GA-based Music Arranging for Guitar

Daniel R. Tuohy and W.D. Potter

***Abstract*— In this paper we describe a system for converting a piece of music into an appropriate arrangement for guitar. The system is supported by previous research into the generation of guitar performance information (tablature) with genetic algorithms. This technology forms the basis of an evaluation function that can effectively differentiate between candidate arrangements on the basis of both faithfulness to the original piece and on playability. We describe a genetic algorithm and a greedy hill-climber that use this function to find good arrangements. Both techniques have demonstrated an ability to create viable arrangements of the pieces tested.**

I. Introduction

Arranging is the process by which a piece of music is adapted so that it can be performed on an instrument or set of instruments other than those for which it was originally written[3]. When arranging a piece of music, the piece may require significant alteration in order to be playable on the new instruments. Such alterations are neccessary when a piece contains notes too high or too low to be played on an instrument, or when there are simply too many notes for a performer to play. A piano player has only ten fingers, a guitar has only six strings, and a trumpet only has one tube. These limits impose restrictions on what can be played on the instrument. When adapting a piece of music, one must be cognizant of such limitations and carefully consider how to work around them. This often requires that notes be moved up or down an octave, shortened, lengthened, or eliminated altogether. A good arranger will make these adjustments in such a way that the original intent of the composer is preserved.

The arrangements we are concerned with here are the subset known as “reductions”. Reductions are arrangements in which as few creative liberties as possible are taken and the goal is only to accurately reproduce the original composition while ensuring that it is possible to be performed. When creating a reduction an arranger does not introduce new harmonies, counter-melodies, or bass lines, but merely takes as much as possible of that which the composer has provided.

Our system is designed to create arrangements for the guitar. In producing these arrangements we are limited by the abilities of the human hand. Only six notes can be simultaneously produced, and we must be sure not to exceed the dexterity or reach of the fingers. In our first section we discuss research on the topic of using evolutionary computation for music generation. We also discuss TabGA, a genetic algorithm for creating guitar tablature developed previously that is vital to our evaluation function. The next section is an explanation of the evaluation function, and the fourth section describes the algorithms that use this function. In the final section we discuss the results of our experimentation with the automatic generation of arrangements.

II. Background

A. Academic Research

There has been an enormous amount of research into the automatic creation of music, particularly with techniques from evolutionary computation[4][23]. One of the first papers on the subject, by Andrew Horner and David Goldberg, used genetic algorithms for thematic bridging[6]. Two successful and often cited projects are those by Al Biles and Bruce Jacob. Biles created an Interactive Genetic Algorithm that could produce pleasing jazz solos with the aid of a human fitness function[2]. Jacob’s system used a Genetic Algorithm to compose music using musical motives provided by a composer. The goal of all of these systems is to create new music, which should seem altogether new to the listener[7]. Our system, on the other hand, is designed to adapt existing music for performance on the guitar.

Work on creating arrangements of compositions automatically is rare. Jacob’s system includes an “arranging” module, but the term is used in a different sense, that of arranging phrases into larger statements. Another system by Nagashima and Kawashima uses chaotic neural networks to “arrange” music, but by this they mean creating variations on melodies[11]. Neither system actually creates arrangements as the word is strictly defined in music. Perhaps this is not surprising because arranging is not a task one would normally be inclined to automate. The process of arranging for an ensemble or piano, for example, is predominantly creative because so few limitations are enforced by the instruments. It is because solo guitar is such a limiting instrument that the problem of automating arranging is so interesting. Ensuring that an arrangement is within the realm of playability is a complicated, and relatively uncreative, task to which it is useful to apply computation[10][14][15][16].

We know of no other research pertaining to automatic arranging for the guitar or any instrument, and we believe this system to be the first of its kind.

B. The Guitar Fingering Problem and TabGA

Stringed instruments are unique in that nearly every note can be played in multiple positions on the instrument. Choosing between these positions can be a laborious process, and guitar players often use tablature to record exactly how a piece is to be played. Instead of notes, tablature encodes music as a sequence of positions on the fretboard. This information is essential to determining how difficult a piece of music is on guitar, or if it can even be played at all. For this reason, tablature is of paramount importance when arranging for guitar. A potentially beautiful arrangement of a piece of music is worthless if it cannot be played.

