Evol. Intel. (2015) 8:23–35
DOI 10.1007/s12065-014-0118-2

SPECIAL ISSUE

Monterey Mirror: an experiment in interactive music performance combining evolutionary computation and Zipf's law

Bill Manaris · Dana Hughes · Yiorgos Vassilandonakis

Received: 2 November 2013 / Revised: 11 July 2014 / Accepted: 30 October 2014 / Published online: 21 November 2014

Abstract Monterey Mirror is an experiment in interactive music performance. It is engages a human (the performer) and a computer (the mirror) in a game of playing, listening, and exchanging musical ideas. The computer side involves an interactive stochastic music generator which incorporates Markov models, genetic algorithms, and power-law metrics. This approach combines the predictive power of Markov models with the innovative power of genetic algorithms, using power-law metrics for fitness evaluation. These power-law metrics have been developed and refined in a decade-long project, which explores music information retrieval based on Zipf's law and related power laws. We describe the architecture of Monterey Mirror, which can generate musical responses based on aesthetic variations of user input. We also explore how such a system may be used as a musical meta-instrument/environment in avant-garde music composition and performance projects.

Keywords Genetic algorithms · Markov models · Zipf's law · Power laws · Stochastic music · Music performance · Music composition

B. Manaris (✉) · Y. Vassilandonakis
College of Charleston, Charleston, SC 29424, USA
e-mail: manarisb@cofc.edu

Y. Vassilandonakis
e-mail: vassilandonakisy@cofc.edu

D. Hughes
University of Colorado Boulder, Boulder, CO 80309, USA
e-mail: dana.hughes@colorado.edu

1 Introduction

Music, over the thousands of years of its existence, has been influenced considerably by mathematical and technological advances in society. These advances have contributed, for instance, to the development of new instruments (e.g., brass instruments, and the violin), to new ways of formulating music (e.g., equal temperament, and stochastic models), and finally, within the last 50 years, to automated, computational techniques contributing to the analysis, generation, composition, and performance of music.

Music narrative in Western music is traditionally advanced through an expansive process, shaped by transforming a few central musical gestures (themes) into larger sections, according to pre-conceived formal designs (i.e., sonata form, or theme and variations). The listener is able to follow the narrative by staying connected to the various transformations of the thematic material, and by being familiar with the general structure of the formal designs employed by the composer. This music language is closely linked to tonality, which fades by the beginning of the twentieth century, at which point composers seek out alternative ways of organizing musical material. The importance of thematic unity and pre-conceived formal design diminish considerably, giving way to novel formal plans that are more ambitious in design, scope and complexity, and are shaped by organizing musical parameters previously considered secondary, such as texture, timbre, registral placement and density.

In the 1960's, composers like Gyorgy Ligeti and Iannis Xenakis started experimenting with cloud-like textures and dense sound masses, where the number of sound particles is so high that the listener cannot possibly follow their individual trajectories, and therefore concentrates on

tracing the behavior of the overall texture and its shifts over time (e.g., [1]). From the perspective of the composer, it became increasingly difficult to plan and execute such complex textures without using computers. Iannis Xenakis, coming from an engineering and mathematics background was among the first composers to turn to computer-generated algorithmic processes in order to calculate the behavior of a large number of sound particles. Another important stylistic development of the late 1960's was the use of aleatoric principles in composition, as a reaction to the serialist tendencies of the post-war era, which exercised total control of the pitch and rhythmic material. Composer John Cage introduced a stylistically important new idea, according to which the composer does not seek to control every minute detail of a composition, but instead leaves certain elements to be decided by the performer, or by pure chance [2]. This development introduces improvisational freedom within certain layers of an otherwise through-composed piece, yet maintains control of the overall texture, character and direction of the composition. Later composers like Witold Lutoslawski adapted these ideas and personalized the use of chance elements in orchestra pieces, otherwise conventional in nature. Finally, as computer science and artificial intelligence techniques matured during the 1980's and 1990's, a new compositional trend emerged in contemporary music, combining elements of aleatorism and algorithmic processes to provide the contemporary composer with choices unavailable before.

Computer-assisted composition and performance involves the use of some process to generate new musical material or expand upon existing one. These processes can assist in exploring material more rapidly and with considerably more precision and detail than one would be able to do alone, and can furthermore steer the composer toward entirely new explorations. They also provide a considerable increase in the level of control of large-scale planning and management of musical gestures, events and formals structures, especially as the latter get more complex and involve a large number of elements.

This paper presents an innovative system, called *Monterey Mirror*, which exists in the intersection of techniques for the analysis, generation, composition, and performance of music. Monterey Mirror is an experiment in interactive music performance. It is engages a human (the performer) and a computer (the mirror) in a game of playing, listening, and exchanging musical ideas (see Fig. 1). Monterey Mirror combines Markov models, genetic algorithms, and power laws to create an environment, which can be viewed both as

- a music meta-instrument, in that it can be used in real-time (through a well-defined MIDI interface) to generate music interactively, and

Fig. 1 One of the authors demonstrating Monterey Mirror (see http://goo.gl/782W3F)

- a compositional space, in that it introduces new possibilities to shaping music form and interactions between human and artificial musicians.

You may watch a demo of the system at http://goo.gl/782W3F.

The remainder of this paper is organized as follows. Section 2 presents related systems and background research. Section 3 describes the architecture of Monterey Mirror. Section 4 discusses how Monterey Mirror may be used in contemporary music composition. Section 5 presents a case study, i.e., one particular musical piece performed with two human performers and two Monterey Mirrors. The paper closes with concluding remarks and future directions.

2 Background

As already mentioned, Monterey Mirror combines Markov models, genetic algorithms, and power-law metrics. Since there are two target audiences for this paper, computer scientists and music composers, this section provides an outline of the essential aspects of these techniques. Additionally, we discuss relevant earlier work in algorithmic music composition and performance.

2.1 Markov models

Markov models are stochastic systems, which capture probabilities of the occurrence of particular events based on earlier context [3]. Markov models can be trained using existing sequences of events (e.g., words in a book, or notes in a musical piece). The training algorithm constructs an n-dimensional matrix (either explicitly or implicitly)

which records how often one subsequence of events resolves into a given event. Once trained, a Markov model can be used to generate a new sequence of events that is statistically similar to the sequences of events used for training. It is possible for a Markov model to recreate an exact training sequence, but it is highly improbable for all practical purposes (i.e., it depends on the training corpus, and the different ways particular subsequences of events resolve).

