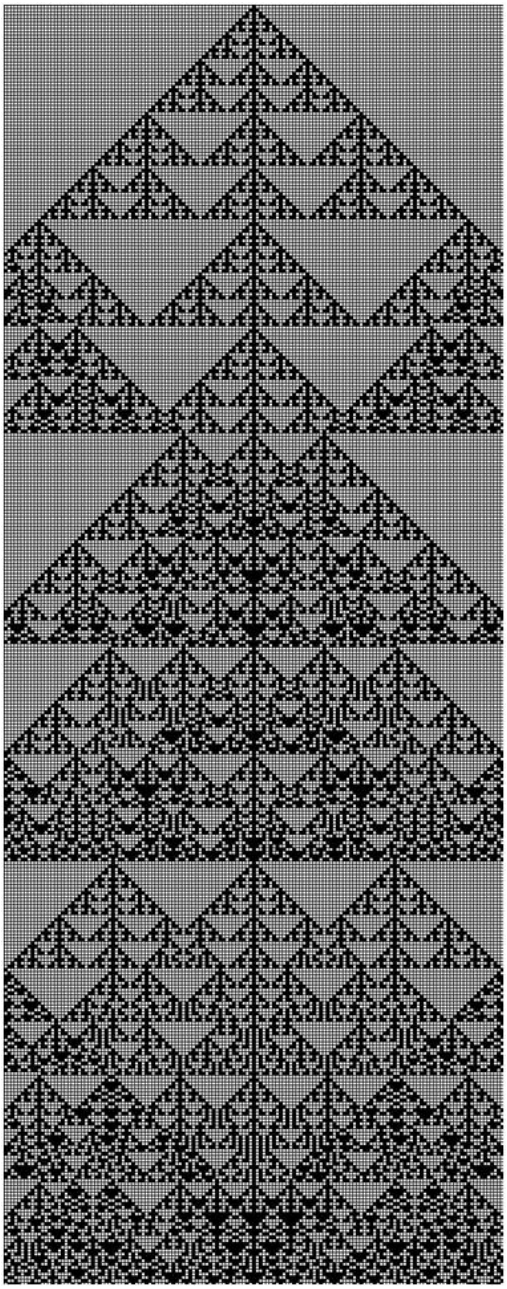


Figure 5.1: Optimal CA (Rule: 10010110) note: cell alive (black), cell dead (white).

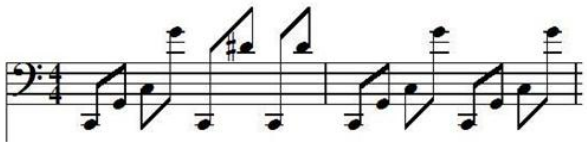


Figure 5.2: Melody segment A.

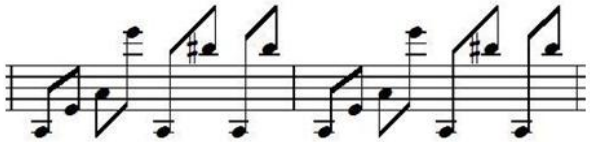


Figure 5.3: Melody segment B.

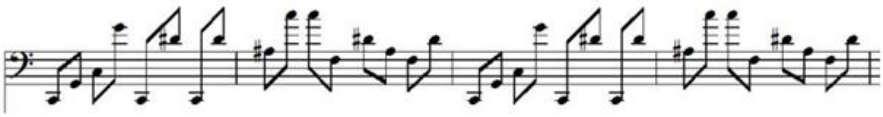


Figure 5.4: Melody segment C.

GA parameters	
Chromosome	32 bits binary string
Population size	100
Generation size	50
Crossover rate	0.8
Mutation rate	0.1
Fitness function	N-gram, NAP-ZLaw, NAP-Entropy and N-gram Histogram.
CA parameters	
Neighbour radius	2
Lattice size	149
Generation size	500
Initial configuration	Random
Music mapping algorithm	• Scale Based Row to Binary (SBRB): Settings is shown in Table 5.5. • Multiple Scale Based Row to Binary (MSBRB): Same as SBRB. • Grouping Notes Duration (GND): Settings is shown in Table 5.6.
Sampling strip	'Scan them all', see section 3.2.7 for more details.

Table 5.4: Experiment 8 - experiment settings.

and polyphonic music mapping algorithms are used; and the duration generation method 'Grouping Notes Duration' (GND) is used. Table 5.4 shows the summary of the experiment settings.

Figure 5.5 shows the evaluation and fitness assignment procedure for each CA rule at each evaluation: first, all possible strips are scanned in the generated space-time pattern, and only the highest fitness strip is returned. In

SBRB settings	
Musical scale	Minor Pentatonic
Maximum octave	4
Start pitch	36 (C2 in MIDI)
Total number of pitches	21

Table 5.5: SBRB settings in experiment 8

GND settings	
Group interval	3
Base note duration	Eighth note
Probability of Whole note	0.0
Probability of Half note	0.1
Probability of Quarter note	0.2
Probability of Eighth note	0.0
Probability of Sixteenth note	0.6
Probability of Thirty-second note	0.1

Table 5.6: Probability assignment of note duration for GND duration mapping.
Probabilities are predefined based on the author's preference.

this case, lattice size 149 has total of 143 strips per pattern. The 'Scan them all' method ensures searching for the highest fitness strip in the space-time pattern, unlike the 'Middle 7-bit' method which only samples a fixed location in the space-time pattern. This might cause any higher fitness strips to be missed. Second, each individual CA's rule will be evaluated 10 times. At each evaluation the CA is seeded by a randomly generated initial lattice. The final fitness of the individual is the average fitness of 10 evaluations. Averaging the fitness is to estimate the average performance of an individual CA rule over the different initial lattice. One remark worth making is that, using random initial configuration may cause an individual to have a different fitness in the next generation as the next randomly generated CA initial configurations might be different. This is similar to what is called the 'Noisy Fitness Function', when two consecutive fitness evaluations of the same individual yield a difference in their fitness, it will exploit the strength of the evolutionary algorithm if the fitness difference is too large.

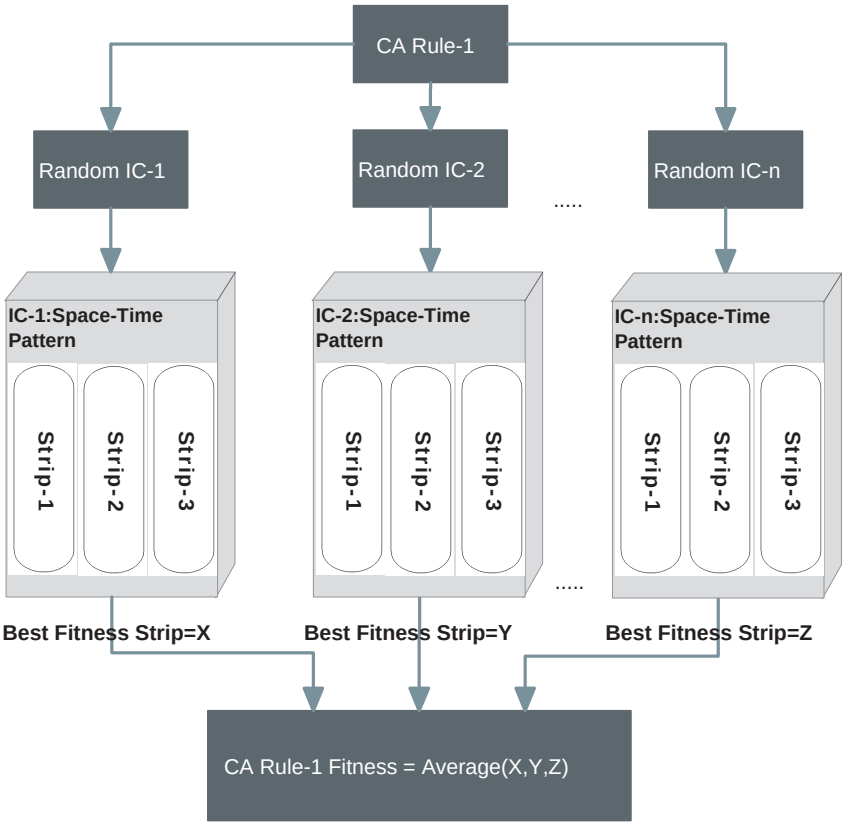


Figure 5.5: Evaluation and fitness assignment procedure for each CA rule.

5.3.1 Part 1: Fitness Functions Analysis

The aim of this section is to analyse the characteristic and convergent power of NAP-ZLaw, NAP-Entropy, N-gram and N-gram Histogram fitness functions. Individual fitness function settings and experimental results are described in the following:

N-gram Absolute Pitch Zipf's Law (NAP-ZLaw)

Three variations of NAP-ZLaw fitness functions were tested, Uni-gram (N=1), Bi-gram (N=2) and 3-gram (N=3) NAP-ZLaw fitness functions. Three fitness functions were individually run three times. Figure 5.6 shows the experimental results of the average fitness of individuals over generations for Uni-gram, Bi-gram and 3-gram NAP-ZLaw. The results showed that all evolved sequences have high fitness close to 1. This indicates that all evolved sequences have strong Zipf's Law properties, their log-log slope and R-Squared are close to -1 and 1. All the results in the Figure 5.6 shows convergence occurred early after 10 generations, in some runs, the first population contains individuals which have a fitness of more than 0.8. This is expected, as at each generation 1,430 individuals are evaluated, so there is a high chance of high fitness individuals being generated early on. Perhaps the NAP-ZLaw fitness function is easy to satisfy; GA has no problem searching for high fitness individuals in here.

Now, we examine the pitch statistic of some evolved sequences. Figures 5.7 - 5.9 show these evolved sequences in log-log plots. In the figures, each circle indicates a unique N-gram window(s) (single pitch or group of pitch

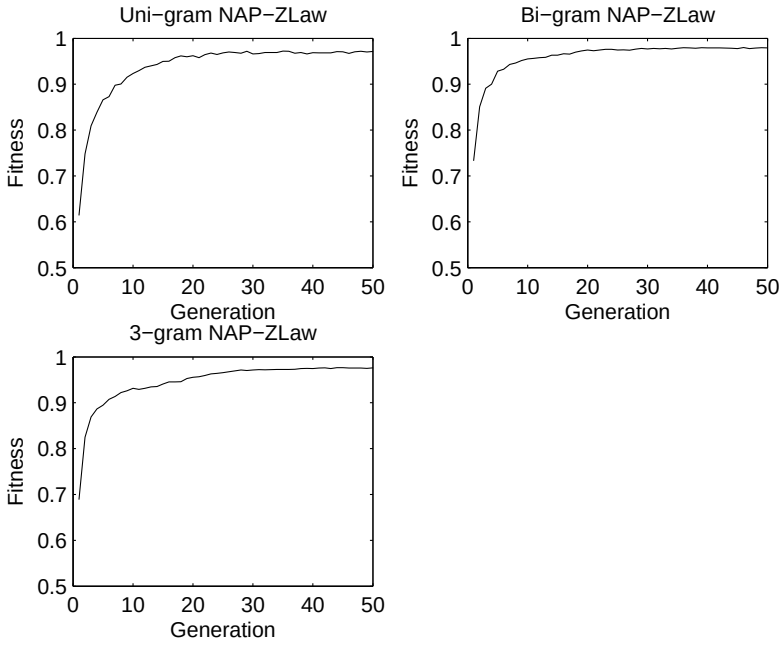


Figure 5.6: Average fitness of individuals given Uni-gram, Bi-gram and 3-gram NAP-ZLaw fitness functions.

if N >1). Based on the results, we found that if N increases in the N-gram model, then the convergence speed decreases. This is because there is less chance for a particular N-gram window in the generated melody to occur more than a few times. Instead there is a high chance of more N-gram windows having a low frequency with ties i.e. a frequency of 1. As the NAP-ZLaw fitness function is the composite of two fitness values (log-log slope and R-Squared), we noticed that the log-log slop fitness is easier to attain than the R-Squared fitness on the evolved sequences.

In order to give a close view of pitches distribution in a particular evolved sequence, in Figure 5.10, a histogram of pitch frequency of the melody 1gNAP-ZLawM0 is shown. Melody 1gNAP-ZLawM0 is evolved by Uni-gram NAP-ZLaw with fitness of 0.97. In the histogram, the number on top of each bar represents the absolute pitch. The X-axis and Y-axis indicate the corresponding pitch's rank and frequency respectively.

As we mentioned in chapter 3.3.2, a perfect Zipf's Law statistic can be obtained if a given melody consists of 100 C notes followed by 50 D notes (repeated-sequential pattern). Even though this melody has the highest Zipf's Law fitness, but it is too dull to listen to. Among the evolved sequences in the experiments, no such repeated-sequential patterns are found. The CA model acts as a constraint to prevent the rendered music containing a repeated-sequential pattern.

N-gram Absolute Pitch Entropy (NAP-Entropy)

The NAP-Entropy fitness function aims to guide the sequence to follow a uniform distribution where it hits the maximum entropy and the frequencies

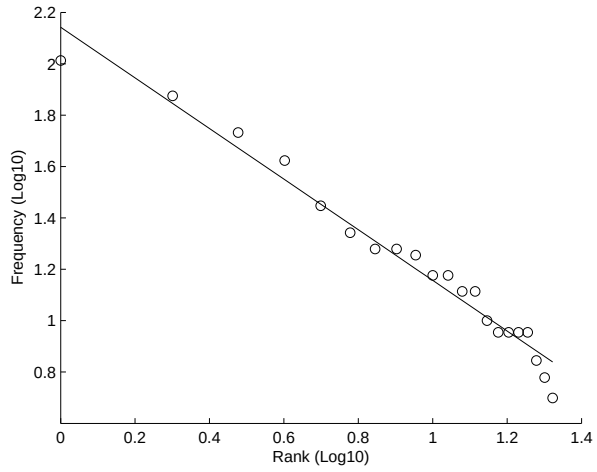


Figure 5.7: Melody: 1gNAP-ZLawM0, is evolved by Uni-gram NAP-ZLaw fitness function.

log-log slope = -0.998, R-Squared = 0.947 and NAP-ZLaw fitness = 0.97.

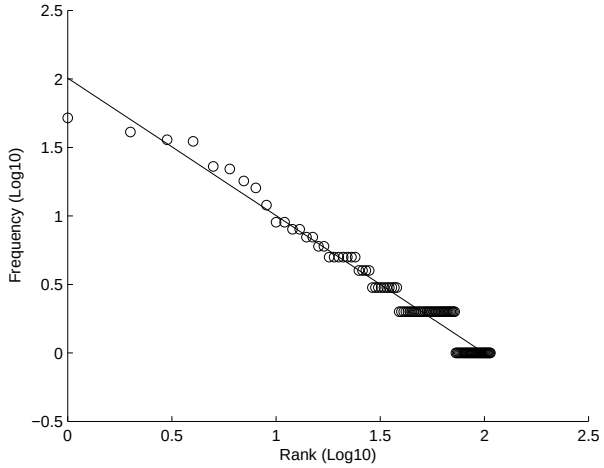


Figure 5.8: Melody: 2gNAP-ZLawM1, is evolved by Bi-gram NAP-ZLaw fitness function.

log-log slope = -1.003, R-Squared = 0.966 and NAP-ZLaw fitness = 0.98.

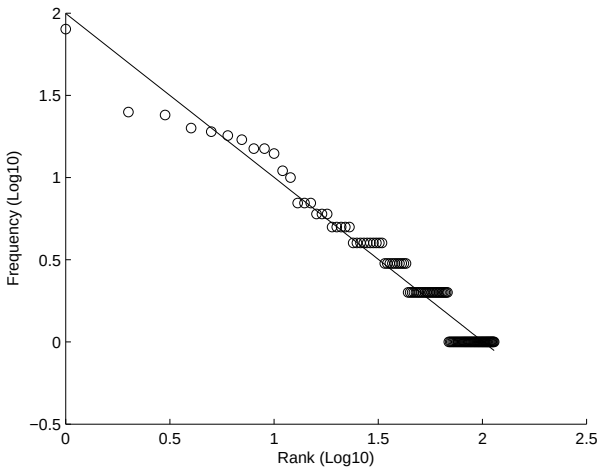


Figure 5.9: Melody: 3gNAP-ZLawM1, is evolved by 3-gram NAP-ZLaw fitness function.

log-log slope = -0.997, R-Squared = 0.963 and NAP-ZLaw fitness = 0.99.

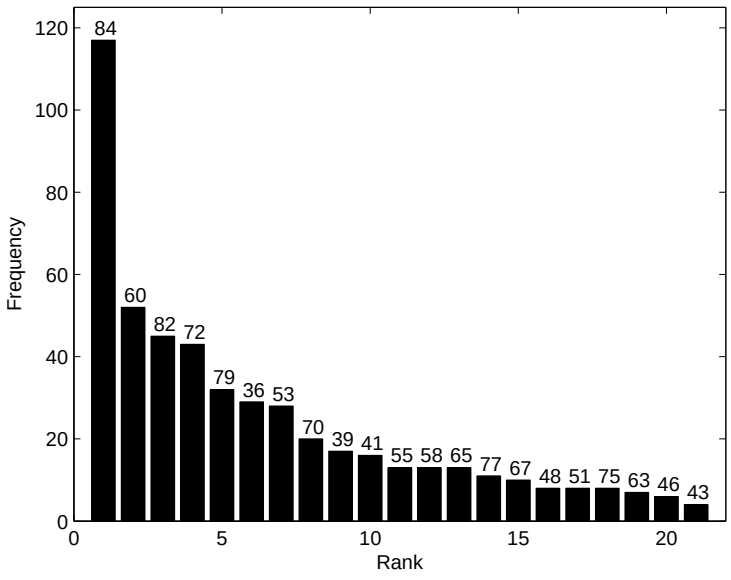


Figure 5.10: Melody: 1gNAP-ZLawM0. Pitch frequency in histogram.

of N-gram windows are the same. In this experiment, sequences are evolved using Uni-gram, Bi-gram and 3-gram NAP-Entropy fitness functions and each of them was run 10 times. Figure 5.11 shows the results summary of the experiment. The convergent power of Uni-gram is the highest and the best evolved sequence has a fitness of 0.998. For the Bi-gram and 3-gram, their best evolved sequences have a fitness of 0.89 and 0.672 respectively. Figure 5.12 shows the absolute pitch frequency of an evolved sequence (Melody: 1gNAP-EntropyM1) in a histogram. The sequence is evolved by using Uni-grams NAP-Entropy fitness function, and its fitness is 0.988. The pitch distribution of the sequence is not fully in a uniform distribution but it is close. The reason it fails to converge to the optimal sequence here is that the CA acts like a constraint that binds the search space, so the optimal sequence might not exist in the search space. Also the chance of having all the N-gram windows with the exact same frequency is low. An extreme case would be where the total number of possible N-gram windows for 3-gram is $21^3 = 9,261$. It is impossible to have a uniform distribution, when there are only 498 N-gram windows extracted from the evolved sequence that has 500 pitches.