Previously, we developed a system called TabGA that reliably produces competent tablature for any piece of guitar music[17]. TabGA is a distributed genetic algorithm that evolves tablature for a given sequence of notes. The fitness function provides selection pressure on the basis of tablature playability. Both this function and the operating parameters of TabGA were optimized with meta-genetic algorithms in order to maximize the consistency of generated tablatures with those created by human experts. The result was a high level of consistency, above 90%, as well as good results even when the generated tablature diverged from published tablature. The system also employs a neural network to assign left-handed fingers to each note, though this feature is supplementary and not particularly relevant to this paper. Both the search algorithm and the fitness function are detailed in [18]. Tablature lends itself to very natural processing by search algorithms and can be encoded as a chromosome for a GA very easily. The reproductive process used in TabGA is illustrated in Figure 1.

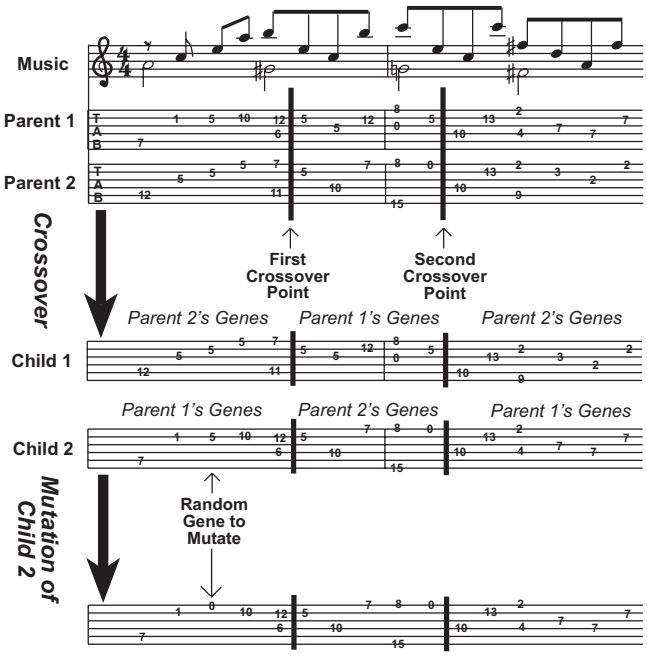

Fig. 1. The creation of a child tablature from two parents in TabGA.

C. The Goal of our System

We are attempting to *preserve* beauty, not *create* it. To automate arranging we attempt to minimize the creative input of the system and maximize the creative input of the composer. The task of endowing a machine with creativity is a daunting one, and one which we do not wish to address in this research. The arrangements produced by our approach will not interweave new musical material with the original composition, but attempt to reproduce the original composition as accurately as possible while ensuring that the arrangement is still within or below a reasonable difficulty level.

III. The Evaluation Function for Assessing an Arrangement

It should be noted that we do not expect our approach to have a fine-grained ability to differentiate between the competence of arrangements, which is the target concept of this domain and is somewhat subjective in nature. We have a set of heuristics by which we judge an arrangement, and expect only that these heuristics are sufficient to be effective. An effective heuristic has the ability to distinguish the good arrangements from the poor. We consider this to be sufficient, because even two humans who can tell the difference between good and bad arrangements could easily disagree on which of two good arrangements is superior.

Our evaluation function for assessing arrangements attempts to satisfy two objectives. These are to determine the playability of the tablature and to assess the degree to which an arrangement preserves the composer's original intent. There are four components constituting these two objectives, and these are added together to assess the competence of an arrangement.

A. Maintaining Playability

The first objective is to keep the arrangement from becoming too difficult. To do this, TabGA is given the arrangement and returns a tablature corresponding to the arrangement and a value assessing the difficulty of this tablature. This value is the first component of our fitness function and is referred to as the Playability Component (PC). A higher PC value indicates an easier tablature.

B. Preserving the Original Composition

The second objective is to maximize the extent to which the arrangement sounds like the original piece of music. In order to do this effectively we acknowledge that not all notes are created equal. Melodic notes are regarded as essential whereas others are supplementary and are important to varying degrees. We currently allow for six categories of note, five of which are illustrated in figure 2. We distinguish notes belonging to the melody, harmony, counter-melody, and bass. We have two additional categories for notes in the harmony and bass that are constituents of moving lines. Each category of note has an associated weight that represents how important it is to the piece of music. There are several tenets of arranging that guide us in deciding how notes are to be weighted[3][8]. We wish to provide contrast between the voicings, avoid heavy doubling of notes in different octaves, and to avoid having so many simultaneous notes that important moving lines can not be clearly distinguished. For our experimentation we specify note types by hand, but only because we have not written an algorithm to read music. Good techniques exist for automating this task[19][9].