2.2 Genetic algorithms

A *genetic algorithm* is a computational approach for building sequences of symbols (e.g., musical phrases) that have certain desirable properties [4]. These properties are evaluated through a *fitness function*, which computes measurements specific to the task at hand (e.g., measurable aspect(s) of musical aesthetics). These measurements estimate the distance of a constructed artifact from the best possible artifact(s). A genetic algorithm progresses by constructing generations of individual artifacts. Each generation is produced by evolving individuals from the previous generation. The first generation is initialized in some random way. The best individuals from each generation (also known as *the elite*) are preserved; they are passed unchanged to the next generation. This serves as a short-term memory of the best "solutions" found so far. The remaining (non-elite) individuals are modified through well-defined operations, such as mutation and crossover. Mutation changes parts of an individual. Crossover swaps parts between two randomly selected *parent* individuals. This process continues until the fitness function identifies an individual that is close enough to the desired outcome, or after a number of generations have passed without finding such an individual.

2.3 Algorithmic composition

The development of computers has enabled a wide variety of programs, within the last 50 years, performing aspects of analysis, generation, composition, and performance of music [5–12]. Approaches similar to Monterey Mirror are briefly described here.

GenJam uses a genetic algorithm to generate jazz improvisations [5]. GenJam is a very mature system that evolves melody responses in real-time. However, the system does not use a fitness function—instead the crossover and mutation operations are hard-wired to produce interesting variations of the immediate user input. GenJam can improvise full-chorus solos, trade fours, eights, 12s or 16s with its human sideman.

A significant advantage of genetic algorithms is that they search a large space of possibilities quickly for optimal solutions. Individuals in a genetic algorithm's population explore different parts of the search space. Individuals with more desirable characteristics attract other individuals to their region of the search space, by surviving across several generations, and contributing genetic material via crossover. Mutation ensures that individuals explore new areas in the search space, potentially arriving to previously unexplored, yet more desirable solutions. If the search space is very large, it is difficult to find desirable solutions in a practical amount of time, making this approach prohibitive for interactive, real-time music applications.

A very important consideration when using genetic algorithms is to determine an appropriate fitness function, which guides the evolutionary process. If a fitness function leaves certain important aspects unmeasured (i.e., has "holes"), the genetic process will exploit these "holes" to arrive at solutions that are substandard. Early versions of *GenJam* used an interactive genetic algorithm, where the human performer judges individuals each generation. This avoids the need of finding a potentially insufficient fitness function, but requires much off-line work from the performer to produce sufficient material.

Experiments in Musical Intelligence (Emmy) is a system that generates novel musical pieces using Markov models [6]. The system works off-line. It has generated several pieces in the style of various composers, including Bach and Chopin. One important aspect of Emmy is the need for human judgment—the system generates melodies using a Markov model, but these melodies require a human to evaluate them for aesthetic quality. Consequently, Emmy cannot be used in a performance setting.

Continuator is an interactive music performance system that accepts partial input from a human musician and continues in the same style as the input [7]. The system utilizes various Markov models to learn from the user input. The Markov models are organized in decreasing order of information retrained, i.e., the first one remembers everything the user has played (e.g., note pitches, durations, start times, etc.). Subsequent Markov models retain less information. The system tries to provide an answer form the most informed Markov model, and if a match is not found with the user input, the system continues with the less-informed Markov models. Thus, the system sometimes generates accurate "reproductions" (or continuations) of the user input, and other times less accurate ones (which sound as interesting variations).

Markov models are useful for capturing statistical similarity of provided training data, such as a corpus of a given composer (as with *Emmy*), or learned during interactive performance (as with *Continuator*). An important limitation of Markov models is that statistical similarity between the training material and the generated material is *local*.

Markov models learn statistical similarities based on a limited history of symbols, and cannot capture long-term dependencies in the material. For instance, it is quite common in music to have long-term dependencies, such as the ABA, and ABBA high-level forms. In other works, material used in the beginning of a song (i.e., initial section A) is repeated (perhaps with some variation) at the end (i.e., final section A). It is possible for a Markov model to generate a sequence with this form, assuming it was trained with music of this form. However, the chances of arriving at such an outcome are exponentially small.

NEvMuse is a system combining evolutionary computation with artificial music critics to evolve variations of musical excerpts [8]. NEvMuse makes extensive use of power-law metrics, which measure statistical proportions of musical artifacts. Power-law metrics are based on Zipf's law, and have been shown to capture essential aspects of music aesthetics. In particular, using power-law metrics as features, earlier experiments with artificial neural networks have demonstrated high accuracies, e.g., for composer identification, 93.6–95 % accuracy; style classification, 71.5–96.6 % accuracy; and pleasantness precision, 90.7 % accuracy. Finally, several psychological experiments (e.g., pleasantness prediction, popularity prediction, and similarity prediction) have demonstrated the aesthetic relevance of power-law metrics relative to human listeners [8, 13–15]. Using power-law metrics allows for generating variations without the need for human judgment of candidates. *NEvMuse* is a system which generates variation in off-line, and cannot be used for performance.

2.4 Power-law metrics

Power laws are statistical models of proportion exhibited by various natural and artificial phenomena. These laws relate the probability of a certain event occurring, $P(f)$, with the frequency of occurrence, f, by the following equation:

$$P(f) \sim 1/f^{\alpha}$$

Zipf's law is a special type of a power law, where α is 1. Zipfian phenomena are those where the 2nd most frequent event appears 1/2 as many times as the 1st most frequent event; the 3rd most frequent event appears 1/3 as many times; the 4th appears 1/4 as many times, and so on. This relationship is found in several natural and man-made phenomena, including magnitudes of earthquakes, thickness of sediment deposition, extinction of species, music, city sizes, income, subroutine calls, opening chess moves, word occurrences in texts, traffic jams, and visits to websites [16–18]. Note that power laws are not an exact predictor of the rank-frequency of occurrence of events, but is a statistical observation of different types of events.

