ARBOR Ciencia, Pensamiento y Cultura
Vol. 189-764, noviembre-diciembre 2013, a082 | ISSN-L: 0210-1963
doi: http://dx.doi.org/10.3989/arbor.2013.764n6005

COMPUTATIONAL CREATIVITY

CREATIVIDAD COMPUTACIONAL

Ramon López de Mántaras Badia
Instituto de Investigación en Inteligencia Artificial
Consejo Superior de Investigaciones Científicas
Campus UAB, 08193 Bellaterra
E-mail: mantaras@iiia.csic.es

Citation/Cómo citar este artículo: López de Mántaras Badia, R. (2013). "Computational Creativity". *Arbor*, 189 (764): a082. doi: http://dx.doi.org/10.3989/arbor.2013.764n6005

Received: 10 July 2013. Accepted: 15 September 2013.

ABSTRACT: New technologies, and in particular artificial intelligence, are drastically changing the nature of creative processes. Computers are playing very significant roles in creative activities such as music, architecture, fine arts, and science. Indeed, the computer is already a canvas, a brush, a musical instrument, and so on. However, we believe that we must aim at more ambitious relations between computers and creativity. Rather than just seeing the computer as a tool to help human creators, we could see it as a creative entity in its own right. This view has triggered a new subfield of Artificial Intelligence called Computational Creativity. This article addresses the question of the possibility of achieving computational creativity through some examples of computer programs capable of replicating some aspects of creative behavior in the fields of music and science.

KEYWORDS: Artificial Intelligence, Computational Creativity.

RESUMEN: Las nuevas tecnologías y en particular la Inteligencia Artificial están cambiando de forma importante la naturaleza del proceso creativo. Los ordenadores están jugando un papel muy significativo en actividades artísticas tales como la música, la arquitectura, las bellas artes y la ciencia. Efectivamente, el ordenador ya es el lienzo, el pincel, el instrumento musical, etc. Sin embargo creemos que debemos aspirar a relaciones más ambiciosas entre los ordenadores y la creatividad. En lugar de verlos solamente como herramientas de ayuda a la creación, los ordenadores podrían ser considerados agentes creativos. Este punto de vista ha dado lugar a un nuevo subcampo de la Inteligencia Artificial denominado Creatividad Computacional. En este artículo abordamos la cuestión de la posibilidad de de alcanzar dicha creatividad computacional mediante algunos ejemplos de programas de ordenador capaces de replicar algunos aspectos relacionados con el comportamiento creativo en los ámbitos de la música y la ciencia.

PALABRAS CLAVE: Inteligencia Artificial, Creatividad Computacional.

INTRODUCTION

Computational creativity is the study of building software that exhibits behaviour that would be deemed creative in humans. Such creative software can be used for autonomous creative tasks, such as inventing mathematical theories, writing poems, painting pictures, and composing music. However, computational creativity studies also enable us to understand human creativity and to produce programs for creative people to use, where the software acts as a creative collaborator rather than a mere tool. Historically, it's been difficult for society to come to terms with machines that purport to be intelligent and even more difficult to admit that they might be creative. Even within computer science, people are still sceptical about the creative potential of software. A typical statement of detractors of computational creativity is that "simulating artistic techniques means also simulating human thinking and reasoning, especially creative thinking. This is impossible to do using algorithms or information processing systems." We couldn't disagree more. As is hopefully evident from the examples in this paper, creativity is not some mystical gift that is beyond scientific study but rather something that can be investigated, simulated, and harnessed for the good of society. And while society might still be catching up, computational creativity as a discipline has come of age. This maturity is evident in the amount of activity related to computational creativity in recent years; in the sophistication of the creative software we are building; in the cultural value of the artefacts being produced by our software; and most importantly, in the consensus we are finding on general issues of computational creativity.

Computational creativity is a very lively subject area, with many issues still open to debate. For instance, many people still turn to the Turing test (Turing, 1950) to approximate the value of the artefacts produced by their software. That is, if a certain number of people cannot determine which artefacts were produced by computer and which were produced by a human, then the software is doing well. Other people believe that the Turing test is inappropriate for creative software. One has to ask the question, "Under full disclosure, would people value the artefacts produced by a computer as highly as they would the human produced ones?" In some domains, the answer could be yes: for instance, a joke is still funny whether or not it is produced by a computer. In other domains, such as the visual arts, however, the answer is very likely to be no. This highlights the fact that the production process, and not just the outcome of it, is taken into account when assessing artworks. Hence, one could argue that such Turing-style tests are essentially setting the Computers up for a fall.

Creativity seems mysterious because when we have creative ideas it is very difficult to explain how we got them and we often talk about vague notions like "inspiration" and "intuition" when we try to explain creativity. The fact that we are not conscious of how a creative idea manifests itself does not necessarily imply that a scientific explanation cannot exist. As a matter of fact, we are not aware of how we perform other activities such as language understanding, pattern recognition, and so on, but we have better and better AI techniques able to replicate such activities.

Since nothing can arise from the emptiness, we must understand that every creative work or creative idea always is preceded by a historical - cultural scheme, it is a fruit of the cultural inheritance and the lived experiences. As Margaret Boden states in her book *Artificial Intelligence and Natural Man*:

> "Probably the new thoughts that originate in the mind are not completely new, because have their seeds in representations that already are in the mind. To put it differently, the germ of our culture, all our knowledge and our experience, is behind each creative idea. The greater the knowledge and the experience, the greater the possibility of finding an unthinkable relation that leads to a creative idea. If we understand creativity like the result of establishing new relations between pieces of knowledge that we already have, then the more previous knowledge one has the more capacity to be creative".

With this understanding in mind, an operational, and widely accepted, definition of creativity is: "a creative idea is a novel and valuable combination of known ideas". In other words, physical laws, theorems, musical pieces can be generated from a finite set of existing elements and, therefore, creativity is an advanced form of problem solving that involves memory, analogy, learning, and reasoning under constraints, among other things, and is therefore possible to replicate by means of computers.

This article addresses the question of the possibility of achieving computational creativity through some examples of computer programs capable of replicating some aspects of creative behaviour in the fields of music and science. We did not intend to cover the full range of AI approaches to computational creativity and we could not include the many existing areas of application.

ARBOR Vol. 189-764, noviembre-diciembre 2013, a082. ISSN-L: 0210-1963 doi: http://dx.doi.org/10.3989/arbor.2013.764n6005

For further reading regarding computational creativity in general, I recommend the books by Boden (1991, 1994, 2009), Dartnall (1994), Partridge and Rowe (1994), and Bentley and Corne (2002); as well as the papers by Rowe and Partridge (1993), Buchanan (2001) and the recent special issue of AI Magazine edited by Colton, López de Mántaras and Stock (2009).

COMPUTATIONAL CREATIVITY IN MUSIC

Artificial Intelligence has played a crucial role in the history of computer music almost since its beginnings in the 1950s. However, until quite recently, most effort had been on compositional and improvisational systems and little efforts had been devoted to expressive performance. In this section we review a selection of some significant achievements in AI approaches to music composition, music performance, and improvisation, with an emphasis on the synthesis of expressive music.

