COMPARING PITCH SPELLING ALGORITHMS

David Meredith **Geraint A. Wiggins**
Centre for Cognition, Computation and Culture
Department of Computing
Goldsmiths' College, University of London
New Cross, London, SE14 6NW.
dave@titanmusic.com, g.wiggins@gold.ac.uk

ABSTRACT

A pitch spelling algorithm predicts the pitch names of the notes in a musical passage when given the onset-time, MIDI note number and possibly the duration and voice of each note. Various versions of the algorithms of Longuet-Higgins, Cambouropoulos, Temperley and Sleator, Chew and Chen, and Meredith were run on a corpus containing 195972 notes, equally divided between eight classical and baroque composers. The standard deviation of the accuracies achieved by each algorithm over the eight composers was used as a measure of its style dependence (SD). Meredith's *ps1303* was the most accurate algorithm, spelling 99.43% of the notes correctly ($SD = 0.54$). The best version of Chew and Chen's algorithm was the least dependent on style ($SD = 0.35$) and spelt 99.15% of the notes correctly. A new version of Cambouropoulos's algorithm, combining features of all three versions described by Cambouropoulos himself, also spelt 99.15% of the notes correctly ($SD = 0.47$). The best version of Temperley and Sleator's algorithm spelt 97.79% of the notes correctly, but nearly 70% of its errors were due to a single sudden enharmonic change. Longuet-Higgins's algorithm spelt 98.21% of the notes correctly ($SD = 1.79$) but only when it processed the music a voice at a time.

Keywords: pitch spelling, transcription, algorithms, evaluation.

1 INTRODUCTION

A *pitch spelling algorithm* is an algorithm that attempts to compute the correct pitch names (e.g., C♯4, B♭5 etc.) of the notes in a passage of tonal music, when given only the onset-time, MIDI note number and possibly the duration and voice of each note.

There are good practical reasons for attempting to develop a reliable pitch spelling algorithm. First, until such an algorithm is devised, it will be impossible to construct a reliable *MIDI-to-notation transcription system*—that is, a system that reliably computes a correctly notated score of a passage when given only a MIDI file of the passage as input. Second, existing audio transcription systems generate not notated scores but MIDI-like, 'piano roll' representations as output (Abdallah and Plumbley, 2004). So, if one needs to produce a notated score from a digital audio recording, one needs not only an audio transcription system but also a MIDI-to-notation transcription algorithm (incorporating a pitch spelling algorithm).

Third, knowing the letter-names of the pitch events in a passage is useful in music information retrieval and musical pattern discovery (see, for example, Meredith et al., 2002, pp. 328–330). In particular, two occurrences of a motif on different degrees of a scale might be perceived to be similar even if the corresponding chromatic intervals in the patterns differ. Figure 1, for example, shows an instance of tonal melodic sequence in which the three patterns A, B and C are perceived to be three occurrences of the same motif even though the corresponding chromatic intervals are different in the three patterns. Note that, in this example, one important aspect of the perceived similarity between patterns A, B and C is nicely represented in the notation by the fact that they all have the same scale-step interval structure (i.e., a descending step followed by two ascending steps). In other words, one result of the choice of pitch names for the notes in this passage is that the scale-step interval structures are the same for these three perceptually similar but chromatically different pat-

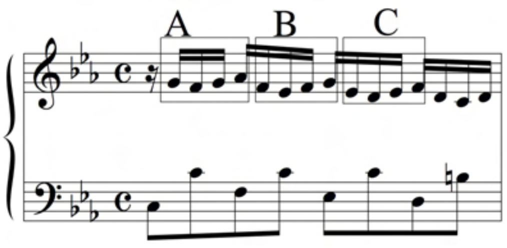

Figure 1: Three perceptually similar patterns with different chromatic pitch interval structures (from the first bar of the Prelude in C minor (BWV 871/1) from Book 2 of J. S. Bach's *Das Wohltemperirte Clavier*).

terns.

If the pitch names of the notes are encoded, matches such as the ones in Figure 1 can be found using fast exact-matching algorithms (e.g., Knuth et al., 1977; Galil, 1979). However, if just MIDI note numbers are used, matches such as the ones in Figure 1 can only be found using slower and more error-prone approximate-matching algorithms (e.g., Cambouropoulos et al., 2002).

In the study reported here, pitch spelling algorithms proposed by several authors were analysed, evaluated and, in some cases, improved. The algorithms studied were those of Longuet-Higgins (1987), Cambouropoulos (1996, 2001, 2003), Temperley (2001), Chew and Chen (2003, 2005) and Meredith (2003, 2005). A number of different versions of each of these algorithms were run on a test corpus containing 195972 notes, equally divided between eight classical and baroque composers.