The presence of power-law distributions in music is of particular interest. Voss and Clarke first noted that the pitch and loudness of music and speech follow Zipf's law [17]. They further observe that melodies generated using $1/f$ noise (pink noise) were found to be more pleasing to listeners than those generated using white noise or $1/f^2$ noise (brown noise). Melodies generated using white noise are found to be too random, while those generated using brown noise are found to be too correlated or monotonous. Hsu and Hsu later found $1/f$ distributions in frequency intervals of Bach and Mozart compositions [21].

A family of metrics has been created based on the power-law distribution of various musical parameters in a piece, such as pitch and tempo [13]. To calculate a metric, a histogram of a particular parameter is created. These are sorted to obtain a rank-frequency distribution of events. Using the log of the rank and frequency of events, a power-law distribution would show a linear relationship between rank and frequency. Simple linear regression is used to fit a line to the log–log rank-frequency distribution, and a corresponding coefficient of determination (R^2). The slope of the resulting line varies between 0 and $-\infty$, and indicates the complexity of the parameter measured. A slope of 0 indicates a very chaotic distribution (white noise), a slope of -1 indicates a Zipf distribution, and slopes less than -2 indicate very monotonous, highly-correlated distribution. The R^2 value indicates how close the distribution fits an ideal power-law distribution. The R^2 value varies from 0 to 1, 1 indicating a perfect fit to the line, and 0 indicating no correlation between the data and the line. Both the slope and R^2 value are used as metrics. The intercept of the line is discarded, as it only indicates the overall number of events.

Several hundred metrics have been created based on different parameters of music, both in the audio and symbolic (i.e., MIDI) domain [13, 14]. Using artificial neural networks, these metrics have been shown to be very useful in identifying the composer of a piece of music, genre, popularity/pleasantness of a piece of music, and as an artificial critic for generated music [8, 13–15]. In this study, we consider metrics based on the following parameters of a piece of music:

- *Pitch*: rank-frequency distribution of the pitches (MIDI note values) of notes.
- *Duration*: rank-frequency distribution of the duration of notes.
- *Pitch Duration*: rank-frequency distribution of pitch durations.
- *Pitch Distance*: rank-frequency distribution of the length of time intervals between pitch repetitions.
- *Duration Distance*: rank-frequency distribution of the length of time intervals between duration repetitions.

The use of power-law metrics for fitness is described in Sect. 3.2.4. A more thorough description of the Zipf metrics is available in [13].

3 System architecture

Monterey Mirror is designed to generate novel responses to user-provided musical gestures and ideas (melodic, rhythmic, and harmonic) in real time. The system is divided into two main components, the *Listener* and the *Mirror* as shown in Fig. 2. The Listener interacts directly with the user to receive and process events generated by MIDI instruments, and to play generated musical responses. The Mirror generates the musical responses when queried by the Listener, utilizing a Markov model, a genetic algorithm, and numerous power-law metrics for fitness.

3.1 Listener

The Listener is an event-driven component, which processes MIDI messages and constructs higher-level note objects (consisting of pitch, rhythm, volume and channel). Interactions with the Listener are achieved through one or more MIDI channels. The user provides input material using a MIDI instrument. Predefined MIDI events (or sequences) are used as triggers to switch among different system modes (see below). Finally, generated responses are played back to the user through a MIDI synthesizer. The listener has a local memory, which is a buffer to store sequences of MIDI notes recently played by the user. A corresponding response buffer is used to store responses generated by the Mirror.

The Listener must convert MIDI events received from the user's instrument to musical notes or triggering events for the Mirror. The data used by the Mirror consist of a note's pitch, duration, time of onset and (optionally) volume. The onset time is measured relative to when the Listener was instructed to record a melody using a *ListenOn* event, as described below. MIDI events, by contrast, may be note-on or note-off events, and contain pitch and velocity information. To properly convert between MIDI events, the Listener maintains an array of note pitches indicating if that note is in the on state (i.e., has received a note-on MIDI event, but not a corresponding note-off MIDI event) or off, as well as the time of the note-on event. When the Listener is instructed to record a melody, every note value in this array is initialized to be in the off state. When a note-on event is received, the note value in the array is set to the on state, and the MIDI event time and note velocity (volume) is recorded. When a note-off event is received, the Listener reads the onset time from the array, calculates the duration of the note, and stores the note's pitch, duration, start time and volume in the local memory buffer. If a note-on event is received for a note which is already in the on state, then the new onset time and velocity is recorded in the array. If a note-off event is received when the corresponding note in the array is also in the off state, then the note-off event is simply discarded. These two constraints ensure that unwanted notes are not introduced into local memory due to a missed or repeated MIDI event.

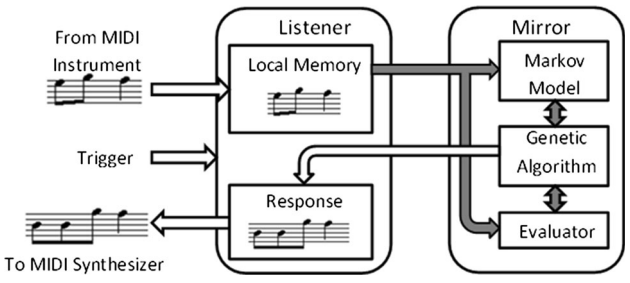

Fig. 2 Overview of the architecture of Monterey Mirror

The Listener is configured to respond to specific MIDI events to allow the user to switch between different modes of the system. These specific MIDI events can either be onset of specific notes, such as the highest or lowest keys of the instrument being played, or specific note-on events from a separate instrument (e.g., when the user is playing a piano), such as a set of MIDI foot pedals (e.g. when the user is playing a guitar). The Listener pre-filters MIDI events, and processes pre-defined events as triggers, rather than notes. The user may trigger the Listener to switch among the following system modes:

- *ListenOn*: The Listener begins to record the user's melody (i.e., sequence of MIDI events) for use as basis for the Mirror's response. The pitch, rhythm and dynamic value of notes played by the user are added to the Listener's local memory. The Listener's local memory is flushed at every ListenOn trigger, allowing the user to generate a new melody for future use.
- *ListenOff*: The Listener stops recording notes played by the user. The local memory is maintained for future use.
- *Respond*: The Listener asks the Mirror to generate a response. The Listener provides the Mirror with the current local memory (i.e., the user's most recently played musical material) for use as learning information, and as a basis for generating a response.
- *Forget*: The Listener instructs the Mirror to forget any previously learned material. This allows learning new material (e.g., user style).

