

COMPOSING FIFTH SPECIES COUNTERPOINT MUSIC WITH A GENETIC ALGORITHM

Gleb Mineev

g.mineev@students.uu.nl

Jinfeng Guo

j.guo3@students.uu.nl

Maguell Sandifort

m.l.t.l.sandifort@students.uu.nl

ABSTRACT

In our study, we implemented the fifth species counterpoint generation using a genetic algorithm. We tested our results against human composed, generated with another counterpoint generation application, called Optimuse, and randomly generated fifth species counterpoint music with the same cantus firmus. During the user tests participants are asked to make preferences while listening to audio pairs of the three combinations. Test results show that the fifth species counterpoint implemented using genetic algorithms gives plausible results. Furthermore, our results are deemed as good as, if not better than, Optimuse by our users.

1. INTRODUCTION

In music theory, counterpoint consists of the matching of one melody to another melody, or multiple other melodies. While there are different kinds of counterpoints, such as the free counterpoint and contrapuntal derivations, this research will focus on the more strict species counterpoint. There are five species of counterpoint: first through fifth species. Examples can be found in appendix A. Formal rules for counterpoint composition can be dated back to as early as the eighteenth century, as J.J. Fux published *Gradus ad Parnassum* in 1725 [2].

With species counterpoint the composer is given a set of rules to create a second melody, or voice over a given basic melody called the cantus firmus. There are different types of voices, called species, each of which has a set of rules on how they should be based on the cantus firmus. The final fifth species is a combination of pieces from the first four species. The challenge of the fifth species lies in the high dimension of search space, for which more flexible constraints and combinations thereof are placed than the first four.

Our research problem is:

Can we generate fifth species counterpoint using a genetic algorithm?

If so, how does the genetic algorithm perform compared to other approaches?

FOOX is a project that aims at generating counterpoints. The first four species have been implemented using a genetic algorithmic approach. Optimuse aims to generate counterpoint compositions as well, but uses a variable neighborhood search algorithm instead.

With this research, we explored how to implement the fifth species counterpoint in FOOX using a genetic algorithm and compared the results to other types of fifth species counterpoint pieces, namely: human composed counterpoint, randomly generated counterpoint and counterpoint generated through Optimuse.

2. RELATED WORK

From a computational viewpoint, generating counterpoint can be regarded as a search problem: for a given melody, find a suitable counterpoint melody from the space of all possible melodies.

Farbood *et al.* [1] proposed a probabilistic Markov Chain approach to generate counterpoint. They show that a Markov Chain approach cannot only generate counterpoint music according to the existing rules for a given melody, but also infer certain rules given a melody and a piece of counterpoint.

Herremans *et al.* [3] proposed a method for fifth species counterpoint generation, based on the variable neighborhood search algorithm. The algorithm presented in this paper is an improvement on their algorithm for first species counterpoint generation [4]. Their method takes into consideration most constraints for the fifth species counterpoint, and produces satisfying results. Also, they mention that their method outperforms genetic approach for first species, and that genetic algorithm is tricky to implement for the fifth species due to irregular rhythmic patterns.

Komosinski *et al.* [5] proposed a dominance relation based method to generate the first species, although constructing fifth counterpoint music was not their goal, this never-so-far-used method might be promising to look into.

Other than the methods mentioned above, most research, including FOOX, focusses on using genetic algorithms to generate counterpoint, which will be our focus as well.

3. METHOD

In our research, we will be working on FOOX to implement the fifth species counterpoint. An example of generated fifth species counterpoint can be found in the on-



© Gleb Mineev, Jinfeng Guo, Maguell Sandifort.

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Gleb Mineev, Jinfeng Guo, Maguell Sandifort. "Composing fifth species counterpoint music with a Genetic Algorithm", 16th International Society for Music Information Retrieval Conference, 2015.

line presentation of FOOX [7]¹. We implemented the fifth species counterpoint using a genetic algorithm. Two user tests were run in this research. Optimal parameter sets were determined by doing a listening test. Finally, we tested our result against human composed, randomly generated and Optimuse generated results. In the first user test, varying weights are tested against one another to determine what the best weight distribution is. In the second user test, the optimal weight distribution is used to generate pieces that are then compared to human composed pieces, pieces that have their counterpoint voice randomly selected and pieces generated by Optimuse.