Composing music

Hiller and Isaacson (1958) work, on the ILLIAC computer, is the best known pioneering work in computer music. Their chief result is the *Illiac Suite*, a string quartet composed following the "generate and test" problem solving approach. The program generated notes pseudo-randomly by means of Markov chains. The generated notes were next tested by means of heuristic compositional rules of classical harmony and counterpoint. Only the notes satisfying the rules were kept. If none of the generated notes satisfied the rules, a simple backtracking procedure was used to erase the entire composition up to that point, and a new cycle was started again. The goals of Hiller and Isaacson excluded anything related to expressiveness and emotional content. In an interview (Schwanauer and Levitt, 1993, p. 21), Hiller and Isaacson said that, before addressing the expressiveness issue, simpler problems needed to be handled first. We believe that this was a very correct observation in the fifties. After this seminal work, many other researchers based their computer compositions on Markov probability transitions but also with rather limited success judging from the standpoint of melodic quality. Indeed, methods relying too heavily on markovian processes are not informed enough to produce high quality music consistently.

However, not all the early work on composition relies on probabilistic approaches. A good example is the work of Moorer (1972) on tonal melody generation. Moorer's program generated simple melodies, along with the underlying harmonic progressions, with simple internal repetition patterns of notes. This approach relies on simulating human composition processes using heuristic techniques rather than on Markovian probability chains. Levitt (1983) also avoided the use of probabilities in the composition process. He argues that: "randomness tends to obscure rather than reveal the musical constraints needed to represent simple musical structures". His work is based on constraint-based descriptions of musical styles. He developed a description language that allows musically meaningful transformations of inputs, such as chord progressions and melodic lines, to be expressed through a series of constraint relationships that he calls "style templates". He applied this approach to describe a traditional jazz walking bass player simulation as well as a two-handed ragtime piano simulation.

The early systems by Hiller-Isaacson and Moorer were both based also on heuristic approaches. However, possibly the most genuine example of early use of AI techniques is the work of Rader (1974). Rader used rule-based AI programming in his musical round (a circle canon such as *Frère Jacques*) generator. The generation of the melody and the harmony were based on rules describing how notes or chords may be put together. The most interesting AI component of this system are the applicability rules, determining the applicability of the melody and chord generation rules, and the weighting rules indicating the likelihood of application of an applicable rule by means of a weight. We can already appreciate the use of metaknowledge in this early work.

AI pioneers such as Herbert Simon or Marvin Minsky also published works relevant to computer music. Simon and Sumner (1968) describe a formal pattern language for music, as well as a pattern induction method, to discover patterns more or less implicit in musical works. One example of pattern that can be discovered is "the opening section is in C Major, it is followed by a section in dominant and then a return to the original key". Although the program was not completed, it is worth noticing that it was one of the firsts in dealing with the important issue of music modelling, a subject that has been, and still is, widely studied. For example, the use of models based on generative grammars has been, and continues to be, an important and very useful approach in music modelling (Lerdahl and Jackendoff, 1983).

Marvin Minsky, in his well known paper "Music, Mind, and Meaning" (Minsky, 1981), addresses the

ARBOR Vol. 189-764, noviembre-diciembre 2013, a082. ISSN-L: 0210-1963 doi: http://dx.doi.org/10.3989/arbor.2013.764n6005

important question of "how music impresses our minds". He applies his concepts of agent and its role in a society of agents as a possible approach to shed light on that question. For example, he hints that one agent might do nothing more than noticing that the music has a particular rhythm. Other agents might perceive small musical patterns such as repetitions of a pitch; differences such as the same sequence of notes played one fifth higher, etc. His approach also accounts for more complex relations within a musical piece by means of higher order agents capable of recognizing large sections of music. It is important to clarify that in that paper Minsky does not try to convince the reader about the question of the validity of his approach, he just hints at its plausibility.

Among the compositional systems there is a large number dealing with the problem of automatic harmonization using several AI techniques. One of the earliest works is that of Rothgeb (1969). He wrote a SNOBOL program to solve the problem of harmonizing the unfigured bass (given a sequence of bass notes infer the chords and voice leadings that accompany those bass notes) by means of a set of rules such as "If the bass of a triad descends a semitone, then the next bass note has a sixth". The main goal of Rothgeb was not the automatic harmonization itself but to test the computational soundness of two bass harmonization theories from the eighteenth century.

One of the most complete works on harmonization is that of Ebcioglu (1993). He developed an expert system, CHORAL, to harmonize chorales in the style of J.S. Bach. CHORAL is given a melody and produces the corresponding harmonization using heuristic rules and constraints. The system was implemented using a logic programming language designed by the author. An important aspect of this work is the use of sets of logical primitives to represent the different viewpoints of the music (chords view, time-slice view, melodic view, etc.). This was done to tackle the problem of representing large amounts of complex musical knowledge.

MUSACT (Bharucha, 1993) uses Neural Networks to learn a model of musical harmony. It was designed to capture musical intuitions of harmonic qualities. For example, one of the qualities of a dominant chord is to create in the listener the expectancy that the tonic chord is about to be heard. The greater the expectancy, the greater the feeling of consonance of the tonic chord. Composers may choose to satisfy or violate these expectancies to varying degree. MUSACT is capable of learning such qualities and generate graded expectancies in a given harmonic context.

In HARMONET (Feulner, 1993), the harmonization problem is approached using a combination of neural networks and constraint satisfaction techniques. The neural network learns what is known as the harmonic functionality of the chords (chords can play the function of tonic, dominant, subdominant, etc) and constraints are used to fill the inner voices of the chords. The work on HARMONET was extended in the MELONET system (Hörnel and Degenhardt, 1997; Hörnel and Menzael, 1998). MELONET uses a neural network to learn and reproduce higher-level structure in melodic sequences. Given a melody, the system invents a baroque-style harmonization and variation of any chorale voice. According to the authors, HARMONET and MELONET together form a powerful music-composition system that generates variations whose quality is similar to those of an experienced human organist.

Pachet and Roy (1998) also used constraint satisfaction techniques for harmonization. These techniques exploit the fact that both the melody and the harmonization knowledge impose constraints on the possible chords. Efficiency is, however, a problem with purely constraint satisfaction approaches.

In (Sabater *et al*., 1998), the problem of harmonization is approached using a combination of rules and case-based reasoning. This approach is based on the observation that purely rule-based harmonization usually fails because, in general, "the rules don't make the music, it is the music that makes the rules". Then, instead of relying only on a set of imperfect rules, why not making use of the source of the rules, that is the compositions themselves? Case-based reasoning allows the use of examples of already harmonized compositions as cases for new harmonizations. The system harmonizes a given melody by first looking for similar, already harmonized, cases, when this fails, it looks for applicable general rules of harmony. If no rule is applicable, the system fails and backtracks to the previous decision point. The experiments have shown that the combination of rules and cases results in much fewer failures in finding an appropriate harmonization than using either technique alone. Another advantage of the case-based approach is that each newly correctly harmonized piece can be memorized and made available as a new example to harmonize other melodies; that is, a learning-by-experience process takes place. Indeed, the more examples the system has, the less often the system needs to resort to the rules and therefore it fails less. MUSE (Schwanauer, 1993) is also a learning system that extends

ARBOR Vol. 189-764, noviembre-diciembre 2013, a082. ISSN-L: 0210-1963 doi: http://dx.doi.org/10.3989/arbor.2013.764n6005

an initially small set of voice leading constraints by learning a set of rules of voice doubling and voice leading. It learns by reordering the rules agenda and by chunking the rules that satisfy the set of voice leading constraints. MUSE successfully learned some of the standard rules of voice leading included in traditional books of tonal music.

