
Anthony R. Burton*
and Tanya Vladimirova†

*Digital Integration Ltd.
Watchmoor Trade Centre
Camberley, Surrey GU15 3AJ, UK
tony@digint.co.uk

†Computer Systems Research Group
School of Electronic Engineering, Information
Technology, and Mathematics
University of Surrey
Guildford, Surrey GU2 5XH, UK
T.Vladimirova@ee.surrey.ac.uk

Generation of Musical Sequences with Genetic Techniques

Biologically inspired computational methods have recently attracted much research interest in the field of computer music. This group of methods is a subset of computational intelligence, and is represented by neural networks (NNs), genetic algorithms (GAs), and genetic programming (GP). A common feature of these methods is that they all mimic biological processes: NNs realize the evolutionary device of learning from experience, as humans and other animals do, while GAs and GP are based upon procedures that imitate the laws of natural selection.

Genetic algorithms have been shown to display distinct performance improvements compared to enumerative, calculus-based, and random searches of a given arbitrary search space (Goldberg 1989). This is achieved by combining aspects of these search methods to result in a "guided random" search. The genetic algorithm samples points throughout the search space for their worth, and is "blind" to any information regarding the search space apart from this measure of worth. This makes genetic search techniques more general and applicable to many search or optimization tasks, as long as an appropriate encoding scheme is employed.

By using a population of candidate solutions, rather than single individuals, an inherent parallelism in the search process is apparent. This is because the search for an optimum solution is "performed over genetic structures (building

blocks) that can represent a number of possible solutions" (Filho, Treleven, and Alippi 1994).

Genetic programming techniques expand on the versatility of GA techniques by evolving generations of functions, rather than representations of a single function. This is based upon the need to evolve computer programs to solve a problem, rather than evolving solutions to a given fixed problem (Koza 1992).

Computational intelligence techniques have been applied to musical problems across a wide range of subject areas, including algorithmic composition, artificial listening, musical cognition, and sound synthesis (Todd and Loy 1991; Balaban, Ebcioğlu, and Laske 1992), while GAs have been applied principally to the musical tasks of composition and synthesis. The search space for both of these tasks is potentially vast, considering the number of possible musical compositions and musical sounds. The use of stylistic or timbral constraints upon a composition or synthesis task reduces the size of the search space, yet the space still remains inefficiently large for traditional search techniques. Genetic techniques are able to perform well with large search spaces, owing to their inherent parallelism.

The aim of this article is to review how GAs and GP have been applied to musical tasks. The following section of this article introduces the background theory to genetic techniques. Next, an overview and hierarchy of the reviewed works are presented, followed by a more detailed description of each work.

Introduction to Genetic Algorithms and Genetic Programming

Genetic Algorithms

The first demonstration of the search and optimization possibilities of genetic algorithms is related to the application of evolutionary processes to binary strings (Holland 1975; Goldberg 1989). Additional work has been reported by Michalewicz (1992), Wasserman (1993), and Srinivas and Patnaik (1994). Using simple means of selection and transformation of individuals, genetic algorithms can iteratively increase the worth, or *fitness*, of the population as a whole. Combined with simple encodings to represent the search space, GAs are able to solve complex problems more quickly and efficiently than other search and optimization techniques. Owing to their high efficiency and performance compared with other techniques, GAs have gained considerable popularity as general-purpose, robust, search-and-optimization techniques (Davis 1987; Goldberg 1989; Michalewicz 1992).

In contrast to other search algorithms, GAs employ *populations* of candidate solutions (*individuals*). A population is initialized randomly as a number of binary strings of a certain length. The number of strings and the string length are determined according to the specific problem being solved. Each of these individual "chromosomes" is then evaluated for fitness—the measure of worth used by a GA to indicate and exploit areas of the search space in which desirable solutions are to be found.

All individuals in a population will fulfill the requirements of the task to a greater or lesser degree. Using a selection procedure that is more likely to select high-fitness individuals, a *mating pool* is created.

The first operation of the GA, *reproduction*, involves the selection of individuals from the mating pool, each with a probability in proportion to its fitness. The next operation, *crossover*, is then applied to these individuals in the mating pool. Pairs of individuals are selected (again at random)

and a point along their length—the crossover site—is selected, also randomly. Two new individuals are created by swapping the bits between individuals after the crossover site.

The crossover procedure is repeated for all the individuals in the mating pool. The third phase of the algorithm is *mutation*, a low-probability event whereby a single bit is changed in a string in the population. The object of this phase of the algorithm is to prevent important genetic material (information that would help the optimization process) from being lost irrecoverably (Goldberg 1989). Mutation also helps prevent premature convergence of the algorithm on a sub-optimal solution.

The algorithm stops when a certain termination condition is satisfied. This condition may be that the population fitness has reached the maximum possible value, that the population fitness has not increased for a given number of iterations, that the number of iterations carried out so far has reached a predetermined limit, or "you run out of computer time, memory, patience, or funding" (Biles 1994).

As the GA iterates, inspection of the increasing number of strings in the population will show a pattern. This will be a pattern of bits in each string that causes the string to have a high fitness value and hence be more likely to advance the optimization of the function. It has been shown that strings with a certain template will be more effective in the optimization process (Goldberg 1989); this template is called a *schema*. Schemata tend to be more persistent in the population than actual strings (Chester 1993), and tend to indicate high-fitness regions of the search space.

Genetic Programming

Genetic programming techniques (Koza 1992) allow for a certain degree of relaxation of the constraints upon the search space imposed by GAs by genetically producing the functions themselves that will solve a given problem. Using GP, it is possible to solve problems where the search space is unknown in advance: GP applications allow genetically evolved functions to adapt to the search space, depending on which of these functions

solves the given problem best. Again, individuals will have varying degrees of fitness, determined by processing the function several times under a variety of conditions, to calculate the average fitness of the function. Also, the genetic operators of selection, crossover, and mutation are applied to GP techniques, but it is the functions and their arguments that are altered in these operations.