Section 2 below is a discussion of the methodology used in this study to evaluate the algorithms. The various versions of the algorithms that were tested are then briefly described in section 3. The results obtained are summarised and discussed in section 4. Finally, in section 5 we present the main conclusions that can be drawn from this study and suggest ways in which the research reported here could be continued.

2 METHODOLOGY

When comparing algorithms, one must first identify relevant *evaluation criteria*—that is, specific ways in which the performance of one algorithm might be considered interestingly different from that of another. Then appropriate *performance metrics* have to be defined for these evaluation criteria. A performance metric for a particular evaluation criterion is a way of measuring the performance of an algorithm with respect to that criterion. When comparing pitch spelling algorithms, these performance metrics are used to measure how well an algorithm performs on some specified test corpus of works.

In this paper, we use two principal evaluation criteria: *spelling accuracy*, that is, how well an algorithm predicts the pitch names of the notes; and *style dependence*, that is, how much the spelling accuracy of an algorithm is affected by the style of the music being processed. The performance metric used here to measure spelling accuracy is *note accuracy*: the note accuracy of an algorithm A over a set of movements S is defined to be the proportion of notes in S spelt correctly by A. In the study reported here, the note accuracies were measured over the complete test corpus and over each of the eight subsets of this corpus containing the works by a particular composer. The standard deviation of the note accuracies over these eight composer subsets was used as a measure of style dependence (SD).

A test corpus should, ideally, be a large, representative sample of the population of works that the algorithms will be used to process in the future. Unfortunately, most of the test corpora used in previous publications for evaluating pitch spelling algorithms have not been properly representative of any wider population of works. In some cases, this has been because the test corpus was either far too small (Longuet-Higgins, 1987; Cambouropoulos, 1996; Chew and Chen, 2003, 2005) or too small to represent the target population of works (Stoddard et al., 2004). In other cases, the test corpus has consisted of movements that are too closely related in terms of genre, composer and instrumentation (Meredith, 2003; Cambouropoulos, 2001, 2003). In yet other cases, the test corpora consisted of small fragments from works that were either incipits and main themes (Cambouropoulos, 1996) or extracts specially chosen to illustrate particular music-theoretical phenomena (Temperley, 2001). Such corpora cannot be considered representative of some interesting wider population of complete musical works. The test corpus used by Meredith (2005) was relatively large (1729886 notes), but it was very unevenly divided between the nine composers represented, ranging from 2962 notes from works by B. Marcello to 627083 notes from works by J. S. Bach. This meant that the results obtained for the different composers were not comparable and thus not appropriate for measuring style dependence.

In the study described here, the test corpus contained 195972 notes, consisting of 216 movements from works by eight baroque and classical composers (Corelli, Vivaldi, Telemann, J. S. Bach, Handel, Haydn, Mozart and Beethoven). This corpus was chosen so that it contained almost exactly 24500 notes for each of the eight composers represented.

Several algorithms tested here achieved note accuracies higher than 99% which prompts one to question whether the differences between these values are statistically significant. To date, only Meredith (2005) has attempted to measure the statistical significance of the difference between the spelling accuracies achieved by pitch spelling algorithms. However, he used McNemar's test for this purpose (McNemar, 1969, pp. 54–8), and this test is not strictly appropriate in this situation because its validity depends on the correctness of any particular pitch name being independent of the correctness of the pitch names assigned to the notes around it—which is usually not the case since pitch spelling algorithms typically use the context surrounding a note to determine its pitch name. In fact, it seems that there is no straightforward statistical method that is entirely appropriate for measuring the significance of the difference between two spelling accuracies. In order to avoid the possibility of giving misleading estimates of significance, we have therefore decided not to provide such estimates in this paper.

Occasionally in tonal music, a modulation occurs that results in a passage being in an extremely flat or extremely sharp key that requires many double flats or double sharps. Composers often choose to notate such passages in enharmonically equivalent keys that require fewer accidentals because this usually makes the music easier to read. When this happens, a sudden enharmonic change occurs in the score in which the notated key suddenly changes to a very distant key even though no such modulation is heard by the listener (Temperley, 2001, p. 135). The test corpus used here contained just one example of such a sudden enharmonic change. This occurs at bar 166 in the fourth movement of Haydn's Symphony No. 100 in G major ('Military') (Hob. I:100) where the notated key sud-

Pitch name class	···	F♭	C♭	G♭	D♭	A♭	E♭	B♭	F	C	G	D	A	E	B	F♯	C♯	G♯	D♯	A♯	E♯	B♯	···
Sharpness	···	−8	−7	−6	−5	−4	−3	−2	−1	0	1	2	3	4	5	6	7	8	9	10	11	12	···

Figure 2: The ‘line of fifths’ showing the sharpness associated with each pitch name class.

denly changes from D♭ major to C♯ major, even though no change in key is perceived by the listener. As no modulation is perceived by the listener, it can be argued that spelling this movement without the enharmonic change (i.e., staying in D♭ major) would also be correct. It was therefore decided that the output of each algorithm for this movement should be compared with two “correct” spellings: one in which the notes are spelt as they are in the original score; and a second, modified version, in which the enharmonic change is omitted. When an algorithm performed better on the modified version than on the original, an alternative value for its note accuracy will be given in the results.