The Listener listens concurrently to the user while requesting and playing Mirror responses, so that no user material is lost. Requesting and playing a response can also

be triggered separately, allowing the user to have the Listener respond at a rhythmically appropriate time during performance.

The Listener interacts with the Mirror through the following API:

- *Learn*: The Listener provides the Mirror with previously unlearned portions of user musical material.
- *Forget*: The Listener instructs the Mirror to forget any previously learned musical material.
- *Request Response*: The Listener passes user musical material and requests a Mirror response to be generated.

When the Listener receives a *Respond* event, it sends a *Learn* message to the Mirror, followed by a *Request Response* message. When the Mirror finishes generating a response, it is provided to the Listener to be played back through a predefined MIDI channel. When a *Forget* event is received, the Listener simply sends a *Forget* message to the Mirror.

3.2 Mirror

The Mirror component is responsible for generating responses to provided musical material. In contrast to the Continuator system (see Background), the Mirror provides "answers" to user "questions". This is more in line with what users usually expect [7].

As mentioned earlier, the Mirror consists of three components, a Markov model, a genetic algorithm, and a set of metrics based on Zipf's law, which are used in the GA's fitness function. In addition, the Mirror can be configured to provide responses by generating new sequences of pitch (i.e., melody), durations (i.e., rhythm), or a combination of the two. Representation of notes and the three main components are discussed in further detail below.

3.2.1 Note representation

The Mirror can be configured to generate responses by generating new sequences of pitch, new sequences of duration, or new sequences of pitch and duration. If only sequences of pitches or durations are generated, a response can be constructed by replacing the sequence of pitches (or durations) in the material provided by the user. The Mirror behaves the same regardless of which musical aspect is being generated—only the symbolic representation of the data is changed. In addition to different symbolic representation, there are unique considerations for generating each aspect, which results in different preprocessing requirements for each approach.

Pitch can be represented using either the absolute pitch, or pitch after transposing the melody to a common key. Absolute pitch can be represented simply using the MIDI pitch value of the note (e.g., 60 for C4). The benefit of this approach is that it requires no pre-processing of the pitch values from provided material. However, this approach will consider the same melody played in different keys as unique, which may not be desired. Transposing to a common key (e.g., C) will account for this, and may increase the variety of the Markov model.

Duration is measured by the Listener with millisecond resolution. Using absolute duration is not feasible, as small variations in duration will result in unique symbols for each note played. Rather, durations of notes need to be approximated to the nearest duration for a given tempo (e.g., a 64th note). The symbol for a note can then simply be the fractional value of the duration.

Modeling both the pitch and duration of a note can be achieved by treating each note as a tuple consisting of the note's pitch and duration. Treating notes in this manner, however, could be problematic for the Markov model. The number of possible symbols for a pitch-duration tuple will be much larger than the number of pitches or durations played. The number of singleton symbols generated will be very high, and the Markov model will require a large amount of provided material to have enough variety to generate unique responses. A more feasible approach to generating both pitch and duration for each note is to maintain two Mirrors—one to generate pitch values, and another to generate duration values. The sequences of generated pitches and durations can then be combined to produce a final result. As the lengths of sequences will likely be different, the longer sequence can simply be truncated to match the length of the shorter sequence.

For this investigation, responses were created by generating new pitch values from the Markov model. These replaced the pitch values of the user provided material, while the duration of the notes was simply copied by those provided by the user.

3.2.2 Markov model

The Markov model captures the transition probabilities in the provided user input (such as pitch, rhythm and timing information). This model maintains information on all the learning material provided, and provides the Mirror with an engine for rapidly generating musical phrases. When the Listener informs the Mirror to Learn, it provides the Mirror with the current contents of its local memory. This content is used to update the transition probabilities in the Markov model. The implied musical structure of the user's input is maintained and reinforced after each learning phase. This allows the user to build a dynamic "relationship" with the Mirror, which can be honed, developed and "learned" over time, similar to how a chamber group of musicians trains to react to each other's musical gestures in the process of rehearsing.

The Markov model used here describes the probabilities of transitioning to a particular symbol (e.g., pitch or duration) given a previous finite history of symbols. The *order* of the Markov model is the length of the history of symbols used to determine transition probabilities. For example, a 2nd order Markov model will consider the previous two symbols when determining the probabilities of the next symbol. This model has been used in natural language processing to generate and determine likelihoods of sentences [20]. Proper selection of the order of the Markov model is critical to ensure the model can generate a wide variety of material. If the order is too high, then most learned histories will have a single possible transition, resulting in the model simply echoing a previously learned melody verbatim. For our system, we used a 1st order Markov model, i.e., the probability of the next note value is based solely on the current note value.

The Markov model used needed to be able to both learn provided material and generate new material. To allow for these two modes of use, the statistics of symbol to symbol transitions were maintained as a count, rather than a probability. The model utilizes a hash map, where the keys to the map are prior histories (tuples) of symbols, and the values consisted of a list of the number of times a particular transition occurred given the history. To learn new material, an observed transition is simply incremented in the map. To generate material, the Markov model simply selects one of the possible transitions based on the probability of the transition occurring, given the history. In order to generate new material, it's necessary to provide an initial history of symbols to the model. For our system, we use the first note of the user provided material as an initial history to generate new material, or the note at the mutation point when mutating existing material. The number of notes used can be extended to account for higher order Markov models as well.

3.2.3 Genetic algorithm

The genetic algorithm component allows the Mirror to evolve a population of musical responses, and select the best response from this evolved population. Genetic algorithms iteratively improve a population of individual solutions (genotypes). The algorithm can be summarized by the following steps:

1. Create an initial population using the Markov model to generate candidate solutions.
2. Repeat the following for a maximum number of iterations:
    a. Determine the fitness of each individual in the population using the fitness function.
    b. Create a new population in the following manner:
        i. Copy the "elite" individuals (those with the best fitness) to the new population directly.
        ii. Create two child solutions by selecting two parent solutions and performing crossover.
        iii. For each child, randomly perform a mutation based on the mutation rate.
3. Return the individual with the best fitness as a response

The population size remains fixed for each iteration of the algorithm. The initial population is provided by the Markov model. The first note(s) of the user provided material is used as an initial history to generate responses from. By using the Markov model, each individual in the population will resemble the user's implied musical style. Each genotype represents a potential Mirror response. Each individual contains a variable-length genotype, avoiding placing any constraint on the length of the material generated. In our system, the genetic algorithm is used to search the space of all possible material generated by the Markov model. Crossover and mutation has been designed to ensure that resulting children could have been generated by the Markov model.