Like GA techniques, the specific way in which any GP approach is applied to solving a problem is highly dependent upon the problem itself. One major difference between GA and GP techniques is that the length of an individual in a GP approach is not constant, as it would most often be in common GA applications. The selection, combination, and transformation operations performed by GP techniques are in principle the same as those found in GA applications. It is the building blocks that make up the individuals, and the way in which these building blocks combine to form the individual, that differ.

Generating Musical Sequences with Genetic Algorithms

Algorithmic composition is a process that involves the generation of musical phrases, themes, or whole pieces by computational methods. There have been many artificial, computer-based methods employed in the creation of musically meaningful works. These include the neural networks to generate musical fragments based upon representative training data, expert system methods to employ rule bases to generate works, and more esoteric applications that deduce musical phrases from random events (Baggi 1991; Camurri et al. 1991; Johnson 1991; Todd and Loy 1991; Balaban, Ebcioğlu, and Laske 1992; Desain and Honing 1992). These methods often have some inherent limitation in the scope of their output: neural methods tend to be limited by their training data, while expert system methods are limited by the rules they employ. Genetic techniques allow a much greater flexibility of application and scope of output due to their "blindness" to the application and by the exploitation of individuals of high fitness.

In applying genetic algorithms to musical composition problems, there are three main areas that must be considered to ensure that a meaningful and efficient algorithm is employed. The first of these is the search domain. Owing to the vast combinatorial possibilities of individual notes in time, rhythm, harmony, and melody, the "search space" for composition is essentially unlimited. It would be overly ambitious to assume that any algorithmic composition process would be able to compose original music from scratch, so it is necessary to limit the size of the search space by imposing constraints upon the decision-making process. This limits the number of possible combinations of musical objects represented by the population of a GA. For example, in the case of a GA that is being used to generate a melody line according to some arbitrary fitness regime, the search domain could be restricted to a certain range of notes from a particular key. By limiting the range and choice of notes in this way, any composition would remain within the chosen aesthetic and musical limits.

The next area of consideration in designing a GA for music composition is the input representation. Two important factors that affect how any algorithmic composer operates are those of musical "knowledge" and rule representation. Musical knowledge is the information used by the compositional system to define pitch, rhythm, meter, and other musical building blocks. Rule representation is a set of rules that defines how a composition can evolve with successive iterations of the algorithmic composer; this is a most important factor when applying genetic algorithms to composition.

Fitness evaluation is the third critical area of importance in GA design. The rules applied to a representation of a musical source have to produce some measure of how well the source obeys the given rules. In a genetic environment, this role is fulfilled by the fitness function, which determines whether a particular individual "survives" to the next generation. It is essential that the fitness computation indicate sources that adhere better to the rule representations.

It is possible to subdivide each GA application according to the type of fitness evaluation used.

Figure 1. Initial musical phrase for melodic development process.

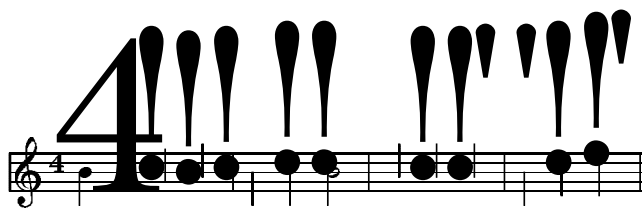
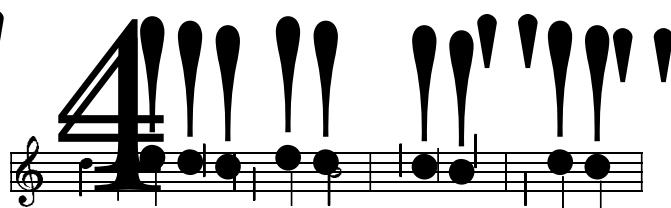


Figure 2. Example output of melodic development process.



The fitness-evaluation methods into which the applications analyzed here can be subdivided are deterministic, formalistic, user-determined, and neural. Deterministic fitness-evaluation methods use some mathematical relationship to transform a musical phrase into a fitness evaluation. Formalistic fitness-evaluation methods assign high worth to those musical phrases that adhere to existing rules of musical theory. Applications that employ user-determined fitness evaluation have been developed in cases where it was thought unfeasible or impractical to implement a rule-based evaluation method. These systems depend on the judgment of the user to assign fitness to individuals by way of some interactive interface. Finally, applications employing a neural fitness-evaluation system implement some type of neural network architecture trained to assign high fitness to those patterns that display the same or similar attributes to training data.

Deterministic Fitness Function Applications

In this section, applications that employ deterministic fitness functions are reviewed. These applications use a mathematical function of the individual encoding of a musical phrase to give the fitness of the individual. The examples discussed below determine fitness as a function of the number of common elements—an example of a pattern-matching scheme.

Melodic Development

One problem central to algorithmic composition is that of artificially generating variations based on preexisting music. This is the essence of thematic development and improvisation. This problem involves taking a certain musical phrase and applying specific rules to the phrase that alter it in some way. Ralley (1995) suggested a genetic composition

system that uses a method of data reduction to compromise two extremes of individual fitness assignment. These extremes are user-assigned fitness, which is time consuming and context dependent; and deterministic fitness assignment, which may prevent any stylistic progression in the results due to the tight constraints of the search space.

The representation used to describe a note phrase consists of two parts. The first defines the "alphabet" from which the notes of the phrase will be taken—essentially a representation of the key signature and starting note of the phrase. The second part of the representation is a series of integers that indicate the intervals between successive pitches. This type of representation allows easy modification of note patterns, such as transposition, inversion, and retrogression.

Figure 2 shows a sample output of the process performed by Ralley's system when applied to a phrase of music (based on "Mary Had a Little Lamb," shown in Figure 1).