The weight for each note is the extent to which its inclusion benefits the fitness of the arrangement and is referred to as the Note Weight Component (NWC).

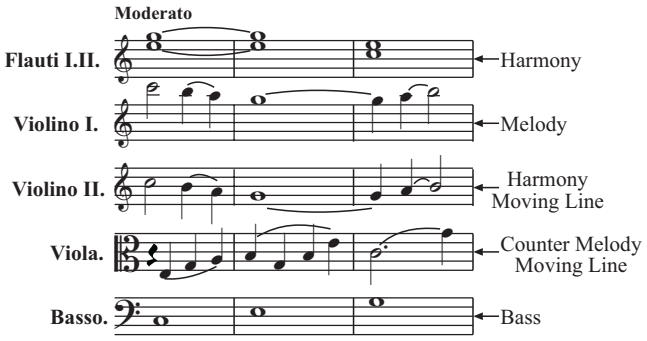

Fig. 2. An excerpt from Rachmaninoff's Symphony No. 2 that contains examples of 5 of the 6 note categories that our system recognizes.

$$N\dot{W}C = \sum_{i=0}^{N} incl(i) \times weight(i) \tag{1}$$

Where *N* is the number of notes in the original piece and *incl(i)* evaluates as one if the note is included in the arrangement and zero otherwise. The weight associated with note *i* is returned by *weight(i)*.

We also recognize the importance of contiguous notes in the moving lines, so there is a fitness bonus when consecutive notes in a moving line are included. This is referred to as the Moving Line Continuity Component (MLCC).

$$M\dot{L}CC = \sum_{i=0}^{L} \sum_{j=0}^{N_i} incl(j) \times incl(j-1) \tag{2}$$

Where *L* is the number of moving lines within the phrase, N_i is the number of notes in line *i* and *incl(j)* evaluates as one if note *j* is included in the arrangement and zero otherwise.

A fitness reward is also assigned for the number of notes in each chord. This factor benefits an arrangement based on the number of notes from each chord included. However, the incremental increase in the reward for each additional note decays exponentially as illustrated in figure 3. The first note included in a chord increases the bonus for that chord far more than the fourth or fifth. In this way we hope to maintain an even distribution of notes throughout the arrangement, which is in line with how actual guitar arrangements are created. This will help discourage the arrangement from becoming intermittently thick, which could obscure important musical content. This is referred to as the Notes in Chord Component (NCC).

$$N\dot{C}C = \sum_{i=0}^{C} 1 - e^{-.5 \times \sum_{j=0}^{O_i} incl(j)} \tag{3}$$

Where *C* is the number of chords in the original piece of music.

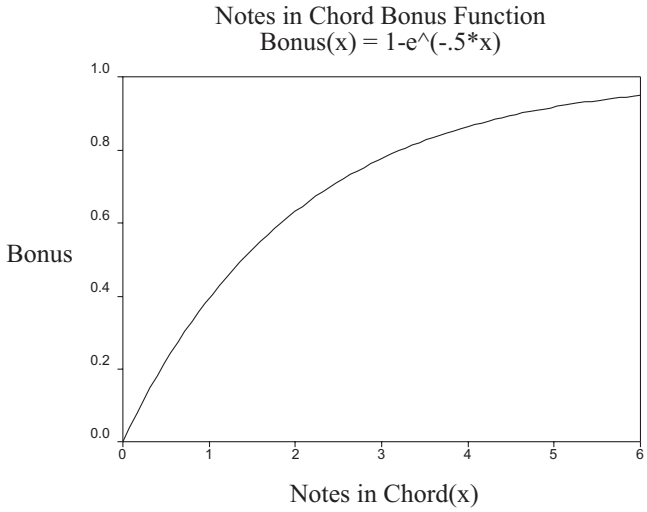

Fig. 3. The function defining the bonus associated with a number of notes in a chord.