After an initial population is generated, the algorithm generates new populations each iteration using crossover and mutation. At each iteration, the elite individuals in the old population (i.e., those with the highest fitness) are copied directly into the new population without performing crossover or mutation. This ensures that the new population does not consist of individuals which are strictly worse than the previous population.

Children for the new population are first generated by performing crossover on two parents in the old population. The parents are selected using roulette wheel selection [4]. This method selects a parent at random based on each individual's fitness score. Those with higher fitness have a higher probability of being selected as a parent.

Once two parents are selected, two children are generated using crossover. In our approach, the crossover rate determines the probability of crossover being performed on the parents. In addition, it is possible that crossover cannot be performed, if the parents' genotypes do not share a common transition. If crossover is not performed, then the two children are simply copies of the parents. When crossover is performed, the first common transition shared between the two parents is found. The genotype of each parent is then split at this point, and the two parts are swapped between the parents, producing two children. An example of this is illustrated in Fig. 3. Since crossover is performed at a common transition, the resulting children could still have been generated by the Markov model. A child simply consists of a particular sequence in the Markov chain provided

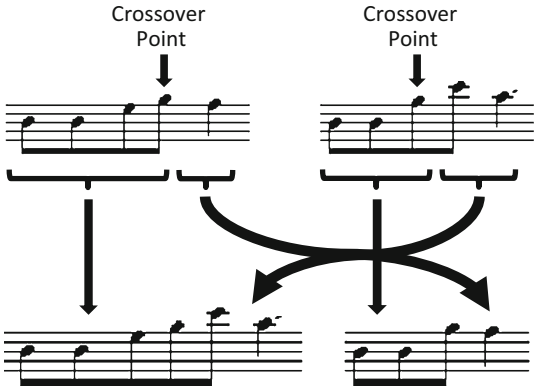

Fig. 3 Crossover operation used by the genetic algorithm

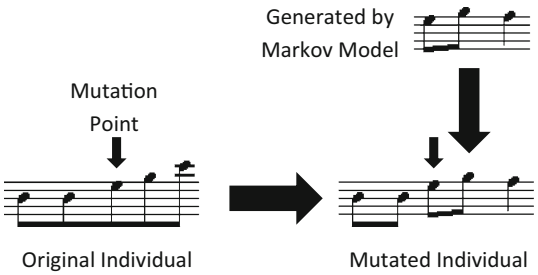

Fig. 4 Mutation operation used by genetic algorithm

by one parent, followed by the sequence provided by the other parent. This new sequence could have been generated by the Markov model had the transition at the crossover point changed to match that of the second parent.

After the new population has been generated, each child may be mutated based on a mutation rate. Mutation involves regenerating a portion of the genotype using the Markov model. A point in the genotype is selected at random. The note at this point is provided as an initial history for the Markov model to generate a new sequence of notes. This new sequence then replaces the existing sequence in the genotype at the mutation point. An example of this is shown in Fig. 4. The mutated child could have been originally generated by the Markov model, had the model performed a different transition at the mutation point.

At each generation, the evaluation function calculates the fitness of each individual by determining the aesthetic similarity between the individual's melody and the user input, as described in the next subsection. There are other possibilities (such as comparing each individual to material by other composers/performers, e.g., J.S. Bach or Thelonious Monk). This way, the system may create a composite Mirror, i.e., a mirror combining elements form the user and other (even non-living) musicians. The genetic algorithm seeks to maximize each individual's similarity (while avoiding identity) to the user input.

There are many adjustable parameters in a genetic algorithm, including the set of symbols (i.e., *alphabet*) used to construct individuals (i.e., *genotypes*); how to convert these individuals to "real-world" artifacts (i.e., *phenotypes*); the number of individuals per generation; what percent of individuals to preserve as the elite; how often to mutate an individual; how often to crossover two individuals; and so on. For our system, we used a population size of 30 and performed 25 iterations of the genetic algorithm per response request. These values ensure that a response can be generated within a few seconds, avoiding long pauses between a request and response. The top 10 % of a population were maintained as elite individuals between iterations. Crossover rate was set at 100 % (i.e., all non-elite individuals are used to generate novel children), and mutation rate was set at 10 %.

3.2.4 *Fitness function*

The fitness function is used to guide the evolution of individuals in the genetic algorithm. The goal of our system is to generate responses which are aesthetically similar to but not exactly the same as a user provided phrase. The fitness is based on comparing the power-law metrics, described in Sect. 2.4, for the user provided material and the generated material. The fitness function can compute a set of power-law metrics based on the music theoretic aspects described in Sect. 2.4 for each melody. This results in a vector of several hundred values for each melody. To compare an individual's melody with the user's melody, the mean-square error (MSE) of the user metrics and the individual metrics is calculated. Individuals with a low MSE are more similar to the user input than those with a high MSE.

Theoretically, it is possible to calculate all the metrics described in Sect. 2.4 for an individual, resulting in hundreds of metrics describing the individual. To increase the speed of this calculation, and the overall responsiveness of the system, we limited the metrics to those directly related to pitch and duration (Pitch, Pitch-Distance, Duration, Duration-Distance, and Pitch-Duration). These metrics produce responses of sufficient quality for our system.

The fitness of an individual is calculated from the individual's MSE, the MSE is first normalized by dividing by the number of metrics calculated. This result is then subtracted from 1. Any negative fitness is set to a very small value (1E−8) to ensure the roulette-wheel selection algorithm works properly. The fitness approaches 1 as an individual becomes more similar to the user provided melody. To avoid accidentally evolving the exact melody the user provided, any individual with an MSE of 0 is punished by setting the fitness to 0.0001. This ensures that

the individual can still be used for genetic material in future evolutions, but is not maintained as an elite.

