Using Discovered, Polyphonic Patterns to Filter Computer-generated Music

Tom Collins, Robin Laney, Alistair Willis, and Paul Garthwaite

The Open University, Walton Hall, Milton Keynes, MK7 6AA, UK
t.e.collins@open.ac.uk

Abstract. A metric for evaluating the creativity of a music-generating system is presented, the objective being to generate mazurka-style music that inherits salient patterns from an original excerpt by Frédéric Chopin. The metric acts as a filter within our overall system, causing rejection of generated passages that do not inherit salient patterns, until a generated passage survives. Over fifty iterations, the mean number of generations required until survival was 12.7, with standard deviation 13.2. In the interests of clarity and replicability, the system is described with reference to specific excerpts of music. Four concepts—Markov modelling for generation, pattern discovery, pattern quantification, and statistical testing—are presented quite distinctly, so that the reader might adopt (or ignore) each concept as they wish.

1 Aim and Motivation

A stylistic composition (or pastiche) is a work similar in style to that of *another* composer or period. Examples exist in 'classical' music (Sergey Prokofiev's Symphony No. 1 is in the style of Joseph Haydn) as well as in music for film and television, and in educational establishments, where stylistic composition is taught 'as a means of furthering students' historical and analytical understanding' (Cochrane 2009). *If* a computational system produces successful stylistic compositions ('successful' in the sense of the 'indistinguishability test' of Pearce and Wiggins (2001) for instance), then it is capable of a task that, in the human sphere, is labelled *creative*. The creativity metric presented below is intended as preliminary to (not a replacement of) an 'indistinguishability test'.

This paper relates ongoing research on a computational system with the aim of modelling a musical style.[1] The motivation for the system is as follows. Cope (2005, pp. 87-95) describes a data-driven model that can be used to generate passages of music in the style of Johann Sebastian Bach's chorale harmonisations. His model can be cast as a first-order Markov chain and we have replicated this aspect of his model, with some modifications (see Sect. 2 for details). Our method is applied to a database consisting of Frédéric Chopin's mazurkas. This choice

[1] See Pearce et al. (2002) on how *computational modelling of musical styles* constitutes one motivation for automating the compositional process.

of database is refreshing (Bach chorales have become the standard choice) and explores the range of music in which Markov chain models can be applied.

A passage generated by a first-order Markov model 'often wanders with uncharacteristic phrase lengths and with no real musical logic existing beyond the beat-to-beat syntax' (Cope 2005, p. 91) and Cope discusses strategies for addressing this problem. One such strategy—of incorporating musical 'allusions' into generated passages—has been criticised for having an implementation that does not use robust, efficient algorithms from the literature (Wiggins 2008, p. 112-113).

Our system is motivated by a desire to investigate the above 'allusion' strategy, armed with more robust algorithms (or their underlying concepts), and is illustrated in Fig. 1. Subsequent sections describe various parts of this schematic, as indicated by the dotted boxes, but an overview here may be helpful. By an 'allusion' we mean that an excerpt is chosen from one of 49 Chopin mazurkas (bottom left of Fig. 1), with the objective of generating mazurka-style music *that inherits salient patterns from the chosen excerpt.*[2] To this end salient patterns are handpicked from the chosen excerpt, using the concept of *maximal translatable pattern* (Meredith et al. 2002). This is the meaning of the box labelled 'pattern discovery' in Fig. 1. The discovered patterns are then stored as a 'template'. Meanwhile the dotted box for Sect. 2 in Fig. 1 indicates that a passage (of approximately the same length as the chosen excerpt) can be generated on demand. Illustrated by the diamond box in Fig. 1, the same type of patterns that were *discovered* and stored as a template are now *sought* algorithmically in the computer-generated passage. For a computer-generated passage to survive the filtering process, we ask that it exhibits the same type of patterns as were discovered in the chosen excerpt, occurring the same number of times and in similar locations relative to the duration of the passage as a whole. In Sect. 4 the concept of *ontime percentage* and the Wilcoxon two-sample test are employed to quantify and compare instances of the same type of pattern occurring in different passages of music.

