Genetic Algorithms and Computer-Assisted Music Composition

Genetic Algorithms and Computer-Assisted Music Composition

Andrew Horner and David E. Goldberg
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Abstract

Genetic algorithms have been used with increasing frequency and effectiveness in a variety of problems. This paper investigates the application of genetic algorithms to music composition. A technique of thematic bridging is presented that allows for the specification of thematic material and delegates its development to the genetic algorithm. A look at the effects of building block linkage subsequently establishes a basis for implementing a GA-optimizable operation set for the problem. Some preliminary results are then discussed with an eye toward future work in GA-assisted composition.

1 Introduction

Genetic algorithms (Goldberg, 1989; Holland, 1975) have been applied to a wide array of problem domains from medical image registration (Fitzpatrick, Grefenstette, & Van Gucht, 1984) to stack filter design (Chu, 1989). The accessibility of their interface has contributed to experimentation on problems from many diverse disciplines. Unlike most traditional optimization techniques, GAs don't rely on a particular problem structure or problem-specific knowledge. The general effectiveness of GAs in blindly optimizing strings in no small part accounts for their growing popularity.

It is this accessibility and effectiveness that suggests that GAs might provide a powerful tool in the computer-assisted composition of music. Various approaches to computer-assisted composition have been employed in generating music. These range from the relatively straightforward mapping of functions to compositional parameters (Dodge, 1988) through elaborate stochastic feedback models (Tipei, 1989; Xenakis, 1971). Recent work using neural network composition methods to learn and generalize structures from existing musical examples is also of particular interest (Todd, 1989).

The approach adopted in this paper is one of thematic bridging. Thematic bridging is the transformation of an initial musical pattern to some final pattern over a specified duration. This method is often characteristic of *phase* or *minimalist* music and was chosen because of the first author's compositional approach to music and the fact that it lends itself so naturally to GA encoding.

The bridging is brought about through modifying or reordering elements of intervening patterns through a sequence of operations. The elements themselves may be discrete pitch, amplitude, or duration values. The resulting musical output consists of the concatenation of each partially developed pattern (or some variation of it) through the final pattern. The initial pattern itself is not part of the output, but could be a part of a preceding section as the final pattern.

As a simple example, with an initial note pattern (Gb,Bb,F,Ab,Db), a final pattern (F,Ab,Eb), and a note duration of 17, a bridge could be as follows:

operation description	resulting pattern
delete the last note of the initial pattern	Gb Bb F Ab
rotate the pattern	Bb F Ab Gb
delete the last note	Bb F Ab
mutate the first note	Eb F Ab
rotate the pattern	F Ab Eb

musical output: Gb Bb F Ab Bb F Ab Gb Bb F Ab Eb F Ab F Ab Eb

In addition to the thematic patterns and bridge duration, the basic operation set for transforming patterns must be specified. In the example above, the operation set includes atomic operations that change and delete notes, and an operation for rotating the pattern as a whole.

Some operation sets may not be capable of bridging the initial and final patterns over the duration specified, regardless of the sequencing. This suggests an iterative approach, where the problem is run to convergence and subsequently rerun with modifications if results are unsatisfactory (figure 1). For example, if the thematic patterns are dissimilar and the bridge duration is short, relatively high-level operations may be needed to bring off the transformation.