The fitness, or competence, of an arrangement is given by the following equation.

$$\dot{Fitness} = PC + NWC + MLCC + NCC \tag{4}$$

It should be noted that each of the four components is subject to being weighted, and these weights are fully intended to be adjustable at the users discretion. We would not, for example, wish to place too high an importance on playability. This would result in thin arrangements with too few notes because it is easier to play fewer notes. Likewise, placing too much importance on faithfulness to the original piece may result in an unplayable arrangement. Adjusting these weights also allows the system to produce arrangements for users of varied ability levels, most effectively by increasing or decreasing the weights associated with the Playability Component and the Note Weight Component. These components are primarily responsible for controlling the balance between ease of performance and the number of notes included.

IV. Methods for Obtaining Arrangements

A. Arranging Music with a Genetic Algorithm

The representation for both the genetic algorithm and the hill-climber is a binary string where each bit represents a note in the original piece of music. A one indicates that the note is included in the phenotypic arrangement while a zero indicates its absence. The GA is steady-state, so only one individual is replaced in each iteration [21]. New individuals are created by two-point crossover and mutation occurs at every bit with a probability of 3%. Parents are chosen with binary tournament selection. Each piece of music is broken into logical phrases by hand so that the search space remains

manageable. Break points between phrases are picked by how much time is given to the performer between two notes to move his/her hand freely. Methods exist for performing this segmentation automatically for the guitar by Radicioni et al.[13]. Once the first phrase has been arranged, each subsequent phrase includes the fretboard positions of the last three notes of the previous phrase. In this way, the GA has information about where on the fretboard the performer's hand is at the end of the previous phrase and can create tablature for the subsequent phrase accordingly. It should be noted that all melodic notes are always included (i.e., set to one in every chromosome) and as such do not contribute to the dimensionality of the search space.

Because each fitness evaluation requires running TabGA to create tablature, the GA can require an hour or more to create an arrangement of a lengthy phrase. In the next section we explore an approach that more closely resembles that of a human arranger, but takes far less time to find a local optima.

B. Arranging Music with a Greedy Hill-Climber

In this model we attempt to arrange the piece starting with the most important notes and ending with the least important. This is accomplished by setting a minimum allowable weight, and decreasing this weight whenever all the notes at or above that weight have been given a chance for inclusion without success. Whenever a note is found to improve the fitness of the arrangement, it becomes a permanent addition to the arrangement. Because of this, we consider the algorithm to be greedy. Whenever this happens, all the other notes considered to be of equal or greater importance are given another chance to be included again, effectively restarting the search at the current minimum allowablw weight level. The algorithm is described by the following instructions.

1. Set a high minimum allowable weight.
2. Attempt to add a note that is more heavily weighted than the current minimum allowable weight.
3. If the note increases the fitness of the arrangement, it becomes a permanent addition. Every note above the current minimum allowable weight is now eligible to be tried again, go to **2**.
4. Else, try the subsequent note above the current minimum allowable weight.
5. If every note has been tried without any improvement, decrease the minimum allowable weight and go to **2**.
6. If the minimum allowable weight is 0, and every note has been tried, we have reached a local optima and are finished.

Each time a note is accepted into the arrangement, we restart at a random point in the bit string and regard the bit string as circular. This is to avoid biasing the arrangement towards notes at the beginning, which would receive more opportunities to be included if the algorithm began at the beginning every time a note was accepted.

V. Results

We selected two pieces of music on which to test our approach. These are the moderato from the second movement of Symphony No. 2 in E Minor by Sergey Rachmaninoff[12], and the Arioso from Cantata No. 156 by J.S. Bach [1]. The former is a piece for full orchestra, the second is adapted for four trombones. We began with the Rachmaninoff, and in our first run created an arrangement that was too difficult for an intermediate guitarist. To bias the search more towards playability we decreased the weight given to the Note Weight Component by 25%. The second run produced an arrangement which we found to be quite acceptable, and the corresponding tablature produced by TabGA was of reasonable difficulty. Part of this arrangement is shown in figure 4. The first run for the Bach gave us an arrangement that seemed too thin and was very easy, so we reset the Note Weight Component to its original value and ran it again. The second run gave us a pleasing arrangement with an easy tablature.

The arrangement in figure 4 was obtained with the greedy hill-climber. This results in local optima which are, on average, of lower fitness than those found with the GA. After obtaining our arrangements for both pieces we ran the genetic algorithm with the same weights. The arrangements obtained with the GA were, in our estimation, of roughly the same quality as those obtained with the hill-climber even though the fitnesses were better. We attribute this to the coarseness of our fitness function. We believe that although it can reliably discern the good arrangements from the bad, it can not reliably distinguish the good from the excellent or the bad from the awful. Any improvements made to the fitness function in the future will likely enhance the legitimacy of the GA.