The experimental results showed that when N increases in the fitness function, the optimal solution or high fitness solutions are much harder to reach. The reason is similar to the case of Melody: 1gNAP-EntropyM1.

We examined some of the evolved sequences, where the arrangement of pitches is not simply monotonic such as when a particular pitch is repeated 10 times, and then another pitch is repeated another 10 times and so on to fulfil the fitness function. Instead, some evolved sequences contain particu-

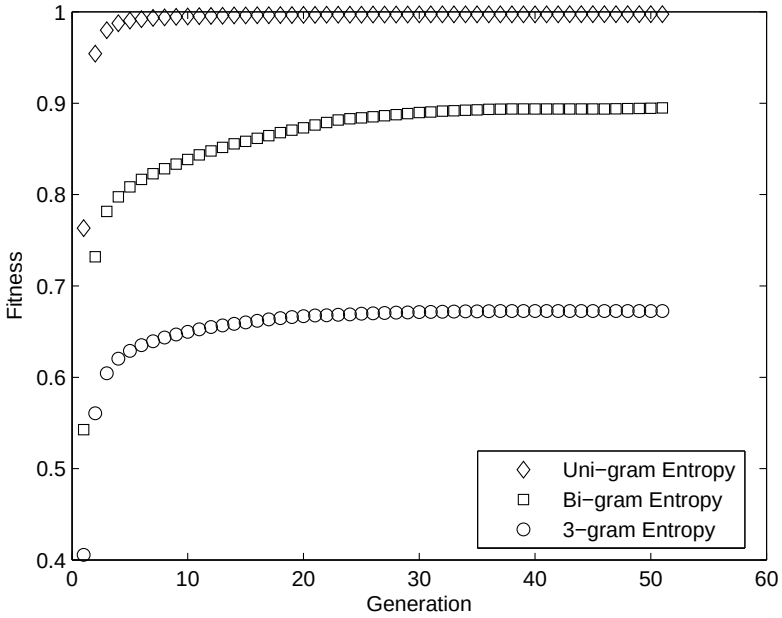


Figure 5.11: Average fitness. Each fitness function ran 10 times.

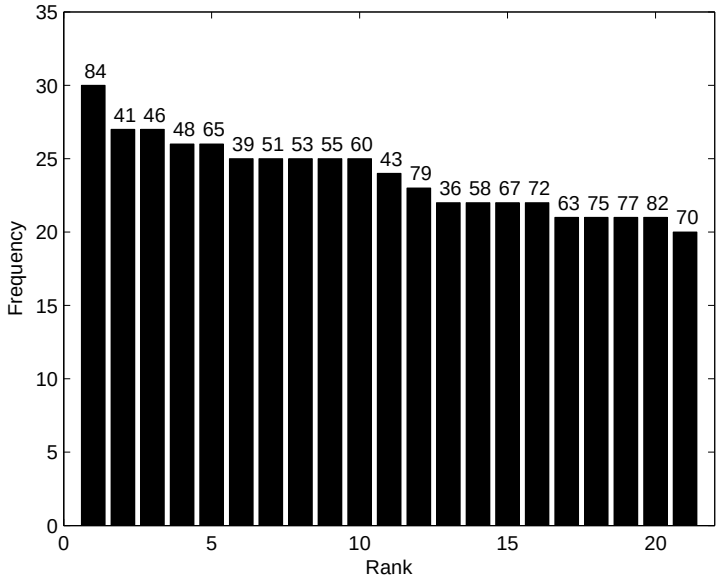


Figure 5.12: Melody: 1gNAP-EntropyM1. Pitch frequency in histogram.

lar groups of pitches repeated in the different locations within the sequence. These musical sequences sound structural in terms of musical phrase arrangement.

N-gram Fitness Function

As we have seen in chapter 4, the N-gram fitness function is a highly effective fitness function that searches for a maximum likelihood sequence, which is an optimal sequence that has a maximum probability of being generated by the given N-gram model. Often, the MLS consists of a single or group of repetitive pitches, it is not interesting to listen to. This problem is intrinsic to the N-gram model. Apart from the 'Bag of notes' constraint, we have also proposed a fitness function called N-gram Histogram (NH) that is meant to cooperate with the N-gram fitness function to address the MLS repetitive problem.

In this experiment, the performances of a standalone N-gram fitness function and a weighted sum fitness function that combines N-gram and N-gram Histogram fitness functions are analysed.

Standalone N-gram Fitness Function

Preliminary, in this section we investigated the performance of using either a standalone Uni-gram or a Bi-gram fitness function to evolve musical sequences indirectly via a CA model. Additionally, another purpose is to approximately determine the MLS given Uni-gram and Bi-gram models. The probabilities of these MLSs will be used for the fitness normalisation in the multi-objectives experiment later.

In the experiment, both N-gram fitness functions are trained using 110 Mozart pieces in absolute pitch representation. The rest of the experiment settings are described in 5.4. The experimental results showed that GA has no difficulty in finding the optima that satisfies these N-gram fitness functions. And the convergent powers of these fitness functions are high. We found that applying the 'Scan them all' method dramatically improved the chance of finding optimal solutions and reduced the length of generations. Figure 5.13 shows the optimal CA's space-time patterns that are evolved using Uni-grams and Bi-gram fitness functions. For Uni-gram pattern (a), after the music mapping the rendered sequence consists of pitch 79 being repeated and it has a fitness of Antilog -587.68. Note that in chapter 4 experiment 3, the MLS generated by the Uni-gram model is a sequence that repeated pitch 128. The result here is different, because the pitch set in the music mapping algorithm here is (36,39,41,43,46,48,51,53,55,58,60,63,65,67,70,72,75,77,79,82,84) the pitch 128 (Rest note) is not included. Additionally, we computed the absolute pitch frequency table of the training samples, see Appendix A.2. Among the pitches in the pitch set, pitch 79 occurred the most in the samples. Hence, EA found the MLS in this experiment. Again, it is an optimal sequence but not interesting to listen to. For the Bi-gram pattern (b), after the music mapping, the sequence is 39, 79, 77, 79, 77, 79, 77...repeat...79, 77 and it has a fitness of Antilog -413.7. This result is expected as this sequence is similar to the computed Bi-gram MLS in the previous chapter 4 - experiment 2. The reason for the difference between these Bi-gram MLSs is similar to the reason mentioned earlier.

A conclusion drawn from the results is that using the N-gram fitness func-

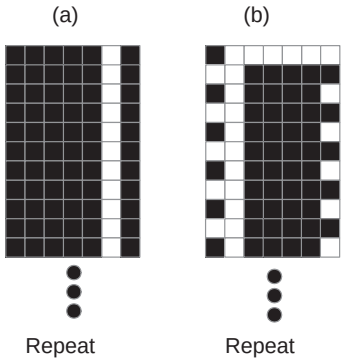


Figure 5.13: (a) Space-time pattern evolved using Uni-gram fitness function. (b) Space-time pattern evolved using Bi-gram fitness function.

tion without an appropriate constraint or musically meaningful operators to directly or indirectly evolve musical sequences is not advisable. The evolved sequences often consist of repetitive pitches and they are not pleasant to hear.

N-gram and N-gram Histogram Fitness Functions

This experiment aims to analyse the performance of using N-gram and N-gram Histogram (NH) fitness functions together to evolve musical sequences to resolve the MLS repetitive problem. The weighted-sum method is used for the fitness calculation. In this case, we set both fitness functions to be equally weighted, it is equivalently to saying that they are equally important to the evaluation. In order to sum up both fitness values returned by these fitness functions, normalisation is applied on both fitness function. It ensures that any fitness falls within the range of 0 to 1. As we have seen from the definition of the N-gram histogram fitness function in section 3.3.4, it always returns a fitness in the range of 0 to 1. Even though the N-gram fitness function

returns a probability which is a real number starting from 0, in practise, the probability of MLS is rarely equal to 1. Therefore, we use Equation 5.1 to normalise the N-gram fitness function that treats the MLS as the maximum probability instead of the optimal probability of 1.

$$N-gram\ fitness = 1 - \left(\frac{|P(MLS) - P(s)|}{Max.Err}\right), \\ if\ P(MLS) - P(s) > Max.Err,\ then\ N-gram\ fitness = 0. \tag{5.1}$$

Where $P(MLS)$ and $P(s)$ are the probabilities (in Antilog) of MLS and a given sequence respectively. $Max.Err$ is the maximum allowable error. In this case it is around Antilog -1000 for Uni-gram and Bi-gram models, although it does vary and depends on the N parameter.

In the experiments, various settings of NH fitness functions are tested. Table 5.7 shows the experimental results summary. In the table header, B indicates the number of weighted bins in the NH fitness function. The data in the table cell shows values for (n, g, h, a), where n is the N parameter in the NH fitness function, g and h are the best fitness returned from the Bi-gram and NH fitness functions respectively and a is the weighted-sum fitness. For a better representation of the results, figure 5.14 and 5.15 show the same experimental results summary in scatter graphs. Since this experiment was run many times with various settings, it is impossible to show all the evolved musical sequences in here. We only picked two examples of evolved sequences and printed them in standard musical notation in Figure 5.16 and 5.17. The rest of the evolved sequences (MIDI) are stored in the attached DVD-rom.

Bi-gram and NH ($B = 5$) fitness functions		
(1, 0.68, 0.93, 0.81)	(2, 0.72, 0.92, 0.82)	(3, 0.64, 0.93, 0.79)
(4, 0.71, 0.86, 0.79)	(5, 0.58, 0.90, 0.74)	(6, 0.64, 0.96, 0.8)
(7, 0.64, 0.93, 0.79)	(8, 0.66, 0.91, 0.78)	(9, 0.63, 0.91, 0.77)
(10, 0.64, 0.88, 0.76)	(11, 0.69, 0.84, 0.76)	(12, 0.59, 0.93, 0.76)
(13, 0.62, 0.88, 0.75)	(14, 0.64, 0.91, 0.77)	
Bi-gram and NH (B= 10) fitness functions		
(1, 0.54, 0.95, 0.75)	(2, 0.55, 0.8, 0.67)	(3, 0.48, 0.89, 0.69)
(4, 0.53, 0.65, 0.59)	(5, 0.45, 0.79, 0.62)	(6, 0.58, 0.71, 0.65)
(7, 0.62, 0.69, 0.65)	(8, 0.61, 0.74, 0.67)	(9, 0.67, 0.73, 0.7)
(10, 0.63, 0.66, 0.65)	(11 0.66, 0.67, 0.66)	(12, 0.67, 0.75, 0.71)
(13, 0.62, 0.79, 0.7)	(14, 0.64, 0.77, 0.7)	

Table 5.7: Experimental results summary of Bi-gram and N-gram Histogram fitness functions.

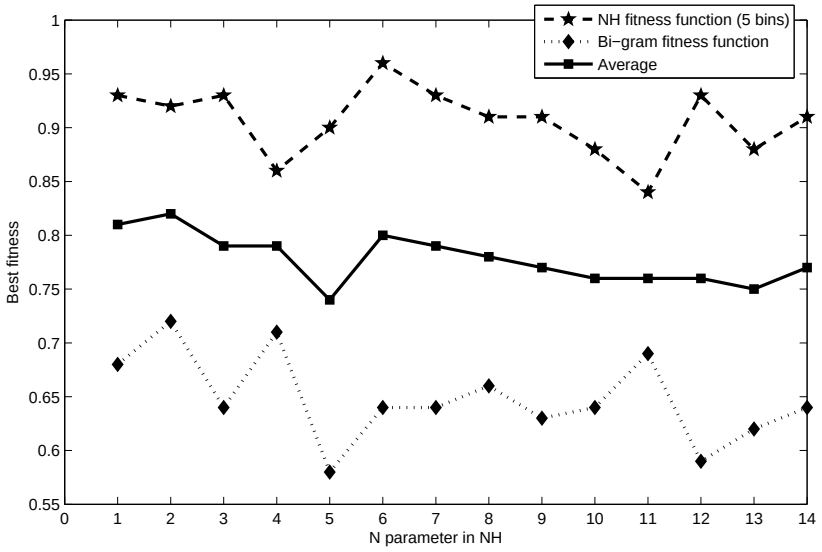


Figure 5.14: Experimental results summary of Bi-gram and NH (B=5) fitness functions.

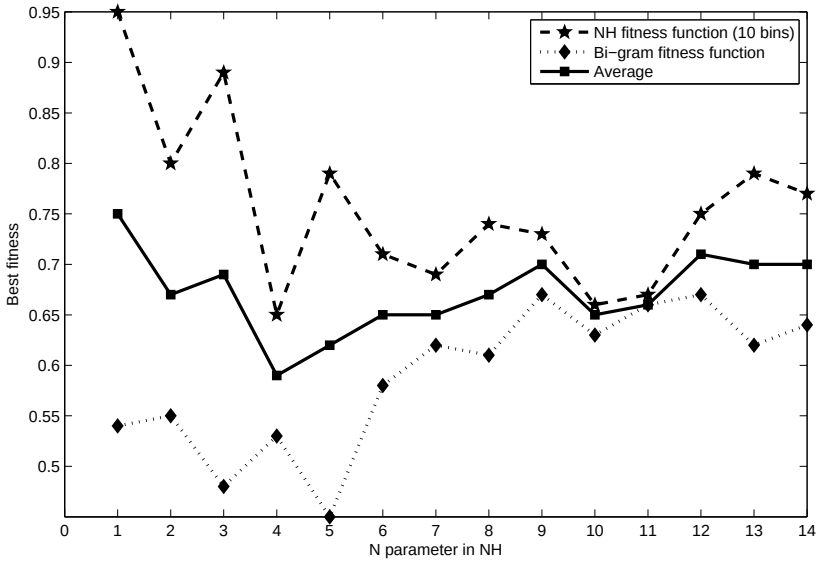


Figure 5.15: Experimental results summary of Bi-gram and NH (B=10) fitness functions.

Here, we perform a pitch statistic analysis over some evolved sequences. This analysis aims to analyse the distribution of repeated pitches that occurred in each sequence. This analysis is named 'repeated-group analysis', each analysis is done by counting the frequencies of unique N-gram windows that occurred in the sequences. Figure 5.18 illustrates the repeated-group analysis, the size of N-gram windows for this analysis is set to 3. In the figure, a set of 3-gram windows is extracted from the original musical sequence (top), their frequencies are then counted, the pitch groups (81,78,79) and (78,79,81) occurred twice and the rest of the pitch groups occurred once in the sequence. The bottom Figure 5.18 showed the analysis result. In the table, the header indicates the parameters of the NH fitness function, the data in the table cell is read as (x, y) where x is the frequency of each unique

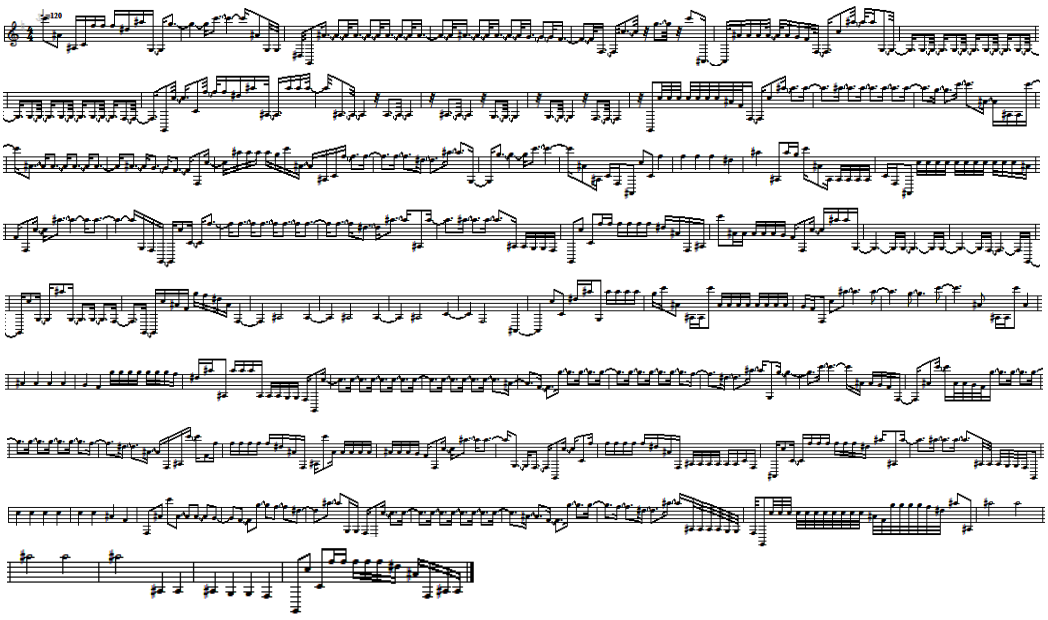


Figure 5.16: A sequence evolved by using Bi-gram and NH (B=5, N=2) fitness functions.

group of pitches and y is the number of unique groups of pitches that have a frequency of x in the sequence. Note that each column in a table represents a single evolved sequence. The analysis result in the table can be interpreted as follows: for the sequence that is evolved using Bi-gram and NH (B=10, N=2) fitness functions, there are three unique 3-pitch groups that occurred once and two unique 3-pitch groups that occurred twice.

