Style Composition With An Evolutionary Algorithm

Zeng Ren

New York University
zr372@nyu.edu

Abstract. This paper presents a general-purpose symbolic music generation framework based on an evolutionary algorithm. It aims to work for any given musical fitness that is definable in a similar way as the ruleset for 18th century counterpoint. This framework is demonstrated through several composition scenarios in the style of 18th century four-part first species counterpoint. The scenarios include free and constrained composition tasks such as voice/part completion and melody harmonization.

1 Introduction

With the advancement in predictive models, style imitation tasks have gained much interest in recent years in the field of algorithmic composition. In contrast, tasks involving solving arbitrary music composition objectives have not received as much attention.

Pearce, Meredith, and Wiggins (2002) identify four main motivations for developing computer programs which compose music:

1. Expansion of compositional repertoire
2. Development of tools for composers
3. Proposal and evaluation of theories of musical styles
4. Proposal and evaluation of cognitive theories of musical composition

Under this framework, the first group includes corpus based style imitation models with various predictive architectures, from markov chains (Farbood & Schöner, 2001; Pachet & Roy, 2011) to artificial neural networks (Liang, 2016; Yang, Chou, & Yang, 2017; Huang, Hawthorne, et al., 2019; Huang, Vaswani, et al., 2019). The second group includes models that give us better control over the music (Hadjeres, Pachet, & Nielsen, 2016; Engel et al., 2017; Dhariwal et al., 2020).

In this paper, the motivation of the proposed composition model is to generate music according to definable musical objectives. This model can be used both as a composition tool and a simulation tool to verify theories of musical styles. Music composition is thus modeled as an optimization problem with musical objectives expressed as a real valued function in $[0, 1]$. The model's architecture is based on evolutionary algorithms (Beyer & Schwefel, 2002), a family of meta-heuristics that has worked well on simple melody search. The user inputs a fitness function together with a music template that represents positional constraints such as fixing the top voice as given, or having a two beats pick-up measure at the beginning. The algorithm will then output a list of candidate solutions that try to optimize the provided fitness function within the constraints of the provided music template.

2 Related Work

In most western polyphonic music, we see the idea of simultaneously working with multiple constraints in different levels. Species counterpoint is the most fundamental representation of this idea where the constraints require composers to think locally and globally, horizontally and vertically, harmonically and rhythmically. From a computational point of view, this is a hard combinatorial problem with vast search space and discontinuous fitness landscape. Thus, metaheuristics, in particular genetic algorithms, are commonly employed for such task. Solving counterpoint-related tasks including full counterpoint composition, harmonization, and figured bass, is the primary focus for many previous works using genetic algorithms. Since I am designing a framework that is supposed to work for arbitrary fitness function, here I will focus on approaches that are not corpus based.

McIntyre (1994) uses a genetic algorithm (GA) to generate short four-part baroque harmonies. When combining several small fitness functions into a large one, McIntyre adopts a three-tiered design to reflect their varying level of musical importance. Phon-Amnuaisuk et al. (2000) experiment with a GA for harmonization and point out the inherent difficulties, primarily the non-local nature of the fitness function and the lack of structured reasoning. They suggest that future works could consider more knowledge-rich mutations, especially those could "leap directly across the barriers in the fitness landscape." Donnelly and Sheppard (2011) design an GA to compose four-part harmony by expansion starting from a single chord. In dealing with a large pool of mutations, they make the probability distributions of mutations change dynamically by encoding them as part of the chromosome. Herremans and Sörensen (2013) use a variable neighborhood search algorithm to optimize two-part fifth species counterpoints. The fitness function is a weighted sum of horizontal and vertical evaluations. The algorithm uses three types of moves that are equivalent to the swap and point mutation in classic genetic algorithms.

Looking at the previous works, I conclude that the main difficulties in using genetic algorithms to optimize a given music fitness function include:

1. It is hard to combine sub-evaluations into a fitness function without introducing unwanted tradeoffs.
2. The fitness landscapes are often filled with local extremes or even discontinuities, making it hard to optimize using neighbor search techniques.
3. Simple genetic operators cannot efficiently explore a complex fitness landscape so we need operators that can "leap" over local extremes.

4. It is hard to design variation operators that can work across different musical fitness functions.
5. Increasing the set of mutations can partially address the first three problems but will increase the number of searches required to properly explore the space. Therefore, we need to know how to choose them in an efficient and effective way.

The main differences between the framework in this this paper and the previous ones are:

1. It employs a hierarchical music representation using n-dimensional arrays unlike the one dimensional chromosome in classic GAs. This, together with the music template, enables an expressive representation of music and musical tasks.
2. It uses a smooth tiered functional form for the fitness function to combine sub-evaluations. This gives us better control over the objective. First, we can describe a more complex preferential logic among the fitness components comparing to the weighted sum design. Second, the activation of each tier has a controllable smooth transition as opposed to the binary jump, enabling the algorithm to gradually prioritize higher tier constraints as the lower tiers' evaluations rise.
3. It uses very few hierarchical variation operators (mutation and crossovers).
4. To guide the probability distribution of the operators, it uses a semi-autonomous *Attention* mechanism that allows human guidance in real-time.

3 Music Encoding

This paper focuses on the symbolic representation of music similar to the conventional staff representation. However, performance related elements such as expressions, dynamic, and articulations will not be considered at the current stage. The proposed representation will preserve **hierarchical rhythmic structures** like voices, measures, beats, and beat divisions. It would also include **rests, and tied notes**. Each piece of music is represented by a 4-dimensional numpy.array of shape

(#voices, #measures, #beats, #beat divisions).

