Submitted exclusively to the *Journal of Mathematics and Music*
Last compiled on November 2, 2016

Data-based melody generation through multi-objective evolutionary computation

Pedro J. Ponce de León[a], José M. Iñesta [a]*, Jorge Calvo-Zaragoza[a], and David Rizo[a]

[a]*Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante, Spain*

()

Genetic-based composition algorithms are able to explore an immense space of possibilities, but the main difficulty has always been the implementation of the selection process. In this work, sets of melodies are utilized for training a machine learning approach to compute fitness, based on different metrics. The fitness of a candidate is provided by combining the metrics, but their values can range through different orders of magnitude and evolve in different ways, which makes it hard to combine these criteria. In order to solve this problem, a multi-objective fitness approach is proposed, in which the best individuals are those in the Pareto-optimal frontier of the multi-dimensional fitness space. *Melodic trees* are also proposed as a data structure for chromosomic representation of melodies and genetic operators are adapted to them. Some experiments have been carried out using a graphical interface prototype that allows one to explore the creative capabilities of the proposed system. An Online Supplement is provided where the reader can find some technical details, information about the data used, generated melodies, and additional information about the developed prototype and its performance.

Keywords: machine learning; evolutionary algorithms; composition; melody; tree representation; multi-objective optimization.

2010 Mathematics Subject Classification: 68T05
2012 Computing Classification Scheme: applied computing; sound and music computing; computing methodologies; learning from critiques; computing methodologies; genetic programming;

1. Introduction

Genetic algorithms (Holland 1975), or more generally evolutionary techniques, are inspired by the biological evolution of living beings and natural selection. It is indeed an optimization technique in which a population of individuals, that represent different possible solutions, are subjected to processes that mimic natural crossovers and mutations, and a selection stage that decides which individuals best fit to solve the problem. Those individuals are allowed to procreate a new generation of, supposedly, better individuals (solutions) until convergence. This basic idea covers a wide range of applications in engineering and machine learning (Goldberg 1989).

Music composition can be seen as an optimization process, in a way that the human composer implicitly searches in the space of all possible musical compositions for a composition that satisfies his or her own artistic criteria (Dostál 2013). Evolutionary-based algorithmic composition has always been well appreciated by researchers and composers

*Corresponding author. Email: inesta@dlsi.ua.es

due to their capability to stochastically explore an immense space of possibilities (Todd and Werner 1998). This is especially due to the mutation operator, that enables them to model something hardly computable like "inspiration" (something very vague that can be viewed as "finding something of artistic value by chance"). An interesting overview of the field can be found in the work of Miranda and Biles (2007).

In this paper, we aim to use machine learning techniques based on a training set of examples to implement an *automatic fitness function*. In the literature, this is considered to be one of the subtlest aspects of such systems (de Freitas, Guimarães, and Barbosa 2012). For that, we will try to assess different properties of melodies, some of them statistical, some music theoretical, combining them via the multi-objective optimization paradigm. This approach permits us to define a *fitness space*, rather than just a fitness function, where values that may be disperse can be put together in an elegant and easy way, in order to guide the composition process of the evolutionary system.

1.1. *Background: evolutionary computation in music composition*

Evolutionary computation in general, and genetic algorithms in particular, have been used in arts over the years (Romero and Machado 2007), with growing interest. Biology-inspired computing techniques offer a metaphoric context for using nature as a source of inspiration, providing ideas and methods for finding other ways of solving problems and discovering new ways of creation.

In the case of algorithmic music composition, from the pioneering works of Xenakis (1992) or Hiller and Isaacson (1959), three main approaches have been used (Nierhaus 2008):

a) *stochastic methods*, in which probability distribution functions are utilized to generate events that can be mapped to pitches, durations, or any other kind of psychoacoustic properties of music (like timbral or dynamical, for example);
b) *deterministic methods*, that are based on rules (usually derived from musical methods or from mathematical theories) or sonification methods, by which mathematical functions (fractals, non-linear dynamic systems, chaos theory, cellular automata, etc.) or signals and models coming from other areas of knowledge (physics, biology, geology, etc.) are used as data for being mapped to music parameters by using systematic transformations; and
c) *machine learning (or data-driven) methods*, in which by using any kind of adaptive algorithm, usually based on examples of a music style or genre, the system makes its own model that is able to generate new data related to those previously used for training.

Of course, hybrid systems have also been proposed, providing very interesting results (Cope 1991), exploiting the best qualities of each approach, at the cost of increasing the system's design complexity.

Genetic algorithms (GA) belong to the third category of methods, where the composition process is guided somehow, aiming towards a particular kind of music. The main constituent parts of a GA are a representation for chromosomes (the candidate solutions), an initial population of chromosomes, a set of operators to generate new candidates, an evaluation function, and a selection method.

The most usual representation of a musical chromosome is a vector of numbers (often in binary code), representing pitch and duration information. Some authors have explored other approaches, like analytic functions (Laine and Kuuskankare 1994), languages like `abc` (Oliwa 2008), or motives represented by patterns (Liu and Ting 2015). More often,

trees have also been used (Phon-Amnuaisuk, Law, and Ho 2007; Komatsu et al. 2010; Hofmann 2015), but always in the context of a genetic programming approach, where the trees are representing a string in a language, formally represented by a syntax, so they are actually, *parsing trees*. In the present work, we introduce the use of *melodic trees* for this task, as a structured representation of a monophonic series of notes. Although the trees in the GA implementation are defined by a grammar, they represent structural melodic trees. We will need to define the proper genetic operators, adapted for this representation.

1.1.1. Style-oriented approaches

Music style is a vague concept that can be interpreted in a number of ways, all of them subjective. Some authors consider the music style as the set of music compositions that provide a given emotion (aggressive, calm, romantic, etc.) while others focus on music genre as the set of authors or pieces sharing some common characteristics.

A style-guided genetic composition scheme can be achieved by using machine learning algorithms and a corpus of training data from a music genre, a given author, or even particular tastes, to implement fitness functions or any kind of guidance in the composition process (Weale and Seitzer 2003; Alfonseca, Cebrián, and Ortega 2007; de Freitas, Guimarães, and Barbosa 2012). Thus, the same overall scheme can be used to generate very different melodies just changing the training data. Such a corpus must be tagged before. This is usually done by a human, so a subjective factor is introduced.

1.1.2. Human supervised fitness functions

One of the key aspects of every evolutionary system is the design and implementation of the fitness functions. This is particularly tricky when the individuals that are being evaluated are artworks, although there are systems like GenDash (Waschka 2007), that is a GA without fitness evaluation, where the individuals are selected at random to the next generation, avoiding what is considered by the author as the *bottleneck* of evolutionary systems. There is not an ultimate evaluation system; even the impressions of some persons listening to the same music can be different and, therefore, the results of automatic composition are subjective.

The aid of human skills for that permits to model the musician's taste and even web-based human assessment of the individuals have been described in the literature (Putnam 1996; Tokui and Iba 2000; Fu et al. 2006; Ayesh and Hugill 2005; Özcan and Erçal 2007; Zhu, Wang, and Wang 2008; Koga and Fukumoto 2014). Sometimes, the user is needed because the system is oriented to user interaction, like in the case of the GenJam system (Biles 1994), but sometimes the user is needed because the system is expected to accelerate the optimization by the user actions (Koga and Fukumoto 2014), like evaluating, removing, or even changing the melodies generated at a given generation. In (Zhu, Wang, and Wang 2008), the system is focused on capturing listener feelings, like happiness and sadness, although the user works in cooperation with a rule system to calculate the fitness values.

