Creative DNA computing: splicing systems for music composition

Roberto De Prisco¹ · Rocco Zaccagnino¹

Accepted: 25 January 2022 / Published online: 12 February 2022
© The Author(s) 2022

Abstract

Splicing systems are a form of DNA computing as they mimic the recombination process among DNA molecules. This work discusses the use of splicing systems to build automatic tools for reproducing human beings' *creativity*, in the context of *automatic music composition*. More specifically, this work describes three general splicing system approaches for automatic music composition, and their application to two specific cases, namely composing 4-voice music and composing Jazz solos in a given style. Examples of music composed by the systems are presented.

Keywords DNA computing · Splicing systems · Automatic music composition · Artificial creativity

1 Introduction

Research on *unconventional models of computation* aims at defining paradigms and algorithms inspired by, or physically implemented in, chemical, biological and physical systems (Braund and Miranda 2014). As explained in Toffoli and Margolus (1991), an unconventional model is a “computing scheme that today is viewed as unconventional because its time hasn’t come yet-or is already gone”. Within the scientific community, there is the growing consensus that the increasingly frequent studies on unconventional models of computation are to be attributed to the fact that one day the limit of today’s conventional computing paradigms will be reached. The consequence is an ever-growing need for new models that can tackle complex problems using the ever-accelerating advances in technological devices.

An additional motivation for the interest about unconventional computational methods, comes from Artificial Intelligence (1995). The search for *mechanical intelligence*, i.e., the attempt to equip machines with intelligence, at some point had to consider a fundamental fact: artificial intelligence is different from biological intelligence. This distinction essentially depends on the deep difference between the learning process of machines and that of humans. Humans

have the ability to learn much more quickly from small sets of data and possess an innate ability to construct abstractions. Turing himself has felt the need to imitate the evolutionary self-learning and organizational modeling capabilities of living beings (Turing 1992). Therefore, it has become essential for AI researchers to focus on phenomena such as emotional and social intelligence (instinct, creativity, emotions, etc).

The problem of reproducing human beings' *creativity* has always been a challenging task for the AI community and specifically for the research on bio-inspired systems. There are several examples of human real-life situations in which creativity plays a fundamental role, such as strategic ability in games, intuition in mathematical calculations and proofs, improvisation ability in unexpected situations, and inspiration in the creation of artistic works. Among these, the *musical* ability, or *musical estrus*, is a challenging task.

“But, what musical ability means exactly”? In the context of this work, musical ability is the ability of “composing” new music. The focus is on systems capable of automatically producing music by means of a computer program, without *any* human intervention.

Bio-inspired computational methods have been largely used in several contexts, such as networks and cryptography (Ogiela and Ogiela 2015, 2017a,b). In this work, we focus on the challenge of reproducing the musical abilities of humans, by using bio-inspired computational methods. Among such methods there are Cellular Automata, e.g., (Adamatzky 2010; Miranda 1993, 1995; Xenakis 1992), Evolutionary Algorithms, e.g., (Biles 1994; Jeong et al. 2017; Liu and Ting 2015; Miranda and Biles 2007; Prisco

✉ Rocco Zaccagnino
rzaccagnino@unisa.it

Roberto De Prisco
robdep@unisa.it

¹ Computer Science Department, University of Salerno, Salerno, Italy

et al. 2020), and various DNA computing approaches, e.g., (Adleman 1994; Braund and Miranda 2013; Miranda 2014; Miranda et al. 2009).

An important role in DNA computing is that played by *Splicing Systems* (Head 1987a). Such a model tries to simulate the recombination process of DNA molecules, using two main operations: (i) *cut*, i.e., cuts two molecules in correspondence of specific patterns individuated by special proteins named *restriction enzymes*, (ii) *paste*, i.e., pastes together the fragments obtained at the previous steps on the basis of *splicing rules*, by exploiting *ligase enzymes*. However, although splicing systems theory is still a challenging field, few application results have actually been obtained so far.

Main contribution In this paper we survey existing splicing algorithms for music composition, providing a novel classification into three distinct general approaches. The goal is that of providing a general view of the use of splicing systems as a practical tool for unconventional music composition, and so, to provide evidence about the effectiveness of splicing systems as unconventional methods for music composition. The entire study is based on the algorithms presented in De Felice et al. (2015), De Felice et al. (2017), De Prisco et al. (2017) and Prisco et al. (2017), which, as far as we know, are the only ones based on splicing systems.

Organization In Sect. 2 we discuss some relevant related work in the bio-inspired musical computing field. In Sect. 3, needed basic notions are reviewed and, in particular, Sect. 3.2 provides details about the use of splicing systems as music composers. Then, subsequent sections, dig into details about the construction of such systems: Sects. 4, 5 and 6 describe three different approaches for the constructions of automatic music composers based on splicing systems. At the end of each section there is a sample music output. Finally, Sect. 7 contains concluding remark.

2 Bio-inspired musical computing

Bio-inspired systems have shown to be able to compose music, in unexpected and natural ways. Several approaches inspired by chemical, biological and physical systems have been proposed, including cellular automata, evolutionary methods, and DNA computing.

Cellular automata Cellular Automata are methods that can be used to model the evolution of a system over time (Adamatzky 2010). A cellular automaton is usually defined as a grid of cells. Every cell can assume a number of states, usually visualized with colors. The evolution process of a cellular automaton is performed by applying specific rules to

the cells informing them to change state according to state of their neighbourhood. The first music piece obtained as result of an evolution of orchestral clusters by using a cellular automaton, named *Horos*, was proposed by Xenakis (1992). Another example can be found in Miranda (1995), in which the authors used a reaction–diffusion computer to control a granular synthesizer, and the grid was divided into several sections, each one assigned to a sine-wave oscillator. The automaton was programmed to model the behaviour of a network of oscillating neurons. Such a system was used to generate sounds for electroacoustic music compositions, including “Olivine Tress” (Miranda 1993), composed in 1993, which is the first music piece composed on a parallel computer.

Evolutionary methods Recently, the interest for *evolving music* has considerably increased, due to the “evolutionary nature” of the compositional process, which, similarly to the standard evolutionary approach, goes through a generation of musical ideas and a selection of the most promising one for further iterated refining (Miranda and Biles 2007). The idea is that such a process could be seen, assuming the existence of a precisely defined metric, as an *optimization problem* that consists in placing a finite number of notes in a music score. To define such a metric usually harmonic and melodic rules are exploited. Obviously, providing clever ways in order to guide an algorithm toward a good solution can be difficult due to the nature of the problem, and so, the use of heuristics can improve the efficiency by restricting the exploration to smaller search spaces.

The use of evolutionary algorithms to define automatic composers has produced several works. For example, in Biles (1994) the authors proposed an automatic composer for Jazz solos. In Jeong et al. (2017) the authors present a multi-objective evolutionary approach to automatically produce more melodies at once, by exploiting music theory. In Liu and Ting (2015) the authors explore the composition styles by miming music patterns of a specific composer. The patterns are used as genes and the composition styles are used for the generation of new music.

As regarding the specific problem of composing 4-voice music in classical style, i.e., music for 4 instruments according to specific harmonic and melodic rules, very few evolutionary algorithms have been proposed. In Prisco et al. (2020) the authors propose *EvoComposer*, an algorithm able to solve the figured bass problem, that is, the input already contains the chords to be used and, thus, the algorithm has to find only the position of the voices for each chord in the input. Such an algorithm used tables of weights for chords and tonality change with arbitrary defined by using statistical information extracted from Bach’s chorales.

Table 1 Comparison of our work against some relevant works available in the literature, according to the *biological paradigm* used for composing music, and the music *target*, i.e., the specific musical genre in which the composed music belongs to, or the specific faced music problem

References	Bio paradigm	Music target
Xenakis (1992)	Cellular automata	Stochastic music
Miranda (1993)	Cellular automata	Stochastic music
Biles (1994)	Genetic algorithm	Jazz
Miranda (1995)	Cellular automata	Stochastic music
Miranda and Biles (2007)	Evolutionary methods	N/A ^o
Miranda et al. (2009)	Neuronal networks	Brain music
Acampora et al. (2011)	Fuzzy + metaheuristics	4-Voice chorales
Miranda (2014)	Biological computing	Physarum polycephalum music
Liu and Ting (2015)	Genetic algorithms	Pop
De Felice et al. (2015)	Splicing systems	4-Voice chorales
De Felice et al. (2017)	Splicing systems	4-Voice chorales
De Prisco et al. (2017)	Splicing systems + LSTM network	Jazz
Prisco et al. (2017)	Splicing systems + LSTM network	Jazz

◊ (Book collection with several explained methods and music targets)

DNA computing Since the first implementation of computational technology based on biological concepts (Adleman 1994), a huge interest in biological computation has developed in all disciplines. From a musical perspective, as mentioned above, biological computing has some very interesting possibilities.

The first hybrid wetware-silicon device in computer music was proposed in Miranda et al. (2009). In such a project the authors were interested in producing sound by using the spiking interactions between neurons. Brain cells were acquired from a seven-day-old Hen embryo and cultured in an vitro environment in order to form synapses. Once grown, the culture is placed on to a multi-electrode array in such a way that at least two electrodes (one is arbitrarily chosen as input and the other as output) make a connection into the neuronal network. Finally, the input is used to stimulate the network with electrical impulses while the output is used to record the subsequent spiking behaviour, which was experimented with sonification methods using additive and granular synthesis techniques to convey the neuronal network's behaviour. A problem with such types of approaches is that they are beyond the reach of the average computer musician.

An example of biological computing system openly accessible is the Physarum polycephalum, used in several works (Adamatzky 2010). As a consequence, many music projects using such type of biological system have been proposed, such as the Physarum polycephalum step sequencer (Miranda 2014) and a sonification work (Braund and Miranda 2013).

Table 1 summarizes some information about the main bio-inspired musical systems available in the literature (in chronological order). Specifically, we report details about the specific type of *biological paradigm* used for composing music, and the music *target*, i.e., the specific musical genre in

which the composed music belongs to, or the specific music problem considered.