Morales-Manzanares *et al*. (2001) developed a system called SICIB capable of composing music using body movements. This system uses data from sensors attached to the dancer and applies inference rules to couple the gestures with the music in real time.

Certainly the best-known work on computer composition using AI is David Cope's EMI project (Cope, 1987, 1990). This work focuses on the emulation of styles of various composers. It has successfully composed music in the styles of Cope, Mozart, Palestrina, Albinoni, Brahms, Debussy, Bach, Rachmaninoff, Chopin, Stravinsky, and Bartok. It works by searching for recurrent patterns in several (at least two) works of a given composer. The discovered patterns are called signatures. Since signatures are location dependent, EMI uses one of the composer's works as a guide to fix them to their appropriate locations when composing a new piece. To compose the musical motives between signatures, EMI uses a compositional rule analyzer to discover the constraints used by the composer in his works. This analyzer counts musical events such as voice leading directions; use of repeated notes, etc. and represents them as a statistical model of the analyzed works. The program follows this model to compose the motives to be inserted in the empty spaces between signatures. To properly insert them, EMI has to deal with problems such as: linking initial and concluding parts of the signatures to the surrounding motives avoiding stylistic anomalies, maintaining voice motions, maintaining notes within a range, etc. Proper insertion is achieved by means of an Augmented Transition Network (Woods, 1970). The results, although not perfect, are quite consistent with the style of the composer.

Synthesizing expressive music

One of the main limitations of computer-generated music has been its lack of expressiveness, that is, lack of "gesture". Gesture is what musicians call the nuances of performance that are uniquely and subtly interpretive or, in other words, creative.

One of the first attempts to address expressiveness in music is that of Johnson (1992). She developed an expert system to determine the tempo and the articulation to be applied when playing Bach's fugues from *The Well-Tempered Clavier*. The rules were obtained from two expert human performers. The output gives the base tempo value and a list of performance instructions on note duration and articulation that should be followed by a human player. The results very much coincide with the instructions given in well known commented editions of *The Well-Tempered Clavier*. The main limitation of this system is its lack of generality because it only works well for fugues written on a 4/4 meter. For different meters, the rules would be different. Another obvious consequence of this lack of generality is that the rules are only applicable to Bach fugues.

The work of the KTH group from Stockholm (Friberg, 1995; Friberg *et al*., 1998, 2000; Bresin, 2001) is one of the best-known long-term efforts on performance systems. Their current *Director Musices* system incorporates rules for tempo, dynamic, and articulation transformations constrained to MIDI. These rules are inferred both from theoretical musical knowledge and experimentally by training, specially using the so-called analysis-by-synthesis approach. The rules are divided in three main classes: Differentiation rules, which enhance the differences between scale tones; Grouping rules, which show what tones belong together; and Ensemble rules, that synchronize the various voices in an ensemble.

Canazza *et al*. (1997) developed a system to analyze how the musician's expressive intentions are reflected in the performance. The analysis reveals two different expressive dimensions: one related to the energy (dynamics) and the other one related to the kinetics (rubato) of the piece. The authors also developed a program for generating expressive performances according to these two dimensions.

The work of Dannenberg and Derenyi (1998) is also a good example of articulation transformations using manually constructed rules. They developed a trumpet synthesizer that combines a physical model with a performance model. The goal of the performance model is to generate control information for the physical model by means of a collection of rules manually extracted from the analysis of a collection of controlled recordings of human performance.

Another approach taken for performing tempo and dynamics transformation is the use of neural network techniques. In Bresin (1998), a system that combines symbolic decision rules with neural networks is im-

plemented for simulating the style of real piano performers. The outputs of the neural networks express time and loudness deviations. These neural networks extend the standard feed-forward network trained with the back propagation algorithm with feedback connections from the output neurons to the input neurons.

We can see that, except for the work of the KTH group that considers three expressive resources, the other systems are limited to two resources such as rubato and dynamics, or rubato and articulation. This limitation has to do with the use of rules. Indeed, the main problem with the rule-based approaches is that it is very difficult to find rules general enough to capture the variety present in different performances of the same piece by the same musician and even the variety within a single performance (Kendall and Carterette, 1990). Furthermore, the different expressive resources interact with each other. That is, the rules for dynamics alone change when rubato is also taken into account. Obviously, due to this interdependency, the more expressive resources one tries to model, the more difficult it is to find the appropriate rules.

We developed a system called SaxEx (Arcos *et al*., 1998), a computer program capable of synthesizing high quality expressive tenor sax solo performances of jazz ballads based on cases representing human solo performances. As mentioned above, previous rule-based approaches to that problem could not deal with more than two expressive parameters (such as dynamics and rubato) because it is too difficult to find rules general enough to capture the variety present in expressive performances. Besides, the different expressive parameters interact with each other making it even more difficult to find appropriate rules taking into account these interactions.

With CBR, we have shown that it is possible to deal with the five most important expressive parameters: dynamics, rubato, vibrato, articulation, and attack of the notes. To do so, SaxEx uses a case memory containing examples of human performances, analyzed by means of spectral modeling techniques and background musical knowledge. The score of the piece to be performed is also provided to the system. The core of the method is to analyze each input note determining (by means of the background musical knowledge) its role in the musical phrase it belongs to, identify and retrieve (from the case-base of human performances) notes with similar roles, and finally, transform the input note so that its expressive properties (dynamics, rubato, vibrato, articulation, and attack) match those of the most similar retrieved note. Each note in the case base is annotated with its role in the musical phrase it belongs to, as well as with its expressive values. Furthermore, cases do not contain just information on each single note but they include contextual knowledge at the phrase level. Therefore, cases in this system have a complex object-centered representation.

Although limited to monophonic performances, the results are very convincing and demonstrate that CBR is a very powerful methodology to directly use the knowledge of a human performer that is implicit in her playing examples rather than trying to make this knowledge explicit by means of rules. Some audio results can be listened at *http://www.iiia.csic.es/%7Earcos/noos/Demos/Example.html*. More recent papers (Arcos and López de Mántaras, 2001; López de Mántaras and Arcos, 2002), describe this system in great detail.

