Dynamic programming algorithms applied to musical counterpoint in process composition: an example using Henri Pousseur's *Scambi*

Louis J. Cochrane & Derek Gatherer

Division of Biomedical & Life Sciences

Lancaster University

Lancaster LA1 4YW, UK

email: d.gatherer@lancaster.ac.uk

Twitter: @DerekGatherer

Abstract:

The Needleman-Wunsch process is a classic tool in bioinformatics, being a dynamic programming algorithm that performs a pairwise alignment of two input biological sequences, either protein or nucleic acid. A distance matrix between the tokens used in the sequences is also required as input. The distance matrix is used to generate a positional pairwise similarity matrix between the input sequences, which is in turn used to generate a dynamic programming matrix. The best path through the dynamic programming matrix is navigated using a traceback procedure that maximises similarity, inserting gaps as necessary. Needleman-Wunsch can align both nucleic acids or proteins, which use alphabets of size 4 and 20 tokens respectively. It can also be applied to any other kind of sequence where distance matrices can be specified. Here, we apply it to chains of Pousseur's *Scambi* electronic music fragments, of which there are 32, and which Pousseur categorised by their sonic properties, thus permitting the consecutive construction of distance, similarity and dynamic programming matrices. Traceback through the dynamic programming matrix thus produces contrapuntal duet compositions in which two *Scambi* chains are played in the maximally euphonious manner, providing also an illustration of the principles of biological sequence alignment in sound.

Keywords: *Scambi*; Bio-Art; Bio-Music; music; counterpoint; process composition; dynamic programming; Needleman-Wunsch algorithm; global alignment

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 June 2020 doi:10.20944/preprints202006.0359.v1

Introduction

Henri Pousseur's *Scambi* was composed in 1957 (Pousseur 1959). Pousseur's recording of one of his own realizations of *Scambi* is available on YouTube[1] and is frequently included in anthologies of early electronic music. Other recorded versions, by Pousseur himself and by Luciano Berio, survive but are not publicly available. Marc Wilkinson's version may have been lost (Wilkinson 1958). More recently, realizations of *Scambi* have been created by Andre Castro and Rudy Ceccato, as well as a jointly realized version by Robin Fencott and Simon Harris. These are available from the website of the AHRC-funded *Scambi* project[2].

These several versions of *Scambi* exist, and many more may be created, because *Scambi* is an open form process piece (Dack 2009). However, it is neither aleatory nor does it permit improvisation. The raw material for *Scambi* is a set of 32 fragments ("sequences") of electronically generated sound of either 30 or 42 seconds in length, together with a set of imposed rules about how those sequences may be put together[3]. *Scambi* is thus in some respects a kind of musical game. The rules that Pousseur defined for assembling the sequences relate to the most euphonious connection of the end of one sequence with the start of the next, with the goal of creating a smooth composition from fragmentary parts.

The rules of *Scambi* also allow for sequences to be played simultaneously, producing polyphonic compositions (Dack 2009). Pousseur's sequences were recorded in mono, so for a polyphonic piece they may be distributed around a stereo (or higher dimension) system. In a previous work, a 5-channel *Scambi* realization was produced[4] to create alternating periods of thicker and sparser sonic texture – a stripy effect inspired by the fruit fly *Drosophila fushi tarazu* gene expression pattern (Gatherer 2020). As professional biologists interested in electronic music, we thus endeavour to bring the spirit of the Bio-Art movement[5] to musical composition. In the present paper, we describe a further Bio-Music project, this time investigating the use of dynamic programming algorithms to generate binaural counterpoint between two previously composed *Scambi* monophonic chains of sequences. Dynamic programming has been applied in bioinformatics to align both protein sequences and nucleic acid

[1] https://www.youtube.com/watch?v=E6vlOFApLnQ
[2] http://scambi.mdx.ac.uk/
[3] It should be noted that a "sequence" in Scambi is the individual token from which a compositional chain is created. In biology, a "sequence" is not the individual element, but the whole chain, and the individual tokens are either "bases" (for nucleic acids) or "residues" (for proteins).
[4] https://www.youtube.com/watch?v=X6qDwhmZ01k
[5] As exemplified by our colleague at Lancaster, Dr Rod Dillon: https://roddillon.com/

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 June 2020 doi:10.20944/preprints202006.0359.v1

sequences. We provide software written in Python (*Scambi Kit*) that automates both the creation of *Scambi* chains and their contrapuntal alignment.