3 Background

The following discussion is not intended to cover all the needed background knowledge; it is assumed that the reader is familiar with both subjects, that is, splicing systems and music theory, and thus that he/she is able to follow what is written in this section. If not, the reader is encouraged to acquire a minimal background about the basic notions, mentioned in Sects. 3.1 and 3.3. Section 3.2 talks about splicing systems used as automatic composers discussing the key points, formulated as *composition questions*, that the system needs to take into account.

3.1 Splicing systems

Splicing systems were introduced by Head (1987b), as an attempt to model biochemical splicing as an operation on strings. Subsequently, different (and more sophisticated) variants of splicing systems have been proposed, in particular by Păun (1996) and Pixton (1996). These alternative models are based on different operations which basically take as input two words and can generate either one new word (in this case we have a *1-splicing* operation), or 2 new words, (in this case we have a *2-splicing* operation). The splicing systems that we consider in this paper, use Păun's 2-splicing operation.

Păun's 2-splicing operation is based on a *splicing rule* r whose form is $r = v_1\#v_2\$v_3\#v_4$, where v_1, v_2, v_3, v_4 are words over an alphabet $\mathcal{A}$ such that $\#, \$ \notin \mathcal{A}$. The words

obtained by concatenating v_1 with v_2 and v_3 with v_4 , i.e., respectively, v_1v_2 and v_3v_4 are named *splicing sites* of r . Each of these sites represents a point of the words given in input, where it is possible to “cut” the string. Thus, a rule r is used to identify two points where the input strings are to be cut. Formally, let r be a splicing rule, and two words x and y such that $x = x_1v_1v_2x_2$ (x contains the first site v_1v_2) and $y = y_1v_3v_4y_2$ (y contains the second site v_3v_4). Then, r generates the words $z = x_1v_1v_4y_2$, $w = y_1v_3v_2x_2$. Such a splicing operation is denoted by $(x, y) \vdash_r (z, w)$.

Splicing systems generate languages based on the splicing operation. From an initial set of words (often called the initial language) the system applies the rules to produce new words which are added to the set of words. This process can generate an infinite language.

Formally, a *splicing system* is a triple $\mathcal{S} = (\mathcal{A}, \mathcal{I}, \mathcal{R})$, where $\mathcal{A}$ is a finite alphabet such that $\#, \$ \notin \mathcal{A}$, $\mathcal{I} \subseteq \mathcal{A}^*$ is the initial language and $\mathcal{R} \subseteq \mathcal{A}^*\#\mathcal{A}^*\$\mathcal{A}^*\#\mathcal{A}^*$ is the set of rules. A splicing system $\mathcal{S}$ is finite if $\mathcal{I}$ and $\mathcal{R}$ are both finite sets. Let $L \subseteq \mathcal{A}^*$ and $\sigma'(L) = \{w', w'' \in \mathcal{A}^* \mid (x, y) \vdash_r (w', w''), x, y \in L, r \in \mathcal{R}\}$. The splicing operation on languages is defined as follows:

$$\begin{aligned}\sigma^0(L) &= L, \\ \sigma^{i+1}(L) &= \sigma^i(L) \cup \sigma'(\sigma^i(L)), \quad i \geq 0, \\ \sigma^*(L) &= \bigcup_{i \geq 0} \sigma^i(L).\end{aligned}$$

Definition 1 (*Păun splicing language*) Given a splicing system $\mathcal{S} = (\mathcal{A}, \mathcal{I}, \mathcal{R})$, the language generated by $\mathcal{S}$ is $L(\mathcal{S}) = \sigma^*(\mathcal{I})$. A language L is $\mathcal{S}$ -generated if there exists a splicing system $\mathcal{S}$ such that $L = L(\mathcal{S})$.

3.2 Splicing systems and music composition

The basic idea in using a splicing system for music composition is that of treating music compositions as words and to view the music compositional process as the results of operations on words. In such a perspective, a splicing system becomes a tool for generating languages of *musical words*.

The definition of a music splicing system is closely related to the specific musical compositional process taken into consideration. Since these are systems that manipulate words to create new words, the first question concerns the choice of a *suitable representation* of the music in terms of words. As shown in De Felice et al. (2017), this choice affects the effectiveness of the systems in generating good quality musical compositions in an acceptable time.

Once the representation to be used has been defined, it is necessary to define *suitable splicing rules* in such a way that they produce new words suited for the musical context being considered. The definition of the rules is very important

because the rules determine the language being generated. Furthermore, from a theoretical point of view, the splicing system is an infinite word generative mechanism. In practical real-word applications, one needs to transform splicing into a finite process, i.e., a process that after a finite number of steps produces one or more qualitatively acceptable solutions. The idea is to transform the splicing process into an *evolutionary process* based on the use of an *evaluation function* (fitness) of the words, and a stop criterion of the generative process. Usually this criterion is chosen as a maximum number of iterations or a qualitative reference value that represents the desired quality of the music solutions produced.

To summarize, in the automatic process used by splicing systems to generate music, it is necessary to tackle the following three key points, that can be formulated as *composition questions*:

- cq1: “What is a suitable word-based representation in order to generate good quality musical compositions in acceptable time?”
- cq2: “How should splicing rules be defined in order to generate new words appropriate for the musical context considered?”
- cq3: “How the splicing process can be transformed into an evolutionary process using appropriate evaluation functions and stop criterion?”

The splicing system composers presented in this work can be classified in three categories: (i) static evaluation-based, (ii) statistical evaluation-based, and (iii) machine learning-based. Sections 4, 5 and 6 will discuss the 3 different approaches analyzing how they address the above key points. For each approach a concrete specific musical problem will be considered (actually for the first two approaches the music problem considered is the same).

3.3 Tempered music system and classical rules

Conventionally, western musicians use the tempered music system as reference model. In such a system, the *musical notes* are modelled by using a given frequency (usually 440 Hz), and the *musical octaves* consist in ranges of note frequencies between sounds obtained by doubling or halving a reference sound. The organization of the musical notes, finds a natural “view” in the structure of a piano keyboard: there are 88 keys, each key of the piano corresponds to a specific pitch (of a specific musical note) organized into *octaves*. Such octaves, roughly 7, contain the 88 keys ($12 \times 7 = 84$ keys in 7 octaves plus 4 extra keys); notes outside this range correspond to frequencies too low or too high to be pleasantly perceived by the human ear. Each octave is split in 12 equally spaced notes, which constitute the *chromatic scale*. The 12 notes of an octave are named using the letters C , $C\#$

or $D\flat$, D , $D\sharp$ or $E\flat$, E , F , $F\sharp$ or $G\flat$, G , $G\sharp$ or $A\flat$, A , $A\sharp$ or $B\flat$, B .

The musical notes are organized in *tonalities*, categorized as *major* or *minor*. Each tonality is built on a reference note, and so, we have one major tonality and one minor tonality, for a total of 12 major tonalities and 12 minor tonalities. Each tonality is associated with a group of notes that “sound good together”, called *scale* (of the tonality), organized as an ordered set of musical notes completely included in the range of an octave. As an example, the scale of the G major tonality is $G, A, B, C, D, E, F\sharp$, while the scale of the tonality F Major is $F, G, A, B\flat, C, D, E$. Given a scale, each contained note has a *degree*, which is given by the position inside the scale, usually denoted with the roman numerals $I, II, III, \dots, VII$. Thus in G major, A is the second degree (II), E is the sixth degree (VI), and so on. Given a tonality and the corresponding scale, the notes that do not belong to the scale are called *non-harmonic tones*, classified in: *auxiliary tones*, *passing tones*, *appoggiatura tones*, and *suspension tones*. We refer to Piston and DeVoto (1987) for further details. Usually, each music composition is characterized by a *main tonality*.

Several music genres, for example Jazz music, have in the *improvisation* (that is extemporary composition, consisting in inventing, on the spot, variations of a given melody) one of the most significant characteristics. Usually, the set of variations of a melody played by a musician during a performance, is called *solo*. The choices and abilities of the musician are strictly depending on several factors, such as, his musical experience, his preferences, the specific type of music, and so on. It is also interesting to notice that, during an improvisation, a musician usually tends to customize musical excerpts from previous performances (both his own and those of other musicians). The set of such music features used during solos, characterizes the specific style of a musician, and can be extracted to recognize such a style.

The common technique used for extracting the significant features of a music style is based on the analysis of the role that each music note assumes in the specific chord in which it is played. Given a chord c , the reference scale s used by the musician and a note n played on chord c , the degree of n in scale is denoted by $\text{degree}(n, c)$. Since there are 12 notes in an octave, and 7 notes in the scale, this means that 7 degrees belong to s and 5 notes are outside s . As an example, let $c = A\text{m}$ and $s = A, B, C, D, E, F\sharp, G$ (dorian mode). Let $n = B$ be the note, then $\text{degree} = \text{II}$. Let $n = B\flat$ be the note, then $\text{degree} = \text{bII}$.

One of the choices that most characterizes a musician's style is the specific scale used on a given chord. In general, a musician has several scales available to use on the same chord. In Jazz music, a performer usually substitutes a dominant chord with its tritone chord. As an example, on chord $c = C7$, traditional performers could play scale $s =$

$C, D, E, F, G, A, B\flat$, i.e., the mixolydian mode built on the V grade of the $F\text{maj}$ scale; modern musicians, instead, would prefer $s = G\flat, A\flat, B\flat, C\flat, D\flat, E\flat, F\flat$, that is the mixolydian mode built on the V grade of the $C\text{bmaj}$ scale. In this case, we say that chord $c = C7$ has been substituted by the chord $c = G\flat7$. This method, in Jazz music, is called *substitution*. For further details, we refer the reader to Levine (2009).

3.4 Splicing models for music composition

The development of a splicing system for music composition requires some crucial choices mainly depending on the specific chosen musical context.