The usefulness of algorithmic composition systems such as this one largely depends upon the type of composition required. In fact, Ralley raises the question of whether the "evolved" compositions are indeed an improvement over the original work. If there is some definite requirement of note pattern upon which the algorithm is to converge—that is, a certain melody is desired—then the system is largely unsuccessful. However, this system was successful at generating useful melodies from the search space with no specific goal in mind, as long as the subjectivity of the user was the only fitness-evaluation process.

Thematic Bridging

The problem of thematic bridging using genetic algorithm composition was investigated by Horner and Goldberg (1991). Thematic bridging is the process of transforming one musical statement into another over a specified interval of time. This bridging

process is considered a modification of the musical elements in the statement preceding the bridge.

Given an initial musical statement, the final desired musical statement, and the number of notes required to bridge the two, it is possible to create an appropriate bridge using genetic techniques. Two fitness measures are used; the first is a measure of the degree to which any generated note pattern matches the desired target musical statement. The second fitness measure is the degree to which the two statements match in duration.

Each individual represents a series of operations that transforms the initial note pattern into the target pattern. Examples of these operations are note deletion, addition, or mutation; pattern rotation; and note exchange between patterns. A specific individual contains the instructions for creating a passage of notes that bridges the initial and final patterns. Applying standard genetic operators such as binary tournament selection and single-point crossover, it is possible to create patterns of musical bridges. The methods employed here can be used in other aspects of composition, such as timbral manipulation.

Formalistic Fitness Function Applications

Applications that use formalistic rule-based fitness functions include the composition of Baroque music employing rules based on the stylistic interpretation of this music (McIntyre 1994); a system that generates chord sequences based upon the rules of how chords should be spelled and by how much each voice is allowed to change in pitch between chords (Horner and Ayers 1995); a system that evolves the melody, rhythm, and dynamics of a musical phrase using rules based upon user-defined requirements (Thywissen 1996); and a system that co-evolves composers and evaluation methods using note-transition probability tables to determine aesthetic properties (Werner and Todd 1997). These systems measure how much a genetically produced individual follows the particular rules of the application to assign fitness to that individual.

It is possible to apply genetic algorithms to create four-part Baroque harmonies using a fitness measure that indicates how well genetically produced chords follow the traditional rules of the style. This task was investigated by McIntyre (1994), where genetic algorithms were used to generate three extra voices to accompany a given melody line.

The measure of fitness is determined for a composition as a whole, and is based upon several attributes of the piece. These attributes include how chords are built from individual notes, how the chords are spaced in a 29-note interval, and how one chord moves to another. Some of these rules are more important to the formation of a piece than others. For example, good motion between chords is irrelevant if the spelling of a chord is initially poor. As with the thematic bridging problem described above, a multitiered fitness regime is used. First, the harmonic fitness of the composition is determined; this is based on how chords are spelled and the spacing of the voices within the chord. If this fitness is above 85 percent of the maximum possible fitness, then the next tier of fitness is calculated; if not, the composition is discarded. The second tier of fitness is related to harmonic motion and interest. Similarly, if this measure of fitness is less than 85 percent of the maximum, the composition is discarded. The final measure of fitness is related to the smoothness and resolution of chord changes—a function of the intervals between successive notes of the same voice and the types of chord involved in certain progressions. The genetic operators employed in this case are the standard types of reproduction, crossover, and mutation as used in Goldberg's SGA.

Using the three-tiered method of fitness determination, the algorithm converges on populations of greater fitness faster than would otherwise occur. The results obtained from these experiments showed that applying genetic algorithms to this nonlinear problem proved to be very successful. The same methods could be easily applied to other styles of music, assuming that rules governing the melodic and harmonic changes associated with that style could be formalized.

Harmonization of Musical Progressions

Another method of harmonizing chord progressions by a genetic algorithm based upon certain formalized rules was described by Horner and Ayers (1995). The rules can be seen as constraints upon how the search space can be explored, and can be divided into two types. The first constraint group defines how individual chords can be voiced, and the second constraint group defines how voices are allowed to change from one chord to the next.

The process of harmonizing a chord progression takes place in two steps, one for each of these constraint groups. Once a satisfactory set of chords has been generated by an enumerative method, a genetic algorithm is used to satisfy the next set of constraints: how the voices of one chord move to the next.

The input to the genetic algorithm is a number of individuals, each a combination of the chord set generated by satisfying the first constraint group. Each individual fulfills the requirements of the voice-leading constraints to a certain degree, and it is this constraint fulfillment that provides the fitness measure for each individual. Specifically, the fitness measure is defined as the number of rules in the constraint group that are not satisfied. This implies that individuals with fitness values closer to zero fulfill the rules to a higher degree.

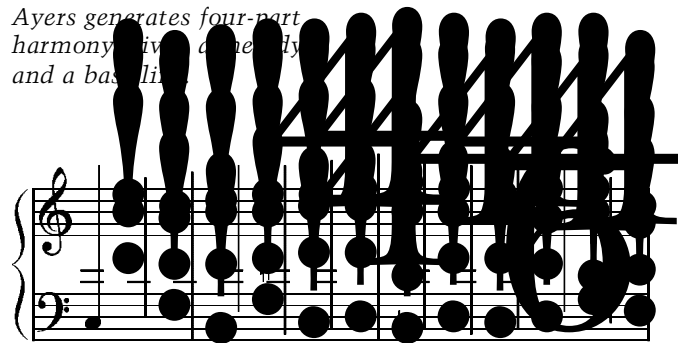
Figure 3 demonstrates the system's ability to fill in the inner voices of a chord sequence, given a melody and a bass line.