For the analysis settings, the sizes for the N-gram windows are set to 1, 2, 5 and 10 (refer as 1-pitch, 2-pitches, 5-pitches and 10-pitches). The analysed sequences were evolved by Bi-gram and NH fitness functions, with the NH settings of B=10 and either $N = 1$, 2, 3, 5, 8 or 14. Table 5.8 shows the result of 1-pitch (single pitch) repeated-group analysis, many of the unique pitches occurred 10 to 50 times in the sequences, only a few pitches occurred more

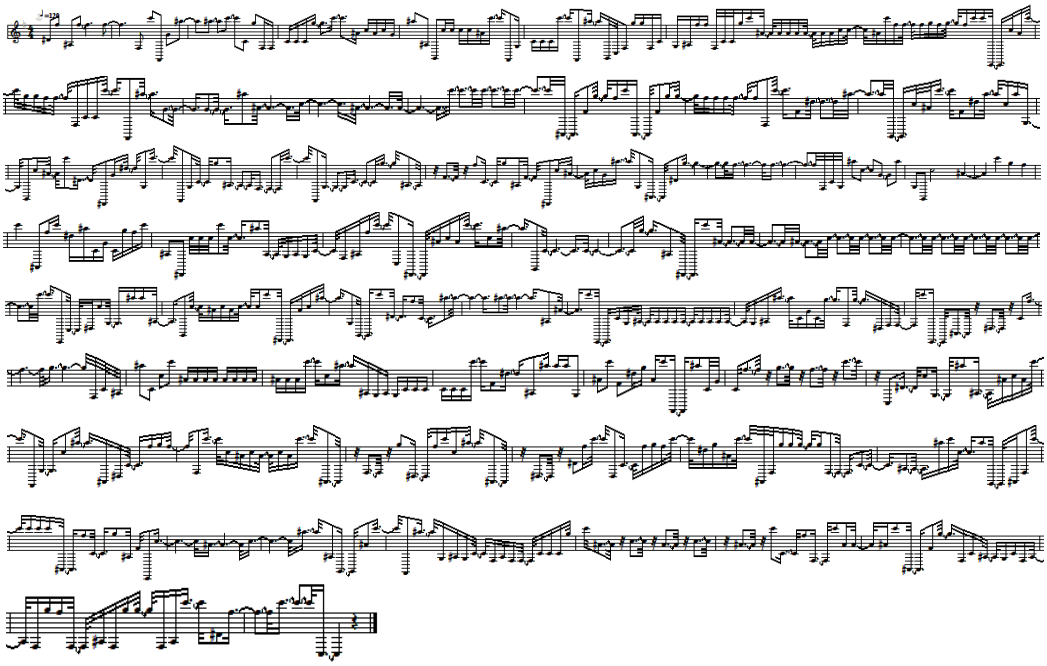


Figure 5.17: A sequence evolved by using Bi-gram and NH (B=10, N=2) fitness functions.

Musical Sequence: 81,78,79,81,78,79,81,69,71
(Absolute Pitch)

3-gram windows extraction: (81,78,79), (78,79,81), (79,81,78)
(81,78,79), (78,79,81), (79,81,69)
(81,69,71)

3-pitches groups: (81,78,79) is occured 2 times.
(78,79,81) is occured 2 times.
(79,81,78) , (79,81,69) and (81,69,71) are occured 1 time.

Result of 3-pitches repeated-group analysis

B=10, N=2
1, 3
2, 2

Figure 5.18: Example of 3-pitches repeated-group analysis.

than 50 times. For an extreme case, the first sequence (B=10, N=1) contains a pitch that was repeated 124 times. Table 5.9 shows the results of a 2-pitches repeated-group analysis, many unique 2-pitches groups occurred once or a few times in the sequences, the rest of the 2-pitches groups have frequencies between 6 and 40. Table 5.10 shows the results of a 5-pitches repeated-group analysis. The results are quite different to the results in Tables 5.8 and 5.9. Apart from the first sequence, more than 300 5-pitches groups occurred only once or a few times. None of the 5-pitches groups occurred more than 26 times. Table 5.11 shows the results of a 10-pitches repeated-group analysis, apart from the first sequence, for rest of the sequences, 90% of their 10-pitches groups only occurred once. This is expected as the possible combinations for 10 pitches are large; a particular group of 10 pitches is likely to occur once or a few times in a sequence that only contains 500 pitches.

In general, if the N parameter increases in the NH fitness function, then it increases the number of context pitches being captured in the model. It makes the evolved sequences tend to contain more repetitive pitches. We suggest that setting the N parameter in the NH fitness function to less than 5 is more likely to produce evolved sequences that contain a reasonable amount of repeated group of pitches. This is closer to the sweet point between the chaotic and static pitch statistic, and is more pleasant to listen to.

Based on the results and observation, using a NH fitness function to cooperate with the N-gram fitness function to evolve musical sequences, it is able to resolve the MLS repetitive problem and the results are convincing. However, we found three facts when using these fitness functions together to evolve musical sequences in our system:

B=10,N=1	B=10,N=2	B=10,N=3	B=10,N=5	B=10,N=8	B=10,N=14
1,1	5,2	2,1	9,1	5,2	4,2
2,2	6,1	5,1	10,1	7,2	8,1
9,2	7,2	11,1	11,1	11,1	11,1
10,1	8,1	13,1	12,1	12,1	13,1
11,1	9,1	14,1	15,1	13,1	15,1
13,1	10,3	15,3	16,1	17,1	17,2
15,1	12,1	19,1	17,1	19,1	18,1
17,1	23,1	22,1	19,1	20,2	20,1
18,1	30,1	26,3	20,1	22,1	23,1
21,1	34,1	27,1	21,3	26,1	26,1
27,1	37,1	30,2	25,1	28,2	28,1
28,1	38,1	36,1	27,2	29,1	29,1
30,1	39,1	39,2	29,1	32,1	32,1
31,1	47,1	43,1	30,1	39,1	33,1
36,1	50,1	47,1	32,1	42,1	34,1
46,1	55,1		35,1	52,1	36,2
50,1	58,1		39,1	66,1	38,1
124,1			64,1		58,1

Table 5.8: 1-pitch repeated-group analysis.

B=10, N=1	B=10, N=2	B=10, N=3	B=10, N=5	B=10, N=8	B=10, N=14
1,16	1,43	1,43	1,63	1,37	1,47
2,2	2,17	2,32	2,35	2,22	2,31
3,5	3,23	3,22	3,25	3,19	3,15
4,4	4,11	4,12	4,19	4,13	4,6
5,4	5,6	5,8	5,10	5,5	5,7
6,3	6,7	6,6	6,3	6,7	6,7
7,3	7,7	7,2	7,2	7,3	7,1
8,6	8,2	8,4	9,1	8,4	8,5
10,3	9,3	9,3	11,1	9,1	10,2
12,3	10,1	10,1	14,1	10,1	11,1
13,4	11,2	11,1	18,1	13,3	12,1
14,1	12,1	12,3	21,1	14,1	13,2
15,1	13,1	13,2	22,1	17,1	14,1
17,4	14,1	18,1	38,1	21,1	15,2
18,1	16,1	28,1		35,1	16,1
19,1	18,1			44,1	20,1
27,1	19,1				22,1
62,1	21,1				26,1

Table 5.9: 2-pitches repeated-group analysis.

B=10, N=1	B=10, N=2	B=10, N=3	B=10, N=5	B=10, N=8	B=10, N=14
1,102	1,404	1,395	1,395	1,322	1,329
2,54	2,34	2,29	2,16	2,33	2,34
3,32	3,3	3,3	3,5	3,10	3,10
4,18	5,1	4,6	4,1	4,2	5,2
5,9	10,1	5,2	5,3	5,4	8,1
6,6			7,1	6,1	9,1
7,4			8,1	18,1	10,1
9,1			9,1	26,1	15,1
			11,1		17,1

Table 5.10: 5-pitches repeated-group analysis.

B=10, N=1	B=10, N=2	B=10, N=3	B=10, N=5	B=10, N=8	B=10, N=14
1,278	1,485	1,483	1,491	1,446	1,461
2,72	2,3	2,4		2,13	2, 3
3,7				3,3	3, 1
4,12				10,1	4, 1
					5, 1
					12,1

Table 5.11: 10-pitches repeated-group analysis.

- It guides the evolving sequences to have a variety of pitches, whilst keeping the sequences which have high fitness. It encourages a number of repetitive groups of pitches to occur in the evolved sequences to make the sequences sound interesting.

- In Table 5.7, when the B parameter in the NH fitness function increased from 5 to 10, the average fitness of evolved sequences is decreased. This is because when the B parameter increased, it created more weighted bins in the histogram and subsequently the NH fitness function becomes more sensitive to the frequencies of N-gram windows that have occurred in the sequence.

- The NH fitness function effectively guides the evolving sequence away from the MLS; hence, there is a conflict between the N-gram and NH fitness functions, the weighted-sum approach never reaches the ideal fitness of 1.

5.3.2 Part 2: Evolving Polyphonic Music

This experiment demonstrates how to extend the CA music evolver to evolve polyphonic music (more than one note playing simultaneously) in a simple

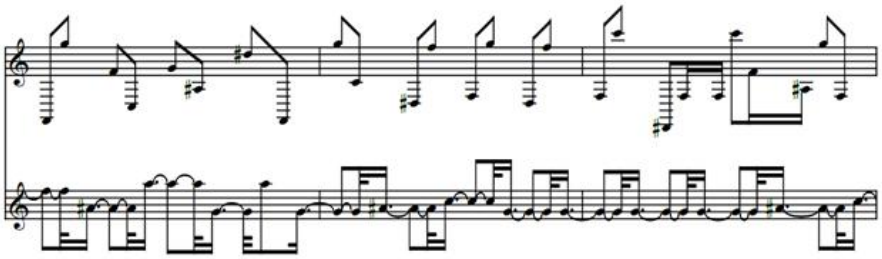


Figure 5.19: A segment of polyphonic music evolved by NAP-Entropy and NAP-ZLaw fitness functions.

way. The settings of this experiment are similar to experiments in part 1, except that a polyphonic music mapping algorithm MSBRB is used. Each individual CA returns two sampled strips from the generated space-time pattern, then two melodies are rendered to form a piece of polyphonic music. The fitness of an individual CA rule is an average fitness of both sequences that were evaluated using the correspondingly assigned fitness functions. It is not necessary to use the same fitness function for both sequences. In this case, NAP-Entropy and NAP-ZLaw fitness functions were used.

Figure 5.19 shows a segment sequence that was taken from the evolved polyphonic sequence. The top and bottom melodies are evolved using Bi-gram NAP-Entropy and Bi-gram NAP-ZLaw fitness functions respectively, and their fitness values are 0.78 and 0.93. The melodic structure of both melodies is strongly reflected its fitness functions.

During the evolution, there was no prior knowledge, constraints or rules for music harmonisation between two musical sequences. In the MSBRB music mapping algorithm, none of the pitches are outside the given scale. Therefore, two sequences are vertically harmonised in some way as the given scales are the same. However, if the two given musical scales are not the

same, to address the harmonisation problem, it is possible to embed rich harmonisation rules into a fitness function that punishes non-harmonised sequences, but this is beyond the scope of this thesis. More details of how to employ fitness functions to harmonise musical sequences can be found in [127] [155] [180] [162]. Some treatise of harmony are studied and suggested by musicologists [194] [195] [90] [177].

5.4 Experiment 9: Evolving CA Music using Multi-Objective Approach

In the experiment 8 part 1, the results showed that the N-gram and NH fitness functions are in conflict with each other. The N-gram model guides the evolving sequence towards MLS while the NH fitness function pulls the evolving sequence away from MLS. This is a common situation in Multi-objectives optimisation. Since these fitness functions are in conflict, the optimal (optimal fitness for both fitness functions) is unlikely to exist realistically. Hence, the natural way of extending the system is to substitute the weighted-sum approach with the multi-object evolutionary algorithm approach.

This experiment provided a fundamental investigation of using MOEA to evolve musical sequences, giving an insight from a multi-objective perspective. Amongst the latest MOEA algorithms, we have chosen NSGA-II [51] [52] and SPEA2 [227] algorithms in this experiment. Both of them are strong elitist driven algorithms for fast convergence and use a distance metric to diversify the Pareto front solutions. The diversity between solutions

is particular important for evolutionary music systems, simply we want the system to be able to evolve music with a difference. However, the total computational complexity of the fitness assignment for both algorithms is low i.e. NSGA-II = $O(GMN^2)$. Where G is the number of generations, M is the number of objectives and N is the population size.

The experiment settings is the same to experiment 8, the fitness functions are Bi-gram and NH fitness functions but the fitness normalisation method is changed from maximisation to minimisation. This change is for the convenience of using the existing source codes of NSGA-II and SPEA2 algorithms taken from an open source Java package: JMetal [60], which is an object-oriented Java-based framework aimed at facilitating the development, experimentation, and study of metaheuristics for solving multi-objective optimization problems.

Experimental Results

In the first part of the experiment, Bi-gram and NH (B=5, N=5) fitness functions were used. Figure 5.20 and 5.21 show the NSGA-II and SPEA2 experimental results. In the figures the fitness of both initial individuals (randomly generated) and Pareto front individuals are shown. There are many initial individuals that have the NH fitness value of 1, which is the worst. This indicates that the KL divergence between the individual and target NH model exceeds the maximum allowable error, which is set to 20 in this case. The Pareto front individuals that are located at the top-left corners are the MLSs. It has the maximum fitness for the Bi-gram fitness function, but the worst for the NH fitness function. In terms of the diversity

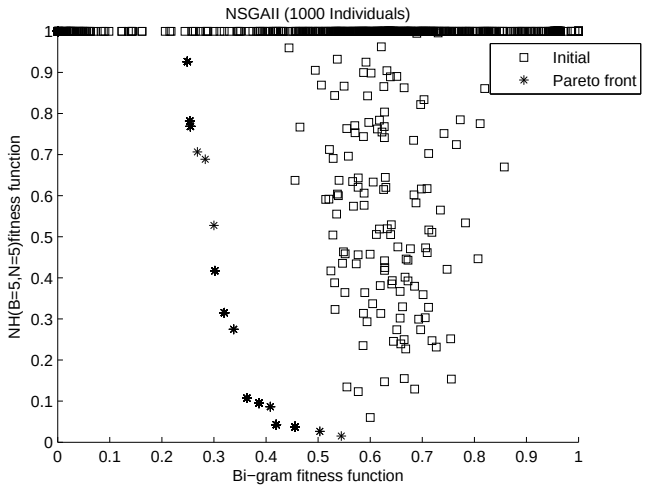


Figure 5.20: Pareto front solutions evolved by NSGA-II with Bi-gram and NH (B=5, N=5) fitness functions.

and optimising the finding of Pareto front solutions, there is no significant difference between NSGA-II and SPEA2 in the experiment. However, NSGA-II takes less computational time. This is because NSGA-II does not require an archive and measures each individual's dominator's strength.

Figure 5.22 shows an example of Pareto front music with Bi-gram and NH fitness of 0.455 and 0.038. This sequence contains balanced repetitive pitches and its pitch statistic is similar to the training samples. The N-gram fitness function maximises the local likelihood sequence, while the NH fitness function is responsible for keeping the global pitch statistic of the sequence similar to the training samples.

In the second part of the experiment, we ran three more experiments to test various settings of the NH fitness function using NSGA-II. Figures 5.23 - 5.25 show the experimental results, revealing that when N decreases in the NH fitness function, the average of the individuals' NH fitness decreased

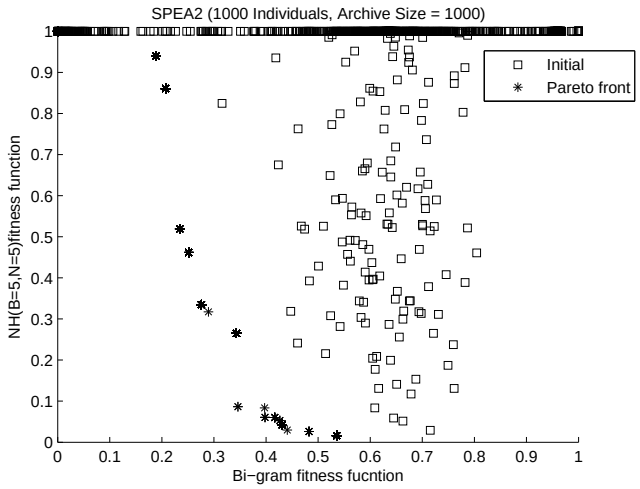


Figure 5.21: Pareto front solutions evolved by SPEA2 with Bi-gram and NH (B=5, N=5) fitness functions.

(better). Since the number of context pitches decreases, the model becomes more generalised, so the fitness function is easier to satisfy. This is similar to the B parameter; if B is increased then the model becomes more specific and sensitive to the frequency of the N-gram window, and the fitness function is harder to satisfy.