The array is filled with "general note" objects which unify pitches, rests, or tied notes, all encoded in strings. Pitches are represented by MIDI note numbers but shifted so that the middle C (C4) is '0'. Rests are represented by '.' and tied note is represented by '*'. The note duration can be unambiguously inferred from the representation. The hierarchical structures are naturally encoded by the array shapes. An example (see Appendix A) of such encoding is shown in Figure 1. The corresponding music is shown in Figure 2.

4 Problem Formulation

Composing in this context can be framed as an optimization problem

$$\max_{x \in S} f(x) \tag{1}$$

where the search space S is the set of music that fits the provided music template, and the objective function $f : S \to [0,1]$ is crafted based on one's composition goals. When handcrafting a fitness/evaluation function f for musical result, we often want to express it in terms of a function of smaller and easier-to-define sub-evaluations. In other words, it is helpful to formulate f as a composition of two parts: $S \to [0,1]^n \to [0,1]$. The first part is a collection of small functions grouped into a vector-valued function $\mathbf{f} : S \to [0,1]^n$ which evaluates specific aspects of the music such as voice leading quality, contour, and voice range. For the second part, many of the previous works employ a linear combination to combine smaller fitness functions to a big one:

$$f = \mathbf{c}^T \mathbf{f}, \quad \mathbf{c} \in [0,1]^n \tag{2}$$

This strategy may not be able to accurately reflect the intended objective since it may lead to unwanted trade-offs among sub-evaluations. In practice, musical objectives can be dependent among each other; we might only want to consider evaluation z only when evaluation x and y are satisfied to a certain threshold. Based on this need, I advocate for a smooth tiered fitness design.

5 Smooth Tiered Fitness Function

The smooth tiered fitness function (3) contains a list of activations $\mathbf{a} \in [0,1]^n$ of the sub-evaluations $\mathbf{f}$ together with its activation thresholds $\mathbf{s} \in [0,1]^n$.

$$f = \mathbf{c}^T diag(\mathbf{a})\mathbf{f} \tag{3}$$

A tier is activated if and only if the previous tiers have met their thresholds. Formally put, $a_i = 1 \iff f_j \geq s_j, \forall j < i$. The activations $\mathbf{a} = \{\mathbf{a}_i\}_{i=1}^n$ are designed as (4):

$$\mathbf{a}_i = \begin{cases} 1, & i = 1 \\ \mathbf{a}_{i-1}\sigma((\mathbf{f} - \mathbf{s})\mathbf{e_{i-1}}), & 1 < i \leq n \end{cases} \tag{4}$$

where σ is the unit step function. To make f smoother, which is useful for optimization, we can substitue σ by a family of smooth functions σ_w that can approximate the unit step function $\mathbb{1}_{[0,\infty]}$. The construction of the family σ_w can be found in Appendix B.

The sub-evaluation function of each tier can also adopt this multi-tier design, enabling a modular and hierarchical design of fitness function. We can now build a big and complex evaluation from the small ones.

6 Genetic Operators

This algorithm uses two mutation operators and one crossover operator. They are similar to the operators in tree encoding of genetic programming. Mutation and crossover happen not just on the note level, but also on the level of beat, measure, voice, or the whole piece. This reduces the necessary steps required to transform one piece to another via these operations.

A n-dimensional array can be visualized as a hierarchical tree and any sub-tree within it will be referred to as a **hierarchical unit**.

Hierarchical Exchange: Exchange one hierarchical unit between two pieces, thus generalizing the classic single-point crossover for one dimensional chromosome.

Hierarchical Pitch Shift: Uniformly move the pitch of the selected hierarchical unit based on the given distance in semitones, generalizing the classic point mutation.

Hierarchical Neighbor Swap: Switch a pair of "neighboring" hierarchical unit $[x_1, ..., x_k]$ in a piece. The "neighbor" is with respect to the swapping level $l, 1 \leq l < k \leq 4$. The mutated array is defined as:

$$NewArray \leftarrow Array \tag{5}$$

$$NewArray[x_1, ..., x_l \pm 1, ..., x_k] \leftarrow Array[x_1, ..., x_l, ..., x_k] \tag{6}$$

$$NewArray[x_1, ..., x_l, ..., x_k] \leftarrow Array[x_1, ..., x_l \pm 1, ..., x_k] \tag{7}$$

Figure. 3 (see Appendix. C) shows three examples of the Hierachical Neighbor Swap with respect the same unit but with different swapping levels (indicated by red arrows) $l = 3, 2, 1$ respectively.

7 Evolution specification and Implementation

There are two kinds of population involved in this evolution: *Piece* and *Attention. Attention* population guides the variation of *Piece* population by controlling the parameters for the applied mutations and crossovers. In other words, the *Attention* can be considered as a evolving "hyper-parameter" of the model. The fitness of an attention individual is measured by how likely it improves the *Piece* population.

Other than that there are two populations, the rest of the algorithm is similar to a $(\mu/\rho + \lambda)$ evolution strategy (ES). The specific parameters are listed in Fig. 5a and Fig. 5b (see Appendix. C).

The implementation of the algorithm is partially based on *Deap: A Python Framework for Evolutionary Algorithms.* (Fortin, De Rainville, Gardner, Parizeau, & Gagné, 2012). Data representation and computation is done by *Numpy* (Van Der Walt, Colbert, & Varoquaux, 2011). The visualization of music scores is done using *Music21.* The data representation is done by *Matplotlib* (Hunter, 2007).

8 Demonstration and Evaluation

The following scenarios involve various tasks related to the 18th century four-part first species counterpoint. The scenarios test the model's ability on mediating global melodic contour and local intervallic constraints. A tiered fitness function is crafted based on the idiomatic theory in *Counterpoint in Composition: The Study of Voice Leading* (Salzer & Schachter, 1989). In these composition scenarios, the music outputs are evaluated by a range of sub-evaluations including global melody contours, horizontal and vertical intervals, voice-leadings, and cadences. More details about the specific sub-evaluations and their implementations can be found in Appendix F. The scale, which can be arbitrarily specified, is chosen to be C major for the demonstrations in this paper.