In most of these cases, the populations are reduced to a very small number of individuals. It is obvious that evaluating hundreds or thousands of music works is not feasible, even when many users are involved, like in (Ayesh and Hugill 2005). Anyway, when human critics are used, these evolutionary systems can produce pleasing and sometimes surprising music, but usually after many tiresome generations of feedback (Todd and Miranda 2003).

1.1.3. Autonomous fitness functions

Given the problems and limitations of human-supervised assessment, there are many works in the literature that deal with the problem of designing fully automatic evaluation functions that can operate without human intervention. This idea of automatic evaluation of music (and artworks in general) has received the name of a virtual "critic" (Machado et al. 2003).

Some works (Wiggins et al. 1998) use music-theoretical knowledge as a fitness function, like Moroni et al. (1994), who use a fitness criterion that takes into account melodic fitness, harmonic fitness, and voice range fitness. Other previous works try to induce musical structure from a corpus to get new melodies (Cope 1991). An extension of this procedure was done by Spector and Alpern (1995), who applied a hybrid rule-based and neural network critic trained with a corpus of well-known works, and Baluja, Pomerleau, and Jochem (1994) who also used a similar combination of neural and genetic techniques.

Language models, like n-gram models (Lo and Lucas 2007; Herremans, Sörensen, and Conklin 2014) have been used as trainable music critics to impose constraints on the space of pitches and intervals that can be explored by the genetic algorithm. This idea will be also explored in the present work.

Neural techniques have been widely used to implement fitness functions. For example, in (Sheikholharam and Teshnehlab 2008) recurrent networks are used both for generating pitch sequences and for evaluation, in particular to cope with the problem of long-term melodic relations, which is impossible to capture using language models, like n-grams. In (Göksu, Pigg, and Dixit 2005) multilayer perceptrons are trained to recognize music of a given genre. Also, Self-Organised Maps were used in (Phon-Amnuaisuk, Law, and Ho 2007), exploiting their ability to map music representations into feature visualization spaces.

1.1.4. Multiple fitness values

A valuable approach that is currently gaining ground is the use of different fitness functions, each of them specialized in a particular aspect of evaluation, providing different values that are combined to get a better combined fitness. The key point here is to find an appropriate way of combining those values.

For example, in (Ayesh and Hugill 2005), the selection process is guided by the responses of the users within an interactive process. The individuals are rated by 3 different factors that are combined by simple summation into a single fitness value. In the context of automatic fitness, in (Özcan and Erçal 2007), multiple musical aspects are evaluated: 10 fitness functions are computed and then combined by using a weighted sum of their respective evaluations. Hofmann (2015) introduces the concept of multi-objective evaluation, that is central in the present work, but what that author actually did is look for the optimum of each of the 11 statistical and 6 structural fitness function modules utilized. Each function is considered as a vector dimension and the system tries to optimize a weighted distance from each individual to the optimum vector.

A proper strategy for combining the different values provided by a number of fitness evaluation functions is needed. Summing them is not appropriate since they can differ even in orders of magnitude, and therefore it is not straightforward to find a suitable way to merge them into a single criterion.

It is at this point where a multi-objective optimization (MOO) approach to establish an ordering relation among individuals is useful, taking into account the whole set of criteria simultaneously. Although the combination of multi-objective optimization with genetic algorithms has been widely studied Purshouse et al. (2013), the proposed application

for music composition is innovative. In the context of music recognition, a previous work (Vatolkin 2013) used MOO for prediction of high-level music categories, such as genres, styles, or personal preferences. That author measured the impact of evolutionary multi-objective audio feature selection on the classification performance, leading to a significant reduction in the classification error.

1.2. *The present approach*

In the present approach, two main novelties are introduced. The first novelty is the use of a tree data structure (Rizo 2010) for coding the individuals. Although, as discussed above, trees have been used before in evolutive composition, they have been always used for coding a programming language string as a formal grammar, so they are actually a tool for implementing genetic programming methods. Standard genetic programming mutation and crossover operators are applied on this data structure. In our case, a tree is a representation of a melody that will compete for survival. The target will be melodies of a fixed length in bars (8 bars by default). A proof of concept of this approach was already presented in a former work (Hidden 0000).

The second novelty of the present paper is the combination of different fitness functions by means of a multi-objective optimization method, that is able to combine multiple and diverse values in order to rank the individual in a single fitness space. This way, the individuals are not ranked by their fitness values, but instead, by their position in the fitness space. This way an optimal locus is defined by some of the individuals that will be ranked as the best adapted. After them, other individuals will be in the ranking by selecting the best after removing the former ones from the fitness analysis. And so on.

The fitness dimensions in this space will be provided by example-based machine learning models, some of them based on statistical properties of the melodies, others based on statistical language models, and others designed on top of musicology rules, trained from the melodic examples contained in a provided training set.

2. Methods and tools

2.1. *Melody tree representation*

There are several ways of coding a melody (Selfridge-Field 1997). In this work, we use the tree encoding proposed by Rizo (2010), extended with the addition of levels for sections and measures. That model is based on the representation of the rhythm structure as a tree. Each bar is represented as a sub-tree, depending on its meter. All these sub-trees are linked together in a common tree, with a binary or ternary arity, depending on the kind of meter. The level of a node determines its duration: the root represents the duration of the whole melody, the nodes of the next level represent a division of the upper node that, together, sum up its same duration.

Pitch codes are found in the leaves of the tree. Any kind of absolute or relative pitch could be used. For this particular application, pitch classes, together with rest and continuation symbols, have been used: $p_i \in \{\text{C,C}\sharp/\text{D}\flat\text{,D},\ldots,\text{B}\} \cup \{rest, continuation\}$. Some pitch classes from the whole chromatic scale can be selected, if required.

These two extra symbols are used for coding rests and the continuation of a note beyond its natural duration, like in the case of syncopations, ties or dotted notes (see Figure 1 for an illustration). The left to right ordering of the leaves encodes the onset times of the notes in the melody.

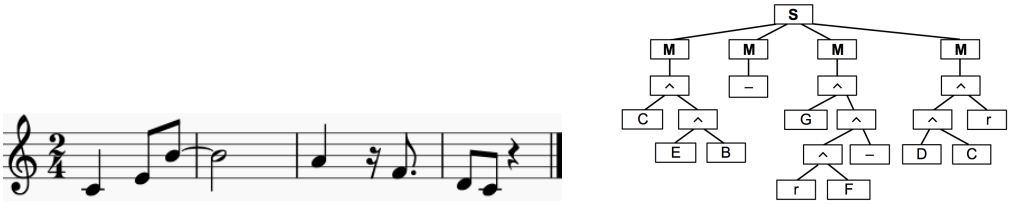

Figure 1. A short melody example and its corresponding tree representation as a section of what can be a longer melody. The labels in the leaves correspond to pitch classes, including 'r' for rests, and '–' for continuations. 'S' denotes a *section*, 'M' is a *measure* node, and '∧' represents every inner node (beat or sub-beat) that splits.

Without loss of generality, we shall restrict this study to binary trees, representing binary subdivisions in measures and beats, diatonic pitches from the major scale, and maximum depth for the trees corresponding to a sixteenth note.