Hierarchical Musical Evolution by Compositional Structure Manipulation

Thywissen (1996) described an environment for the evolution of musical phrases in which a GA was used to "guide the composer through the search space of possible compositions." This compositional tool, called GeNotator, treats a composition as a set of distinct parts, which all contribute to the overall musical structure. Examples of these parts include melody, rhythm, harmony, and instrumentation.

Like the genetic compositional applications detailed by Ralley (1995) and Horowitz (1994), this

Figure 3. A genetic algorithm designed by Andrew Horner and Lydia Ayers generates four-part harmony given a melody and a bass line.



application uses a rule-based fitness-evaluation process to evaluate candidate melodies, chord sequences, and rhythm patterns. Additionally, a subjective fitness-evaluation procedure allows the system to direct the search to compositions similar to those previously evaluated by the user as being of high fitness.

The composition representation used by GeNotator consists of a two-layer hierarchy: the *phrase* and the *system*. A phrase consists of a list of musical events, e.g., pitched notes or rhythm patterns. The musical event list can represent more than one event at any time, which allows chords to be included in the representation. A system consists of a sequence of phrases and represents the composition as a whole.

Two evolutionary devices are employed: the *phrase-evolver* and the *system-evolver*. These allow compositions to be approached from two perspectives, either evolving note sequences to form new phrases, or combining existing phrases genetically to produce new compositions.

The material used to initialize the evolutionary process consists of both randomly generated representations of phrases and systems, and representations of phrases and systems that are known to be useful to the user. This implies that some of this system's output should conform to the user's tastes.

The fitness of a phrase is determined by how well a candidate solution fulfills the rules dictating melodic structure, note durations, note densities, and interval constraints. Other fitness rules are also implemented for the evolution of harmonic and rhythmic structures. Systems of phrases are evolved using functions that manipulate the content of the system in a musically meaningful way, e.g., transposition or inversion.

This genetic compositional system is capable of producing "interesting music," but requires "balancing [of] starting conditions with a degree of

form space bounding" (Thywissen 1996). This implies that the mixture of randomly generated and user-defined material that initiates the evolution requires fine tuning to produce satisfactory results, which itself leads to tight constraints upon the output of the evolution.

Co-Evolution of Composers and Evaluators

Werner and Todd (1997) modeled populations of composers and song evaluators in his investigation of the diversity of communication signals displayed by a species over time and at each instant in its evolution. This involves the *co-evolution* of two sub-populations, where the characteristics of one sub-population affect subsequent sub-populations of the other.

The model delineates two gendered sub-populations of songbirds. Here, each "male" sub-population has a single "song" that it uses in an attempt to attract a "female" mate.

The male is represented exclusively by his song, a sequence of 32 notes selected from a 2-octave chromatic scale. Each female has her own means of evaluating the male's song, and this is represented by an evaluation matrix that gives the expectational probability value of one note, given that it is preceded by any other. This 24×24 -element table is used to assess each male's song, based upon sequential note-by-note transitions. As each individual is represented by a male/female pair, the genetic information stored by an individual contains both the song and the transition-table representations. The male is initialized randomly, and the first generation of females is initialized using information regarding note transitions derived from simple folk songs. Although this implies that the evaluation system will use some human-based note-transition judgments, the genetic modification of the female-song evaluators may make the resulting songs unmusical. However, it is the selection and evaluation methods that are under investigation, rather than the musicality of the generated songs.

The song-assessment process is carried out by the female in one of three ways. The first way is a *local transition preference*, whereby songs with

many expected individual transitions are rated highly. The second is a *global transition preference*, in which a transition table is generated for each song and compared to the expectational transition table; songs that result in high degrees of similarity between these two tables are rated highly. The third assessment method is called *surprise preference*. This gives a high rating to songs that *violate* the expectational note transitions.

The selection process differs from other selection methods in genetic music applications in that sexual selection is being performed instead of natural selection. This means that the requirements of individuals are being maximized, rather than the ability of individuals to survive in a modeled environment.

The results of these investigations showed that replacing natural selection with sexual selection results in an increased diversity in the communication methods simulated by the system. This increased diversity is maintained by the use of the surprise-preference evaluation method. Whereas the "realism" of the composed songs of this application is not of much interest, the means of evaluation and representation are important to the study of how populations communicate and evolve.

User-Determined Fitness-Function Applications

Because music composition is a highly subjective task, some applications rely on the user to assign fitness to individuals or to stipulate the type or style of output. At the most basic level, the user is required to enter a fitness response for each individual from a computer keyboard (Biles 1994). An improvement on this was suggested by averaging the response from an audience (Biles 1995). Other applications required the user to determine how various parameters of individuals were judged (Horowitz 1994), or used material generated from phrases provided by the user as building blocks (Jacob 1995).

Rhythm Generation

Horowitz (1994) proposed a system that employs interactive genetic algorithms to generate rhythmic

patterns using subjective fitness evaluation. The representation employed was a series of notes and rests occurring on specific beat intervals, each series being 1 measure in length. The user determined the fitness of each individual of a randomly generated initial population; fitness was assigned depending on the degree to which the individual rhythm pattern matched the user's tastes and desired result. Using standard reproduction, crossover, and mutation operations, new populations were generated that approached the desired requirements.

Additionally, objective fitness functions allowed the automatic evolution of populations of rhythmic patterns. These functions included such parameters as syncopation, beat density, and beat repetition. The user set target levels for the selected fitness functions, and judged each generation of rhythms as they were produced. For example, if syncopated sixteenth notes were desired, the user would specify high target values for the syncopation fitness criteria and the beat-density fitness criteria. Another genetic algorithm was used to generate populations of parameters employed in this process. This made the search faster, because the user could judge the fitness of populations without having to explicitly modify values for the objective fitness functions as each generation evolved. The use of several genetic algorithms working in parallel in this way vastly increased the convergence rate of the population toward rhythmic patterns that satisfied the user's requirements.

