EFFECTS OF MUSICALLY MEANINGFUL OPERATORS IN EVOLUTIONARY COMPOSITION

John Huddleston *Jianna Zhang*
Western Washington University
Computer Science
516 High St.
Bellingham, WA 98225

ABSTRACT

Musically meaningful genetic operators have become an accepted aspect of evolutionary composition despite a lack of objective evidence to support their use. This research investigates the impact of four musically meaningful mutation operators from previous research on population fitness relative to a standard mutation operator. Two-dimensional rhythmic patterns are evolved using user-defined fitness trajectories. Although none of the meaningful operators has led to drastic improvements, the Invert-All operator has been consistently successful. These experimental results suggest that the combination of the Invert and Rotate mutation operators will lead to the best improvement over the standard mutation.

1. INTRODUCTION

Musically meaningful genetic operators have been a part of evolutionary composition for over a decade despite a dearth of experimental data to justify their use. We investigate the effects of four musically meaningful mutation operators (MMMOs) on populations of simple rhythmic patterns. The goal of this project is to determine the individual effects of these MMMOs on population fitness and provide a numerical basis for justifying their use in evolutionary composition of rhythms. The performance of each MMMO is judged in relation to a standard mutation operator.

The remainder of this paper is organized into five sections: a background on MMMOs, the approaches used in this research, the results of these approaches, and a discussion of the results. The discussion of results is followed by our conclusions and a summary of future work.

2. BACKGROUND

The introduction of MMMOs to the field of evolutionary composition was first made in Biles' "GenJam" [1], although that research did not conclusively state the impact of this new breed of mutations on population fitness. The theory behind meaningful genetic operators extends back to the original research on reordering operators summarized in [4]. Most of the research from this time manifested as mutation-inspired crossover operators rather than standalone mutation operators. In relation to music, operators are considered "meaningful" when their algorithms take into account the musical semantics of the input representation as well as principles of music theory. The purpose of these knowledge-based operators is to evolve better individuals faster than "dumb" classic operators [1].

Biles defines mutation operators for populations of measures including reverse, rotate right, invert, sort notes ascending, sort notes descending, and transpose notes. At a second level composed of measure-based phrases, he implements mutators such as genetic repair, super phrase, lick thinner, and orphan phrase. These operators attempt to maintain diversity and quality in the phrase populations by weeding out poor or frequently used measures and building new phrases from the best or infrequently used measures. The two most successful mutators replace existing measures with the least frequently occurring measures in the entire population [1].

Papadopoulos and Wiggins [6] describe significant improvement in the evolution of melodic patterns using a restricted copy mutation which copies melodic fragments of a constant size to metrically logical locations elsewhere in the piece. The impact of this mutation is limited by the number of valid locations to which a fragment can be copied. Thus, the copy mutation could potentially reduce the population diversity by creating many similar mutants.

Tokui and Iba [7] describe a MMMO, Timbre Exchange, which exchanges timbres amongst instruments within an individual. The same research also uses the rotate and reverse operators inspired by [1]. The impact of these mutators on population fitness or diversity is not addressed.

Dostál's [3] four MMMOs accentuate quiet notes that match strong rhythms in a piece of source music, syncopate strong beats, randomly change the type of notes played in a given measure, and even rewrite rhythms to match the source music. The operators are the sole basis of recombination and mutation and without their influence a population of rhythms cannot achieve successful results.

3. METHODS

Throughout this paper the term "gene" is used to describe a boolean value indicating the presence or absence of a musical beat. The terms "mutator" and "MMMO" refer to a "musically meaningful mutation operator".

To isolate the effectiveness of the different operators from potential problems caused by an exponentially

complex search domain, this research focuses solely on the generation of rhythmic music. A genetic individual is represented by a two-dimensional matrix or *pattern* with a cell, $p[x][y]$, denoting the beat value of instrument x at time y in pattern p as in [7]. The slice $p[x]$ represents an instrument part or row.

Fitness evaluation is loosely based on Birchfield's multi-level system of user-defined fitness trajectories [2] and consists of a level of columnar and row trajectories defined over each pattern. Columnar trajectories are defined by cubic Bézier curves and can be smoothly connected between patterns by setting the initial value of a pattern's trajectory to the final value of the same trajectory in the previous pattern. Row trajectories are defined by the rows to be affected by the trajectory's rule.