The following sections will give details on the method and our user tests.

3.1 Generation algorithm

For generating fifth species counterpoint compositions we utilize a genetic algorithm, used by FOOX to generate the first four species, together with a fitness function and method of rhythmic pattern generation, which was proposed by Herremans *et al.* [3].

3.1.1 Producing rhythmic pattern

To form a population we first calculate an appropriate rhythmic pattern, which will be common for every chromosome in the population. This approach allows us to simplify crossover and mutation operators, making sure that they will not accidentally produce fragments with an unfeasible rhythm. Generated rhythmic patterns should satisfy the following rules:

- The first measure is 
- The penultimate measure is 
- The last measure is 
- Rhythmic patterns for other measures are selected from the patterns, proposed in [6] (Table 1)
- No two similar measures in the row are allowed (no more than two, in case a measure consists of two half notes or four quarter notes)
- When deciding whether two measures are similar or not, we consider two eighth notes as one quarter note.

When a rhythmic pattern is generated we convert it into a list of note durations. As two tied notes in the fifth species counterpoint always share the same pitch, we represent them as a single duration in the list, to ensure their pitches do not become different during crossover.









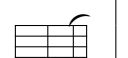
		
		
		

Table 1. List of possible rhythmic patterns

3.1.2 Generating initial population

The initial population is created based on the rhythmic patterns. For each duration we select a random note that satisfies the following criteria:

- All notes are higher than the corresponding cantus firmus note.
- For the first note the vertical interval is unison, fifth or octave.
- For the last note the vertical interval is unison or octave.
- Notes on the first beat are vertically consonant.
- Eight notes must move in step.
- The ending is a dissonant suspension.

Using these criteria the initial population can be generated, which has a better average score than a purely random population and yet remains quite diverse to ensure that the genetic algorithm will not get stuck in local minima too quickly.

3.1.3 Mutation and crossover operators

For the crossover operation we use a standard single-point crossover, as implemented in FOOX. Mutation is done by iterating over the chromosome, and changing each single note with a certain probability, ensuring this new note stays within *mutation_range* tones away from the original note. All criteria from the previous section also hold for this new note with the exception of moving in step of eight notes.

3.1.4 Fitness function

In our algorithm we use a slightly modified fitness function from [6]. For horizontal rules we use all rules except for rules 13 (*the ending note is tonic*) and 14 (*the penultimate note is a leading tone*), as for the cantus firmus that ends with a note that is different from tonic these rules would contradict with vertical rule 10 (*the ending is unison or octave*). In addition we combined vertical rules 3 and 4 into a single rule (*halves and quarters should be consonant on the first beat or can be dissonant on other beats, if passing tone*), and omit several rules that are already ensured from generation, since these would always give a score of 0.

¹ <https://github.com/ntoll/foox>

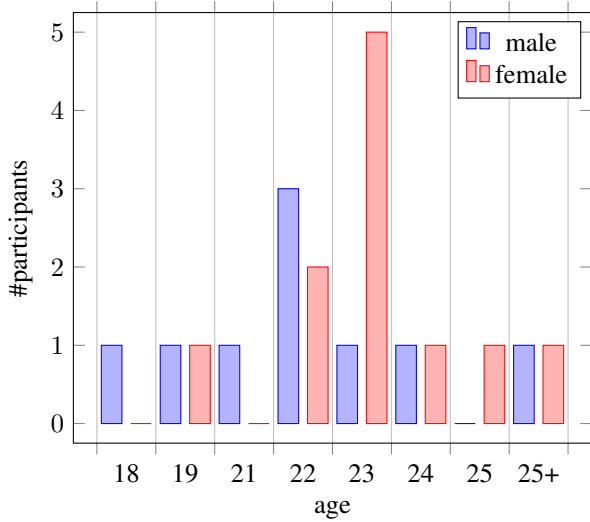


Figure 1. Age and gender distribution of the 20 participants of test 1.