Horowitz (1994) also proposed a more sophisticated system whereby multitimbral rhythmic patterns were generated genetically. This involved the generation of multiple voice rhythms in a manner similar to how a human drummer would create drum patterns using the snare drum, bass drum, hi-hat, and other percussion. Representations of rhythms were used that defined the activity, syncopation, and accent structure of a voice such that each individual of a population represented a collection of voices with a specific texture. An interesting crossover procedure was employed whereby similar voices swapped information to create new individuals, rather than information from the same voices being swapped between existing individuals.

Genetic Jazz Improvisation

Biles (1994) used genetic algorithms as a means of creating spontaneous improvised music with a system called GenJam. This system is described as analogous to a student musician playing with experienced musicians who offer encouragement when the student plays well and respond in a less positive manner when the student performs poorly. The system creates jazz-like solo improvisations from information relating to the chord structure of a piece of music. This hypothetical student's performance is judged by a "mentor," the user of the system.

Mr. Biles emphasizes the importance of using a good input representation for the genetic algorithm to perform successfully. The format used for genotype representation was a cooperating, two-level, position-based binary scheme. This scheme differs from a representation that would be used in a simple genetic algorithm, in two ways. First, GenJam employs two populations rather than one, representing measures and phrases respectively. A phrase consists of four measures, and the information contained in a phrase-population individual maps to the index of a measure. The information contained in a measure-population individual maps a sequence of MIDI events. The second difference between GenJam and simple genetic algorithms is that all of both populations, measures and phrases, are used to build a solo, not just a single fittest individual. In this way, GenJam creates a library of usable phrases, rather than the "perfect" solo to accompany a certain chord progression.

Each phrase individual consists of the indices of the four measures that constitute that phrase, and associated with the phrase is the fitness as determined by the mentor. Each measure individual consists of eight encoded representations of either a note, a rest, or a "hold." The rest infers that no note is played, and the hold infers that the previous note or rest is maintained for another beat. Each measure is made up of 8 concatenated encodings of a possible 16 events (14 possible notes plus the rest and hold events). Each of the 16 events can be uniquely identified by a 4-bit code; therefore, each measure is comprised of a 64-bit code.

Depending on the chord progression, a different set of notes from the 14 available must be chosen to sound contextually correct. For example, if a chord in the progression is a major seventh chord as opposed to a dominant seventh, then the respective seventh note of the scale will be a semitone higher. Given a certain chord type, GenJam will always select the right note from those available, so no "wrong" notes will be played.

The fitness of each phrase and measure is not computed or determined from how close the composed sequence matches any musical theory, but rather from how much the mentor likes the sequence. Mr. Biles states that he "decided to put off the search for an algorithm that implements 'I Know What I Like,' and use myself as the fitness function."

Although this method seems simpler than algorithmic fitness computation, the rhythmic pattern-generation process is slowed immensely due to the time it takes for the mentor to listen to and judge each individual in sequence.

Unlike simple genetic algorithms, the mutation operations used in this application are more complicated than a single bit inversion. They consist of operations that mutate measure individuals in a musically meaningful way, including such operations as note-order reversal, note-value inversion, and note transposition. A similar set of mutation operations exists for phrase mutations. In experiments, GenJam was able to generate solo improvisational material for a variety of jazz standards in different song structures, with musically interesting results. Certain improvements are required, most importantly a solution to the fitness bottleneck created by having to assign fitness to individuals sequentially. Mr. Biles addresses this problem by suggesting an alternative means of fitness evaluation, which will be described in the "neural fitness" section.

Motivic Genetic Composition

Most genetic composition systems address the problem of a large search space by applying constraints to how this space can be explored by the genetic algorithm. Jacob (1995) proposed an alter-

native to this, allowing the algorithm to deal with larger "building blocks." Rather than building a melody from individual notes, it built them from previously constructed musical phrases that had already been assessed as possessing aesthetic value. This method was applied to an artificial compositional system that used three separate modules in its composition process. These modules were the "composer," the "ear," and the "arranger," which respectively carried out raw music generation, aesthetic evaluation, and construction of compositions.

Initially, the user provided a number of primary motives, which were used by the composer to genetically generate similar musical statements. The user then evaluated the fitness of each of these statements. Next, these motives were concatenated to form longer musical phrases, which were then presented to the ear for further fitness evaluation. Any phrase that contained invalid harmonic or tonal variations was discarded. These individuals were evolved genetically by presenting the user with randomly generated ear critics, to which the user assigned a fitness, depending on how well that critic matched the user's own personal musical judgments. The final module, the arranger, took phrases that the ear had judged to be acceptable according to the user's tastes, and constructed different combinations of these phrases. Each of these combinations was then assessed for optimal aesthetic worth; this final combinatorial optimization process is analogous to the "traveling salesman" problem, a task to which genetic techniques are well suited.

A selection of the results obtained by B. L. Jacob from the variations composition system can be viewed on the World Wide Web at http://www.ee.umd.edu/~blj/algorithmic_composition/.

Neural Fitness-Function Applications

The use of neural networks as fitness evaluators enables the user to train the network to recognize individuals that are a close match to classes of patterns as defined by the training data. Neural networks have been applied to a genetic system that

generates rhythmic patterns, where the neural network has been trained to distinguish "good" rhythms from "bad" (Gibson and Byrne 1991). Another application, which follows on from the user/audience reaction system mentioned above, attempts to model genetically improvised music statistically in order to construct a reliable neural fitness evaluator (Biles, Anderson, and Loggi 1996).

Composition of Short Rhythms and Melodies

Gibson and Byrne (1991) attempted to design a system that produces short compositions using diatonic, four-part Western harmony. The composition domain space was simplified based upon a particular composer's style, by limiting the result in length, tonality, and harmony. The length limitation ensured that all resulting compositions were of the same duration; the tonality limitation prevented changes in key signature; and the harmonic limitation constrained the four-part harmony to the tonic, dominant, and sub-dominant chords of the key.

