Group project: Turning Charlie Parker into Ornette Coleman

Andrew Martin, James Schoff, Thomas Carey
1704466, 1705565, 1704570

May 18, 2018

Report submitted for **APP MTH 3001** at the School of Mathematical Sciences, University of Adelaide

Project Area: **Applied Probability**

In submitting this work I am indicating that I have read the University's Academic Honesty Policy. I declare that all material in this assessment is my own work except where there is clear acknowledgement and reference to the work of others.

I give permission for this work to be reproduced and submitted to other academic staff for educational purposes.

Executive Summary

The purpose of this report is to generate new jazz improvisations in the style of jazz saxophonist Charlie Parker. This is achieved by using three different stochastic, Markovian processes to produce new improvisations based on of the original pieces within Charlie Parker's Omnibook. Each new improvisation uses the same 'lead' timing and length as original piece, while randomly choosing notes.
The first process is formed based only on the order of notes used within the original piece, where the proportion of times that one note transitions to the next (in one step) is used to form a probability for that transition. Each note within the new improvisation is then chosen randomly, dependant only on the previous note played. This improvisation is then created using the same chord progression and timing as the original piece. This approach lead to a piece with a good underlying flow as it mimicked the original chord progression, but suffered from note clashing due to certain notes not partnering with particular chords.

The second process is formed by considering that certain notes 'work' with certain chords. The approach was to, create note-chord pairs, such that the probability matrix looked at the transition of each note-chord pair within the original piece. Each new note-chord pair was then chosen randomly dependent on the previous note-chord pair played. This improvisation is then created using the timing of the 'lead' part from the original piece. This improvisation removed the clashing from the first process, but due to the chords been allocated randomly, the improvisation lacked flow, with almost an aggressive feel to the solo, which is uncharacteristic of the genre.

The third process looked to build on the strengths of the previous processes by choosing notes based on note-chord pairs, but maintaining the original chord progression of the piece. This involved looking to allocate notes based on the pre-allocated timing of chords. For each chord, a probability matrix is formed by the transitions between notes for that particular chord within the original piece. The notes are then chosen by randomly allocating a section for each chord through Markovian process for each unique chord. Overall, this produced a solo which had a strong flow as seen in the first process, but also significantly reducing the clashes within the notes and chords as seen within the second process. One issue was that some transitions had to be randomly generated to avoid particular transition issues. However, this approach significantly improved the listening quality of the piece.

Finally, a 'human' element was added to the third process, by removing notes which didn't sound aesthetically pleasing to hear due to note clashes. As well as this, the note 'velocities' or volumes were changed for the improvisation, to add depth to the playing. This element this created a smoother, cleaner sound which made for a 'good' sounding improvisation.

Overall, the third process produced a solo which was most enjoyable to listen to, due to the combination of the chord progression of the original song partnered with notes that matched the chords. However adding the Human element to the solo produced the cleanest, and most realistic improvisation.

1 Project Aim

The objective of this report is to generate new jazz improvisations in the style of the jazz music of Charlie Parker. Namely, based off the solo in the piece, "Moose the Mooche". Despite the jazz improvisations being created through technical and algorithmic means, the aim of the final product is to have created a jazz solo that is not only technically correct but also has an appealing to listen to.

2 Initial Set up

In order to manipulate the data with `MATLAB`, the data first has to be transferred to a file which `MATLAB` can read and understand. The data representing the original piece is initially within a MusicXML (.mxml) file which is a file format that stores information on the notes, times and volumes. This data is converted into a `MATLAB` .mat file, using the mxl toolbox, which can then be accessed on `MATLAB`. Within `MATLAB`, the data is the loaded in, where the chords and lead melody sections are separated into their own note matrices - where the columns contained start-time, duration, midi channel, pitch, velocity, start-time (seconds), and duration (seconds), respectively.
Notes are denoted within the pitch column with a number between 0 and 127, where each value represents a different octave + semitone. There are 12 semi-tones within each octave, where each greater value of octave has a greater pitch than the previous. Hence note 38 would be represented by 3 octaves + 3 semitone, whilst note 83 would be represented by 6 Octaves + 12 semitones and would be a higher pitch, where 60 is a middle C (on a piano).

Both the lead and chord note matrices are represented by rows which indicate the note played at a particular time during the song. It was found that 24 separate semitones where played in various sequences throughout the song, ranging from note note 54 (represented as 4 octaves + 7 semitones) to note 77 (6 Octaves + 6 semitones). In order to simplify probability matrix, the value of each note from note 54 to 77 is shifted down by 53 to create a range of notes from 1 to 24. It was also found that 16 separate chords were played throughout the piece. The total amount of notes played throughout the piece was 373.