Another interesting feature of this coding is that it is very easy to change the structure of the melody tree representation in the genetic algorithm, by changing the underlying grammar in a configuration file. Currently, the structure that is being used is as follows:

```
Melody := Section Section
Section := Measure Measure Measure Measure
Measure := (Beat | Symbol) (Beat | Symbol) (Beat | Symbol) (Beat | Symbol)
Beat := (SubBeat | Symbol) (SubBeat | Symbol)
SubBeat := Symbol Symbol
Symbol := Pitch | Rest | Continuation
Pitch := C | D | E | F | G | A | B
Rest := r
Continuation := _
```

but it can be changed by specifying another structure to the system.

Note that one of the main advantages of this representation is that it is not possible to generate invalid representations when crossover operations mix the chromosomes of the different individuals, which is very useful in the evolutionary paradigm.

2.2. *Evolutionary algorithms with trees*

Each individual of our population will be a 4/4 meter, 8-bars long, monophonic melody coded as a binary tree, as described above.

The size of the population is a free parameter of the system. The population is initialized at random following the tree generator procedure proposed by Koza (1992), and the evolutionary search loops. Hypothetically, better compositions are obtained through the evolutionary crossover and mutation operators. Individuals are evaluated and selected so that only those who encode the best solutions, according to their *fitness*, can survive for the next generation.

The crossover operator creates two individuals (children) from two existing ones of the previous generation (parents). The idea is that the children contain features of their parents, but represent new melodies. For this, one non-leaf node is randomly selected in each of the parents. The whole sub-tree hanging from the selected node is interchanged by that in the other parent tree. Due to the representation utilized, the bar length is preserved. Measure nodes can also be interchanged by crossover (see Figure 2).

The mutation operator is applied on each individual separately. It consists of randomly changing its chromosome. Each node of the tree has a small probability of being modified. In that case, the mutation operator generates a sub-tree whose root is compatible with

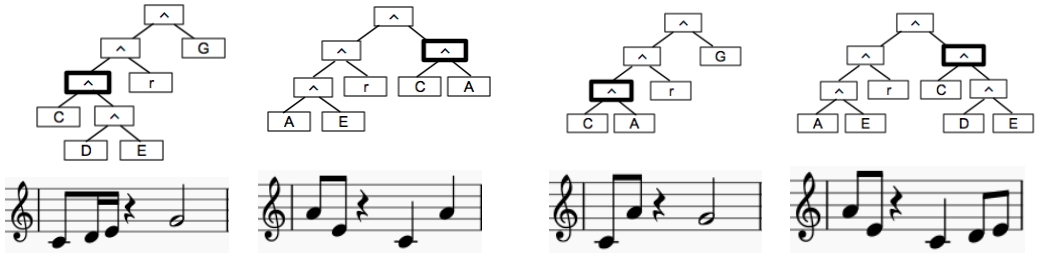

Figure 2. Illustration of the crossover operator applied on 1-bar trees in 4/4, and the effect on the melodies they represent. The two parents are those displayed on the left, and on the right the two children are shown. A node is selected for both parents (thick node borders) and the sub-trees are interchanged.

the kind of the selected node, according to the grammar structure displayed above in Section 2.1.

By applying these operators, an offspring is created from a population of size N. It is done as follows: 90% of the times, a series of two tournaments of seven individuals, randomly chosen, are carried out. The winners of each are used as parents for a crossover process, breeding two new individuals. This procedure is repeated until generating N individuals. The new individuals are assessed and ranked together with their parents. In the remaining 10% of the times, the population is simply cloned. Note that, whatever the case, the size of the original population is doubled.

From the new individuals generated by the crossover, the mutation pipeline is computed. It consists in performing tournaments of 7 individuals for which the winner is mutated. At the end, each mutated individual will be evaluated by the set of selected fitness metrics, as it will be described in the next sections.

At the end, only the best N individuals among the parents and the offspring are kept to maintain the size of the population. The survivals are considered for the next generation. The process starts over again, until reaching a maximum number of iterations, that can be set by the user.

2.3. *Automated feature extraction and fitness*

One of the goals of this work is to study how to use different machine learning techniques to rate evolutionary generated music according to the parameters tuned from melodies in a set, that supposedly share some common properties, like genre, style, mood, etc. For that, the evolving melodies have to be described using the features that are the input to those assessment systems.

In this regard, we have considered three families of fitness evaluations.

a) Global statistical descriptors that embed the melodic tree in a vector space representing the whole melody as a point. This way, similarities can be computed as distances in such a space;
b) Local n-gram probabilistic models for assessing how likely some notes can be used together in a short-term context; and
c) Music theory-based melodic evaluations, in order to introduce domain-specific knowledge in the fitness evaluation process.

These three families comprise a total of 12 separate fitness functions. By using these functions, we create a *fitness space* in which the multi-objective optimization algorithm is used to select the best individuals. The final user may decide to use all or a part of those functions. For that, a graphical interface has been designed for experimentation

that will be introduced in the experiments and results section.

2.3.1. Global statistical evaluations

The first model is the *global shallow description scheme* (Ponce de León and Iñesta 2007). In this case, an individual melody is represented as a vector of statistical descriptors that are related to melodic, harmonic, and rhythmic properties of the melody, by analysing how pitches, silences, durations, inter-onset intervals, pitch intervals, diatonic notes, and syncopations are distributed in the melody (the reader is referred to the paper cited above for details).

In this particular implementation we have selected the following descriptors, grouped by the kind of musical property they are describing.

- Notes: total number of notes and average number of notes per beat.
- Rests: total number of silences.
- Pitches: average pitch relative to the lowest one and typical deviation.
- Pitch intervals: largest, range of interval sizes, average, typical deviation, most repeated one, and number of different intervals.
- Durations: occupation rate (sum of note duration versus total duration), duration range (from shortest to longest), average duration relative to the shortest note, and typical deviation.
- Inter-onset intervals: range (from shortest to longest), average relative to the shortest one, and typical deviation.
- Rest durations: range (from shortest to longest), average relative to the shortest rest, and typical deviation.
- Syncopations: total number of syncopated notes and rate (syncopated versus total number of notes).

These descriptors create a vector $\mathbf{x} \in \mathbb{R}^{23}$. This way, the melodies $\mathbf{x}_i$ in the training set $\mathcal{X}$ are represented as a cloud of points that are our target style. Under the hypothesis that new melodies of that style should be close to those already existing, the distance from the vector $\mathbf{m}$ representing each evolving melody to the cloud is given as a measure of style for it.

The global centroid of the cloud, $\mathbf{c} = \frac{1}{|\mathcal{X}|}\sum_i \mathbf{x}_i$, can be a target for that matter, but if all the population $\mathbf{m}_j$ tries to minimize the distances $d_c = d(\mathbf{m}_j, \mathbf{c})$, this may not favor diversity. For that, the distance to be minimized should be more local, and we have considered the distance to a neighborhood of size k of each individual, in such a way that a local centroid is computed as $\mathbf{c}_k = \frac{1}{k}\sum_{i=1}^{k} \mathbf{x}_i$ and the distance to be minimized is $d_k = d(\mathbf{m}, \mathbf{c}_k)$. The problem with this approach is that the point $\mathbf{m}$ may be far from the global centroid, that is supposed to be a good reference for the style of music being represented. Taking into account the pros and cons of the two approaches, both distances, d_c and d_k, have been used for computing fitness functions to be minimized.

In any case, the features computed are very different, so in order to cope with the variety of ranges and dispersions they may show, the Mahalanobis distance has been used:

$$d_c^2(\mathbf{m}, \mathbf{c}) = (\mathbf{m} - \mathbf{c})^T \Sigma^{-1} (\mathbf{m} - \mathbf{c})$$
$$d_k^2(\mathbf{m}, \mathbf{c}_k) = (\mathbf{m} - \mathbf{c}_k)^T \Sigma^{-1} (\mathbf{m} - \mathbf{c}_k)$$

where Σ is the covariance matrix in $\mathbb{R}^{23\times 23}$ computed from $\mathcal{X}$.