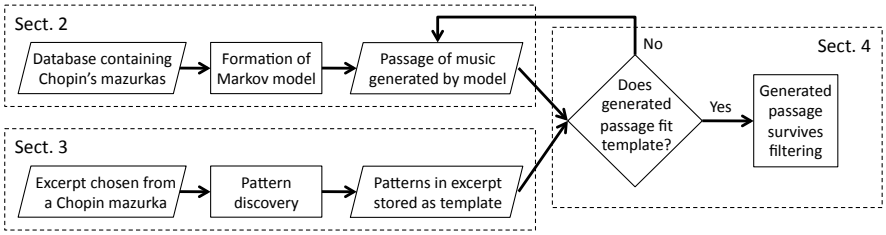

Fig. 1. A schematic of the system to be described.

[2] We would like to thank Craig Stuart Sapp for creating these kern scores and MIDI files, hosted at http://kern.humdrum.net. The database used in this paper consists of opuses 6, 7, 17, 24, 30, 33, 41, 50, 56, 59, 63, 67, and 68.

2 Generation of Computer Music Using a Markov Model

Since initial attempts to marry music theory and Markov chains (Hiller and Isaacson 1959), their application to music analysis and composition has received considerable attention (see Loy 2005 for an accessible introduction and Norris 1997 for the supporting mathematics). The generation of a so-called 'Markovian composition' is outlined now. Suppose that we join the Markovian composition partway through, and that the chord D4-C6 indicated by an arrow in bar 2 of Fig. 2 has just been composed, but nothing further ('middle C' is taken to be C4, and look out for clef changes). The 49 mazurkas by Chopin having been analysed, it is known, say, that in this corpus Chopin uses the D4-C6 chord (or any of its transpositions) on 24 occasions. The computer generates a random number uniformly between 1 and 24, say 17. The 17th instance of this chord (actually one of its transpositions, A2-G4) occurs in bar 38 of the Mazurka in B Major, op. 41/3. The chord that follows *in this mazurka*, A2-D♯4, is transposed appropriately (to D4-G♯5) and becomes the next chord of the Markovian composition. To continue composing, one can repeat the last few sentences with D4-G♯5 as the current chord. The passages shown in Figs. 2–3 were generated in this way.

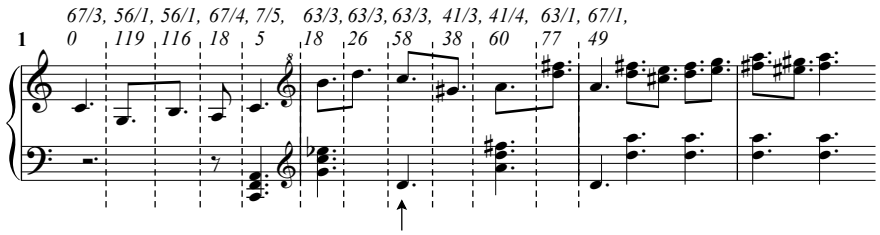

Fig. 2. A passage from the computer-music generator that does *not* survive filtering (contrast with Fig. 3). The italicised numbers refer to opus and bar numbers of particular fragments. For instance '67/3, 0' means opus 67, number 3, bar 0 (anacrusis). The durations in this system are dotted to improve the legibility of the first bar.

The above explanation is simplified slightly, for what happens if one or more already-articulated notes are held while others begin? This is discussed with reference to the chord C3-F♯4-D5 indicated by an arrow in bar 6 of Fig. 3. In this chord, the bottom C3 is held over to the next chord, C3-E♯4-C♯5. We observe that a note in a given chord can be held in one of four ways:

1. Not held beyond the given chord
2. Held over to the next chord
3. Held from the previous chord
4. Held both from the previous chord and to the next chord

Thus in our model, the chord C3-F♯4-D5 indicated by an arrow in Fig. 3 would be represented by the pair of vectors $(18, 8)$, $(2, 1, 1)$. The first vector is the chord