3 First Process

3.1 Background

The first process aims to create a new improvisation of the original piece based on only the order of notes from the lead melody within the original piece of music. Due to this, the state space is set to represent only the notes played within the original piece. As these notes where shifted and represented by numbers 1 to 24, the state space S is the range of notes after the conversion in the set up, i.e.:

$$S = \{1, 2, ..., 23, 24\}$$

The next aim is to form a probability transition matrix that represents the proportion of time one note transitions to the another over the whole original piece. By creating this matrix, we are able to run a stochastic process which is Markovian. It is Markovian as the next note within the new improvisation is only depends on the the current note, and independent of all other notes played.

From this, a random DTMC walk is conducted by running through the total amount of notes played within the original piece of music, which in this case was 373. This sequence of 373 notes represents the sequence of notes played within the newly formed solo. The new improvisation is then backed with the same timing and the same chord progression as the original piece.

3.2 Methods

The method to form the transition probability matrix (represented by $\mathbb{P}$-Matrix) was initially to form a 24×24 zero matrix. Each row and column would represent one of the 24 semi-tones that were played throughout the piece of music, or similarly each element within the state space S. The rows represent the note that was previously played, whilst the columns represent probability of choosing the note played next.
The number of times semitone j immediately follows note i within the whole original piece, excluding the final note, is then calculated, with this number being represented by row i, column j of the previously formed zero matrix. Each row of the matrix was then normalised to 1 by summing each row, then dividing each element in that specific row by that number. This allowed us to construct a $\mathbb{P}$-Matrix, where each row summed to 1.

The final step in forming the new solo was to simulate a random DTMC walk through the length of the number of notes in the original song. This produces a new sequence of notes that make up a new solo based on the original sequence of notes. These new notes are re-shifted up to match the original range of notes from note 54 to 77, and matched with the same timing and chord progression. Hence from this, our new improvisation is formed, outputting a midi file.

3.3 Analysis

3.3.1 Last Note of the Original Song

The manner in which we dealt with the last note of the original song was to consider the transition to the last note but to *ignore* the transition away from the last note - as the solo ends there. This is due to there being no note to transition too, hence the transition is non-existent. The limitation to this method is that the $\mathbb{P}$-Matrix has one less transition to be formed around. Voiding this one transition may cause us to form an inaccurate $\mathbb{P}$-matrix which means that the final solo is based on inaccurate data.

An alternative method is to make the final note transition to the pedal note of the song - or finding the first note played after the solo, which in the case of the original song is an F. This creates an addition transition to form the $\mathbb{P}$-matrix. The limitation to this is that the root note is a guess to what the next note is, and not an actual representation of the evolution of the song. The fundamental consequence is similar to the previous case where a wrong $\mathbb{P}$-matrix is formed. The solo created due to this would then also be based on wrong data.

3.3.2 $\mathbb{P}$-Matrix

The formed $\mathbb{P}$-Matrix has no zero columns indicating that each note is transitioned into at least once from a previous note. Similarly each row sums to 1, as expected. If S_+ denotes a subset of S where the proportions is strictly positive, then $S_+ = S$. Due to this, the probabilities of $P(X_1 = K|X_o = i)$ for $i \in S_+$ is just represented by the initial $\mathbb{P}$-Matrix. For the case of analysis, we set the value of K to note 21. The probabilities for note 17 ($P(X_1 = 21|X_o = i)$) can be seen in Figure 1a. The probabilities of after the 10^{th} step ($P(X_{10} = K|X_o = i)$) can be found by taking the $\mathbb{P}$-matrix to the power of 10. Hence again by making K=21, $P(X_{10} = 21|X_o = i)$ can be seen in Figure 2

	21
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0.0789
13	0
14	0
15	0
16	0
17	0.0909
18	0
19	0.0870
20	0.6000
21	0.1250
22	0.2500
23	0.5000
24	0.3077

(a) $P(X_1 = 21|X_o = i)$

	21
1	0.0565
2	0.0569
3	0.0601
4	0.0560
5	0.0561
6	0.0593
7	0.0596
8	0.0584
9	0.0606
10	0.0592
11	0.0625
12	0.0653
13	0.0610
14	0.0649
15	0.0632
16	0.0666
17	0.0653
18	0.0646
19	0.0667
20	0.0715
21	0.0701
22	0.0705
23	0.0727
24	0.0757