Once the proposed distances are computed as $d^2 = \{d_c^2, d_k^2\}$, the fitness values $F_{\text{global},d}$ are normalized by using

$$F_{\text{global},d} = 1 - \frac{1}{1+d^2} = \frac{d^2}{1+d^2} \,.$$

This value is what the genetic algorithm will try to minimize.

2.3.2. Local musical n-gram evaluations

The fitness functions $F_{\text{global},d}$ provide a global statistical context for the possible melodies, driven by the training examples represented in the feature vector space. Nevertheless, there are still a huge number of melodies that could give the same global statistics, so a more local, restrictive, view is needed. This way, more restrictive constraints are imposed on adjacent notes. We could say that proximity to the cloud representing the training set provides a necessary condition but not a sufficient condition for a melody to sound style-like.

An n-gram model is a statistical language modeling technique widely used in natural language processing, that works under the Markov assumption that the probability of a symbol depends only on the probability of a short-term history of previous symbols. Each parameter is the probability of a symbol s_i to appear after seeing the sequence of $n-1$ symbols $s_{i,n-1} = s_{i-n+1}s_{i-n+2}\dots s_{i-1}$. This probability is estimated from the relative frequency of the string $s_{i,n} = s_{i,n-1}s_i$ in the training set, as

$$P(s_i|s_{i,n-1}) = \frac{\mathcal{N}(s_{i,n-1}s_i)}{\sum_{s\in\mathcal{A}}\mathcal{N}(s_{i,n-1}s)} \,.$$

Here $\mathcal{A}$ is the alphabet of possible symbols from where s_i can take values, and $\mathcal{N}$ is a counting function. One of the problems with this approach is that, if a sequence of n symbols does not appear in the training set, its probability is zero and therefore, the probability of any sequence in which it may appear will also be zero. To solve this problem, the probability distribution is smoothed by using the Kneser-Ney method (Kneser and Ney 1995), reallocating some probability mass from the 3-grams or 4-grams utilized, to simpler unigram models.

Once we have the n-gram model probabilities inferred, we can assign a probability to a new sequence S of $|S|$ symbols by computing

$$P(S) = P_{n-1}(s_1\dots s_{n-1})\prod_{i=n}^{|S|} P(s_i|s_{i,n-1}) \,,$$

where $P_{n-1}(s_1\dots s_{n-1})$ denotes the probability of a string beginning with the $n-1$ symbols that cannot be computed with the n-gram model, and is estimated by looking for how often the strings in the training set begin with that sub-string, $\mathcal{N}(s_1\dots s_{n-1})/\mathcal{N}(S)$.

The length of the string coding the melody, $|S|$, is an issue in this case, because the lower the number of n-grams, the higher the probability will be, and vice-versa. To cope with that problem, the length of the melody is also modeled as a Poisson distribution

$$p(|S|,\lambda) = \frac{e^{-\lambda}\lambda^{|S|}}{|S|!}$$

where the expected value for the length, λ, is estimated from the melodies in the training set, $\mathcal{X}$. So eventually, the probability of a sequence S will be

$$P(S) = p(|S|, \lambda) P_{n-1}(s_1 \dots s_{n-1}) \prod_{i=n}^{|S|} P(s_i | s_{i,n-1}) \,,$$

For using these models in our case, we construct strings of symbols (n-words) (Doraisamy and Rüger 2003) from n consecutive notes in the tree representation as a string of $n-1$ intervals coded by symbols in an alphabet $\mathcal{A}_P$ and $n-1$ inter-onset ratios (IOR) coded by symbols in an alphabet $\mathcal{A}_D$ (see details in the cited paper). This way, for example, a series of $n = 3$ notes is coded in a *coupled* representation of relative pitches and durations by 4 characters in a '3-word'. There is also the possibility of *de-coupling* the representation by computing separately 3-words with the 2 intervals and with the 2 inter-onset ratios. This may be useful if the data available are limited, because for the coupled representation, the cardinality of the alphabet is $|\mathcal{A}| = (|\mathcal{A}_P| \times |\mathcal{A}_D|)^{n-1}$, while for the decoupled representations the cardinality of the alphabet is just $|\mathcal{A}_P|^{n-1}$ or $|\mathcal{A}_D|^{n-1}$. These lower cardinalities make the parameters easier to learn from the training melodies.

All possible n-words are extracted from an individual melody, except those containing a silence lasting four or more beats, that are ignored. From the melody coded as a series of n-words, the n-gram sequence, S, of n-words is analyzed and its probability $P(S)$ is computed, and the fitness function will be

$$F_{\text{local},n-\text{gram}} = -\log P(S)$$

for $n = \{3, 4\}$, and for the coupled (intervals and durations together) and decoupled (either for intervals or IORs) versions of the n-words coding.

2.3.3. Local bag-of-notes evaluations

The other approach to style modeling is similar to that above in terms of coding n notes into a series of characters by using a n-word, but focuses on how probable a string based on the probability of appearance of substrings of n-words is, rather than in the order of how they appear, like in the former case.

The same n-word based representation described above for a melody (only the coupled version for $n = 3$) is now evaluated probabilistically by a multinomial distribution model, that takes into account n-word frequencies in the string. In this model, each melody is encoded as a vector $\mathbf{s} \in \mathbb{N}^{|\mathcal{V}|}$, where $\mathcal{V}$ is the vocabulary made of the most frequent n-words found in the training set, and each component s_t represents the number of occurrences of the n-word w_t in the melody.

The same problem with the string coding length $|S|$ as in the former case is present here, so shorter notes would imply less products, and therefore, always higher probabilities, so a new Poisson distribution is estimated from the training data, $p(|S|, \lambda)$, by computing the average number of n-words found in 8-bar melody segments.

The probability that the whole series of n-words in the melody has been generated by the multinomial distribution found for the melodies in the training set $\mathcal{X}$ is

$$P(\mathbf{s}|\mathcal{X}) = p(|S|, \lambda)|S|! \prod_{t=1}^{|\mathcal{V}|} \frac{P(w_t|\mathcal{X})^{s_t}}{s_t!}$$

and the conditional word probabilities are estimated as

$$P(w_t|\mathcal{X}) = \frac{1+\mathcal{N}_t}{|\mathcal{V}| + \sum_{k=1}^{|\mathcal{V}|} \mathcal{N}_k} ,$$

where $\mathcal{N}_t$ is the sum of occurrences of word w_t in the melodies of the training set.

Like in the former case, once the $P(\mathbf{s}|\mathcal{X})$ is computed, the fitness function will be

$$F_{\text{local,multi}} = -\log P(\mathbf{s}|\mathcal{X}) .$$

2.3.4. Melodicity fitness evaluations

A complementary approach to the fitness evaluations described in the sections above is based on elements of music theory, that permits one to implement knowledge specific to the application. Music theory provides a general conceptual framework, that can be adapted to the data in a training set, by learning parameter values and distributions.

The kind of evaluations we are going to perform are two-fold: on one side, a melodic analysis that tags the notes in the melody as harmonic or non-harmonic (with sub-classes) and then a language model inferred from the training data is applied over the tags, and on the other side, a segmentation algorithm is applied and the number of segment is compared to the number of segments found in the training set for melodies of the same length.