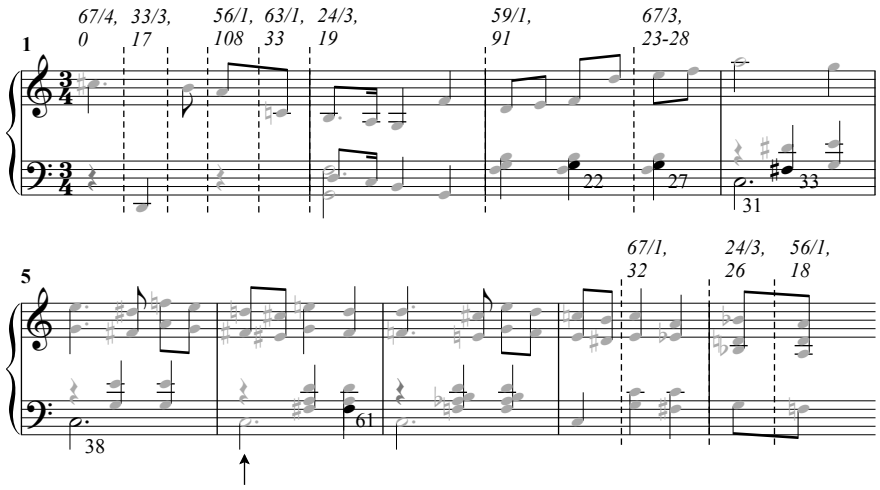

Fig. 3. A passage from the computer-music generator that *does* survive filtering. The darker noteheads in the left hand are referred to as pattern P^*, and indexed (in order of increasing ontime and pitch height) to help with understanding Table 1 (p. 8).

spacing (18 semitones from C3 to F♯4, and 8 from F♯4 to D5) and the second vector encodes how each of the three notes are held (or not), according to the list just given. How is an *actual* chord of certain duration produced in our model, if it only uses chord spacing and holding information? Above, an example was given in which the 17th instance of a chord spacing was chosen from 24 options. By retaining the bar and opus numbers of the original mazurka to which this choice corresponds, we are able to utilise any contextual information (that is not already implied by the chord spacing and holding information) to produce an actual chord of certain duration. This Markov model alone is no match for the complex human compositional process, but it *is* capable of generating a large amount of material that can be analysed for the presence of certain patterns.

3 Discovering Patterns in Polyphonic Music

Meredith et al. (2002) give an elegant method for intra-opus analysis of polyphonic music. In 'intra-opus analysis' (Conklin and Bergeron 2008, p. 67), a single piece of music or excerpt thereof is analysed with the aim of discovering instances of self-reference. The human music analyst performs this task almost as a prerequisite, for it is arguable that music 'becomes intelligible to a great extent through self-reference' (Cambouropoulos 2006, p. 249). Listening to the passage in Fig. 4, for instance, the human music analyst would notice:

1. The repetition of bars 11-12 at bars 13-14
2. The repetition of the rhythms in bar 12 at bar 14 (implied by the above remark), at bar 15, and again at bar 16 (here except the offbeat minim B4)

No doubt there are other matters of musical interest (the tonicization of the dominant at bar 16, the crossing of hands in bars 17-18), as well as 'lesser' instances of self-reference, but here attention is restricted to remarks 1 and 2. It should be emphasised that these remarks are 'human discoveries'; not from an algorithm. However, the key concepts of Meredith et al. (2002) can be used to automate the discovery of these *types of pattern*, so that they *can* be sought *algorithmically* in a computer-generated passage.[3]

