RL-MUSICIAN: A CONFIGURABLE OPEN-SOURCE TOOL FOR SPECIES COUNTERPOINT COMPOSITION

A DRAFT

Nikolay Lysenko
nikolay.lysenko.1992@gmail.com

April 14, 2020

ABSTRACT

In this paper, automatic composition of first species counterpoint is framed as a tree search problem close to reinforcement learning. An environment representing a polyphonic piece in progress is introduced. An action is filling of the current measure (bar) for all melodic lines simultaneously. Once a specified number of measures is added, an episode terminates and the resulting piece is evaluated based on some hand-written rules derived from music theory. Best sequences of actions are found with a variation of Monte Carlo Beam Search.

Although current setup of the problem may seem simplistic, it is a step in a promising direction. Numerous recent advances in music generation are achieved with supervisedly trained neural networks and reinforcement learning is not used there as a standalone paradigm. Since collections of existing pieces are involved as datasets, results are "inspired" by these reference pieces. However, the notion of creativity is wider – it also includes generation of something that meets criteria of being an art, but is not backed by existing pieces. Thus, new aspects of creativity can be revealed by further research of pure reinforcement learning approaches.

Keywords: algorithmic composition · music generation · reinforcement learning · species counterpoint

1 Introduction

Algorithmic music composition is automatic generation of outputs representing musical pieces and written in some formal notation. To name a few of common notations, there is sheet music, tablature, and MIDI standard. It is not required from output representation to unambiguously define sound waveform. For example, sheet music leaves exact loudness of played notes to discretion of a performer and may include only imprecise hints like pianissimo (very quiet). Anyway, there are parameters of sounds that must be determined by their representation. Usually, such parameters are pitch, start time, and duration.

Currently, there are no perfect tools for algorithmic composition. Plenty of various approaches to composing music automatically exist, but none of them produce well-structured and novel pieces that cannot be distinguished from works of a talented human composer. Thus, there is an open research problem. Several recent breakthroughs in it are accomplished with machine learning and both supervised learning [1, 2] and reinforcement learning [3, 4] are applicable. Moreover, some researchers combine them [5, 6, 7]. Meanwhile, fitness function optimization also yields prominent results[8, 9].

In this paper, a new approach to music composition is suggested. Although it is quite straightforward, to the best of the author's knowledge it is not described anywhere, so the first contribution is its rigorous definition. The second contribution[1] is proper setup and tuning of the parameters. It is often the case that accurately tuned simple methods outperform complex methods [10] and, in addition, they are more transparent and less demanding.

The approach relates to a well-known framework for simplified music composition called species counterpoint. It is usually used by beginning composers in order to practice in voice leading (also known as part writing) and polyphony. From automatic composition point of view, species counterpoint is interesting due to two its properties:

[1] As of now, it is in progress.

- There are rules prescribing what to do and what not to do. So there are less degrees of freedom and the problem is computationally more feasible.
- There are clear hints of what is better to avoid and what should be added. So formal rules for automatic evaluation of resulting piece can be introduced.

The word "species" in the name of the framework indicates that there are some types – each of them has its own rhythmic pattern. Here, first species counterpoint is covered. This type has trivial rhythmic structure such that only whole (semibreve) notes can be used and every note must be placed in a beginning of a measure. Thus, interaction between rhythm and tonality is eliminated and the problem becomes even simpler.

Brief outline of the setup is as follows. There is an environment that keeps a piece in progress and the piece consists of a fixed number of measures. Each measure must be either empty (if it is not filled yet) or containing exactly one note per a melodic line. Initially, only the first measure and the last measure are filled with user-defined values and the current measure is the second one. An action is filling of the current measure, i.e., adding a pitch to each of the melodic lines. After an action, the next measure becomes the current one. When the last measure is reached, episode is ended and a reward based on evaluation of the created composition is assigned to the sequence of passed actions.

The remainder of the paper is organized as follows. Section 2 describes the most relevant works and the place of the current paper amongst them. Namely, Section 2.1 discusses algorithmic composition, whereas Section 2.2 contains preliminaries on Monte Carlo Beam Search and lists modifications made in this study. Details of the setup (in particular, rules used for evaluation of pieces) are defined in Section 3. Section 4 reports experimental results and the way to reproduce them. Finally, Section 5 concludes.

2 Background and Related Work

2.1 Algorithmic Composition

Usage of computers for automatic composition of music dates back to 1950s and, what is more, approaches that can be labeled as algorithmic are known since Guido of Arezzo (11th century). Therefore, it is impossible to list here all studies about the subject. An interested reader can find more information in specialized reviews such as [11]. What is discussed below is just some examples related to the current study.