This algorithm comes with an interactive interface (Fig. 6, Appendix. C) where the user can see the real-time best piece of the population and guide the attention accordingly.

8.1 Scenario 1: Free Four-Part First Species Counterpoint

In this experiment, the piece is configured to be (encoded in the musical template) note-against-note with a $\hat{5} - \hat{1}$ bass cadential movement. The generated result is shown in Fig. 7 (see Appendix. D).

Analysis The contour of the top is good at the first two measures but there is a leap of M9 in the third measure from G to F. The fitness function does not set the maximum size of leap, which is an octave, so it could be fixed by adding such constraint. The same goes for the jump at bass in measure two beat two. The tenor voice has a unnecessary jump at the second beat of the first measure. It could have held the previous note over. The reason this jump occurs is that it contributes to the "climax arrangement fitness"; it makes the G a climax of the tenor voice, thus spreading out the high points in all voices. This jump could be discouraged by increasing the activation threshold for the last tier "climax arrangement" sub-evaluation.

For voice-leading, there is a hidden fifth between the outer voices such as the start of measure three, and also some involving the inner voices. Those deficiencies are reflected in the fitness report (see Appendix. E). The G at the first beat of the third bar could be changed to E but that gives rise to a M7, indicating that this is a local maximum of the fitness landscape and the model would benefit from improvement on the optimization algorithm.

8.2 Scenario 2: Constrained Composition based on an given piece

In this set of experiments, the algorithm is given an existing piece segment (reduction of BWV 307, see Fig. 4) and it completes the piece with various sets of constraints. Scenario 2-1 (Fig. 8) is a harmonization task, asking to complete the piece given the top voice. Scenario 2-2 (Fig. 9) is a voicing task where the bass and top voices are given. Scenario 2-3 (Fig. 10) is when bass and chords (without voicing) are given.

Analysis 2-1 One of the most obvious problems in this piece is the minor 9th chord at the end of the fourth measure. As a response we could try to increase the weight of the outer voices dissonance evaluation. An interesting observation is that the model used a lot of first inversion chords in this piece. This could be a combined result of the design of the "Interval Dissonance," where interval related to bass note are given more weight and "Interval Variety," where the presence of sixths and thirds are given more weights than the presence of fifth in a chord's interval composition. Those two evaluations accidentally make first inversion chords more favorable, since first inversion chords most likely result in bass related intervals to be sixths and thirds. To address this issue, we should have some adjustments to these two sub-evaluations. Another point is that the first chord has very unbalanced spacing. Future fitness function will add a sub-evaluation specifically for controlling spacing.

Analysis 2-2 One issue is the unresolved $\hat{4}$ of the V7 chord at the cadence. Resolution of harmonic tension is currently not implemented in the fitness function since they don't very often appear in the first species. In future work, especially as the music complexity increases, we will add a collection of sub-evaluations to control harmonic tension.

Analysis 2-3 The cadence in the middle is not correctly resolved (unresolved leading tone). This is because the fitness function assumes that there is only one cadence at the end. To address this problem, we need the Music Template to include meta-data such as the location of cadence and let the fitness function to be able to respond to that. Another problem is that the soprano voice is jumpy in the second half of the piece. This could be fixed by adjusting the melody evaluation such that more weight is assigned to the "Prefer step motion" sub-evaluation.

8.3 Scenario 3: Hard Harmonization

The algorithm is given a hard melody to harmonize (Fig. 11), with non-chord tones that require secondary chords to harmonize. The piece is now twice as long as the previous one in Experiment 2, which squares the size of the search space. This melody is suggested in (de Vega, 2017) as a hard testing melody for harmonization algorithms.

Analysis The piece doesn't start on the I chord. As the number of notes increase, the melody sub-evaluation that controls the start of the melody becomes weaker and results in a non-tonic start. New evaluations should be added to control the harmonic anchor points (specific locations for specific chords). This could also allow us to specify the presence of cadence in the middle, where the piece used a iii chord. The model has a hard time dealing with non-diatonic tones. For the Bb in the soprano, it comes up with a G and for the F# it came up with note A and F. It could be a program implementation problem resulting in the fitness to only check diatonic intervals instead of absolute interval. Further tests are needed to verify the cause of the issue.

9 Conclusion

This algorithmic composition framework is able to optimize the fitness function to a satisfactory level, except for the hard harmonization problem. One noticeable shortcoming about the musical output is that the chord progressions seem to lack a sense of direction. This is because the current fitness function design has not considered this factor. To further improve the generated result, we need to add more sub-evaluations to the fitness function and perform a systematic optimization of the model's parameters.

The general aim for future work is to do more complex musical tasks such as fugues and invertible counterpoints. Two lines of research will continue simultaneously and support each other. First we need to expand the collection of sub-evaluations for future fitness function crafting. This collection will be implemented in a modular way and will be open sourced. Second, we need to improve the current genetic algorithm. This means more systematic studies of the effects of the hierarchical operators and attention mechanism. In particular, we want to identify the small music situations that cause the bottle neck in the optimization process and then try different different hierarchical operators on them to test their ability to overcome the local maximums of the fitness landscape.

The effects of the *Attention* mechanism also need further investigation. My current experiments speculate that it improves the beginning stage of the optimization fitness but causes earlier plateau. But it is too soon to conclude at this stage whether it leads to premature convergence.

More complex musical tasks may require the algorithm to perform a series of mutations as opposed to one to overcome the local extremes. This usually requires some level of previously acquired experience. Using reinforcement learning to direct "Attention" to achieve high fitness could be a viable way to enable the model to acquire the experience of optimizing similar fitness functions.

Acknowledgement

I want to thank Panayotis Mavromatis and Brian McFee for their valuable suggestions on this research project. I also want to thank the three anonymous reviewers for their time spent to provide feedback on this paper.