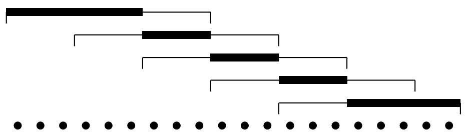
Figure 3: Cambouropoulos’s own caption to this figure reads: “Shifting overlapping window technique. For each window only the middle section of spelled pitches (bold line) is retained. Dots represent the pitches of the input sequence.” (Reproduced (with minor corrections and alterations) from Cambouropoulos, 2003, p. 420, Fig. 6 and Cambouropoulos, 1996, p. 245, Fig. 8.)

3 THE ALGORITHMS TESTED

The algorithms considered in this study were those of Longuet-Higgins (1987), Cambouropoulos (1996, 2001, 2003), Temperley (2001), Chew and Chen (2003, 2005) and Meredith (2003, 2005). A number of different versions of each of these algorithms were run on the test corpus. These algorithms will now be briefly described.

3.1 Longuet-Higgins’s pitch spelling algorithm

Pitch spelling is one of the tasks performed by Longuet-Higgins’s (1987) `music.p` program, which was designed to be used only on monophonic melodies (Longuet-Higgins, 1987, p. 114). Given the MIDI note number and onset time of each note, Longuet-Higgins’s algorithm estimates a value of *sharpness*, q, which is an integer indicating the position of the pitch name of the note on the line of fifths (Temperley, 2001, p. 117) (see Figure 2). The pitch name of each note can then be computed from its MIDI note number and its sharpness. Longuet-Higgins’s algorithm is based on a “theory of tonality” (Longuet-Higgins, 1987, p. 115) which consists of six rules. The first of these rules ensures that each note is spelt so that it is as close as possible to the local tonic on the line of fifths (Longuet-Higgins, 1987, pp. 112–113). The other rules control the way in which the algorithm deals with chromatic intervals and modulations. In particular, the second rule states that if the current key implies two consecutive chromatic intervals (i.e., intervals spanning more than 6 steps on the line of fifths), then the key should be changed so that both intervals become diatonic (i.e., they span less than 6 steps on the line of fifths) (Longuet-Higgins, 1987, p. 113). The algorithm is also restricted to assigning pitch names between G♭♭ and A𝄪 on the line of fifths.

In the version of the algorithm implemented in `music.p` the second of the six rules in Longuet-Higgins’s theory is not implemented correctly. Specifically, this rule implies that a subdominant preceded and followed by a sharpened subdominant ($\sharp\hat{4} - \hat{4} - \sharp\hat{4}$) should trigger a modulation, as should the sequence $\flat\hat{2} - \hat{2} - \flat\hat{2}$. However, neither of these sequences actually triggers a modulation in Longuet-Higgins’s implementation. A new version of the algorithm was therefore constructed in which the implementation of this second rule was corrected. Both the original and corrected versions were also further modified to produce other versions that were not restricted to assigning pitch names within any particular range on the line of fifths.

Each of these versions of Longuet-Higgins’s algorithm was run on two different versions of the test corpus: one in which the notes within each movement were sorted so that the voices were arranged ‘end-to-end’; and a second version in which the notes were sorted by onset time and MIDI note number (with preference given to onset time). Longuet-Higgins’s insistence that his program should only be used on monophonic music led us to expect his algorithm to perform better on the version of the dataset in which the voices were arranged ‘end-to-end’.

3.2 Cambouropoulos’s pitch spelling algorithms

Cambouropoulos (1996, 2001, 2003) has published descriptions of three versions of his pitch spelling algorithm. In all three, it is assumed that the passage of music to be processed has been represented as a sequence of MIDI note numbers in which the notes are in the same order as in the music. This sequence of MIDI note numbers is then processed using a “shifting overlapping windowing technique” (Cambouropoulos, 2003, p. 420), in which each window contains a certain number of contiguous elements in the input sequence (see Figure 3).