These examples can be divided into two groups. The first one is formed by approaches where an explicitly given fitness function is optimized. The second group is formed by composition of music with reinforcement learning in its narrow sense (i.e., methods based on value function estimation). This group is something in between the first group and supervised learning approaches. On the one hand, reward function is a domain-specific name for fitness function. On the other hand, data generated during past episodes are used for training supervised models.

The closest to the current study works [8, 9] belong to the first group. A local search algorithm named Variable Neighborhood Search is used there in order to create first and fifth species counterpoint pieces that maximize fitness function. What these studies and the current one share in common is that fitness (reward) function is backed by the theory of species counterpoint. However, the current study is a bit closer to reinforcement learning, because composition of a piece is considered an episode consisting of sequential steps after each of which a list of valid actions changes. Conversely, in [8, 9], the whole piece is created at random and then improved by applying small alterations. As a result, pieces that break some fundamental rules of counterpoint are also evaluated there (they are marked as unfeasible), whereas here such pieces cannot be created. This way of that, the core drawback of the two discussed papers is that download links to the software implementation named Optimuse are broken and so the results cannot be reproduced.

There are other optimization-based approaches as well. Instead of local search metaheuristics, population-based metaheuristics (genetic algorithms, evolutionary strategies, cross-entropy method), or construction metaheuristics (ant colony algorithms) can be used. However, such papers are less relevant to the current study and so let us switch our attention to reinforcement learning. It can be applied to algorithmic composition either as a standalone paradigm or in connection with supervised deep learning. As for the first case, probably, there are no influential works and only separate studies like [3, 4] are available. Conversely, the second group recently gained attention and several interesting approaches were developed there.

For example, reinforcement learning can be used for altering weights of recurrent neural networks (RNNs) trained to generate music sequentially [5, 6]. The goal is to make generated pieces more structured and conformed with music theory rules. To define an environment, let its state be composed of recurrent neural networks states and previously played notes, let an action be an output for current time step and let reward depend on both evaluational rules and probability of output according to initial RNN. Rewards are granted immediately after a step and DQN (Deep Q-Networks [12]) are preferred as a training algorithm.

Usually, softmax activation function is used in the last layer of generative RNN. However, it is a common practice in reinforcement learning domain to generalize it with Boltzmann softmax which is a family of activation functions parametrized by one parameter called temperature and denoted as t. Depending on temperature, output distribution can vary from atomic distribution concentrated at the most probable action ($t = 0$) to distribution returned with softmax activation ($t = 1$) and to uniform distribution ($t \to +\infty$). In [7], Boltzmann softmax is used and also input vector is extended by introducing an additional part (so called plan) indicating origination of initial input. An environment is defined so that state is a pair of temperature and plan, actions are changes in either temperature or plan, and reward depends on evaluation of produced with these settings piece.

Since algorithmic composition can be framed as training of a generative model, it sounds natural to try one of the most salient examples of generative models – generative adversarial networks (GANs) [13]. However, widespread notations for music assume sequences of discrete values but classical GANs work well only with continuous data, because gradient of discrete-valued functions is uninformative. Techniques originating from reinforcement learning can be used to overcome this obstacle [14, 15]. Namely, generator is trained with policy gradient method [16]. Such methodology is applied to various tasks and music composition is amongst them [17].

Let us highlight again that above cases of reinforcement learning usage still require supervised learning and datasets. The arguments for not involving them at all are as follows:

- Finding new ways of music creation is a more challenging task than imitation of famous pieces. If no known pieces are used, chances are that the harder problem is considered and it is not replaced with the simpler problem of imitation.
- There are tuning systems other than equal temperament (for instance, in microtonal music). For some of them it may be impossible to collect a dataset large enough to allow training models in a supervised fashion. However, developers of a tuning system should know some underlying principles and so (at least, in theory) it is possible to define a fitness function and find pieces that optimize it or to define a reward and train an agent based on it.

This implies that optimization-based approaches and pure reinforcement learning approaches must not be overlooked due to advances in supervised learning application.

2.2 Monte Carlo Beam Search

Consider a tree search problem. To be more detailed, suppose that there is a tree such that every its leaf is mapped to a reward and the goal is to find the leaf with the maximum reward. However, leaves and corresponding to them rewards are not known initially. Information about a leaf becomes available only after a path from the root of the tree to this leaf is constructed.