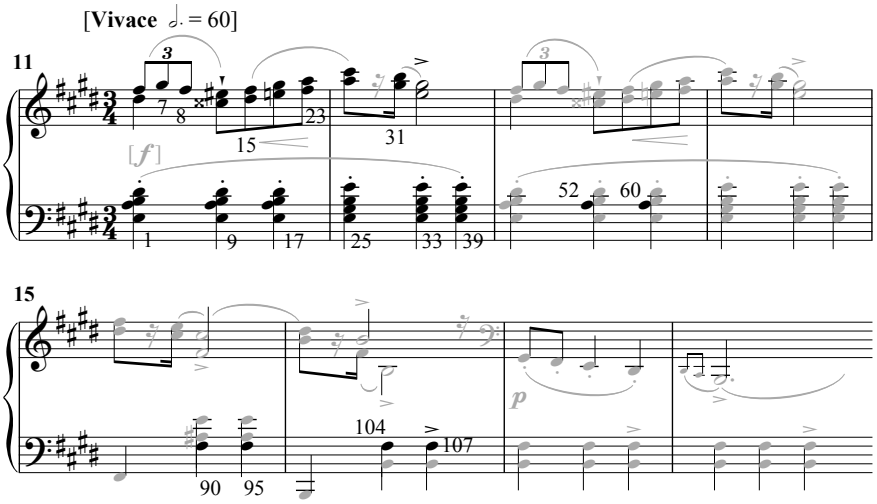

Fig. 4. Bars 11-18 of the Mazurka in E Major, op. 6/3 by Frédéric Chopin. As in Fig. 3, some noteheads are darker and indexed to help with understanding Table 1 (p. 8).

The formal representation of *music as points in multidimensional space* can be traced back at least as far as Lewin (1987). Each note in Fig. 4 can be represented as a point in multidimensional space, a 'datapoint' $\boldsymbol{d} = (x, y, z)$, consisting of an ontime x, a MIDI note number y and a duration z (a crotchet is set equal to 1). The set of all datapoints for the passage in Fig. 4 is denoted D, for 'dataset'. For any given vector $\boldsymbol{v}$, the *maximal translatable pattern* (MTP) of the vector $\boldsymbol{v}$ in a dataset D is defined by Meredith et al. (2002) as the set of all datapoints in the dataset that, when translated by $\boldsymbol{v}$, arrive at a coordinate corresponding to another datapoint in D.

$$MTP(\boldsymbol{v}, D) = \{\boldsymbol{d} \in D \mid \boldsymbol{d} + \boldsymbol{v} \in D\}. \tag{1}$$

For instance,

$$P = MTP(\boldsymbol{w}, D), \quad \text{where } \boldsymbol{w} = (6, 0, 0), \tag{2}$$

is indicated by the darker noteheads in Fig. 4. The vector $\boldsymbol{w} = (6, 0, 0)$ identifies notes that recur after 6 crotchet beats, transposed 0 semitones and with

[3] Further automation of this part of the system is a future aim.

unaltered durations (due to the final 0 in $\boldsymbol{w}$). This is *closely related* to remark 1 (on p. 4), which observes the repetition of bars 11-12 at 13-14, that is after $13 - 11 = 2$ bars (or 6 crotchet beats). It can be seen from Fig. 4, however, that P contains two notes each in bars 13, 15 and 16, which are repeated in bars 15, 17, 18 respectively. This is *less closely related* to remark 1, showing that the human and computational analytic results do not align exactly. One highlights an idiosyncrasy—perhaps even a shortcoming—of the other, depending on your point of view.

As well as the definition of a maximal translatable pattern, the other key concept in Meredith et al. (2002) is the *translational equivalence class* (TEC). Musically, the translational equivalence class of a pattern consists of the pattern itself and all other instances of the pattern occurring in the passage. Mathematically, the translational equivalence class of a pattern P in a dataset D is

$$TEC(P, D) = \{Q \subseteq D \mid P \equiv_\tau Q\}, \tag{3}$$

where $P \equiv_\tau Q$ means that P and Q contain the same number of datapoints and there exists *one* vector $\boldsymbol{u}$ that translates each point in P to a point in Q. We return to the specific example of the dataset D containing the datapoints for the passage in Fig. 4, and suppose P is defined by (2). It can be verified that the translational equivalence class of P in D is

$$TEC(P, D) = \{P,\ \tau(P, \boldsymbol{w})\}, \tag{4}$$

where $\tau(P, \boldsymbol{w})$ denotes the set of all vectors $\boldsymbol{p} + \boldsymbol{w}$, and $\boldsymbol{p}$ is a datapoint in P. Equation (2) helps to identify notes whose durations *and* MIDI note numbers recur after 6 crotchet beats. The set in (4) contains the pattern P and $\tau(P, \boldsymbol{w})$, the only other instance of the pattern in the excerpt. Together, the equations suggest how to automate discovery of the type of pattern described in remark 1.