A neural network was used to classify exemplar four-beat rhythms using examples of "good" and "bad" rhythms, as judged by the authors. Future rhythms could be assigned a fitness value according to how close it was to a "good" or a "bad" rhythm. Next, the neural network was used to guide the genetic algorithm in a search for "good" combinations of rhythms. Each possible combination of four-beat rhythms was assigned a fitness value by the neural network. A rhythm similar to the "good" rhythms was given a high fitness value, and a rhythm similar to the "bad" rhythms received a low fitness value.

The genetic process started by randomly generating a population of four rhythms, each 4 bits long. Each rhythm was assigned a fitness value generated by the neural network. Reproduction and crossover were carried out in the same way as by Goldberg's SGA, and so subsequent iterations of the algorithm would provide more "good" strings, and hence a "good" composition. Once a four-beat rhythm had been developed in this way, a melody was created to accompany it. Two neural networks were used to accomplish this; one that

determined pitch separation between subsequent notes, and one that controlled the overall harmonic structure.

By using genetic algorithms and neural networks together, short musical compositions could be created with some success. However, owing to the constraints described above, the resulting compositions were not very musically adventurous. Longer compositions could be created by using multiple genetic algorithms with neural networks having different structures, and the tonal and harmonic structure could be expanded by using a more sophisticated input representation.

Methods of Reducing Fitness-Bottleneck Effects

As mentioned in the previous section, a means of fitness evaluation was described by J. A. Biles and co-workers to reduce the fitness-bottleneck problem encountered in genetic systems that employ user-evaluated fitness measures. This neural network augments the human fitness-determination process (Biles, Anderson, and Loggi 1996).

The neural fitness function was designed to act as a filter to prevent those individuals with low musical content being presented to the mentor for evaluation. This allows populations of fitter individuals to be generated earlier in the evolutionary process. After several attempts of modeling fitness computation using generalization techniques, no neural model consistently assigned the same fitness level to individuals as would a human mentor. On several occasions, it was found that individuals with almost identical chromosomes would be assigned markedly different fitness values.

One last suggestion for reducing the effect of the fitness bottleneck is to use several mentors simultaneously. Whereas the original GenJam system relied on interaction from one mentor to indicate fitness, the next proposed system used multiple concurrent mentors to indicate fitness. This was achieved by using either a visual feedback method or a secret electronic push-button method to relay fitness information to the GenJam operator. However, the mentors in these cases were unable to provide a consistent feedback of fitness.

Compositional Applications of Genetic Programming

The compositional applications of GP examined here use various types of fitness evaluation—generally mathematical or parametric functions—to generate melodies. Fitness is evaluated in one of three ways: (1) according to the error between the output of each individual function and some predetermined target result (Laine and Kuuskankare 1994), (2) by comparison with a database of stylistically similar works (Spector and Alpern 1994), or (3) by the output of a neural network trained with a selection of "good" and "bad" melodies (Spector and Alpern 1995).

Composition by Function Manipulation

Genetic programming was applied to the composition process by Laine and Kuuskankare (1994), whose aim was to generate simple single-voice melodic patterns using mathematical representations of how the melody changes with time. A melody line is considered as a discrete-time function, the numerical value of which indicates the note value at any given time point. For example, a melody that ascends one scale step with each time step can be represented by a simple linear function such as

$$f(x_1) = f(x_1 - 1) + 1. \quad (1)$$

Similarly, a pattern that rises and then falls could be represented by a scaled or mapped version of a trigonometric function such as

$$f(x) = \sin(x). \quad (2)$$

More complex musical statements can be realized by using nonlinear mathematical functions and chaotic bifurcation functions. It is possible to apply the principles of genetic evolution to these mathematical functions to generate output that is stylistically similar to the initial population.

The fitness of each function that describes a musical statement is, in this case, determined by an error measurement that requires minimization. This error is determined by calculating the differ-

ence between each successive function in a representation of a musical statement, and a target value previously determined for this particular function. Individuals are selected proportionally to this fitness measure for crossover and mutation. An elitist selection process is also carried out in this application: a certain percentage (10 percent, in this case) of the population is copied to the next generation unaffected by reproduction, crossover, or mutation. The crossover operation used here is essentially the same as in a standard genetic algorithm, but applied to a mathematical function. Similarly, mutation affects a single part of the mathematical function.

This type of mutation replaces one "branch" of a function with another one that is either selected randomly or chosen from a similarly generated function. Using this method to generate melodies produces results that resemble the original population material, but as genetic processes have no knowledge of the domain to which they are applied, there is little musical interest in the output. Another drawback to this means of composition is that the production of long pieces is difficult, due to the amount of individual functions required to model a long phrase. Similarly, if longer musical statements are represented by functions, then the size of each individual becomes unwieldy as the mathematical function required to model the statement becomes more complex.

Composition by Genetic Programming Using a Web Interface for Fitness Evaluation

Putnam (1994) applied GP techniques to the task of generating note sequences by using a parametric note representation that included information regarding note amplitude, frequency, duration, and the delay until the next note. Individual fitness values were assessed by sequential user evaluation: a unique interface was employed whereby anyone could listen to and judge the sample note sequences via the World Wide Web. In order to preserve the interest of the assessment audience, the population was evolved for 100 generations before presentation for fitness assessment. This helped to suppress transient behavior not typical

of the steady-state performance of the system. Still, the evolution of musically interesting compositions was slow and dissatisfying to the author.

One reason that may explain this is the convergent nature of the genetic programming technique used, especially in small populations of individuals of small function length, such as those used in this application. Another reason stated for this was discovered while applying this genetic programming system to an image-generation problem. It was discovered that not only is it important to use those individuals with aesthetic worth in future generations, but also those with lower fitness values that include potentially useful sub-patterns that do not appear in individuals of higher fitness.