First, a suitable *music representation*. As explained above, splicing systems are generative mechanisms of languages (of words). Hence, the music composed by the system has to be represented as “words” on a finite alphabet of symbols. The complexity of the chosen representation depends on which musical information we decide to embed in the words (for example, the notes, the tonality, the degree of the chord). As we will see, the complexity of the words also conditions the application of the rules of the splicing system: in general, as the information in the words increases, the representation becomes more complex but the set of words on which the rules are applied becomes smaller.

Second, we need to choose a set of effective *music rules*. The generation of words in a splicing system is carried out by applying the splicing rules on the current set of words (starting from an initial set). The compositional process is then emulated in terms of splicing operations (cut and paste). Consequently, it is necessary to appropriately define these rules to achieve the chosen musical goals. This task is not easy. When the specific musical genre allows it, it is possible to define precisely the set of rules. For example, in 4-voice music compositions, the melodic and harmonic rules are well defined, and this can lead to a fairly immediate modeling of the splicing rules. The problem becomes more complex when the specific musical genre strongly depends on the musical or stylistic preferences of the individual musicians, such as in Jazz music. In these cases, a possible approach could be to support the splicing system with a mechanism for evaluating the result of the rules, which is directly extracted from a set of initial examples. In Sect. 4, for example, the stylistic characteristics of the chosen composer have been extracted through a static analysis carried out on a corpus of works written by the composer. In Sect. 6, instead, the stylistic characteristics of the chosen music performer have been learned by a machine learning-based model (LSTM network), trained on a set of “solos” executed by the performer.

Finally, an *evaluation function*. As we have seen, from a theoretical point of view, the language generated by a splicing system is infinite. However, in a purely applicative context

such as that of musical composition, it is necessary to find a mechanism to extract from this language only a finite number of words that represent the best compositions obtained with respect to the chosen musical target. To this end, the idea is to transform the splicing process into an evolutionary process typical of well-known meta-heuristics, such as genetic algorithms and swarm optimization techniques. Thus, it is necessary to define an evaluation function that allows us to keep only the best solutions, evaluated from a musical point of view, at each iteration of the splicing process. It is important to underline that the choice of this musical point of view depends, once again, on the specific musical target chosen. For example, in Sects. 4 and 5, the evaluation function assesses the harmonic and melodic goodness of the solutions, since the chosen music target is the 4-voice music composition. In Sect. 6, instead, the chosen music target is the Jazz music, and specifically the problem of reproduce the style of a music performer. Thus, the evaluation function has been defined to evaluate the similarity between the solutions produced by the system and the stylistic characteristics of the performer.

Concerning the evaluation function, existing music splicing systems can be classified as follows.

1. Music splicing systems which use an evaluation function that measures the quality of the composition whose definition is based on “weights”, chosen by empirical observations, so that good patterns have heavier weights.
2. Music splicing systems which use an *hybrid* evaluation function that, on one hand, adheres to specific music rules, and on the other hand, extracts statistical information from a corpus of existing music compositions, with the goal to assimilate a specific composer’s style or genre’s style.
3. Music splicing systems which use an evaluation function that measures the quality of the composition by using a method (usually a machine learning-based prediction method such as LSTM) to predict patterns coherent with specific style and used to guide the splicing system during the composition.

In the following sections, for each of the scenarios described above, we will illustrate practical real-world music applications, by providing details about the design choices made. We also include some original music pieces produced by these systems.

4 Approach 1: Splicing systems based on static weights

In this section the first approach is presented: music splicing systems that use an evaluation function, for measuring

the quality of the composition, whose definition is based on “weights” chosen by empirical observations, so that good patterns have heavier weights. The specific music problem considered is that of polyphonic k -voice compositions. The specific case of $k = 4$ (chorales) has been considered in De Felice et al. (2015), De Felice et al. (2017), Prisco et al. (2017) and Acampora et al. (2011). Here the description is generalized to any k . First the music problem is formally described. Then the music splicing composer for polyphonic compositions is presented. Details regarding how the splicing composer addresses each one of the composition questions cq1, cq2, and cq3 (see Sect. 3.2) are provided. At the end of the section an example of music output is given.

4.1 The music composition problem: k -voices music

A k -voices music piece is composed of k voices (instruments). Each voice can play notes in an admissible range. Music is organized in a sequence of *measures*, each one divided in *beats*. In each beat k notes can be simultaneously played (one voice plays/sings one note). The k notes played in a specific beat form a *chord*. Chords are built on the degrees of the scale of the tonality used. The degrees of a scale are indicated with: *I, II, III, IV, V, VI, VII*. Usually, capital letters indicate major chords while small letters indicate minor chords (Piston and DeVoto 1987).

A k -voices music piece can be analyzed from two main points of view: (i) *vertical*, that is the *harmonic* structure of the music piece, represented by the sequence of chords of the piece, and (ii) *horizontal*, that is the *melodic* lines of the music piece, represented by the sequence of notes (melodies) played by each voice. Obviously, *music rules* can regard both the harmonic and melodic aspects. Harmonic rules are defined by using specific chord sequences, called *cadences*, having special musical functions. The most common used cadences in classical music are: $II \rightarrow V$, $V \rightarrow I$, $VI \rightarrow II$, $IV \rightarrow IV$, $IV \rightarrow I$, $V \rightarrow VI$, and $III \rightarrow VI$. As we will see, each cadence will be “encoded” through a set of splicing rules. Also for the melodic lines, there are strict rules. These rules concern both the movement of a single line (jump) and the relationship between the movements of two different lines, and they are based on *intervals*, i.e., the distance between two notes of two different melodic lines. According to music theory rules, for any given pair of lines, some specific patterns should be avoided (see Piston and DeVoto 1987 for details): (1) two lines that move by creating two consecutive unisons; (2) two lines that create two consecutive octaves or fifths; (3) two melodic lines that intersect. Splicing rules can model such musical rules.

Fig. 1 A fragment of BWV 6.6

4.2 Approach details

As explained in Sect. 3, a music splicing composition system is made up of 3 components: an alphabet, an initial set of words and a set of rules. The system $\mathcal{S}_1$ considered in this section will be specified by describing these three components, denoted with $\mathcal{S}_1 = (\mathcal{A}_1, \mathcal{I}_1, \mathcal{R}_1)$, with respect to the three composition questions discussed earlier.

cq1: word-based representation The first step towards the definition of a word-based representation is the choice of “how to represent a composition using a word”. Let us consider a k -voice composition $C = (c_1, \dots, c_n)$ where each c_i is a chord, with $i = 1, \dots, n$. Obviously there are several ways to represent C by using a word. The representation that we use, consists in encoding each chord c_i with a specific word w_i and then representing C with the word $\mathcal{W}(C) = w_1 \dots w_n$.

At this point, the problem comes down to deciding “how to define each word w_i ”. Such a decision depends on how much information regarding c_i one wants to put into w_i . As explained in Sect. 4.1, for the k -voice music problem, a chord is the set of the k notes played in a specific beat. Thus, in w_i one has to insert information regarding the k notes in c_i . The complexity of w_i depends on how much information regarding each note needs to be considered. One very simple solution proposed in this approach consists in representing each note using information regarding 3 basic parameters: (i) the voice that plays the note, (ii) the name of the note, and (iii) the octave in which it is located. More precisely, a chord c_i will be represented as $w_i = v_1 x_1 o_1 \dots v_k x_k o_k$, where, for each $j = 1, \dots, k$:

- v_j is the voice which plays the j th note in c_i ,
- x_j is the name of the j th note in c_i ,

- o_j is the octave in which the j th note in c_i is placed.

Clearly, it is necessary to define the *voice alphabet* used to indicate the voices, the *note alphabet* used to indicate the name of the notes, and the *octave alphabet* used to indicate the octaves. These alphabets are as follows: (i) the voice alphabet $\mathcal{A}_V = \{v_1, \dots, v_k\}$, where v_i indicates the i -th voice, (ii) the note alphabet $\mathcal{A}_N = \{C, C\#, Db, D, D\#, Eb, E, F, F\#, Gb, G, G\#, Ab, A, A\#, Bb, B\}$, and (iii) the octave alphabet $\mathcal{A}_O = \{1, \dots, m\}$.

Finally, let $\mathcal{A}_1 = \mathcal{A}_V \cup \mathcal{A}_N \cup \mathcal{A}_O$. Using $\mathcal{A}_1$ it is possible to represent k -voice music compositions as words.

Example 1 Consider the 4-voice music fragment C in Fig. 1 (a fragment of Chorale BWV 6.6), $C = (c_1, c_2, c_3, c_4)$. Each chord c_i is represented by a word, w_i , with $w_1 = “v_1 A4 v_2 C5 v_3 F5 v_4 C6”$, $w_2 = “v_1 F4 v_2 F5 v_3 A5 v_4 C6”$, $w_3 = “v_1 Bb4 v_2 F5 v_3 Bb5 v_4 D6”$, $w_4 = “v_1 G4 v_2 G5 v_3 D5 v_4 Bb5”$. The entire music fragment is represented by the word $w = w_1 w_2 w_3 w_4 = “v_1 A4 v_2 C5 v_3 F5 v_4 C6, v_1 F4 v_2 F5 v_3 A5 v_4 C6, v_1 Bb4 v_2 F5 v_3 Bb5 v_4 D6, v_1 G4 v_2 G5 v_3 D5 v_4 Bb5”$. Passing notes are notes that do not fall on a beat and whose length is smaller than a beat; such notes are ignored. In Fig. 1 there are two passing notes for v_1 (the final F and the previous A) and they do not appear in w .

cq2: rules definition The rules use the word-based definition above described to generate words representing k -voices music composition. Notice that, in this approach, only harmonic theory rules are considered, and thus the splicing rules are defined according to such music rules, as we explain in the following.

Consider first the initial set of words $\mathcal{I}_1$ on which the rules will be initially applied. This presupposes the choice of a corpus of pre-composed k -voices music pieces, which is representative of the specific musical style (and genre) on which one wants to set the problem. As an example, in De Felice et al. (2015); De Felice et al. (2017) the initial corpus of music is a set of J. S. Bach’s chorales.