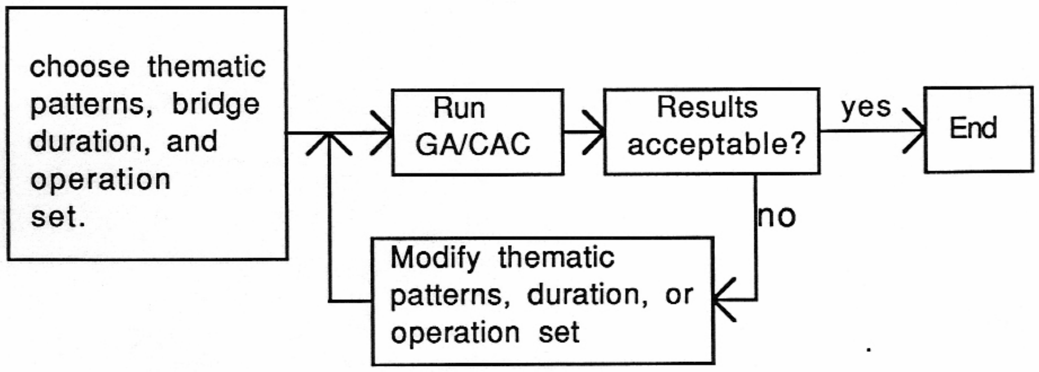
choose thematic patterns, bridge duration, and operation set.
Run GA/CAC
Results acceptable?
yes
End
no
Modify thematic patterns, duration, or operation set

Figure 1: Iterative modification in GA-assisted composition.

Thematic bridging contrasts with more conventional algorithmic composition techniques in that the transition rules are not explicitly applied. Instead, the GA selects a sequence of operations or atomic transition rules to apply from a user-supplied operation set in constructing the bridge. This bridging bears some resemblance to the interpolation of melodies in neural network algorithmic composition (Todd, 1989), though learning is not a part of the bridging process.

2 GAs and Thematic Bridging

Thematic bridging bears a close resemblance to another GA application, job shop scheduling (Davis, 1985; Cleveland & Smith, 1989). In job shop scheduling a group of tasks must be scheduled so that the total waiting time of the tasks is minimized. We can draw an analogy between the number of notes produced by a bridging operation and the wait of a corresponding job. Further, pattern bridging corresponds to constraints imposed upon job sequencing. This analogy may be useful in visualizing the problem.

The following section examines a particular operation set, and looks at its encoding. A given encoded operation sequence can be evaluated by simply executing the sequence and comparing its results to those desired. A simple 3-operator GA with binary tournament selection and 1-point crossover was used in all the examples presented in this paper. Only minor modifications of the GA were required to accommodate bridging operators and operands as a genic ordered pair.

2.1 An operation set and encoding

The following is a general description of the operation set used in generating the results presented in this paper. Other operation sets can include any variety of customized operations. Note that the **Mutate** operation below should not be confused with the GA mutation operator.

Operation	Function
No-op	"do nothing" operator (takes up space)
Add	add an element to the pattern (either a needed element or one specified by the operand)
Delete	delete an element of the pattern (either an extraneous element or one specified by the operand)
Mutate	change an element of the pattern (tries to change an extraneous element to a needed one, otherwise as specified by the operands)
Rotation	rotate the pattern (number of rotations and direction specified by the operands)
Exchange	exchange elements of the pattern (moves first element to the position specified by the operand through a series of exchanges with neighboring elements)
IncorrectRotation	rotate incorrectly positioned elements of the pattern
IncorrectExchange	exchange incorrectly positioned elements of the pattern

The chromosome encodes the operation sequence from left to right with one operation per gene. Each gene is an operator that may or may not be accompanied by operands. In the actual implementation auxiliary structures were used to store the operands, which were crossed in parallel with the chromosome.

Each operand is represented by a real number between 0.0 and 1.0. Initially, a random number in this range is selected; when mutation occurs another random number is chosen as the replacement. These raw operands are subsequently scaled to appropriate values according to their operator's function. The use of real codings has a long if controversial history in the history of evolutionary methods. A recent paper (Goldberg, 1990) has put that usage on firmer theoretical footing.

Since simple GAs optimize fixed-length chromosomes, an appropriate length estimate for potential operation sequences must be found. This figure can be safely overestimated, since **No-ops** can effectively fill out extra positions. Without control flow instructions, execution proceeds in a simple left-to-right scan of the chromosome operations. In general, the length estimate should be based on the particular thematic patterns, operation set, and bridge duration under consideration. The examples tried to date have varied from a handful of operations up to lengths of 40. If control flow operations are included in the instruction set, their effect should also be considered.