References

Beyer, H.-G., & Schwefel, H.-P. (2002, 03). Evolution strategies - a comprehensive introduction. *Natural Computing*, *1*, 3-52. doi: 10.1023/A:1015059928466

de Vega, F. F. (2017). Revisiting the 4-part harmonization problem with gas: A critical review and proposals for improving. In *2017 ieee congress on evolutionary computation (cec)* (p. 1271-1278).

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., & Sutskever, I. (2020). *Jukebox: A generative model for music.*

Donnelly, P. J., & Sheppard, J. W. (2011). Evolving four-part harmony using genetic algorithms. In *Evoapplications.*

Engel, J., Resnick, C., Roberts, A., Dieleman, S., Eck, D., Simonyan, K., & Norouzi, M. (2017). *Neural audio synthesis of musical notes with wavenet autoencoders.*

Farbood, M., & Schöner, B. (2001). Analysis and synthesis of palestrina-style counterpoint using markov chains. In *Icmc.*

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., & Gagné, C. (2012, jul). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, *13*, 2171–2175.

Hadjeres, G., Pachet, F., & Nielsen, F. (2016). Deepbach: a steerable model for bach chorales generation. In *Icml.*

Herremans, D., & Sörensen, K. (2013, 11/2013). Composing fifth species counterpoint music with a variable neighborhood search algorithm. *Expert Systems with Applications*, *40*. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0957417413003692` doi: 10.1016/j.eswa.2013.05.071

Huang, C.-Z. A., Hawthorne, C., Roberts, A., Dinculescu, M., Wexler, J., Hong, L. L., & Howcroft, J. (2019). Approachable music composition with machine learning at scale. In *Ismir.*

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Simon, I., Hawthorne, C., Shazeer, N., ... Eck, D. (2019). Music transformer: Generating music with long-term structure. In *Iclr.*

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. doi: 10.1109/MCSE.2007.55

Liang, F. (2016). Bachbot: Automatic composition in the style of bach chorales. *University of Cambridge*, *8*, 19–48.

McIntyre, R. A. (1994). Bach in a box: the evolution of four part baroque harmony using the genetic algorithm. In *Proceedings of the first ieee conference on evolutionary computation. ieee world congress on computational intelligence* (p. 852-857 vol.2).

Pachet, F., & Roy, P. (2011). Markov constraints: steerable generation of markov sequences. *Constraints*, *16*(2), 148–172.

Pearce, M., Meredith, D., & Wiggins, G. (2002). Motivations and methodologies for automation of the compositional process. *Musicae Scientiae*, *6*(2), 119–147.

Phon-Amnuaisuk, S., Tuson, A., & Wiggins, G. (2000, 06). Evolving musical harmonisation.
doi: 10.1007/978-3-7091-6384-9_39

Salzer, F., & Schachter, C. (1989). *Counterpoint in composition: The study of voice leading.* Columbia University Press. Retrieved from `https://books.google.com/books?id=7IP0Jo9Dr9QC`

Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, *13*(2), 22.

Yang, L.-C., Chou, S.-Y., & Yang, Y.-H. (2017). Midinet: A convolutional generative adversarial network for symbolic-domain music generation. *ArXiv, abs/1703.10847*.

A Music Encoding

```
[
[[['16', '17'], ['.', '.'], ['5', '10']], [['1', '11'], ['11', '12'], ['3', '2']], [['4', '6'], ['15', '.'], ['.', '16']]]
[[['-2', '-3'], ['11', '-4'], ['0', '.']], [['6', '6'], ['11', '0'], ['.', '5']], [['*', '6'], ['.', '7'], ['.', '*']]]
[[['-9', '.'], ['-3', '-9'], ['*', '*']], [['.', '*'], ['4', '3'], ['3', '2']], [['-7', '.'], ['-5', '*'], ['-1', '-2']]]
[[['-15', '.'], ['*', '*'], ['-13', '*']], [['-17', '*'], ['-1', '*'], ['-5', '-11']], [['.', '.'], ['-20', '*'], ['*', '-3']]]
]
```

Fig. 1: A randomized array of shape (4,3,2,2).

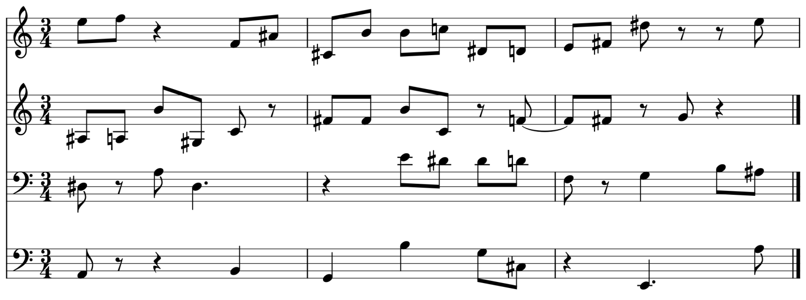

Fig. 2: The music corresponding to the randomized array

B Contruction of $\boldsymbol{\sigma_w}$

Let

$$h(x) = \begin{cases} 0, & x \leq 0 \\ e^{-\frac{1}{x}}, & x > 0 \end{cases}, \tag{8}$$

and

$$g_w(x) = \frac{h(x)}{h(x) + wh(1-x)}, \tag{9}$$

$$\sigma_w(x) := g_w(x+1), \tag{10}$$

then

$$\sigma_w \xrightarrow[w \to \infty]{pt.wise} \mathbb{1}_{[0,\infty]}. \tag{11}$$