Choosing the music among the Pareto front music can be tricky. Low N-gram fitness (better) and/or low NH fitness (better) of sequences is less likely to be interesting, as low N-gram fitness sequences are close to MLS and low NH fitness sequences tend to have similar patterns to the training samples. From our observation for the Pareto front music evolved using Bi-gram and NH (B=5, N=5) with NSGA-II, the most pleasant music tends to have a Bi-gram fitness of between 0.2 and 0.5, and an NH fitness between 0.2 and 0.6. We suggest that if sequences are chosen which have middle range fitness for both fitness functions, it is more likely that the music will be interesting

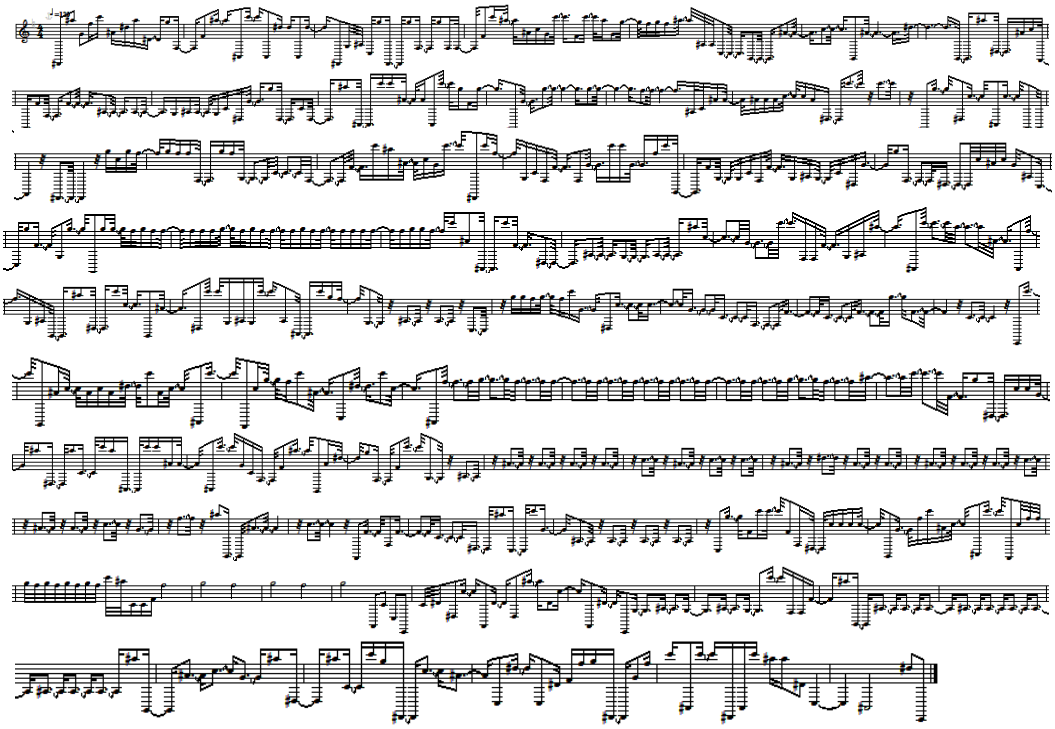


Figure 5.22: Melody 15 evolved by NSGA-II with Bi-gram and NH(B=5,N=5) fitness functions.

Bi-gram fitness =0.455 and NH fitness=0.038.

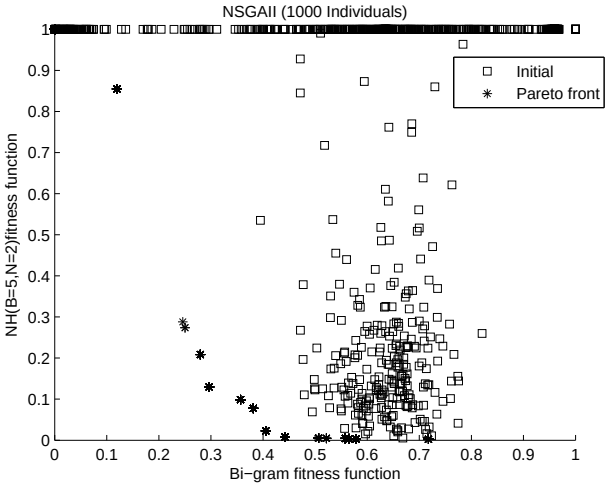


Figure 5.23: Pareto front solutions evolved by NSGA-II with Bi-gram and NH (B=5, N=2) fitness functions.

and these sequences will satisfy to some degree of both fitness functions.

5.5 Chapter Summary

This chapter provided a fundamental investigation of evolved musical sequences using an indirect approach and demonstrated its ability to evolve structured musical sequences.

Experiment 6 performed a Zipf's Law analysis that determines whether the music samples have some degree of Zipf's Law properties. The main idea behind this experiment is to search for the types of Zipf Law properties that commonly occurred in the samples, and these metrics might also be good as a fitness function for music evaluation. The results showed that the highest pass rate of Zipf's Law metrics is 'Bi-gram absolute pitch', which has an average of 84% pass rate for all music samples. We suggest that using Zipf's

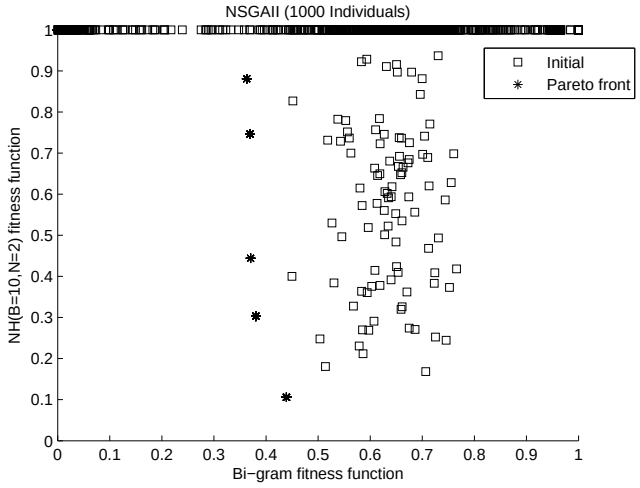


Figure 5.24: Pareto front solutions evolved by NSGA-II with Bi-gram and NH (B=10, N=2) fitness functions.

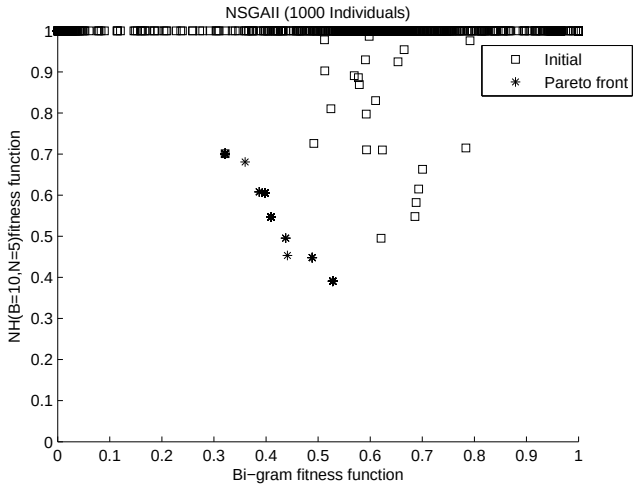


Figure 5.25: Pareto front solutions evolved by NSGA-II with Bi-gram and NH (B=10, N=5) fitness functions.

Law based fitness functions are suitable for evaluating the global statistic of the musical elements. The ordering of the pitches does not affect the metrics unless the metric takes the pitch ordering into account such as 'Bi-gram absolute pitch' metrics.

In experiment 7, the results showed that using a Uni-gram NAP-ZLaw fitness function without any constraint to evolve elementary 3 neighbourhood CA is able to generate structural musical sequences. The best generated musical sequence contains several repetitive phrases interspersed with other pitches, and the non-repetitive phrase acts as a chorus or the climax in the sequence. This pitch arrangement makes the sequence sound more interesting.

Experiments 8 and 9 provided further investigations of the performance of the cellular automata music evolver when given an elementary 5 neighbourhood CA model with the four defined fitness functions. In experiment 8, part 1, the performance of each fitness function is analysed and given some examples of evolved sequences. The experimental results showed that using a standalone N-gram fitness function with standard genetic operators to evolve musical sequences is not suitable for a music composition task. The optimal sequence is the MLS, but even though it satisfied the statistical requirement, it sounds monotonic and not pleasant to hear. Furthermore, we investigated the use of multiple fitness functions to address the MLS repetitive problem. The experimental results showed that using N-gram and NH fitness functions together is enables the evolving sequences to be guided away from the MLS while its global statistic of the pitch frequency is similar to the given N-gram histogram. The MLS repetitive problem is not a trivial

problem that is easy to solve, because it is hard to find sequences that are in the sweet spot between ordered and chaotic patterns. We believe that interesting sequences lie in the area near the MLS but not exactly. To evolve pleasant musical sequences, the N-gram fitness function must cooperate with other fitness functions or use a constraint that counteracts the MLS repetitive problem. Note that fitness functions NAP-ZLaw, NAP-Entropy, N-gram and N-gram Histogram guide evolved sequences to different directions. These fitness function are in conflict with each other, especially, the N-gram fitness function which is highly sensitive to pitch value and its ordering. Experiment 8 part 2 demonstrated that the system is capable of evolving polyphonic music in a simple way by combining all individual sequences to form a piece of polyphonic music and simultaneously playing all sequences that started at the same time. Due to the used music mapping algorithm constraints all the notes are taken from the same musical scale, simultaneous notes are harmonised with each other. Finally, experiment 9 provided a demonstration of using MOEA techniques to evolve musical sequences. This technique not only solved the MLS repetitive problem, it also allows the user to have an option in selecting their preferred musical sequences among the Pareto front musical sequences.

Overall, the experimental results showed that fitness functions guide the evolving sequences toward a desired local pitch arrangement and global pitch statistic, while the cellular automata model handles the global structures of the sequences. The CA model acts as a constraint to limit the pitches and their arrangement in the sequence, because the possible number of generated space-time patterns depends on the settings of the CA model. The pro-

posed CA music evolver is robust for evolving structured musical sequences. The structural complexity of the evolved musical sequences is dependent on the capability of the cellular automata model to produce complex patterns. Finally, we believe that future work in this direction is promising.

Chapter 6

Online Music Surveys and Fitness Distance Analysis

This chapter is divided into two parts: part one describes the detail of how we conducted the online music survey that allow us to evaluate the success of the proposed systems and receive some feedback on the evolved music from human listeners. The analysis results of the surveys are used for verifying hypotheses 2, 3, 4 and 5. Part two describes an experiment that aims to analyse the difficulty of evolving cellular automata using a genetic algorithm by Fitness Distance Correlation (FDC) [95].

6.1 Experiment 10: Online Music Surveys

The final phase of the framework for evolutionary music composition systems is a system evaluation that is performed by a human listening test suggested

in section 3.2.8. In this experiment, two online music surveys A[1] and B[2] are conducted. These surveys are designed to evaluate the satisfaction of the evolved music from the viewpoint of human listeners. It aims to test the claim that the music evolved by the proposed systems is pleasant to human listeners and the claim that human listeners will mistake the evolved music is composed by a human. The surveys are based on asking human subjects to rate our music samples in terms of how pleasant the given music is, in the range of 1 to 5 (1 being the worst and 5 being the best), and to indicate whether they believe the music to be composed by a machine or a human. The design of both surveys layouts are similar. A screenshot of survey B is shown in Figure 6.1. Both surveys were circulated by email in the Computer Science and Electronic Engineering department at University of Essex and invitation letters were sent to other researchers who are in the field of evolutionary music.

6.1.1 Music Surveys Results and Analysis

Music Survey A

Music survey A contains 15 music samples that are taken from three groups of music: 7 pieces are composed by humans; 5 pieces are the evolved music taken from experiments 5, 7, 8 and 9; and 3 pieces are generated by a random music generator. Table 6.1 shows the details of each music sample. A total of 40 candidates completed in this survey. Table 6.2 shows the result of each music sample, the table is sorted by mean rating. The result showed that

[1]Survey A was posted from November 2008 until January 2009.
[2]Survey B was posted from January 2009 until April 2009.

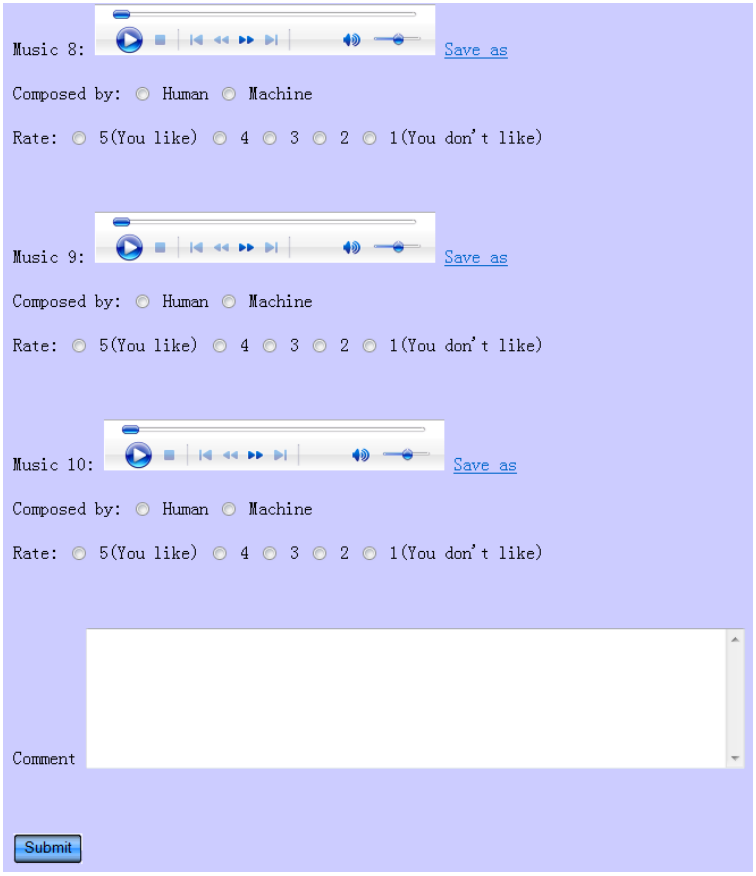


Figure 6.1: Survey B - screenshot.

in order of preference, candidates tend to give a higher rating to human-composed music, then system-evolved music, and lastly randomly-generated music.

Tables 6.3 and 6.4 shows summaries of candidates answers to the composer question. These table are sorted by mistake rate, which indicates how likely the listeners will mistake identify the correct composer (Human or Machine) for a piece of music. The mistake rate defined as: $Mistake\,Rate = \frac{No.\,of\,false\,answsered}{Total\,answsered}$ E.g. the mistake rate for Music 10 is 12/40=0.30. In the results, none of the music have a mistake rate bigger than 0.5, which means none of the music could misleading at least half of the listeners to give incorrect answers. Among the machine-generated music, music 10 (evolved music) has a mistake rate of 0.30, it is the one that comes closest to being able to deceive the candidates into thinking it is not generated by machine. Music 10 is the only polyphonic system-evolved music in music survey A. We suspect that polyphonic music might be an important feature that influences human choice when performing our human-machine classification task. This aspect will be investigated in survey B.

Music Survey B

The general layout of music survey B is similar to survey A, but a number of changes have been made to improve it to allow precise investigation some aspects and to eliminate some facts that could skew the results. The changes included:

- The number of music samples is changed to 10 in the music survey B

Music No.	Composed by	
Music 1	Machine	Random Generator
Music 2	Human	Chopin: Etude in Am, Op.10, No.2
Music 3	Human	Mozart: K333, 1778 Paris Sonata for Piano in B Flat Mvt. 13
Music 4	Machine (Experiment 7)	Evolved by Zipf Law fitness function with 3 neighbourhood CA
Music 5	Machine (Experiment 5)	Evolved by N-gram fitness function with "Bag of notes" constraint
Music 6	Machine	Random Generator
Music 7	Human	Beethoven: Piano Concerto No.3 in Cm, Op.37 Rondo Allegro
Music 8	Machine (Experiment 8)	Evolved by N-gram and NH fitness functions (weighted-sum)
Music 9	Human	Mozart: K331, 1788 Paris Sonata in A, Part 3 Alla Turca, Mvt. 2
Music 10	Machine (Experiment 8: Polyphonic Music)	Evolved by NAP-Entropy and NAP-ZLaw fitness functions (weighted-sum)
Music 11	Human	Chopin: Etude in C#m, Op.25, No.7
Music 12	Machine	Random Generator
Music 13	Human	Chopin: Mazurka in Bm, Op.30, No.2
Music 14	Machine (Experiment 9)	Evolved by NSGA-II with N-gram and NH fitness functions
Music 15	Human	Mozart: K332, 1778 Paris Sonata for Piano in F, Mvt. 2

Table 6.1: Music survey A - music samples.

Rank	Music	Mean Rating	Composed by	ANS-Human	ANS-Machine
1	Music 7	4.08	H	34	6
2	Music 3	4.00	H	36	4
3	Music 13	3.93	H	36	4
4	Music 9	3.86	H	37	3
5	Music 15	3.60	H	32	8
6	Music 2	3.28	H	24	16
7	Music 11	3.10	H	28	12
8	Music 10	2.45	M	12	28
9=	Music 8	2.23	M	6	34
9=	Music 14	2.23	M	9	31
10	Music 4	2.05	M	5	35
11	Music 1	1.93	RM	6	34
12	Music 6	1.80	RM	7	33
13	Music 12	1.68	RM	8	32
14	Music 5	1.63	M	5	35

Table 6.2: Survey A Result - ranking by mean rating (Total 40 candidates).
H=Human, M=Machine (system-evolved), RM=Randomly-generated (by machine), ANS-Human=No. of answered human and ANS-Machine=No. of answered machine.

Rank	Music	Composed by	Mistake Rate
1	Music 2	H	0.40
2	Music 11	H	0.30
3	Music 15	H	0.20
4	Music 7	H	0.15
5=	Music 3	H	0.10
5=	Music 13	H	0.10
6	Music 9	H	0.08

Table 6.3: Survey A Result - ranking by mistake rate (Only for human-composed music).

H=Human and Mistake Rate=$\frac{No.\ of\ answsered\ machine}{Total\ answsered}$.

Rank	Music	Composed by	Mistake Rate
1	Music 10	M	0.30
2	Music 14	M	0.23
3	Music 12	RM	0.20
4	Music 6	RM	0.18
5=	Music 8	M	0.15
5=	Music 1	RM	0.15
6=	Music 4	M	0.13
6=	Music 5	RM	0.13

Table 6.4: Survey A Result - ranking by mistake rate (Only for machine-generated music).

M=Machine (system-evolved), RM=Randomly-generated (by machine) and Mistake Rate=$\frac{No.\ of\ answsered\ human}{Total\ answsered}$.

and each sample is limited to around one minute long. The one minute long segment can be chosen from anywhere in the music. This changes because we observed some sessions of candidates taking music survey A. We found that candidates got bored or irritated if the length of music was over one or two minutes long. Some candidates even dropped the survey half way through. Also we found that candidates with some classical music background might have listened to some of the samples before taking the survey. Therefore, we chose some not so well-known but relevant classical music samples in music survey B.

- Music survey B consists of human-composed and system-evolved music. Therefore, the performances of our evolved music can be directly compared with the human-composed music.

- From our observation candidates tended to think that the note volume in human-composed music is usually more dynamic when expressing the emotions of the composer and to create intensive/non-intensive musical movements. For a fair test, all the music samples in music survey B are edited in such way that all pitches are played at a fixed volume.