Returning to the simple example presented earlier, the operation sequence would look something like:

(**Delete** 5) (**Rotate** 1 forward) (**Delete** 4) (**Mutate** 1 3) (**Rotate** 1 forward)

The **Mutate** and **Delete** operations use the first operand to specify the position affected. The second operand of the **Mutate** operation specifies the position in the final pattern to get the new value. The rotation operations indicate that a single forward rotation should be performed.

2.2 A reasonable fitness function

The fitness function of a given operation sequence consists of a two-part hierarchy. The initial part is based on how close the developed pattern matches the final pattern. If there is an exact match, the fitness is set to a value derived by comparison of the resulting and desired output durations (i.e. the number of notes produced verses the number of notes desired).

The pattern-matching aspect of the fitness is computed from two components, the contents of the patterns and the ordering of their elements. These two checks insure that the right elements appear in the resulting pattern and in the correct order. The fitness function for this part is given by (maximizing):

$$\text{fitness} = 0.5 * (P_{common} + P_{correct})$$

where P_{commom} is the proportion of elements in the resulting pattern also in the final pattern, and $P_{correct}$ is the proportion of elements in the resulting pattern which match their corresponding element in the final pattern.

With an exact match of the resulting and final patterns (fitness=1.0), the fitness is augmented through a calculation which compares the resulting and desired musical output durations. The resulting fitness is maximal in the case of an exact duration match. The evaluation of the second part of the fitness is as follows:

$$\text{fitness} = 1.0 + N_d - \text{abs}(N_d - N_r)$$

where N_d is the desired duration, and N_r is the resulting duration.

In practice, a resulting duration slightly greater than the desired duration is often acceptable as a satisficing solution. In these cases, the extra output is simply truncated to the desired length.

3 Musical Results and Interpretation

This section considers some musical results and their relation to GA theory. Several levels of refinement were necessary in the implementation to arrive at these results. Problems with building block linkage arose in the initial implementation, but were subsequently overcome.

3.1 Musical results

So what does genetic music sound like? This depends on the operation set and, of course, the fitness function.

The problem, operation set, and GA have been programmed in Smalltalk. A short piece has been created by the GA using thematic bridging for pitch and amplitude development. The piece is entitled "Epistasis" and consists of five voices in canon (the voices play the same material at different times) over six bridge passages. The GA was run for each of the passages independently, given the thematic patterns and duration of each. The musical output was then scheduled among the five voices so that the passages would overlap. The thematic material itself is quite simple, which is something of a challenge since its development must sustain interest. The results are musically pleasing to the authors with the usual qualifications regarding personal taste. Readers may contact the first author to obtain ordering information to receive a tape of GA-generated music.

3.2 First results

The initial runs performed using the operation set presented above were not able to find useful solutions to even fairly trivial test problems. In these cases the run progressed to find a good collection of elements, but was unable to correctly order it. Inspection of the operations encoded in bad chromosomes revealed that no single operation change was sufficient to correct the ordering; generally, several changes were needed. Moreover, these changes were often widely spaced within the chromosome, a case of poor linkage.

The **Add** and **Delete** operations seemed to contribute most to these conditions. Generally, these operations used their operands to determine the position of the element to be added or deleted, which means picking the right element for the pattern while putting it in the right place. The nonlinearities introduced by simultaneously determining the correct collection and positioning introduced numerous local optima in the search space with poor linkage.

3.3 Improved results

An alternative approach was adopted in an attempt to reduce the problem's nonlinearities to a reasonable level and improve linkage. This entailed a modification of operations such as **Add** and **Delete**. With the **Add** operation, an operand again determined the position of the added element, but the selection itself was made from elements in the final pattern not in the current pattern. This heuristically moves selection toward the correct collection of elements, and leaves room for their positioning. This significantly improves linkage since useful combinations of operator changes can occur closer together. All runs to date have converged to useful solutions using this strategy.

