A Genetic Programming Approach to Generating Musical Compositions

David M. Hofmann(✉)

Institute for Musicology and Music Informatics,
University of Music Karlsruhe, Karlsruhe, Germany
hofmann@hfm.eu

Abstract. Evolutionary algorithms have frequently been applied in the field of computer-generated art. In this paper, a novel approach in the domain of automated music composition is proposed. It is inspired by genetic programming and uses a tree-based domain model of compositions. The model represents musical pieces as a set of constraints changing over time, forming musical contexts allowing to compose, reuse and reshape musical fragments. The system implements a multi-objective optimization aiming for statistical measures and structural features of evolved models. Furthermore a correspondent domain-specific computer language is introduced used to transform domain models to a comprehensive, human-readable text representation and vice versa. The language is also suitable to limit the search space of the evolution and as a composition language for human composers.

Keywords: Automated music generation · Multi-objective genetic programming · Domain-specific languages

1 Introduction

Since the beginning of the computer era it has been a question if computers could ever be considered creative. The idea of computer-generated music goes back to 1843, when Ada Lovelace mentioned the “Analytical Engine’s potential for automated composition” [1]. Since then, numerous attempts have been made to create music with computer programs. Applied programming techniques include pseudo-randomly generated musical sequences, generative grammars, recursive transition networks, Markov models, artificial neural networks and cellular automata [2]. Another approach is the application of evolutionary algorithms (EAs). A widely held belief is that the highest observable extent of creativity exhibited by computer programs is limited by the creativity of the programmer. However, the application of evolutionary algorithms is especially promising because at times results of considerable innovative and subjectively perceived “creative” quality are produced. These results often surprise experts, including the programmer [3]. In this paper, a system inspired by genetic programming (GP, which is a subset of evolutionary algorithms) is presented.

© Springer International Publishing Switzerland 2015
C. Johnson et al. (Eds.): EvoMUSART 2015, LNCS 9027, pp. 89–100, 2015.
DOI: 10.1007/978-3-319-16498-4_9

2 Related Work

A number of relevant systems have been proposed, most of which are based on genetic algorithms (GAs) [3]. Horner and Goldberg proposed an approach to thematic bridging, evolving transitions from an initial musical pattern to another pattern [4]. Biles created an interactive system named *GenJam* which evolves jazz solos using musically meaningful mutation and crossover operators [5,6]. Horowitz focused on evolving rhythmic patterns [7]. GAs have also been applied to the problem of harmonization, adhering to a number of constraints according to traditional music theory [8,9]. Jacob developed a composition system named *variations* [10,11]. It uses three GA-based components which are responsible for generating, evaluating and arranging music respectively. A system generating short melodies using GP was introduced by Johanson and Poli [12]. Specialized crossover and mutation operators were used by Marques et al. [13] and a grammar-based approach was proposed by de la Puente et al. [14]. *SARAH* is a composition language introduced by Fox [15], enabling to define musical phrases and arrange them hierarchically. A GA-based program named *GenDash* was developed by Waschka II [16], in which the whole evolutionary process, transforming musical user input, represents the composition.

3 Motivation

The intended purpose of the proposed model is to overcome some limitations inherent in genetic representations of previous works. In most of the models, the evolution is focused on only one or few musical aspects such as pitches or rhythms. Misleadingly, "music" is often regarded as simply a sequence of notes and rests. In fact, these elements have to be seen within their context, taking metre, rhythm, tonal system, tonal center, harmony, scale and loudness into account. In many of the mentioned systems these contexts are hard-coded and are not part of the actual evolution. The proposed model is designed to (a) represent a large number of musical aspects and parameters, (b) encode compositions of arbitrary complexity and length with minimal redundancy, (c) allow not only the evolution of one-dimensional sequences, but also their hierarchical context, (d) enable to reuse and vary existing material rather than introducing new material all the time and (e) incorporate some mechanisms of human creativity, as explained in the next section. None of the systems mentioned above combine all of these characteristics in their representation.

4 Creativity and Constraints

It is highly implausible that all aspects of human creativity can ever be modeled by a computer. However, the proposed software system tries to incorporate some aspects of human creativity. Boden outlines an interesting relation between constraints and creativity as follows: "Constraints – far from being opposed to creativity – make creativity possible. To throw away all constraints would be to destroy the capacity for creative thinking" [17]. As soon as constraints are present in creative processes, three common principles of creativity can be applied: **Exploration**, where the space of possibilities is searched, **Combination**, where aspects in the possibility space are combined in order to form new ideas and **Transformation**, in which the constraint space itself is modified, leading to a different, mostly larger space of possibilities [18].