Table 6.5 shows the details of each music sample in music survey B. A total of 31 candidates completed music survey B. Table 6.6 shows the mean rating of each music sample and, Table 6.7 and 6.8 show the summary of candidates' answers to the composer question. The complete details of the result are in Appendix B, including the candidates' comments.

The results showed that all the evolved music received lower mean rating than the lowest mean rating of human-composed music (Music 3). Also none

Music No.	Composed by	
Music 1	Machine	Evolved by CA music evolver. Bi-gram+NH(N=2,B=5)(Track1) and Bi-gram NAP-Entropy(Track2)
Music 2	Human	Burgmueller: The pearl
Music 3	Human	Chopin: 1839 Piano sonata in Bbm, Opus 35, No. 2, Mvt. 4
Music 4	Machine	Evolved by CA music evolver. Bi-gram NAP-Entropy(Track1) and Bi-gram NAP-ZLaw(Track2)
Music 5	Machine	Evolved by CA music evolver. Bi-gram+NH(N=5,B=5)(Track1) and Bi-gram NAP-Entropy(Track2)
Music 6	Human	Liszt: Grandes Etudes de Paganini, No. 4
Music 7	Human	Grieg: Butterfly
Music 8	Machine	Evolved by CA music evolver. Bi-gram NAP-Entropy(Track1) and Bi-gram NAP-ZLaw(Track2)
Music 9	Machine	Evolved by CA music evolver. Bi-gram+NH(N=5,B=5)(Track1) and Bi-gram NAP-Entropy(Track2)
Music 10	Human	Schubert: Impromptus Op90, No. 2

Table 6.5: Music survey B - music samples.
All machine music is evolved by CA evolver using various settings. (Taken from experiment 8: polyphonic music)

of the evolved music has a mistake rate more than 0.5.

6.1.2 Discussion of Music Surveys Results, Hypotheses and Observation

According to the results, candidates tend to give a higher rating to the human-composed music, then the system-evolved music, and lastly the randomly-generated music. This result is expected, otherwise if the system-evolved music received the highest rating on average, then the revolution of using machines to compose music as good as human-composed music would be complete. The evolved music in the music survey B performed better than the evolved music in music survey A. Most of the evolved music received mean rating higher than 2.3. And one of the evolved music has a higher mistake rate of 0.48, which is very close to half chance of the human listeners thought it to be composed by human.

Rank	Music	Mean Rating	Composed by	ANS-Human	ANS-Machine
1	Music 7	3.84	H	28	3
2	Music 10	3.77	H	24	7
3=	Music 2	3.35	H	15	16
3=	Music 6	3.35	H	24	7
4	Music 3	3.23	H	20	11
5	Music 5	2.71	M	15	16
6	Music 4	2.48	M	12	19
7	Music 1	2.39	M	9	22
8	Music 9	2.32	M	11	20
9	Music 8	2.23	M	6	25

Table 6.6: Survey B Result - ranking by mean rating (Total 31 candidates).
H=Human, M=Machine (system-evolved), ANS-Human=No. of answered human and ANS-Machine=No. of answered machine.

Rank	Music	Composed by	Mistake Rate
1	Music 2	H	0.52
2	Music 3	H	0.35
3=	Music 6	H	0.23
3=	Music 10	H	0.23
4	Music 7	H	0.09

Table 6.7: Survey B Result - ranking by mistake rate (Only for human-composed music).
H=Human and Mistake Rate=$\frac{No.\ of\ answsered\ machine}{Total\ answsered}$.

Rank	Music	Composed by	Mistake Rate
1	Music 5	M	0.48
2	Music 4	M	0.39
3	Music 9	M	0.36
4	Music 1	M	0.29
5	Music 8	M	0.19

Table 6.8: Survey B Result - ranking by mistake rate (Only for system-evolved music).
M=Machine (system-evolved) and Mistake Rate=$\frac{No.\ of\ answsered\ human}{Total\ answsered}$.

Hypotheses Verification

The followings are the conclusions of verifying the hypothesis 2, 3, 4 and 5 which are defined in section 1.3.3.

Hypothesis 2 "The direct approach to music generation with a trained N-gram fitness function using a "Bag of notes" constraint will produce pleasing music."

Hypothesis 2 is refuted in the result of survey A, the "Bag of notes" music received a lower rating than randomly generated sequences.

Hypothesis 3 "The indirect approach to music generation using a CA and trained N-gram fitness functions will produce pleasing music."

Hypothesis 3 is accepted, this is supported by result of the survey A, where the evolved music scored more highly than the random sequences. However, the performance of the CA music is not yet at the level of human generated music: in both survey A and survey B the human music had a significantly higher rating than the evolved CA/GA music.

Hypothesis 4 "Adding polyphony to evolved CA music results in more pleasing music being generated."

Hypothesis 4 is accepted, this is supported by survey A, where music 10 (evolved polyphony music) scores more highly than the other three CA/GA pieces, which are monophonic music.

Hypothesis 5 "The indirect approach to music generation using a CA and trained N-gram fitness functions with added polyphony will produce music which the listeners will mistake for human-composed music."

Hypothesis 5 is refuted in the result of survey B, none of the evolved music have mistake rate higher than 0.5. Although some of the evolved music do have a moderately high rating for their human-like quality.

Result Observation

It is interesting to note that in survey B there is one piece of human-composed music received a mistake rate higher than 0.5. This indicates that human-composed music also has a chance of being thought of as being machine generated; this because sometimes human-composed music is systematic and too ordered. Finally, here are a few suggestions based on our observation and interpreted from the candidates' comments:

- We noticed that the evolved polyphonic music receives a higher rating than evolved monophonic music. Candidates preferred polyphonic music to monophonic music.

- Highly musically trained candidates tend to spot the harmonic error in the given music. Once the music is classified as machine generated then a lower rating is given by the candidate, instead of being judged fairly by the satisfaction given in terms of music perception. Candidates tend to use logical reasoning when filling out the survey rather than a perception of musical quality. A similar problem was experienced in the experiments carried out by Pearce and Wiggins [160].

6.2 Experiment 11: Fitness Distance Correlation Analysis

In chapter 5, we focused on using GA to evolve CA for music generation. The fundamental question arises of how hard it is for a GA to evolve CA? This experiment aims to analyse the difficulty of using GA to evolve CA with respect to the fitness and genotype distance between an optimal CA rule and sample CA rules. The analysis tool being used in this experiment is Fitness Distance Correlation (FDC) which is described in section 2.1.4.

6.2.1 Experiment Settings and Results

In the proposed system, the GA system evolves a population of CA transitional rules, which is a binary string. One way to measure the distance between binary strings is to use Hamming Distance [83]. Hamming distance measures the minimum number of steps taken to transform one string into the other, for example, distance between strings "0010" and "0111" is 2. In the experiment, hamming distance is used for measuring the distance between the optimal CA rule and sampled CA rules. For the fitness function, the evaluation is measuring the difference between the sampled CA and optimal CA space-time patterns by comparing their state probability distributions. The difference between two probability distributions is calculated by using Kullback-Leibler divergence. This fitness function is based on the N-gram model; each state is a unique N-gram window. The fitness function is named N-gram State Distribution (NSD). Since the problem is to search for the

Strip	State
0001000	A
0011100	B
0111110	C
0001000	A
0011100	B
0111110	C
0001000	A
0011100	B
1111111	D
1111111	D

Probabilities of Uni-gram states:
A=0.3, B=0.3, C=0.2 and D=0.2

Probabilities of Bi-gram states:
AB=0.333, BC=0.222, CA=0.222,
BD=0.111 and DD=0.111.

Figure 6.2: Example of obtain the probabilities of N-gram states.

CA that has a minimum error of state probability distribution away from the optimal CA's state probability distribution, then this is a minimisation problem. The optimal value of the divergence is zero. Higher values of the divergence are less fit (worse), the higher the value, the greater the divergence of the individual from the optimal. Figure 6.2 illustrates how to obtain the probabilities of N-gram states and its probabilities from a CA space-time pattern. In the table, the left column 'Strip' represents the sampled strip in the space-time pattern. Each lattice (row) is a state. For a Uni-gram model, there are four unique Uni-gram states: A, B, C and D. For a Bi-gram model, there are five unique Bi-gram states: AB, BC, CA, BD and DD. The probability of the N-gram state is calculated as the frequency of the N-gram state occurrence divided by total number of N-gram states.

For the experiment settings, each experiment was run ten times and averaged the FDC. At each run an optimal CA rule is randomly generated and then a space-time pattern (500 time step) is generated in order to obtain its state probability distribution. The settings and parameters of the CA

model in this experiment is the same as in chapter 5 - experiment 8. For the fitness function, various settings of NSD fitness functions and sizes (7, 13 and 19 bits) of sampling strip are tested. The column locations of the sampling strips are always in the middle of the space-time pattern. The number of CA rule samples at each run is 64,000 and they are obtained using the Random-Walks sampling algorithm. (Listing 6.1 shows the Random-Walks sampling algorithm in Java code.) At each run, there are 1,000 walks and each walk has 64 steps. This algorithm ensures that some samples are close in distance to the optimal CA rule. Each step is equivalent to one distance unit (flip one bit in the current CA rule).

Listing 6.1: Random-Walks algorithm in Java code

```
for(int i = 0; i < nWalks; i++) {
    currentCA = optimalCA;
    for(int j = 0; j < nSteps; j++) {
        nextCA = flipOneBit(currentCA);
        distance = hammingDist(optimalCA, nextCA);
        fitness = klDivergence(optimalCA, nextCA);
        updateStatistics(distance, fitness);
        currentCA = nextCA;
        }
    }
```

Experimental Results

The results summary is shown in Table 6.9. In the table, the header indicates the size of the sampling strip. Each data in the table is read as (n, f) where n is the N parameter in the NSD fitness function and f is the averaged

Sampling Strip size: 7-bit				
(1, 0.099)	(2, 0.510)	(3, 0.388)	(4, 0.309)	(5, 0.258)
(6, 0.219)	(7, 0.188)	(8, 0.167)	(9, 0.151)	(10, 0.137)
Sampling Strip size: 13-bit				
(1, 0.201)	(2, 0.240)	(3, 0.194)	(4, 0.175)	(5, 0.162)
(6, 0.152)	(7, 0.143)	(8, 0.136)	(9, 0.130)	(10, 0.124)
Sampling Strip size: 19-bit				
(1, 0.171)	(2, 0.176)	(3, 0.135)	(4, 0.123)	(5, 0.116)
(6, 0.110)	(7, 0.103)	(8, 0.097)	(9, 0.094)	(10, 0.091)

Table 6.9: Summary of FDC results (averaged ten runs).

FDC. The results show that if the size of the sampling strip increases, the average correlation tends to decrease. The problem becomes more difficult as the number of possible states in the strip is increased and it causes the size of the search space to increase. E.g. 7-bit and 13-bit strips have a total of 127 and 8,192 states respectively. Figures 6.3, 6.4, 6.5, 6.6, 6.8 and 6.9 show some selected results taken from Table 6.9. Each result is represented in a 3D scatter graph; X-axis presents the hamming distance between the target CA rule and the sampled CA rule. Y-axis presents the fitness of the sampled CA rule and Z-axis presents the frequency. Often more than one sampled CA rule has the same distance and fitness, and the rules are not necessarily the same. The frequency axis is useful for inspecting the number of sampled CA rules that have the same fitness and distance. Note that the fitness function here is a minimisation problem; fitness represents the error distance. A higher fitness of the CA rule has a higher divergence between its and the target CA's N-gram state distributions.

The immediate interpretation from the results is that for a small size of sampling strip and a low N order in NSD fitness function, the FDCs

belong to class 3 (Straightforward). This indicates that using an evolutionary algorithm to evolve CA rules with the given NSD fitness function is suitable and efficient if N is low and the size of the sampling strip is less than 13 bits. In the table, there is no FDC that belongs to class 1 (Misleading), either the problem belongs to Straightforward or Difficult. Such a class 2 (Difficult) problem found in the Uni-gram NSD fitness function with a 7-bit sampling strip, has an average FDC of 0.099. Figure 6.3 shows the result of a particular run with the Uni-gram NSD fitness function. In the figure, individuals with the same distance vary in their fitness ranging from 0 to 24. At the distance from 12 to 20, there are many sampled CA rules (see frequency axis) that have fitness between 24 and 28, the rest of the sampled CA rules lie between fitness 0 and 24. This result is reflected the class 2 problem. The reasons for this result are, first, some CA rules are far in hamming distance but their generated state distributions are similar, because the Uni-gram model takes no account of state ordering. If the state distributions are similar then the fitness is similar no matter whether or not the state ordering is different. Second, the size of the sampling strip is 7 bits, the number of possible states is as small as 128. Hence, the chance of having a similar state distribution is high when compared to a larger size of sampling strip.

For all runs, Bi-gram NSD has the highest correlation among others for all sizes of sampling strip. The problems belong to class 3 (Straightforward). Figure 6.4 shows the result of a particular run with the Bi-gram NSD fitness function. In this figure, there are more than 7,000 individuals which fall in between the distance of 12 to 20 and their fitness is higher than 20. For the individuals that have distances lower than 4, their fitness range is from 8

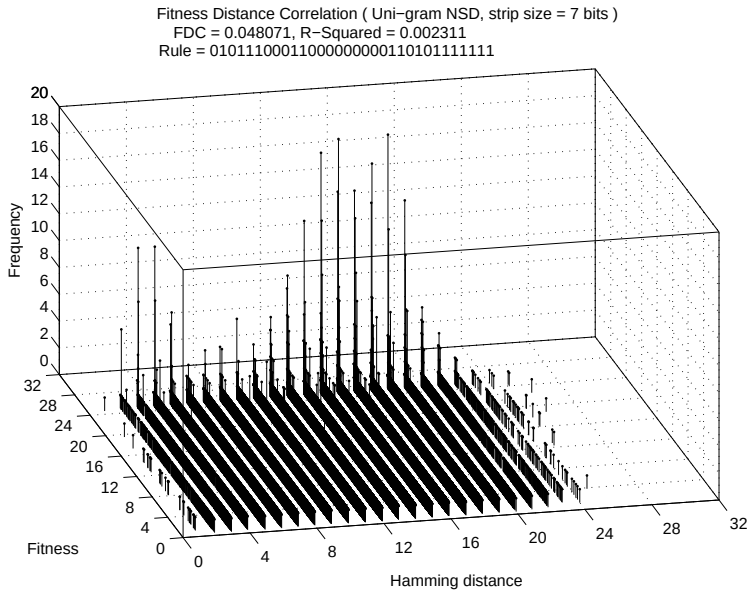


Figure 6.3: FDC analysis result of Uni-gram NSD fitness function.

to 24. This indicates that even a single or slight change in a CA rule, can change its state distribution dramatically. Although this characteristic of class 3 problem was found, when individuals distance goes up, their fitness range generally decreases. Figures 6.5 and 6.6 show two results of runs with 3-gram and 10-gram NSD fitness functions. In the figures, no individual has a fitness of less than 12. Even though some individuals are 1 distance away from the optimal CA rule, their fitness is dramatically increased, and the increased interval is higher than the Bi-gram NSD, because when the N parameter goes up, the ordering of the states is more sensitive. Subsequently, it is difficult for any individual to have a state distribution that is close to the global optimum.

In the results, we have found there to be an issue when the optimal CA's space-time pattern has a static or simple ordered pattern (see Figure 6.7),

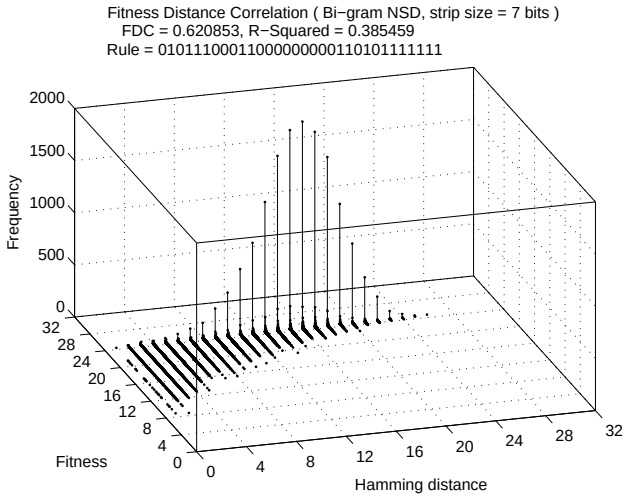


Figure 6.4: FDC analysis result of Bi-gram NSD fitness function.

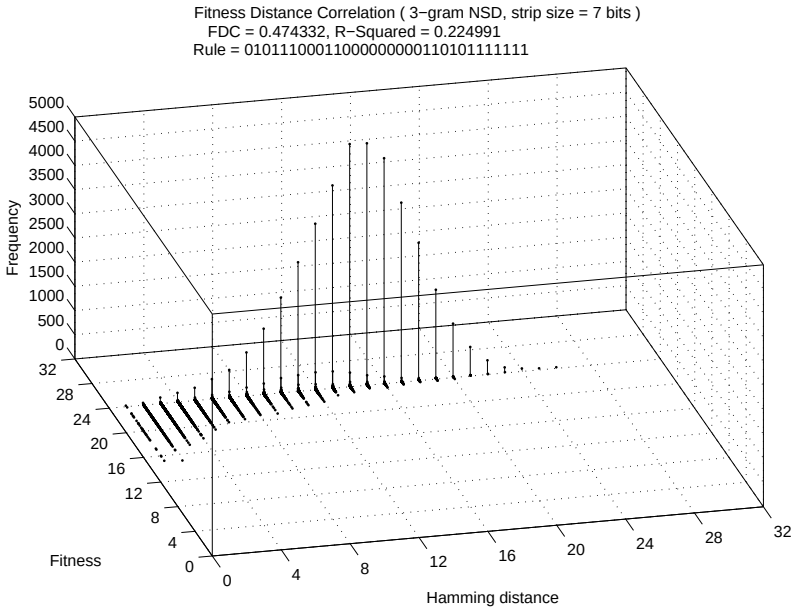


Figure 6.5: FDC analysis result of 3-gram NSD fitness function.