Runs using a hill-climbing GA (a GA with only selection and mutation operators) with the heuristic operation set have also been successful, but have been more susceptible to stalling. About one in five hill-climbing runs encounters problems, with long waits ensuing. These problems again occur when any single operation change to a suboptimal chromosome results in decreased fitness. Unlike hill-climbing, the normal GA with crossover recombines parts from distinct individuals to break out of this problem. With heuristic operations, these critical crosses are fairly likely since elements are generally added and deleted as needed, which makes operations easier to mix and match. Thus, a recombinative GA generally fares much better than a hill-climbing GA on the same problem.

4 Future Work

Several extensions to the work presented in this paper follow quite naturally. The thematic material used to date has been relatively simple; more sophisticated material and textures are needed to gage the general effectiveness of the technique. As mentioned earlier, experiments using control flow and other customized operations are likely to be fruitful. Beyond pitch, amplitude, and rhythmic development, manipulation of timbre (sound quality) via music synthesis techniques or MIDI control changes promises further compositional control using the same basic technique (interpolation may be used between discrete values for continuous changes).

In addition, there are other aspects of algorithmic composition that might lend themselves to GA optimization. These include the generation of coherent sets of material for thematic bridging and the subsequent scheduling of the GA-generated passages. Moreover, the application of messy GAs (Goldberg, Deb, and Korb, 1990; Goldberg, Korb, and Deb, 1989) to the problem would be a natural encoding, since the operation sequence is inherently variable-length. Genetic programming (Koza, 1990) might also be used in conjunction with hierarchical operation trees.

5 Conclusions

This paper has applied genetic algorithms to music composition through a technique of thematic bridging. The technique is only one method of using GAs in computer-assisted composition, but it raises many of the issues common to other approaches. For instance, any approach will have to consider the operation set, encoding, and fitness function. Similarly, consideration of building block linkage is useful in constructing a GA-friendly operation set. Finally, several directions for further study have been outlined and discussed which promise interesting musical results in future genetic composition.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants CTS-8451610 and ECS-9022007. The first author also acknowledges support provided by the CERL Sound Group at the University of Illinois. Musical realization of this work was facilitated by Symbolic Sound Corporation's Kyma Workstation.

References

Chu, C. (1989). A genetic algorithm approach to the configuration of stack filters. *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications*, 112-120.

Cleveland, G., & Smith, S. (1989). Using genetic algorithms to schedule flow shop releases. *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications*, 160-169.

Davis, L. (1985). Job shop scheduling with genetic algorithms. *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, 136-140.

Dodge, C. (1988). Profile: A musical fractal. *Computer Music Journal 12*(3): 10-14.

Fitzpatrick, J. M., Grefenstette, J. J., & Van Gucht, D., (1984). Image registration by genetic search. *Proceedings of IEEE Southeast Conference*, 460-464.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Goldberg, D. E. (1990). *Real-coded genetic algorithms, virtual alphabets, and blocking* (IlliGAL Report No. 90001). Urbana: University of Illinois, Illinois Genetic Algorithms Laboratory.

Goldberg D. E., Deb, K., & Korb B. (1990). Messy genetic algorithms revisited: Studies in mixed size and scale. *Complex Systems*, *4*, 415-444.

Goldberg D. E., Korb B., & Deb, K. (1989). Messy genetic algorithms: motivation, analysis, and first results. *Complex Systems*, *3*, 493-530.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: The University of Michigan Press.

Koza, J. (1990). *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems* (Report No. STAN-CS-90-1314). Stanford: Stanford University, Department of Computer Science.

Tipei, S. (1989). The computer: A composer's collaborator. *Leonardo*, *22*(2): 189-195.

Todd, P. (1989). A connectionist approach to algorithmic composition.*Computer Music Journal 13*(4): 27-43.

Xenakis, I. (1971). *Formalized music*. Bloomington: Indiana University Press.