(b) $P(X_{10} = 21|X_o = i)$

Figure 1: The probabilities of playing note 21 after 1 and 10 steps

The general probabilities of ending up in each specific note is after 10 steps seen

in Figure 2, where the rows represent i and the columns represent the different values of K. It can be seen that regardless of what note is initially played, there is some probability of being in any state. This implies that any note can be reached after 10 steps from any initial note. This also indicates that the all the states in S are recurrent. This is further shown in the state diagram of the $\mathbb{P}$-Matrix in Figure 3 where no states appear to be either ephemeral or transient.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0.0036	0.0166	0.0067	0.0317	0.0688	0.0060	0.0762	0.0255	0.0804	0.0631	0.0410	0.0967	0.0265	0.0800	0.0383	0.0467	0.0854	0.0152	0.0568	0.0239	0.0565	0.0183	0.0050	0.0310
2	0.0035	0.0164	0.0066	0.0314	0.0681	0.0060	0.0758	0.0253	0.0801	0.0631	0.0410	0.0970	0.0265	0.0803	0.0385	0.0469	0.0855	0.0152	0.0571	0.0241	0.0569	0.0185	0.0050	0.0312
3	0.0031	0.0150	0.0060	0.0294	0.0643	0.0057	0.0733	0.0236	0.0780	0.0627	0.0408	0.0983	0.0267	0.0816	0.0394	0.0476	0.0868	0.0156	0.0591	0.0253	0.0601	0.0198	0.0052	0.0327
4	0.0035	0.0168	0.0067	0.0319	0.0690	0.0060	0.0765	0.0257	0.0806	0.0633	0.0411	0.0968	0.0265	0.0800	0.0383	0.0467	0.0852	0.0151	0.0566	0.0238	0.0560	0.0181	0.0050	0.0308
5	0.0036	0.0167	0.0067	0.0319	0.0692	0.0060	0.0765	0.0256	0.0807	0.0632	0.0411	0.0967	0.0265	0.0800	0.0382	0.0467	0.0852	0.0151	0.0566	0.0238	0.0561	0.0181	0.0050	0.0308
6	0.0032	0.0155	0.0062	0.0299	0.0652	0.0058	0.0739	0.0241	0.0784	0.0628	0.0408	0.0979	0.0266	0.0812	0.0391	0.0474	0.0865	0.0155	0.0586	0.0250	0.0593	0.0195	0.0051	0.0324
7	0.0032	0.0153	0.0061	0.0298	0.0650	0.0058	0.0738	0.0239	0.0783	0.0627	0.0408	0.0980	0.0266	0.0813	0.0392	0.0474	0.0866	0.0156	0.0587	0.0251	0.0596	0.0195	0.0051	0.0325
8	0.0033	0.0159	0.0063	0.0305	0.0665	0.0059	0.0747	0.0246	0.0791	0.0629	0.0409	0.0975	0.0266	0.0808	0.0388	0.0472	0.0861	0.0154	0.0580	0.0246	0.0584	0.0191	0.0051	0.0319
9	0.0031	0.0150	0.0060	0.0292	0.0639	0.0057	0.0730	0.0235	0.0776	0.0625	0.0407	0.0983	0.0267	0.0817	0.0394	0.0476	0.0870	0.0157	0.0593	0.0254	0.0606	0.0200	0.0052	0.0330
10	0.0032	0.0155	0.0062	0.0300	0.0654	0.0058	0.0741	0.0241	0.0785	0.0628	0.0409	0.0979	0.0266	0.0812	0.0391	0.0474	0.0865	0.0155	0.0585	0.0249	0.0592	0.0194	0.0051	0.0323