Use of Existing Works as Fitness Evaluators

One major issue facing all designers of algorithmic composition systems is that of how to judge the output of their system. Several quality-assessment methods have been applied to various types of systems that generate different works of art—not just music—and each has its own advantages. Spector and Alpern (1994) examined these assessment methods and suggested one that may be more applicable to genetic composition systems.

Assessment methods can generally be divided into one of two groups: those that rely on the judgments of the users of the system, and those that rely on formalized evaluation. However, there are disadvantages associated with each of these methods.

A more appropriate type of assessment method is suggested based upon those works that have been previously judged to be of artistic merit. It follows that the artificially produced artworks are stylistically similar to these aesthetically meritorious works. Knowledge of what makes an artwork possess high aesthetic worth is not required; it is sufficient to be able to create a work of art by using artificial methods that meet the criteria applied to the artwork's value assessment.

The justification for this type of assessment is that when evaluating any art, it is impossible to do so without considering it in the context of previous work from the same and similar artists. A compositional system that uses genetic program-

ming is described that incorporates a fitness-evaluation process based upon this artwork-assessment method. The task addressed by this system is that of generation of 4-measure melodies in response to another 4-measure melody. This is a type of "call-and-response" playing, common in jazz improvisation. This genetic "bebop" artist works with functions, rather than encoded versions of individuals. It is the output of these functions that are the individual musical artworks presented for critical assessment.

The initial population is a number of randomly generated functions that operate on an input, which is a numerical representation of a sequence of notes. Each function is one means of altering the note sequence in some way, such as a transposition, inversion, or extension of the sequence. Each individual will be a different combination of functions which in turn will produce different variations of the melody—each with a different fitness value.

The fitness of each individual is assigned as a result of a number of comparisons—called critical criteria—between tonal and rhythmic aspects of the generated material, and a library of previous artworks of an artist of a similar style. In this case, the artist used as a criterion was the jazz saxophonist Charlie Parker. Selection, crossover, and mutation are then applied to the fitter artist functions to generate new artists. This system was able to produce musical results that were realistic in the context of the phrases used as the "call" part of the call-and-response improvisation idiom.

Use of a Neural Fitness Evaluator to Detect Musical Relevance

An improvement to the work by Spector and Alpern (1994) was achieved by using more information from the music as a whole produced for artistic judgment, rather than just the tonal and rhythmic comparisons used so far. Fitness evaluation is based upon a three-layered neural network that deduces musically relevant information in the generated compositions—and thus differentiates "good" material from "bad" (Spector and Alpern 1995).

The network, which contains 96 hidden-layer units and 2 output units, was trained using 4 types

of training data—again drawn from the repertoire of Charlie Parker. The training data set contained 2 measures of Charlie Parker; 1 measure of Parker followed by 1 measure of silence; 1 measure of Parker followed by 1 measure of a random melody; and 1 measure of Parker followed by randomly manipulated Parker melodies. This combination of training data ensured that the network learned measure pairs that showed general musical traits. In fact, the ability of the network to deduce musicality was reflected by the response to a 2-measure phrase by rock guitarist Jimi Hendrix; a high level of fitness was indicated in response to this particular input.

It was found that this neural fitness evaluator learned certain aspects of the musical style that the genetic composition system attempted to emulate, yet some of the output of the network was still deemed to be unmusical, having long periods of dense note groupings, and large intervallic steps. One means of improving the critical response of the system is to employ multiple critics, both neural and symbolic. A system implemented with multiple critics produced better output than the “pure” neural critic system.

Conclusion

This article attempted to review how the evolutionary procedures of genetic algorithms and genetic programming have been applied to the field of musical composition. Each of the reviewed applications demonstrates a different approach to using genetic techniques to achieve the same basic task—the composition of a sequence of musical events.

While many applications of genetic composition have been successful, there are certain drawbacks involved with this form of algorithmic music creation. One of the most crucial aspects of making a successful genetic composition system is the implementation of an effective fitness-evaluation regime. However, it is this part of the system that creates one of the most significant drawbacks, the fitness bottleneck. The usual cause of this bottleneck is the need to evaluate the fitness of individual musical phrases one at a time—a much

more context-dependent and time-consuming task than other subjective fitness-evaluation tasks. This is because music is sequential, and highly contextually dependent individuals evaluated for fitness in one order may result in different fitness values if evaluated in another order.

Additionally, the diversity of musical tastes and the subjectivity of the composition process may prevent specific genetic composition systems from appealing to all users. Many of the systems investigated here utilize fitness-evaluation regimes that are based upon either the tastes of the user of the system or the tastes of the programmer of the system. This illustrates the difficulty of producing a general algorithmic composition system—one that can generate musical events regardless of stylistic or individual taste constraints.

Most of the applications discussed fulfill the requirements of the desired task, yet none succeeds in approaching a general algorithmic composer. Of the two applications that utilize deterministic fitness functions, one relies on the subjectivity of the user, and the other is more an exercise in pattern matching. Applications that use formalistic fitness functions are generally successful, yet they are limited in scope to those stylistic musical forms encoded in the rule base used in fitness evaluation. Applications with user-determined fitness functions are generally the most successful type of genetic composer, as they share the tastes of the user by design. However, these applications are limited by the fitness bottleneck described above. Neural fitness evaluators are also successful, yet are limited by the training data used and their inability to generalize outside of the classes defined in this data. A method for solving the problems of neural fitness evaluators has been proposed (Burton and Vladimirova 1997) based on the clustering behavior of the adaptive resonance theory (ART) neural network.

Genetic programming techniques have generally been more successful in composition applications, owing to their ability to work in less constrained search spaces than GA applications. The most well-founded applications of the GP approach implement fitness evaluation based upon works already judged to be of high value. These types of fitness evaluators do not require implementing

specific features in these works in terms of musical aesthetic or theoretical adherence (Spector and Alpern 1994, 1995).