We are pleased with the results of our experimentation. We were able to obtain arrangements as good as those we would have been able to create ourselves yet in a fraction of the time. Examples of arrangements produced by our system, both as tablatures and as mp3 recordings are available at http://www.ai.uga.edu/tuohy/excerpts.html.

VI. Conclusions and Further Directions

There are many contexts in which we believe our system is useful. For the user uninterested in arranging music themselves, the internet is a superb resource for free and easily accessible MIDI files for almost any well-known piece of music. MIDI files are electronic versions of music from which can be read all of the information used to create arrangements with our system. Combined with software for extracting this information, one could quickly obtain arrangements for any piece of music for which MIDI or sheet music can be found. Algorithms for automating both this process and that of automatically determining the relative importance of notes would be very useful additions to our current system, potentially eliminating the need for any user input[20][19].

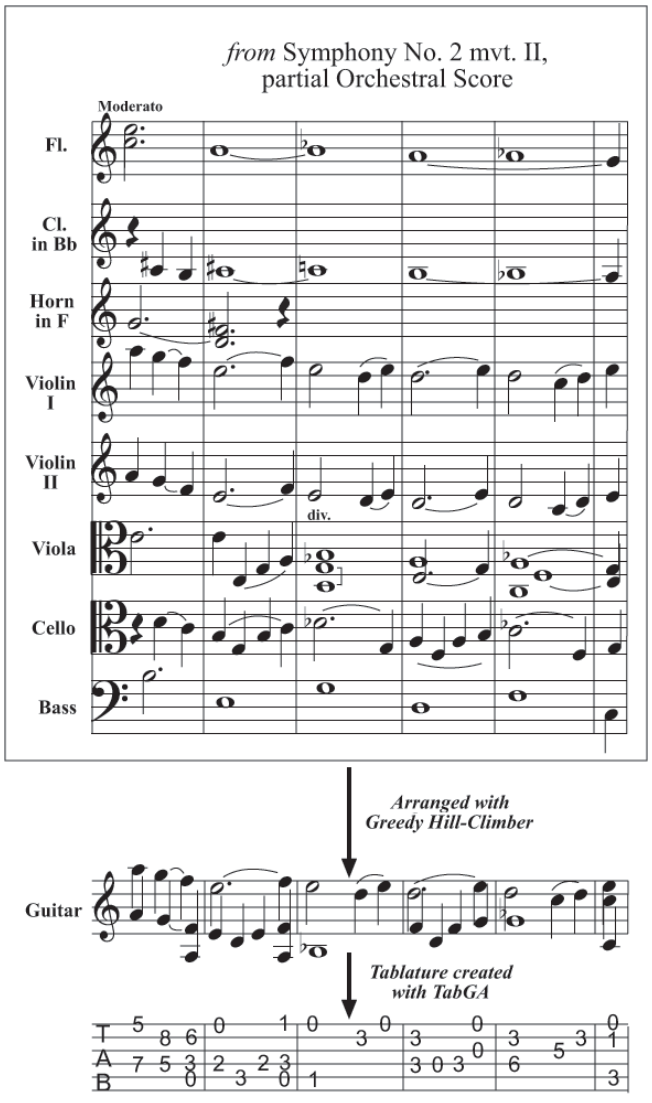

Fig. 4. An arrangement of an excerpt from Rachmaninoff's Symphony No. 2 by our system. We use either a greedy hill-climber or genetic algorithm (in this case, the former) to generate an arrangement from the orchestral score. The tablature for the arrangement is produced by TabGA.

Those who wish to have more control over the arranging process can also benefit from our system. A user can arrange the piece themselves, then set all the notes in the arrangement as being essential and run the search. This will likely add notes to the existing arrangement, potentially enriching the sound. Users can also search for the best guitar tuning and key for the piece by adapting the input. The system can be used as a source of musical ideas. A user can easily scavenge the output of the system for useful material that might not have occurred to them in the course of arranging a piece themselves.

REFERENCES

[1] J.S. Bach, *Arioso from Cantata No. 156, arr. Patrick McCarty for four trombones*, Rochester, NY: Ensemble Publications, 1961.