11	0.0029	0.0141	0.0056	0.0280	0.0616	0.0055	0.0716	0.0225	0.0764	0.0623	0.0406	0.0990	0.0268	0.0825	0.0400	0.0480	0.0878	0.0159	0.0605	0.0261	0.0625	0.0207	0.0053	0.0339
12	0.0026	0.0131	0.0052	0.0264	0.0584	0.0053	0.0694	0.0211	0.0745	0.0618	0.0403	0.0999	0.0269	0.0835	0.0407	0.0485	0.0889	0.0162	0.0621	0.0272	0.0653	0.0219	0.0054	0.0354
13	0.0030	0.0148	0.0059	0.0289	0.0634	0.0057	0.0727	0.0232	0.0774	0.0625	0.0407	0.0984	0.0267	0.0818	0.0395	0.0477	0.0872	0.0157	0.0595	0.0256	0.0610	0.0201	0.0052	0.0332
14	0.0026	0.0132	0.0053	0.0265	0.0587	0.0053	0.0697	0.0213	0.0748	0.0619	0.0404	0.0999	0.0269	0.0834	0.0406	0.0485	0.0887	0.0162	0.0619	0.0270	0.0649	0.0217	0.0054	0.0351
15	0.0028	0.0138	0.0055	0.0276	0.0608	0.0055	0.0710	0.0221	0.0759	0.0622	0.0405	0.0993	0.0268	0.0828	0.0401	0.0482	0.0880	0.0160	0.0609	0.0264	0.0632	0.0210	0.0053	0.0343
16	0.0024	0.0124	0.0049	0.0254	0.0566	0.0052	0.0684	0.0203	0.0737	0.0617	0.0403	0.1006	0.0270	0.0842	0.0411	0.0489	0.0894	0.0164	0.0629	0.0277	0.0666	0.0224	0.0055	0.0359
17	0.0025	0.0130	0.0052	0.0263	0.0582	0.0053	0.0694	0.0210	0.0745	0.0619	0.0404	0.1001	0.0269	0.0836	0.0407	0.0486	0.0889	0.0162	0.0621	0.0272	0.0653	0.0219	0.0054	0.0353
18	0.0026	0.0132	0.0052	0.0266	0.0588	0.0054	0.0699	0.0213	0.0749	0.0621	0.0405	0.1000	0.0269	0.0835	0.0406	0.0486	0.0886	0.0162	0.0618	0.0270	0.0646	0.0216	0.0054	0.0349
19	0.0024	0.0124	0.0049	0.0254	0.0565	0.0052	0.0683	0.0203	0.0736	0.0617	0.0403	0.1006	0.0270	0.0842	0.0411	0.0489	0.0894	0.0164	0.0630	0.0277	0.0667	0.0225	0.0055	0.0360
20	0.0019	0.0106	0.0042	0.0227	0.0512	0.0048	0.0647	0.0181	0.0705	0.0608	0.0398	0.1020	0.0272	0.0858	0.0423	0.0497	0.0914	0.0169	0.0658	0.0294	0.0715	0.0245	0.0057	0.0384
21	0.0021	0.0112	0.0044	0.0235	0.0528	0.0049	0.0657	0.0188	0.0714	0.0610	0.0399	0.1016	0.0271	0.0852	0.0419	0.0494	0.0908	0.0168	0.0650	0.0289	0.0701	0.0239	0.0057	0.0378
22	0.0020	0.0110	0.0044	0.0233	0.0524	0.0049	0.0654	0.0186	0.0711	0.0610	0.0398	0.1017	0.0272	0.0854	0.0420	0.0495	0.0910	0.0168	0.0652	0.0291	0.0705	0.0241	0.0057	0.0379
23	0.0018	0.0102	0.0041	0.0221	0.0500	0.0047	0.0638	0.0176	0.0697	0.0606	0.0396	0.1023	0.0272	0.0861	0.0426	0.0499	0.0919	0.0171	0.0664	0.0298	0.0727	0.0249	0.0058	0.0390
24	0.0016	0.0094	0.0037	0.0207	0.0471	0.0045	0.0616	0.0165	0.0678	0.0598	0.0391	0.1028	0.0273	0.0867	0.0432	0.0502	0.0932	0.0174	0.0681	0.0308	0.0757	0.0263	0.0059	0.0407