Once the corpus of pre-composed k -voices music pieces has been chosen, each one of the pieces needs to be transposed in every tonality, and each of such transposed piece is inserted into the *ground data set*, named $\mathcal{G}$. Obviously, if t is the cardinality of the chosen corpus, then $|\mathcal{G}| = 12 * t$ pieces. Let $\mathcal{I}_1$ be the set containing the $12 * t$ words (obtained by applying the word-based representation above defined) that represents the pieces in $\mathcal{G}$. Notice that these are not the only words in $\mathcal{I}_1$: there are other words that will be explained shortly, when the rules will be presented.

Since the harmonic rules are defined on the sequences of chords, also single chords, from the pieces in $\mathcal{G}$, are added to $\mathcal{I}_1$, so that the produced music will be similar to that in the ground data set. During the chord extraction, information about its original function, that is, the degree of the scale

on which the chord is built, is attached to the chord. This information is crucial in re-arranging the chords, by means of splicing rules, so that specific sequences of chords will be produced. As a side note, notice that Approach 2 (described in Sect. 5) will use an enhanced representation to embed such information directly in the words.

As done for the initial set, each extracted chord is transposed in all 12 tonalities. Let $Chords(\mathcal{G})$ be the set of these chords. For each extracted chord $c \in Chords(\mathcal{G})$, we keep information, provided by the harmonic analysis, about the degree on which the chord is built. The set of words associated to $Chords(\mathcal{G})$ is $\mathcal{W}(Chords(\mathcal{G}))$.

Consider now the definition of the splicing rules $\mathcal{R}_1$. As explained before, the set of splicing rules are modeled on classical harmonic rules. In particular, this approach considers a set of *cadences*, which are specific sequence of chords. In particular the cadences considered are $V \rightarrow I$, $I \rightarrow IV$, $II \rightarrow V \rightarrow I$, $IV \rightarrow V \rightarrow I$, $I \rightarrow II \rightarrow III$, $III \rightarrow II \rightarrow I$, and $V \rightarrow VI \rightarrow VII \rightarrow I$.

Additionally, it is customary to impose that a composition starts with a chord built on a specific degree of the scale d_s and ends with a chord built on a specific degree of the scale d_e , since this is what happens normally (usually $d_s = d_e = I$). Notice that for each of these situations (each cadence, and the starting and ending of the composition) the splicing system has splicing rules. As a result, the splicing rules can be organized in three sets:

1. **Starting with d_s :** For each $c_i, c_j \in Chords(\mathcal{G})$, such that $Degree(c_i) = d_s$, $\mathcal{R}_1$ contains the rule $r = w_i \# \epsilon \# w_j$ where ϵ is the empty word, w_i is the word associated to c_i and w_j is the word associated to c_j . Moreover also w_i is added to $\mathcal{I}_1$.
2. **Cadences:** For each quadruple of chords $c_i, c_j, c_s, c_t \in Chords(\mathcal{G})$, such that $Degree(c_1) \rightarrow Degree(c_4) \in Cadences$ and $Degree(c_3) \rightarrow Degree(c_2) \in Cadences$, $\mathcal{R}_1$ contains the rule $r = w_1 \# w_2 \# w_3 \# w_4$ where w_i is the word associated to c_i , w_j is the word associated to c_j , w_s is the word associated to c_s , and w_t is the word associated to c_t . Words w_i, w_j, w_s , and w_t are also inserted into $\mathcal{I}_1$.
3. **Ending with d_e :** For each $c_i, c_j \in Chords(\mathcal{G})$, such that $Degree(c_j) = d_e$, $\mathcal{R}_1$ contains the rule $r = w_i \# \epsilon \# w_j$ where w_i is the word associated to c_i and w_j is the word associated to c_j . Word w_j is also inserted into $\mathcal{I}_1$.

cq3: evolutionary process Approach 1 has been described by providing the 3 components of the system $\mathcal{S}_1 = (\mathcal{A}_1, \mathcal{I}_1, \mathcal{R}_1)$. What remains to describe is how the system produces the output composition. To this end, the splicing process is transformed into an evolutionary process, and using the system $\mathcal{S}_1$, through such a process, the language $L = L(\mathcal{S}_1)$ is generated.

- **Evolution:** Each iteration of the evolution corresponds to one application of all the rules in $\mathcal{I}_1$ to all possible pairs of words in the current language. So the language $L(\mathcal{S}_1)$ evolves by acquiring new words at each iteration.
- **Stop criterion:** Several possibilities can be considered as stop criterion, such as, fixing a maximum number of iterations, or fixing a *quality threshold* that the solutions should reach in order to be able to say that a good quality has been achieved. In De Felice et al. (2015); De Felice et al. (2017), a fixed maximum number of iterations has been considered, because the choice of a threshold that expresses the concept of good quality in music is often too tied to subjective evaluations.

So, let max be a fixed maximum number of iterations. Let $L_{max}(\mathcal{S}_1)$ be the language obtained after max iterations. Once the language $L_{max}(\mathcal{S}_1)$ has been generated, choose one single word $w \in L_{max}(\mathcal{S}_1)$ as the output of the algorithmic composer. The output of the composer is the composition represented by such a word; the choice of the word is made exploiting an evaluation function.

- **Evaluation function:** a function f_h measures the harmonic quality of the compositions. The function is defined by considering well-known and widely accepted rules in music theory. Specifically, f_h assigns “weights” to pairs of consecutive chords. As specified at the beginning of the section, this approach uses static weights following common conventions in music, as proposed in De Felice et al. (2015).

Thus, given a composition (word) $C = (c_1, \dots, c_n)$, the harmonic value $h(C)$ of C is obtained by giving a weight for $f_h(c_i, c_{i+1})$, with $i = 1, 2, \dots, n - 1$. The value of $f_h(C)$ is given by the sum of the weights for each pair (c_i, c_{i+1}) for $i = 1, 2, \dots, n - 1$. A composition with maximum f_h is given in output.

4.3 Implementation

This section describes the result of an execution of the music splicing system built with Approach 1. The system has been implemented in Python by using the library music21.¹ All experiments have been conducted with a 2,8 GHz Intel Core i7 quad-core machine equipped with 16 GB 2133 MHz LPDDR3 RAM. The time required to complete all the experiments was approximately 3h 20m.

Evolution experiments Since J.S. Bach’s 4-voice chorales are widely available, it is convenient to focus on the case of $k = 4$. The set $\mathcal{I}_1$, is defined considering the *ground set* $\mathcal{G}$ containing 10 J. S. Bach’s chorales: BWV 2.6, BWV 10.7, BWV 11.6, BWV 14.5, BWV 16.6, BWV 20.7, BWV 28.6, BWV 32.6, BWV 40.8 and BWV 44.7.

¹ <http://web.mit.edu/music21/>.

Since the evolutionary process grows the language without bounds, any implementation risks of running out of memory quickly. For this reason it is good to keep bounded the size of the language by setting a maximum threshold $p_{\max}$ for the words in the language and exploiting the evaluation function to keep the best words in the language. For this reason, at the end of an iteration, if the cardinality of the generated language is greater than $p_{\max}$, only the $p_{\max}$ solutions that have the highest harmonic value are kept in the set.

Several experiments have been executed, using a different number t of iterations and several values for the maximum size $p_{\max}$ of the generated language. More precisely, the values for t are the ones in the set $T = \{50, 100, 250, 500, 750, 1000, 5000, 7500, 10000\}$, and values for $p_{\max}$ are the ones in the set $P = \{50, 100, 250, 500, 750, 1000\}$.

For each pair $(t, p_{\max})$, 5 executions of $\mathcal{S}_1$ have been run. For each experiment the *average* harmonic value of the 5 executions has been computed. The best result has been obtained with $t = 5000$ and $p_{\max} = 1000$. Observe that the process is deterministic. Obviously, by varying t and $p_{\max}$, one obtains different solutions. A degree of non-determinism is present if, at the end of an iteration, there are many words with the same harmonic value and, in order to limit the size of the language to $p_{\max}$, some of them have to be kept and other discarded. Thus, also the choice of which ones are kept in case of ties can influence the results obtained. The implementation described here makes random choices in such cases.

Figure 2 shows the music score of the best solution obtained. Notice that when a k -voice-like composition is generated, it only represents a sequence of chords organized in tonalities areas. In order to make it a musical composition by inserting rhythmic variations, it is needed to set some rhythmic parameters. Specifically, the *meter* of the composition and the duration of each note. As done in De Felice et al. (2015), De Felice et al. (2017) the implementation used here (i) *sets the meter value* by choosing randomly one value in a set of typical meter values, *sets the duration of the notes*, by adding, with a uniform distribution of probabilities, non-harmonic tones after the notes, ensuring no alteration in the total original duration. Figure 2 shows the music obtained.

5 Approach 2: Splicing systems based on statistical weights

The approach described in this section is an enhancement of the one presented in the previous section. With respect to Approach 1, Approach 2 uses an *hybrid* evaluation function that, on one hand, adheres to specific music rules, and on the other hand, extracts statistical information from a corpus of existing music compositions, with the goal of assimilating a specific composer's style or genre's style. The approach is

applied to the same music composition problem that has been considered in Sect. 4, that is the k -voice music composition problem.

5.1 The music composition problem: k -voices music

The problem is the same as the one considered in Sect. 4, so the reader is referred to that section for details about the problem.