The described in Section 1 procedure of first species counterpoint writing can be considered a tree search problem. The root is the initial state of an environment. Each valid option to fill the current measure is an edge from the current node of the tree to a new node. Leaves are pieces where all measures are filled.

If size of a tree is small enough, exhaustive search is possible. There are well-known classical algorithms such as depth-first search (DFS) that allow iterating over all paths from the root to leaves. Unfortunately, trees are large in numerous real problems. For instance, a rough estimate of total number of leaves for first species counterpoint with 2 voices and 16 measures is about 7^{14} which is equal to several hundreds of billions.

An obvious idea for a large tree is to use random search. To create a path, every next node is chosen at random until a leaf is reached. After specified number of paths are created, the best one amongst them is chosen as an approximate solution.

This idea can be further improved in the following way. Let an optimum path be constructed sequentially with adding one new node per a series of random trials. If so, there is a stub initially containing only the root. At every iteration, there are several trials to continue stub at random until a leaf is reached. The best one amongst them (and amongst trials from previous iterations if applicable) is chosen and the first non-stub node from there is added to the stub. This method can be called Monte Carlo Search. The gain in comparison with plain random search is due to allocating more exploration to subtrees that contain the current best leaf.

Above version of Monte Carlo Search is greedy, because it ignores that some subtrees might be underexplored. A well-known method that explores more than greedy search but uses less computational resources than exhaustive search, is called beam search. Instead of just one stub, w distinct stubs are constructed and used, where w is a hyperparameter named beam width. It is natural to call this method Monte Carlo Beam Search.

Probably, the term Monte Carlo Beam Search was coined in [18]. However, it has different meaning there. In [19], the same author defined Nested Monte Carlo Search as a combination of Monte Carlo Search with DFS. Given a current stub, all its extensions by n new nodes (where n is a hyperparameter named level of nestedness) are found with DFS and then continued at random. So pre-defined number of trials per iteration is absent, because number of trials is equal to number of extensions by n nodes ahead. Comparing with Monte Carlo Search, Nested Monte Carlo Search distributes exploration more evenly between short extensions of a stub. This way or that, in [18], Monte Carlo Beam Search is defined as beam search based on Nested Monte Carlo Search.

Nevertheless, here, beam search is based on Monte Carlo Search. The reason for it is quite mundane – parallel implementation of DFS looks like overcomplication that may yield too few. The second difference with [18] is that dependence of trials number on stub length is introduced. The longer the stub is, the smaller a subtree to be explored is. Therefore, it is better to reallocate computational resources in order to explore more when a stub is short and to explore less when a stub is long.

3 Methodology

3.1 Setup

To start with, define representation of a musical piece. Suppose that counterpoint is written in l melodic lines (parts, voices), where l is usually 2 or 3. Suppose also that a diatonic scale (like C-major or A-minor) is chosen. Let each line be a list of pitches from the scale (i.e., only 7 out of 12 notes per octave can be used and so chromaticism is prohibited). Set length of all lines to the same number m, where m is usually between 8 and 24. From music point of view, this means that a piece to be created is rather a phrase that can be inserted into a longer composition. Such limitation goes back to species counterpoint, because it is designed exactly for writing short phrases. This way or that, define a piece as a matrix P of size $l \times m$ where element P_{ij} is a pitch played in i-th line in j-th measure.

Every line must have only small intervals between its successive elements in order to be perceived as a single line and not as a heterogeneous collection of sounds. Suppose that maximum allowed melodic interval is s scale degrees (here, $s = 3$). If so, there are no more than $(2s + 1)^l$ options to fill the next measure given current measure. Actually, the number is smaller, because some options are filtered out by counterpoint rules. The rules enforced in this study can be either rules of voice leading (part writing) or rules of harmony.

Here, five rules of voice leading are imposed:

- Only pitches from tonic triad (i.e., tonic, mediant, and dominant) can be rearticulated (repeated), because only they are stable;
- If a line has submediant followed by leading tone, tonic must be used after leading tone, because there is strong attraction to it; similarly, if a line has leading tone followed by submediant, dominant must be used after submediant;
- If a melodic interval between two successive pitches from the same line is larger than second (i.e., if there is a skip), the second pitch must be from tonic triad, because skip creates enough tension and something stable is needed;
- Movement by step in the opposite direction must happen immediately after a skip larger than third in order to resolve tension associated with the skip;
- For every pitch, there must be a way to reach the final pitch of this line with step motion (the final measure is filled initially with user-defined sonority), because resolution must be present during last measures.