Cambouropoulos allows ‘white note’ pitch classes (i.e., 0, 2, 4, 5, 7, 9 and 11) to be spelt in three different ways (e.g., pitch class 0 can be spelt as B♯, C♮ or D♭♭) and ‘black note’ pitch classes to be spelt in two different ways (e.g., pitch class 6 can be spelt as F♯ or G♭) (see, for example, Cambouropoulos, 1996, p. 242). All possible spellings for each window are then generated and a penalty score is computed for each spelling. This penalty score is found by computing a penalty value for each pitch interval in the spelling and summing these interval penalty values. A given interval is penalised more heavily the less frequently it occurs in the major and minor scales, a principle that Cambouropoulos (2003, p. 421) calls *interval optimisation*. An interval is also penalised if either

TPR 1 (Pitch Variance Rule) Prefer to label nearby events so that they are close together on the line of fifths.

TPR 2 (Voice-Leading Rule) Given two events that are adjacent in time and a half-step apart in pitch height: if the first event is remote from the current center of gravity, it should be spelled so that it is five steps away from the second on the line of fifths.

TPR 3 (Harmonic Feedback Rule) Prefer TPC representations which result in good harmonic representations.

Figure 4: Temperley's preference rule system for pitch spelling (from Temperley, 2001, pp. 124–132, 359).

of the pitch names forming the interval is a double-sharp or a double-flat, a principle that Cambouropoulos (2003, p. 421) calls *notational parsimony*. For each window, the algorithm chooses the spelling that has the lowest penalty score.

The earliest published version of this method (Cambouropoulos, 1996) was designed for processing monophonic melodies and includes a rule, based on Krumhansl's (1990, pp. 150–151) principle of contextual asymmetry, that is applied as a 'tie-breaker' when two or more spellings for a given window achieve the least penalty score. The two more recent published versions (Cambouropoulos, 2001, 2003) are adapted for processing polyphonic music. However, they differ from each other in that the earlier of the two (Cambouropoulos, 2001, p. 5) uses a "variable length window" which always contains a fixed number of *distinct* MIDI note numbers.

An analysis of the versions of this algorithm described by Cambouropoulos in his publications revealed a number of ways in which two versions might differ in their detailed features. For example, one might use a variable length window and the other a fixed length window; one might use the 'tie breaker' function and the other might not; and so on. Twenty-six versions of the algorithm were tested in this study, including those described by Cambouropoulos himself together with other versions that were carefully selected so that an estimate could be obtained as quickly as possible of the best combination of detailed features. This 'optimal' combination was then tested by implementing it in a final version of the algorithm which was run on the test corpus.

3.3 Temperley and Sleator's pitch spelling algorithm

Temperley's (2001) theory of music cognition consists of *preference rule systems* for six aspects of musical structure: metre, phrasing, counterpoint, harmony, key and pitch spelling. Most of this theory has been implemented by Daniel Sleator in a suite of computer programs called *Melisma*.[1] These programs take "note list" representations as input (Temperley, 2001, pp. 9–12) in which the pitch of each note (or sequence of tied notes) is represented by its MIDI note number and its onset-time and offset-time are given in milliseconds.

Temperley's theory of pitch spelling—or, as he calls it, "tonal-pitch-class labeling" (Temperley, 2001, pp. 123–132)—consists of the three *tonal-pitch-class preference rules* (TPRs) shown in Figure 4. In Temperley's theory, the *tonal pitch class* (TPC) of a note is an integer that indicates the position of the pitch name class of the note on the line of fifths (Temperley, 2001, pp. 118, 123–125). Temperley (2001, p. 125) claims that TPR 1 (see Figure 4) is "the most important" TPR and that "in many cases, this rule is sufficient to ensure the correct spelling of passages". TPR 2 is designed to account for the way that notes are typically spelt in chromatic scale segments (Temperley, 2001, pp. 127–130). TPR 3 states that the system should "prefer TPC representations which result in good harmonic representations" (Temperley, 2001, p. 131). Temperley formally defines the concept of a "good harmonic representation" in the first rule of his theory of harmony, HPR 1 (Temperley, 2001, p. 149), which states that, in choosing the roots for chords, certain specified TPC-root relationships should be preferred over others. Temperley's theories of pitch spelling and harmony are therefore interdependent and this is reflected in the fact that they are both implemented in the `harmony` program in *Melisma*.

The complexity of Temperley's pitch spelling algorithm is increased still further by the fact that his theory of harmony depends on his theory of metrical structure. For example, the second harmonic preference rule states that the system should "prefer chord spans that start on strong beats of the meter" (Temperley, 2001, pp. 151, 359). The `harmony` program therefore requires as input both a "note list" and a representation of the metrical structure of the passage in the form of a "beat list" of the type generated by the *Melisma* `meter` program. Consequently, if one wishes to use Temperley's theory to determine the pitch names of the notes in a passage, one must carry out a process which we denote here by "MH" in which one first uses the `meter` program to generate a beat list from a note-list and then processes both the note list and the beat list using the `harmony` program to compute the pitch names of the notes in the passage.