5.2 Approach details

As for the previous approach the music splicing system $\mathcal{S}_2$ is described by instantiating the three components $\mathcal{S}_2 = (\mathcal{A}_2, \mathcal{I}_2, \mathcal{R}_2)$, that is the alphabet, the initial set of words and the rules, and explaining how this specific approach tackles the 3 composition questions.

cq1: word-based representation The word-based representation used in this approach is an extension of the word-based representation described in Sect. 4. The main difference lies in the additional information considered to represent each note. Specifically, for each note the information considered is the one used in Sect. 4 (i.e., the *voice* that plays such a note, the *name* of such a note, and the *octave* in which it is located) and, on top of that, also: (i) the *tonality* of the chord in which the note is played, (ii) the *quality* of the chord in which the note is played, and (iii) the *degree* of the chord in which the note is played. More precisely, a chord c_i is represented as $w_i = t_1 q_1 d_1 v_1 x_1 o_1 t_1 q_1 d_1 \dots v_k x_k o_k t_k q_k d_k v_k x_k o_k$, where, for each $j = 1, \dots, k$:

- v_j is the voice which plays the j th note in c_i ,
- x_j is the name of the j th note in c_i ,
- o_j is the octave in which the j th note in c_i is placed.
- t_j is the tonality of c_i ,
- q_j is the quality of c_i ,
- d_j is the degree of c_i .

Observe that, given the j -th note in the chord c_i , the information regarding tonality (t_j), quality (q_j) and degree (d_j), is concatenated twice, before and after the information regarding voice (v_j), name (x_j) and octave (o_j), resulting in the word $t_j q_j d_j v_j x_j o_j t_j q_j d_j$. As shown in De Felice et al. (2017), this choice is crucial for the definition of the rules.

For the alphabet, let $\mathcal{A}_V$ be the voice alphabet, $\mathcal{A}_N$ be the note alphabet, and $\mathcal{A}_O$ be the octave alphabet, defined in Sect. 4. Then, let the *tonality alphabet* $\mathcal{A}_T$ be $\{C, C\#, D\flat, D, D\#, E\flat, E, F, F\#, G\flat, G, G\#, A\flat, A, A\#, B\flat, B\}$, the *quality alphabet* $\mathcal{A}_Q$ be $\{M, m\}$, where M stands for major tonality and m for minor tonality, and the degree alphabet $\mathcal{A}_D$ be $\{1, 2, 3, 4, 5, 6, 7\}$.

Fig. 2 Approach 1: The composition given in output by S_1

APPROACH 1

$\text{♩} = 100$

The musical score is presented in four systems, each containing four staves labeled VOICE 1, VOICE 2, VOICE 3, and VOICE 4. The key signature is two flats (Bb, Eb) and the time signature is 4/4. The tempo is indicated as $\text{♩} = 100$.

System 1: VOICE 4 starts with a half note G4, followed by quarter notes A4, Bb4, A4, G4, F4, E4, D4. VOICE 3 starts with a half note G4, followed by quarter notes A4, Bb4, A4, G4, F4, E4, D4. VOICE 2 starts with a half note G3, followed by quarter notes A3, Bb3, A3, G3, F3, E3, D3. VOICE 1 starts with a half note G2, followed by quarter notes A2, Bb2, A2, G2, F2, E2, D2.

System 2: VOICE 4 continues with quarter notes C4, Bb3, A3, G3, F3, E3, D3. VOICE 3 continues with quarter notes C4, Bb3, A3, G3, F3, E3, D3. VOICE 2 continues with quarter notes C4, Bb3, A3, G3, F3, E3, D3. VOICE 1 continues with quarter notes C4, Bb3, A3, G3, F3, E3, D3.

System 3: VOICE 4 continues with quarter notes C4, Bb3, A3, G3, F3, E3, D3. VOICE 3 continues with quarter notes C4, Bb3, A3, G3, F3, E3, D3. VOICE 2 continues with quarter notes C4, Bb3, A3, G3, F3, E3, D3. VOICE 1 continues with quarter notes C4, Bb3, A3, G3, F3, E3, D3.

System 4: VOICE 4 continues with quarter notes C4, Bb3, A3, G3, F3, E3, D3. VOICE 3 continues with quarter notes C4, Bb3, A3, G3, F3, E3, D3. VOICE 2 continues with quarter notes C4, Bb3, A3, G3, F3, E3, D3. VOICE 1 continues with quarter notes C4, Bb3, A3, G3, F3, E3, D3.

Finally, let the alphabet $\mathcal{A}_2$ be $\mathcal{A}_2 = \mathcal{A}_V \cup \mathcal{A}_N \cup \mathcal{A}_O \cup \mathcal{A}_T \cup \mathcal{A}_D \cup \mathcal{A}_Q$.

Example 2 Consider again the 4-voice music fragment C in Fig. 1, $C = (c_1, c_2, c_3, c_4)$. Each chord c_i is represented by a word w_i . Specifically: $w_1 = \text{“BbM5 } v_1 A4 v_2 C5 v_3 F5 v_4 C6 \text{ BbM5”}$, $w_2 = \text{“BbM5 } v_1 F4 v_2 F5 v_3 A5 v_4 C6 \text{ BbM5”}$, $w_3 = \text{“BbM1 } v_1 Bb4 v_2 F5 v_3 Bb5 v_4 D6 \text{ BbM1”}$, $w_4 = \text{“BbM6 } v_1 G4 v_2 G5 v_3 D5 v_4 Bb5 \text{ BbM6”}$, so $w = \text{BbM5 } v_1 A4 v_2 C5 v_3 F5 v_4 C6 \text{ BbM5 BbM5 } v_1 F4 v_2 F5 v_3 A5 v_4 C6 \text{ BbM5 BbM1 } v_1 Bb4 v_2 F5 v_3 Bb5 v_4 D6 \text{ BbM1 BbM6 } v_1 G4 v_2 G5 v_3 D5 v_4 Bb5 \text{ BbM6”}$.

Notice that boldface text is used only to emphasize the novel music information embedded in the words (tonality, degree and quality), with respect to that defined in Sect. 4.

cq2: rules definition As regarding the rules, first, we have built the *ground data set* $\mathcal{G}$ considering both all the chosen pieces and their transpositions in all the 12 tonalities, for a total of $|\mathcal{G}| = 12 * t$ pieces, where with t we indicate the number of chosen pieces; then, we have defined the set $\mathcal{I}_2$ as the set including the word-based representation of the $12 * t$ pieces contained in $\mathcal{G}$, and in addition, other words explained in the following. In this approach, to avoid the construction of all the combinations of the extracted chords, we decided to integrate the degree and the tonality of a chord directly in its word representation. In such a way, the definition of each rule is only based on extracted chords having specific tonalities and degrees. This approach presents several advantages. First, we are considering a significant smaller number of rules. Secondly, rules about the modulations can be directly extracted using the word representation of chords. Finally, the complexity of the music splicing system proposed in this approach, in terms of time and space, is linear in $|\text{Chords}(\mathcal{G})|$, i.e., $\mathcal{O}(|\text{Chords}(\mathcal{G})|)$, where let $\mathcal{G}$ is the initial set of chorales, and $\text{Chords}(\mathcal{G})$ is the set of chords extracted from $\mathcal{G}$.

The rules in $\mathcal{R}_2$ are partitioned in four sets:

1. **Group 1 (forcing d_s to start).** For $c_i, c_j \in \text{Chords}(\mathcal{G})$, satisfying $\text{Degree}(c_i) = d_s$ and $\text{Tonality}(c_i) = \text{Tonality}(c_j)$, $\mathcal{R}_2$ contains the rule $r = w_i | \epsilon \$ \epsilon | w_j$ where, w_i and w_j are the word associated to c_i and c_j , respectively. The word w_i is also added to $\mathcal{I}_2$.
2. **Group 2 (forcing cadences).** For each quadruple of chords $c_i, c_j, c_s, c_t \in \text{Chords}(\mathcal{G})$, if $\text{Degree}(c_i) \rightarrow \text{Degree}(c_t)$, with $\text{Tonality}(c_i) = \text{Tonality}(c_t)$ is in *Cadences*, and $\text{Degree}(c_s) \rightarrow \text{Degree}(c_j)$, $\text{Tonality}(c_s) = \text{Tonality}(c_j)$, is also in *Cadences*, $\mathcal{R}_2$ contains the rule $r = w_i | w_j \$ w_s | w_t$, where w_i is the word associated to c_i , w_j is the word associated to c_j , w_s is the word associated to c_s , and w_t is the word associated to c_t . The words w_i, w_j, w_s , and w_t are also added to $\mathcal{I}_2$.
3. **Group 3 (forcing d_e as ending).** For $c_i, c_j \in \text{Chords}(\mathcal{G})$, satisfying $\text{Degree}(c_j) = d_e$ and $\text{Tonality}(c_i) = \text{Tonality}(c_j)$,

$\mathcal{R}_2$ contains the rule $r = w_i | \epsilon \$ \epsilon | w_j$ where w_i and w_j are the words associated to c_i and c_j , respectively. The word w_j is also inserted into $\mathcal{I}_2$.

4. **Group 4 (forcing modulations).** Let $c_i, c_{i+1} \in \text{Chords}(\mathcal{G})$ be consecutive chords such that $\text{Tonality}(c_i) \neq \text{Tonality}(c_{i+1})$. Let $c_j, c_{j+1} \in \text{Chords}(\mathcal{G})$ be two other consecutive chords, such that $\text{Tonality}(c_j) \neq \text{Tonality}(c_{j+1})$. Then, $\mathcal{R}_2$ contains the rule $r = w_i | w_{i+1} \$ w_j | w_{j+1}$ where w_k is the word associated with c_k , for $k = i, i+1, j, j+1$. The words w_i, w_{i+1}, w_j , and w_{j+1} are also added to $\mathcal{I}_2$.

cq3: evolutionary process Having defined the system $S_2 = (\mathcal{A}_2, \mathcal{I}_2, \mathcal{R}_2)$ what remains to do is to describe the evolutionary process that generates the language $L = L(S_2)$. As for the system described in the previous section, there is an evolution process, a stop criterion and an evaluation function. Actually the first two are the same as the ones already seen in Sect. 4. The evaluation function is different.