5 Domain Model

The proposed model represents compositions in terms of musical constraint sets changing over time. Possibilities in this space can be explored, aspects can be combined and the space can be transformed by a computer program. This is accomplished by arranging the following elements in a tree-based structure: Musical *constraints*, constraint *modifiers*, constraint *generators* and *control structures*. Constraints include basic compositional elements such as tonal systems, instruments, beats, rhythms, harmonies, chords, scales and pitches. Compositions are often structured in such a way that established constraints are not completely changed, but only slightly modified in order to reshape already introduced material. Constraint modifiers represent modifications such as rhythmic and tonal variations, inversions, displacements and transpositions. Generators have the functionality of generating new constraints based on already existing ones in a specific pattern. For example, an *arpeggio generator* is responsible for generating pitch constraints based on the current harmony using the current rhythm and a specified pattern of sequentially sounding chord notes.

5.1 Example

The function of the individual components is explained using a constraint model of the first four measures of Bach's *Prelude in C major* from *The Well-Tempered Clavier (BWV 846)* which is shown in Fig. 1. The nodes are arranged in an acyclic graph structure. Every model has a root node which is displayed labeled *composition*. Models are evaluated from top to bottom, aggregating all constraints at the leaf nodes. This implies that every leaf node has a musical context, which is defined as a set of all constraints in the path from the root node to the leaf node. Multiple child nodes are interpreted as a consecutive sequence of constraints. If a path contains a constraint of the same type more than once, the constraint which is nearest to the leaves overwrites all upper constraints of that type. This model architecture allows temporary transformations of previously established constraint spaces, enabling to model surprising musical twists in a composition. The model supports modularization in the form of fragments. These are reusable subtrees which can be referenced from anywhere else in the model as long as no cycle is produced. In this way musical ideas can be reused and even reshaped by creating child nodes under the reference node, supporting context-dependent variations. The model also features control structures (e.g. repetitions and iterations). Polyphonic compositions are represented using a control structure named *parallelization* indicating that its child nodes are arranged concurrently in time.

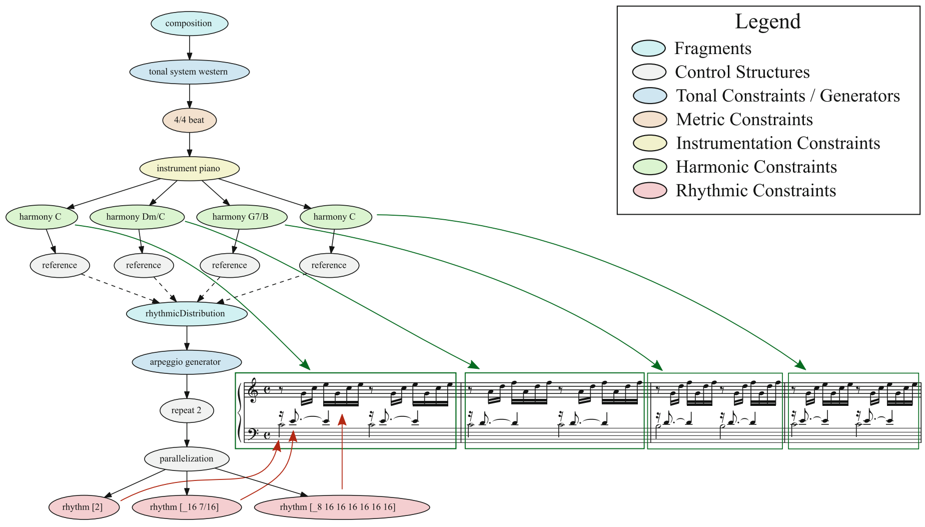

Fig. 1. Constraint model of the first four measures of Bach's *Prelude in C major* from *The Well-Tempered Clavier (BWV 846)*

6 The Domain-Specific Language

A domain-specific language (DSL) correspondent to the introduced model was developed. Its name is MC^{3}L, abbreviating **M**usical **C**onstraint, **C**ontext and **C**omposition **L**anguage. Its main purpose is to provide a comprehensive textual representation of models. Thus the evolutionary process can be monitored in a very convenient way as the progress can be persisted in simple text files at any time. The process also works vice versa which enables users to express musical compositions in terms of constraint spaces. By this means, a "side-product" of the system is a comprehensive computer language that can be used by human composers to capture compositions and ideas. This makes the language a multi-purpose tool for both traditional and algorithmic composition.