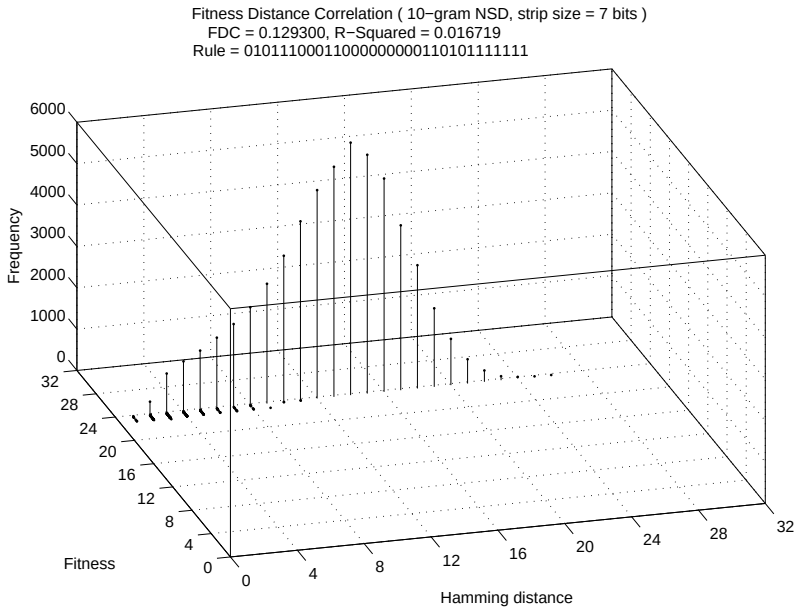


Figure 6.6: FDC analysis result of 10-gram NSD fitness function.

as the problem becomes difficult for GA evolution. Figures 6.8 and 6.9 show the results of runs with Uni-gram and Bi-gram NSD fitness functions, the space-time pattern generated by the given optimal CA shown in Figure 6.7. The optimal CA's space-time pattern is very ordered as apart from the initial state, the rests are in the same state for 500 iterations. In Figure 6.8 and 6.9, some individuals with high fitness are polar opposites of one another even though their distances are the same. This characteristic makes the GA extremely difficult for evolution. However, the proportions of the high fitness (high divergence away from the optimal) individuals are higher than those individuals with lower fitness. This result indicates that a high proportion of individuals have a complex space-time pattern and a small proportion of individuals have a static or ordered pattern in the search space.

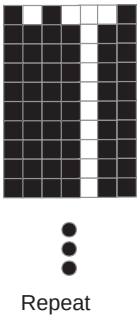


Figure 6.7: Static space-time pattern.
CA rule = 01110000101101001111010101101110.

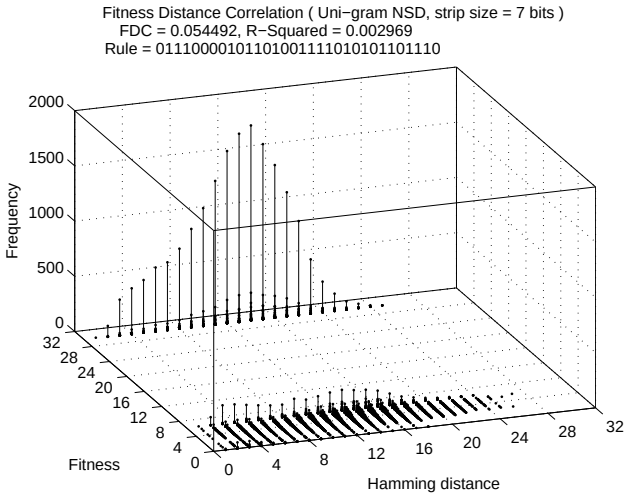


Figure 6.8: FDC analysis result of Uni-gram NSD fitness function.

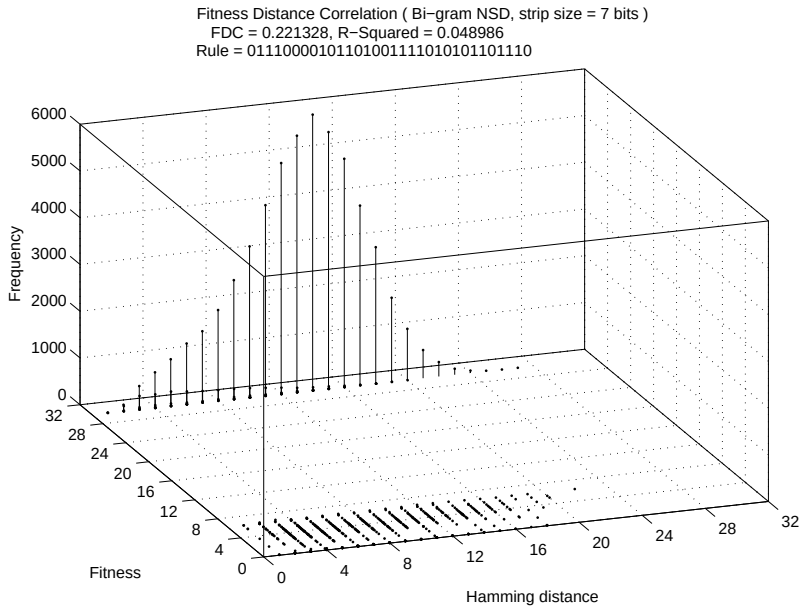


Figure 6.9: FDC analysis result of Bi-gram NSD fitness function.

6.2.2 Discussion of FDC Analysis

The experimental results showed that cellular automata is evolvable in general when using an N-gram based fitness function and the FDC showed the problems are in class 3 (Straightforward) if the sizes of N order in the NSD fitness function is low and the sampling strip is lower than 13 bits. Among the experiments, Bi-gram NSD has the highest correlation. However, there are exceptional cases when the target optimums have static or ordered space-time patterns, the FDCs showed the problems become class 2 (Difficult) with lower correlation. These results indicate that a small change in the CA rule causes a large impact in the space-time pattern. Also we found that some individuals have similar fitness or generated space-time patterns but their hamming distance is far. Therefore, the optimal space-time pattern can be

generated by many different CA rules and the FDC is low. Sometimes it is the other way round, some individuals have similar hamming distance, but their fitness or generated space-time patterns are significantly different.

We suggest that for efficiently using a GA to evolve CA rules, the fitness function should aim to guide the CA rules towards an optimal CA rule that generates a non-static space-time pattern, this can avoid the exceptional cases mentioned above. For an example, using an N-gram entropy fitness function, the GA is expected to evolve an optimal CA rule that generates a non-static space-time pattern. Another issue that needs to be considered is that any CA rule could produce various patterns if the initial configuration is randomly generated at each evaluation, its fitness might be different in the next evaluation. In order to evaluate the fitness of an individual CA rule, averaging the number of evaluations is a simple solution.

6.3 Chapter Summary

In part one, the experimental settings, results and analysis of the music surveys are described. The music surveys played an important role in this research. They provided an external evaluation for the proposed systems from a human listener's perspective. The external evaluation serves as a critique for evaluating these music evaluators and the system's design via the satisfaction of human listener's perceptions of the evolved music. The surveys not only tell us about the listeners' appreciation of the evolved music but also whether the evolved music is perceptually sound like a human-composed music. The results and feedback are important for estimating what needs to

be improved in our research for future development.

The second part of the chapter provided a preliminary investigation of how difficult it is for GA to evolve CA rules when given an N-gram-based fitness function. This investigation is important, because if the given fitness function is too difficult or misleading for the evolution process then the evolutionary algorithm should not be applied in the first place, and alterative search algorithms or fitness functions need to be considered.

Chapter 7

Conclusion and Future Directions

This chapter is organised as follow, section 7.1 is the conclusion of this thesis. Section 7.2 summarises the main contributions of this thesis. Section 7.3 discusses some shortcomings for the field of automatic music composition. Finally section 7.4 discusses some directions and gives suggestions for future study.

7.1 Conclusion

This research focuses on evolutionary computation techniques for music composition. It intended to investigate using N-gram based trainable fitness functions to evolve musical sequences in an evolutionary system. In this thesis, two types of evolutionary music composition systems have been developed following the proposed direct and indirect methodologies that were described

in section 3.2.7. The experimental results showed that: First, from the aspect of solution searching, EA has been shown to be an efficient random search algorithm for problems with a large search space. In this case the composition system relied on EA to search and explore interesting music in a large search space.

Second, from a developmental perspective, the N-gram based music evaluator is easy to implement when compared to expert systems and some other machine learning techniques such as ANN. It does not require any musical knowledge to be embedded into the model; users are only required to provide training samples for training the N-gram model. Also trainable fitness functions have an advantage over interactive fitness functions in terms of the speed of music evaluation.

Third, the the Musical Sequence Evolver (Direct approach) has the advantages of making it easier to control the melodic arrangement of the evolving musical sequence with appropriate sequence-oriented variation operators; and maximising the evolution power of directly manipulating the musical sequences to satisfy the fitness functions. The Cellular Automata Music Evolver (Indirect approach) has the advantage of evolving structured music sequences; the arrangement of pitches is handled by the cellular automata, which has been shown to be a dynamic model that is able to generate simple to complex patterns. Also, in terms of flexility and reusability, the cellular automata model can be replaced by any generative model if the user thought it better at matching the aim of the music composition system.

Fourth, the section of online music surveys demonstrated how to evaluate the performance of an automatic music composition system by listening

tests. We argue that for automatic music composition systems, the designers should not evaluate their systems' generated music based on their own judgements and make a statement such as "our system generates pleasant music" without conducting any music survey to collect results or feedback from human listeners.

7.2 Contributions

The following describes a number of major contributions being made by this thesis:

- I invented a music composition system that use an N-gram model as a fitness function in an evolutionary system for music composition. The experimental results in chapter 4 showed that using EA with the N-gram fitness function is guaranteed to evolve high probabilities of musical sequences, since the genetic operators directly manipulate the sequences. This addressed the problem of the conventional N-gram based music generator is no guaranteed to generate high probabilities of musical sequences.

- I invented a music composition system that use an evolutionary algorithm to evolve cellular automata for music generation. The experimental results showed that the cellular automata music evolver is able to indirectly evolve high level structural music without separating the evolution process[1]. The proposed indirect approach for evolutionary

[1]One approach for achieving this is to separate the process of evolving phrase and phrase arrangement. This approach has been explored by Thywissen [199] and Khalifa et

music opens up a new direction for evolutionary music. To the best of the author's knowledge and at the time this thesis was written, I have not found any papers that reported using any kind of automatic fitness function to evolve cellular automata for music generation. The best similar research is found in [143] [144] [15] [16] which used interactive fitness functions to evolve CA for music composition. And [115] [80] [101] [81] used EA to evolve CA for solving engineering problems.

- I invented a music composition system that use multi-objectives evolutionary algorithm for music composition. This system is based on the proposed cellular automata music evolver replacing the genetic algorithm with a Pareto-based multi-objectives optimisation algorithm the NSGA-II and SPEA2 to indirectly evolve musical sequences. This work provided a fundamental investigation of evolutionary music in multi-objectives perspective. Using a Pareto-based multi-objective algorithm to evolve music has the advantage of searching for multiple Pareto front music and the user is allowed to select the preferred music among them.

- I am the first researcher who used fitness distance correlation method to analyse the difficulty of evolving cellular automata using a genetic algorithm with an N-gram based fitness function. The analysis results showed that CA is suitable for the evolutionary process when using N-gram based fitness function.

al. [99].

7.3 Shortcoming for Automatic Music Composition

This section attempts to identify a shortcoming for the current development in the field of automatic music composition. Benchmarking is probably the most immediate shortcoming needing to be addressed in the near future. It can be described as running a set of evaluation tests to assess the performance of the target system from a number of perspectives. Benchmarking indicates how successful is the tested system and shows its marking for certain specific tasks. Currently, there is no benchmarking for automatic music composition systems. Conventionally, researchers evaluate their music composition systems by human listening tests or evaluating it according to a set of rules derived from music theory or a critic developed by the system designers.

The research in this area might be difficult, the immediate questions that arise are: What are the tasks for evaluating different perspectives of the system? What are the standard training music samples? We further suggest that the community might create a set of music samples in each musical genre and assume these samples are the most likely to be appreciated by the majority. In this way, at least system designers have some ideas about what sort of music it is expected that the automatic music composition systems will be able to generate. A successful automatic music composition system might be required to demonstrate it has an ability to generate music that is similar to the well-defined music samples. However, Tichy [200] stated that music composition is too subjective and creating a benchmark test is debatable. Tichy's statement argues that as any experiments involving human

subjectivity is so-called soft science; in the same experiment, results cannot be fully predictable or reproducible the next time.

7.4 Future Work and Suggestions

There is always room for improvement and possible ways of solving the problems that were encountered throughout in this thesis. N-gram and cellular automata models played essential parts in this thesis. To simplify the problem and make the variables more controllable, these models are used in a simple way. Enhancing these models in the current systems might improve the system performance. Taking an example, a variable-length N-gram model is one way of extending standard N-gram models that aim to counteract the unseen N-gram windows in the test data by dynamically changing the N order during the evaluation [147] [66] [188] [96]. Also it is fascinating to see more results from replacing the cellular automata model with different generative models e.g. L-system in the music evolver for music composition.

Finally we hope that we can improve our systems in the near future so that more people will enjoy our system-evolved music.

Appendix A

A.1 'Leave-one-out' cross validation

Leave-one-out cross validation is a statistical metric that evaluates how accurately a learning algorithm is able to predict unseen data. The evaluation starts from using a single data as the validation data, the rest of the data are the training data for the learning algorithm. This evaluation is repeated until each data is used once as the validation data. Then the prediction error rate is an average of all the evaluations. This evaluation method is computationally expensive, but there is no wasted data.

A.2 Absolute pitch frequency table of Mozart 110 samples

Pitch	Frequency	Pitch	Frequency	Pitch	Frequency	Pitch	Frequency
0-28	0	49	22	70	2511	90	12
29	7	50	89	71	4505	91	31
30	2	51	13	72	6293	92	0
31	21	52	156	73	3061	93	9
32	0	53	183	74	9197	94	0
33	11	54	56	75	2372	95	6
34	0	55	521	76	8072	96-127	0
35	9	56	54	77	5142	128	21894
36	79	57	355	78	4477		
37	2	58	182	79	7832		
38	9	59	506	80	1585		
39	5	60	1003	81	6693		
40	47	61	353	82	1781		
41	43	62	1618	83	3174		
42	17	63	355	84	2531		
43	154	64	2198	85	837		
44	18	65	1808	86	2326		
45	43	66	1282	87	154		
46	7	67	3947	88	518		
47	102	68	1217	89	88		
48	302	69	5275	90	12		

Appendix B

B.1 Survey B: composer answers and music ratings

(Answer, Rating), M=Machine and H=Human

Candidate	Music 1	Music 2	Music 3	Music 4	Music 5	Music 6	Music 7	Music 8	Music 9	Music 10
1	M 3	M 3	H 1	H 4	H 3	H 4	M 3	M 1	H 1	M 3
2	M 2	H 3	H 3	H 3	M 3	H 4	H 4	M 2	H 3	H 4
3	M 2	H 4	H 4	M 2	M 2	H 4	H 3	M 2	M 2	H 4
4	M 3	M 4	H 5	M 2	H 4	H 4	M 2	M 2	H 4	H 4
5	M 2	M 3	M 4	M 3	M 2	H 4	H 5	M 2	M 2	H 5
6	M 2	H 3	H 4	M 1	M 2	H 5	H 5	M 2	M 2	H 4
7	M 1	M 3	M 4	M 1	M 1	H 4	H 5	M 1	M 1	H 5
8	M 1	H 2	H 2	M 1	M 1	H 2	H 2	M 1	M 1	H 2
9	M 3	M 3	H 4	H 4	H 4	M 3	H 4	H 4	M 3	H 5
10	H 3	H 2	H 3	M 1	H 4	H 2	H 3	M 1	M 2	H 2
11	H 3	M 4	H 4	M 3	M 2	H 4	H 4	H 3	M 3	H 4
12	H 4	H 5	M 5	H 4	H 4	H 5	H 5	M 4	H 4	H 5
13	M 1	H 1	H 3	M 1	M 1	H 1	H 5	H 1	H 1	H 5
14	M 2	M 3	H 4	M 2	H 3	M 5	H 4	M 2	H 3	H 4
15	M 2	M 2	H 5	H 4	M 3	H 2	H 4	H 3	M 3	H 2
16	M 2	M 2	H 3	M 2	H 2	H 2	H 3	M 2	M 2	M 2
17	H 4	M 5	M 2	H 3	H 4	H 4	H 4	M 2	M 5	H 4
18	M 1	H 3	H 3	M 2	M 1	H 3	H 4	M 2	H 2	H 4
19	H 2	M 4	M 3	H 4	M 3	M 2	M 2	H 3	H 3	H 4
20	M 3	H 5	M 1	H 2	H 2	H 4	H 3	M 1	M 1	H 5
21	H 3	M 3	M 2	H 4	M 4	H 2	H 5	H 3	M 2	M 1
22	M 2	H 4	H 3	M 2	M 3	H 3	H 4	M 4	M 2	H 4
23	M 2	M 4	M 1	M 1	M 3	H 4	H 5	M 2	M 1	H 4
24	H 3	H 4	M 4	H 2	M 2	M 3	H 3	M 3	M 2	H 5
25	H 3	H 3	M 4	M 2	H 3	M 4	H 3	M 2	M 2	M 3
26	M 2	H 5	H 4	M 2	H 3	M 4	H 5	M 2	M 1	H 5
27	H 4	H 5	H 4	M 4	H 4	H 4	H 4	M 3	M 2	M 2
28	M 1	M 3	H 3	H 3	H 2	H 3	H 4	M 2	H 2	M 4
29	M 2	H 5	H 2	M 1	M 1	H 4	H 4	M 2	M 2	H 5
30	M 2	M 3	H 4	H 4	H 5	M 2	H 3	M 2	H 5	M 2
31	M 4	M 1	M 2	M 3	H 3	H 3	H 5	M 3	H 3	H 5

B.2 Survey B: candidates comments

Note: The comment numbers does not correspond to the candidate numbers.