Based on the work on SaxEx, we developed TempoExpress (Grachten *et al*. 2004), a case-based reasoning system for applying musically acceptable tempo transformations to monophonic audio recordings of musical performances. TempoExpress has a rich description of the musical expressivity of the performances, that includes not only timing deviations of performed score notes, but also represents more rigorous kinds of expressivity such as note ornamentation, consolidation, and fragmentation. Within the tempo transformation process, the expressivity of the performance is adjusted in such a way that the result sounds natural for the new tempo. A case base of previously performed melodies is used to infer the appropriate expressivity. The problem of changing the tempo of a musical performance is not as trivial as it may seem because it involves a lot of musical knowledge and creative thinking. Indeed, when a musician performs a musical piece at different tempos the performances are not just time-scaled versions of each other (as if the same performance were played back at different speeds). Together with the changes of tempo, variations in musical expression are made (Desain and Honing, 1993). Such variations do not only affect the timing of the notes, but can also involve for example the addition or deletion of ornamentations, or the consolidation/fragmentation of notes. Apart from the tempo, other domain specific factors seem to play an important role in the way a melody is performed, such as meter, and phrase structure. Tempo transformation is one of the audio post-processing tasks manually done in audio-labs. Automatizing this process may, therefore, be of industrial interest.

ARBOR Vol. 189-764, noviembre-diciembre 2013, a082. ISSN-L: 0210-1963 doi: http://dx.doi.org/10.3989/arbor.2013.764n6005

Other applications of CBR to expressive music are those of Suzuki *et al*. (1999), and those of Tobudic and Widmer (2003, 2004). Suzuki *et al*. (1999), use examples cases of expressive performances to generate multiple performances of a given piece with varying musical expression, however they deal only with two expressive parameters. Tobudic and Widmer (2003) apply instance-based learning (IBL) also to the problem of generating expressive performances. The IBL approach is used to complement a note-level rule-based model with some predictive capability at the higher level of musical phrasing. More concretely, the IBL component recognizes performance patterns, of a concert pianist, at the phrase level and learns how to apply them to new pieces by analogy. The approach produced some interesting results but, as the authors recognize, was not very convincing due to the limitation of using an attribute-value representation for the phrases. Such simple representation cannot take into account relevant structural information of the piece, both at the sub-phrase level and at the inter-phrasal level. In a subsequent paper, Tobudic and Widmer (2004), succeeded in partly overcoming this limitations by using a relational phrase representation.

Widmer *et al*. (2009) describe a computer program that learns to expressively perform classical piano music. The approach is data intensive and based on statistical learning. Performing music expressively certainly requires high levels of creativity, but the authors take a very pragmatic view to the question of whether their program can be said to be creative or not and claim that "creativity is in the eye of the beholder." In fact, the main goal of the authors is to investigate and better understand music performance as a creative human behaviour by means of AI methods.

The possibility for a computer to play expressively is a fundamental component of the so-called *hyper-instruments*. These are instruments designed to augment an instrument sound with such idiosyncratic nuances as to give it human expressiveness and a rich, live sound. To make an hyper-instrument, take a traditional instrument, like for example a cello, and connect it to a computer through electronic sensors in the neck and in the bow, equip also with sensors the hand that holds the bow and program the computer with a system similar to SaxEx that allows to analyse the way the human interprets the piece, based on the score, on musical knowledge and on the readings of the sensors. The results of this analysis allow the hyper-instrument to play an active role altering aspects such as timbre, tone, rhythm and phrasing as well as generating an accompanying voice. In other words, you get an instrument that can be its own intelligent accompanist. Tod Machover, from MIT's Media Lab, developed such an hyper cello and the great cello player Yo-Yo Ma premiered, playing the hyper cello, a piece, composed by Tod Machover, called "Begin Again Again..." at the Tanglewood Festival several years ago.

Improvising music

Music improvisation is a very complex creative process that has also been computationally modelled. It is often referred to as "composition on the fly" and, therefore, it is, creatively speaking, more complex than composition and it is probably the most complex of the three music activities surveyed here. An early work on computer improvisation is the Flavours Band system of Fry (1984). Flavours Band is a procedural language, embedded in LISP, for specifying jazz and popular music styles. Its procedural representation allows the generation of scores in a pre-specified style by making changes to a score specification given as input. It allows combining random functions and musical constraints (chords, modes, etc.) to generate improvisational variations. The most remarkable result of Flavours Band was an interesting arrangement of the bass line, and an improvised solo, of John Coltrane's composition "Giant Steps".

GenJam (1994) builds a model of a jazz musician learning to improvise by means of a genetic algorithm. A human listener plays the role of fitness function by rating the offspring improvisations. Papadopoulos and Wiggins (1998) also used a genetic algorithm to improvise jazz melodies on a given chord progression. Contrarily to GenJam, the program includes a fitness function that automatically evaluates the quality of the offspring improvisations rating eight different aspects of the improvised melody such as the melodic contour, notes duration, intervallic distances between notes, etc.

Franklin (2001) uses recurrent neural networks to learn how to improvise jazz solos from transcriptions of solo improvisations by saxophonist Sonny Rollins. A reinforcement learning algorithm is used to refine the behaviour of the neural network. The reward function rates the system solos in terms of jazz harmony criteria and according to Rollins style.

The lack of interactivity, with a human improviser, of the above approaches has been criticized (Thom, 2001) on the grounds that they remove the musician from the physical and spontaneous creation of a me-

ARBOR Vol. 189-764, noviembre-diciembre 2013, a082. ISSN-L: 0210-1963 doi: http://dx.doi.org/10.3989/arbor.2013.764n6005

lody. Although it is true that the most fundamental characteristic of improvisation is the spontaneous, real-time creation of a melody, it is also true that interactivity was not intended in these approaches and nevertheless they could generate very interesting improvisations. Thom (2001) with her Band-out-of-a-Box (BoB) system addresses the problem of real-time interactive improvisation between BoB and a human player. In other words, BoB is a "music companion" for real-time improvisation. Thom's approach follows Johnson-Laird's (1991) psychological theory of jazz improvisation. This theory opposes the view that improvising consists of rearranging and transforming pre-memorized "licks" under the constraints of a harmony. Instead he proposes a stochastic model based on a greedy search over a constrained space of possible notes to play at a given point in time. The very important contribution Thom makes is that her system learns these constraints, and therefore the stochastic model, from the human player by means of an unsupervised probabilistic clustering algorithm. The learned model is used to abstract solos into user-specific playing modes. The parameters of that learned model are then incorporated into a stochastic process that generates the solos in response to four bar solos of the human improviser. BoB has been very successfully evaluated by testing its real-time solo tradings in two different styles, that of saxophonist Charlie Parker, and that of violinist Stephane Grapelli.

Another remarkable interactive improvisation system was developed by Dannenberg (1993). The difference with Thom's approach is that in Dannenberg's system, music generation is mainly driven by the composer's goals rather than the performer's goals. Wessel's (1998) interactive improvisation system is closer to Thom's in that it also emphasizes the accompaniment and enhancement of live improvisations.

COMPUTATIONAL CREATIVITY IN SCIENCE

BACON (Langley *et al*., 1987) is a representative example of a program capable of rediscovering important scientific laws using the "generate and test" mechanism. BACON takes as inputs non-interpreted numerical data and, when successful, produces scientific laws that fit the data. Before proceeding with BACON operational details, it is important to notice that the fact that it "rediscovers" known laws does not preclude its interest as a computational model of a creative process since, in principle, there is no reason to believe that the cognitive processes involved in a genuine discovery are different from those involved in a rediscovery.