In an attempt to take harmonic rhythm into account when computing metrical structure, Temperley and Sleator also experimented with a "two-pass" method (Temperley, 2001, pp. 46–47), in which the `meter` and `harmony` programs are both run twice, once in a special "prechord" mode and then again in "normal" mode. We denote this "two-pass" method by "MH2P".

The output of the `meter` program depends on tempo. Therefore, in the evaluation described here, both MH and MH2P were run on six different versions of the test corpus, one in which the music is at a "natural" tempo and five other versions in which the tempo is multiplied by 2, 4, $\frac{1}{2}$, $\frac{1}{4}$ and $\frac{1}{6}$. To test the extent to which pitch spelling depends on metrical structure, the half-speed version of the test corpus was run through the `meter` program and then the beat list generated for each movement was modified so that every beat had the same strength. These beat lists were then used to generate pitch names using the `harmony` program. We denote this procedure by HNM (for "Harmony-No-Meter"). Finally, a very simple implementation of TPR 1 was run on the dataset to test Temperley's (2001, p. 125) claim that "in many cases, this rule is sufficient to ensure the correct spelling of passages" .

[1]Available online at
`<http://www.link.cs.cmu.edu/music-analysis/>`.

3.4 Chew and Chen's pitch spelling algorithm

Chew and Chen (2003, 2005) describe several variants of a real-time pitch spelling algorithm based on Chew's (2000) "Spiral Array Model" which is a geometric model of tonal pitch relations. In the spiral array, the pitch name classes are arranged on a helix so that adjacent pitch name classes within this helix are a perfect fifth apart and adjacent pitch name classes along the length of the cylinder in which the helix is embedded are a major third apart. Let S be a set of notes, let $\mathbf{p}(n)$ denote the vector representing the position in the spiral array of the pitch name class of the note n, and let $d(n)$ be the duration of note n. Chew and Chen (2005, p. 67) define the *center of effect* (CE) of S, denoted by $CE(S)$, to be

$$CE(S) = \frac{\sum_{n\in S} d(n)\mathbf{p}(n)}{\sum_{n\in S} d(n)}.$$

That is, the CE of a set of notes is the weighted centroid of the position vectors of the pitch name classes of the notes in the spiral array, each note being weighted by its duration.

In Chew and Chen's algorithm, it is assumed that the input data gives the MIDI note number, together with the onset and duration in milliseconds of each note. The data is then divided into equal time slices called *chunks* (Chew and Chen, 2005, p. 67) and the algorithm spells the notes a chunk at a time. Let's denote by $W_{\text{sound}}(i, j)$ the set of notes that are sounding in chunks i to j; and let $W_{\text{start}}(i, j)$ denote the set of notes that *start* in chunks i to j. Let's suppose that the algorithm is about to spell the notes in chunk j. According to Chew and Chen (2005, pp. 70–71), the algorithm first computes the *global CE*, $CE_{\text{global},j}$, which is the CE of the set of notes in a sliding *global context window* that consists of the w_s chunks immediately preceding the jth chunk. In other words, the algorithm computes the value

$$CE_{\text{global},j} = CE(W_{\text{sound}}(j - w_s, j - 1)), \qquad (1)$$

where w_s is a parameter of the algorithm. Next, each note in chunk j is assigned a pitch name which is as close as possible to $CE_{\text{global},j}$ in the spiral array. Then a *local CE*, $CE_{\text{local},j}$, is computed which is the CE of the set of notes in a *local context window* consisting of the chunk j that has just been spelt, together with the $(w_r - 1)$ chunks immediately preceding the jth chunk. That is, the algorithm computes the value of

$$CE_{\text{local},j} = CE(W_{\text{sound}}(j - w_r + 1, j)), \qquad (2)$$

where w_r is another parameter of the algorithm. Next the algorithm computes the *cumulative CE*, $CE_{\text{cum},j}$, which is the CE of the notes in a *cumulative window* consisting of all the chunks preceding the jth chunk. That is, it computes the value of $CE_{\text{cum},j} = CE(W_{\text{sound}}(1, j - 1))$. Finally, the notes in chunk j are re-spelt so that their pitch names are as close as possible to the *hybrid CE*, $CE_{\text{hybrid},j} = f.CE_{\text{local},j} + (1 - f).CE_{\text{cum},j}$, where f is a parameter with a value between 0 and 1 which determines the relative weight given to the local and cumulative CEs.

Table 1: The sets of parameter values used to evaluate Chew and Chen's algorithm.