Melodic analysis

Melodic analysis determines the importance and role of each note in a particular harmonic context. Thus, a note is classified as a harmonic tone ('H'), when it belongs to the underlying chord, and as a non-harmonic tone otherwise, in which case it should be further assigned to a category, such as *passing tone* ('P'), *neighbour tone* ('N'), *suspension* ('S'), and *appoggiatura* ('A'). There are more possible categories (see (Willingham 2013)), but the ones considered here are those based on the stability of the beat and the intervals before and after the analysed note. There are other kinds of non-harmonic notes but the features required for performing the analysis need harmonic information, like chords, that are not available for a monodic melody (see (Rizo, Illescas, and Iñesta 2015) for details).

A n-gram model is constructed from the series of melodic tags obtained when the fragments of 8-bar melodies in the training set are analysed. The same approach as in Section 2.3.2 is applied to the 5 tags, listed above $\mathcal{A}_A = \{$'H','P','N','S','A'$\}$ and the probability $P(A)$ of the analysis sequence $A \in \mathcal{A}_A{}^\star$ of an individual is computed. From it, $F_{\text{melodic},n} = -\log(P(A))$ is established as a fitness value to be minimized for $n \in \{3, 4\}$.

Melodic segmentation

The local boundary segmentation model (LBDM) (Cambouropoulos 2001) is a simple and well-known algorithm that permits one to partition a melody into segments using the sizes of intervals, the length of notes, and the length of silences. These three measures are weighted by normalized coefficients, w_i that tune the algorithm's behaviour. We have heuristically set them to $w_1 = 0.6$, $w_2 = 0.2$, and $w_4 = 0.2$, respectively, in order to give more importance to intervallic relations. The threshold over the boundary strength for local maxima detection was $\theta = 0.3$.

An excessive number of segments, σ, denotes a lack of coherence in the melody, while a very low number of segments is an indication of a dull, flat melody. In any case, the good number of segments depends on both the total length of the melody and the style of music. In order to adapt this to data, we have made a statistical study on how many segments per fragment are in the training set, considering melodies of the same length of those generated by the genetic algorithm (8 bars in this case). A new Poisson distribution $P(\sigma) = p(\sigma, \lambda)$ was estimated from the number of segments, and the fitness to be minimized will be $1 - P(\sigma)$, in order to favor a number of segments close to λ.

2.4. *Multi-objective optimization*

Without loss of generality, we are going to consider that optimizing a function refers to its minimization. Therefore, a typical optimization problem can be defined as finding an $\hat{\mathbf{x}}$ such that

$$\hat{\mathbf{x}} = \arg\min_{x \in \mathcal{S}} f(x)$$

in which $\mathcal{S}$ stands for the set of solutions that fulfill the implicit constraints of the problem.

Quite often, applications require us to have multiple functions or objectives to be optimized at the same time. In fact, a total number of $F = 12$ fitness functions have been proposed to implement the evaluation of a music composition. Then, the optimization problem is reformulated as

$$\hat{\mathbf{x}} = \arg\min_{x \in \mathcal{S}} \{f_1(x), f_2(x), \ldots, f_F(x)\} \ .$$

In this case, the definition of optimal solution cannot be directly taken from the scalar concept of the single-objective optimization. When multiple functions have to be optimized, simultaneously, optimality is defined through the concept of *dominance*.

Definition 2.1 A solution $\mathbf{x}_1$ is said to *dominate* another solution $\mathbf{x}_2$ if, and only if, the following conditions hold:

(1) The solution $\mathbf{x}_1$ is better or equal than $\mathbf{x}_2$ in every single objective function.
(2) The solution $\mathbf{x}_1$ is strictly better than $\mathbf{x}_2$ in any of the objective functions.

The set of non-dominated solutions, those for which no one dominates them, is referred as *Pareto frontier*. The elements within this set have the property of being Pareto-optimal, which means it is impossible to make an improvement in one criterion without making at least another one worse.

In this paper we are interested in developing an evolutionary algorithm that is able to optimize several functions simultaneously. Hence, the solution space is not explored exhaustively but new points are obtained through iterative generations of an evolutionary search. To make this process generate solutions that converge to the optimal values, it is necessary to establish an order relationship among individuals in the population that allows selecting those most promising in relation to the concept of Pareto-optimality. To this end, this work follows a multi-objective evolutionary scheme.

A number of ways of addressing this problem have been proposed over the last years. In a first batch of algorithms (Srinivas and Deb 1994; Schaffer 1985; Horn et al. 1994), non-dominated individuals of the population were kept through the evolutionary generations.

Although these strategies were able to find Pareto-optimal solutions, the set of surviving individuals tended to concentrate on a small portion of the target space, in which only one of the functions is actually optimized. That is why a second group of algorithms arose (Zitzler, Laumanns, and Thiele 2001; Deb et al. 2002), which also promoted the diversity of solutions throughout the Pareto frontier. An example of this algorithm is the Non-dominated Sorting Genetic Algorithm II, which will be used in this work. The next section briefly describes the operation of this algorithm given its interest in the present paper.

2.4.1. Non-dominated Sorting Genetic Algorithm II

Non-dominated Sorting Genetic Algorithm II (NSGA-II) is an efficient multi-objective optimization evolutionary scheme proposed by Deb et al. (2002). Although it was developed for genetic algorithms, NSGA-II can run on any evolutionary search that contains evaluation and selection processes, as it focuses on establishing a relationship of order or priority among individuals in the population according to their multi-valued fitness.

The general operation of the algorithm is to divide the individual into fronts of non-dominance. In the first front *the algorithm finds* those individuals are found that are non-dominated (Pareto frontier); in the second front, those individuals who would form a new optimal frontier in the absence of the individuals of the first one; and so on. These fronts are iteratively created until every single individual in the population is assigned to one.

Once this process is over a new relationship is established, but only among individuals belonging to the same rank. For this purpose, NSGA-II defines a *crowding distance* function to estimate the diversity that brings each of the individuals within that frontier. The idea is to favor those individuals that provide a higher variety of solutions.

Formally speaking, NSGA-II defines a partial relation among the individuals of the population. Let $P = \{p_1, p_2, \ldots, p_{|P|}\}$ be a population. We denote by $r : P \to \mathbb{N}$ the front assigned to an element of P and by $c : P \to \mathbb{R}^*$ the crowding distance function. NSGA-II establishes a partial order ($\succ$) among the elements of P such that

$$p_i \succ p_j \Leftrightarrow (r(p_i) < r(p_j)) \vee (r(p_i) = r(p_j) \wedge c(p_i) > c(p_j)) \tag{1}$$

Therefore, when the evolutionary algorithm reaches the selection step, individuals of the population, as well as the offspring generated from the crossover and mutation operators, are divided into non-dominance fronts ($F_1, F_2, \ldots$). Within each rank, individuals are ordered according to the crowding function. Eventually, the selection operator takes the individuals in descendant order until reaching the criteria established to maintain the population size for the next generation.

3. Experiments and results

It is very difficult to show and evaluate the performance of a system that is designed to generate artworks. We can show that there are signs of convergence: the fitness functions that have to be minimized are actually decreasing their values, the Pareto optimal front is getting closer to $\mathbf{0}$ in the fitness space, etc. Next, some of these indicators will be presented. But, in addition, it is very important to be able to experiment with the different fitness functions and parameters, specially in the case of techniques like evolutionary computing that has so many free parameters.