6.1 Language Syntax

The language syntax is demonstrated with a code example which is equivalent to Fig. 1. As demonstrated in the first lines, model fragments can be distributed over several files in order to minimize redundancy. The language allows users to specify custom tonal systems and instruments. The grammar features a basic expression language which enables to formulate parameters using arithmetical and logical expressions, whereupon functions can be invoked. The explanation of the complete language syntax, however, would go beyond the scope of this paper.

```
import"../../tonalSystems/western.mcl"
import "../../instruments/piano.mcl"
title "Prelude in C major"
composer "J.S. Bach"
root {
    tonalSystem western {
        beat 4/4 {
            instrument piano {
                harmony C {
                    fragmentRef rhythmicDistribution
                }
                harmony Dm/C {
                    fragmentRef rhythmicDistribution
                }
                harmony G7/B {
                    fragmentRef rhythmicDistribution
                }
                harmony C {
                    fragmentRef rhythmicDistribution
                }
            }
        }
    }
}
fragment rhythmicDistribution {
    arpeggioGenerator [startOctave 3, numberOfNotes 5, noteIndexSequence 0 1 2
     3 4 2 3 4, includeBassNote true]
    {
        repeat 2 {
            parallel {
                rhythm 2
                rhythm _16 7/16
                rhythm _8 16 16 16 16 16 16
            }
        }
    }
}
```

Listing 1.1. Syntactical Representation of the Prelude Constraint Model in Fig. 1

7 Transformation Infrastructure

The system supports a number of transformations in order to produce graphical and audible material from models. Graphical representations of models, as already seen in Fig. 1, are generated using a graph language called *DOT*. Models can be transformed to a sequential *stream model*. This is accomplished by a compiler which traverses the tree structure until it reaches a leaf node. It then evaluates the constraint space and writes sequences of constraints, one for each constraint type, into a corresponding timeline depending on the parallelization context. Stream models can in turn be converted into a *score model* containing score-specific events such as notes, rests and loudness instructions. Currently an export module to the music notation language *LilyPond* is implemented which enables to export the score as PDF and MIDI files.

8 Evolutionary Composition System

The core of the automated composition system is inspired by the genetic programming paradigm. An initial constraint space and the fitness function configuration are given as input. The initial constraint space can either be unbounded (which means virtually any composition can be the result) or contain constraints limiting the possibility space of the evolution. For example, a set of applicable instruments, musical fragments to be incorporated or musical forms and structures can be purported. The initial constraint space can conveniently be specified using MC^{3}L. Note that it is possible to define which parts of the initial constraint space may be modified during the evolution. This is possible using two keywords: The *fixed* keyword indicates that the correspondent node must not be moved or removed. The *final* keyword indicates the same policy applied recursively for a whole subtree. Additionally, no more child nodes may be added to a final subtree. For example, if the user would like the output to be a canon with three voices, the initial constraint space shown in Listing 1.2 could be specified. It defines a parallelization with three voices, each of which references the same fragment named *melody*. The references of the second and third voice are delayed by two respectively four whole measures of rests. This effectively limits the evolution to happen under the last repetition node.

```
root final {
    parallel {
        fragment voice1 {
            fragmentRef melody
        }

        fragment voice2 {
            rhythm _2!
            fragmentRef melody
        }

        fragment voice3 {
            rhythm _4!
            fragmentRef melody
        }
    }
}
fragment melody fixed {
    repeat 3 fixed
}
```

Listing 1.2. Syntactical Representation of a Constraint Space Yielding a Canon with Three Voices

9 Fitness Function

A particularly challenging part when generating music using evolutionary algorithms is the design of a suitable fitness function to evaluate and compare the evolved compositions on a scalar basis. Considering that every human has a different taste in music which in turn is dependent on cultural influences, personal experiences, social periphery and probably also the mood of the person, there apparently can not be a universal fitness function for music. The approach in the proposed system is a configurable, modular fitness function which is optimized in respect of statistical measures and structural features. The system implements a multi-objective optimization process that aims to minimize the total absolute distance from optimum values for all modules, which can be weighted individually. The author is currently investigating which module configurations yield musically appealing results.

9.1 Statistical Fitness Function Modules

Implemented fitness function modules regarding statistical musical measures are listed in Table 1. All measures starting from the third row can be applied to either the global composition or to a single voice (respectively instrument) in the piece. Some modules are explained in greater detail below.