A pattern's fitness is the sum of the squared differences between its trajectories' expected values and the pattern's actual values and is, thus, to be minimized. The current population is defined as $P(t) = m_{\text{rand}}(c(s(P(t-1))))$ for a selection function s, a crossover operator c, and a randomly mutator m_{rand} selected from the set of available MMMOs (Table 1.).

- **Classic:** Mutates a randomly selected gene or randomly mutates all genes (S)
- **Invert-All:** Exchanges an individual's rests for beats and its beats for rests (L)
- **Invert-One:** Invert the values of one instrument row (M)
- **Reverse-All:** Reverse the temporal order of values in all instrument rows (L)
- **Reverse-One:** Reverse the temporal order of values in one or more instrument rows (M)
- **Rotate-All:** Rotate the values of one or more instrument rows to the right by a specified or random amount (L)
- **Rotate-One-N:** Rotate the values of a single instrument row by n time units (M)
- **Rotate-One:** Rotate the values of a single instrument row by 1 time unit (S)
- **Timbre-Exchange:** Exchange instrument parts between two randomly selected instruments (S).

Table 1. Mutator descriptions followed by impact classifications (Small, Medium, and Large)

Each mutator affects the individuals it mutates to a different degree. The degree of change, or "impact", a mutator affects on an individual can be quantified by the percentage of genes whose locus or value has changed. The Invert mutator, for example, has an impact of 1 because it swaps all beats and silences in a pattern which affects the entire pattern. In contrast, the Classic mutator performs a random mutation of each gene based on the gene's own mutation probability so its impact is p_m where p_m is the probability of mutation of each gene. To allow control over the use of mutators with different impacts, each mutator is intuitively classified into one of three groups: "small", "medium", and "large" impact (see Table 1.).

Mutator	Example
None	01101111
Classic	0**0**1011**0**1
Rotate Right 1	10110111

Table 2. Examples of two mutation operators

One assumption made in this research is that the closer an individual gets to the global optimum, the less important it becomes to stimulate large scale diversity within it. If a solution stagnates around a local optimum, a dramatic mutation may introduce the necessary diversity for it to move on. However, if an individual is already quite close to a global optimum, a major mutation will destroy generations of quality building blocks. For this implementation, limits on mutators of each impact type classified above (Table 1.) are hard-coded such that large impact mutators cannot be used on an individual when the individual's raw fitness value is less than 30 and medium impact mutators cannot be used when that value is less than 10. The goal of this annealing configuration is to protect quality solutions from overly dramatic mutations.

4. RESULTS

Each experimental trial was run with a mutation probability of 0.02, a crossover probability of 0.6, an initially random population of 200, and a population of 20 at the end of each generation. At the beginning of each generation after the first, 20 children were generated from the 20 parents. Unless stated otherwise, all trials were all also tested with two fitness trajectories: temporal beat density and instrument beat density. Temporal beat density refers to the number of instruments playing simultaneously at any point in time. Instrument beat density refers to the number of total beats played in each instrument row. The former trajectory controls the dynamic of the pattern while the latter trajectory simply limits the complexity of any one instrument pattern.

The effectiveness of the MMMOs was tested by pairing each one with the Classic mutator. These mutator pairs were used to evolve 16 beat, 8 instrument (16×8) patterns and 16 beat, 16 instrument (16×16) patterns over 25 individual trials of 50 generations for 16×8 patterns and 80 generations for 16×16 patterns. For these preliminary experiments, these pattern dimensions were estimated as common dimensions for practical use. The measurement of success was the average of the best fitness value for each generation over all 25 trials as well as the average of the average population fitness per generation. Finally, all of the mutators were tested together under the same conditions. The effectiveness of each group of mutators was judged by its performance relative to the standard mutation.

A similar set of trials to the above mutator tests were run in which 25 different populations were evolved with and without mutator limitations. However, instead of stopping after a fixed number of generations, the system