In addition, there are three rules of harmony:

- All harmonic intervals between two simultaneously sounding pitches from different lines must be consonant;
- Within a measure, all pitch classes must be different unless the measure is the first one or the last one;
- For every pitch, the closest to it pitch from the same measure must be no more than tenth apart.

Above rules are absolute and it is not possible to break any of them, because pitches for the next measure can be selected only from options that are filtered according to these rules. If a set of such options is empty, episode terminates and an unfinished sequence of actions receives negative reward for dead end.

3.2 Evaluational Rules

At the current revision of the study, seven properties of a final piece are evaluated separately and total reward is a weighted sum of these seven scores.

- **Absence of looped pitches.** A stable pitch may be rearticulated, but not more than once in a row. Every excessive occurrence of the same pitch within a line adds constant negative reward.
- **Absence of looped fragments.** Define a fragment as one or more consecutive measures (at a piece level, not at a melodic line level). For every fragment followed by exactly the same fragment, a constant negative reward is added.
- **Motion.** For every pair of lines every motion between two successive measures adds a reward that depends on a type of motion. Since contrary motion makes lines more distinguishable, reward for it is positive. Reward for oblique motion is zero, reward for similar motion is slightly negative and reward for parallel motion is negative, because parallel motion makes lines sounding like copies of each other.
- **Correlation between lines.** Consider each line as a sequence of real numbers equal to pitch positions within a range of available pitches. Then average over all pairs of lines correlation between two lines can quantify independence of lines. Actually, reward is rescaled to be from 0 to 1 – it is half of difference of 1 and the above average correlation.
- **Explicity of climax.** Music theory suggests that good melodies must have goal-oriented motion and that such goal-orientedness may be achieved by having exactly one climax point. For each line, constant reward is decreased for each duplication of climax point and for not so high climax point.
- **Number of skips.** Positive reward is associated with every line such that number of skips in it lies within a specified range. The motivation behind this is that lines with almost no skips are not interesting, but lines with too many skips are not coherent.
- **Absence of downward skips**. Counterpoint theory prescribes to have downward step motion. To encourage its usage, there are negative rewards for every downward skip and the larger a skip is, the higher a penalty its.

It can be seen that the first two properties deal with non-triviality of results, the next two properties deal with interaction of lines and the last three properties deal with melodic quality of each separate line.

4 Experimental Results

A software implementation of the above methodology in Python programming language is available on GitHub[2]. The code has built-in documentation, is covered with unit tests, and is released as a package on PyPI[3]. All important settings are placed to a separate configuration file and so it is easy to experiment with them.

The implementation relies on some open-source tools [20, 21, 22].

In this section, results are reported for experiments with 2 melodic lines and 16 measures starting from (G4, C5) and ending with (C4, C5). Properties to be evaluated are weighted so that explicity of climax has 3 times more impact than any other property.

Table 1: Scores of sample pieces

	Human	MCS	MCBS	Maximum
Absence of looped pitches	0	0	0	0
Absence of looped fragments	-1	0	0	0
Motion	-0.267	-0.1	-0.033	1
Correlation between lines	0.269	0.321	0.321	1
Explicity of climax	3	3	3	3
Number of skips	1	1	1	1
Absence of downward skips	0	-0.6	0	0
Total reward	3.003	3.621	4.288	6

In Table 1, scores of a piece found with Monte Carlo Beam Search are compared with benchmarks. The piece was found with beam width set to 5 and number of trials schedule set to $(10000 \times 3, 5000 \times 3, 1000 \times 8)$ – this means that the second measure of the piece was chosen after 10000 random continuations and, for example, the eighth measure of the piece was chosen after 1000 random continuations. One of the benchmarks is achieved by Monte Carlo Search with 53000 random trials (i.e., total number of random trials is the same). The other benchmark relates to a piece manually

[2] `https://github.com/Nikolay-Lysenko/rl-musician`

[3] `https://pypi.org/project/rl-musician/0.3.0/`

created by the author in about 10 minutes. Ideally, scores should be collected during multiple independent experiments and then statistical significance of difference should be tested, but it is not done yet.

The results look like scores for climax explicity, absence of looped pitches, absence of looped fragments, and absence of downward skips impose hard constraints. If so, the goal is to maximize sum of scores for lines independence subject to counterpoint rules and above new constraints.

Table 2 demonstrates found by Monte Carlo Beam Search piece for which the scores are reported.