As in the original fitness function there are weights for each rule, that allow emphasis on them during the generation. For the sake of simplicity, when referring to the rules or their weights the original rule numbers from [6] will be used.

3.1.5 Creating next generation

The next generation consists of three equal parts. The first part is the top 1/3 of the previous generation, the second part consists of non-mutated children, and the third part consists of children after mutation.

To select two chromosomes for crossover a roulette wheel selection is used, with the probability of a chromosome being chosen set according to its normalized fitness function. Normalization is done as follows. The minimal and maximal fitness value in the population is calculated. Then the normalized fitness value are calculated as follows:

$$fitness_{norm} = \frac{fitness - fitness_{min}}{fitness_{max} - fitness_{min}}$$

Normalization allows us to use a similar selection process for both FOOX' fitness functions (for which highest score is better) and our fifth species fitness function (lower is better), and also leads to better results.

3.2 Participants

For both test, the participants consisted of fellow students, friends and family members. The first test had 20 participants and the second test had 30 participants. The age and gender distribution for both tests can be found in tables 1 and 2. A participant of the first test could also be a participant for the second test. Musical knowledge level distribution for participants can be found in B. For the second test, participants that did the test under supervision were given some cookies for their trouble. The effects of cookies on the results is not evaluated.

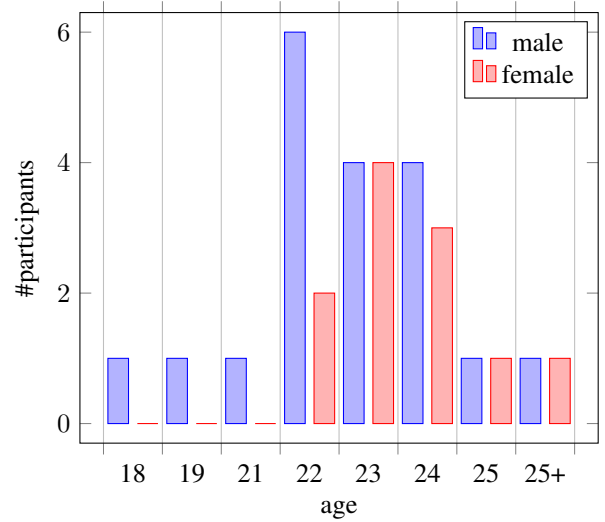


Figure 2. Age and gender distribution of the 30 participants of test 2

3.3 Material

In order to run the user tests, a digital questionnaire was created for each of the tests through Google forms. For both tests, the user is asked to provide their gender and age. They are also asked about their level of musical knowledge.

For the first test, there are initially 9 parameters to tune and we finally managed to narrow it down to 3 sets of 5 counterpoint pieces by grouping possibly correlated parameters. All pieces are generated by the algorithm. Each of the 5 pieces in each set uses a different weight distribution. High H5 weights (predominately step-wise movement), high H18 weights (avoids repetition of sequences; groups of at least two notes), equalized weights, high vertical weights (focus on harmonic rules) and high horizontal weights (focus on melodic rules). The sets used can be found in table 2.

Set	1	2	3
H5	1	3	7
H18	2	8	9
EQUAL	4	5	6
Vert	a	d	e
Hor	b	c	f

Table 2. The sets used for usertest 1.

For the second test, 2 sets are created. Each set consists of 3 groups, each having 3 pairs: vs human composition, vs random and vs Optimuse. For the human composition the CF is used to generate fifth counterpoint music. For the random composition, the CF that is generated is used for both the random composition and the algorithm composition. For Optimuse the same CF is used as for the algorithm as well. The sets can be seen in table 3.