C Figures for Illustration

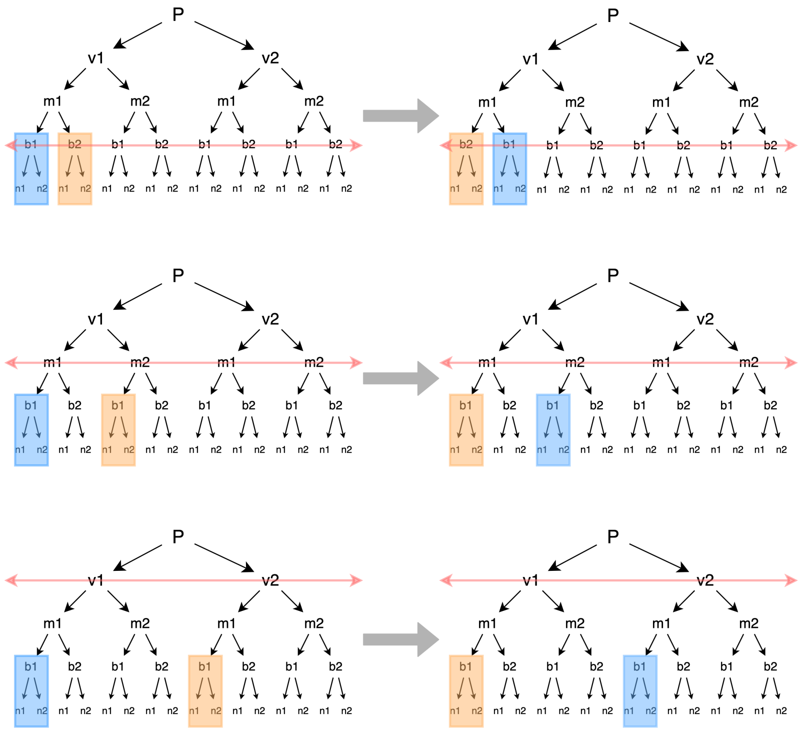

Fig. 3: Visualizing Hierarchical Neighbor Swap as subtree exchange. Here we choose the piece shape to be (2,2,2,2) for demonstration. The changing unit can be any subtree, here only two-level subtrees are demonstrated

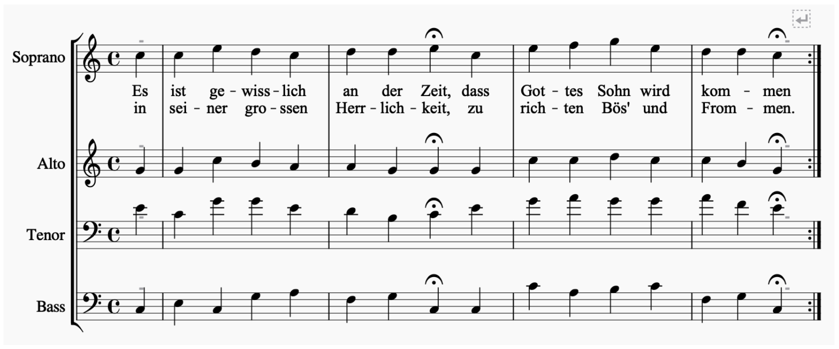
Soprano
Alto
Tenor
Bass
Es ist ge - wiss - lich an der Zeit, dass Got - tes Sohn wird kom - men
in sei - ner gro - ssen Herr - lich - keit, zu rich - ten Bös' und From - men.

Fig. 4: The reduction of BWV 307 to first species and transposed to C major

Fig. 5: Table of parameters

(a) Evolution Strategy Parameters

Piece Population	
ES type	$(\mu/\rho + \lambda)$
μ_p population size	150
ρ_p mating size	135
λ_p off-spring size	150
Mutation rate	0.3
Crossover rate	0.3
reproduction rate	0.4
Hierarchy probabilities	
Attention Population	
ES type	$(\mu/\rho + \lambda)$
μ_a population size	$\propto$ piece shape
ρ_a mating size	0.1 μ_a
λ_a off-spring size	$\propto$ piece shape
Mutation rate	0.3
Crossover rate	0.1
Reproduction rate	0.6

(b) Scenario Configurations

Scenario 1: Free Composition	
Piece Shape	(4,8,2,2)
Generation	150
Runtime	5 min
Scenario 2: Music Completion	
Piece Source	First Phrase of BWV 307
Piece Shape	(4,5,4,2)
Generation	300
Runtime	15 min
Scenario 3: Hard Harmonization	
Melody Source	Exercise from a Spanish Music Conservatory
Piece Shape	(4,8,4,2)
Generation	1000
Runtime	1h

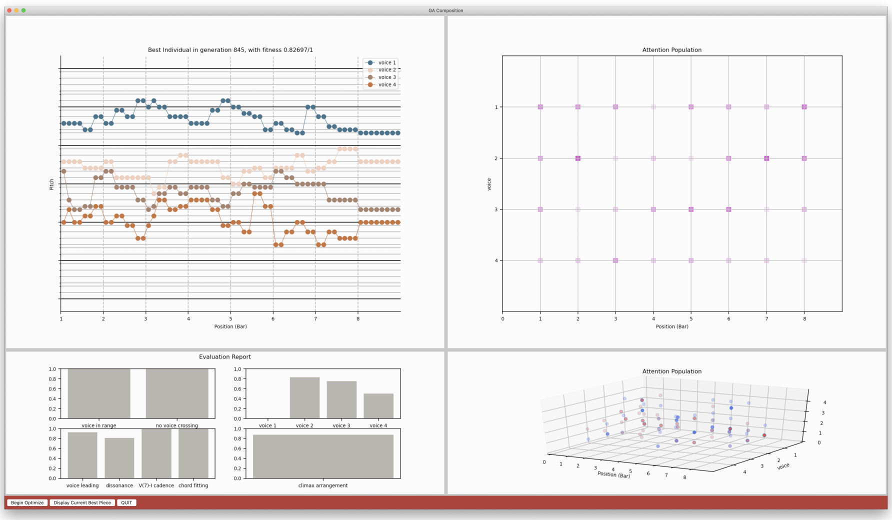

Fig. 6: The real-time optimizing interface. User can direct the attention by clicking on the attention plot on the top right. The current best individual of the population is displayed on the top left. The evaluation report of the best individual is displayed at the bottom left. The bottom right shows the 4-d distribution of the attention population.