What of remark 2, the repetition of the rhythms in bar 12 at bar 14 (after 6 beats), bar 15 (after 9 beats) and bar 16 (after 12 beats)? As this is a rhythmic pattern, it is useful to work with a 'rhythmic projection' D' of the dataset D. If $\boldsymbol{d} = (x, y, z)$ is a member of D then $\boldsymbol{d'} = (x, z)$, consisting of an ontime and duration, is a member of the projected dataset D'. It should be noted that two distinct datapoints $\boldsymbol{d}, \boldsymbol{e} \in D$ can have a conicident projection, that is $\boldsymbol{d'} = \boldsymbol{e'}$, just as two objects placed side by side might cast coincident shadows. The repetition observed in remark 2 occurs at 6 *and* 9 *and* 12 beats after the original, so let

$$S = MTP(\boldsymbol{u'}, D') \cap MTP(\boldsymbol{v'}, D') \cap MTP(\boldsymbol{w'}, D'), \tag{5}$$

where $\boldsymbol{u'} = (6, 0)$, $\boldsymbol{v'} = (9, 0)$, and $\boldsymbol{w'} = (12, 0)$. The set $MTP(\boldsymbol{u'}, D')$ in (5) corresponds to notes whose durations recur after 6 crotchet beats. The second set $MTP(\boldsymbol{v'}, D')$ corresponds to notes whose durations recur after 9 beats, and the third set $MTP(\boldsymbol{w'}, D')$ to notes whose durations recur after 12 beats. Taking their intersection enables the identification of notes whose durations recur after 6, 9 and 12 beats, which is closely related to remark 2. It can be verified that

$$TEC(S, D') = \{S,\ \tau(S, \boldsymbol{u'}),\ \tau(S, \boldsymbol{v'}),\ \tau(S, \boldsymbol{w'})\}. \tag{6}$$

As with pattern P, the human and computational analytic results for pattern S do not align exactly. All of the notes in bar 12 of Fig. 4 are identified as belonging to pattern S, but so are a considerable number of left-hand notes from surrounding bars. While it is not the purpose of this paper to give a fully-fledged critique of Meredith et al. (2002), Sect. 3 indicates the current state of progress toward a satisfactory pattern discovery algorithm for intra-opus analysis.

4 Filtering Process

4.1 Quantifying an Instance of a Musical Pattern

When an instance of an arbitrary pattern P has been discovered within some dataset D, as in the previous section, how can the position of the pattern be quantified, relative to the duration of the excerpt as a whole? Here the straightforward concept of *ontime percentage* is used. For a note having ontime t, appearing in an excerpt with total duration T, the note has *ontime percentage* $100t/T$. For instance, the excerpt in Fig. 4 has total duration 24 (= 8 bars × 3 beats). Therefore, taking the F♯ at the top of the first chord in bar 13, with ontime 6, this note has ontime percentage $100t/T = 100 \cdot 6/24 \approx 33\%$.

When calculating the ontime percentage of each datapoint $\boldsymbol{p}$ in a pattern P, a decision must be made whether to include repeated values in the output. For instance, the six notes in the first chord in bar 13 will have the same ontime percentage, so repeated ontime percentages indicate a thicker texture. The inclusion of repeated values does not affect the appropriateness of the statistical test described in Sect. 4.2, but it may affect the result: two otherwise similar lists of ontime percentages might be distinguishable statistically due to a high proportion of repeated values in one collection but not the other. Here the decision is taken *not* to include repeated values. Two lists of ontime percentages are shown in columns 2 and 5 of Table 1 (overleaf). The bottom half of column 5 is derived from the darker notes in Fig. 3, referred to as pattern P^*. Column 2 and the top half of column 5 are derived from the darker notes in Fig. 4, pattern P.

4.2 Applying Wilcoxon's Two-sample Test in Musical Scenarios

Let us suppose we have two random samples, one consisting of m observations $x_1, x_2, \ldots, x_m$, and the other consisting of n observations $y_1, y_2, \ldots, y_n$. Columns 2 and 5 of Table 1 serve as an example, with $m = 17$ and $n = 6$. It should be pointed out that the random-sample supposition *almost never* applies in musical scenarios. Increasingly however, the assumption is being made in order to utilise definitions such as the *likelihood* of seeing a pattern (Conklin and Bergeron 2008). Wilcoxon's two-sample test helps to determine *whether two sets of observations have the same underlying distribution.* The calculation of the test statistic will be demonstrated, with the theoretical details available elsewhere (Neave and Worthington 1988). The test statistic W is calculated by assigning ranks $R_1, R_2, \ldots, R_n$ to the set of observations, $y_1, y_2, \ldots, y_n$, as though they

Table 1. The note indices, ontime percentages and combined sample ranks of two patterns are shown, P indicated by the darker noteheads in Fig. 4, and P^* from Fig. 3.