Category	Pair	Ours	Pair	Ours
Ours vs Human	1	B	1	B
	2	A	2	A
	3	B	3	B
Ours vs Random	4	B	4	B
	5	A	5	A
	6	A	6	A
Ours vs Optimuse	7	B	7	B
	8	B	8	A
	9	A	9	B

Table 3. The sets used for usertest 2

3.4 Procedure

For the first test, each participant was told they were to listen to a set of 5 MIDI files, which they had to order from most to least liked. They would then fill in a questionnaire afterwards. At the end they had the possibility to give comments about the test or the music pieces. The participants were given as much time as they needed and could listen to the files in any order, as long as they would like. Each participant was given one of the three sets randomly, in such a way that the amount of tests for each set were distributed as equally as possible. Tests were held at different locations. All participants were given instructions. Part of the participants did the test supervised, while others did it through the form on their own.

For the second test, each participant was told they were to listen to nine pairs of audio files, where for each pair they had to decide which of the two they liked more and to what degree. Aside from being able to comment at the end of the test, participants could now also comment on each individual piece. The participants were given as much time as they needed. The pairs were presented in a set order, but participants could backtrack to previously heard pairs if they wanted to. No time limit was given. Each participant was presented one of the two sets randomly, in such a way that the amount of tests for each set were distributed as equally as possible. Tests were held at different locations. All participants were given instructions. Part of the participants did the test supervised, while others did it through the form on their own.

4. RESULTS

Results from the first test consist of a ranked order of 5 musical pieces, each generated through different weights. This ranking order is then converted to a score for each piece. If a piece is chosen as best out of the five, it will be given a score of 5, if chosen as best out of the five, it will be given a score of 4 and so on. The lowest score is 1, for the piece that was placed last in the ranking order. For each piece, the individual scores are added together to create a total score, which is then divided by the amount of individual scores, resulting in a final normalized score. The results can be seen in table 4 and 9.

H5 and equal weight parameters are the highest scoring

V	2.86	H5	2.76	EQ	3.22
H5	2.48	H18	2.48	H	2.00
EQ	2.10	V	1.71	H5	1.67
H	1.33	EQ	1.52	H18	1.56
H18	1.24	H	1.52	V	1.56

Table 4. Normalized results after ranking test 1

Nr.	1	2	3	4	5	6	7	8	9
A	-6	-3	9	14	15	9	0	-1	-4
B	1	-8	0	12	12	18	-2	1	-4

Table 5. Total results after ranking test 2

weights sets. EQ is chosen as these have a better fitness function.

For the second test, the results consist of 9 scores, indicating which of the two pieces in each pair is considered better by the participant. These scores were converted to a number from -2 to 2. A score of 2 is given if the participants considers the algorithm generated piece to be much better than its opposition, and a score of 1 if it is slightly better. Likewise, a score of -2 is given if the participants considers the opposing piece to be much better than the algorithm's one, and a score of -1 if it is slightly better. Finally, a score of 0 is given if the pieces were deemed equal. The results can be seen in table 5 and 6.

Additionally, the scores of low musical knowledge participants (score 1 or 2) are compared to the scores of high musical knowledge participants (score 3, 4 or 5). The results of this can be seen in table 7 and 8.

5. DISCUSSION

What is interesting to see in the results of the first test is that none of the weight preferences scored consistently as the lowest parameter and scores vary over different test sets. This could indicate that, while weight choices do seem to affect whether a piece is considered likable or not, some lower quality weight scores can still create a piece that is likable. Due to the random nature of generating the pieces, it is possible that a certain composition had a "bad run", despite being a higher quality weight score set. While the ordering does give an indication of which weight scores work better than others, it is hard to determine how much better they work, as there is no measure of "scoring distance" between the two. None of the participants in the first test claimed to be a musical expert and none of them knew how to compose counterpoint compositions, or knew what they were. Perhaps one who does would have a dif-

Nr.	1	2	3	4	5	6	7	8	9
A	-0.4	-0.2	0.6	1	1	0.6	0	-0.1	-0.3
B	0.1	-0.5	0	0.8	0.8	1.2	-0.1	0.1	0.3

Table 6. Normalized results after ranking test 2

Nr.	1	2	3	4	5	6	7	8	9
High	-0.4	-0.7	0.3	1.3	1.1	0.7	-0.2	-0.3	-0.8
Low	-0.3	0.5	1	0.3	0.8	0.5	0.3	0.3	0.5

Table 7. Normalized results of set A high vs low knowledge

Nr.	1	2	3	4	5	6	7	8	9
High	0.5	-0.9	0	0.9	1.2	1.4	-0.2	0	-0.1
Low	-0.8	0.2	0	0.6	0	0.8	0	0.2	-0.6

Table 8. Normalized results of set B high vs low knowledge

ferent insight into what weight sets score more favorably.