Sound Files and Software

Pousseur's original magnetic tape sequences have been kindly provided by Dr John Dack (Middlesex University) in aiff format. These were converted into 16-bit wav format in Audacity[6]. *Scambi Kit* (available at https://github.com/ljcochrane/Scambi_Kit) was written in Python version 3.7.2 with the Python library *PyDub* used to aid in audio manipulation and processing. Figure 1 shows the opening menu of *Scambi Kit*.

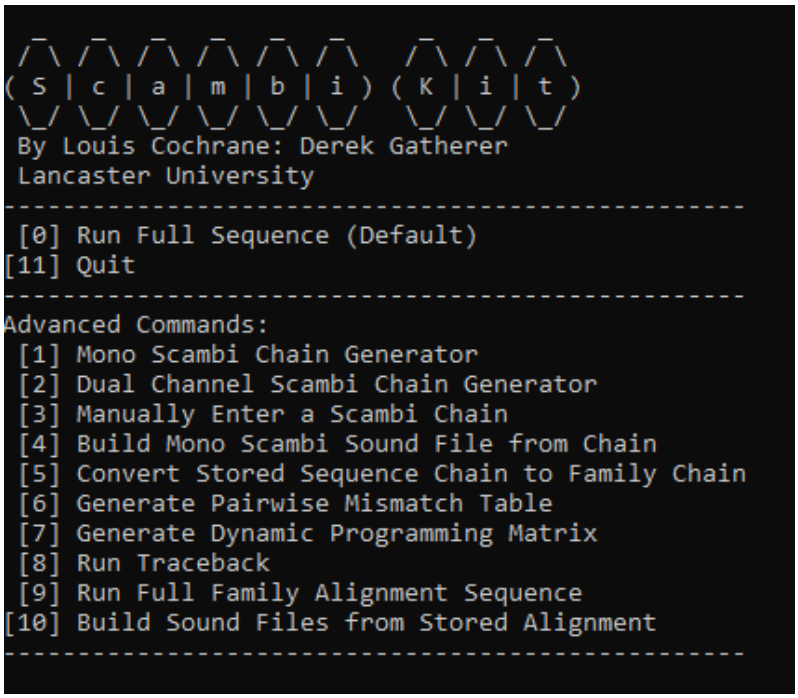

Figure 1: Screenshot of *Scambi Kit* main menu

As in the previous paper (Gatherer 2020), a reversed version is created for each *Scambi* sequence. Table 1 shows the sequence classification. Pousseur's start and end binary digit codes refer to the musical characteristics of the first and second half of each sequence. From left to right: pitch (low '0' to high '1'), tempo (slow '0' to fast '1'), sound quality (dry '0' to reverberated '1') and continuity (inclusion of pauses '0' to continuous sound '1'). So, in Table 1, sequence family A starts 0110 – low, fast, reverberated and with pauses – and ends 1100 – high, fast, dry and with pauses. We render the binary strings in denary for ease of comparison.

[6] https://www.audacityteam.org/

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 June 2020 doi:10.20944/preprints202006.0359.v1

		Binary		Denary	
Family	sequence	start	end	start	end
A	1,2	0110	1100	6	12
Ar	1r,2r	1100	0110	12	6
B	3,4	0101	1111	5	15
Br	3r,4r	1111	0101	15	5
C	5,6	1100	0101	12	5
Cr	5r,6r	0101	1100	5	12
D	7,8	1111	0110	15	6
Dr	7r,8r	0110	1111	6	15
E	9,10	1111	1000	15	8
Er	9r,10r	1000	1111	8	15
F	11,12	1100	1011	12	11
Fr	11r,12r	1011	1100	11	12
G	13,14	1000	1100	8	12
Gr	13r,14r	1100	1000	12	8
H	15,16	1011	1111	11	15
Hr	15r,16r	1111	1011	15	11
I	17,18	0010	0101	2	5
Ir	17r,18r	0101	0010	5	2
J	19,20	0001	0110	1	6
Jr	19r,20r	0110	0001	6	1
K	21,22	0101	0001	5	1
Kr	21r,22r	0001	0101	1	5
L	23,24	0110	0010	6	2
Lr	23r,24r	0010	0110	2	6
M	25,26	1011	0001	11	1
Mr	25r,26r	0001	1011	1	11
N	27,28	1000	0010	8	2
Nr	27r,28r	0010	1000	2	8
O	29,30	0001	1000	1	8
Or	29r,30r	1000	0001	8	1
P	31,32	0010	1011	2	11
Pr	31r,32r	1011	0010	11	2