The discovery process of BACON is not a random one and it could not be because the space of possible functions to try is not finite and even if the search were limited to a finite subset, any useful scientific domain would be too large to allow random search. BACON uses several heuristics for searching selectively. First, starts with simple functions (as the linear function), then proceeds with more complex ones that are formed by multiplying or dividing pairs of functions. Second, BACON uses data to guide the selection of the next function to try. More precisely, it notices if one variable increases or decreases monotonically with respect to another. If it increases, it will test whether the ratios of the values of the variables are constant. If it decreases, it will test whether the products are constant. The main point is that BACON selects the next function to test depending on how previously tried functions fit the data. Third, in situations involving more than two variables, BACON follows the well-known experimental procedure of changing one independent variable at a time. Having found conditional dependencies among small sets of variables, it explores the effects of altering other variables.

With these simple means, and supplied with the actual data used by the original discoverers, BACON rediscovered Kepler's Third Law of planetary motion, Ohm's Law of electric current and resistance, Black's Law of temperature equilibrium for mixtures of liquids and many others.

In validating BACON as a theory of human discovery, Herbert Simon pointed out that BACON, interestingly enough, initially arrived at the same erroneous square law that Kepler himself had initially formulated, that is "the period of revolution of the planets varied as the square of their distance to the Sun". However, BACON rejected it because it did not fit the data well enough, and went on to discover the correct law. If BACON had used a larger error tolerance parameter it would have also made Kepler's mistake. According to Simon "BACON shows that theories of inspiration are constructed and tested in exactly the same manner as other scientific theories", that is, scientists need not to be "seized by god" to discover new laws. This is true; Simon proceeds, even in discovering new concepts as BACON also shows. Indeed, using the heuristic that when it discovers that there is an invariant relation in the interaction between two or more elements in a situation, it should assign a new property to the elements, and measure its magnitude by the relative strength of each element's action, BACON rediscovered the concept of inertial mass after noticing

ARBOR Vol. 189-764, noviembre-diciembre 2013, a082. ISSN-L: 0210-1963 doi: http://dx.doi.org/10.3989/arbor.2013.764n6005

that when pairs of bodies collide, the ratio of accelerations of any given pair is always the same. To do so, according to Simon, BACON defined a new property "P" and assigned a value 1 to that property for body A and a value inversely proportional to the magnitude of their accelerations in collisions with A to the other bodies. This procedure turns out to be a quite general heuristic for discovering new concepts and has been used by BACON in discovering the concepts of specific heat, refractive index, voltage, molecular weight and atomic weight among others. Again, inspiration turns out to be a by-product of ordinary heuristic search in Simon's saying.

In my opinion, Simon was too optimistic. BACON is too inflexible to be an acceptable computational model of scientific discovery. Indeed, its behaviour is completely determined by its fixed set of heuristics that guide an ad-hoc process of function approximation. There is no flexibility in the representation. To improve this, other programs such as GLAUBER (Langley *et al*., 1987) were written. These programs are able to induce structural and explanatory models of certain phenomena. In particular GLAUBER examines qualitative data and produces classifications that induce general laws. For example, given a series of results of chemistry experiments such as:

(reacts input (C1H, NaOH) output (NaCl))

GLAUBER induces the law:

(reacts input (acid, alkali) output (salt))

Another well known scientific discoverer is the *Automated Mathematician* or AM (Lenat, 1983), a program that (re)discovered mathematical concepts in number theory based on a hierarchy of about 100 basic concepts (sets, ordered pairs, basic operations such as union, intersection, etc.) and using some 250 heuristic rules to guide the discovery process. Each concept is represented by a set of slots containing information such as definition, examples, specializations, worth, and in the case of operations its domain and range. The heuristic rules are of four different types: *Fill*, *check*, *suggest* and *interest*. Thus AM can use *interest* rules to evaluate the interest of the concepts it discovers and, therefore, guide the search towards more promising concepts. *Fill* rules try to fill the examples slot with examples of a concept. *Check* rules verify correctness of the examples and also look for regularities. *Suggest* rules are considered when the program is running low on interesting things to do. Overall, AM is controlled by an agenda which maintains a list of tasks sorted according to the interest of the concepts involved, the number of reasons for suggesting the task and their worth and also the type of task. The top task is chosen relevant rules are gathered and applied resulting possibly in new concepts and new tasks.

AM rediscovered natural numbers, addition, multiplication, prime numbers, the prime factorization theorem and the, yet unproven, Goldbach's conjecture (any even number greater than two is the sum of two different primes) but soon reached its limits and failed to produce other interesting concepts. Instead it proposed many boring tasks. According to Lenat, such early limitation was due to the fixed nature of the heuristics that worked well with simple concepts such as sets but did not work with higher-level concepts such as numbers. With the aim of overcoming this limitation, he extended AM to automatically modify its heuristic rules. The result was very disappointing since a large number of useless rules were created. According to Lenat (Rowe and Partridge, 1993), the reason was that many heuristics generated new concepts by syntactic manipulation of void concepts. When these were definitions of mathematical concepts they were implemented as pieces of Lisp code and a minor modification to this Lisp code usually produced meaningful results. However, when similar modifications were made to the Lisp code representing a heuristic rule, the results were meaningless. This is because heuristics operate at a much higher level than Lisp and many lines of Lisp code are required to implement each heuristic. Lenat, therefore, decided that a new representation language at a higher level than Lisp was needed and he developed EURISKO (Lenat, 1983) in which heuristics were represented as frames with slots containing small pieces of Lisp code. The idea was to produce meaningful changes in the heuristics by mutations in the values of the slots. EURISKO was applied to VLSI design, space-ship fleet design and number theory and it worked quite well when interacting with a user that could eliminate obviously bad heuristics. However it did not go further than AM in number theory. It seems that, once more, the heuristics were not good enough. Going further in number theory requires dealing with concepts in many other mathematical domains such as algebra, geometry, graph theory, etc. and neither AM nor EURISKO could deal with such concepts.

Besides, AM has been criticized by Ritchie and Hanna (1984) on other grounds. According to Ritchie and Hanna, many rules contain "special purpose hacks", never described by Lenat, that seem to have been written for the specific purpose of making the most

ARBOR Vol. 189-764, noviembre-diciembre 2013, a082. ISSN-L: 0210-1963 doi: http://dx.doi.org/10.3989/arbor.2013.764n6005

interesting "discoveries". The precise extent of AM creativity is, therefore, rather unclear.

The major limitation of all these programs is that what to look for is built into the heuristics and that such heuristics are too inflexible. A creative idea also involves "breaking rules" or dropping constraints, and a mechanism to do so is that of reasoning by analogy. An excellent example of a human discovery, based on analogy and constraints dropping, is that of Kekule's discovery of the benzene-ring structure (other well known examples of constraint dropping are non-Euclidean geometry and Schoenberg's non tonal music). Kekule described his discovery as follows (Boden, 1991):

> "I turned the chair to the fire and dozed. Again the atoms were gamboling before my eyes... (My mental eye) could distinguish larger structures, of manifold conformation; long rows, sometimes more closely fitted together; all twining and twisting in snakelike motion. But look! What was that? One of the snakes had seized hold of its own tail, and the form whirled mockingly before my eyes. As if by a flash of lightning I awoke."