Table 1. List of statistical fitness function modules

Name	Description
Duration	Optimizes the piece to span a specified duration in beats or measures
Number of voices	Biases the system to favor compositions with a given number of instruments or voices
Note duration	Optimizes note length average and standard deviation
Rest duration	Optimizes rest length average and standard deviation
Note duration ratio	Considers the ratio between the total duration of notes and rests
Global dissonance	Analyzes simultaneously sounding notes in order to compute an average dissonance value and its standard deviation
Dissonance distribution	Considers simultaneously sounding notes and aims for a given distribution of interval occurrences
Dissonance in rhythmic context	Optimizes dissonance values depending on the point of time they appear in a measure
Interval leap distribution	Analyzes consecutively sounding note intervals and aspires a given distribution of interval leaps
Chord compliance	Checks the relative occurrence of notes matching their context harmony
Scale compliance	Aims for a relative occurrence of notes matching a scale corresponding to the context harmony or tonal center

9.2 Dissonance Analysis

A measure to compare the perceived dissonance of two simultaneously sounding notes is the Tenney Harmonic Distance or Tenney Height, defined as $log_2 ab$, where $\frac{a}{b}$ is the ratio between the two note frequencies [19, p. 407]. Tenney Heights

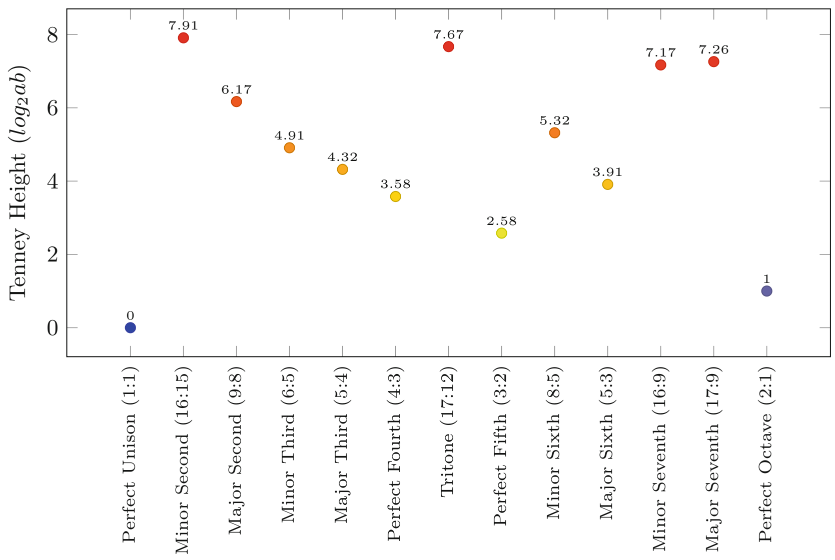

Fig. 2. Dissonance values of simultaneously sounding note intervals within an octave

for common intervals are visualized in Fig. 2. To compute a dissonance value of more than two simultaneously sounding notes, all combinations of intervals are analyzed. For example for an A minor chord the intervals A-C, A-E and C-E are analyzed. The dissonance value is then determined by calculating the average dissonance value of all note combinations in the chord. The system is capable of optimizing compositions regarding their global average dissonance value and its standard deviation. Furthermore the system can be instructed to aim for a given frequency distribution of simultaneously sounding intervals. In order to be independent from the total number of notes in the piece, the relative frequencies (i.e. the number of times an interval appears divided by the total number of intervals detected) can be specified as part of the fitness function.

Another approach is to optimize simultaneously sounding note intervals depending on the rhythmic context (i.e. the point of time they appear in a measure). In order to address points of time in a measure (i.e. pulses in a metric context) independently from the beat signature, the pulses are given numbers according to their importance in the metric context. The formula, yielding the pulse strengths of any multiplicative metre, was developed by Barlow [20, pp. 44–47]. For the quarter notes of a $\frac{4}{4}$ beat, for example, the formula yields the pulse strengths (or "indispensabilities", as referred to by Barlow) `3 0 2 1`. The higher the number, the higher is the importance of the pulse. For a metre with 5 pulses the formula produces the series `4 0 3 1 2` and for the six eighth notes in a $\frac{3}{4}$ bar `5 0 3 1 4 2`. A distribution of dissonance values depending on the pulse indispensability can be specified as fitness function objective.