Table 1: Binary to denary conversion of the binary string classifications of the standard *Scambi* sequence set, modifying Decroupet's table in Dack (2004).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 June 2020 doi:10.20944/preprints202006.0359.v1

Implementation

Basic chain generation

Menu option 1 of *Scambi Kit* generates a chain of *Scambi* sequences following the rules described by Dack (2009) and the previous paper (Gatherer 2020). The user can specify the length of the chain and can either start on a given sequence or with a randomly selected sequence. Figure 2 shows a typical output.

```
1


How long should the scambi chain be?
10
Enter the first sequence to use, or enter 'r' for a random first sequence.
r


Chain Generation Complete
-------------------------
2r
7r
3r
4
16r
26
26r
31r
18
4
-------------------------
```

Figure 2: Screenshot of *Scambi Kit* menu 1 output

Table 1 may be used to check that this chain obeys the rules, as follows:

		Binary		Denary	
Family	sequence	start	end	start	end
Ar	1r,2r	1100	0110	12	6
B	3,4	0101	1111	5	15
Br	3r,4r	1111	0101	15	5
Dr	7r,8r	0110	1111	6	15
Hr	15r,16r	1111	1011	15	11
I	17,18	0010	0101	2	5
M	25,26	1011	0001	11	1
Mr	25r,26r	0001	1011	1	11
Pr	31r,32r	1011	0010	11	2

Table 2: Sub-table of Table 1 displaying sequences (yellow) generated in Figure 2.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 June 2020 doi:10.20944/preprints202006.0359.v1

The chain generated in Figure 2 is:

2r, 7r, 3r, 4, 16r, 26, 26r, 31r, 18, 4

which Table 2 shows represents sequences beginning and ending, in denary notation:

12-6, 6-15, 15-5, 5-15, 15-11, 11-1, 1-11, 11-2, 2-5, 5-15

The matching of the end of each sequence with the start of the succeeding sequence is thus illustrated.

Dual chain generation

Option 2 performs the same process, but generates two chains, as shown in Figure 3.

```
2


How long should the scambi chain be?
10
Enter the first sequence to use, or enter 'r' for a random first sequence.
r


Chain Generation Complete
-------------------------
First Sequence: 4


Left Channel: ['9', '27', '17', '6r', '12', '15', '4r', '21']


Right Channel: ['7', '19r', '30', '9r', '4r', '17r', '28r', '29r']


Last Sequence: 22r
-------------------------
```

Figure 3: Screenshot of *Scambi Kit* menu 2 output

The dual channel output begins with a single sequence that bifurcates into two simultaneously running chains. The reverse occurs at the end, with both chains ending in a single sequence. In Figure 3, the random starting sequence is 4 (5-15), which then serves to generate one chain continuing with 9 (15-8) and another with 7 (15-6). The two chains then evolve according to the rules, until the 9th sequence pair is reached. In Figure 3 these are 21 (5-1) and 29r (8-1), which both lead into the final sequence, 22r (1-5).

The duet generated by the dual chain function has no provision for counterpoint. Each generated line is of the same length and played simultaneously. Since each may be following a very different set of start and end values (until these converge in the second last sequence), there may be sonic clashes. One might even say that there is a danger of “dissonance”, if such a concept may be used in the context of this kind of electronic music.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 June 2020 doi:10.20944/preprints202006.0359.v1

Contrapuntal generation

To generate a contrapuntal composition, menu 3 is first applied to generate a chain. The top chain from the duet composition in Figure 3 is entered manually.