This vision was the origin of his discovery that the benzene molecule was a ring and not a chain. The analogy between snakes and molecules was certainly a fundamental aspect of the discovery that triggered the dropping of the constraint that molecules should be chains (open curves) and allowing them to be closed curves (rings).

Psychologists have studied analogy for a long time and one of the main conclusions was that analogy is context-sensitive. Indeed, analogy in the context of poetry is not the same as in the context of science. COPYCAT (Mitchell and Hofstadter, 1990) was the first computational model of analogy and is heavily based on human psychology and particularly on context sensitivity. This system generates many candidate analogies but only those that are contextually appropriate are kept. COPYCAT constructs analogies between strings of letters. Let us see an example taken from Boden (1994). COPYCAT is given as input that the string *abc* rewrites into *abd*, and then is asked what the string *mrrjjj* will rewrite into. There are several acceptable answers to this depending on the context that COPYCAT has considered. For instance, if the context favours "successors" over "repetitions", then one answer would be *mrrjjk*. But if it favours repetitions then the answer is *mrrjjjj*. Another interesting analogy (Boden, 1994) found by COPYCAT is the following one: If *abc* rewrites into *abd*, what will *xyz* rewrite into? One very creative answer of COPYCAT was *wyz*. This answer involves simultaneously relating "left" with "right", "first letter of the alphabet" with "last letter of the alphabet", and "successor" with "predecessor". In spite of operating in a very limited domain, COPYCAT shows interesting features of creative processing.

Other programs such as MECHEM (Valdés-Pérez, 1995) incorporate sophisticated techniques of reasoning by analogy and constraint satisfaction that allow them to reason about the structural transformations that take place in chemically reacting molecules with the aim of eliciting the internal, non observable, mechanisms involved in the reactions based on empirical evidence. MECHEM has discovered and explained mechanisms in the hydrogenolosis of methane that coincide fully with results published just a couple of years earlier in the Journal "Catalysis Today". Understanding the internal mechanisms involved in chemical reactions has an enormous practical interest since that knowledge can suggest better ways of controlling the reactions. It has been conjectured that programs like MECHEM may well be necessary to understand the mechanisms implicit in complex chemical reactions.

Bob Holmes (1996) reported the development of a "Creativity Machine" developed by the materials scientist Steve Thaler. Among other applications, it discovered ultra-hard materials. This system is based on a large network with inputs and outputs representing every possible quantum state for every electron in every atom of a molecule. Thaler trained this network by showing it about 200 examples of two-element molecules, such as water and iron oxide, to teach it plausible combinations and proportions. Holmes reports that the machine correctly identified known ultra-hard materials such as boron nitride and boron carbide, even though it had never seen these during training. It also proposed a material, C3N4 that a group of theoreticians from Harvard had almost simultaneously suggested as a likely ultra-hard material. Holmes adds that the system also pointed out several untested polymers of boron, beryllium, or carbon doped with small amounts of hydrogen. The "Creativity Machine" was licensed to a company to develop new ultra-hard materials and high-temperature superconductors.

Another recent very remarkable achievement is a robot-scientist, called ADAM (King *et al*. 2004), which conducted experiments on yeast using AI techniques. The goal of these experiments was to determine the function of several gene knockouts by varying the quantities of nutrient provided to the yeast. The robot

ARBOR Vol. 189-764, noviembre-diciembre 2013, a082. ISSN-L: 0210-1963 doi: http://dx.doi.org/10.3989/arbor.2013.764n6005

used a machine learning technique known as *inductive logic programming* to select those experiments that could discriminate between different hypotheses. Feedback on each experiment was provided by data reporting yeast survival or death. The most accurate robot strategy outperformed humans doing the same task. Stephen Muggleton, one of the designers of ADAM, predicts the development of the first micro-fluidic robot scientist within the next years. A micro-fluidic robot scientist, according to Muggleton (2006), will combine active learning and autonomous experimentation with micro-fluidic technology (Fletcher *et al*., 2001). He argues that nowadays scientists can already build miniaturized laboratories on a chip using micro-fluidics and since these chips can perform chemical synthesis and testing at high speed, one can imagine miniaturizing the robot-scientist technology with the goal of reducing the experimental cycle time from hours to milliseconds. Furthermore, Muggleton speculates that more flexibility could be added to the micro-fluidic machines by developing what he calls a "Chemical Turing Machine". The Chemical Turing Machine, according to Muggleton, would be "a universal processor capable of performing a broad range of chemical operations on both the reagents available to it at the start and those chemicals it later generates. The machine would automatically prepare and test chemical compounds but it would also be programmable, thus allowing much the same flexibility as a real chemist has in the lab."

Similarly to a standard Turing Machine, an automaton, introduced by Alan Turing to give a mathematically precise definition of algorithm (Turing, 1936, 1938), which consisted of an infinite tape and a set of very simple rules for moving the tape and manipulating the symbols that contains, a Chemical Turing Machine would be an automaton connected to a conveyor belt containing a series of flasks: the chemical Turing machine can move the conveyor to obtain distant flasks, and can mix and make tests on local flasks.

CONCLUDING REMARKS: APPARENTLY OR REALLY CREATIVE?

Margaret Boden pointed out that even if an artificially intelligent computer would be as creative as Bach or Einstein, for many it would be just apparently creative but not really creative. I fully agree with her in the two main reasons for such rejection. These reasons are: the lack of intentionality and our reluctance to give a place in our society to artificially intelligent agents. The lack of intentionality is a direct consequence of Searle's Chinese room argument (Searle, 1980), which states that computer programs can only perform syntactic manipulation of symbols but are unable to give them any semantics. This criticism is based on an erroneous concept of what a computer program is. Indeed, a computer program does not only manipulate symbols but also triggers a chain of cause-effect relations inside the computer hardware and this fact is relevant for intentionality since it is generally admitted that intentionality can be explained in terms of causal relations. However, it is also true that existing computer programs lack too many relevant causal connections to exhibit intentionality, but perhaps future, possibly anthropomorphic, "embodied" artificial intelligences --that is agents equipped not only with sophisticated software but also with different types of advanced sensors allowing to interact with the environment-- may have sufficient causal connections to exhibit intentionality.

Regarding social rejection, the reasons why we are so reluctant to accept that non biological agents can be creative (even biological ones as it is the case with "Nonja", a 20 years old painter from Vienna whose abstract paintings had been exhibited and appreciated in art galleries but that after knowing that she was an orang-utan from the Vienna Zoo, her work was much less appreciated!) is that they do not have a natural place in our society of human beings and a decision to accept them would have important social implications. It is therefore much simpler to say that they appear to be intelligent, creative, etc. instead of saying that they are. In a word, it is a moral but not a scientific issue. A third reason for denying creativity to computer programs is that they are not conscious of their accomplishments. However I agree with many AI scientists in thinking that the lack of consciousness is not a fundamental reason to deny the potential for creativity or even the potential for intelligence. After all, computers would not be the first example of unconscious creators, evolution is the first example as Stephen Jay Gould (1996) brilliantly points out: "If creation demands a visionary creator, then how does blind evolution manage to build such splendid new things as ourselves?"