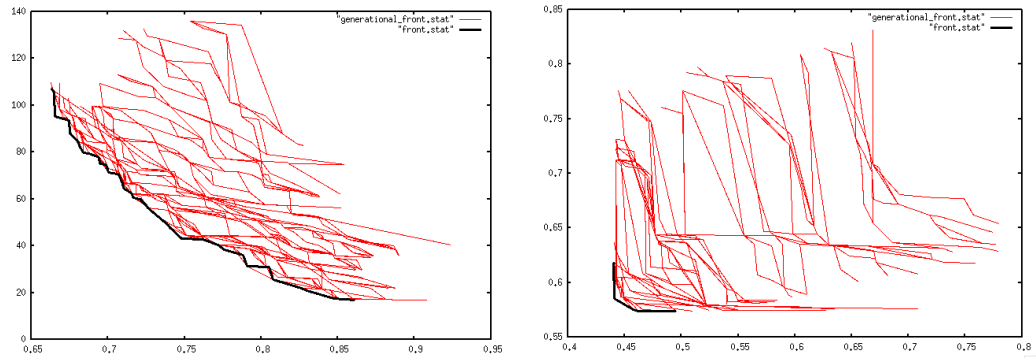

Figure 3. Evolution of the Pareto optimal frontier over the generations. It is depicted for two pairs of fitness functions. Left: $F_{\text{local,multi}}$ and F_{global,d_c}, right: F_{global,d_c} and F_{global,d_k}. It starts closer to the upper-right corner and it moves towards (0,0). The Pareto frontier at the end of the iterations is highlighted.

For that, a graphical interface prototype implemented in Java has been developed (see the Online Supplement for details). The genetic algorithms were programmed using the Java-based Evolutionary Computation Research System (ECJ) library.[1] All the mathematical calculations were programmed from scratch, except for the n-gram models, for which the BerkeleyLM library (Pauls and Klein 2011) was used.

The prototype permits one to select a training dataset in the form of single-track monophonic MIDI files contained in a folder. Some information about the datasets used for testing our system is provided in the supplemental online material. There is also the possibility of changing the configuration parameters for the genetic algorithm by a configuration file in text format. The prototype is available for download in the URL provided.[2]

Concerning the running time, it takes about 1 minute on an iMac with a 2.7 GHz Intel Core i5 processor to run a 1 000 generations evolution of a population of 100 individuals, using the 12 different fitness functions that can be utilized. If only 4 functions are used (one global, one local n-grams, one local multinomial probability, and one melodic segmentation) the running time is reduced to 24 seconds for the same population and generations.

Figure 3 displays how the Pareto optimal frontier evolves over the generations for two pairs of fitness functions. Note how it gets closer to the origin (the algorithm is minimizing the functions), with longer jumps at the beginning and then stabilizing in the last generations. It is not possible to represent the actual Pareto frontier in the 12-dimensional space, but just a selection of two dimensions, like those in Figure 3. There are more plots of the fitness evolution for pairs of functions in the online supplement.

3.1. *Single fitness function analysis*

In this section, salient features of phenotypes obtained by the application of single objective functions to the population are described. The fitness functions were trained on a dataset made up of MIDI files of popular music melodies from three subgenres: blues, pop, and celtic music. The system is designed to produce 8-bar melodies in the 4/4 meter.

The melodic phenotypes are described qualitatively in terms of their:

structure - presence of repeated melodic patterns,

[1] https://cs.gmu.edu/~eclab/projects/ecj/

[2] http://grfia.dlsi.ua.es/gen.php?id=software

segmentation - presence of phrase boundaries within the melodic sequence,
pitch - presence of implicit tonality, pitch variety, size of intervals ,
note durations - most frequent note durations, or IOI, and
convergence - has the system converged to local minima? Is phenotype variety preserved across generations?

The analysis is done manually, in search of strong evidence for the melodic features described above in the best individuals from the final population of several runs.

Structure. The n-gram based local evaluators produced melodies which often have repeated patterns. It is also the case with global statistical evaluators, although the repeated patterns were of shorter length, on average (for example, half a measure).

Segmentation. Both segmentation and melodic analysis evaluators produced phenotypes with clear phrase boundaries. Global statistical evaluators, on the other hand, seldomly generated phrase boundaries. There was no clear evidence of those boundaries in phenotypes produced by other evaluators.

Pitch. Segmentation and local evaluators based on n-grams using pitch information often produced individuals with an excess of unisons. However, when duration-based n-grams were used, there was no such problem. Local bag-of-notes models produced degenerated melodies containing often a single n-gram or no notes at all. Melodies generated by melodic analysis evaluators often had a sense of tonality, with evidence of pitch variety, but with rare presence of intervals wider than a fifth. Global statistical evaluators produced a fair variety of pitches in melodies.

Note durations. Higher variety in terms of note durations was found when using global statistical evaluators, while the others tended to produce rhythmically monotone melodies. Local n-gram models based on duration, for example, mostly produce melodies containing quarter and half note durations. Segmentation fitness produced many more of sixteenth notes compared to other evaluators, while the rest of criteria produced mainly eight or quarter notes.

Convergence. Figure 4 shows two examples of fitness convergence from different evaluator functions. Local bag-of-notes, melodic analysis, local n-grams based on pitch and duration, and local n-grams based on pitch-only evaluators exhibited a fast convergence (less than ten generations, on average) to local minima when tested in isolation. Local n-grams based on evaluators converge slowly, while global statistical functions converged at slow rates, an indication that they are less prone to be trapped too soon on local minima. When used as single fitness evaluators, all functions led the evolutionary process to converge to local minima most of the times, and produce a rather homogeneus population.

3.2. *Combined fitness function analysis*

When considering pairs of fitness functions, the outcome improved with respect to the performance when using them in isolation, but they still fell in local minima. For example, when one of the evaluators uses trigrams based on pitch, still appear flat melodies, with

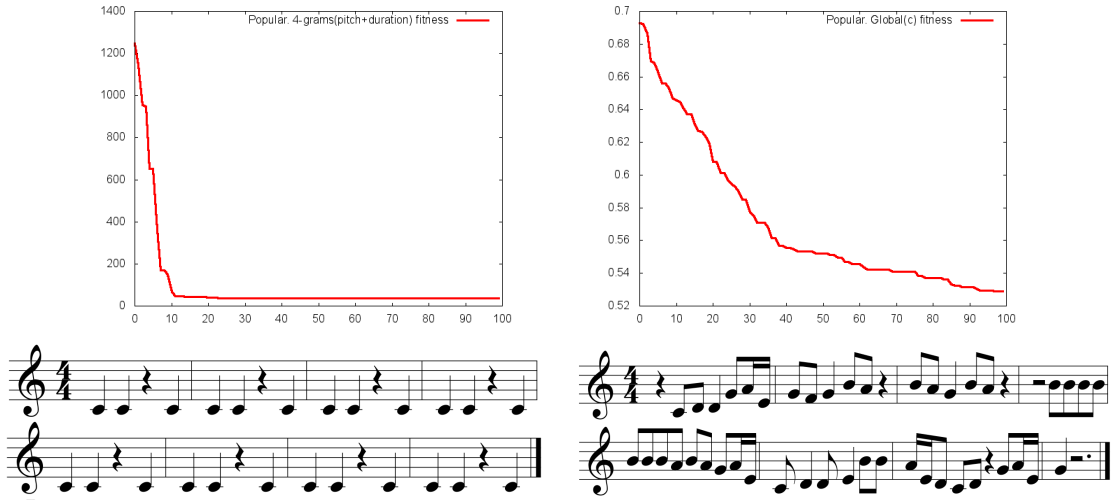

Figure 4. Fitness convergence and best individual from last generation. Left: Fast convergence of fitness for a $F_{\text{local},4-\text{gram}}$ evaluator function. Right: Smooth convergence for a F_{global,d_c}. Both evaluators were trained on a dataset of popular music. See the supplemental online material for details of the dataset.

many successive unisons.

By combining all families of evaluators, the results were closer to what one might expect. Even so, melodies clearly affected by the features inherent to certain fitness functions appear, but these are often compensated by other evaluators. For example, the result of using an evaluator that tends to produce flat melodies in terms of pitch or monotonous durations of the notes, may be balanced by other evaluators, that favor the dispersion of pitches and durations. Those evaluators that contribute to the repetition of patterns and tend to produce excessive repetition, can help in producing some structure in the melody. On the other hand, its harmful effect is mitigated by functions that favor the melodic diversity. In any case, this operation is heavily modulated by the data used as the basis for the training of evaluators.

4. Discussion and conclusions

Genetic algorithms present interesting features for music creation. Systems like those presented here can be useful for generation of interesting musical ideas that can be developed or refined by human composers.

The main issue with this kind of techniques has always been how to evaluate the fitness of maybe hundreds of candidate individuals along thousands of generations, what makes the participation of human assessments in this kind of systems unfeasible in practice, unless the size of the population is reduced to a few individuals. Machine learning and pattern recognition techniques permit one to define style-based automatic evaluation functions, considering *style* as something in common that a set of melodies may have.

In this paper, we have proposed a number of diverse evaluation functions. Some of them try to describe statistically the notes and silences in a melody, comparing it with those in a training set by distance in a vector space. Other kinds of functions intend to limit the number of possibilities by introducing statistical language models inferred from the training set with the aim of giving a very low probability to melodies containing series of notes that seldom or never appear in the reference melodies. Finally, music theory-based evaluations are proposed that, by using melody segmentation algorithms or models of

melodic analysis tags, try to favor those melodies with better behavior in terms of what is found for those algorithms in the training set.

The problem of combining so different evaluations together in such a Frankensteinian method (term already used by Todd and Werner (1998)) can be solved by multi-objective optimization techniques. They are able to rank the individuals according to their position in a fitness space relative to a geometric locus of Pareto-optimal solutions.

The research, still in an early stage, suggests that this is a flexible and powerful algorithmic composition scheme that opens a door to future studies on the influence of training data and fitness functions in the system's performance. The developed graphical interface is an important tool to help in exploring and refining the methodology.

Acknowledgements

This work was supported by the Spanish Ministerio de Educación, Cultura y Deporte through FPU fellowship (AP2012-0939) and the Spanish Ministerio de Economía y Competitividad project TIMuL (No. TIN2013–48152–C2–1–R supported by UE FEDER funds).

Supplemental online material

There is an Online Supplement where the reader can find some technical details, information about the data used, generated melodies, and additional information about the developed prototype and its performance.

Disclosure statement

The authors have no conflict of interest.

References

Alfonseca, Manuel, Manuel Cebrián, and Alfonso Ortega. 2007. "A simple genetic algorithm for music generation by means of algorithmic information theory." In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2007, 25-28 September 2007, Singapore,* 3035–3042.

Ayesh, Aladdin, and Andrew Hugill. 2005. "Genetic Approaches for Evolving Form in Musical Composition." In *IASTED International Conference on Artificial Intelligence and Applications, part of the 23rd Multi-Conference on Applied Informatics, Innsbruck, Austria, February 14-16, 2005,* 318–321.

Baluja, Shumeet, Dean Pomerleau, and Todd Jochem. 1994. "Towards automated artificial evolution for computer-generated images." *Connection Science* 6 (2-3): 325–354.

Biles, John A. 1994. "GenJam: A genetic algorithm for generating jazz solos." In *Proc. of International Computer Music Conference,* 131–137.

Cambouropoulos, Emilios. 2001. "The Local Boundary Detection Model (LBDM) and its application in the study of expressive timing." In *Proc. of the International Computer Music Conference, Havana,* .

Cope, David *Computers and musical style.* A-R Editions, Inc.

de Freitas, Alan R. R., Frederico G. Guimarães, and Raonne Barbosa Barbosa. 2012. "Ideas in Automatic Evaluation Methods For Melodies in Algorithmic Composition." In *Proceedings of the 9th Sound and Music Computing Conference, SMC 2012,* 514–520.

Deb, Kalyanmoy, Samir Agrawal, Amrit Pratap, and T Meyarivan. 2002. "A fast and elitist multiobjective genetic algorithm: NSGA-II." *IEEE Transactions on Evolutionary Computation* 6 (2): 182–197.

Doraisamy, Shyamala, and Stefan Rüger. 2003. "Robust polyphonic music retrieval with n-grams." *Journal of Intelligent Information Systems* 21 (1): 53–70.

Dostál, Martin. 2013. "Evolutionary Music Composition." In *Handbook of Optimization - From Classical to Modern Approach,* 935–964. Springer.

Fu, Tao-yang, Tsu yu Wu, Chin te Chen, Kai chu Wu, and Ying ping Chen. 2006. "Evolutionary interactive music composition." In *Proc. of 8th Annual Conf. on Genetic and evolutionary computation, GECCO'06,* 1863–1864.

Göksu, Hüseyin, Paul Pigg, and Vikas Dixit. 2005. "Music Composition Using Genetic Algorithms (GA) and Multilayer Perceptrons (MLP)." In *Advances in Natural Computation, First International Conference, ICNC 2005, Changsha, China, August 27-29, 2005, Proceedings, Part III,* 1242–1250.

Goldberg, David E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning.* 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Herremans, D., K. Sörensen, and D. Conklin. 2014. "Sampling the extrema from statistical models of music with variable neighbourhood search." In *International Computer Music Conference, ICMC/SMC,* Athens, Greece, 1096–1103.

Hidden. 0000. "A cooperative approach to style-oriented music composition." In *Proc. of the Int. Workshop on Artificial Intelligence and Music,* .

Hiller, Lejaren, and Leonard M. Isaacson. 1959. *Experimental Music: Composition with an Electronic Computer.* McGraw-Hill.

Hofmann, David M. 2015. "A Genetic Programming Approach to Generating Musical Compositions." In *Evolutionary and Biologically Inspired Music, Sound, Art and Design – 4th International Conference, EvoMUSART 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings,* 89–100.

Holland, John H. 1975. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press.

Horn, Jeffrey, Jeffrey Horn, Nicholas Nafpliotis, Nicholas Nafpliotis, David E. Goldberg, and David E. Goldberg. 1994. "A Niched Pareto Genetic Algorithm for Multi-objective Optimization." In *Proceedings of the 1st IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence,* Orlando, Florida, 82–87. IEEE.

Kneser, Reinhard, and Hermann Ney. 1995. "Improved backing-off for M-gram language modeling." In *1995 International Conference on Acoustics, Speech, and Signal Processing, ICASSP '95, Detroit, Michigan, USA,* 181–184.

Koga, Shimpei, and Makoto Fukumoto. 2014. "A Creation of Music-Like Melody by Interactive Genetic Algorithm with User's Intervention." In *HCI International 2014 - Posters' Extended Abstracts - International Conference, HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014. Proceedings, Part I,* 523–527.

Komatsu, Kyoko, Tomomi Yamanaka, Masami Takata, and Kazuki Joe. 2010. "A Music Composition Model with Genetic Programming." In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA 2010, Las Vegas, Nevada, USA, July 12-15, 2010, 2 Volumes,* 686–692.