The majority of the applications under review have achieved a high degree of success in generating "realistic" musical compositions, with little or no formal implementation of musical theory. This success suggests a wide scope for future musical applications of genetic techniques.

References

- Baggi, D. L. 1991. "Neurswing: An Intelligent Workbench for the Investigation of Swing in Jazz." *Computer* (July):60–64.
- Balaban, M., K. Ebcioğlu, and O. Laske. 1992. *Understanding Music with AI: Perspectives on Music Cognition*. Cambridge, Massachusetts: MIT Press.
- Biles, J. A. 1994. "GenJam: A Genetic Algorithm for Generating Jazz Solos." *Proceedings of the 1994 International Computer Music Conference*. San Francisco: International Computer Music Association.
- Biles, J. A. 1995. "GenJam Populi: Training an IGA via Audience-Mediated Performance." Available on the World Wide Web at <http://www.it.rit.edu/~jab/GenJamPop.html>.
- Biles, J. A., P. G. Anderson, and L. W. Loggi. 1996. "Neural Network Fitness Functions for a Musical IGA." Available on the World Wide Web at <http://www.it.rit.edu/~jab/SOCO96/SOCO.html>.
- Burton, A. R., and T. Vladimirova. 1997. "A Genetic Algorithm Utilising Neural Network Fitness Evaluation for Music Composition." *Proceedings of the 1997 International Conference on Artificial Neural Networks and Genetic Algorithms*. Vienna: Springer-Verlag.
- Camurri, A., C. Canepa, M. Frixione, and R. Zaccaria. 1991. "HARP: A System for Intelligent Composer's Assistance." *IEEE Computer* 24(7):64–67.
- Chester, M. 1993. *Neural Networks—A Tutorial*. Englewood Cliffs, New Jersey: Prentice-Hall.
- Davis, L., ed. 1987. *Genetic Algorithms and Simulated Annealing*. San Francisco: Morgan Kaufmann.
- Desain, P., and H. Honing. 1992. *Music, Mind, and Machine*. Amsterdam: Thesis Publishing.
- Filho, J. L. R., P. C. Treleaven, and C. Alippi. 1994. "Genetic Algorithm Programming Environments." *IEEE Computer* 27(6):28–43.
- Gibson, P. M., and J. A. Byrne. 1991. "Neurogen, Musical Composition Using Genetic Algorithms and Cooperating Neural Networks." *Proceedings of the Second International Conference on Artificial Neural Networks*, pp. 309–313. Stevenage, England: Institute of Electrical Engineers.
- Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Massachusetts: Addison-Wesley.
- Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor, Michigan: University of Michigan Press.
- Horner, A., and L. Ayers. 1995. "Harmonization of Musical Progressions with Genetic Algorithms." *Proceedings of the 1995 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 483–484.
- Horner, A., and D. E. Goldberg. 1991. "Genetic Algorithms and Computer-Assisted Music Composition." *Proceedings of the 1991 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 479–482.
- Horowitz, D. 1994. "Generating Rhythms with Genetic Algorithms." *Proceedings of the 1994 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 142–143.
- Jacob, B. L. 1995. "Composing with Genetic Algorithms." Working Paper, Advanced Computer Architecture Laboratory, EECS, University of Michigan. Available on the World Wide Web at http://www.ee.umd.edu/~blj/algorithmic_composition/.
- Johnson, M. L. 1991. "Toward an Expert System for Expressive Musical Performance." *IEEE Computer* 24(7):30–34.
- Koza, J. R. 1992. *Genetic Programming*. Cambridge, Massachusetts: MIT Press.
- Laine, P., and M. Kuuskankare. 1994. "Genetic Algorithms in Musical Style Oriented Generation." *Proceedings of the First IEEE Conference on Evolutionary Computation*. Washington, DC: IEEE, pp. 858–861.
- McIntyre, R. A. 1994. "Bach in a Box: The Evolution of Four-Part Baroque Harmony Using the Genetic Algorithm." *Proceedings of the IEEE Conference on Evolutionary Computation*. Washington, DC: IEEE, pp. 852–857.
- Michalewicz, Z. 1992. *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag.
- Putnam, J. B. 1994. "Genetic Programming of Music." Technical Report, New Mexico Institute of Mining and Technology. Available on the World Wide Web at <http://www.nmt.edu/~jefu/notes/ep.ps>.

-
- Ralley, D. 1995. "Genetic Algorithms as a Tool for Melodic Development." *Proceedings of the 1995 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 501–502.
- Spector, L., and A. Alpern. 1994. "Criticism, Culture, and the Automatic Generation of Artworks." *Proceedings of the Twelfth National Conference on Artificial Intelligence, AAAI-94*. Cambridge, Massachusetts: AAAI, pp. 3–8.
- Spector, L., and A. Alpern. 1995. "Induction and Recapitulation of Deep Musical Structure." *Working Notes of the IJCAI-95 Workshop on Artificial Intelligence and Music*, pp. 41–48. San Francisco: Morgan Kaufmann
- Srinivas, M., and L. M. Patnaik. 1994. "Genetic Algorithms: A Survey." *Computer* (June):17–26.
- Thywissen, K. 1996. "GeNotator: An Environment for Investigating the Application of Genetic Algorithms in Computer-Assisted Composition." *Proceedings of the 1996 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 274–277.
- Todd, P. M., and D. G. Loy, eds. 1991. *Music and Connectionism*. Cambridge, Massachusetts: MIT Press.
- Wasserman, P. D. 1993. *Advanced Methods in Neural Computing*. New York: Van Nostrand Reinhold.
- Werner, G. M., and P. M. Todd. 1997. "Too Many Love Songs: Sexual Selection and the Evolution of Communication." *Proceedings of the Fourth European Conference on Artificial Life*, pp. 434–443. Cambridge, MA: MIT Press.