ACKNOWLEDGEMENTS

This research has been partially supported by the EU FP7 PRAISE project #318770 and by the 2009-SGR-1434 Grant from the Generalitat de Catalunya.

REFERENCES

Arcos, J. L.; López de Mántaras, R. (2001). "An Interactive Case-Based Reasoning Approach for Generating Expressive Music". *Applied Intelligence,* 14 (1), pp. 115-129.

Arcos, J. L.; López de Mántaras, R.; Serra, X. (1998). "Saxex: A Case-Based Reasoning System for Generating Expressive Musical Performances. *Journal of New Music Research*, 27 (3), pp. 194-210.

Bentley, P. J.; Corne, D.W. (eds.) (2002). *Creative Evolutionary Systems*. Morgan Kaufmann.

Bharucha, J. (1993). MUSACT: "A connectionist model of musical harmony". In S. M. Schwanauer and D.A. Levitt (eds.), *Machine Models of Music* (pp. 497-509). Cambridge, Mass.: The MIT Press.

Biles, J. A. (1994). "GenJam: A genetic algorithm for generating Jazz solos". In *Proceedings of the 1994 International Computer Music Conference*. San Francisco, Calif., International Computer Music Association.

Boden, M. (1991). *The Creative Mind: Myths and Mechanisms*. New York: Basic Books.

Boden, M. (ed.) (1994). *Dimensions of Creativity*. Cambridge, Mass.: The MIT Press.

Boden, M. (2009). "Computers Models of Creativity". *AI Magazine*, 30 (3), pp. 23-34.

Bresin, R. (1998). "Artificial neural networks based models for automatic performance of musical scores". *Journal of New Music Research*, 27 (3), pp. 239-270.

Bresin, R. (2001). "Articulation rules for automatic music performance". In *Proceedings of the 2001 International Computer Music Conference*. San Francisco, Calif.: International Computer Music Association.

Buchanan, B. G. (2001). Creativity at the Metalevel: AAAI-2000 Presidential Address, *AI Magazine*, 22 (3), pp. 13-28.

Canazza, S.; De Poli, G.; Roda, A.; Vidolin, A. (1997). "Analysis and synthesis of expressive intention in a clarinet performance". In *Proceedings of the 1997 International Computer Music Conference* (pp. 113-120). San Francisco, Calif.: International Computer Music Association.

Colton, S.; López de Mántaras, R.; Stock, O. (2009). "Computational Creativity: Coming of Age". Special Issue of *AI Magazine*, vol. 30 no. 3 pp. 11-88.

Cope, D. (1987). "Experiments in Music Intelligence". In *Proceedings of the 1987 International Computer Music Conference*. San Francisco, Calif.: International Computer Music Association.

Cope, D. (1990). "Pattern Matching as an engine for the computer simulation of musical style". In *Proceedings of the 1990 International Computer Music Conference*. San Francisco, Calif.: International Computer Music Association.

Dannenberg, R. B. (1993). "Software design for interactive multimedia performance". *Interface*, 22 (3), pp. 213-218.

Dannenberg, R. B.; Derenyi, I. (1998). "Combining instrument and performance models for high quality music synthesis". *Journal of New Music Research*, 27 (3), pp. 211-238.

Dartnall, T. (ed.) (1994). *Artificial Intelligence and Creativity*. Kluwer Academic Pub.

Desain, P.; Honing, H. (1993). "Tempo curves considered harmful". In J. D. Kramer (ed.), Time in contemporary musical thought. *Contemporary Music Review,* 7 (2).

Ebcioglu, K. (1993). "An expert system for harmonizing four-part chorales". In S.M. Schwanauer and D.A. Levitt (eds.), *Machine Models of Music* (pp. 385-401). Cambridge, Mass.: The MIT Press.

Feulner, J. (1993). "Neural Networks that learn and reproduce various styles of harmonization". In *Proceedings of the 1993 International Computer Music Conference*. San Francisco: International Computer Music Association.

Fletcher, P.; Haswell, S.; Watts, P. & Zhang, X. (2001). "Lab-on-a-Chip Micro Reactors for Chemical Synthesis". *Dekker Encyclopedia of Nanoscience and Nanotechnology*.

Franklin, J. A. (2001). "Multi-phase learning for jazz improvisation and interaction". In *Proceedings of the Eight Biennial Symposium on Art and Technology*.

Friberg, A. (1995). *A quantitative rule system for musical performance*. (unpublished PhD dissertation). KTH, Stockholm.

Friberg, A.; Bresin, R.; Fryden, L.; Sunberg, J. (1998). "Musical punctuation on the microlevel: automatic identification and performance of small melodic units". *Journal of New Music Research*, 27 (3), pp. 271-292.

Friberg, A.; Sunberg, J.; Fryden, L. (2000). "Music From Motion: Sound Level Envelopes of Tones Expressing Human Locomotion". *Journal on New Music Research*, 29 (3), pp. 199-210.

Fry, C. (1984). "Flavors Band: A language for specifying musical style". Reprinted in S.M. Schwanauer and D.A. Levitt (eds.) (1993), *Machine Models of Music* (pp. 427-451). Cambridge, Mass.: The MIT Press.

Grachten, M.; Arcos, J. L.; Lopez de Mantaras, R. (2004). "TempoExpress, a CBR Approach to Musical Tempo Transformations". In P. Funk and P. A. Gonzalez Calero, (eds.), Proceedings of the 7th European Conference on Case-Based Reasoning), *Lecture Notes in Artificial Intelligence,* vol. 3155, pp. 601-615.

Gould, S. J. (1996). "Creating the Creators". *Discover Magazine*, October, pp. 42-54.

Hiller, L.; Isaacson, L. (1958). "Musical composition with a high-speed digital computer" (1958). Reprinted in S.M. Schwanauer and D.A. Levitt (eds.) (1993), *Machine Models of Music* (pp. 9-21). Cambridge, Mass.: The MIT Press.

Holmes, B. (1996). "Steve Thaler's 'Creativity Machine'". *New Scientist*, Vol. 2013, pp. 22-24.

Hörnel, D.; Degenhardt, P. (1997). "A neural organist improvising Baroque-style melodic variations". In *Proceedings of the 1997 International Computer Music Conference* (pp. 430-433). San Francisco: International Computer Music Association.

Hörnel, D.; Menzel, W. (1998). "Learning musical strucure and style with Neural Networks". *Journal on New Music Research*, 22 (4), pp. 44-62.

Johnson, M. L. (1992). "An expert system for the articulation of Bach fugue melodies". In D.L. Baggi (ed.), *Readings in Computer Generated Music* (pp. 41-51), Los Alamitos, Calif.: IEEE Press.

Johnson-Laird, P. N. (1991). "Jazz improvisation: A theory at the computational level". In P. Howell, R. West, and I. Cross, (ed.), *Representing Musical Structure*. London, UK and San Diego, Calif.: Academic Press.