[2] J.A. Biles, "GenJam: A genetic algorithm for generating jazz solos," *Proc. International Computer Music Conference*, Aarhus, Denmark, 1994, pp.131–137.

[3] V. Corozine, *Arranging music for the real world: classical and commercial aspects*, Pacific, MO: Mel Bay, 2002.

[4] A. Gartlant-Jones and P. Copley, "The Suitability of Genetic Algorithms for Musical Composition," *Contemporary Music Review*, vol. 22(3), pp. 43–55, 2003.

[5] H. Heijink and R.G.J. Meulenbroek, "On the complexity of classical guitar playing: functional adaptations to task constraints," *Journal of Motor Behaviour*, vol. 34(4) pp. 339–351, 2002.

[6] A. Horner and D.E. Goldberg, "Genetic algorithms and computer-assisted music composition," *Proc. Fourth International Conference on Genetic Algorithms*, San Francisco, USA, 1991, pp. 479–482.

[7] B.L. Jacob, "Composing with Genetic Algorithms," *Proc. International Computer Music Conference*, Banff Alberta, Sept. 1995, pp. 452–455.

[8] D. Michael, *Arranging for Open Guitar Tunings*, Anaheim Hills, CA: Centerstream Publishing, 2003.

[9] C. Meek and W.P. Birmingham, "Automatic Thematic Extractor," *Journal of Intelligent Information System: Special Issue on Music Information Retrieval*, vol. 21(1), pp. 9–33, July 2003.

[10] M. Miura, I. Hirota, N. Hama, and M. Yanigida, "Constructing a System for Finger-Position Determination and Tablature Generation for Playing Melodies on Guitars," *Systems and Computers in Japan* vol. 35(6), pp. 755–764, 2004.

[11] T. Nagashima and J. Kawashima, "Experimental Study on Arranging Music by Chaotic Neural Network", *International Journal of Intelligent Systems*, vol. 12, pp. 323–339, 1997.

[12] S. Rachmaninoff, *Symphony No. 2 in E Minor, Op. 27, in Full Score*, Mineola, NY: Dover Publications.

[13] D. Radicioni, L. Anselma and V. Lombardo, "A segmentation-based prototype to compute string instruments fingering," *Proc. Conference on Interdisciplinary Musicology*, Graz, Austria, 2004.

[14] D. Radicioni and V. Lombardo, "Guitar Fingering for Music Performance," *Proc. International Computer Music Conference*, Barcelona, Spain, Sept. 2005, pp. 527–530.

[15] A. Radisavljevic and P. Driessen, "Path Difference Learning for Guitar Fingering Problem," *Proc. International Computer Music Conference*, Miami, USA, Nov. 2004.

[16] S. Sayegh, "Fingering for String Instruments with the Optimum Path Paradigm", *Computer Music Journal*, vol. 13(6), pp.76–84, 1989.

[17] D.R. Tuohy and W.D. Potter, "A Genetic Algorithm for the Automatic Generation of Playable Guitar Tablature," *Proc. International Computer Music Conference*, Barcelona, Spain, Sept. 2005, pp. 499-502.

[18] D.R. Tuohy and W.D. Potter, "Creating Guitar Tablature with Neural Networks and Distributed Genetic Search,", to appear in *Proc. International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems* Annecy, France, 2006.

[19] H.-H. Shih, S.S. Narayanan, and C.-C.J. Kuo, "Automatic main melody extraction from MIDI files with a modified Lempel-Ziv algorithm", *Proc. International Symposium on Intelligent Multimedia, Video and Speech Processing*, May 2001.

[20] J. Wand and L. Tsai-Yen, "Generating Guitar Scores from a MIDI Source", *Proc. International Symposium on Multimedia Information Processing*, Taipei, Taiwan, 1997.

[21] D. Whitley and J. Kauth, "GENITOR: A Different Genetic Algorithm," *Proc. Rocky Mountain Conference on Artificial Intelligence*, Denver, CO, 1988, pp. 118–130.

[22] D. Whitley and T. Starkweather, "GENITOR II: a distributed genetic algorithm", *Journal Expt. Theor. Artif. Intel.*, vol. 2, pp. 189–213, 1990.

[23] G. Wiggins, G. Papadopolous, S. Phon-Amnuaisuk, and A. Tuson. "Evolutionary Methods for Musical Composition", *International Journal of Computing and Anticipatory Systems*, vol. 4, 1999