Pattern P			Pattern P continued		
Note index	Ontime %	Rank	Note index	Ontime %	Rank
1	0.0	1	90	54.2	19
7	1.4	2	95	58.3	20
8	2.8	3	104	66.7	21
9	4.2	4	107	70.8	23
15	6.3	5			
17	8.3	6	Pattern P^*		
23	10.4	7	22	28.0	12
25	12.5	8	27	32.0	14
31	15.6	9	31	36.0	16
33	16.7	10	33	40.0	17
39	20.8	11	38	48.0	18
52	29.2	13	61	68.0	22
60	33.3	15	P^* **rank total**:		99

appear in a combined sample with the other set. This has been done in column 6 of Table 1 (see also column 3). Then $W = \sum_{i=1}^{n} R_i$ is a random variable, and from Table 1, a value of $w = 99$ has been observed. Either the exact distribution of W or, for large sample sizes, a normal approximation can be used to calculate $\mathbb{P}(W \leq w)$. Using a significance threshold of $\alpha = 0.05$ and with $m = 17, n = 6$, a value of W outside of the interval $[43, 101]$ needs to be observed in order to reject a null hypothesis that the two sets of observations have the same underlying distribution. As we have observed $w = 99$, the null hypothesis *cannot* be rejected.

What does the above result mean in the context of musical patterns? We have taken P and P^*, two instances of the same type of pattern occurring in different passages of music, and compared their ontime percentages. *Not* being able to reject the null hypothesis of 'same underlying distribution' is taken to mean that a computer-generated passage *survives* this element of the filtering process. We are notionally content that the relative positions of P and P^* are not too dissimilar. There are five further elements to the filtering process here, with the Wilcoxon two-sample test being applied to the ontime percentages of:

1. $\tau(P, \boldsymbol{w})$ and $\tau(P^*, \boldsymbol{w})$, where $\boldsymbol{w} = (6, 0, 0)$
2. S and S^*, where S is given in (5), and S^* denotes the corresponding pattern for the computer-generated passage in Fig. 3
3. $\tau(S, \boldsymbol{u'})$ and $\tau(S^*, \boldsymbol{u'})$, where $\boldsymbol{u'} = (6, 0)$
4. $\tau(S, \boldsymbol{v'})$ and $\tau(S^*, \boldsymbol{v'})$, where $\boldsymbol{v'} = (9, 0)$
5. $\tau(S, \boldsymbol{w'})$ and $\tau(S^*, \boldsymbol{w'})$, where $\boldsymbol{w'} = (12, 0)$

At a significance threshold of $\alpha = 0.05$, the passage in Fig. 3 survives each element of the filtering process, whereas the passage in Fig. 2 does not.

5 Discussion

This paper has presented a metric for evaluating the creativity of a music-generating system. Until further evaluation has been conducted (by human listeners rather than just by the creativity metric), we are cautious about labelling our overall system as *creative*. The objective in the introduction was to generate mazurka-style music that inherits salient patterns from a chosen excerpt. It is encouraging that Fig. 3—which arguably sounds and looks more like a mazurka than Fig. 2—survives the filtering process, whereas Fig. 2 does not. In terms of meeting the aim of pattern inheritance, there is considerable room for improvement: Figs. 3 and 4 do not sound or look much alike, and a human music analyst would be hard-pressed to show how the filtered output (Fig. 3) inherits *any* salient patterns from the chosen excerpt (Fig. 4). One solution would be to include more filters. Another solution would be to raise the significance threshold, α. By making the null hypothesis of 'same underlying distribution' easier to reject, it becomes harder for a generated passage to survive filtering. Over fifty iterations, the mean number of generations required until survival was 12.7, with standard deviation 13.2. Raising α may increase pattern inheritance, but it may also have a non-linear impact on these statistics.