9.3 Interval Leap Analysis

Not only is it functional to analyze simultaneously sounding intervals, but also to take consecutively played intervals into account. Therefore, the system compares all pitches of notes played in direct succession. Analyzing the distribution of successively played intervals for human compositions, it is remarkable that interval leaps of two semitones appear far more often (with a frequency of occurrence up to nearly 40 % dependent on the musical style) than intervals of 1 or 0 semitones [21, p. 218ff.]. In general the probability of intervals greater than two semitones decreases with ascending interval size with local minima at tritones and major sevenths as well as a local maximum for octaves. It is possible to define a target distribution of interval leaps as part of the fitness function for ascending as well as descending intervals (i.e. the distribution does not necessarily have to be symmetric).

9.4 Structural Fitness Function Modules

Since compositions are represented using tree-structured model instances, it is possible to define additional fitness function modules considering structural features of the model as shown in Table 2.

Table 2. List of structural fitness function modules

Module	Description
Tree depth	Controls the maximum tree depth of the model
Superfluous elements	Aims to eliminate nodes in the model which are syntactically allowed but semantically obsolete
Modifier ratio	Prefers compositions with a specified ratio of modifiers and constraints. The higher the ratio, the more compositions in which existing material is reshaped are preferred
Reference count	Favors compositions which reuse existing material by referencing fragments from different places in the model
Canonic pattern detector	Detects structures in which a fragment is successively referenced from different voice contexts (as applied in canons or fugues, for example)
Variation pattern detector	Promotes compositions repeating existing musical material in a varied form

10 Mutation and Crossover Operators

The following basic mutation operators are applied: node additions, replacements and removals are carried out for all types of nodes. The elements are either added as child nodes to existing ones or inserted between two already existing nodes. Replacements and removals can either happen for a single node (node replacement mutation) or whole subtrees (subtree mutation) [22]. For rhythmic constraints the following specialized mutations are applied: adding, removing or replacing single rhythmic notes or rests and turning rhythmic notes into rests and vice versa. For pitch constraints individual pitches are added, removed or replaced.

The system applies three different crossover operators. The first one swaps random subtrees of two compositions determined by roulette wheel selection. Another operator selects on average 50 % of the nodes of a specific type in a model. Nodes of the same type are randomly selected from the other model and exchanged with the previously selected ones. The third operator works similarly to the second one, though it additionally mixes sequences (e.g. pitches or rhythmic notes) contained in the nodes. During the evolution the system assures that neither the root node nor any of the locked model nodes (as described in Sect. 8) are replaced or removed, that no referential cycles are produced and that referenced fragments are copied to the target models recursively.

Note that the system does not differentiate between terminal and non-terminal nodes like in conventional genetic programming. In fact, all constructable models that have a root node are syntactically correct. Models that produce little or no output are semantically obsolete and will become extinct quickly as they are considered inferior in terms of fitness.

11 Results

The system successfully evolves compositions largely complying with the requested features. An example fitness function configuration is shown in Table 3. Although the compositions meet the statistical requirements, the pieces are only partially aesthetically pleasing. Evolved pieces clearly become more appealing when optimizing for high ratios of chord compliance and scale compliance in combination with a normally distributed, symmetric frequency distribution of consecutive intervals. This aligns with previous research with systems producing musically pleasant results by considering notes in a harmonic context and thereof derived scale context (e.g. [5,6,13,15]). When increasing the allowed standard deviation for note durations, the rhythms sound unstructured and random. This can be improved by implementing fitness functions analyzing pulse strengths in the metric context or considering approaches proposed by Horowitz [7]. Another weakness of the system is probably that there are no restrictions regarding the arrangement of the model nodes. A layer-based approach, where nodes of the same hierarchy level have the same type (with less strict conditions at the leaf node levels) similar to Fig. 1 could help to organize the evolved compositions. Then it is possible to develop enhanced crossover and mutation operators. The ones used in the current implementation seem to be too generic for the task of musical composition, so “musically intelligent” operators are required (as proposed by [6,13,15]). The system can be further improved by introducing more specialized modules, such as algorithms considering voice leading rules, harmonic progressions and redundancy measures of several musical aspects.