D Generated Results

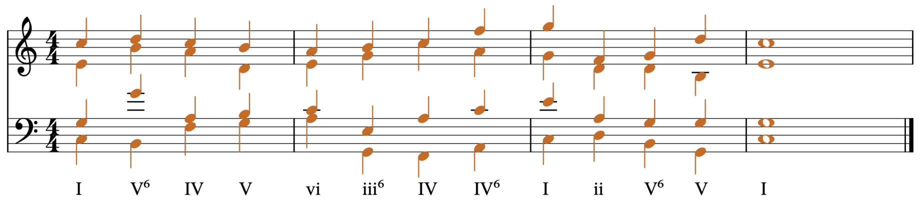

Fig. 7: Scenario 1 Four-Part Free

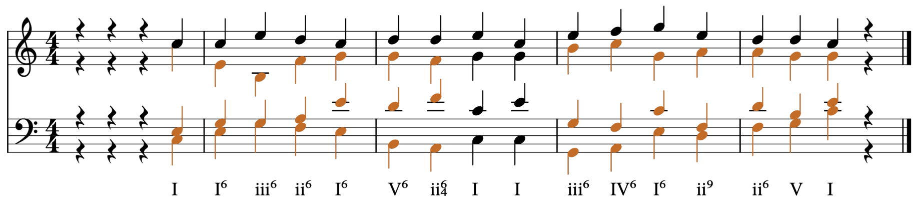

Fig. 8: Scenario 2-1: Given Soprano

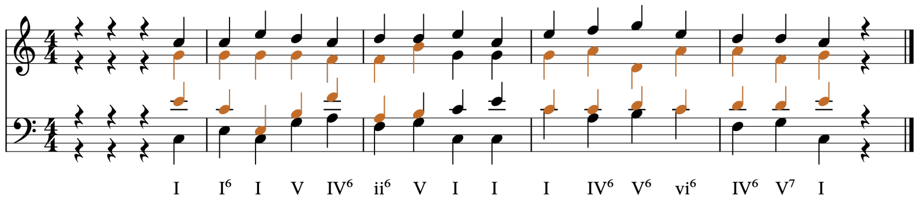

Fig. 9: Scenario 2-2: Given Bass and Soprano

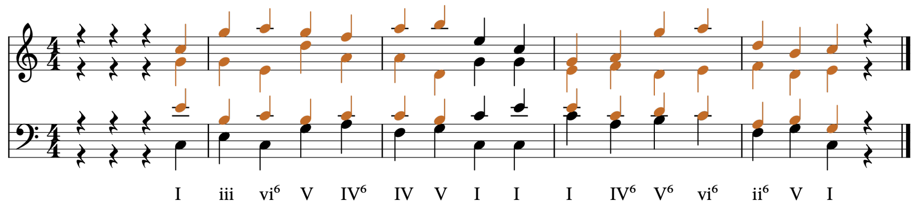

Fig. 10: Scenario 2-3: Given bass

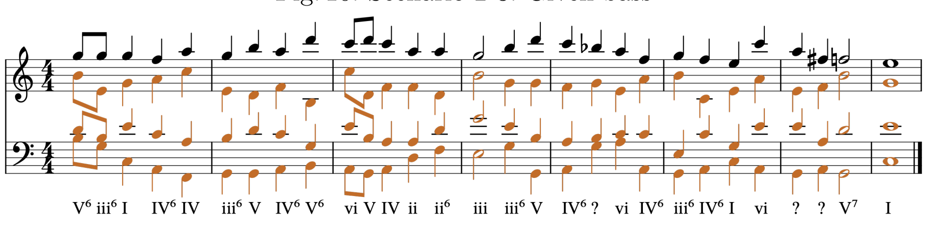
V6 iii6 I IV6 IV iii6 V IV6 V6 vi V IV ii ii6 iii iii6 V IV6 ? vi IV6 iii6 IV6 I vi ? ? V7 I