Considering the results of the second user test, the algorithm seems to consistently outperform the randomly generated counterpoint pieces. Most users detected irregularities or false notes in the random pieces, causing them to favor the algorithm pieces over the random ones. For human composers there are some pieces where the algorithm can run similar or outperform humans, probably because composers tend to use repetitive consonant patterns and our program generates a bit more of a cheerful counterpoint. Where the composer’s idea is clear, however, they still outperform the algorithm. The algorithm cannot yet replace a human composer. For short pieces the algorithm performs similar to Optimuse, but for longer pieces Optimuse performs slightly better. One thing to note is that by reducing the size of the population to about 400 chromosomes it is possible to produce results faster with the presented algorithm than with Optimuse, without a significant degradation in the quality of music pieces.

Splitting the participants in low (LK) and high (HK) knowledge participants also provides some insight. For human compositions, LK scored our generated pieces somewhat higher than HK in set A, but there was no relation in set B. With random compositions, HK scored ours consistently higher than LK. With Optimuse, LK scored ours higher than HK on set A and slightly higher on set B. LK participants seem to be more hesitant in their scoring.

A thing to note is that the three different comparisons were always done in the same order. After the human composed ones, they are always presented with the random ones, which a lot of users thought sounded a lot worse than the human composed ones. After the random ones came the Optimuse comparisons, which were slower paced and considered to sound better by most participants than the random ones. This order may have biased the scores the participants have given. For example, after listening to three relatively bad pieces with a lot of different notes being used, they are then presented with a slower paced, more structured piece, perhaps causing them to score these higher than if they had not heard three potentially lesser pieces beforehand. For future work, it might produce different results if the different comparisons would be shuffled.

	H5	H18	EQ	V	H
Test1	2.48	1.24	2.10	2.86	1.33
Test2	2.76	2.48	1.52	1.71	1.52
Test3	1.67	1.56	3.22	1.56	2.00
Total	6.91	5.28	6.84	6.13	4.85

Table 9. Total scores of test 1

6. CONCLUSION

With the the discussions above, we would like to conclude that genetic algorithms can in fact be used to implement fifth species counterpoint. As for performance, our method gains an obvious advantage compared to randomly generated pieces, but loses to human composed counterpoint in most cases. In comparison with Optimuse, we show equal performance for short pieces, but less good performance for longer pieces.

7. ACKNOWLEDGMENT

This project is the course project for Sound and Music Technology, taught by Anja Volk, Utrecht University, the Netherlands. During the project, we get invaluable guidance from the teacher and we would like to thank her this. We also want to thank Dorien Herremans, Queen Mary University of London, for giving us the source code for Optimuse, which makes our comparison possible.

8. REFERENCES

- [1] Mary Farbood and Bernd Schoner. Analysis and synthesis of palestrina-style counterpoint using markov chains. In *Proceedings of the International Computer Music Conference*, pages 471–474, 2001.
- [2] Johann Joseph Fux and Jean-Philippe Navarre. *GRADUS AD PARNASSUM (1725): TEXTE ORIGINAL INTEGRAL/ED. BILIN. FRANÇAIS-LATIN*, volume 1. Editions Mardaga, 2000.
- [3] Dorien Herremans and Kenneth Sörensen. Composing fifth species counterpoint music with a variable neighborhood search algorithm. *Expert systems with applications*, 40(16):6427–6437, 2013.
- [4] Dorien Herremans, Kenneth Sörensen, et al. Composing fifth species counterpoint music with variable neighborhood search. Technical report, 2012.
- [5] Maciej Komosinski and Piotr Szachewicz. Automatic species counterpoint composition by means of the dominance relation. *Journal of Mathematics and Music*, 9(1):75–94, 2015.
- [6] Felix Salzer and Carl Schachter. *Counterpoint in composition: the study of voice leading*. Columbia University Press, 1969.
- [7] Nicolas Tollervy. Music theory - genetic algorithms and python, 2012.