Kendall, R. A., Carterette, E. C. (1990). "The communication of musical expression". *Music Perception*, 8 (2), pp. 129.

ARBOR Vol. 189-764, noviembre-diciembre 2013, a082. ISSN-L: 0210-1963 doi: http://dx.doi.org/10.3989/arbor.2013.764n6005

King, R. D.; Whelan, K. E.; Jones, F. M.; Reiser, P. G. K.; Bryant, C. H.; Muggleton, S. H.; Kell, D. B.; Oliver, S. G. (2004). "Functional genomic hypothesis generation and experimentation by a robot scientist". *Nature*, Vol. 427, pp. 247-252.

Langley, P.; Simon, H. A.; Bradshaw, G. L.; Zytkow, J. M. (1987). *Scientific Discovery*, *Computational Explorations of the Creative Mind*. Cambridge, Mass.: The MIT Press.

Lenat, D. B. (1983). "The role of heuristics in learning by discovery: Three case studies". In *Machine Learning: An Artificial Intelligence Approach*. Tioga.

Lerdahl, F.; Jackendoff, R. (1983). "An overview of hierarchical structure in music". *Music Perception,* vol. 1, pp. 229-252.

Levitt, D. A. (1993). "A Representation for Musical Dialects". In S.M. Schwanauer and D.A. Levitt (eds.), *Machine Models of Music* (pp. 455-469). Cambridge, Mass.: The MIT Press.

Lopez de Mantaras, R.; Arcos, J. L. (2002). "AI and Music: From Composition to Expressive Performance". *AI Magazine*, 23 (3), pp. 43-57.

Minsky, M. (1981). "Music, Mind, and Meaning". Reprinted in S.M. Schwanauer and D.A. Levitt (eds.) (1993), *Machine Models of Music* (pp. 327-354). Cambridge, Mass.: The MIT Press.

Mitchell, M.; Hofstadter, D. (1990). "The Emergence of Understanding in a Computer Model of Concepts and Analogy Making". *Physica D*, vol. 42, pp. 322-334.

Moorer, J. A. (1972). "Music and computer composition". Reprinted in S.M. Schwanauer and D.A. Levitt (eds.) (1993), *Machine Models of Music* (pp. 167-186). Cambridge, Mass.: The MIT Press.

Morales-Manzanares, R.; Morales, E. F.; Dannenberg, R.; Berger, J. (2001). "SICIB: An Interactive Music Composition System Using Body Movements". *Computer Music Journal*, 25 (2), pp. 25-36.

Muggleton, S. H. (2006). "Exceeding human limits". *Nature*, Vol. 440, pp. 409-410.

Pachet, F.; Roy, P. (1998). "Formulating constraint satisfaction problems on part-whole relations: The case of automatic harmonization". In *ECAI'98 Workshop on Constraint Techniques for Artistic Applications*. Brighton, UK.

Papadopoulos, G.; Wiggins, G. (1998). "A genetic algorithm for the generation of Jazz melodies". In *Proceedings of the SteP'98 Conference*, Finland.

Partridge, D.; Rowe, J. (1994). *Computers and Creativity*. Intellect Books.

Rader, G. M. (1974). "A method for composing simple traditional music by computer". Reprinted in S.M. Schwanauer and D.A. Levitt (eds.) (1993), *Machine Models of Music* (pp. 243-260). Cambridge, Mass.: The MIT Press.

Ritchie, G. D.; Hanna, F. K. (1984). "AM: A case study in AI methodology". *Artificial Intelligence*, vol. 23, pp. 249-268.

Rothgeb, J. (1969). "Simulating musical skills by digital computer". Reprinted in S.M. Schwanauer and D.A. Levitt (eds.) (1993), *Machine Models of Music* (pp. 157-164). Cambridge, Mass.: The MIT Press.

Rowe, J.; Partridge, D. (1993). "Creativity: A survey of AI approaches". *Artificial Intelligence Review*, vol. 7, pp. 43-70.

Sabater, J.; Arcos, J. L.; Lopez de Mantaras, R. (1998). "Using Rules to Support Case-Based Reasoning for Harmonizing Melodies". In *AAAI Spring Symposium on Multimodal Reasoning*, 147-151. Menlo Park, Calif.: American Association for Artificial Intelligence.

Schwanauer, S. M. (1993). A learning machine for tonal composition. In S.M. Schwanauer and D.A. Levitt (eds.), *Machine Models of Music* (pp. 511-532). Cambridge, Mass.: The MIT Press.

Schwanauer, S. M., Levitt, D. A. (eds.) (1993). *Machine Models of Music*. Cambridge, Mass.: The MIT Press.

Searle, J. (1980). "Minds, Brains and Programs". *Behavioral and Brain Sciences*, 3 (3) pp. 417-457.

Simon, H. A.; Sumner, R. K. (1968). "Patterns in Music". Reprinted in S.M. Schwanauer and D.A. Levitt (eds.) (1993), *Machine Models of Music* (pp. 83-110). Cambridge, Mass.: The MIT Press.

Suzuki, T.; Tokunaga, T.; Tanaka, H. (1999). "A Case-Based Approach to the Generation of Musical Expression". In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann pp. 642-648.

Thom, B. (2001). *BoB: An improvisational music companion*. (unpublished PhD dissertation). School of Computer Science, Carnegie-Mellon University.

Tobudic, A.; Widmer, G. (2003). "Playing Mozart Phrase by Phrase". In K.D. Ashley and D.G. Bridge (eds.), Proceedings of the 5th International Conference on Case-Based Reasoning. *Lecture Notes in Artificial Intelligence,* vol. 3155, pp. 552-566.

Tobudic, A.; Widmer, G. (2004). "Case-Based Relational Learning of Expressive Phrasing in Classical Music". In P. Funk and P. A. Gonzalez Calero (eds.), Proceedings of the 7th European Conference on Case-Based Reasoning. *Lecture Notes in Artificial Intelligence,* vol. 3155, pp. 419-433.

Turing, A. M. (1936). "On Computable Numbers, with an Application to the Entscheidungsproblem". *Proceedings of the London Mathematical Society*, vol. 42, pp. 230–265.

Turing, A. M. (1938). "On Computable Numbers, with an Application to the Entscheidungsproblem: A correction". *Proceedings of the London Mathematical Society* vol. 43, pp. 544–546.

Turing, A. M. (1950). "Computing Machinery and Intelligence". *Mind*, vol. LIX no. 236, pp. 433-460.

Valdés-Pérez, R. E. (1995). "Machine discovery in chemistry: New results". *Artificial Intelligence Journal*, vol. 74, pp. 191-201.

Wessel, D.; Wright, M.; Kahn, S. A. (1998). "Preparation for improvised performance in collaboration with a Khyal singer". In *Proceedings of the 1998 International Computer Music Conference*. San Francisco, Calif.: International Computer Music Association.

Widmer, G.; Flossmann, S.; Grachten, M. (2009). "YQX Plays Chopin". *AI Magazine*, 30 (3), pp. 35-48.

Woods, W. (1970). "Transition network grammars for natural language analysis". *Communications of the ACM*, 13 (10), pp. 591-606.