Comment 1 "Some fun pieces. I like the uniformly mechanical performance, which serves to put the pieces on the same disadvantageous playing field. Basically the opposite of the EMI demos where a live performer plays all the pieces. I didn't really hate any of the pieces,

but some of the computer generated ones were less than compelling, so to speak... Fun stuff! I look forward to seeing your results."

Comment 2 "Hmmm, this was indeed challenging. Obviously tracks such as 10 are clearly composed by a professional (Schubert of course!) and even if someone did not know the composer they would recognise the subtle use of harmonic rhythm and fantastic pivotal modulations.. The tracks that were avant garde or second Viennese school style are harder to detect machine v human because the music can be so naturally mathematical in these styles anyway! Track 2 was difficult as the music is based on chords in the left hand that could have been put together by a beginner, and scales in the right hand that are very predictable - so a human (of limited musicianship) or a machine could have done this. Hope my comments help!"

Comment 3 "The ones I've labelled Machine sound just like the alpha version attempts, except that now there are (sometimes) two parts. The Human variants sometimes just have a single line, but they're all written in a bog-standard 19th century idiom of a kind which is quite hard to programme (baroque or classical music, such as Handel or Mozart is more predictable, for instance), so I would guess they had to be written by humans. One of the Machine pieces was played so fast that it could have been one of John Nancarrow's compositions, but I'm not sure whether you should classify his music as Human or Machine. Finally, the term song properly refers to a piece of music which is sung, not played on a piano (or piano-like MIDI file). It would be less irritating

if you dropped the iPod jargon and referred to the pieces as pieces".

Comment 4 "This is so interesting but also very difficult. I try to go by intuition but I am not sure at all whether I got it right. The reason why I hardly got any strong likes or dislikes is because the sound is the same in each case. It would be different, I guess, if it was played on a real piano. I am looking forward to finding out the results!! I would also like to know more about this research."

Comment 5 "There wasn't really a tab to know what was what. some pieces were beautiful but very repetitive. I didn't know if they were written by a human who wanted repetition to be a key factor or by a machine who repeats the data it has been given. Some pieces didn't sound so well but could be maybe the work of a writer who tries to compose like a machine without taking into account some harmonies that could have been done or maybe altering the tempo."

Comment 6 "No. 10 is Schubert! Now I don't know if it was programmed to be composed by a machine, or if and how this could be done, but I chose to say that it was composed by a human. Don't know if I got anything right but it seemed to me that those composed by a machine were either too random, or had a certain pattern that was repeated. Thanks a lot. I really enjoyed the test."

Comment 7 "I feel as if they're all composed by a machine, but have given my answers based on aesthetic responses to pre-concieved notions of human and synthesised. The MIDI with no velocity variation doesn't

help - perhaps you could repeat the experiment with real audio piano played by a human to determine whether we simply dislike the sound of relentless MIDI..."

Comment 8 "Nice work, but I think you should add another option in the 'Composed by' section which might say 'Not sure'. Cause some of them are really hard to tell. Good Luck"

Comment 9 "Would you be able to record the machine composed music using real instruments and human players(and the human composed as well)? I think that could make for an interesting (and difficult) test."

Comment 10 "The hardest songs to judge were those which had a repetitive melody such as songs 2, 6 and 10. They may be the product of a machine, however songs 6 and 10 incorporated several deviations to their repetition, which act as a kind of chorus, so I assumed a human created these. Hope that helps!"

Comment 11 "Some nice compositions there. I found many of them hard to judge."

Comment 12 "I'm arrogant enough to think I got all these correct. The basic problem is that the machine generated ones are just notes but have absolutely NO musicality. I can only assume that neither of you have any classical music training at all if the differences are not glaringly obvious to you."

Comment 13 "Hi all, I think I feel loops in a machine composition, though they are not very clearly and that is why I say feel loops. I choose

a human composition in songs where I feel more variations in tempo, and spacing of notes. Another thing is that when I feel a dissonance of different chords I incline to choose machine composition. I know there is music where dissonance exits, but somehow I feel these dissonances where not quite right. I hope this helps!"

Bibliography

[1] E. Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2004.

[2] C. Ames. The markov process as a compositional model: A survey and tutorial. *Leonardo Music Journal*, 22:175–187, 1989.

[3] T. Back. *Evolutionary Algorithms in Theory and Practice*. Oxford Univeristy Press, New York, 1996.

[4] M. Baroni. *Corpus linguistics: An international handbook*, chapter Distributions in text, pages 803–822. Berlin: Mouton de Gruyter, 2007.

[5] J. Barthelemy and A. Bonardi. Figured bass and tonality recognition. In *Proceedings of the 2nd International Conference on Music Information Retrieval*, 2001.

[6] E. Batlle and P. Cano. Automatic segmentation for music classification using competitive hidden markov models. In *Proceedings of the 1st International Conference on Music Information Retrieval*, 2000.

[7] L. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41:164–171, 1970.

[8] P. Bellini, P. Nesi, and G. Zoia. Symbolic music representation in mpeg. *IEEE Multimedia*, 12(4):42–49, 2005.

[9] J. Bello, G. Monti, and M. Sandler. Techniques for automatic music transcription. In *Proceedings of the 1st International Conference on Music Information Retrieval*, 2000.

[10] P. Bentley. *Evolutionary Design by Computers*. Morgan Kaufmann, San Mateo, California, 1999.

[11] P. Bentley. Exploring component-based representations - the secret of creativity by evolution? In *Proceedings of the 4th International Conference on Adaptive Computing in Design and Manufacture*, pages 26–28, 2000.

[12] P. Bentley and D. Corne. *Creative Evolutionary Systems*. Morgan Kaufmann, 2001.

[13] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kegl. Aggregate features and adaboost for music classification. *Machine Learning*, 65(2-3):473–484, 2006.

[14] P. Beyls. The musical universe of cellular automata. In *Proceedings of the 15th International Computer Music Conference*, pages 34–41, 1989.

[15] P. Beyls. Selectionist musical automata: Integrating explicit instruction and evolutionary algorithms. In *Proceedings of the 9th Brazilian Symposium on Computer Music*, 2003.

[16] P. Beyls. Cellular automata mapping procedures. In *Proceedings of the 30th International Computer Music Conference*, 2004.

[17] J. Biles. Genjam: A genetic algorithm for generating jazz solos. In *Proceedings of the 20th International Computer Music Conference*, pages 131–137, 1994.

[18] J. Biles. Composing with sequences:...but is it art? In *Proceedings of the 8th International Research Conference of Fibonacci Numbers and their Applications*, pages 61–73, 1998.

[19] M. Boden. *The Creative Mind: Myths and Mechanisms*. George Weidenfeld and Nicolson, 1990.

[20] A. Bregman. *Auditory Scene Analysis*. MIT Press, 1990.

[21] F. Brooks, A. Hopkins, P. Neumann, and W. Wright. *Machine Models of Music*, chapter An Experiment in Musical Composition, pages 23–40. MIT Press, 1992.

[22] L. Bui, D. Essam, H. Abbass, and D. Green. Performance analysis of evolution multi-objective optimisation algorithms in noisy environments. *Complexity International*, 11:29–39, 2005.

[23] D. Burraston. One dimensional cellular automata musical experiments with max. In *Proceedings of the 11th International Conference on Human-Computer Interaction*, 2005.

[24] D. Burraston and E. Edmonds. Cellular automata in generative electronic music and sonic art : A historical and technical review. *Digital Creativity*, 16(3):165–185, 2005.

[25] A. Burton and T. Vladimirova. Generation of musical sequences with genetic techniques. *Computer Music Journal*, 23(4):59–73, 1999.

[26] L. Candy and E. Edmonds. Introducing creativity to cognition. In *Proceedings of the 3rd Conference on Creativity and Cognition*, 1999.

[27] L. Chambers. *The Practical Handbook of Genetic Algorithms: Applications*. Chapman and Hall, 2000.

[28] M. Chang and J. Lee. Stimulative mechanism for creative thinking. In *Proceedings of the 1st International Association of Societies of Design Research Conference*, 2007.

[29] M. Chemillier and C. Truchet. Two musical csps. In *Musical Constraint Workshop. Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming*, 2001.

[30] M. Chemillier and C. Truchet. Computation of words satisfying the rhythmic oddity property (after simha arom's works). *Information Processing Letters*, 86:255–261, 2003.

[31] C. Chen and R. Miikkulainen. Creating melodies with evolving recurrent neural networks. In *Proceedings of the 12th International Joint Conference on Neural Networks*, pages 2241–2246, 2001.

[32] S. Choudhury, T. DiLauro, B. Harrington, M. Droettboom, I. Fujinaga, and K. Macmillan. Optical music recognition system within a large-scale digitization project. In *Proceedings of the 1st International Conference on Music Information Retrieval*, 2000.

[33] D. Cliff and H. Freeburn. Exploration of point-distribution models for similarity-based classification and indexing of polyphonic music. In *Proceedings of the 1st International Conference on Music Information Retrieval*, 2000.

[34] P. Codognet and D. Diaz. Yet another local search method for constraint solving. In *Proceedings of the 1st International Symposium on Stochastic Algorithms: Foundations and Applications*, 2001.

[35] P. Codognet, D. Diaz, and C. Truchet. The adaptive search method for constraint solving and its application to musical csps. In *1st International Workshop on Heuristics*, 2002.

[36] T. Cook and C. Congdon. Preliminary results with gauguin, an evolutionary computation approach to creating art in the suprematis style. In *Proceedings of the 9th IEEE Congress on Evolutionary Computation*, 2007.

[37] D. Cope. An expert system for computer-assisted music composition. *Computer Music Journal*, 11(4):30–46, 1987.

[38] D. Cope. *EMI: Experiments in Musical Intelligence*. A-R Editions, 1996.

[39] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, 1991.

[40] M. Crochemore, C. Iliopoulos, Y. Pinzon, and W. Rytter. Finding motifs with gaps. In *Proceedings of the 1st International Conference on Music Information Retrieval*, 2000.

[41] I. Cross. Ai and music perception. *AISB Quarterly*, 102:12–25, 1999.

[42] I. Cross. Music, mind and evolution. *Psychology of Music*, 29:95–102, 2001.

[43] I. Cross. Music, science & culture. *Imaginative Minds*, 147, 2007.

[44] P. Cruz-Alcazar and E. Vidal. Two grammatical inference applications in music processing. *Applied Artificial Intelligence*, 22(1-2):53–76, 2008.

[45] P. Dahlstedt. Autonomous evolution of complete piano pieces and performances. In *Musical Workshop, Proceedings of the 9th European Conference on Advances in Artificial Life*, 2007.

[46] R. Dannenberg and N. Hu. Pattern discovery techniques for music audio. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, 2002.

[47] C. Darwin. *On the Origin of Species by Means of Natural Selection*. John Murray, 1859.

[48] L. Davis. Bit-climbing, representational bias, and test suite design. In *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 18–23, 1991.

[49] A. de la Puente, R. Alfonso, and M. Moreno. Automatic composition of music by means of grammatical evolution. In *Proceedings of the 16th International Conference on APL: Array Processing Languages*, pages 148–155, 2002.

[50] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley and Sons Ltd, 2001.

[51] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pages 849–858, 2000.

[52] K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transaction on Evolutionary Computation*, 6(2):181–197, 2002.

[53] E. Dobson. *Understanding Fibonacci Numbers*. Traders Press, 1984.

[54] S. Doraisamy. *Polyphonic Music retrieval: The N-gram Approach*. PhD thesis, Imperial College London, University of London, 2004.

[55] S. Doraisamy and S. Ruger. A comparative and fault-tolerance study of the use of n-grams with polyphonic music. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, pages 101–106, 2002.

[56] S. Doraisamy and S. Ruger. Robust polyphonic music retrieval with n-grams. *Journal of Intelligent Information Systems*, 21(1):53–70, 2003.

[57] W. Dowling. Scale and contour: Two components of a theory of memory for melodies. *Psychological Review*, 85(4):341–354, 1978.

[58] J. Downie. *Evaluating a Simple Approach to Music Information Retrieval: Conceiving Melodic N-Grams as Text*. PhD thesis, University of Western Ontario, Canada, 1999.

[59] J. Downie and M. Nelson. Evaluation of a simple and effective music information retrieval method. In *Proceedings of the 23rd International Conference on Research and Development in Information Retrieval*, pages 73–80, 2000.

[60] J. Durillo, A. Nebro, L. Francisco, B. Dorronsoro, and E. Alba. jmetal: A java framework for developing multi-objective optimization metaheuristics. Technical report, University of Malaga, 2006.

[61] K. Ebcioglu. An expert system for harmonizing chorales in the style of j. s. bach. *Journal of Logic Programming*, 8(1-2):145–185, 1990.

[62] D. Eck and J. Schmidhuber. A first look at music composition using lstm recurrent neural networks. Technical report, Dalle Molle Institute for Artificial Intelligence, 2002.

[63] A. Eiben. *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*, chapter Evolutionary Reproduction of Dutch Masters: The Mondriaan and Escher Evolvers, pages 211–224. Springer Berlin Heidelberg, 2008.

[64] A. Eiben and J. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[65] L. Fausett. *Fundamentals of Neural Networks: Architectures, Algorithms And Applications*. Prentice Hall, 1994.

[66] L. Feng, K. Umemura, M. Yamamoto, and K. Church. Using variable length ngrams for retrieving technical abstracts in japanese. In *Proceedings of the 5th International Workshop on Information Retrieval with Asian Languages*, 2000.

[67] D. Fogel. *Evolving Artificial Intelligence*. PhD thesis, University of California at San Diego, 1992.

[68] J. Foote. Arthur: Retrieving orchestral music by long-term structure. In *Proceedings of the 1st International Conference on Music Information Retrieval*, 2000.

[69] J. Foote, M. Cooper, and U. Nam. Audio retrieval by rhythmic similarity. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, 2002.

[70] S. Forrest and M. Mitchell. Relative building-block fitness and the building-block hypothesis. *Foundations of Genetic Algorithms 2*, pages 109–126, 1993.

[71] J. Franklin. Jazz melody generation from recurrent network learning of several human melodies. In *Proceedings of The 18th International*

Conference of the Florida Artificial Intelligence Research Society, pages 57–62, 2005.

[72] M. Gardner. The fantastic combinations of john conway's new solitaire game life. *Scientific American*, 223:120–123, October 1970.

[73] A. Gartland-Jones and P. Copley. The suitability of genetic algorithms for musical composition. *Contemporary Music Review*, 22(3):43–56, 2003.

[74] J. Gero and V. Kazakov. An exploration-based evolutionary model of a generative design process. *Microcomputers in Civil Engineering*, 11:209–216, 1996.

[75] P. Gibson and J. Byrne. Neurogen, musical composition using genetic algorithms and cooperating neural networks. In *Proceedings of the 2nd International Conference on Artificial Neural Networks*, pages 309–313, 1991.

[76] D. Goldberg. *Evolutionary Design by Computers*, chapter The Race, the Hurdle, and the Sweet Spot: Lessons from Genetic Algorithms For the Automation of Design Innovation and Creativity, pages 105–118. Morgan Kaufmann, 1999.

[77] M. Good. Representing music using xml. In *Proceedings of the 1st International Conference on Music Information Retrieval*, 2000.

[78] M. Grachten. Jig: Jazz improvisation generator. In *Workshop on Current Research Directions in Computer Music*, pages 1–6, 2001.

[79] C. Grinstead and L. Snell. *Introduction to Probability*. American Mathematical Society, 1997.

[80] S. Guan and S. Zhang. An evolutionary approach to the design of controllable cellular automata structure for random number generation. *IEEE Transaction on Evolutionary Computation*, 7(1):23–36, 2003.

[81] Y. Guo, E. Keedwell, G. Walters, and S. Khu. Hybridizing cellular automata principles and nsgaii for multi-objective design of urban water networks. In *Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization*, pages 546–559, 2007.

[82] L. Ha, D. Stewart, P. Hanna, and F. Smith. Zipf and type-token rules for the english, spanish, irish and latin languages. *Web Journal of Formal, Computational & Cognitive Linguistics*, 8, 2006.

[83] R. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 26(2):147–160, 1950.

[84] R. Haupt and S. Haupt. *Practical Genetic Algorithms*. John Wiley & Sons, 2004.

[85] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[86] T. Heittola and A. Klapuri. Locating segments with drums in music signals. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, 2002.

[87] L. Hiller and L. Isaacson. *Experimental Music: Composition with an Electronic Computer*. McGraw-Hill, 1959.

[88] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[89] H. Hoos, K. Hamel, K. Renz, and J. Kilian. The guido notation format a novel approach for adequately representing score-level music. In *Proceedings of the 24th International Computer Music Conference*, 1998.

[90] J. Howard. *Learning to Compose*. UK: Cambridge University Press, 1990.

[91] A. Hunt, R. Kirk, and R. Orton. Muiscal application of a cellular automata workstation. In *Proceedings of the 17th International Computer Music Conference*, pages 165–168, 1991.

[92] B. Jacob. Algorithmic composition as a model of creativity. *Organised Sound*, 1(3):157–165, 1996.

[93] F. Jelinek and R. Mercer. Interpolated estimation of markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397, 1980.

[94] B. Johanson and R. Poli. Gp-music: An interactive genetic programming system for music generation with automated fitness raters. In *Proceedings of the 3rd International Conference on Genetic Programming*, pages 181–186, 1998.

[95] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, 1995.

[96] D. Jurafsky and J. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, 2000.

[97] D. Jurafsky, C. Wooters, G. Tajchman, J. Segal, A. Stolcke, E. Fosler, and N. Morgan. The berkeley restaurant project. In *Proceedings of the 13th International Conference on Speech and Language Processing*, pages 119–128, 1994.

[98] Y. Khalifa and B. Al-Mourad. Autonomous evolutionary music composer. In *Proceedings of the 8th International Conference on Genetic and Evolutionary Computation*, pages 1873–1874, 2006.

[99] Y. Khalifa, B. Khan, J. Begovic, A. Wisdom, and A. Wheeler. Evolutionary music composer integrating formal grammar. In *Proceedings of the 9th International Conference on Genetic and Evolutionary Computation*, pages 2519–2526, 2007.

[100] L. Khreisat. A machine learning approach for arabic text classification using n-gram frequency statistics. *Journal of Informetrics*, 3(1):72–77, 2009.