Table 3. Example fitness function configuration

Module	Target value(s)
Duration	16 measures
Average note duration	0.25 (quarter note)
Note duration σ	0.15
Note duration ratio	80 %
Global average dissonance	3.5
Superfluous elements	0
Interval leap distribution	Symmetric distribution centered at perfect unisons: 5 % with ± minor seconds: 10 %, ± major seconds: 17.5 %, ± minor thirds: 5 %, ± major thirds: 5 %, ± perfect fourths: 2.5 %, ± perfect fifths: 2.5 %, ± perfect octaves: 5 %
Chord compliance	60 %
Scale compliance	90 % relative to default scale matching the harmony

12 Conclusions and Future Work

The application of genetic programming in the field of automated music generation is promising considering that the system is capable of generating short musical pieces which are at least tonally pleasing. Improvements are still to be implemented for other aspects such as rhythms and harmonic progressions. Another goal to be pursued is to extend the system in such a way that it generates longer pieces with multiple sections which have different statistical and structural properties. The next ambition is to establish mechanisms which connect different sections in a musically satisfying way. Furthermore additional node types for relative harmonic constraints and modifiers are planned as well as a real-time capable compiler for direct playback of the compositions.

References

1. Collins, N.: Introduction to Computer Music. Wiley, Chichester (2010)
2. Nierhaus, G.: Algorithmic Composition: Paradigms of Automated Music Generation. Springer, New York (2009)
3. Fogel, D.B.: Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. Wiley, Hoboken (2006)
4. Horner, A., Goldberg, D.E.: Genetic algorithms and computer-assisted music composition. In: Belew, R., Booker, L. (eds.) Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 437–441. Morgan Kaufmann, San Mateo (1991)

5. Biles, J.A.: GenJam: a genetic algorithm for generating jazz solos. In: Proceedings of the 1994 International Computer Music Conference, ICMA, San Francisco, pp. 131–137 (1994)
6. Biles, J.A.: Improvizing with genetic algorithms: GenJam. In: Miranda, E.R., Biles, J.A. (eds.) Evolutionary Computer Music, pp. 137–169. Springer, London (2007)
7. Horowitz, D.: Generating rhythms with genetic algorithms. In: Proceedings of the 1994 International Computer Music Conference, ICMA, San Francisco, pp. 142–143 (1994)
8. McIntyre, R.A.: Bach in a box: the evolution of four part baroque harmony using the genetic algorithm. In: Proceedings of the IEEE Conference on Evolutionary Computation, vol. 14, No 3. IEEE Press, New York, pp. 852–857 (1994)
9. Horner, A. and Ayers, L.: Harmonization of musical progressions with genetic algorithms. In: Proceedings of the 1995 International Computer Music Conference, ICMA, San Francisco, pp. 483–484 (1995)
10. Jacob, B.: Composing with genetic algorithms. In: Proceedings of the 1995 International Computer Music Conference, ICMA, San Francisco, pp. 452–455 (1995)
11. Jacob, B.: Algorithmic composition as a model of creativity. Organised Sound **1**(3), 157–165 (1996)
12. Johanson, B., Poli, R.: GP-Music: an interactive genetic programming system for music generation with automated fitness raters. In: Koza, J.R., et al. (eds.) Genetic Programming 1998: Proceedings of the Third Annual Conference (GP 1998), pp. 181–186. Morgan Kaufmann, San Francisco (1998)
13. Marques, M., Oliveira, V., Vieira, S., Rosa, A.C.: Music composition using genetic evolutionary algorithms. In: Proceedings of the IEEE Conference on Evolutionary Computation 2000. IEEE Press, New York (2000)
14. de la Puente, A.O., Alfonso, R.S., Moreno, M.A.: Automatic composition of music by means of grammatical evolution. In: Proceedings of the 2002 Conference on APL, pp. 148–155. ACM Press, New York (2002)
15. Fox, C.: Genetic hierarchical music structures. In: Proceedings of the 19th International FLAIRS Conference. AAAI Press, Menlo Park (2006)
16. Waschka II, R.: Composing with genetic algorithms: GenDash. In: Miranda, E.R., Biles, J.A. (eds.) Evolutionary Computer Music, pp. 117–136. Springer, London (2007)
17. Boden, M.A.: Creativity and computers. In: Dartnall, T. (ed.) Artificial Intelligence and Creativity: An Interdisciplinary Approach, pp. 3–26. Kluwer Academic Publishers, Dordrecht (1994)
18. Boden, M.A.: Creativity and Art: Three Roads to Surprise. Oxford University Press, Oxford (2010)
19. Deza, M.M., Deza, E.: Encyclopedia of Distances. Springer, Heidelberg (2013)
20. Barlow, C.: On musiquantics. Technical report, Johannes Gutenberg-Universität Mainz (2012)
21. Patel, A.D.: Music, Language, and the Brain. Oxford University Press, New York (2008)
22. Poli, R., Langdon, W.B., McPhee, N.F., Koza, J.R.: A Field Guide to Genetic Programming (2008). Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk