EVOLUTIONARY PERSPECTIVES IN COMPUTER MUSIC

CECÍLIA REIS
Institute of Engineering of Porto
Electrical Engineering Department, Porto, Portugal
cmr@isep.ipp.pt

VIRIATO M. MARQUES
Institute of Engineering of Coimbra
Computer Science Department, Coimbra, Portugal
viriato@isec.pt

J. TENREIRO MACHADO
Institute of Engineering of Porto
Electrical Engineering Department, Porto, Portugal
jtm@isep.ipp.pt

This paper presents a brief overview of music evolution - western and non-western music - from its genesis to serialism and the Darmstadt school. Some mathematical aspects of music are then presented and confronted with music as a form of art. Some questions follow: are these two (very) distinct aspects compatible? Can computers be of real help in automatic composition? Evolutionaty Algorithms (EAs) - Genetic Algorithms (GAs), Genetic Programming (GP), Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) - are then introduced and some results of GAs and GPs application to music generation are analysed. Variable fitness functions and PSO application seems a promising way to explore. However, what output should be envisaged? Should we expect that computer music sounds as human music, or should we look for a totally different way to explore and listen? How far can go computer creativity and in what direction?

1. A Brief History of the Western Music

What is music? Citing Edgar Varèse [19], music is *organized sound*. A well known definition from the late 19th century states that music is *the artful or pleasing organization of sound and silence*. However, the way sounds are organized has dramatically changed over the years and among different cultures. In fact, music is found in every culture and even in the most primitive and isolated societies. Perhaps invented in Africa at least 50.000 years ago, music spread all over the world giving rise to completely different forms of expression ranging

from primitive tribal songs to sophisticated concerts, from Gregorian chant to serialism, also including Indian ragas, (Chinese) pentatonic scale based music, opera, jazz, The Beatles, etc.

The history of Western Music is typically divided into the following periods [21, 29, 31, 32, 33]: Medieval (500-1400); Renaissance (1400-1600); Baroque (1600-1750); Classical (1750-1800); Romantic (1800-1900); Modern and Contemporary (1900...).

The Medieval period includes the Gregorian chant, the invention of polyphony (*organum*, Notre Dame School of polyphony - Léonin, Pérotin), the troubadours, Ars Nova and styles like *ballade*, *virelai*, *rondo*, etc. The Renaissance period is characterized by forms like the *motet* (sacred) and the *madrigal* (secular); the *counterpoint*, also born in this period, essentially consists of sets of rules (organized in five *species*) aiming to conjugate different "independent" voices in the same musical piece. *Opera* is born in the Baroque period (Dafne by Jacopo Peri) where Bach, Handel and Vivaldi belong too. The Classical period is marked by the birth of the symphony, Haydn and Mozart. In the Romantic period music becomes more expressive and emotional, frequently showing some form of national appeal and/or a folkloric basis; some outstanding composers of this period are Beethoven, Schubert, Chopin, Liszt, Verdi and Puccini; the music chromatic complexity rises reaching a maximum (perhaps) with Wagner. After this period it seems that there's nothing more to explore in this direction. So, a search for completely different styles begins. The Modern period starts with Claude Débussy; in the XX century *atonalism* makes its appearance too; about 1920 Schoenberg creates *dodecaphony*, a composition method strongly based on rules that act as a means of ensuring that all 12 notes of the chromatic scale have equal importance. Extended to other musical aspects like rhythm, dodecaphony evolved to *serialism* divulgated by the Darmstadt School. Iannis Xenakis uses computers to write music based on statistical and probabilistic methods. John Cage (1912-1992), an experimentalist compositor, creates compositions such as 4'33" where, during this lapse of time, the orchestra doesn't play a single note (http://www.youtube. com/watch?v=hUJagb7hL0E).

2. Music, Mathematics and Physics

Apart some experimentalist work, this impressive repertoire of melodies, rhythms and styles is based on the principles of frequency and time integer division. In fact, there are very interesting relationships between music, mathematics and physics in what respects to the three fundamental

characteristics of a sound: pitch, time duration and timbre.

2.1. *Pitch*

The term *interval* designates the pitch difference between any two notes. A scale is an ordered sequence of musical intervals [23, 27]. Western European music is primarily based on the *diatonic scale* composed by seven notes: C, D, E, F, G, A, B (C...) that repeat along each musical instrument range (Figure 1).

An *octave interval* is defined between any note and the next one with the same name. In a diatonic scale the octave is divided into five intervals of one *tone* and two of *half-tone*. The half-tones occur between E-F and B-C.

To position a note in an octave an index is used. So, A1 and A2 define an octave interval as well as C3 and C4, for instance.

The pitch of each note is defined by its (fundamental) frequency. For instance, A4 corresponds to 440 Hz. An ascending octave is defined by frequency duplication. So $f_2 = 2f_1$ where f_2=*frequency of the* X_n *note* and f_1=*frequency of* X_{n-1} *note*. Computing $3f_{C3}$ one gets f_{G4}. So:

$$f_{G3} = \frac{3}{2} f_{C3} \tag{1}$$

This 3/2 ratio defines an interval of *fifth*. A sequence of six fifth intervals generates, although in distinct octaves, the seven pitches of the diatonic scale: C-G-D-A-E-B-F. These type of relationships have already been devised by Pythagoras [*].

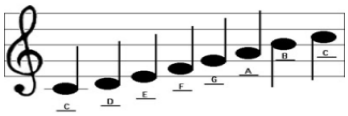

Figure 1. Example of a diatonic scale

Pitch generation according to the above described, gives rise to the *Pythagoric temperament* - a way of tuning musical instruments - where a *tone* corresponds to a 9/8 frequency ratio.

The term *chromatic* appeared in the 16th century. A *chromatic scale* consists of an ascending (or descending) sequence of 12 notes all separated by a half-

[*] Other interesting relationships, that have to do with wavelength, are string and tube laws that relate their length with the sound frequency generated.

tone. Briefly, it contains all the ♯ (sharps) and ♭ (flats). As the first and the last notes are separated by an octave, the frequency relationship for any half-tone in an ascending chromatic scale is defined by $f_2 = \sqrt[12]{2} f_1$. For keyboard instruments and due to the physical limitations of their construction any X♯ is enharmonic of the following Y ♭ note (for instance C ♯ and D ♭ are represented by the same piano key). This tuning system, consecrated by J.S.Bach in his masterpiece *Das Wohltemperierte Klavier*, is called *Twelve-tone Equal Temperament*. The $f_2 = \sqrt[12]{2} f_1$ relationship differs from the 9/16 of the Phytagoric semitone, but this difference is not perceptible by the human year.

On this basis any piece of music may start in any note: it will sound the same as long as the intervals between the notes that compose it stay the same, as for the human hearing what makes sense (*i.e.*, what defines a melody) is exactly this sequence of intervals (*i.e.*, not the absolute pitch of each note). This gives rise to the concept of key signature: in order to keep the melodic intervals the same, groups of fixed ♯ or ♭ must always be applied to some notes. These are written in the start of the staff. These groups, raising from zero to seven ♯ or ♭ define the *key signature*. A *transposition* consists in changing the key signature of a music piece as well as the notes that compose it, this way keeping the original intervals between notes (Figure 2).

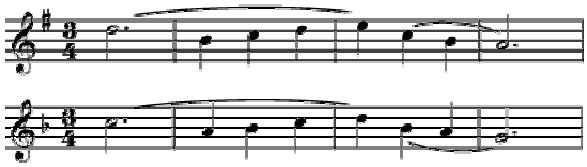

Figure 2. *Transposition*: original melody using the key signature of G and the same melody, transposed, using the key signature of F.

2.2. *Duration*

The duration and absence of sounds is defined by *notes* and *rests* (Figure 3). In its basic form, the duration of each note is just 1/2 of its precedent [23, 27].

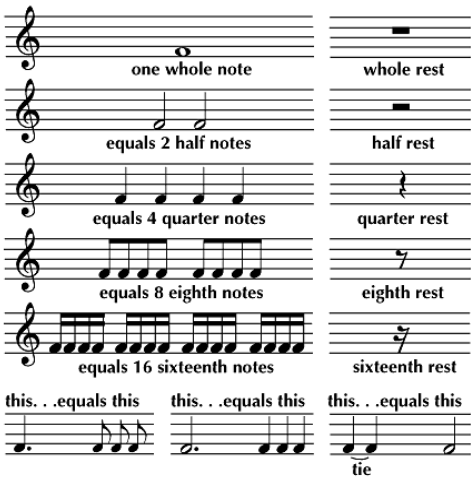

Figure 3. Some notes, rests, tuplets and their time duration

The duration of notes can also be altered by *dots* - that add a duration equal to 1/2 of the note they are applied to - *double dots*, *ties* - that extend the duration of a note by adding to it the duration of the following one - and *tuplets* - groups of notes (3, 5, 6 or any other number) that have, as a whole, a certain duration (Figure 3). In this last case one may have notes with durations such as 1/3, 1/5, 2/3 or any other, but all expressed by integer number ratios.

2.3. *Timbre*

Timbre allows the distinction between different musical instruments and is the basis of the *instrumentation* phase in music composition [23, 27].

The timbre of different musical instruments is determined by the harmonic contents of the sound they produce. There are no sinusoidal sounds in music, but rather complex waveforms resulting from distinct harmonic contents in amplitude and phase. One can approach classical instruments timbre by adding the relevant harmonic contents to the base frequency of the pitch to produce. It is a matter of Fourier analysis (Figure 4).

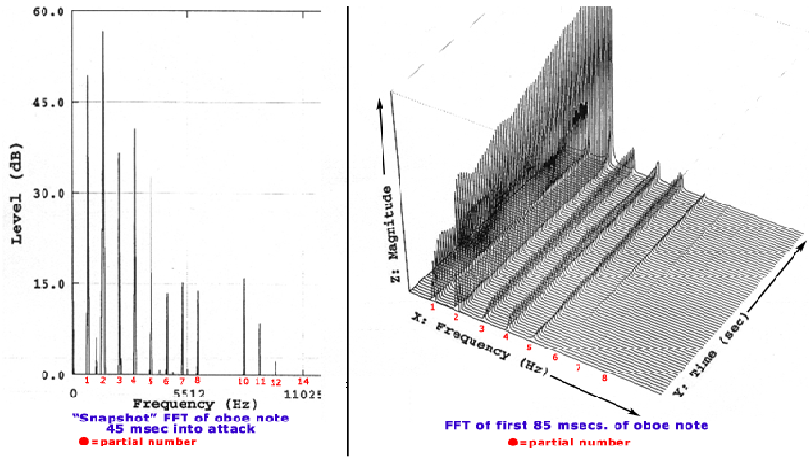

Figure 4. Fourier Transform of an oboe note (www.indiana.edu/~emusic/acoustics/wave_shape.htm)

But a musical sound is also characterized by its *envelope* that comprises 4 basic phases named Attack, Decay, Sustain and Release (ADSR). Together they determine the external form of the waveform. A piano, for instance, has always a crisp attack phase followed by a quick Decay, a Sustain (that can be expanded by the sustain pedal) and a final extinction (release).

3. Foundations *versus* Styles

While there are no sounds that can be described as inherently unmusical, musicians in each culture have tended to restrict the range of sounds they will admit [22].These restrictions are basically imposed by the mode or scale in which a music piece is based, as the notes to be used tend to be, most of the time, those that compose it [23, 27, 28, 29, 30, 31].

In fact, Greek cultured already treatises on music referred to modes (or scales). Modes are "sets" of ordered notes, a kind of scale beginning in different notes.

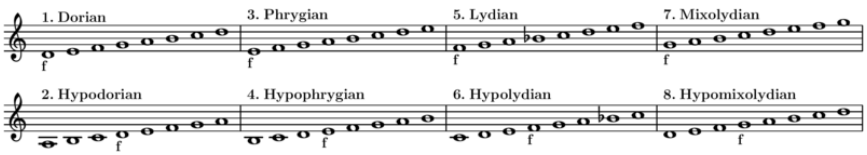

Figure 5. The Gregorian modes (Wikipedia http://en.wikipedia.org/wiki/Musical _mode)

In the 9th century the Western church modes, called Gregorian modes, were born. These modes are called Dorian, Hypodorian, Phrygian, Hypophrygian, Lydian, Hypolydian, Mixolydian and Hypomixolydian (Figure 5).

Western music has been *tonal* from Baroque to the advent of dodecafonism in the 20th century. The *tonal* system defines some hierarchical relationships between the notes of a scale. The more important is the first degree, called *tonic*; the following ones are the fifth (*dominant*) and the fourth (*sub-dominant)* degrees. A chord is a combination of three or more notes played simultaneously. There are many chord types but the basic ones, for Western music, are obtained by building a *third interval* (*major* or *minor*) and a *fifth interval* on a fundamental pitch. When correctly combined with a melody, chords fulfil the musical ambience and give the listener the complete music sense and feeling. In this context, tonic and dominant degrees are so important that very simple melodies (like "Malhão", a Portuguese folk song) can be *harmonized* by constructing chords only on these two degrees.

Moreover, for centuries the Western European music is based on *major* and *minor* scales. The *C major* scale is shown in figure 1. The *A minor* scale - its *relative minor* - starts in A and has a G♯ (for the more common *harmonic minor* scale). A major scale and its relative minor are separated by a *third minor* interval (1.5 tones). A piece based on a major scale tends to transmit happiness; a piece based on a minor scale evocates sadness or nostalgia. This is one of the things that allow us to recognize Western music. These two flavours can be mixed and interchanged along a melody in very pleasant forms, particularly if the pitches belong to a *major/minor* scale and to its relative *minor/major*, respectively.

Tonality and the *major/minor* system are in fact the two great supports of the traditional European musical style, traditional meaning, here, primarily classic and romantic erudite music as well as popular songs of every style.

Impressionism, with Débussy and Ravel, tries to evocate landscapes, sensations or images, not focusing on emotions, although they tend to persist...

By the contrary, dodecafonism states that there should not be any hierarchy between any of the 12 notes of an octave. A set of composition rules tries to ensure this principle. Also being a chromatic scale based music, major and minor flavours are lost. That is why dodecafonism seems so distant from the traditional European style. Emotions are, in great part, absent. Dodecafonism is reasoning, not feeling.

Music of Eastern Asia makes a wide use of *pentatonic* scales (only composed by five notes). A pentatonic based melody will probably be

immediately taken as "Chinese" by a European.

Indian music neither cares about the harmony, nor uses a major/minor system. It features a single instrument or voice accompanied by drone and percussion. Its originality resides on melodies and rhythms, a complex system based on *Ragas*. The tuning system also differs from the Western one. Indian music uses the concept of *sruti*: the interval between C and D would be 4 *srutis*, for instance.

Another interesting example comes from the *blues*: along with their particular beat and rhythm, the *blues* make use of a typical pitch that lies between 1.5 *tone* and 2 *tone* above the tonic. As a matter of fact keyboard instruments don't even have such a note, replacing it by pressing two keys simultaneously. The *blues* flavour is much characterized by this particular sound.

4. Music and Art

There are profound relationships between music, mathematics and physics, as well as solid theoretical foundations, in part responsible for the flavour of each style. But music is, perhaps above all, a form of art. In fact, [24] defines music as *an art concerned with combining vocal or instrumental sounds for beauty of form or emotional expression, usually according to cultural standards of rhythm, melody, and, in most Western music, harmony.*

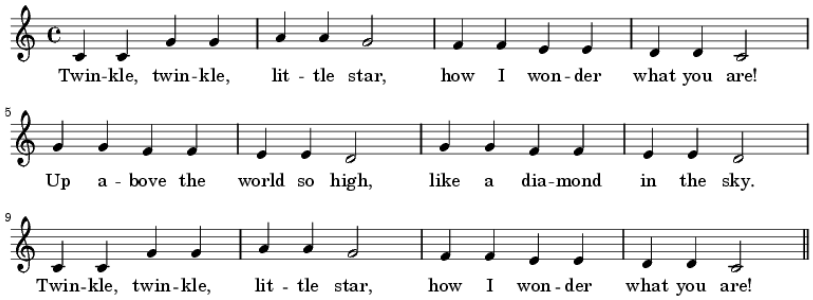

Figure 6 . 'Twinkle, twinkle little star' popular English song written on the French melody 'Ah vous dirai je maman'

All of these aspects are eminently subjective and evaluable by aesthetic considerations. A complete discussion of this definition cannot be presented here due to space limitations. However, we will focus some of its aspects. *Beauty of*

form[†] refers to the beauty of melody and/or harmony as a whole. Here we will focus just a particular aspect of Western music, perhaps more noticeable from baroque to impressionism: a melody is not a simple sequence of beautiful musical sounds: in fact, it can be divided into phrases that form a kind of dialogue, a succession of questions and answers. Take, for instance, the example of *Ah vous dirai je maman* (figure 6).

This melody can be clearly divided into six phrases, each one composed of two *bars* (or *measures*). Moreover, the first phrase needs a second one: if it wasn't there the melody would seem unfinished, incomplete. By the contrary, the last phrase (last two bars) give a sensation of end, terminus, a final declaration or conclusion. Nothing else needs to be there. So, in these sense, *beauty of form* has to do with human understanding of melody. Music is not exactly a language, but it is very near it.

We have already talked a little bit about *emotional expression* in the last section. As mentioned, melodies based on *major* and *minor* modes can easily transmit *happiness* and *sadness* emotions. However, anger - evident in some Beethoven compositions, for instance - peace and devotion - remember Schubert's Ave Maria, for instance - can be evocated too. Moreover, why do we like music based on diatonic scales? Why do they sound "well" having half-tones in some specific positions? Is it just a matter of habitude, culture, or is there any profounder reason to that? An integer-tone scale is not appealing. It does not "make sense" and even pentatonic scales maintain the interval irregularity characteristic of the diatonic. It seems, though, there is some profounder reason for this, something that may have to do with cognitive science.

Briefly, traditional Western music deals with profound emotions, human consciousness and cognitive processes.

Some *cultural standards of rhythm [and] melody* have already been presented in the previous sections. Every art is a product of the culture it belongs to and innovation generally brings with it some form of contestation. Fortunately this does not mean that new exploration directions do not win. That is the key of progress in every field, art included.

Finally, let us consider *harmony*, an extremely subjective concept even when considering Western music only. Consider just two examples: *third* intervals have been considered dissonant until Renaissance but are, nowadays,

[†] Composition form in itself - like *sonata* or *symphony*, for instance - would lead us to the discussion of the sequence of themes presentation and their development, movements, key signatures, etc. Here we won't focus this subject.

perhaps the simplest form of two voices singing. The *third* harmony is almost intuitive, used even in folk music. The tonal music of late 19th century - Wagner, for instance - is full of very expressive chromatic passages that difficulty can be classified as harmonic: chromatic passages of this kind transmit a characteristic tension sensation that is, perhaps, the heart of this style.

5. Music and Algorithms

As music is a product of the culture it belongs to, it seems perfectly natural that 20th and 21th centuries produce computer based music.

Iannis Xenakis, Lejaren Hiller and John Cage are examples of composers that used computers to produce music, or as a tool for an *a posteriori* composition process. The work of Xenakis and Hiller was based on stochastic methods. Other approaches include neural networks, rule-based systems, knowledge-based systems, evolutionary computation and hybrid systems that combine two or more of the preceding ones. But, from the musical point of view, what tasks are these systems supposed to perform? We classify these tasks as follows:

1. Melody generation
2. Harmony generation
3. Instrumentation of a given musical piece
4. Variation/improvisation on a given musical theme

The three former items implement a (possible) complete musical composition process, from the melody draft to the final music piece. The fourth one is of another kind, although it has some similarities with melody generation. Examples are the twelve variations of Mozart on *Ah vous dirai je maman* and *jazz* improvisation.

From the algorithmic point of view anyone of these four tasks can be considered a search task: we are looking for solutions that are aesthetical appealing according to some restrictions that represent beauty and/or composition rules satisfaction. As the musical search space is virtually unlimited, these restrictions are welcome as they allow search space delimitation.

For *melody generation* one can shrink the search space by basing the melody on a given musical scale so restricting the pitches to use and/or giving them some kind of hierarchy. If we are looking for a serial theme, then the base scale will be the chromatic one and the serialism composition rules will act as

guide. *Harmony* generation is primarily a deterministic task and so harmonization rules and counterpoint may be applied. *Instrumentation* has been referred in section 2.3 and, once again, it depends on music kind and aesthetical considerations. Musical instruments limitations and previous knowledge about their successful combinations may act as restrictions.

From the author's point of view, the more appealing subject is the *melody generation* process as it seems the more challenging and/or susceptible of innovation. The more appealing algorithmic approach is *evolutionary computation* as it offers creativity potential along with criticizing possibilities by means of their *fitness function* (see section 6).

In fact, a melody can be played or sung alone, can evocate emotions and is susceptible of being appreciated by its own, harmony and instrumentation acting as a complement. Creating a beautiful melody according to the tonal Western music style is, no doubt, a very challenging task. However, melody generation opens new search directions too: why should computer music focus just on the known musical patterns? Proceeding this way wouldn't we just be doing reverse engineering? It seems perfectly possible to create and explore new musical directions somewhat far away from human music as we know it today. This means "strange" musical pieces according to today patterns, as they can be built, among other things, of uncommon pitches and rhythms, eventually giving rise to new harmonies made of synthesized timbres produced by unknown instruments.

By its characteristics, evolutionary computation has been widely used for creative tasks, from digital circuit synthesis [34] to robotics [35], from painting to design and from poetry [36] to music [12]. In art applications it becomes evident the main role played by their fitness function that should, according to some patterns, correctly judge the quality of the solutions produced, acting as painter, poet or musician. In this context the fitness function is, in fact, the big deal! Solutions based on heuristics, rules, neural networks and interactive - i.e. where human users intervene - have already been tried.

In the next sections we present an overview of Evolutionary Computation techniques and some applications to computer music.

6. Evolutionary Computation (EC)

Evolutionary Computation (EC) is a subfield of artificial intelligence, more particularly computational intelligence that involves combinatorial optimization problems. EC uses iterative progress, such as growth or development in a population. This population is then selected in a guided random search using parallel processing to achieve the desired end. Such processes are often inspired

by biological mechanisms of evolution [37, 38]. Therefore, EC is the general term for several computational techniques which are based, to some degree, on the evolution of biological life in the natural world. Evolutionary techniques mostly involve metaheuristic optimization algorithms such as:

- *Evolutionary Algorithms*: genetic algorithms, evolutionary programming, evolution strategy, genetic programming and learning classifier systems.
- *Swarm Intelligence*: ant colony optimization and particle swarm optimization.

Evolutionary Algorithms (EAs) are search methods that take their inspiration from natural selection and survival of the fittest in the biological world. EAs differ from more traditional optimization techniques in that they involve a search from a "population" of solutions, not from a single point. Each iteration involves a competitive selection that weeds out poor solutions. The solutions with high "fitness" are "recombined" with other solutions by swapping parts of a solution with another. Solutions are also "mutated" by making a small change to a single element of the solution. Recombination and mutation are used to generate new solutions that are biased towards regions of the space for which good solutions have already been seen. Figure 7 presents a pseudo-code for an evolutionary algorithm.

1. Initialize the population
2. Calculate the fitness of each individual in the population
3. Reproduce selected individuals to form a new population
4. Perform evolutionary operations such as crossover and mutation on the population
5. Loop to step 2 until some condition is met

Figure 7. Evolutionary computation algorithm.

Swarm Intelligence (SI) is artificial intelligence based on the collective behavior of decentralized, self-organized systems. The expression was introduced by Gerardo Beni and Jing Wang in 1989, in the context of cellular robotic systems [11]. SI systems are typically made up of a population of simple agents interacting locally with one another and with their environment. The

agents follow very simple rules, and although there is no centralized control structure dictating how individual agents should behave, local interactions between such agents lead to the emergence of complex global behavior. Natural examples of SI include ant colonies, bird flocking, animal herding, bacterial growth, and fish schooling.

6.1. *Genetic Algorithms (GAs)*

Genetic Algorithms (GAs) are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetic. The basic concept of GAs is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Charles Darwin of survival of the fittest [41, 44].

In 1859 the British naturalist, Charles Darwin, published a book that would change the way humans view the world. In this book entitled 'The Origin of Species', Darwin proposed that humans, and in fact all creatures, were not put on this planet by God and made unchanging, but rather that they evolved from other creatures. Over time, creatures change to adapt to their environment to survive and thrive [40].

GAs were invented by John Holland in the sixties and were developed by Holland, his students and colleagues at the University of Michigan. Holland's 1975 book Adaptation in Natural and Artificial Systems presented the genetic algorithm as an abstraction of biological evolution and gave a theoretical framework for adaptation under the GA [39]. Holland's GA is a method for moving from one population of *chromosomes* to a new population by using a kind of *natural selection* together with the genetics−inspired operators of crossover, mutation, and inversion. Essentially, they are a method of "breeding" computer programs and solutions to optimization or search problems by means of simulated evolution. Processes loosely based on natural selection, crossover, and mutation are repeatedly applied to a population of binary strings which represent potential solutions. Over time, the number of above-average individuals increases, and highly-fit building blocks are combined from several fit individuals to find good solutions to the problem at hand.

6.1.1. *Chromosome representation and genetic operators*

Each chromosome consists of genes [43]. The selection operator chooses those chromosomes in the population that will be allowed to reproduce. Crossover exchanges subparts of two chromosomes, roughly mimicking biological recombination between two organisms. Mutation randomly changes the values

of some locations in the chromosome. Inversion reverses the order of a contiguous section of the chromosome.

6.2. *Genetic Programming (GPs)*

Genetic Programming (GP) is an EA based methodology inspired by biological evolution to find computer programs that perform a user-defined task. It is a specialization of GAs where each individual is a computer program. Therefore, it is a machine learning technique used to optimize a population of computer programs according to a fitness landscape determined by a program's ability to perform a given computational task.

GP has begun with the evolutionary algorithms first utilized by Nils Aall Barricelli in 1954 as applied to evolutionary simulations, but evolutionary algorithms became widely recognized as optimization methods as a result of the work of Ingo Rechenberg in the sixties and early seventies - his group was able to solve complex engineering problems through evolution strategies (1971 PhD thesis and resulting 1973 book). Also, highly influential was the work of John Holland in the early 1970s, and particularly his 1975 book.

The first results on the GP methodology were reported by Stephen F. Smith (1980) [1] and Nichael L. Cramer (1985) [2]. In 1981 Forsyth reported the evolution of small programs in forensic science for the UK police. John R. Koza is a main proponent of GP and has pioneered the application of genetic programming in various complex optimization and search problems [3, 42].

GP is very computationally intensive and so in the nineties it was mainly used to solve relatively simple problems. More recently, thanks to improvements in GP technology and to the exponential growth in CPU power, GP produced many novel and outstanding results in areas such as quantum computing, electronic design, game playing, sorting, searching and many more.

6.2.1. *Chromosome representation and genetic operators*

GP evolves computer programs, traditionally represented in memory as tree structures as shown in figure 8. Trees can be easily evaluated in a recursive manner. Every tree node has an operator function and every terminal node has an operand, making mathematical expressions easy to evolve and evaluate.

The main operators used in GP are crossover and mutation.

Crossover is applied on an individual by simply switching one of its nodes with another node from another individual in the population. With a tree-based representation, replacing a node means replacing the whole branch. This adds greater effectiveness to the crossover operator. The expressions resulting from

crossover are very much different from their initial parents.

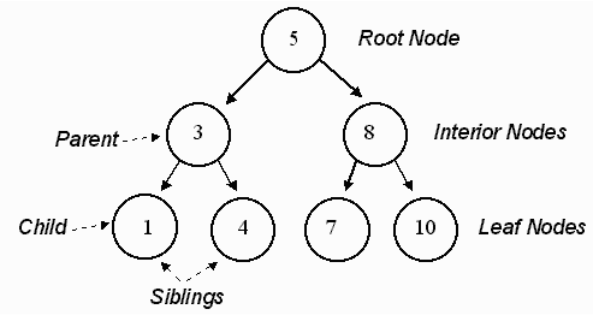

Figure 8. Programming tree structure.

Mutation affects an individual in the population. It can replace a whole node in the selected individual, or it can replace just the node's information. To maintain integrity, operations must be fail-safe or the type of information the node holds must be taken into account. For example, mutation must be aware of binary operation nodes, or the operator must be able to handle missing values.

6.3. *Particle Swarm Optimization (PSO)*

In the literature about PSO the term *Swarm Intelligence* appears rather often and, therefore, we begin by explaining why this is so.

Non-computer scientists (ornithologists, biologists and psychologists) did early research, which led into the theory of particle swarms. In these areas, the term *swarm intelligence* is well known and characterizes the case when a large number of individuals are able to accomplish complex tasks. Motivated by these facts, some basic simulations of swarms were abstracted into the mathematical field. The usage of swarms for solving simple tasks in nature became an intriguing idea in algorithmic and function optimization [45].

Eberhart and Kennedy were the first to introduce the PSO algorithm (figure 9) [7], which is an optimization method inspired in the collective intelligence of swarms of biological populations and was discovered through simplified social model simulation of bird flocking, fishing schooling and swarm theory.

1. Initialize population in hyperspace
2. Evaluate fitness of individual particles
3. Modify velocities based on previous best and global (or neighborhood) best
4. Terminate on some condition
5. Go to step 2

Figure 9. Particle swarm optimization process.

6.3.1. *Parameters*

In the PSO, instead of using genetic operators, as in the case of GAs, each particle (individual) adjusts its flying according with its own and its companions experiences. Each particle is treated as a point in a *D*-dimensional space and is manipulated as described below in the original PSO algorithm:

$$v_{id} = v_{id} + c_1 rand()(p_{id} - x_{id}) + c_2 Rand()(p_{gd} - x_{id}) \quad (2a)$$

$$x_{id} = x_{id} + v_{id} \quad (2b)$$

where c_1 and c_2 are positive constants, *rand*() and *Rand*() are two random functions in the range [0,1], $X_i = (x_{i1}, x_{i2},\ldots, x_{iD})$ represents the ith particle, $P_i = (p_{i1}, p_{i2},\ldots, p_{iD})$ is the best previous position (the position giving the best fitness value) of the particle, the symbol *g* represents the index of the best particle among all particles in the population, and $V_i = (v_{i1}, v_{i2},\ldots, v_{iD})$ is the rate of the position change (velocity) for particle *i*.

Expression (2) represents the flying trajectory of a population of particles. Equation (2a) describes how the velocity is dynamically updated and equation (2b) the position update of the "flying" particles. Equation (2a) is divided in three parts, namely the momentum, the cognitive and the social parts. In the first part the velocity cannot be changed abruptly: it is adjusted based on the current velocity. The second part represents the learning from its own flying experience. The third part consists on the learning group flying experience [8].

6.3.2. *Topologies*

There are two different PSO topologies, namely the global version and the local version. In the global version of PSO, each particle flies through the search space with a velocity that is dynamically adjusted according to the particle's

personal best performance achieved so far and the best performance achieved so far by all particles. On the other hand, in the local version of PSO, each particle's velocity is adjusted according to its personal best and the best performance achieved so far within its neighborhood. The neighborhood of each particle is generally defined as topologically nearest particles to the particle at each side.

6.3.3. *Algorithm*

PSO is an evolutionary algorithm simple in concept, easy to implement and computationally efficient. The initial population of particles has a random generation. The initial velocity of each particle is initialized with zero. The following velocities are calculated applying equation (2a) and the new positions result from using equation (2b). In this way, each potential solution (particle) flies through the problem space. For each gene the corresponding velocity is calculated. Therefore, the new positions are as many as the number of genes in the chromosome. If the new values of the input genes result out of range, then a re-insertion function is used. If the calculated gate gene is not allowed, a new valid one is generated at random. These particles have memory and each one keeps information of its previous best position (*pbest*) and its corresponding fitness. The swarm has the *pbest* of all the particles and the particle with the greatest fitness is called the global best (*gbest*).

The basic concept of the PSO technique lies in accelerating each particle towards its *pbest* and *gbest* locations with a random weighted acceleration.

6.4. *Ant Colony Optimization (ACO)*

Ant Colony Optimization is a class of optimization algorithms modeled on the actions of an ant colony. Artificial 'ants' - simulation agents - locate optimal solutions by moving through a parameter space representing all possible solutions. Real ants lay down pheromones directing each other to resources while exploring their environment. The simulated 'ants' similarly record their positions and the quality of their solutions, so that in later simulation iterations more ants locate better solutions.

The first algorithm which can be classified within this framework was presented in 1991 by proposed by Colorni, Dorigo and Maniezzo [4, 5, 6] and, since then, many diverse variants of the basic principle have been reported in the literature. The essential trait of ACO algorithms is the combination of *a priori* information about the structure of a promising solution with *a posteriori* information about the structure of previously obtained good solutions.

In the real world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep traveling at random, but to instead follow the trail, returning and reinforcing it if they eventually find food.

Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path, by comparison, gets marched over faster, and thus the pheromone density remains high as it is laid on the path as fast as it can evaporate. Pheromone evaporation has also the advantage of avoiding the convergence to a locally optimal solution. If there was no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained.

Thus, when one ant finds a good (*i.e.* short) path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leads all the ants following a single path. The idea of the ant colony algorithm is to mimic this behavior with "simulated ants" walking around the graph representing the problem to solve.

6.4.1. *Algorithm*

ACO algorithms are typically used to solve minimum cost problems. Usually there are *N* nodes and *A* undirected arcs. There are also two working modes for the ants: either forwards or backwards and pheromones are only deposited in backward mode. The ant's memory allows them to retrace the path they have followed while searching for the destination node and before moving backward on their memorized path, they eliminate any loops from it. While moving backwards, the ants leave pheromones on the arcs they traversed. The ants evaluate the cost of the paths they have traversed and the shorter paths will receive a greater deposit of pheromones. An evaporation rule will be tied with the pheromones, which will reduce the chance for poor quality solutions.

At the beginning of the search process, a constant amount of pheromone is assigned to all arcs. When located at a node *i* an ant *k* uses the pheromone trail to compute the probability of choosing *j* as the next node:

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}}{\sum l \in N_i^k \tau_{il}^{\alpha}} & if\ j \in N_i^k \\ 0 & if\ j \notin N_i^k \end{cases} \qquad (3)$$

where N_i^k is the neighborhood on ant *k* when in node *i*.

When the arc (*i,j*) is traversed, the pheromone value changes as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau^k \quad (4)$$

By using rule of equation 4, it the probability that forthcoming ants will use this arc increases.

After each ant *k* has moved to the next node, the pheromones evaporate by the following equation to all the arcs:

$$\tau_{ij} \leftarrow (1-p)\tau_{ij}, \forall (i,j) \in A \quad (5)$$

where $p \in (0,1)$ is a parameter. Iteration is a completer cycle involving ants' movement, pheromone evaporation, and pheromone deposit.

7. Applications of EAs to Computer Music

Computational Intelligence techniques have been applied to musical problems across a wide range of areas, including algorithmic composition, artificial listening, musical cognition and sound synthesis [13, 14].

The most commonly used evolutionary computation techniques in music are genetic algorithms and genetic programming.

In 1991 Gibson and Byrne [15] employed a GA to produce and combine musical fragments and a neural network (trained on examples of "real" music) to evaluate their fitness.

A genetic algorithm is also a key part of the improvisation and accompaniment system called GenJam which has been developed since 1993 by Al Biles [16]. Al and GenJam are together known as the Al Biles Virtual Quintet and have performed many times to human audiences.

Since 1996 Rodney Waschka II has been using genetic algorithms for music composition including works such as Saint Ambrose [9] and his string quartets [10].

Thywissen (1996) developed a compositional system called GeNotator that introduces more structure into the evolutionary process by applying genetic algorithms to several aspects of a composition, at different levels of abstraction [18].

In 1997 Brad Johanson and Riccardo Poli [17] developed the GP-Music System which used genetic programming to breed melodies according to both human and automated ratings. Several systems for drum loop evolution

produced, including one commercial program called MuSing.

A good survey on applications of genetic techniques to music, up to 1999, can be found in [12].

CONGA is an interesting system by Tokui and Iba [50]. The paper focuses on rhythmic composition and combines genetic algorithms with genetic programming. A neural network learns user criteria so combining interactive evolutionary computation with automated evaluation of fitness function.

Marques et al [46] describe an application of GA to music composition based on the variant *Familial Competition*. The fitness function evaluates three aspects of the solutions namely harmony (chord intervals), tone (suitability of pitch for tone and scale) and melody (intervals between consecutive notes). The experiments enhance the importance of octave operators for good melody convergence.

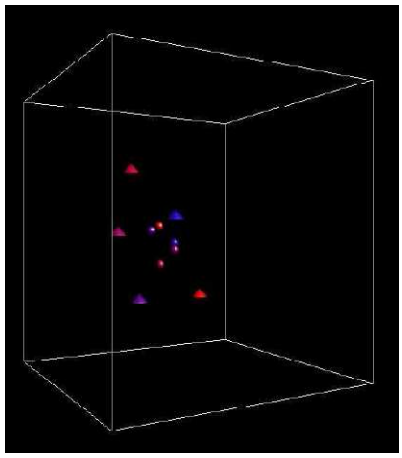

Figure 10. A five particle swarm (Blackwell, 2006, Swarming and Music [48])

Gartland-Jones and Copley [47] focus on the fitness function problem - human critic and automatic fitness assessment. Citing Bentley, *to date there has been no significant research aimed at understanding the difference between exploration and optimization*. New musical directions such as [48] Blackwell's *Swarm Music* are referred and the *IndagoSonus System* is described, a system that *tries to overcome […] difficulties by evolving a supplied musical starting point to a given musical target, a technique that has similarities to previous work […] for thematic bridging* [49].

In [51] Blackwell's approach consists of generating a *human comparable*

improviser within the context of freely improvised music. Music space is populated by musical events, each one of which corresponds to a note played at a certain time and with a definite loudness. *The three axes of music space are […] loudness, pulse and pitch* (figure 10). Interactions between humans and swarms and between two swarms are possible. You can listen to some results in http://www.timblackwell.com. Blackwell's approach focuses new search directions, the discovery of new music.

8. Conclusions and Future Work

Evolutionary computation seems the more promising tool when trying to algorithmically produce music. However, the term music designates an enormous and complex set of styles and tasks that we've summarized.

Melody generation has called the author's interest as being one of the more challenging problems, particularly when considering Western tonal music that, besides any theoretical, mathematical and physical foundations has the capability of evocating human emotions. From the evolutionary computation point of view, the evaluation of the generated solutions is made by the fitness function that, in this context, must be able to evaluate beauty and the emotional strength of the melodies produced. According to [25] *Affective Computing is computing that relates to, arises from, or deliberately influences emotion or other affective phenomena*. Not forgetting the advice of Biles [26] - *don't set the bar too high, don't try to solve the "western tonal music" problem* - a fitness function incorporating some kind of affective computing suggests further investigation.

However, research may also progress in a completely different direction: as music is, as every art, a matter of aesthetical evaluation and a result of cultural patterns, computers may be called to produce completely new musical pieces eventually based on unusual pitches, as well as different harmonies and timbres. The evaluation of such results according to human patterns is, at this moment, unpredictable. But what would J.S. Bach think of Elvis Presley?

Acknowledgments

The authors would like to acknowledge the GECAD Unit.

References

1. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D. (1998), Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications, Morgan Kaufmann.

2. Barricelli, Nils Aall (1954), Esempi numerici di processi di evoluzione, Methodos, pp. 45-68.
3. Crosby, Jack L. (1973), Computer Simulation in Genetics, John Wiley & Sons, London.
4. Dorigo, M., Maniezzo, V., Colorni, A., (1991), The ant system: an autocatalytic optimizing process, Technical Report TR91-016, Politecnico di Milano.
5. Colorni, A., Dorigo, M., Maniezzo, V., (1991), Distributed optimization by ant colonies, Proceedings of ECAL'91, European Conference on Artificial Life, Elsevier Publishing, Amsterdam.
6. Colorni, A., Dorigo, M., Maniezzo, V., (1991), Distributed optimization by ant colonies, Proceedings of ECAL'91, European Conference on Artificial Life, Elsevier Publishing, Amsterdam.
7. Kennedy, J., Eberhart, R. C., (1995), Particle Swarm Optimization. In Proc. of the IEEE Int. Conf. Neural Networks, pp 1942-1948, November.
8. Clerc, M., Kennedy, J., (2002), The Particle Swarm: explosion, stability, and convergence in a multi-dimensional complex space. In IEEE Trans. on Evolutionary Comp., vol. 6, pp. 58-73.
9. Capstone Records: Rodney Waschka II - Saint Ambrose. Mary Simoni, Computer Misc Journal, Vol. 31 Issue 3.
10. Miranda, E. R., Biles, J. A., (2007), Evolutionary Computer Music. Chapter: Composing with Genetic Algorithms: GenDash. Springer London
11. Beni, G., Wang, J., (1989), Swarm Intelligence in Cellular Robotic Systems, Proceed. NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy, June 26–30.
12. Burton, A. R., Vladimirova, T., (1999), Generation of Musical Sequences with Genetic Techniques. Computer Music Journal, 23:4, pp. 59-73, Winter
13. Todd, P. M., D. G. and Loy, eds. (1991), Music and Connectionism. Cambridge, Massachusetts: MIT Press.
14. Balaban, M., Ebcioglu, K. , Laske, O., (1992), Understanding Music with AI: Perspectives on Music Cognition. Cambridge, Massachusetts: MIT Press.
15. Gibson, P. M., Byrne, J. A., (1991), Neurogen, Musical Composition Using Genetic Algorithms and Cooperating Neural Networks. Proceedings of the Second International Conference on Artificial Neural Networks, pp. 309-313. Stevenege, England: Institute of Electrical Engineers.
16. Biles, J. A., (1994), GenJam: A Genetic Algorithm for Generating Jazz Solos. Proceedings of the 1994 International Computer Music Conference. San Francisco: International Computer Music Association.
17. Johanson, B. E., Poli, R., (1998), GP-Music: An Interactive Genetic Programming System for Music Generation with Automated Fitness Raters, Technical Report CSRP-98-13, School of Computer Science, The University of Birmingham.
18. Thywissen, K. (1996), GeNotator: An environment for investigating the

application of genetic algorithms in computer assisted composition. In Proceedings of the 1996 International Computer Music Conference. San Francisco: International Computer Music Association.
19. Varèse, E., (1998), The Liberation of Sound, *in* Contemporary Composers on Contemporary Music, New York Da Capo Press.
20. Nils Lennart Wallin, Steven Brown, Björn Merker, 2001, The origins of music, MIT Press.
21. Stehman, J., (1979), História da Música Europeia, 2ª ed., Bertrand.
22. Encyclopaedia Britannica 15th edition, 2007, Encyclopædia Britannica, Inc.
23. Zamacois, J., (1986), Teoria de la música, Livros 1 e 2, Barcelona: Editora Labor.
24. Britannica online, (2008), http://www.britannica.com/ebc/article-9372879.
25. MIT Affective Computing Group, 2008, http://affect.media.mit.edu/ index.php.
26. Biles, J.A., (2005), Evolutionary Music Tutorial, GECO 2005, Washington DC, USA.
27. Stephan, R., (1978), Enciclopédia Meridiano Fischer Vol.7 Música, Editora Meridiano.
28. Albert, M., (1979), A Música Contemporânea, Salvat Editora do Brasil, S.A.
29. Bennet, R., (1997), Uma Breve História da Música, Cadernos de Música da Universidade de Cambridge, Rio de Janeiro, Jorge Zahar Editor.
30. Kennedy, M., (1994), Dicionário Oxford de Música, Publicações Dom Quixote.
31. Galway, J., (1983), A Música no Tempo, Gris Impressores.
32. Robertson, A., Stevens, D. (editores), (1960), História da Música Pelicano, Editora Ulisseia.
33. Deutsche Grammophon, (1997), História da Música, Edilibro, S.L.
34. Reis, C., (2007), Síntese de Sistemas Digitais por Computação Evolutiva, Tese de Doutoramento, UTAD - Universidade de Trás os Montes e Alto Douro, Vila Real, Portugal.
35. Baldassarre, G., Trianni, V., Bonani, M., Mondada, F., Dorigo, M., (2007), Self-Organized Coordinated Motion in Groups of Physically Connected Robots, IEEE Transactions on Systems, Man and Cybernetics, Vol. 37, N.1, February.
36. Millan, N., (2001), MSc. Computer Science Dissertation, The University of Birmingham.
37. Goldberg, D., (1989), Genetic Algorithms in Search, Optimization and Machine Learning, Addison- Wesley.
38. Bäck, T., Fogel, D. B., Michalewicz, Z., (1997), Handbook of Evolutionary Computation, Institute of Physics Publishing, Bristol, Oxford University Press.
39. Holland, J. H., (1975), Adaptation in natural and artificial systems, The University of Michigan Press.

40. Darwin, C., (1859), On the Origin of Species by Means of Natural Selection, or, the Preservation of Favoured Races in the Struggle for Life, London: J. Murray.
41. Davis, L., (1991), Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York.
42. Koza, J. R., (1992) Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press.
43. Michalewicz, Z., (1992), Genetic Algorithms + Data Structures=Evolution Programs, Third Revised and Extended Edition, Springer-Verlag Berlin Heidelberg, New York.
44. Mitchell, M., (1997), An introduction to genetic algorithms, Cambridge, MIT Press.
45. Pomeroy, P., (2003), An Introduction to Particle Swarm Optimization, http://www.adaptiveview.com.
46. Marques, M., Oliveira, V., Vieira, S., Rosa, A.C., (2000) Music composition using genetic evolutionary algorithms, Proceedings of the 2000 Congress on Evolutionary Computation (CEC-2000), vol. 1, 2000, pp. 714–719.
47. Gartland-Jones, A., Copley P., (2003) The Suitability of Genetic Algorithms for Musical Composition, Contemporary Music Review, Vol. 22, N0.3, 43-55.
48. Blackwel, T., (2006), Swarming and Music, http://www.timblackwell.com
49. Horner, A., Goldberg, (1991), Genetic Algorithms and Computer-Assisted Music Composition, proceedings of the 1991 International Computer Music Conference, pp. 479-482.
50. Tokui, N., Iba, H., (2000), Music Composition with Interactive Evolutionary Computation, http://www.miv.t.u-tokyo.ac.jp/ibalab/papers/2000/toku i-GA2K.pdf.
51. Blackwel, T., (2003), Swarm-Music: Improvised Music with Multi-Swarms, Blackwell, T. Swarm music: improvised music with multi-swarms In Proceedings of the AISB 03 Symposium on Artificial Intelligence and Creativity in Art and Science, University of Wales UK, pp.41-99.