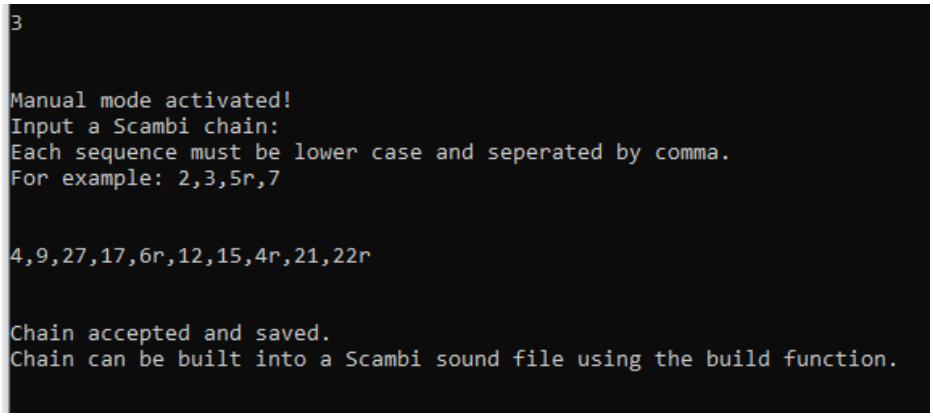

Figure 4: Screenshot of *Scambi Kit* menu 3 output

Menu 5 is then used to convert this into a chain in Family notation (see Table 1)

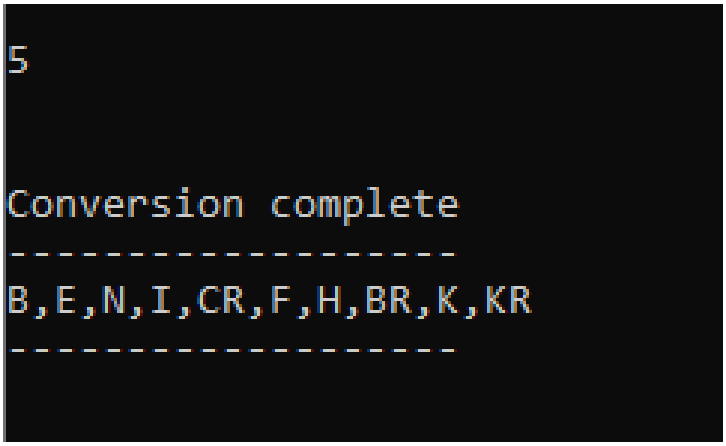

Figure 5: Screenshot of *Scambi Kit* menu 5 output

This is then repeated for the lower chain. We now have two chains in family notation as follows:

B	E	N	I	Cr	F	H	Br	K	Kr
B	D	Jr	O	Er	Br	Ir	Nr	Or	Kr

Table 3: Two chains parsed into Family notation by menu option 5.

Menu option 6 is now used to generate a pairwise scoring matrix between these two chains (Figure 6).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 June 2020 doi:10.20944/preprints202006.0359.v1