Figure 2: The probabilities of $P(X_{10} = K|X_o = i)$ for $i \in S_+$

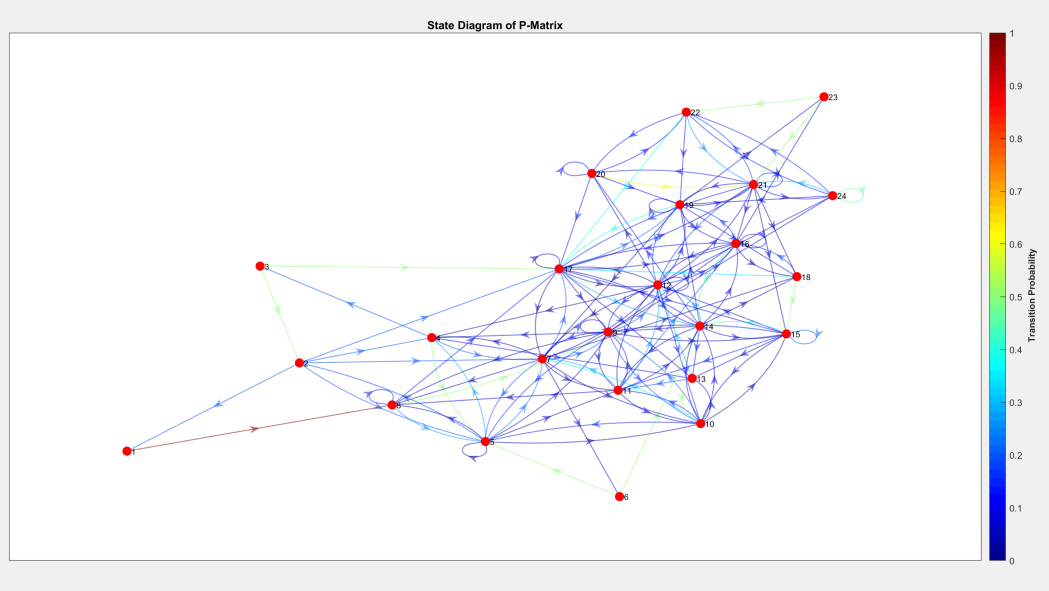

Figure 3: The state diagram of the $\mathbb{P}$-Matrix

3.3.3 Expected Hitting Time

The generic formula for finding the mean hitting time is:

$$k_i^{\mathcal{A}} = \begin{cases} 0 & for\ i \in \mathcal{A} \\ 1 + \sum_{j \notin \mathcal{A}} p_{i,j} k_j^{\mathcal{A}} & for\ i \notin \mathcal{A} \end{cases}$$

When looking at the case for mean hitting time of K, which is state 21 in our case, this formula becomes:

$$k_i^{21} = \begin{cases} 0 & for\ i = 21 \\ 1 + \sum_{j \neq 21} p_{ij} k_j^{21} & for\ i \neq 21 \end{cases}$$

This represents all (k_i)'s for all (i)'s in (S), not including note 21. The solved equations for can be seen in Figure 4. The equations were solved using Matlab (see Appendix A.1 for code).

1	27.5231
2	27.0079
3	25.2547
4	27.5527
5	27.5807
6	25.7942
7	25.8815
8	26.5231
9	25.2497
10	26.3688
11	24.5261
12	20.8701
13	25.3506
14	22.0076
15	24.4948
16	21.9123
17	21.5016
18	24.0666
19	20.2679
20	7.2614
21	0
22	13.5834
23	7.7917
24	8.6335

Figure 4: The Expected Hitting times of state 21 from state i

These values are sensible because as we know, the state can be reached from every other state (so all of these times exist and are $< \infty$).

3.4 Resultant Improvisation

The improvisation generated through this approach has a seemingly good flow and feel to it, however there is a significant number of clashes between the produced notes and the backing chords. However the strongest parts to the improvisation is the chord progression and timing, which were not produced by the original process. Whilst the produced notes have various sections which sound good, too much of the song is effected by the clashing of notes to be considered and overly good and enjoyable improvisation.

4 Second Process

4.1 Background

The second process aims to take into account how notes and chords are paired throughout the original song, as well as how these 'note-chord' pairs transition to other note-chord pairs. Due to this, the state space for this process is represented by each of the 24 semi-tones paired with each of the 16 unique chords, which creates 384 pairs. Each unique is chord is given an index from 0 to 15. Due to this state space T is given by:

$$T = \{1, 2, ..., 384\}$$

Elements 1 to 24 are represented by each note paired with index 0 chord, elements 25 to 48 are represented by each note paired with the index 1 chord, etc.

The probability transition matrix ($\mathbb{P}$-Matrix) is then formed looking at the proportion of note-chord pair transitions. Similarly to the first process, this is Markovian. This is due to the fact that the next note-chord pair that is produced is only dependent on the previous note-chord played, and independent of all previous note-chord pairs. From this, a random DTMC walk is conducted over the 373 steps, producing a note-chord pair for each step, and creating the basis of the improvisation. The improvisation is then formed based on the produced notes-chord pairs with the timing of the original piece.