Parameter	*Set of values used*
w_s	$\{0, 4, 8, 16\}$
w_r	$\{2, 4, 6\}$
f	$\{0, 0.25, 0.5, 0.75, 1\}$
r/h	$\left\{\sqrt{2/15}, \sqrt{15/2}\right\}$
Chunk size in ms	$\{500, 1000, 2000\}$
W_{sound} or W_{start}	$\{W_{\text{sound}}, W_{\text{start}}\}$
Spiral array or line of fifths	{Spiral array, Line of fifths}
Range of permitted pitch name classes	{F♭♭ . . . B𝄪, F♭♭♭ . . . B𝄪♯}

In our implementation of Chew and Chen's algorithm, another parameter is r/h, the aspect ratio of the spiral array, where r is the radius of the cylinder in which the helix is embedded and h is the distance parallel to the axis of the helix corresponding to one step along the spiral array (i.e., two pitch name classes a perfect fifth apart). Other parameters in our implementation allow the user to: (1) choose to use the line of fifths instead of the spiral array; (2) specify the size of each chunk in milliseconds; (3) specify the segment of the spiral array (or line of fifths) from which the algorithm is permitted to choose pitch names; and (4) replace W_{sound} with W_{start} in Eqs. 1 and 2. Our implementation of Chew and Chen's algorithm was run on the test corpus 1260 times, each time with a different combination of parameter values. All possible combinations were used of the parameter values shown in Table 1.

3.5 Meredith's pitch spelling algorithms

Meredith (2003, 2005) presents an algorithm called *ps13* which takes as input a sequence of ordered pairs, I, in which each ordered pair $\langle t_{\text{on}}, n\rangle$ gives the onset time t_{on} and the MIDI note number, n, of a note or sequence of tied notes in the passage to be analysed.[2] The elements of I are sorted by increasing onset time and MIDI note number (with preference given to onset time).

If S is an ordered set, let $|S|$ denote the length of S and let $S[i]$ denote the $(i + 1)$th element of S (e.g., $S[0]$ is the first element in S). An ordered set of pitch classes, C, is first derived from I such that $|C| = |I|$ and $C[i]$ is the pitch class of the note represented by $I[i]$ for all $0 \leq i < |I|$. *ps13* consists of two stages, *Stage 1* and *Stage 2*.

Stage 1 consists of the following steps:

1. For each $0 \leq i < |C|$ and each pitch class $0 \leq p \leq 11$, compute a value $\mathit{CNT}(i, p)$ giving the number of times that p occurs within a context surrounding $C[i]$ that includes $C[i]$, K_{pre} notes immediately preceding $C[i]$ and $K_{\text{post}} - 1$ notes immediately following $C[i]$. K_{pre} is called the *precontext* and K_{post} is called the *postcontext*.

2. For each $0 \leq i < |C|$ and each pitch class $0 \leq p \leq 11$, compute the number of diatonic steps $(\text{mod}7)$,

[2] *ps13* is currently the subject of patent applications in the UK (GB0406166.9) and US (US10/821962). Please contact the author at `dave@titanmusic.com` if you wish to use the algorithm. Permission will normally be granted for free use of the algorithm for non-commercial purposes.

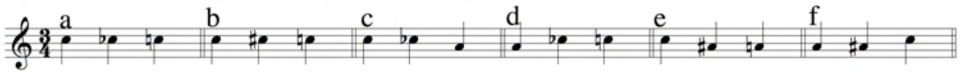

Figure 5: Examples of the types of passing and neighbour note errors corrected in *Stage 2* of *ps13*.

$D(i,p)$, that there would be from the tonic to the pitch name of $C[i]$ if p were the pitch class of the tonic at the point where $C[i]$ occurs. Assume that the notes are spelt so that they are as close as possible to the pitch name of p on the line of fifths, with every note 6 semitones away from p $(\text{mod } 12)$ being assigned a pitch name which is an augmented fourth *above* that of p. In other words, the notes are spelt as they would be in a harmonic chromatic scale whose tonic has pitch class p.

3. For each $0 \leq i < |C|$ and each pitch class $0 \leq p \leq 11$, compute the value

$$D'(i,p) = (D(i,p) - D(0,p)) \bmod 7.$$

$D'(i,p)$ gives the number of diatonic steps $(\text{mod } 7)$ from the pitch name of the first note (i.e., the note corresponding to $C[0]$) to the pitch name of $C[i]$ if the tonic pitch class is p.

4. For each $0 \leq i < |C|$ and each diatonic interval $0 \leq d \leq 6$, compute the set of tonic pitch classes,

$$X(i,d) = \{p \mid D'(i,p) = d\}.$$

$X(i,d)$ contains the tonic pitch classes that would lead to the diatonic interval from the first note to $C[i]$ being d.