Fig. 11: Scenario 3: hard harmonization

E Evaluation Reports

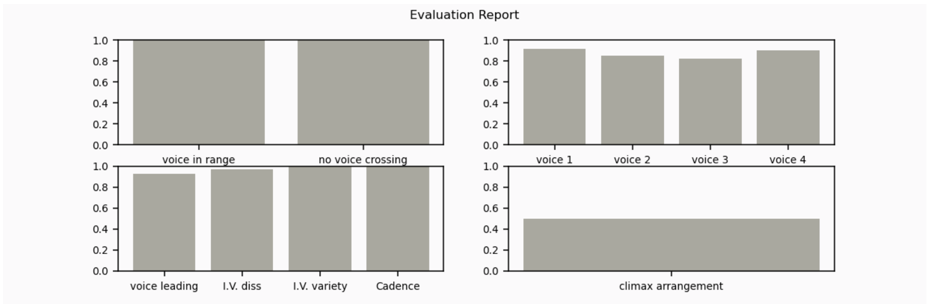

Fig. 12: Scenario 1 Report

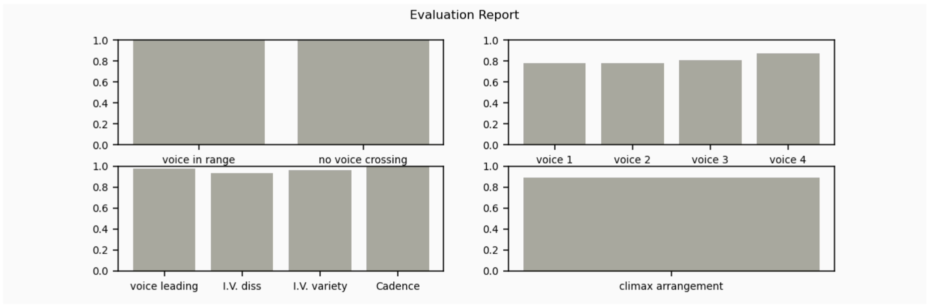

Fig. 13: Scenario 2-1 Report

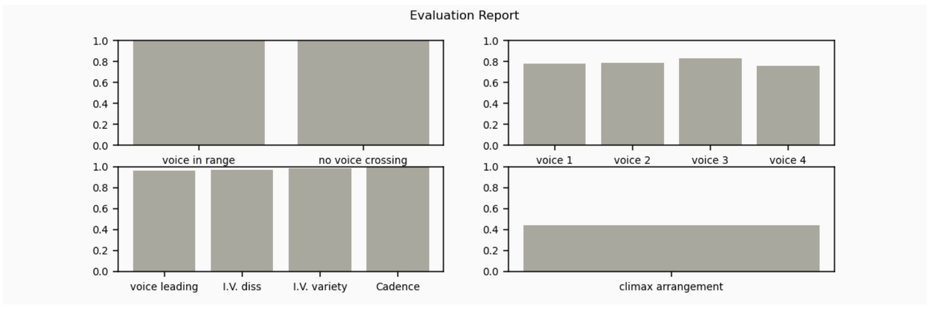

Fig. 14: Scenario 2-2 Report

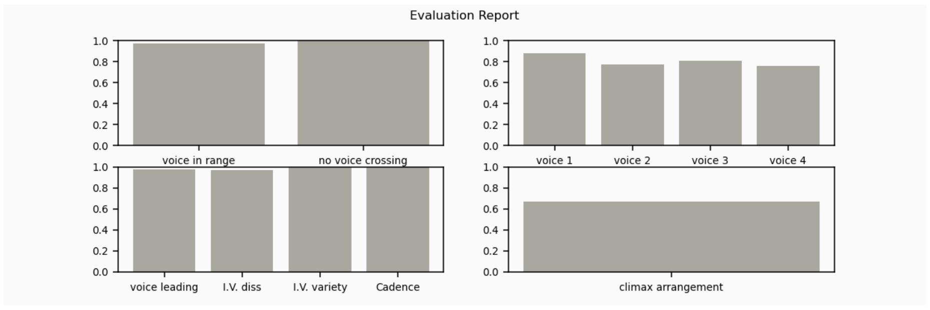

Fig. 15: Scenario 2-3 Report

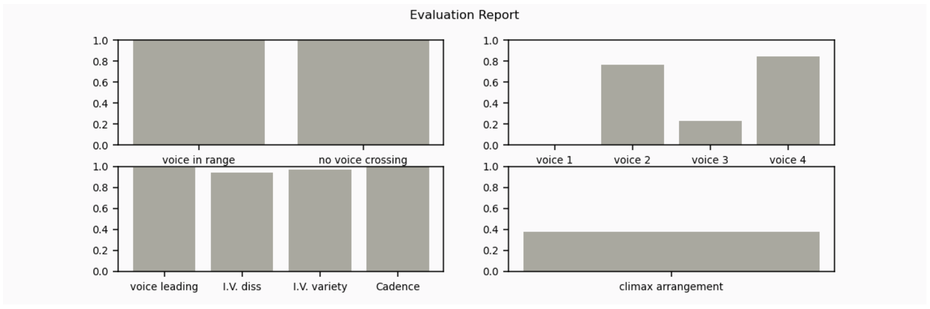

Fig. 16: Scenario 3 Report

F Fitness components implementation

Table 1: Melody Fitness f

Sub-evaluations	Description	Fitness
Tier 1: Hard Rules		
Interval Dissonance	Avoid tritone and sevenths.	$f_{1,1}$
Leap		$f_{1,2}$
Number limit	*occurrence* $\in [2, L/10 + 2]$.	$f_{1,2,1}$
Big leap limit	*occurrence* $\in [0, 2]$.	$f_{1,2,2}$
Separation of leaps	Prefer leaps in outer voices spread apart.	$f_{1,2,3}$
Sequence	Avoid long sequence (5) of same motion.	$f_{1,3}$
Range		$f_{1,4}$
Pitch span size	$P5 \leq size < P12$.	$f_{1,4,1}$
Melody range	Notes belong to voice range.	$f_{1,4,2}$
Start and end		$f_{1,5}$
Target notes	Start and end on I (outer voices on $\hat{1}$).	$f_{1,5,1}$
Cadence motion	(Soprano only) ends by step.	$f_{1,5,2}$
Tier 2: Local Shape		
Leading Tone	Penultimate leading tone resolve up to $\hat{1}$.	$f_{2,1}$
Motion Damping	Prefer decaying size of consecutive leaps.	$f_{2,2}$
Avoid Halting	Outer voices avoid repeating same note.	$f_{2,3}$
Tier 3: Global Shape		
Unique Extremes	The highest/lowest notes should be unique.	$f_{3,1}$
Avoid centering	Examined using window of fixed length (5).	$f_{3,2}$
Prefer step motion	Up to 3/4 of all motions.	$f_{3,3}$
Direction change	*occerance* $\leq L/4$	$f_{3,4}$