4.2 Method

The manner in which producing the $\mathbb{P}$-Matrix was to first create a 384×384 zero matrix which would initially represent the $\mathbb{P}$-Matrix. As for first process, each row and column represent one element from the state space T, with the rows representing the note-chord currently being played and the columns represent the new note-chord pair that could be next played. Each chord within the original piece is then iterated through. Based on the start and end time for the chord, each lead note transition in that time causes the appropriate element in the set up $\mathbb{P}$-Matrix to be incremented by 1 for $(note_i, chord_i) \rightarrow (note_j, chord_j)$. After this, rows which have values in are normalised to 1 in the same way as the first process.

After this, the final step in forming the improvisation was to simulate the random DTMC walk through the 373 steps. This produced a new combination/sequence of 373 note-chord pairs that would create the new improvisation. The note and chord values are then shifted back to their original value, i.e. the notes were shifted back to the range of 54-77 whilst the chords converted from their index into its notes. The same timing used within the original piece is also used for the new improvisation. The improvisation is then created using a midi file.

4.3 Analysis

4.3.1 Mean Expected Hitting Times

The formula for finding the mean hitting time is the same is in the first approach. In order to interpret the mean hitting time for this process, we need to remove the rows and columns which have no values in them. This is because the matlab script would be unable to produce the mean hitting times otherwise. Hence the K we are looking for is the (K^{th}) row that has an initial transition in it. 152 out of the 384 note-chord pairs pass this. For this case we will look at K=1, which represent the mean hitting time of the first non-zero note-chord pair (i.e ordered pair (1,1)) from the other note-chord pairs.For this case, we have

$$k_i^1 = \begin{cases} 0 & for\ i = 1 \\ 1 + \sum_{j \neq 1} p_{ij} k_j^1 & for\ i \neq 1 \end{cases}$$

This represents all (k_i)'s for all (i)'s in 151 viable note-chord pairs, not including note 1. The solved equations for K=1 can be seen in the Appendix A.2. The equations were solved using Matlab (See Appendix A.1).

4.4 Resultant Improvisation

The resultant improvisation fixes the main issue from the first process, such that the notes no longer significantly clash with the backing chords. However due to the note-chord pairs being chosen randomly, there is seemingly no flow to the improvisation. This is due to the constant changing of backing chords with no particular patten, with the resultant product having a somewhat violent and choppy feel. This is the opposite to the first process where the chord progression from the original piece gives that improvisation a better flow and feel to it. Overall, it appears the importance of having a flowing chord progression as seen within the first process is more important to the end product than having notes that don't constantly clash with the backing chords.

5 Third Process

5.1 Background

The third process aims to build on the strengths of both the first and second processes. It uses the premise that notes 'work' better with certain chords, forming the note-chord pairs as seen in the second process, but the chord progression follows that from the original piece. This implies that each new note-chord pair is chosen randomly dependent on the previous note-chord pair, but the chord from the new pair is already known from the original chord progression. This implies that for each step within the stochastic process, the chord being played is always known.

The way the probability matrix is formed is to store it as a 3D matrix. The idea is that for each chord, a probability matrix is created looking at the proportion of transformations between notes for that specific chord. This is done for all 16 chords, and combined into one probability matrix. Hence each chord will have a unique probability matrix that will interpret randomly what the next note will be. This causes the Markov chain to be time-inhomogeneous as the probability matrix changes when the chord changes. This would cause the probability from moving from note i to j potentially different at two different steps, hence there is the possibility that $P(X_{n+1} = j|X_n = i) \neq P(X_1 = j|X_0 = i)$.

The improvisation is then formed by iterating through the chords, simulating lead notes for that specific chord until the next chord is played. Overall this produces an improvisation which has the same chord progression and timing as the original piece, whilst playing notes that fit the chord progression.

5.2 Methods

The initial set up was to find the chords played in the song, and then to create the $\mathbb{P}$-Matrix. This approach is similar to the second process, where a chord space is generated in the same manner. 16 different probability matrices are then formed, one representing each unique chord. These individual probability matrices are then built in the same manner as the first process, where they are a 24 by 24 matrix where each column and row represent the state space S. Then it iterates through the notes, with the transition from note i to j incremented by 1 in the probability matrix for that specific chord. As before, for each unique probability matrix, the rows are normalised if there are data points within them. In general, this forms a 3D ($\mathbb{P}$-Matrix of dimension 24 by 24 by 16 , representing the current note, the next note and the chord played.