5. For each $0 \leq i < |C|$ and each diatonic interval $0 \leq d \leq 6$, compute the sum, $N(i,d)$, of the values of $\mathit{CNT}(i,p)$ for all the tonic pitch classes $p \in X(i,d)$. That is,

$$N(i,d) = \sum_{p \in X(i,d)} \mathit{CNT}(i,p).$$

6. For each $0 \leq i < |C|$, compute $d_{\text{max}}(i)$, the value of d for which $N(i,d)$ is a maximum.

7. Assign a letter name to the first note, $C[0]$. This can be done, for example, by using the following table

$C[0]$	0	1	2	3	4	5
Letter name of $C[0]$	C	C	D	E	E	F
$C[0]$	6	7	8	9	10	11
Letter name of $C[0]$	F	G	A	A	B	B

8. For each $0 \leq i < |C|$, make the letter name of the note corresponding to $C[i]$, $d_{\text{max}}(i)$ steps above the letter name assigned to the note corresponding to $C[0]$.

Stage 2 of the algorithm corrects those instances in the output of *Stage 1* where a neighbour note or passing note is erroneously predicted to have the same letter name as either the note preceding it or the note following it (see Figure 5).

ps13 was run on the test corpus used in this study with $K_{\text{pre}} = 33$ and $K_{\text{post}} = 23$, these being the values that resulted in the highest note accuracy in a pilot study in which the algorithm was run on the first book of J. S. Bach's *Das Wohltemperirte Clavier* (Meredith, 2003; Meredith, 2005, pp. 182–183).

A second version of this algorithm, called *ps1303*, was also run on the test corpus. *ps1303* first predicts the pitch names of the notes in the passage using *ps13*. Then, for each note, it determines whether the pitch name predicted by *ps13* is relatively distant from the pitch names in its context along the line of fifths. If the pitch name assigned to a note is relatively distant from its neighbouring notes along the line of fifths, and it can be made closer to these neighbours by transposing it either up or down a diminished second, then it is transposed by the appropriate interval.

4 RESULTS AND DISCUSSION

Table 2 gives the results obtained for selected versions of the algorithms tested (including the best versions for each author), sorted by decreasing overall note accuracy. The results in parentheses in this table give the note accuracies obtained when the output for the fourth movement of Haydn's 'Military' Symphony was compared with the version in which the sudden enharmonic change was omitted (see section 2).

Of the versions of Longuet-Higgins's algorithm tested, the original version performed best, spelling 98.21% of the notes correctly when the notes were sorted so that the voices were 'end-to-end'. Correcting the implementation of Longuet-Higgins's second rule had no effect when the notes in the data were sorted by onset time and MIDI note number, and increased the number of errors when the notes were sorted so that the voices were 'end-to-end'. All versions of Longuet-Higgins's algorithm made roughly half as many errors when the voices were end-to-end as when the notes were sorted by onset time and MIDI note number. This supports Longuet-Higgins's prediction that the algorithm would work less well on polyphonic music. Removing the restriction that pitch names must be between G𝄫 and A𝄪 on the line of fifths more than doubled the number of errors made by the algorithms. Finally, the style dependence of the versions of the algorithm tested increased monotonically with note error rate so that the most accurate version of the algorithm was also the one whose accuracy was least affected by musical style.

Of the 26 versions of Cambouropoulos's algorithm tested here, the one that performed best was based on the variable-length window version described by Cambouropoulos (2001). This version achieved a note accuracy of 99.07% and a style dependence of 0.46. In gen-

Table 2: Summary of results for selected versions of the algorithms tested. Values in columns 2 to 10 are note accuracies, expressed as percentages. Each value in column headed SD measures style dependence and gives the standard deviation of the percentages in the same row in columns 2 to 9.

Algorithm	Bach	Beethoven	Corelli	Handel	Haydn	Mozart	Telemann	Vivaldi	Complete	SD
ps1303	99.92	98.31	99.99	99.71	99.19	99.28	99.69	99.32	99.43	0.54
ps13	99.86	98.27	99.92	99.71	98.90	99.04	99.57	99.21	99.31	0.56
Temperley-Sleator (MH2P, half-speed)	99.88	96.67	99.98	99.93	86.82 (98.91)	99.31	99.89	99.87	97.79 (99.30)	4.57 (1.13)
Temperley-Sleator (MH, half-speed)	99.87	96.61	99.98	99.91	86.72 (98.82)	99.33	99.91	99.89	97.78 (99.29)	4.61 (1.16)
Chew-Chen (sounding)	99.29	98.73	99.38	99.44	98.51	99.06	99.39	99.40	99.15	0.35
Chew-Chen (starting)	99.39	98.80	99.44	99.46	98.28	99.10	99.37	99.37	99.15	0.42
Cambouropoulos (optimal)	99.08	99.01	99.44	99.49	98.07	99.36	99.28	99.43	99.15	0.47
Cambouropoulos (2001)	98.85	99.05	99.42	99.48	98.06	99.18	99.30	99.22	99.07	0.46
Temperley's TPR 1 alone	99.38	97.83	99.51	99.32	98.25	99.04	99.39	99.57	99.04	0.65
Temperley-Sleator (HNM, half-speed)	99.71	96.72	99.96	99.87	86.95 (99.04)	99.38	99.88	95.48	97.25 (98.76)	4.49 (1.70)
Longuet-Higgins (original, voices end-to-end)	95.74	98.64	99.70	99.83	95.09	98.93	99.14	98.62	98.21	1.79
Longuet-Higgins (original, sorted by onset)	99.53	96.37	99.58	97.22	89.13	84.94	98.29	95.82	95.11	5.28