- **Evolution process:** As before, in each iteration, the rules $\mathcal{R}_2$ are applied to the current language.
- **Stop criterion:** As before, there is a fixed maximum number of iterations max and the evolution processes repeated for such a number of iterations to obtain $L_{max}(S_2)$. From this language, one single word is chosen as the output composition.
- **Evaluation function:** The evaluation function is used to select the single word from $L_{max}(S_2)$. This approach is different because it defines an evaluation function that, (i) on one hand adheres to rules from classical music, and (ii) on the other hand exploits *statistical information* from a corpus of existing music, which is, somehow, representative of the specific musical style (and genre) that is being considering. For example, in De Felice et al. (2017), statistical information is extracted from a set of Bach's chorales. The result is a *multi-objective evaluation function* composed of two sub-functions: an *harmonic function* f_h and a *melodic function* f_m . Both functions have the following form:

$$f = \sum_i a_i w_i$$

where w_i are *weights* and a_i are *coefficients*.

Intuitively, the weights w_i are used to express the objective part of the evaluation and represent well known rules from the theory of harmony. The coefficients, instead, are used to express a subjective component, and are obtained with a statistical analysis of the chosen corpus of existing music (capture the style of the music pieces in the corpus chosen). The interested reader can find further details about the weights and the coefficients in De Felice et al. (2017).

Harmonic function Similarly to what was done in in Sect. 4, for the unique evaluation function f , given a k -voice composition $C = (c_1, \dots, c_n)$, the *harmonic value* $f_h(C)$ is calculated by considering all pairs of consecutive chords c_i, c_{i+1} . The objective is to maximize $f_h(C)$. However, in this approach the f_h definition is more complex with respect to that proposed in Sect. 4: the weights are not statically fixed, but are the result of transforming musical conventions into a probability distribution. Furthermore, they are multiplied by coefficients obtained with a statistical analysis of the chosen corpus.

Regarding the *coefficients*, they have been obtained by performing a statistical analysis over the chosen corpus. Specifically, by looking for adjacent chords it is possible to figure out the percentage of passages from one chord to the subsequent one. As an example, see Table 6 and Table 7 in De Felice et al. (2017), which show the result obtained by performing a statistical analysis on a subset of the J.S. Bach's chorales corpus.

For what concerns the *weights*, instead, usually in classical harmony the *frequency distribution* of usage for the cadences, is organized in classes “often”, “sometimes”, and “seldom” (see and De Felice et al. 2017—Tables 2 and 3, for further details). As an example, in classical music, the degree I (first major degree):

- “often” goes to I, IV and V ,
- “sometimes” goes to vi ,
- “seldom” goes to ii and iii .

Suppose to have a corresponding *probability distribution*, i.e., $(X_{\text{often}}, X_{\text{sometimes}}, X_{\text{seldom}})$. Then, given a specific chord c_i , the weight for the next chord will be a function of the preceding one, according to such a distribution. More precisely, it is possible to use the probability distribution $(X_{\text{often}}, X_{\text{sometimes}}, X_{\text{seldom}})$ to assign the weight as follows: c_{i+1} will be one of the chords in the “often” class, X_{often} percent of the times, one of the chords in the “sometimes” class, $X_{\text{sometimes}}$ percent of the times, and one of the chord in the “seldom” class, X_{seldom} percent of the times. Obviously, such a probability distribution can be defined in several ways. As an example, in De Felice et al. (2017), the distribution chosen after a statistical evaluation) of Bach's chorales was $(X_{\text{often}}, X_{\text{sometimes}}, X_{\text{seldom}}) = (80, 15, 5)$. Finally, to compute the weights, given the degrees of two chords c_i and c_{i+1} (for example: I for c_i and V for c_{i+1}):

1. Select the value X in the probability distribution $(X_{\text{often}}, X_{\text{sometimes}}, X_{\text{seldom}})$ by searching the passage $c_i \rightarrow c_{i+1}$ in the frequency distribution (for example: I “often” goes to V , and so $X = X_{\text{often}}$).

2. Count the number N of degrees which are the X frequency (for example: there are $N = 3$ degrees in the “often” frequency of I , i.e., I, IV , and V).
3. Calculate the weight as $\frac{X/100}{N}$ to obtain a value in percentage (for example: if $X_{\text{often}} = 80$ and $N = 3$, then the weight is $\frac{0.8}{3} = 0.26$).

Notice that the argument explained above regards passages of chords in the same tonality (major or minor). However, the same argument can be applied to passages of chords in different tonalities, named *modulations* in music. As an example, see Table 8 in De Felice et al. (2017).

Melodic function The melodic function f_m evaluates the melodic quality of a chorale C and is defined as $f_m(C) = \sum_i a_i w_i$, where the index i runs over all “errors”. The objective is to minimize $f_m(C)$. Such errors can be found by performing an “exception analysis” to identify stylistic anomalies and formal errors. Each exception has an associated severity level that indicates its relative importance: “warning” and “error”. A warning exception is intended to highlight a feature that might be stylistically unusual; an error exception indicates a formal problem that should be corrected. There exist two exception classes: *motion exceptions* and *voicing exceptions*. Examples of motion exceptions are parallel octaves and parallel fifths. Examples of voice exceptions are voice jumps, voice crossings (see Table 9 in De Felice et al. 2017).

Once defined the exceptions, the idea is to assign to each of them a coefficient and a weight. The coefficients can be obtained with a statistical analysis of a reference corpus of music, as done for the harmonic function. The weights can be assigned according to the severity level of the exception. As an example, in classical music the parallel fifths exception has a high value of severity, while the voice jump has a lower value of severity. As an example, in Table 9 of De Felice et al. (2017), the weight 2 is assigned to parallel fifths exceptions, and the weight 1 is assigned to voice jump exceptions. Furthermore, again in such a table, the statistical coefficient calculated for the parallel fifths exception is 0.05, while that for voice jump exception is 6.90.

5.3 Implementation

This section reports the result of an experiment with an implementation of the described approach. The details regarding the language used to implement the system, and the hardware platform used to conduct the experiments are the same as in Sect. 4.3. The time required to complete all the experiments was slightly longer, i.e., approximately 4 h 15 min.

Evolution experiments As for the Approach 1, 4-voices chorales were considered and the same set of Bach's chorales has been used as the ground set $\mathcal{G}$: BWV 2.6, BWV 10.7, BWV 11.6, BWV 14.5, BWV 16.6, BWV 20.7, BWV 28.6, BWV 32.6, BWV 40.8 and BWV 44.7. And, as before, each one has been transposed in every tonality. The set $\mathcal{I}_2$ consists of the words corresponding to each chorale and to each transposition.

As for the experiment in the previous section, a set of values for the maximum number t of iterations and for the maximum size $p_{\max}$ of the language has been considered, namely $t \in T = \{50, 100, 250, 500, 750, 1000, 5000, 7500, 10000\}$, and $p_{\max} \in P = \{50, 100, 250, 500, 750, 1000\}$.

Also in this case for each pair $(t, p_{\max})$ 5 executions of $\mathcal{S}_2$ were run. Notice that, differently from the approach described in Sect. 4.3, in this case there is a multi-objective problem, i.e., it is necessary to look for solutions that simultaneously maximize the harmonic value and minimize the melodic value. To this, at the end of each iteration only solutions in the Pareto front are considered for the next iteration. After the last iteration, the output solution is chosen from the Pareto front selecting the one with the best harmonic value. For each experiment both the average harmonic value the average melodic value of the 5 runs have been computed.

The best output was obtained for $t = 7500$ and $p_{\max} = 1000$. Figure 3 shows the music obtained. Notice that, as for Approach 2, the final score has been obtained by applying the same rhythmic operator used in Sect. 4.3 and defined in De Felice et al. (2015); De Felice et al. (2017).

6 Approach 3: evaluation based on the prediction of stylistic patterns

This section presents a music splicing system based on an evaluation function with a completely different approach: it measures the quality of the composition by using a method to predict patterns coherent with a specific style; the patterns are used to guide the splicing system during the composition. For this approach, the specific music problem that is considered is that of recognizing a music style and/or to compose music in a specific style. For this approach, beside the music splicing system, two additional components, called *recognizer* and *predictor*, are needed.

6.1 The music composition problem: music style recognition and composition

The problem considered is the problem of both recognize and compose music for a specific performer's style. To address such a problem, the approach exploits: (i) a *recognizer*, i.e., a machine learning-based classifier to learn a specific music performer's style, (ii) a *composer*, i.e., a music splicing sys-

tem to compose melodic lines in the style learned by the recognizer, and (iii) a *predictor*, i.e., a machine learning method to predict patterns coherent with the style learned by the recognizer, and used by the composer to evaluate the "stylistic" goodness of the composed music pieces.

The composer is defined by an initial set of melodies coherent with a specific style, and a set of rules built by using the predictor. The goal is to generate a language containing words that represent pieces of "new" melodies coherent with the chosen style. The chosen music style is denoted, generically, with "STYLE". When the composer generates a composition, it uses the predictor to verify that the musical patterns within such a composition are actually similar to the expected patterns, i.e., the patterns predicted by the predictor. The best solution is the one that contains the largest number of relevant patterns expected by the predictor.

6.2 The recognizer

The *recognizer* $\mathcal{R}ec$ is a machine learning-based method able to recognize a specific style. As will be discussed in Sect. 6.3, $\mathcal{R}ec$ is necessary to build the predictor used by the composer to evaluate the compositions generated.

Suppose that $\mathcal{R}ec$ needs to be trained to recognizing the style STYLE. Formally, given a melody m , the objective of $\mathcal{R}ec$ is that of classifying m in two possible classes: *coherent* with STYLE or *not coherent* with it. To achieve this $\mathcal{R}ec$ is trained on a corpus of music pieces $\mathcal{M}$, containing solos performed in the style STYLE. In order to train $\mathcal{R}ec$ in an effective way, it is necessary to decide what information, or *features*, of the melodies in the training set will be useful to $\mathcal{R}ec$ in understanding the style. Let v_j be the feature vector containing such significant features of some melody m_j , obtained through the feature extraction model to be discussed shortly. The recognizer classifies m_j by using v_j as input.