The manner in which the improvisation is simulated is by simulating a section for each chord within the chord progression. Initially a random note is chosen to start a random Walk for the first chord. The random walk is of ran for the amount of notes within that chord. The final note played for that chord is then used to start the simulation of the next chord. If there is no possible transition for this note, a random note associated with non-zero columns will be chosen to start the next random walk for that chord. This is repeated for each chord within the chord progression of the

original piece. These notes are then combined in order and converted back to the original values from 54-77. The same timing used within the original piece is also used for the new improvisation. The midi file is then created using this improvisation.

5.3 Resultant Solo

The resultant solo has a relatively similar feel to the first solo, however it is clear that the amount of clashes that occur are vastly reduced - though there are still some 'bad' combinations. This indicates that it combined the best parts/fixes the worst part of the first and second process, matching the flow of the chord progression of the first process with the lack of note clashes in the second process. There are still sparse note clashes within the piece which may be due to the existence of accidental notes within the original piece, which are unable to represented correctly in the computer generated improvisation. However, overall the end product is a significant improvement over the first two processes.

6 Extension

6.1 Introduction

To extend the resultant improvisation created from the third process, a 'human' component is added to the solo. The manufactured solos have sections which don't sound aesthetically pleasing to the hear; various note clashes due to the notes being chosen randomly and the incorrect use of accidentals. 3 main changes were made to the improvisation produced in process 3. The first was to change the velocities (volumes) throughout the piece of music, the second was to remove 'bad' notes that clashed with certain chords and finally, the instrument used to play the improvisation was changed. The aim of adding the human element is to produce music which is more enjoyable to listen to, whilst fixing the sections within the third process improvisation that can be improved

6.2 Method

The first change was to look at changing the velocities (volumes) within the piece of music. The was used to create contrasts within section of the piece, and add depth to the playing. This was done by observing periods of trills, 'licks' and other fast parts, and then applying a velocity curve to the notes to add emphasis. Overall this adds some depth of the improvisation, improving the final end product.
The next part was to remove the 'bad' notes that clashed in the improvisation. The aim of this was to limit the amount of unnecessary clashes within the improvisation to produce a cleaner sounding end product - without removing accidentals entirely. This was achieved by listening for obviously clashing notes, and then shifting them to a suitable note in the scale. There was no exact method to this as it was based on subjective interpretation. Overall, this step created a cleaner, more aesthetically pleasing end product.
Finally, the instrument that played the final improvisation was changed. The default instrument did not cooperate with the changing velocities. First a saxophone was tried - but unfortunately it was not the best quality. We instead used a synthesiser, so that the velocity changes could be observed. This gave the music a different aspect, creating a seemingly smoother end product. The saxophone sound package was also attempted to try and mimic Charlie Parker's original piece, however the end product sounded unrealistic as was discarded.

6.3 Resultant Piece

Overall, the three changes significantly improve the final end product of the improvisation by reducing and eliminating the key weaknesses within the improvisation within the third process. This produces a improvisation which not only sounds good, but successfully mimics the quality of music produced by Charlie Parker.

7 Limitations

One of the biggest limitations that impacts each process is the use of accidentals in the original piece of music. Accidentals are notes which don't appear in-key, and hence appear to slightly clash with the backing chord on purpose. Due to the processes producing the solos by random discrete time Markov chain walks, the accidentals are unable to be produced in the manner of the original song, but will be placed randomly throughout. This causes bad clashes between the notes and the chords within the solo that reduces the listening quality.

In an attempt to minimise the limitation the accidentals have on the improvised solos, the generated outputs have been manually "cleaned" by erasing the notes that seem to lower the listening quality, this method in itself is a limitation as it is based off human opinion which is not consistent among every human.

Another limitation for the second and third process is that not all note-chord pairs 'work'. There is also insufficient data from one piece of music to prove an accurate and realistic $\mathbb{P}$-matrix. Having more data to work with would allow for more potential note-chord pair transitions that produce a superior end product.

A further limitation is that the timing of the notes has been kept the same as the original piece. The objective was to generate a new solo, but this software purely looks to map each note to a new note - irrespective of timing, rests and 'feel'. Whilst the objective was to try to create a solo that mirrors the original piece, the changing of timing can create a uniqueness within each different solo.

8 Conclusion

The ability to produce new improvisations based on a piece within Charlie Parker's Omnibook proved to be a challenging but rewarding experience. Using a variety of Markovian and `MATLAB` techniques, 3 different and unique improvisations of the original work were produced. It was found that the third improvisation which kept the original chord progression as well as the randomly selecting notes which go with the chord playing produced the best overall sound. Finally, the process of adding a 'human' element allowed the flaws within the third improvisation to be cleaned up in order to produce a reasonable improvisation which successfully mimics the style of Charlie Parker.