4 Music design and Monterey Mirror

In musical terms, Monterey Mirror can be described as an interactive stochastic music generator, able to interact with human musical creativity and return intelligent and meaningful responses, based on variations within the user's input. As a material generator, a compositional environment designer, and a meta-instrument with interactive capabilities, Monterey Mirror is a powerful tool for contemporary music creation, targeted toward two main areas: compositional design and interactive improvisation.

The most obvious way of tapping into Monterey Mirror's possibilities are within the context of (structured or free) improvisation, by exploring its capacity of generating material much like a human performer would, as well as an intelligent meta-instrument, that can be learned and performed upon, much like a violin. A performer will quickly realize and take musical advantage of the system's ability to "listen" to a phrase and "answer" it with a meaningful musical response that can then be interacted with to create a musical dialog. It's very interesting to experiment with altering specific elements of the input material, in order to elicit a specific response from the system. With a little practice, the performer can easily develop a vocabulary of interactive gestures that can be integrated into a wide variety of performance situations in almost any genre. In improvisational performance settings, a performer can use Monterey Mirror to enrich his individual output and create the illusion of richer textures and more polyphony.

In its role as a meta-instrument, Monterey Mirror can be integrated within a musical ensemble of mixed media (acoustic and computer-generated), or could be employed as a stand-alone system for performance. Within the context of a mixed media ensemble, new possibilities of orchestration and ensemble interaction become available, by exploring the concept of *multiplicity*, where one performer can appear to be producing several strands of independently controlled material with the help of the system. In this scenario, it is not only the instrument that gets multiplied, but also the performer, along with her personal idiosyncrasies, training and approaches to technique, phrasing, and timbre.

Within the context of composition, Monterey Mirror can function as a planning template of considerable insight in exploring material and its possibilities within the musical space. It can allow a composer to tangibly blow up and time-stretch gestures and musical cells into complex textures of vast spatial and formal scope, while maintaining close relationship to the generating material, allowing for precise control, shaping and shifting power of the total musical space (e.g., see [1], pp. 37–42).

Monterey Mirror also presents new possibilities with visualizing and realizing formal designs, where the composer can maintain tight control of certain musical parameters (registral orientation and trajectories, temporal blocking and timbral identities), and simultaneously be relaxed, almost improvisational in nature with others (pitch patterns and rhythmic permutations), because the internal architecture of the system will preserve the integrity of the material that was fed into it. This allows for a new kind of aleatoric approach to form that is not possible otherwise.

5 Evaluation: composing and performing 2 × 2

2 × 2 is a composition intended to demonstrate some of Monterey Mirror's possibilities. It explores modes of interaction between two performers and two Monterey Mirror systems, within a carefully planned compositional space.

The setup (see Fig. 5) calls for two performers, playing MIDI controllers of any kind (keyboard, guitar, wind or mallet based), each connected via MIDI to a separate laptop running a Monterey Mirror system. This setup allows for the following modes of interaction: performer to performer, performer to Monterey Mirror system, and duo 1 (Performer 1 and Monterey Mirror 1) to duo 2 (Performer 2 and Monterey Mirror 2).

Both performer and system MIDI controller outputs play the same sound, and all audio outputs are routed through the same speaker setup, purposefully blurring the origin of the material to the listener, who cannot locate aurally whether sounds are coming from a human or are computer generated.

The form of the piece combines aleatoric/quasi-improvisational properties in the surface structure with precise large-scale form design.

As seen in Figs. 6 and 7, the piece is globally organized in 19 sections, each successive one introducing a new combination of interactivity, with a clear overall direction to a climactic point (section 9), where all possible combinations of material are superimposed, creating the most complex texture of the piece. The piece comes to a close

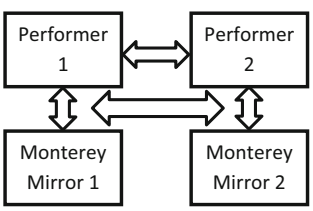

Fig. 5 2 × 2 Setup

Section	1	2	3	4	5	6	7	8	9
Performer 1	**Play A**		**Play B**				**Play D**	**Play D**	**Play A**
System 1			Listen A	**Play B**			Listen D	Listen D	**Play D** Listen A
Performer 2		**Play A**	**Play A**		**Play B**	**Play C**		**Play C**	**Play C**
System 2	Preload long B						**Play B**	**Play B**	**Play B** Listen C

Section	10	11	12	13	14	15	16	17	18	19
Performer 1		**Play C**		**Play D**		**Play C**	**Play B**	**Play A**	**Play A**	**Play A**
System 1	**Play D+A**	Listen C	**Play C**	Listen D		**Play D** Listen C	**Play C** Listen B	**Play B** Listen A	**Play A**	
Performer 2		**Play D**		**Play D**		**Play C**	**Play B**	**Play A**	**Play A**	**Play A**
System 2	**Play B+C**			Listen D	**Play D**	Listen C	**Play C** Listen B	**Play B** Listen A	**Play A**	