Model features The most significant features of a melody can be effectively extracted by using well-known string matching-based techniques. This approach uses the n -gram-based method (Hsiao et al. 2014). The idea is to identify the *tokens* within melodies whose importance can be established using a *statistical measure*. Thus, two problems need to be solved: (i) first, to determine what tokens are and from which parts of the melody they can be extracted, and (ii) then, to define a statistical measure for estimate the importance of such tokens. In our approach, we use a specific set of information about a note to define a token.

- **The music token.** Given a music note, we use three features for defining the corresponding token. Specifically:
 - *chord name*, denoted with k_1 ,
 - *chord type*, denoted with k_2 ,

Fig. 3 Approach 2: The composition given in output by S_2

APPROACH 2

The musical score is titled "APPROACH 2" and shows a composition for four voices. The tempo is indicated as $\text{♩} = 50$. The key signature has two flats (B-flat and E-flat). The score is divided into three systems of staves. The first system contains measures 1-3, the second system contains measures 4-5, and the third system contains measures 6-7. The notation includes various note values, rests, and bar lines.

- *role* of the note with respect to the chord, denoted with k_3 .

The chord used for extracting such information about the note is derived from the modes (major, melodic and harmonic minor scales Levine 2009). Table 1 in De Prisco et al. (2017) reports the description of such chords. More formally, given the music note played at the i^{th} beat, the triple $K^i = [k_1^i, k_2^i, k_3^i]$ is the corresponding token. As an example, the token $K^5 = [0, 4, 4]$ says that the note played at the 5^{th} beat has degree *III* ($k_3 = 4$ means degree *III*) in the scale corresponding to the chord *C7*

($k_1 = 0$ means chord *C* and $k_2 = 4$ means chord type 7). Thus the note played is *E*.

- **A statistical measure for token importance.** The statistical measure defined to establish the importance of n -grams is the *term frequency with inverse document frequency* (*tfidf*). We use such measure for giving more weight to terms that are less common in $\mathcal{M}$ (more likely terms to make the corresponding melody stand out), and for transforming the corpus $\mathcal{M}$ to a feature vector space. Given a sequence of n music notes, we compute the n tokens, and such sequence of tokens is used as n -gram

(or term) t . Then, using such term definition, the tf and the idf can be defined as:

- if $t \in m_j$, then $tf(t, m_j) = 1$ (this means that the sequence of notes occurs in the melody m_j), 0 otherwise,
- $idf(t, m_j, \mathcal{M}) = \log\left(\frac{|\mathcal{M}|}{|\{m_j \in \mathcal{M} : t \in m_j\}|}\right)$.

In conclusion, given a melody $m_j = n_1, \dots, n_k$ (k notes), the corresponding feature vector v_j is built as: for each note n_i , with $i = 1, \dots, k$, the component $v_j[i]$ is equal to $tfidf(t_i, m_j, \mathcal{M})$, where $tfidf(t, m_j, \mathcal{M}) = tf(t, m_j) \cdot idf(t, m_j, \mathcal{M})$.

The classifier Let m_j be a melody and v_j be its corresponding feature vector. Such a vector contains an element for each significant feature of the melody, and the value of this element is the $tfidf$. Several machine learning models can be trained to perform this task. In De Prisco et al. (2017) a One-Class Support Vector Machine has been used.

6.3 The predictor

The *predictor* Pre is a machine learning method to predict patterns coherent with the style learned by Rec . The predictor Pre is used by the music splicing composer defined in the following, to evaluate the “stylistic” goodness of the composed music pieces.

Let $\mathcal{M}$ be the chosen set of solos, and n be the value used for the construction of the n -grams as described in Sect. 6.2. Then, the idea is to define a machine learning predictor Pre that given an n -gram at time i has to predict the n -gram at time $t+1$. This is equivalent to saying that given the sequence of n music notes at time i , Pre has to predict the sequence of n music notes at time $i+1$.

The training set The data set of n -grams is defined as follows. Let $\mathcal{T} \subseteq \mathcal{M}$ be the training set used for the training of Rec . For each $m_j \in \mathcal{T}$ such that Rec says that m_j is coherent with the style $STYLE(f(m_j) < 0)$, consider the sequence of n -grams extract by m_j . Let $Ngrams(m_j) = (ng_1, \dots, ng_{k_j})$ be the sequence of n -grams extract from m_j , and insert the pair (n_i, n_{i+1}) in the training set for Pre , for each $1 \leq i \leq k_j - 1$.

The architecture Several machine learning models can be trained on the set $\mathcal{T}$ above described. In De Prisco et al. (2017) an Long short-term memory (LSTM) network has been used.

6.4 The splicing system: approach details

This section describes the splicing system $\mathcal{S}_3 = (\mathcal{A}_3, \mathcal{I}_3, \mathcal{R}_3)$, which is the core of Approach 3. In the following, details

Fig. 4 First 2 measures of Charlie Parker's Now's the time

about $\mathcal{A}_3, \mathcal{I}_3, \mathcal{R}_3$ are provided together with a discussion about the composition questions.

cQ1: word-based representation A suitable alphabet $\mathcal{A}_3$ is needed. Let $\mathcal{A}_3$ be $\mathcal{A}_3 = \mathcal{A}_N \cup \mathcal{A}_T \cup \mathcal{A}_D \cup \mathcal{A}_S$ where $\mathcal{A}_N$ is the *chord name alphabet*, $\mathcal{A}_T$ is the *chord type alphabet*, $\mathcal{A}_D$ is *note degree alphabet* and $\mathcal{A}_S$ is the *separator alphabet*. Specifically, $\mathcal{A}_N = \mathcal{A}_D = \{0, \dots, 11\}$, $\mathcal{A}_T = \{0, \dots, 20\}$ and $\mathcal{A}_S = \{\tau, \mu_N, \mu_T, \mu_D\}$. Alphabet $\mathcal{A}_3$ is used to represent solos (melodies) as words.

As explained in Sect. 6.2 each token represents a music note. Thus, given a melody $m = (n_1, \dots, n_l)$, for each note n_i , the token K^i is represented as a word w_i over $\mathcal{A}_3$. Specifically $w_i = \mu_N x_i \mu_T y_i \mu_D v_i$, where $x_i \in \mathcal{A}_N$, $y_i \in \mathcal{A}_T$ and $v_i \in \mathcal{A}_D$, for each $1 \leq i \leq l$. Thus, m is represented by the word $\mathcal{W}(m) = \tau w_1 \tau w_2 \tau \dots \tau w_n \tau$.

Example 3 Consider the melody m , shown in Fig. 4 (first 2 measures of the standard Jazz “Now’s the time” of Charlie Parker), and consider the chords organization proposed in De Prisco et al. (2017) (see Table 1). For this melody $m = (n_1, \dots, n_{12})$. Each n_i is played in F7 chord. The chord name F has value 5. Chord F7 is the associated scale is the mixolydian mode of the major scale, thus the chord type has value 4. Now, let us analyze the role of notes respect the F7: $n_1 = n_5 = n_7 = n_{11} = C$ which has degree v (position value 7), $n_2 = n_3 = n_6 = n_8 = n_9 = n_{12} = F$ which has degree i (position value 0), and $n_4 = n_{10} = G$ which has degree ii (position value 2). Thus, there are 10 tokens: $K^1 = K^5 = K^7 = K^{11} = (4, 5, 0)$, $K^2 = K^3 = K^6 = K^8 = K^9 = K^{12} = (4, 5, 7)$, and $K^4 = K^{10} = (4, 5, 2)$. So, $\mathcal{W}(m) = \tau w_1 \tau \dots \tau w_{12} \tau$ and $w_1 = \mu_N 4 \mu_T 5 \mu_D 0$, $w_2 = \mu_N 4 \mu_T 5 \mu_D 7$, $w_3 = \mu_N 4 \mu_T 5 \mu_D 7$, $w_4 = \mu_N 4 \mu_T 5 \mu_D 2$, $w_5 = \mu_N 4 \mu_T 5 \mu_D 0$, $w_6 = \mu_N 4 \mu_T 5 \mu_D 7$, $w_7 = \mu_N 4 \mu_T 5 \mu_D 0$, $w_8 = \mu_N 4 \mu_T 5 \mu_D 7$, $w_9 = \mu_N 4 \mu_T 5 \mu_D 7$, $w_{10} = \mu_N 4 \mu_T 5 \mu_D 2$, $w_{11} = \mu_N 4 \mu_T 5 \mu_D 0$, and $w_{12} = \mu_N 4 \mu_T 5 \mu_D 7$.

cQ2: rules definition The idea is to start from an initial set of melodies known to be coherent with the style $STYLE$. Consider the following set of n -grams (the n used by Rec): let $\mathcal{T}$ be the training set used for the training of Rec . For each $m_j \in \mathcal{T}$ such that Rec says that m_j is coherent with the style $STYLE(f(m_j) < 0)$, consider the list $Ngrams(m_j)$ of n -grams extracted by m_j . Let $Ngrams(m_j) = (ng_1, \dots, ng_{k_j})$, and insert $\mathcal{W}(ng_i)$ in the set $\mathcal{I}_3$, for each $1 \leq i \leq k_j - 1$.

Fig. 5 Approach 3: The composition given in output

APPROACH 3

Unlike the problem considered in Sects. 4 and 5, in which it was possible to model rules by using music theory, in this case there are no such rules (composing melodies coherent with a style is not coded through rules). Here the predictor Pre as needed and is used as follows.