The verbatim quotation in Fig. 3 (of bars 23-28 from the Mazurka in C Major, op. 67/3) raises several issues that relate to further work. First, we will consider including in the system a mechanism for avoidance of verbatim quotation. Second, the quotation contains a prominent *sequential* pattern that is different in nature to the *intended* inheritance (the two patterns observed in remarks 1 and 2 on p. 4). Using the concept of *morphetic pitch* defined in Meredith et al. (2002) it is possible to identify such sequential patterns, so the sequence itself is not a problem, only that its presence was unintended. Measures exist for the *prominence* (Cambouropoulos 2006) or *interest* (Conklin and Bergeron 2008) of a pattern relative to others in a passage of music. The adaptation of these measures to polyphonic music would constitute a worthwhile addition, both to Meredith et al. (2002) and to the use of discovered, polyphonic patterns in filtering computer-generated music.

We have more general concerns about the extent to which the first-order Markov model generalises from Bach chorales to Chopin mazurkas. From a musical point of view the mazurkas may be too rich. The verbatim quotation mentioned above is indicative of a *sparse* transition matrix, which might be made more dense by including more mazurkas or other suitable compositions. There are several ways in which the system described could be fine-tuned. First, computational time could be saved by filtering incrementally, discarding generated passages *before* they reach the prescribed length if for some reason they are already bound not to survive filtering. Second, both Lewin (1987) and Meredith et al. (2002) propose (differing) methods for ordering notes. These could be used instead of or as well as ontime percentages, to investigate the effect on the output of the system described. Third, if a region of original music is spanned entirely by a pattern (so that there are no non-pattern notes in this region) and this is also true of its recurrence(s), then this ought to be stored in the template (see

the definition of *compactness* in Meredith 2006). Again this would save computational time that is currently wasted in our system. Finally, sometimes the occurrence of a certain type of pattern *implies* the occurrence of another type of pattern. For example, bars 11-12 of Fig. 4 (approximately pattern P) recur at bars 13-14, *implying* that the *rhythms* of bar 12 (approximately pattern S) will recur in bar 14. This may seem obvious for only two discovered patterns, P and S, but when more patterns are discovered, the way in which these might be arranged into a hierarchy is worthy of further investigation.

6 Acknowledgements

This paper benefited from a helpful discussion with David Meredith. We would also like to thank the three anonymous reviewers for their comments.

References

Cambouropoulos, E.: Musical parallelism and melodic segmentation: a computational approach. *Music Perception* 23(3), 249-267 (2006)

Cochrane, L.: "pastiche." A. Latham (ed.). *The Oxford Companion to Music. Oxford Music Online*, http://www.oxfordmusiconline.com (accessed 20 September, 2009)

Conklin, D., and Bergeron, M.: Feature set patterns in music. *Computer Music Journal* 32(1), 60-70 (2008)

Cope, D.: *Computational models of musical creativity.* Cambridge, Massachusetts: MIT Press (2005)

Hiller, L., and Isaacson L.: *Experimental music.* New York: McGraw-Hill (1959)

Lewin, D.: *Generalized interval systems and transformations.* New Haven, Connecticut: Yale University Press (1987)

Loy, G.: *Musimathics: the mathematical foundations of music*, vol. 1. Cambridge, Massachusetts: MIT Press (2005)

Meredith, D.: Point-set algorithms for pattern discovery and pattern matching in music. T. Crawford and R.C. Veltkamp (eds.). *Proceedings of the Dagstuhl Seminar on Content-Based Retrieval.* Dagstuhl, Germany (2006)

Meredith, D., Lemström, K., and Wiggins, G.A.: Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research* 31(4), 321-345 (2002)

Neave, H.R., and Worthington, P.L.: *Distribution-free tests.* London: Unwin Hyman (1988)

Norris, J.R.: *Markov chains.* Cambridge: Cambridge University Press (1997)

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

Pearce, M.T., and Wiggins, G.A.: Towards a framework for the evaluation of machine compositions. *Proceedings of the AISB Symposium on Artificial Intelligence and Creativity in Arts and Sciences* (2001)

Wiggins, G.A.: Computer models of musical creativity: a review of computer models of musical creativity by David Cope. *Literary and Linguistic Computing* 23(1), 109-115 (2008)