Fig. 6 2 × 2 Formal scheme

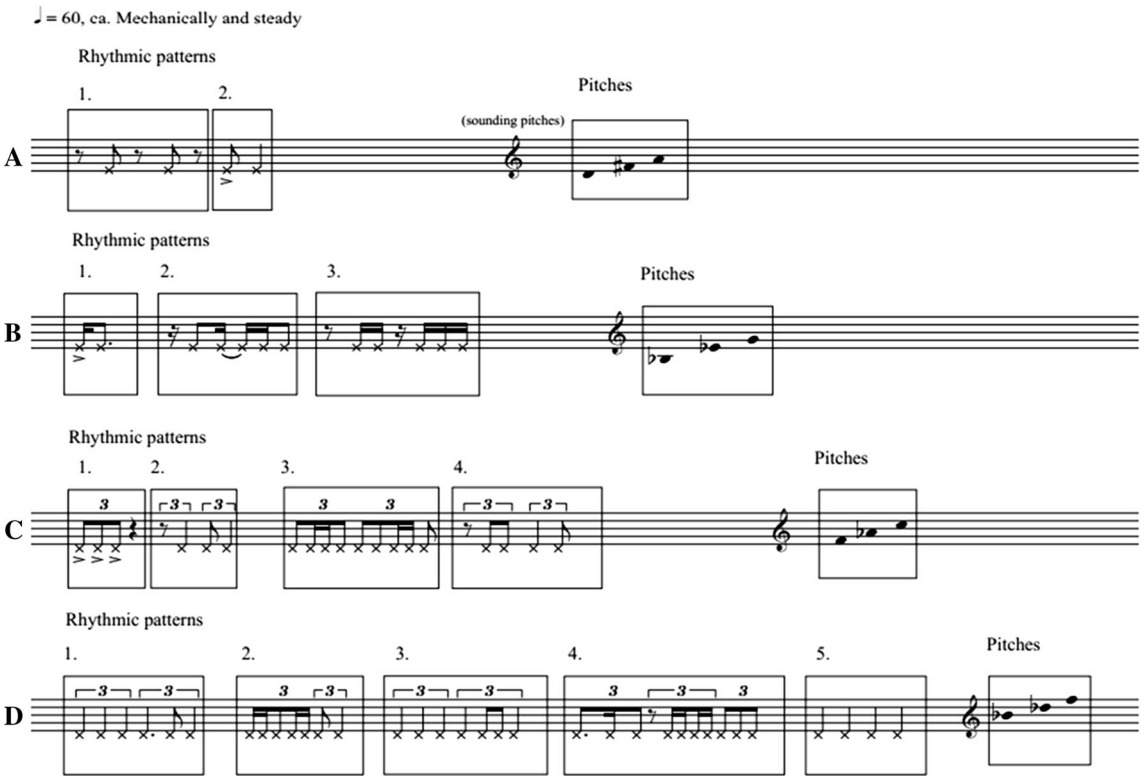

Fig. 7 2 × 2 Score

(section 19) with a coda that features interaction between the two performers only, with no additional output from the Monterey Mirror systems.

The musical material within each section is propagated through a "listening" process. One of the performers acts as an initiator of new material (sections 1, 3, 6, 7, 11, 13, 15–17), and the system connected to his individual MIDI controller is triggered to "listen" to his material, process it, and then later triggered to play material that is based on the section it has processed. As it plays processed material, it can be triggered to "listen" concurrently, and either combines new material with previously learned one, or "dump" older material, and start anew. The duration of each section is fluid and controlled by the performers (it is suggested to be between 15 and 25s). The Monterey Mirror systems will output material lasting about the same time as the one they listened to, therefore, the performers can anticipate the shape each section by interacting with each other and create smooth transitions moving forward.

5.1 Preparation and performance

2×2 received its world premiere in a Faculty Composers concert during the Monday Night Concert series at the College of Charleston on November 28, 2011. It was realized in a version for two MIDI keyboards, performed by College of Charleston piano artist certificate students Chee-Hang See and Amy Tan (see Fig. 8) [19]. 2×2 introduced several challenges to classically trained performers, requiring a particular set of reflexes during the performance. In addition to having to combine pitch and rhythm cells in an improvisatory manner that appears deliberate yet remains flexible and ever changing during the performance and advancing through the sections in a flowing manner, the two performers had an additional demanding task, that of shaping interactivity into form. This can be broken down to a series of actions that informs a decision making process and communication between the two performers in order to realize the formal scheme in a musically convincing way, receiving feedback from the output of the Monterey Mirror systems at the same time. After experimenting with different ways of combining pitches and rhythmic cells into intelligible phrases, the performers had to spent considerable time on their own "trying out" feeding the above different phrases into the Monterey Mirror system, to "learn" what kind of material the systems would generate in response. It was especially challenging to anticipate the length of the generated musical phrases, in order for the performers to time their next entrance against the Monterey Mirror material and against each other. In addition to the timing idiosyncrasies, the shaping of registral and timbral space needed to be considered, as it affected the formal shape of the performance.

Fig. 8 Chee-Hang See and Amy Tan performing 2×2 at the Monday Night Concert Series at the College of Charleston (Image courtesy of Wade Spees and The Post and Courier)

As far as timbre is concerned, the piece is built on the confluence of similar patterns coming from four different sources (two human, two Monterey Mirror systems), therefore the sounds generated needed to be as close to each other as possible, with similar attack patterns and minimal variability across extreme registers. Marimba sounds were chosen, as they were found to be quite similar when triggered by humans or Monterey Mirror systems, and seem to behave well in dense textures and across the vertical space.

As the piece progresses, the performers needed to be careful with saturating certain registers. This was challenging because at any given moment, the performer needed to assess what was happening momentarily, while planning ahead for the next sections, taking into account the behavior of the Monterey systems once pitches outside the range playing at the time were introduced. So, if for example, the performer introduced pitches in the lowermost register, she would expect the Monterey Mirror system to act similarly as well, not to mention that upon hearing activity in the new register, her human partner would likely react in some way as well, by either mirroring her action, or by counteracting it with pitches in a high register. And while this complication seemed like an added difficulty to the performers at first, by the time of the concert it allowed them to add more of their own input into the piece, by superimposing a registral trajectory to the performance, allowing them to have certain arrival points, as well as places they could feel they were "in control".

Despite the open form structure of the piece, which seemed puzzling to the performers at first, within a couple of weeks, they started to get quite comfortable with it, as they got more familiar with the responses that the Monterey Mirrors returned, and got to the point where they could impose their own manner of navigating through the formal

scheme to the point where they could actively influence when a section would begin or end, control dynamic shapes and texture density with remarkable ease and feel comfortable with the new interfaces. There were a few tweaks to the GUI that were made during the rehearsal process upon the performers' suggestion, mostly dealing with the size and physical placement of activity status indications. The concert performance was quite successful, and the audience was engaged and interested in finding out more about the design of the systems.

6 Discussion

2×2 is concerned with Monterey Mirror's interactive applications as applied to a duet of performers. There are endless possibilities to be further explored in future compositions, for which the system can be adjusted and tailored to accommodate other kinds of input material, pre-planned or not. There can be human initiation, or input from sources of any kind, including randomly generated events, or non-musical processes providing data, which is in turn applied to generate sound. Monterey Mirror would also work great for cite-specific sound installations, or sensor-based interactive systems.