1. *Group 1 (forcing sequence n -grams).* Let $w_i \in \mathcal{I}_3$ and n_i such that $w_i = \mathcal{W}(n_i)$; the rule is defined as $r = w_i | \epsilon \in \mathcal{W}(Pre(n_i))$ where, $Pre(n_i)$ is the n -gram predicted by Pre with n_i as input. These rules force the composer to paste words corresponding to n -grams that really follows n_i in the training set $\mathcal{T}$.
2. *Group 2 (forcing sequence tokens).* Given $w_i, w_j \in \mathcal{I}_3$, n_i such that $w_i = \mathcal{W}(n_i)$, n_j such that $w_j = \mathcal{W}(n_j)$, where $n_i = (k_{i,1}, \dots, k_{i,n})$, $n_j = (k_{j,1}, \dots, k_{j,n})$ and there exist l_1, l_2 such that $k_{i,l_1} = k_{j,l_2}$. Then, the rule is defined as $r = \mathcal{W}(k_{i,1}, \dots, k_{i,l_1-1}) | \epsilon \in \mathcal{W}(k_{j,l_2}, \dots, k_{j,n_j})$. If n_i and n_j share a token, then these rules force the composer to cut w_i and w_j in correspondence of such a token, and paste the remaining words.

cq3: evolutionary process Also in this approach a maximum number of iterations is used as stop criterion. As for the definition of the above rules, also for the definition of the function f_e to evaluate the melodies produced, there are no music rules that can be used. So the idea is to exploit the predictor Pre as follows. Let w be a word generated by the composer S_3 . Let $m = (n_1, \dots, n_l)$ the melody such that $w = \mathcal{W}(m)$. Now, let $Ngrams(m) = (ng_1, \dots, ng_{l-1})$ be the sequence of n -grams extracted from m . The function f_e is defined as:

$$f_e(m) = \sum_{1 \leq i \leq l-1} (tfidf(ng_i, m, \mathcal{M})) + Diff(ng_{i+1}, Pre(ng_i))$$

where $Diff(ng_i, Pre(ng_i))$ is the *difference* between ng_{i+1} and $Pre(ng_i)$ that is the n -gram predicted by Pre with ng_i as input. Such a difference is defined as follows: let $K^{i+1} = [k_1^{i+1}, k_2^{i+1}, k_3^{i+1}]$ be the token for ng_{i+1} and $K'^{i+1} = [k_1'^{i+1}, k_2'^{i+1}, k_3'^{i+1}]$ be the token for $Pre(ng_i)$. Then $Diff(ng_{i+1}, Pre(ng_i)) = |k_1^{i+1} - k_1'^{i+1}| + |k_2^{i+1} - k_2'^{i+1}| + |k_3^{i+1} - k_3'^{i+1}|$.

6.5 Implementation

This section describes an implementation of a music splicing system based on Approach 3. As explained in Sects. 6.2 and 6.3, in this case it is necessary to implement a recognizer $\mathcal{R}ec$ to learn a specific music performer's style, and a predictor Pre to predict patterns coherent with the learned style and used to guide the splicing composition. Following De Prisco et al. (2017), the implementation described here uses a One-Class Support Vector Machine to implement $\mathcal{R}ec$, and a Long short-term memory network to implement Pre .

The details regarding the language used to implement the music splicing system, and the hardware platform used to conduct the experiments are the same as in Sect. 4.3. However, in this case, the recognizer $\mathcal{R}ec$ and the predictor Pre have been implemented using the Python library scikit-learn.² The time required to complete all the experiments was approximately 4h 5m.

Evolution experiments The focus is on Jazz music, that is $STYLE = Jazz$. The data set $\mathcal{M}$ consisted of Jazz solos, transcribed in MusicXML format, from one of the most popular Jazz musicians: Louis Armstrong. Specifically, $|\mathcal{M}| = 50$.

² <https://scikit-learn.org/stable/>.

As for the previous cases, the experiments have been run using a number t of iterations in the set $T = \{50, 100, 250, 500, 750, 1000, 5000, 7500, 10,000\}$, and the a max size $p_{\max}$ in the set $P = \{50, 100, 250, 500, 750, 1000\}$.

For each pair $(t, p_{\max})$ 5 executions have been run. For each experiment the average f_e function value described in Sect. 6.4 has been computed. Notice that the function f_e evaluates a composition in terms of “stylistic” goodness. The best solution has been obtained by setting $t = 500$ and $p_{\max} = 1000$.

An example of music composition generated Observe that, by definition, when a word w is generated it only represents a sequence of music notes $m = (n_1, \dots, n_k)$. A rhythmic structure can be added to m by defining an operator that: (i) applies *rhythmic transformations* to m , (ii) *modify the duration* of the notes, (iii) *create rest notes*, (iv) *tie notes* together, and (v) *create triplet notes*. Each of these operation is performed with a uniform distribution of probabilities. In Fig. 5 shows the music score of the best solution after applying such operator, as defined in De Prisco et al. (2017).

7 Conclusions

Starting from the famous Adleman experiment, DNA Computing has aroused an increasing interest, both from an *applications* point of view, coming to be used in various contexts, and from a *theoretical* point of view, with researchers that have borrowed tools from various areas of theoretical computer science, such as formal language theory, coding theory, and combinatorics on words. Recently, several works have used DNA Computing for reproducing human beings’ creativity and in particular music creativity. We have provided a survey of automatic music composers based on splicing systems, and, as a novel contribution we have identified the crucial points behind the proposed solutions. Each specific approach is characterized by the way it tackles these problems.

Funding The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Data Availability Examples of music compositions produced during the experiments can be found online (<https://cutt.ly/Tbke2EP>).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as

long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Acampora G, Cadenas JM, De Prisco R, Loia V, Muñoz E, Zaccagnino R (2011) A hybrid computational intelligence approach for automatic music composition. In: 2011 IEEE international conference on fuzzy systems (FUZZ-IEEE 2011), pp 202–209. <https://doi.org/10.1109/FUZZY.2011.6007524>
- Adamatzky A (2010) Physarum machines: computers from slime mould, vol 74. World Scientific, Singapore
- Adleman LM (1994) Molecular computation of solutions to combinatorial problems. Science 266(5187):1021–1024
- Biles J (1994) Genjam: a genetic algorithm for generating jazz solos. In: ICMC, vol 94. Ann Arbor, MI, pp 131–137
- Braund E, Miranda ER (2013) Music with unconventional computing: towards a platform for physarum polycephalum sound synthesis. In: Proceedings of 10th international symposium on computer music multidisciplinary research (CMMR): sound music and motion
- Braund E, Miranda ER (2014) Unconventional computing in music. In: Proceedings of the 9th conference on interdisciplinary musicology—CIM14. Berlin, Germany. Citeseer
- De Felice C, De Prisco R, Malandrino D, Zaccagnino G, Zaccagnino R, Zizza R (2017) Splicing music composition. Inf Sci 385:196–212
- De Felice C, De Prisco R, Malandrino D, Zaccagnino G, Zaccagnino R, Zizza R (2015) Chorale music splicing system: an algorithmic music composer inspired by molecular splicing. In: International conference on evolutionary and biologically inspired music and art. Springer, pp 50–61
- De Prisco R, Malandrino D, Zaccagnino G, Zaccagnino R, Zizza R (2017) A kind of bio-inspired learning of music style. In: International conference on evolutionary and biologically inspired music and art. Springer, pp 97–113
- Head T (1987) Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. Bull Math Biol 49(6):737–759
- Head T (1987) Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviours. Bull Math Biol 49:737–759
- Hsiao CH, Cafarella M, Narayanasamy S (2014) Using web corpus statistics for program analysis. OOPSLA. ACM, pp 49–65
- Jeong J, Kim Y, Ahn CW (2017) A multi-objective evolutionary approach to automatic melody generation. Expert Syst Appl 90(C):50–61
- Levine M (2009) The jazz theory book. Curci
- Liu CH, Ting CK (2015) Music pattern mining for chromosome representation in evolutionary composition. In: 2015 IEEE congress on evolutionary computation (CEC), pp 2145–2152
- Miranda ER (1993) Cellular automata music: an interdisciplinary project. J New Music Res 22(1):3–21
- Miranda ER (1995) Granular synthesis of sounds by means of a cellular automaton. Leonardo 28(4):297–300

- Miranda ER (2014) Harnessing the intelligence of physarum polycephalum for unconventional computing-aided musical composition. *Int J Unconv Comput* 10(3):251–268
- Miranda ER, Biles JA (2007) *Evolutionary computer music*. Springer, Berlin
- Miranda ER, Bull L, Gueguen F, Uroukov IS (2009) Computer music meets unconventional computing: towards sound synthesis with in vitro neuronal networks. *Comput Music J* 33(1):9–18
- Ogiela MR, Ogiela L (2015) Personalized cryptography in data sharing and management. In: 2015 18th international conference on network-based information systems. IEEE, pp 269–272
- Ogiela MR, Ogiela L (2017) Behavioral keys in cryptography and security systems. In: International conference on intelligent networking and collaborative systems. Springer, pp 296–300
- Ogiela MR, Ogiela L (2017) Cognitive keys in personalized cryptography. In: 2017 IEEE 31st international conference on advanced information networking and applications (AINA). IEEE, pp 1050–1054
- Păun G (1996) On the splicing operation. *Discret Appl Math* 70:57–79
- Piston W, DeVoto M (1987) *harmony*. W. W. Norton, New York
- Pixton D (1996) Regularity of splicing languages. *Discret Appl Math* 69(1–2):101–124
- Prisco RD, Malandrino D, Zaccagnino G, Zaccagnino R, Zizza R (2017) Splicing-inspired recognition and composition of musical collectives styles. In: Martín-Vide C, Neruda R, Vega-Rodríguez MA (eds) *Theory and practice of natural computing - 6th international conference, TPNC 2017, Prague, Czech Republic, December 18–20, 2017, Proceedings, Lecture Notes in Computer Science*, vol 10687. Springer, Berlin, pp 219–231
- Prisco RD, Zaccagnino G, Zaccagnino R (2020) Evocomposer: An evolutionary algorithm for 4-voice music compositions. *Evol Comput* 28(3):489–530
- Simon HA (1995) Artificial intelligence: an empirical science. *Artif Intell* 77(1):95–127
- Toffoli T, Margolus N (1991) Programmable matter: concepts and realization. *Phys D* 47(1):263–272. [https://doi.org/10.1016/0167-2789\(91\)90296-L](https://doi.org/10.1016/0167-2789(91)90296-L)
- Turing AM (1992) *Mechanical intelligence*. North-Holland, Amsterdam
- Xenakis I (1992) *Formalized music: thought and mathematics in composition*, vol 6. Pendragon Press, New York

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.