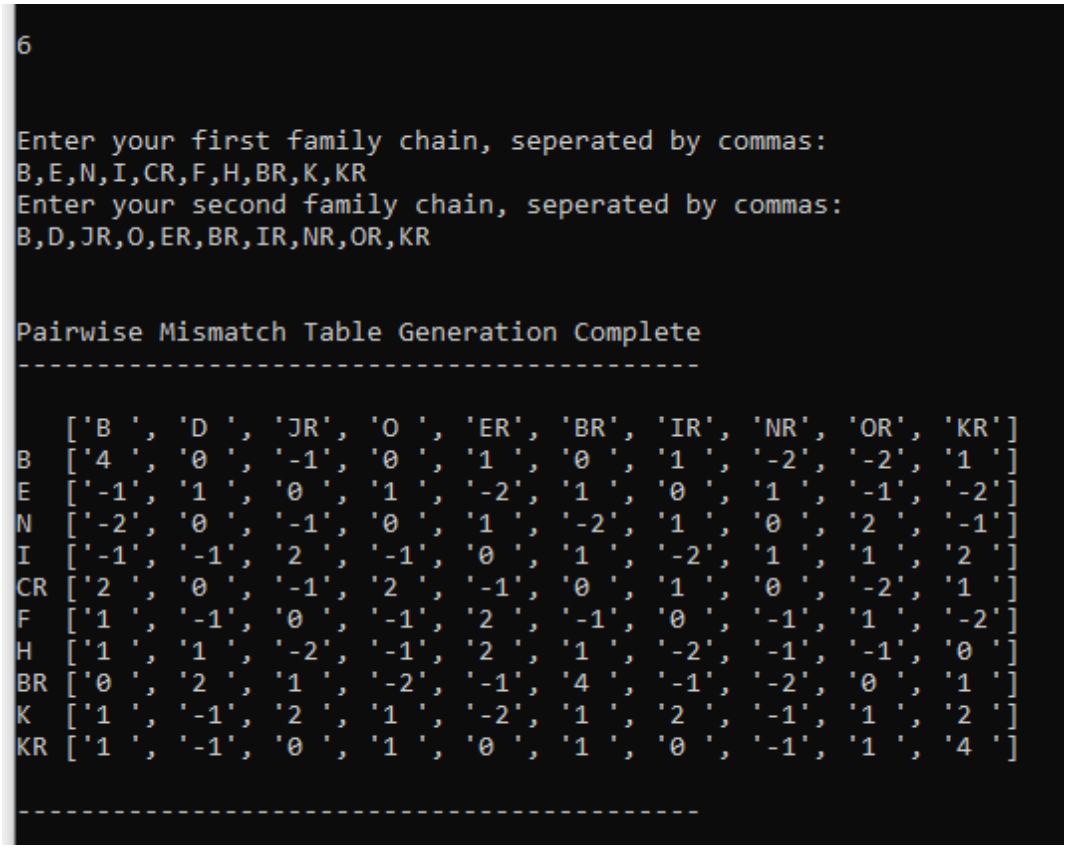

Figure 6: Screenshot of *Scambi Kit* menu 6 output

Figure 6 shows a pairwise similarity scoring matrix between each sequence in the two chains. For instance, in the top left cell of the matrix, "4" is the score of B compared to B. Moving down one, "-1" is the score of B compared to E, and so on. The basis of the scores may be understood by reference to Table 1. Family B has binary notation of 0101 1111. Compared to itself, there is a Hamming Distance of zero. This is designated with a top score of 4. Family E has binary notation of 1111 1000. The binary string of family B has a Hamming Distance of 5 to family E. Therefore the score of B compared to E is the top score minus the Hamming Distance, i.e. 4 - 5 = -1. Similarly, Family N has binary notation of 1000 0010, with a Hamming Distance to Family B of 6. The score for this combination is therefore 4 – 6 = -2. The maximum score of 4 is arbitrary, and was chosen by trial and error to generate the most amenable output for the next stage. Pairwise scoring matrices are standard in bioinformatics in the pairwise comparison of both proteins and nucleic acids.

Menu option 7 is now used to generate a dynamic programming matrix (Figure 7). A gap score must be entered. The more negative the gap score the less likely that a gap will be generated in the final alignment of the two chains.

```
7


Please enter a gap Score:
-2


Dynamic Programming Matrix Generation Complete
-----------------------------------------------

    ['B  ', 'D  ', 'JR ', 'O  ', 'ER ', 'BR ', 'IR ', 'NR ', 'OR ', 'KR ']
B   ['4  ', '2  ', '0  ', '-2 ', '-4 ', '-6 ', '-8 ', '-10', '-12', '-14']
E   ['2  ', '5  ', '3  ', '1  ', '-1 ', '-3 ', '-5 ', '-7 ', '-9 ', '-11']
N   ['0  ', '3  ', '4  ', '3  ', '2  ', '0  ', '-2 ', '-4 ', '-5 ', '-7 ']
I   ['-2 ', '1  ', '5  ', '3  ', '3  ', '3  ', '1  ', '-1 ', '-3 ', '-3 ']
CR  ['-4 ', '-1 ', '3  ', '7  ', '5  ', '3  ', '4  ', '2  ', '0  ', '-2 ']
F   ['-6 ', '-3 ', '1  ', '5  ', '9  ', '7  ', '5  ', '3  ', '3  ', '1  ']
H   ['-8 ', '-5 ', '-1 ', '3  ', '7  ', '10 ', '8  ', '6  ', '4  ', '3  ']
BR  ['-10', '-6 ', '-3 ', '1  ', '5  ', '11 ', '9  ', '7  ', '6  ', '5  ']
K   ['-12', '-8 ', '-4 ', '-1 ', '3  ', '9  ', '13 ', '11 ', '9  ', '8  ']
KR  ['-14', '-10', '-6 ', '-3 ', '1  ', '7  ', '11 ', '12 ', '12 ', '13 ']

-----------------------------------------------
```