[101] R. Kicinger, T. Arciszewski, and K. De Jong. Morphogenesis and structural design: Cellular automata representations of steel structures in

tall buildings. In *Proceedings of the 6th IEEE Congress on Evolutionary Computation*, pages 411–418, 2004.

[102] Y. Kim, W. Chai, R. Garcia, and B. Vercoe. Analysis of a contour-based representation for melody. In *Proceedings of the 1st International Conference on Music Information Retrieval*, 2000.

[103] K. Koffka. *Principles of Gestalt Psychology.* Routledge and Kegan Paul, 1935.

[104] J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* The MIT Press, 1992.

[105] S. Kullback and R. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[106] W. Langdon. *Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming!*, volume 1. Kluwer, 1998.

[107] W. Langdon and R. Poli. *Foundations of Genetic Programming.* Springer-Verlag, 2002.

[108] C. Langton. Computation at the edge of chaos. *Physica D*, 42, 1990.

[109] W. Latham, M. Shaw, S. Todd, and F. Leymarie. Using dna to generate 3d organic art forms. In *Proceedings of the 9th International Conference on Genetic and Evolutionary Computation*, 2007.

[110] J. Lee, H. Cho, and H. Park. N-gram-based indexing for korean text retrieval. *Information Processing & Management*, 35(4):427–441, 1999.

[111] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. The MIT Press, 1983.

[112] K. Lin and T. Bell. Integrating paper and digital music information systems. In *Proceedings of the 1st International Conference on Music Information Retrieval*, 2000.

[113] M. Lo and S. Lucas. Evolving musical sequences with n-gram based trainable fitness functions. In *Proceedings of the 8th IEEE Congress on Evolutionary Computation*, pages 601–608, 2006.

[114] M. Lo and S. Lucas. N-gram fitness function with a constraint in a musical evolutionary system. In *Proceedings of the 9th IEEE Congress on Evolutionary Computation*, pages 4246–4251, 2007.

[115] J. Lohn and J. Reggia. Automatic discovery of self-replicating structures in cellular automata. *IEEE Transactions on Evolutionary Computation*, 1(3):165–178, 1997.

[116] M. Lothe. Knowledge based automatic composition and variation of melodies for minuets in early classical style. In *Proceedings of the 23rd German Conference on Artificial Intelligence*, 1999.

[117] S. Louis. *Genetic Algorithms As a Computational Tool for Design*. PhD thesis, Indiana University, 1993.

[118] B. Lourenco, J. Ralha, and M. Brandao. L-systems, scores, and evolutionary techniques. In *Proceedings of the 6th Sound and Music Computing Conference*, 2009.

[119] A. Machwe, I. Parmee, and J. Miles. Overcoming representation issues when including aesthetic criteria in evolutionary design. In *Proceedings of the International Conference on Computing in Civil Engineering*, 2005.

[120] B. Manaris, T. Purewal, and C. McCormick. Progress towards recognizing and classifying beautiful music with computers: Midi-encoded music and the zipf-mandelbrot law. In *Proceedings of the 3rd IEEE SoutheastCon*, pages 52–57, 2002.

[121] B. Manaris, J. Romero, P. Machado, D. Krehbiel, T. Hirzel, W. Pharr, and R. Davis. Zipf's law, music classification, and aesthetics. *Computer Music Journal*, 29(1):55–69, 2005.

[122] B. Manaris, P. Roos, P. Machado, D. Krehbiel, L. Pellicoro, and J. Romero. A corpus-based hybrid approach to music analysis and composition. In *Proceedings of the 22nd International Conference on Artificial Intelligence*, pages 839–845, 2007.

[123] B. Manaris, D. Vaughan, C. Wagner, J. Romero, and R. Davis. Evolutionary music and the zipf-mandelbrot law: Developing fitness functions for pleasant music. In *Proceedings of the 1st European Workshop on Evolutionary Music and Art*, pages 522–534, 2003.

[124] J. McCormack. Interactive evolution of l-system grammars for computer graphics modelling. *Complex Systems: From Biology to Computation*, pages 118–130, 1993.

[125] J. McCormack. Grammar-based music composition. *Complex International*, 3, 1996.

[126] J. McCormack. Aesthetic evolution of l-systems revisited. In *Proceedings of the 2nd European Workshop on Evolutionary Music and Art, Applications of Evolutionary Computing*, pages 477–488, 2004.

[127] R. McIntyre. Bach in a box: The evolution of four part baroque harmony using the genetic algorithm. In *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, 1994.

[128] P. McNamee, J. Mayfield, and C. Piatko. A language-independent approach to european text retrieval. In *Proceedings of the 2000 Workshop on Cross-Language Information Retrieval and Evaluation*, pages 129–139, 2000.

[129] J. McPherson. Introducing feedback into an optical music recognition system. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, 2002.

[130] C. Meek and W. Birmingham. Automatic thematic extractor. *Journal of Intelligent Information Systems*, 21(1):9–33, 2003.

[131] S. Meyn and R. Tweedie. *Markov Chains and Stochastic Stability*. Springer-Verlag, 1993.

[132] M. Mike. Did mozart use the golden section? *American Scientist*, 84:118–119, 1996.

[133] G. Miller. *The Mating Mind: How Sexual Choice Shaped the Evolution of Human Nature*. Anchor, 2001.

[134] E. Miranda. Evolving cellular automata music: From sound synthesis to composition. In *Proceedings of the 1st Workshop on Artificial Life Models for Musical Applications*, 2001.

[135] E. Miranda and J. Biles. *Evolutionary Computer Music*. Springe, 2007.

[136] E. Miranda and P. Todd. A-life and musical composition: A brief survey. In *Proceedings of the 9th Brazilian Symposium on Computer Music*, 2003.

[137] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1998.

[138] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[139] N. Morgan. Transformation and mapping of l-systems data in the composition of a large-scale instrumental work. In *Proceedings of the Workshop on Music and Artificial Life, 9th European Conference on Artificial Life*, 2007.

[140] A. Moroni, J. Manzolli, F. Zuben, and R. Gudwin. Vox populi: An interactive evolutionary system for algorithmic music composition. *Leonardo Music Journal*, 10:49–54, 2000.

[141] D. Morris, I. Simon, and S. Basu. Exposing parameters of a trained dynamic model for interactive music creation. In *Proceedings of the 23rd International Conference on Artificial Intelligence*, pages 784–791, 2008.

[142] M. Mozer. *Music and Connectionism*, chapter Connectionist Music Composition Based on Melodic, Stylistic, and Psychophysical Constraints, pages 195–211. MIT Press, 1991.

[143] G. Nelson. Sonomorphs: An application of genetic algorithms to the growth and development of musical organisms. In *Proceedings of the Fourth Biennial Art & Technology Symposium*, 1993.

[144] G. Nelson. Further adventures of the sonomorphs. In *Proceedings of the Fifth Biennial Art & Technology Symposium*, 1995.

[145] J. Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, 1966.

[146] J. Nie, J. Gao, J. Zhang, and M. Zhou. On the use of words and n-grams for chinese information retrieval. In *Proceedings of the 5th International Workshop on Information Retrieval with Asian Languages*, pages 141–148, 2000.

[147] T. Niesler and P. Woodland. A variable-length category-based n-gram language model. In *Proceedings of the 21st International Conference on Acoustics, Speech, and Signal Processing*, 1996.

[148] T. Oliwa. Genetic algorithms and the abc music notation language for rock music composition. In *Proceedings of the 10th International Conference on Genetic and Evolutionary Computation*, 2008.

[149] R. Ovans. An object-oriented constraint satisfaction system applied to music composition. Master's thesis, Simon Fraser University, 1990.

[150] R. Ovans and R. Davison. An interactive constraint-based expert assistant for music composition. In *Proceedings of the 9th Canadian Conference on Artificial Intelligence*, 1992.

[151] E. Ozcan and T. Ercal. A genetic algorithm for generating improvised music. In *Proceedings of the 8th International Conference on Artificial Evolution*, 2007.

[152] F. Pachet and D. Laigre. A naturalist approach to music file name analysis. In *Proceedings of the 2nd International Conference on Music Information Retrieval*, pages 51–58, 2001.

[153] F. Pachet and P. Roy. Mixing constraints and objects: A case study in automatic harmonization. In *Proceedings of the TOOLS Europe'95 Conference*, 1995.

[154] F. Pachet and P. Roy. Musical harmonization with constraints: A survey. *Constraints Journal*, 6(1):7–19, 2001.

[155] G. Papadopoulos and G. Wiggins. A genetic algorithm for the generation of jazz melodies. In *Proceedings of the 3rd Finnish Artificial Intelligence Conference (STeP)*, 1998.

[156] G. Papadopoulos and G. Wiggins. Ai methods for algorithmic composition: A survey, a critical view and future prospects. In *Proceedings of the AISB'99 Symposium on Musical Creativity*, 1999.

[157] R. Parncutt. Systematic musicology and the history and future of western music scholarship. *Journal of Interdisciplinary Music Studies*, 1(1):1–32, 2007.

[158] D. Parsons. *The Directory of Tunes and Musical Themes*. S. Brown, 1975.

[159] N. Patel and P. Mundur. An n-gram based approach for finding the repeating patterns in musical data. In *Proceedings of the IASTED European Conference on Internet, Multimedia Systems and Applications*, pages 407–412, 2005.

[160] M. Pearce and G. Wiggins. Towards a framework for the evaluation of machine compositions. In *Proceedings of the AISB 01 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, 2001.

[161] M. Pearce and G. Wiggins. Aspects of a cognitive theory of creativity in musical composition. In *Proceedings of the Workshop on Creative Systems, 15th European Conference on Artificial Intelligence*, 2002.

[162] S. Phon-Amnuaisuk, A. Tuson, and G. Wiggins. Evolving musical harmonisation. In *Proceedings of the Fourth International Conference on Neural Networks and Genetic Algorithms*, 1999.

[163] J. Pickens. A comparison of language modeling and probabilistic text information retrieval approaches to monophonic music retrieval. In *Proceedings of the 1st International Conference on Music Information Retrieval*, 2000.

[164] S. Pinker. *How the Mind Works*. W.W. Norton, 1997.

[165] R. Pinkerton. Information theory and melody. *Scientific American*, 194(2):77–86, 1956.

[166] D. Ponsford, G. Wiggins, and C. Mellish. Statistical learning of harmonic movement. *Journal of New Music Research*, 28:150–177, 1999.

[167] L. Prechelt and R. Typke. An interface for melody input. *ACM Transactions on Computer-Human Interaction*, 8(2):133–149, 2001.

[168] P. Prusinkiewicz. Score generation with l-systems. In *Proceedings of the 1986 International Computer Music Conference*, pages 455–457, 1986.

[169] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. New York: Springer-Verlag, 1990.

[170] J. Putz. The golden section and the piano sonatas of mozart. *Mathematics Magazine*, 68:275–281, 1995.

[171] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *IEEE*, 77(2):257–286, 1989.

[172] W. Raynor. *The International Dictionary of Artificial Intelligence*. Lessons Professional, 1999.

[173] J. Reddin, J. McDermott, and M. O'Neill. Elevated pitch: Automated grammatical evolution of short compositions. In *Proceedings of the 7th European Workshop on Evolutionary Music and Art, Applications of Evolutionary Computing*, pages 579–584, 2009.

[174] P. Reiners. Cellular automata and music: Using the java language for algorithmic music composition, 2004.

[175] C. Roads. Research in music and artificial intelligence. *ACM Computing Surveys*, 17(2):163–190, 1985.

[176] P. Roland. Xml4mir: Extensible markup language for music information retrieval. In *Proceedings of the 1st International Conference on Music Information Retrieval*, 2000.

[177] R. Rooksby. *Melody: How to Write Great Tunes*. Backbeat Book, 2004.

[178] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[179] C. Ryan, J. Collins, and M. O'Neill. Grammatical evolution: Evolving programs for an arbitrary language. In *Proceedings of the 1st European Workshop on Genetic Programming*, 1998.

[180] P. Salosaari and K. Jarvelin. Musir - a retrieval model for music. Technical report, University of Tampere, Department of Information Studies, 1998.

[181] P. Sarkar. A brief history of cellular automata. *ACM Computing Surveys*, 32(1):80–107, 2000.

[182] R. Scherle and D. Byrd. The anatomy of a bibliographic search system for music. In *Proceedings of the 5th International Conference on Music Information Retrieval*, 2004.

[183] C. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[184] C. Shannon. Prediction and entropy of printed english. *The Bell System Technical Journal*, 30:50–64, 1950.

[185] S. Shtrikman. Some comments on zipf's law for the chinese language. *Journal of Information Science*, 20(2):142–143, 1994.

[186] I. Simon, D. Morris, and S. Basu. Mysong: Automatic accompaniment generation for vocal melodies. In *Proceeding of the 26th International Conference on Human Factors in Computing Systems*, pages 725–734, 2008.

[187] K. Sims. Artificial evolution for computer graphics. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, 1991.

[188] M. Siu and O. M. Variable n-grams and extensions for conversational speech language modeling. *IEEE Transactions on Speech and Audio Processing*, 8(1):63–75, 2000.

[189] K. Soo. Zipf's law for cities: A cross-country investigation. *Journal of Regional Science and Urban Economics*, 35(3):239–263, 2005.

[190] D. Sornette, L. Knopoff, Y. Kagan, and C. Vanneste. Rank-ordering statistics of extreme events: Application to the distribution of large earthquakes. *Journal of Geophysical Research*, 101:13883–13894, 1995.

[191] L. Spector and A. Alpern. Criticism, culture, and the automatic generation of artworks. In *Proceedings of the 9th National Conference on Artificial Intelligence*, 1994.

[192] L. Spector and A. Alpern. Induction and recapitulation of deep musical structure. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 41–48, 1995.

[193] L. Spector, J. Klein, and K. Harrington. Selection songs: Evolutionary music computation. *YLEM Journal (Artists Using Science and Technology)*, 25(6):24–26, 2005.

[194] P. Sturman. *Harmony, Melody and Composition.* UK Longman Group, 1983.

[195] P. Sturman. *Advanced Harmony, Melody and Composition.* UK Longman Group, 1986.

[196] H. Takagi. Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. *IEEE*, 89(9):1275–1296, 2001.

[197] E. Taylor. *The AB Guide To Music Theory Part I.* The Associated Board of the Royal School of Music, 1989.

[198] E. Taylor. *The AB Guide To Music Theory Part II.* The Associated Board of the Royal School of Music, 1991.

[199] K. Thywissen. Genotator: An environment for exploring the application of evolutionary techniques in computer assisted composition. *Organised Sound*, 4(2):127–133, 1999.

[200] W. Tichy. Should computer scientists experiment more? *IEEE Computer*, 31(5):32–40, 1998.

[201] P. Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 13(4):27–43, 1989.

[202] P. Todd and G. Loy. *Music and Connectionism*. The MIT Press, 1991.

[203] P. Todd and G. Werner. *Musical Networks: Parallel Distributed Perception and Performance*, chapter Frankensteinian Methods for Evolutionary Music Composition, pages 313–340. MIT Press, 1999.

[204] N. Tokui and H. Iba. Music composition with interactive evolutionary computation. In *Proceedings of the 3rd International Conference on Generative Art*, pages 215–226, 2000.

[205] M. Towsey, A. Brown, S. Wright, and J. Diederich. Towards melodic extension using genetic algorithms. *Journal of Educational Technology & Society*, 4(2):54–65, 2001.

[206] C. Truchet, C. Agon, and P. Codognet. A constraint programming system for music composition, preliminary results. In *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming*, 2001.

[207] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, 1993.

[208] R. Typke, F. Wiering, and R. Veltkamp. A survey of music information retrieval systems. In *Proceedings of the 6th International Conference on Music Information Retrieval*, pages 153–160, 2005.

[209] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[210] A. Uitdenbogerd and J. Zobel. Matching techniques for large music databases. In *Proceedings of the 7th ACM International Conference on Multimedia*, pages 57–66, 1999.

[211] A. Uitdenbogerd and J. Zobel. Music ranking techniques evaluated. In *Proceedings of the 25th Australasian Conference on Computer Science*, 2002.

[212] M. Unehara and T. Onisawa. Construction of music composition system with interactive genetic algorithm. In *Proceedings of the 6th Asian Design International Conference*, 2003.

[213] B. Vorn. Lifttools. http://billvorn.concordia.ca/, 1996.

[214] R. Voss and J. Clarke. 1/f noise in music and speech. *Nature*, 258:317–318, 1975.

[215] G. Wiggins. The use of constraint systems for musical composition. In *Proceedings of the workshop on Constraints for Artistic Applications, 13th European Conference on Artificial Intelligence*, 1998.

[216] S. Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55:601–644, 1983.

[217] S. Wolfram. *A New Kind of ScienceA New Kind of Science*. Wolfram Media, 2002.

[218] P. Worth and S. Stepney. Growing music: Musical interpretations of l-systems. In *Proceedings of the 3rd European Workshop on Evolutionary Music and Art, Applications of Evolutionary Computing*, pages 545–550, 2005.

[219] A. Wuensche and M. Lesser. *The Global Dynamics of Cellular Automata: An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata*. Addison Wesley, 1992.

[220] C. Yang. The macsis acoustic indexing framework for music retrieval: An experimental study. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, 2002.

[221] C. Yip and B. Kao. A study of musical features for melody databases. In *Proceedings of the 10th International Conference on Database and Expert Systems Applications*, pages 724–733, 1999.

[222] C. Yip and B. Kao. A study on n-gram indexing of musical features. In *Proceedings of the 1st IEEE International Conference on Multimedia and Expo*, pages 869–872, 2000.

[223] D. Zanette. Zipf's law and the creation of musical context. *Musicae Scientiae*, 10:3–18, 2006.

[224] G. Zipf. *The Psychobiology of Language*. Houghton-Mifflin, 1935.

[225] G. Zipf. *Human behavior and the principle of least effort*. Addison-Wesley, 1949.

[226] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[227] E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm. TIK Report 103, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, 2001.

[228] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[229] J. Zydallis, D. Van Veldhuizen, and G. Lamont. A statistical comparison of multiobjective evolutionary algorithms including the momga-ii. In *Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization*, pages 226–240, 2001.