Table 2: The piece found with Monte Carlo Beam Search

			C6												
				B5											
					A5										
		G5				G5									
							F5								
	E5							E5							
									D5						
C5	C5									C5		C5	C5		C5
		B4									B4			B4	
			A4												
G4				G4				G4		G4	G4				
					F4				F4			F4			
						E4							E4		
							D4							D4	
															C4

Subjective evaluation of music quality can be used for revising setup of the problem. Probably, 16 measures are too much, because long step motions are resembling descending scales.

5 Conclusion

This paper introduces a new approach to composition of music without usage of existing pieces. For a problem of first species counterpoint writing, the approach finds pieces that maximize weighted sum of associated with particular musical properties scores subject to boundary constraints. These boundary constraints (the first measure and the last measure) are user-defined and weights for the sum of scores are user-defined too.

Results are good from formal metrics point of view – the method outperforms random search and manual trials. However, created pieces are too simple to be enjoyable music. This is an inevitable consequence of too strict limitations of first species counterpoint. To eliminate them, higher species can be chosen or frameworks other than species counterpoint can be explored.

If the problem becomes computationally complex after introducing more musical freedom, concepts from reinforcement learning such as value function can be involved. State-free search can be replaced with a search where decision are made based on observations and estimated values of states and actions.

References

[1] Daniel Johnson. Generating Polyphonic Music Using Tied Parallel Networks. In *International Conference on Evolutionary and Biologically Inspired Music and Art*, pages 128–143, 03 2017.

[2] Christine Payne. MuseNet. `openai.com/blog/musenet`, Apr 2019.

[3] Liangrong Yi and Judy Goldsmith. Automatic Generation of Four-Part Harmony. In *Fifth UAI Bayesian Modeling Applications Workshop*, Jan 2007.

[4] B.D. Smith and G.E. Garnett. Reinforcement Learning and the Creative, Automated Music Improviser. *Evolutionary and Biologically Inspired Music, Sound, Art and Design. EvoMUSART. Lecture Notes in Computer Science*, 7247, 2012.

[5] Natasha Jaques, Shixiang Gu, Richard E. Turner, and Douglas Eck. Generating Music by Fine-Tuning Recurrent Neural Networks with Reinforcement Learning. 2016.

[6] Nikhil Kotecha. Bach2Bach: Generating Music Using a Deep Reinforcement Learning Approach. *arXiv e-prints*, page arXiv:1812.01060, Dec 2018.

[7] Harish Kumar and Balaraman Ravindran. Polyphonic Music Composition with LSTM Neural Networks and Reinforcement Learning. *arXiv e-prints*, page arXiv:1902.01973, Feb 2019.

[8] Dorien Herremans and Kenneth Sörensen. Composing First Species Counterpoint with a Variable Neighbourhood Search Algorithm. *Journal of Mathematics and the Arts*, 6(4):169–189, 2012.

[9] Dorien Herremans and Kenneth Sörensen. Composing Fifth Species Counterpoint Music with Variable Neighborhood Search. *Expert Systems with Applications*, 2013.

[10] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 101–109, 2019.

[11] Jose David Fernandez and Francisco Vico. AI Methods in Algorithmic Composition: A Comprehensive Survey. *Journal of Artificial Intelligence Research*, 48:13–582, Feb 2013.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv e-prints*, page arXiv:1312.5602, Dec 2013.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. 2014.

[14] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *arXiv e-prints*, page arXiv:1609.05473, Sep 2016.

[15] R Devon Hjelm, Athul Paul Jacob, Tong Che, Adam Trischler, Kyunghyun Cho, and Yoshua Bengio. Boundary-Seeking Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1702.08431, Feb 2017.

[16] Ronald J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.*, 8(3-4):229–256, May 1992.

[17] Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. *arXiv e-prints*, page arXiv:1705.10843, May 2017.

[18] Tristan Cazenave. Monte Carlo Beam Search. *IEEE Transactions on Computational Intelligence and Ai in Games*, 4:68–72, Mar 2012.

[19] Tristan Cazenave. Nested Monte Carlo Search. *IJCAI International Joint Conference on Artificial Intelligence*, pages 456–461, Jan 2009.

[20] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv e-prints*, page arXiv:1606.01540, Jun 2016.

[21] Travis E. Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[22] Colin Raffel and Daniel P. W. Ellis. Intuitive Analysis, Creation and Manipulation of MIDI Data with pretty_midi. In *Music Information Retrieval Late Breaking and Demo Papers, 15th International Conference on*, 2014.