Figure 3. Example of fifth species counterpoint composed by human



Figure 4. Example of fifth species counterpoint generated by our algorithm

Appendices

A. EXAMPLES OF FIFTH COUNTERPOINT MUSIC

See Figures 3 and 4.

B. MUSICAL KNOWLEDGE OF PARTICIPANTS FOR BOTH TESTS

See Figures 5 and 6.

C. FEW HIGHLIGHTS OF USER COMMENTS DURING USER TEST 2

- "First piece had a lot less going on than the second." — Set A Pair 3
- "B sounds like there are a lot of false notes." — Set A Pair 5
- "Didn't like either of them, A was less bad than B." — Set A Pair 6
- "I like these ones a lot. There actually seems to be some progression in them. Also B is a bit heavier/more emotional." — Set B Pair 7
- "I liked the part where it went 'Dee dee boodoop da'." — Overall comment
- "The overall impression is positive. However, some compositions are 'annoying'" — Overall comment
- "The music all sounds like it has been composed locally; short pieces sound okay, but there is no overall structure; no recurring themes..." — Overall comment

D. SOURCE CODE

The source code of our project can be easily accessed through <https://git.io/vo7oZ>.

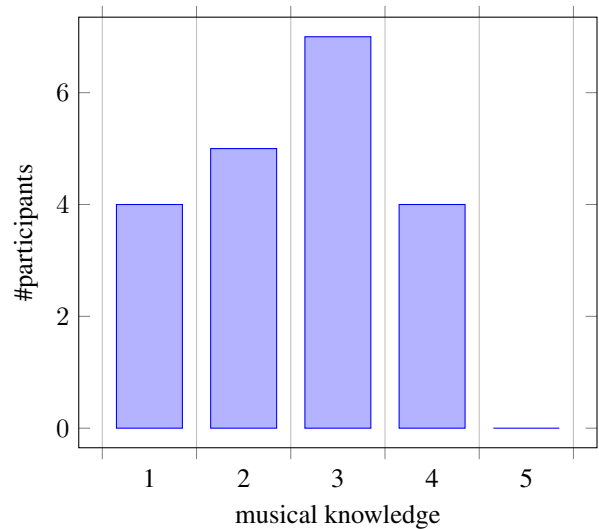


Figure 5. Musical knowledge distribution of the 20 participants of test 1, from 1 (*know nothing*) to 5 (*expert*)

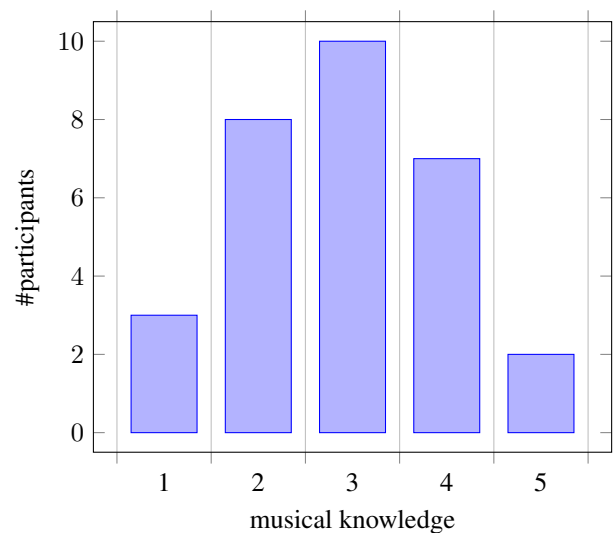


Figure 6. Musical knowledge distribution of the 30 participants of test 2, from 1 (*know nothing*) to 5 (*expert*)