Koza, John R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA, USA: MIT Press.

Laine, Pauli, and Mika Kuuskankare. 1994. "Genetic Algorithms in Musical Style Oriented Generation." In *International Conference on Evolutionary Computation,* 858–862.

Liu, Chien-Hung, and Chuan-Kang Ting. 2015. "Music pattern mining for chromosome representation in evolutionary composition." In *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015,* 2145–2152.

Lo, Man Yat, and Simon M. Lucas. 2007. "N-gram fitness function with a constraint in a musical evolutionary system." In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2007, 25-28 September 2007, Singapore,* 4246–4251.

Machado, Penousal, Juan Romero, Bill Manaris, Antonino Santos, and Amílcar Cardoso. 2003. "Power to the Critics – A Framework for the Development of Artificial Art Critics." In *Proc. of 3rd Workshop on Creative Systems, 18th IJCAI,* Acapulco, Mexico, August, 55–64.

Miranda, Eduardo Reck, and John Al Biles. 2007. *Evolutionary Computer Music.* Springer.

Moroni, Artemis, Jonatas Manzolli, Fernando Von Zuben, and Ricardo Gudwin. 1994. "Vox Populi: An interactive evolutionary system for algorithmic music composition." *Leonardo Music Journal* 10: 49–54.

Nierhaus, Gerhard. 2008. *Algorithmic Composition: Paradigms of Automated Music Generation.* 1st ed. Springer Publishing Company, Incorporated.

Oliwa, Tomasz Michal. 2008. "Genetic algorithms and the abc music notation language for rock music composition." In *Genetic and Evolutionary Computation Conference, GECCO 2008, Proceedings,*

Atlanta, GA, USA, July 12-16, 2008, 1603–1610.

Özcan, Ender, and Türker Erçal. 2007. "A Genetic Algorithm for Generating Improvised Music." In *Artificial Evolution, 8th International Conference, Evolution Artificielle, EA 2007, Tours, France, October 29-31, 2007, Revised Selected Papers,* 266–277.

Pauls, Adam, and Dan Klein. 2011. "Faster and smaller N-gram language models." In *HLT '11: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies,* Jun. Association for Computational Linguistics.

Phon-Amnuaisuk, Somnuk, Edwin Hui Hean Law, and Chin Kuan Ho. 2007. "Evolving Music Generation with SOM-Fitness Genetic Programming." In *Applications of Evolutinary Computing, EvoWorkshops 2007: EvoCoMnet, EvoFIN, EvoIASP,EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog, Valencia, Spain, April11-13, 2007, Proceedings.*, 557–566.

Ponce de León, P. J., and J. M. Iñesta. 2007. "A Pattern Recognition Approach for Music Style Identification Using Shallow Statistical Descriptors." *IEEE Transactions on Systems Man and Cybernetics C* 37 (2): 248–257.

Purshouse, Robin C., Peter J. Fleming, Carlos M. Fonseca, Salvatore Greco, and Jane Shaw, eds. 2013. *Evolutionary Multi-Criterion Optimization - 7th International Conference, EMO 2013, Sheffield, UK, March 19-22, 2013. Proceedings*, Vol. 7811 of *Lecture Notes in Computer Science*. Springer.

Putnam, Jeffrey B. 1996. "A Grammar-Based Genetic Programming Technique Applied to Music Generation." In *Evolutionary Programming V: Proc. of the 5th Annual Conf. on Evolutionary Programming,* Feb. 29-Mar. 3, 277–286. MIT Press.

Rizo, David. 2010. "Symbolic music comparison with tree data structures." Ph.D. thesis, Universidad de Alicante.

Rizo, David, Plácido R. Illescas, and Jose M. Iñesta. 2015. *Interactive melodic analysis*, chap. 7, 191–219. Springer.

Romero, Juan, and Penousal Machado, eds. 2007. *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*. Natural Computing Series. Springer Berlin Heidelberg.

Schaffer, David J. 1985. "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms." In *Proceedings of the First International Conference on Genetic Algorithms and Their Applications,* Pittsburgh, PA, 93–100. Carnegie-Mellon University.

Selfridge-Field, Eleanor. 1997. *Beyond MIDI: The Handbook of Musical Codes*. MIT Press.

Sheikholharam, Peyman, and Mohammad Teshnehlab. 2008. "Music Composition Using Combination of Genetic Algorithms and Recurrent Neural Networks." In *8th International Conference on Hybrid Intelligent Systems (HIS 2008), September 10-12, 2008, Barcelona, Spain,* 350–355.

Spector, Lee, and Adam Alpern. 1995. "Induction and Recapitulation of Deep Musical Structures." In *Proc. of the IJCAI-95 Workshop on Music and AI,* 41–48.

Srinivas, Nidamarthi, and Kalyanmoy Deb. 1994. "Multi-objective Optimization Using Nondominated Sorting in Genetic Algorithms." *Evolutionary Computation* 2: 221–248.

Todd, Peter, and Eduardo R. Miranda. 2003. "Putting some (artificial) life into models of musical creativity." In *Musical creativity: Current research in theory and practise,* edited by Irene Deliege and Geraint Wiggins. Psychology Press.

Todd, Peter M., and Gregory M. Werner. 1998. *Frankensteinian methods for evolutionary music composition*. MIT press/Bradford books.

Tokui, Nao, and Hitoshi Iba. 2000. "Music composition with interactive evolutionary computation." In *Proc. of 3rd Int. Conf. on Generative Art,* 215–226.

Vatolkin, Igor. 2013. "Measuring the Performance of Evolutionary Multi-Objective Feature Selection for Prediction of Musical Genres and Styles." In *Informatik 2013, 43. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Informatik angepasst an Mensch, Organisation und Umwelt, 16.-20. September 2013, Koblenz,* 3012–3025.

Waschka, Rodney. 2007. "Avoiding the fitness bottleneck using genetic algorithms to compose orchestral music." In *Proc. of the Int. Workshop on Artificial Intelligence and Music, MUSIC-AI,* Beijing, China, 25–36.

Weale, Timothy, and Jennifer Seitzer. 2003. "EVOC: A Music Generating System using Genetic Algorithms." In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003,* 1383–1384.

Wiggins, Geraint A., George Papadopoulos, Somnuk Phon-Amnuaisuk, and Andrew Tuson. 1998. "Evolutionary methods for musical composition." In *Proc. of CASYS'98 Workshop on Anticipation, Music and Cognition, .*

Willingham, Timothy J. 2013. "The harmonic implications of the non-harmonic tones in the four-part chorales of Johann Sebastian Bach." Ph.D. thesis, Liberty University, Liberty University.

Xenakis, I. 1992. *Formalized Music: Thought and Mathematics in Composition.* Harmonologia series. Pendragon Press.

Zhu, Hua, Shangfei Wang, and Zhen Wang. 2008. "Emotional Music Generation Using Interactive Genetic Algorithm." In *International Conference on Computer Science and Software Engineering, CSSE 2008, Volume 1: Artificial Intelligence, December 12-14, 2008, Wuhan, China,* 345–348.

Zitzler, Eckart, Marco Laumanns, and Lothar Thiele. 2001. "SPEA2: Improving the strength pareto evolutionary algorithm for multi-objective optimization." In *Proceedings of Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems,* edited by Kyriakos C. Giannakoglou, 95–100. Athens, Greece: International Center for Numerical Methods in Engineering.