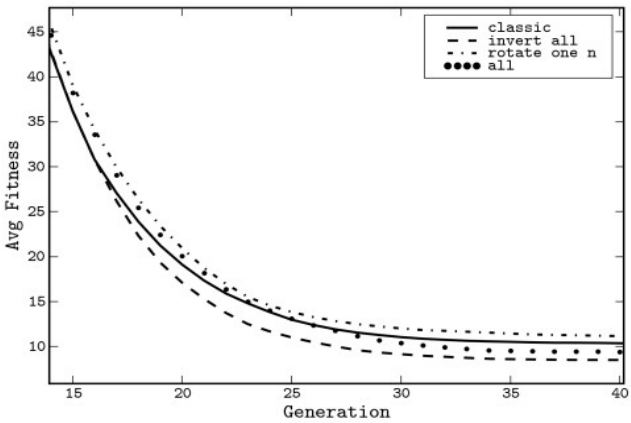

Figure 3. Mutator comparison on 16×8 patterns

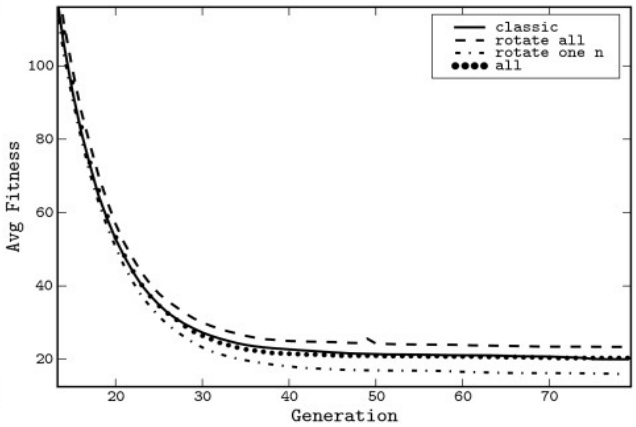

Figure 4. Mutator comparison on 16×16 patterns

Mutator	Best	Avg.	Bests Classic
Invert-All	8.20	8.36	Y
Rotate-All	7.97	8.44	Y
Rotate-One	8.63	9.03	Y
All mutators	8.70	9.3	Y
Reverse-All	9.59	9.67	Y
Timbre-Exchange	9.96	10.04	-
Classic	**9.87**	**10.04**	**-**
Reverse-One	9.82	10.34	N
Invert-One	10.01	10.37	N
Rotate-One-N	9.92	10.98	N

Table 4. Fitness values for last generation of 16×8 patterns sorted by average population fitness

Mutator	Best	Avg.	Bests Classic
Rotate-One-N	14.27	15.85	Y
Invert-All	18.04	18.28	Y
Invert-One	18.04	18.36	Y
Classic	**19.63**	**19.96**	**-**
Rotate-One	18.97	20.24	N
Timbre-Exchange	19.98	20.27	N
All mutators	17.99	20.32	N
Reverse-One	21.02	22.11	N
Reverse-All	22.11	22.35	N
Rotate-All	22.99	23.34	N

Table 5. Fitness values for last generation of 16×16 patterns sorted by average population fitness

was configured to stop after the best fitness value remained unchanged for ten generations. The total time required to perform the 25 trials was recorded to determine the difference between average time per trial for trials with and without mutator limitations.

5. DISCUSSION

The limited/unlimited mutator trials revealed no difference between limited and unlimited use of large and medium impact mutators. In both sets of 25 trials, the tenth consecutive unchanged population fitness value was reached in the same amount of time with nearly the same best fitness value for the final generations. One explanation for the similar behavior between limited and unlimited mutator populations is the relative simplicity of the mutations on the test hardware. When tested on a slower system, each generation takes minutes to evolve instead of seconds. In such an environment, the limited mutators would benefit from performing fewer complex computations. Another explanation for the similar behavior is that each population adapts to disadvantageous mutations to good individuals by keeping enough similar individuals in the population to allow the destruction of one by improper mutation.

For the smaller 16×8 patterns, the Invert-All, Rotate-All, Rotate-One, and Reverse-All mutators performed better paired with the Classic mutator than the standalone Classic mutator. The trials using all mutators also outperformed the Classic mutator. However, all mutators used together did not perform better than Invert-All, Rotate-All, and Rotate-One mutators paired with the Classic mutator. This suggests that certain mutators detract from overall performance.

For the larger 16×16 patterns, both Invert mutator variants and the Rotate-One-N mutator outperformed the Classic mutator. In these trials the combination of all mutators performed worse than the Classic mutator. Interestingly, the best mutator for 16×16 patterns (Rotate-One-N) was the worst mutator for 16×8 patterns. The Invert-All mutator was the only meaningful mutator to consistently improve fitness for both pattern sizes.