Figure 7: Screenshot of *Scambi Kit* menu 7 output

The dynamic programming algorithm is created using the similarity matrix and the gap score. For those wishing to look more deeply into how this matrix is generated, the Wikipedia page https://en.wikipedia.org/wiki/Needleman–Wunsch_algorithm provides the easiest introduction. A deeper introduction can be found in Lesk (2014 pp. 182-196) and a more formal treatment in the original paper (Smith and Waterman 1981). The exact values in the dynamic programming matrix will vary according to the gap score adopted (in this case -2), and of course will be different for different input chains. The dynamic programming matrix is converted into an alignment by running the traceback algorithm using menu option 8, generating the final alignment (Figure 8). Option 9 runs the entire process.

```
Sequence Alignment Complete
------------------------------------------------------------

   ['* ', 'x ', '  ', 'x ', 'x ', '  ', 'x ', '* ', 'x ', '  ', '  ', '* ']
X  ['B ', 'E ', 'N ', 'I ', 'CR', 'F ', 'H ', 'BR', 'K ', '- ', '- ', 'KR']
Y  ['B ', 'D ', '- ', 'JR', 'O ', '- ', 'ER', 'BR', 'IR', 'NR', 'OR', 'KR']

* = Match    x = Mismatch    Blank = Gap
------------------------------------------------------------
```

Figure 8: Screenshot of *Scambi Kit* menu 8 output

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 June 2020 doi:10.20944/preprints202006.0359.v1

In the case shown, the chains play in duet for the first two sequences, then the lower chain rests while the upper chain plays sequence N, they then duet again for a further two sequences, followed by another rest in the lower chain. Three more duet chains are followed by a rest of two sequences in the upper chain.

Audio output can be generated using option 10 (Figure 9). A single mono sound file can be produced or one mono file corresponding to each chain. The latter is preferable as each file can be entered into Audacity and a stereo effect added (Figure 10).

```
10


Last generated alignment:
['B', 'E', 'N', 'I', 'CR', 'F', 'H', 'BR', 'K', '-', '-', 'KR']
['B', 'D', '-', 'JR', 'O', '-', 'ER', 'BR', 'IR', 'NR', 'OR', 'KR']


Do you wish to build this alignment into two sound files? (Y/N)
y


Build complete!
Sound files exported to Alignment (Left/Right) Channel.wav
```