Within a more traditional electro-acoustic composition environment, the system can be adapted to have triggers programmed to fire when they encounter a specific event (e.g., MIDI note) in its input, so that the performer can just perform off a traditional score without having to additionally trigger events, as the system will be "waiting" for a specific note as a trigger. MIDI controllers have also a lot of potential in similar scenarios, and elements like MIDI volume, panning, portamento, etc. can be "listened to" and incorporated into the compositional design.

The ability to operate a Monterey Mirror with a portable laptop setup and relatively little computing power can allow for larger ensembles incorporating several systems, multiplying the sonic possibilities, or even eliminating performers altogether and setting up multiple systems which can feed each other in a loop-like situation.

7 Conclusion and future work

Monterey Mirror is an interactive stochastic music generator based on Markov models, genetic algorithms, and power-law metrics. It combines the predictive power of Markov models with the innovative power of genetic algorithms, using power-law metrics for fitness evaluation. Unlinked other systems, the evolutionary process is confined within the search space defined by the Markov model. Initially, this might appear as a limitation. It would have been straightforward to include random mutation and crossover operations to generate "novel" material. However, given the real-time constraint of Monterey Mirror, it was a conscious decision to evolve material consistent with the Markov model. Since Markov models only capture local dependencies, they tend to generate output lacking the long-term structure or coherence of the input (training) material. However, once in a while the Markov model may "get lucky" and generate an output that exhibits such dependencies. It is precisely this possibility that the genetic algorithm exploits. By laboriously exploring the search space defined by the Markov model, it eventually will encounter material exhibiting longer-term dependencies. The fitness function ensures that such individuals are sought and rewarded, given that power-law metrics capture long-term dependencies in musical data [8].

The use of power-law metrics allows for evaluation of the generated musical phrases during performance. There are two distinct advantages to this. First, Monterey Mirror requires no training prior to performance, and does not require interactive evaluation of generated material. Second, because evaluation relies solely on the statistical proportions of the input material, the system is not limited to a certain style of music.

In musical terms, Monterey Mirror is able to interact with human musical creativity and return intelligent and meaningful responses, based on variations within the user's input. Both as a compositional environment, and as a meta-instrument with interactive capabilities, Monterey Mirror is a powerful tool for contemporary music creation, targeted toward two main areas: compositional design and interactive improvisation.

Acknowledgments This work has been supported in part by National Science Foundation (Grants IIS-0736480, IIS-0849499 and IIS-1049554). The first author would like to acknowledge the contribution of David Cope, Peter Elsie, Daniel Brown, and Michael O'Bannon in shaping the Monterey Mirror concept during the 2010 Workshop in Algorithmic Computer Music (WACM) at the University of California, Santa Cruz. We also would like to acknowledge the following for their contributions in formulating and evaluating power-law metrics: Thomas Zalonis, Brys Sepulveda, Patrick Roos, Dwight Krehbiel, Luca Pellicoro, Timothy Hirzel, Penousal Machado, and Juan Romero. The authors would like to thank the reviewers for their valuable comments and suggestions to improve the quality of the paper.

References

1. Xenakis I (2001) Formalized Music: Thought and Mathematics in Music. Pendagon Press, Hillsdale
2. Cage J (1961) Silence: lectures and writings of John Case. Wesleyan University Press, Middletown
3. Rabiner LR, Juang BH (1986) An introduction to hidden Markov models. IEEE ASSP Mag 3(1):4–16
4. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, MA

5. Biles JA (2013) Straight-ahead jazz with GenJam: a quick demonstration. In: Proceedings of 2nd International Workshop on Musical Metacreation (MUME 2013), AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE'13), Boston, pp 67–74
6. Cope D (2001) Virtual music: computer synthesis of musical style. MIT Press, Cambridge
7. Pachet F (2002) Playing with virtual musicians: the continuator in practice. IEEE Multimed 9(3):77–82
8. Manaris B, Roos P, Machado P, Krehbiel D, Pellicoro L, Romero J (2007) A corpus-based hybrid approach to music analysis and composition. In: Proceedings of the 22nd Conference on Artificial Intelligence, Vancouver, pp 839–845
9. Wiggins G, Papapoulos G, Phon-Amnuaisuk S, Tuson A (1999) Evolutionary methods for musical composition. In: Dubois DM (ed) International journal of computer anticipatory systems, vol 4. CHAOS, pp.312–325
10. Brown AR (2002) Opportunities for evolutionary music composition. In: Proceedings Australasian Computer Music Conference, Melbourne, pp 27–34
11. McCormack J (1996) Grammar-based music composition. In: Stocker R, Jelinek H, Burnota B, Bossomaier T (eds) Complex-systems 96: from local interactions to global phenomena. IOS Press, Amsterdam, pp 321–336
12. Burraston D, Edmonds E, Livingstone D, Miranda ER (2004) Cellular automata in MIDI based computer music. In: Proceedings of the International Computer Music Conference, pp 71–78
13. Manaris B, Romero J, Machado P, Krehbiel D, Hirzel T, Pharr W, Davis RB (2005) Zipf's law, music, classification and aesthetics. Comput Music J 29(1):55–69
14. Manaris B, Armstrong JR, Zalonis T, Krehbiel D (2010) Armonique: a framework for web audio archiving, searching, and metadata extraction. Int Assoc Sound Audiov Arch J 35:57–68
15. Manaris B, Machado P, McCauley C, Romero J, Krehbiel D (2005) Developing fitness functions for pleasant music: Zipf's law and interactive evolution system. In: EvoMUSART2005—3rd European Workshop on Evolutionary Music and Art, Lausanne, Switzerland, pp 298–507
16. Zipf GK (1949) Human behavior and the principle of least effort. Hafner Publishing Company, New York
17. Voss RF, Clarke J (1975) '1/f noise' in music and speech. Nature 258(5533):317–318
18. Schroeder M (1991) Fractals, chaos, power laws: minutes from an infinite paradise. W.H. Freeman, New York
19. Knich D, Real, artificial brains make magical music. In: The post and courier. http://tinyurl.com/musical-intelligence. Accessed 01 Nov 2013
20. Jurafsky D, Martin JH (2008) Speech and language processing: an introduction to natural language processing, computational linguistics and speech recognition. Prentice Hall, Upper Saddle River
21. Hsu KJ, Hsu AJ (1990) Fractal geometry of music. Proc Natl Acad Sci 87(3):938–941