The results of the 16×8 and 16×16 trials (Tables 4 and 5) show little consistency between successful mutator combinations for the different pattern sizes. The worst mutator for the 16×8 patterns, Rotate-One-N, is the best mutator for the 16×16 patterns. Similarly, the single-row invert mutator performs poorly on 16×8 patterns while besting the Classic mutator on 16×16 patterns. An obvious solution to this drastic difference in mutator performance for different pattern sizes is to use all mutators and let the GA adapt to pattern size accordingly. Unfortunately, the trials using all mutators on 16×16 patterns reveal no improvement over the Classic mutator by itself. A lack universal improvement regardless of pattern size cannot justify the computational complexity of the non-Classic mutators. An approach for future work would be to experiment with different combinations of the most successful mutators for both pattern sizes to see if any combinations would perform better than all mutators together.

The addition of domain knowledge to genetic operators in the form of rules should cause the GA to search a more constrained solution space. Thus, the GA with domain knowledge is not expected to find solutions that the standard GA cannot also find, but it could be expected to find those same solutions in less time. However, the more domain knowledge integrated into the operators, the smaller the solution space becomes. If domain rules are not prerequisites for evolution of a valid phenotype, they could exclude original solutions that might have otherwise evolved by chance.

6. CONCLUSION

We have tested the effects of four types of musically meaningful mutation operators in the domain of rhythmic music with eight mutator variants. The performance of these operators has been compared to that of a standard mutation operator to determine if the use of MMMOs is associated with an improvement in overall population fitness.

Each of the eight mutation operator variants was paired with the Classic mutator and populations were evolved in 25 independent trials for 16×8 and 16×16 patterns. An additional set of 25 trials using all meaningful mutator variants and the Classic mutator was run for both pattern sizes.

Based on the results from trials for both sizes, the best combination of mutators includes all of the Invert and Rotate mutator variants. The Reverse mutators do not lead to a significant improvement in smaller patterns and they perform worse than the Classic mutator for larger patterns. This latter case is most likely due to the effects of the limited mutations rules. For larger patterns the initial fitness values will be much higher than those of the smaller patterns. Thus, the large impact mutators are used in the evolution of larger patterns for at least 5 more generations than in the evolution of small patterns. The longer these poorer mutators are allowed to run, the greater their effect on overall fitness.

Despite the improvement achieved by some MMMOs over the standard operators, the best improvements do not lead to drastically better solutions than the standard mutation operators within the scope of the current experiments. The Rotate-One-N mutator improves on the Classic mutator by 20% for 16×16 patterns. The next best improvement by the Invert-All mutator is only 8%. Correspondingly for the 16×8 patterns, the best and next best improvements over the standard classic mutation are 17% and 16% with Invert-All and Rotate-All respectively.

These improvements, while insignificant in the context of this research, could lead to drastically better solutions for more difficult problems. There is not solid evidence to support a claim that improvements due to MMMOs scale linearly with an increase in pattern size. Future research in this area will need to explore the scaling of improvements due to meaningful operators in a more complex search domain to determine if the results presented here are truly universal.

Future work will include the implementation of more fitness functions, replacement of Bézier curves with NURBs, and a redefinition of fitness calculation in terms of Pareto-optimal comparisons.

7. REFERENCES

[1] Biles, John A. "GenJam: A Genetic Algorithm for Generating Jazz Solos". Available: http://www.it.rit.edu/~jab/GenJam94/Paper.html, 1994.

[2] Birchfield, David. "Generative Model for the Creation of Musical Emotion, Meaning, and Form", *Proceedings of the 2003 ACM SIGMM workshop on Experiential telepresence*, 2003, 99-104.

[3] Dostál, Martin. "Genetic Algorithms as a Model of Musical Creativity - On Generating of a Human-like Rhythmic Accompaniment". *Computing and Informatics* 22, 2005, 1001-1020.

[4] Goldberg, David E. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading: Addison-Wesley, 1989.

[5] Horowitz, Damon. "Generating Rhythms with Genetic Algorithms", *Proceedings of the 1994 International Computer Music Conference*. Aarhus, Denmark: International Computer Music Association, 1994.

[6] Papadopoulos, G and Wiggins, G. "A Genetic Algorithm for the Generation of Jazz Melodies." *Proceedings of STeP 98*, Jyväskylä, Finland, 1998.

[7] Tokui, N. and Iba, H. "Music Composition with Interactive Evolutionary Computation". Third International Conference on Generative Art, 2000.