Figure 9: Screenshot of *Scambi Kit* menu 10 output

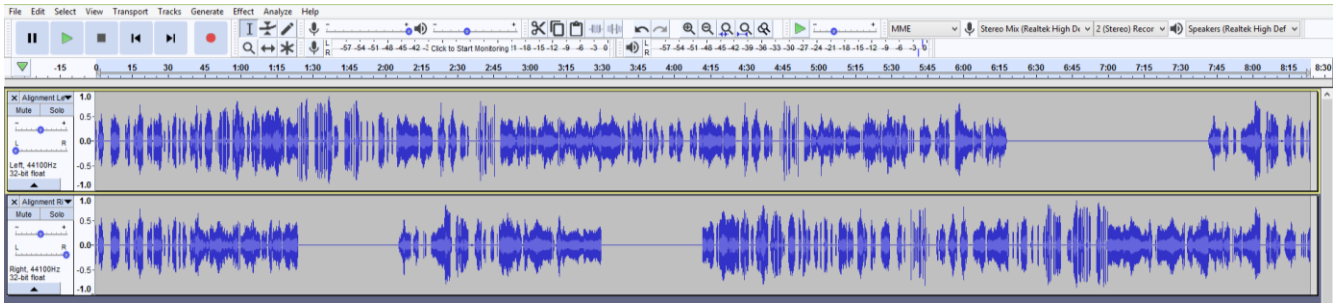

Figure 10: Output in Audacity

Fully Automated Contrapuntal generation

To simplify the process of creating a contrapuntal composition, option 0 (Figure 1), sequentially runs the processes seen above. The two initial chains can either be entered manually or generated automatically (Figure 11). In this mode, the sequence, rather than family notation is used throughout. This eliminates the need for a stochastic choice to be made by the program when converting back to sequence notation in defining which sequences will be used to build the final composition.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 June 2020 doi:10.20944/preprints202006.0359.v1

In this mode different sequences of the same family are displayed as matches when viewing the alignment. (Figure 12)

```
Manual Mode [M]
To input your own Scambi Chain.

OR

Automatic mode [A]
To automatically build a chain according to Pousseur's rules.
```

Figure 11: Manual or automatic selection in fully automated contrapuntal generation

```
Sequence Alignment Complete
------------------------------------------------------------------

    ['*  ', '   ', 'x  ', '   ', '*  ', '*  ', '   ', '   ', '   ', '   ', '*  ', 'x  ', '*  ']
1   ['19 ', '28 ', '27r', '-  ', '13 ', '2r ', '7r ', '15r', '-  ', '-  ', '12r', '6  ', '18r']
2   ['19 ', '-  ', '8r ', '10 ', '14 ', '13r', '-  ', '-  ', '30r', '25r', '15 ', '4r ', '18r']

* = Match    x = Mismatch    Blank = Gap
------------------------------------------------------------------
```

Figure 12: Alignment output in fully automated contrapuntal generation.

Remaining Issues

Scambi Kit uses a set of *Scambi* files converted to a uniform length of 42 seconds. The 30 second sequences are therefore decelerated in tempo. This is necessary when two sequences of different lengths are to be aligned. This issue was discussed in the previous paper (Gatherer 2020).

Acknowledgments

Dr John Dack (Middlesex University) kindly provided the component *Scambi* sequences, as well as much encouragement and personal reminiscences of Henri Pousseur. We are also very grateful to Dr Alan Marsden (Lancaster Institute for the Contemporary Arts) for his advice on many aspects of the project, and to Dr Rod Dillon (Division of Biomedical & Life Sciences, Lancaster University) for his pioneering advocacy for Bio-Art at Lancaster. This work was completed as part of an MSci degree in Natural Sciences for author LJC.

Supplementary Materials

Scambi Kit is available at https://dx.doi.org/10.17635/lancaster/researchdata/373, along with three representative *Scambi* contrapuntal compositions created using the software. Any future updates of Scambi Kit will be made available at https://github.com/ljcochrane/Scambi_Kit. Pousseur's *Scambi*

components, along with material relevant to the preceding paper (Gatherer 2020), are available at: https://doi.org/10.17635/lancaster/researchdata/334.

References

Dack, J. (2004). "Notes on the Realization of Scambi." Retrieved 9th December 2019, 2019, from http://www.Scambi.mdx.ac.uk/documents.html

Dack, J. (2009). The electroacoutic music of Henri Pousseur and the 'open' form. The Modernist Legacy: Essays on New Music. B. Heile. London, Routledge**:** 177-190.

Gatherer, D. (2020). "Scambi fushi tarazu: A Musical Representation of a Drosophila Gene Expression Pattern." Preprints **2020, 2020010026** (doi: 10.20944 preprints/202001.0026.v1).

Lesk, A. M. (2014). Introduction to Bioinformatics. Oxford, UK, Oxford University Press.

Pousseur, H. (1959). "Scambi." Gravesaner Blätter **IV**: 36-54.

Smith, T. F. and M. S. Waterman (1981). "Identification of common molecular subsequences." J Mol Biol **147**(1): 195-197.

Wilkinson, M. (1958). "Two Months in the 'Studio di Fonologia'." The Score **22/Feb**: 41-48.