A Appendices

A.1 Expected Hitting Time

EHT.m - generating the Expected Hitting times

```
%EHT.m should be run after any one of the approach files
%Generates the expected hitting time for the ktosolve'th non-zero entry
%May 2018
%Andrew Martin, James Schoff, Thomas Carey
%will solve for the ktosolve'th NONZERO note
approach2=0;
%This is a changed version of the matrix from the approach used
%removes all zero/nan columns and rows.
absorbprobmatrix = markovMatrix.P;
absorbprobmatrix(~any(absorbprobmatrix,2),:) =[];
absorbprobmatrix(:,~any(absorbprobmatrix,1)) =[];
numNotes = length(absorbprobmatrix);

K =sym ('k',[1,numNotes]);
%the one we want to solve the E(hitting time)for
ktosolve = 1;

absorbprobmatrix(ktosolve,:) =zeros(1,numNotes);
absorbprobmatrix(ktosolve,ktosolve) = 1;
syms eqnarray
for(j = 1:numNotes)
     eqnarray(j) = K(j)- sum(absorbprobmatrix(j,:).*K)  ==1;
end
eqnarray(ktosolve) = 0;
[A,B] = equationsToMatrix(eqnarray,[K(1:ktosolve-1),K(ktosolve+1:end)]);

%linear solve the equations
%since 'k21' == 0 we just subtract it so we can convert to double
expHittingTime = linsolve(A,B) - ('k'+string(ktosolve));
expHittingTime=double(expHittingTime);
expHittingTime= [expHittingTime(1:ktosolve-1) ; 0 ; expHittingTime(ktosolve:length(expHittingTime))];
```

A.2 Expecting Hitting Time for the Second Process

1	0		41	143.8785		81	119.211		121	165.326	
2	52.43374		42	142.8785		82	121.6084		122	165.826	
3	109.2241		43	143.1612		83	132.8471		123	161.826	
4	147.9446		44	143.3269		84	133.8471		124	158.826	
5	123.1717		45	144.0198		85	133.8471		125	160.826	
6	163.3121		46	136.998		86	134.8471		126	159.826	
7	150.7694		47	147.8554		87	152.3244		127	160.826	
8	149.7694		48	141.9511		88	153.3244		128	163.826	
9	150.7694		49	140.9511		89	119.1979		129	164.826	
10	151.2694		50	153.5187		90	117.1979		130	155.3991	
11	149.4318		51	151.0857		91	123.1979		131	156.3991	
12	146.5131		52	153.0857		92	118.1979		132	159.6126	
13	147.5131		53	130.3309		93	118.1979		133	158.6126	
14	149.9116		54	131.3336		94	119.1979		134	160.6126	
15	149.3244		55	132.3336		95	122.1979		135	161.6126	
16	150.3101		56	133.3336		96	123.1979		136	157.826	
17	151.5187		57	124.3215		97	175.3548		137	158.826	
18	143.5994		58	128.3255		98	174.3548		138	159.6126	
19	145.9843		59	129.3255		99	173.3548		139	154.3991	
20	144.9843		60	146.0857		100	173.8548		140	152.7325	
21	144.1477		61	147.0857		101	174.3548		141	154.0658	
22	144.2551		62	113.3093		102	174.8548		142	151.7325	
23	146.9446		63	120.3147		103	175.3548		143	149.253	
24	145.5908		64	130.3363		104	168.3548		144	150.253	
25	146.5908		65	131.3363		105	166.8548		145	148.8787	
26	145.9588		66	132.3363		106	165.8548		146	113.2241	
27	146.5908		67	56.43374		107	169.3548		147	110.2241	
28	144.5044		68	55.43374		108	166.5119		148	111.2241	
29	146.3505		69	54.43374		109	170.3548		149	112.2241	
30	146.3741		70	57.43374		110	166.8548		150	145.2551	
31	147.149		71	58.43374		111	165.5119		151	146.2551	
32	151.8554		72	153.3244		112	167.8548		152	147.2551	
33	147.3741		73	154.3244		113	162.1977				
34	150.8554		74	155.3244		114	163.1977				
35	144.1612		75	152.3244		115	162.1977				
36	142.5994		76	153.3244		116	161.1977				
37	137.998		77	120.1979		117	161.826				
38	141.1361		78	123.1979		118	164.5119				
39	140.1361		79	124.1979		119	164.826				
40	137.8571		80	123.4032		120	164.326				