Table 2: Four-part Counterpoint Fitness F

Sub-evaluations	Description	Fitness
Tier 1: Hard Rules		
Range	Each voice in their respected range.	$F_{1,1}$
Voice crossing	Avoid voice crossing but small amount of unison is fine	$F_{1,2}$
Tier 2: Basic skeleton		
Outer voices	Evaluation of individual outer voices.	$F_{2,1} = f$
Voice Leading	Avoid parallel/hidden fifths and octaves.	$F_{2,2}$
Interval Dissonance	discourage dissonant in chords' interval vector.	$F_{2,3}$
Interval Variety	Encourage a balanced distribution among perfect/imperfect consonances in chords' interval vectors.	$F_{2,4}$
Cadential Movement	End with a correctly constructed perfect authentic cadence.	$F_{2,5}$
Tier 3: Inner Voices		
Inner Voices evaluation	Evaluation of individual inner voices.	$F_{3,1} = f$
Tier 4: Global Shape		
Climax Arrangement	The climaxes of the voices should be spaced out and not overlap	$F_{4,1}$

Notations and definitions

$$n_{object}\text{: number of an object in the melody} \tag{12}$$

$$bump_{[a,b]}(x) = \begin{cases} 1 - \frac{a-x}{a} & \text{if } x < a \\ 1 & \text{if } x \in [a,b] \\ \frac{1}{1+0.1(x-b)} & \text{if } x > b \end{cases} \tag{13}$$

$$dist(x, A) = min_{a \in A}|x - a| \tag{14}$$

$$Dist(A, B) = min_{a \in A, b \in B}|a - b| \tag{15}$$

$$S(x) = \frac{1}{1 + e^{-x}} \tag{16}$$

Sub-evaluations

$$f_{1,1}(melody) = \text{ratio of motions with eligible intervals} \tag{17}$$

$$f_{1,2,1}(melody) = bump_{[2,4]}(\#leaps) \tag{18}$$

$$f_{1,2,2}(melody) = bump_{[0,2]}(\#bigleaps) \tag{19}$$

$$f_{1,2,3}(melody) = \min(1, \frac{1}{4}\text{smallest gap size between leaps}) \tag{20}$$

$$f_{1,3}(melody) = \text{whether exists long sequence } (> 4) \text{ of repeating motion} \tag{21}$$

$$f_{1,4,1}(melody) = bump_{[5,19]}(highestnote - lowestnote) \tag{22}$$

$$f_{1,4,2}(melody) = 1 - tanh(\underset{notes}{Avg} dist(note, range)) \tag{23}$$

$$f_{1,5,1}(melody) = (1 - \underset{x \in melody[0,-1]}{Avg} tanh \circ dist(x, \text{I chord})) \tag{24}$$

$$f_{1,5,2}(melody) = bump_{[1,2]}(\text{ending interval}) \tag{25}$$

$$f_{2,1}(melody) = \text{whether the penultimate leading tone (if exists) resolves up by step} \tag{26}$$

$$f_{2,2}(melody) = \underset{leaps}{Avg} \quad bump_{[0,0.4]}(|\frac{size_{\text{interval after leap}}}{size_{leap}}|) \tag{27}$$

$$f_{2,3}(melody) = \frac{\#\text{halting}}{L} \tag{28}$$

$$f_{3,1}(melody) = \frac{2}{\#highestnote + \#lowestnote} \tag{29}$$

$$f_{3,2}(melody) = \min\{g(segment) : segment \in \text{length 4 segments in melody}\} \tag{30}$$

$$\text{where } g(segment) = 1 - \frac{n_{mostoccurednote}}{\#notes} \tag{31}$$

$$f_{3,3}(melody) = min(1, (\frac{3}{4})\frac{\#stepmotions}{\#motions}) \tag{32}$$

$$f_{3,4}(melody) = bump_{[0,4]}(\# \text{ direction change}) \tag{33}$$

$$F_{1,1}(Piece) = 1 - \underset{voices}{Avg}\, tanh(dist(voice, range)) \tag{34}$$

$$F_{1,2}(Piece) = \left(\underset{(v_+,v_-)\in Pairs}{Avg} \min(2S(\min(v_+ - v_-), 1) \right) - Penalty_{overlap} \tag{35}$$

$$\text{where } Penalty_{overlap} = \begin{cases} 0, & overlaps < 0.2allnotes \\ S(0.1overlaps), & else \end{cases} \tag{36}$$

$$\text{where } Pairs = \{(voice_i, voice_j) : i < j\} \tag{37}$$

$$\tag{38}$$

$$\tag{39}$$

$$F_{2,2}(Piece) = \underset{\text{chord transitions}}{Avg} \text{ whether transition free of parallel/hidden P8 or P5} \tag{40}$$

$$F_{2,3}(Piece) = \underset{chords}{Avg}\, C(chord) \tag{41}$$

$$\text{where } C(chord) = \frac{(10,5,1)^T}{\|(10,5,1)\|}(\mathrm{CR}_{\text{outer}}, \mathrm{CR}_{\text{bass related}}, \mathrm{CR}_{\text{inner}}) \tag{42}$$

$$\text{where CR refers to the consonance ratio of a set of intervals} \tag{43}$$

$$F_{2,4}(Piece) = \underset{chords}{Avg} \frac{(1,1,1,0.2)^T}{\|(1,1,1,0.2)\|} np.isin([0,3,4,5], pc_{\mathbb{Z}_2}(chord)) \tag{44}$$

$$\text{where } pc_{\mathbb{Z}_2}(chord) = \text{chord's directed intervals taking smallest inversion} \tag{45}$$

$$F_{2,5}(Piece) = \frac{1}{2} \underset{\text{cadential chords}}{Avg} dist(chord, targetchord) + eva_{bass\hat{5}\hat{1}} \tag{46}$$

$$\text{where } eva_{bass\hat{5}\hat{1}} = \frac{1}{24}\|(bassmotion - (7,0))\|_{L^1} \tag{47}$$

$$F_{4,1}(Piece) = \min(\frac{n_{voices} l^*}{len(voice)}, 1) \tag{48}$$

$$\text{where } l^* = \min_{i<j} Dist(argmax(voice_i), argmax(voice_j)) \tag{49}$$