eral, the versions using a variable-length window achieved higher note accuracies than those using a fixed-length window of the same size. In the version of the algorithm described by Cambouropoulos (1996), the penalty score for each window spelling is computed by summing the penalty values only for intervals between contiguous notes. However, in the versions described by Cambouropoulos (2001, 2003), the penalty score for each window is computed by summing the penalty values for all the intervals between both contiguous and non-contiguous notes within the window. Our results showed that ignoring intervals between non-contiguous notes caused both a relatively large drop in spelling accuracy and a relatively large increase in style dependence. Nearly all the versions of Cambouropoulos's algorithm achieved higher note accuracies than the best version of Longuet-Higgins's algorithm and were markedly less dependent on style than those of Longuet-Higgins. Increasing the size of the window increased spelling accuracy but also exponentially increased running time. If each window contained more than 12 notes, the algorithm was too slow to be practical. The results were used to estimate an 'optimal' combination of features which were then implemented in a new version of the algorithm. This version achieved a note accuracy of 99.15% and a style dependence of 0.47 on the test corpus used here. However, it took nearly 7 times longer to process the test corpus than the most accurate of the other 26 versions tested.

It was found that both the MH and MH2P versions of Temperley and Sleator's algorithm performed very similarly and did best when the music in the corpus was at half speed (see Table 2). Both systems were highly sensitive to tempo: when the music was at 4 times its natural tempo, the note accuracy of MH was only 82.74% and that of MH2P was 74.58%. However, 70% of the errors made by MH and MH2P on the half-speed version of the corpus were caused by the sudden enharmonic change in the fourth movement of Haydn's 'Military' Symphony discussed in section 2. When the outputs of MH and MH2P were compared with the version of this movement with the enharmonic change omitted, their overall note accuracies rose to 99.3%. Ignoring metrical information in the HNM procedure caused the overall number of errors to rise by 24% when the original score of the fourth movement of Haydn's 'Military' Symphony was used as a ground truth and by 75% when the modified version was used. However, this fall in note accuracy was primarily due to HNM's inability to cope with one particular chromatic passage in bars 85–96 of the third movement of Vivaldi's Concerto in G minor ('L'estate') from 'Le Quattro Stagioni' (Op. 8, No. 2, RV 315). This suggests that the use of metrical structure in Temperley and Sleator's system helps it to cope with certain types of chromatic passage, which, in turn, makes it less likely to spell whole segments in the wrong key. It was found that a very simple implementation of TPR 1 alone, without any metrical information, was able to spell 99.04% of the notes in the test corpus correctly ($SD = 0.65$). Also, the performance of this version was independent of tempo.

Of the 1260 versions of Chew and Chen's algorithm tested here, 12 achieved a note accuracy of 99.15%. These 12 versions of the algorithm were those in which $w_s = 8$, $w_r = 2$, $f = 0.5$ and the chunk size was set to 500 milliseconds. The parameters that were critical for high note accuracy were therefore the duration of the windows used and the relative weighting given to local and global context. Amongst these 12 best versions of the algorithm, it did not matter whether the spiral array or the line of fifths was used, nor did the aspect ratio of the spiral array make any difference. Of these 12 versions, the six that considered the notes *sounding* within each window were slightly less dependent on style ($SD = 0.35$) than those that considered only the notes *starting* in each window ($SD = 0.42$).

Both versions of Meredith's algorithm achieved higher note accuracies in this study than any of the other algorithms tested. *ps13* spelt 99.31% of the notes in the corpus correctly. This value is only slightly more than that achieved by Temperley and Sleator's MH and MH2P systems when the enharmonic change in the fourth movement of Haydn's 'Military' Symphony was omitted. However, *ps13* was considerably less dependent on style ($SD = 0.56$) than the Temperley-Sleator algorithms ($SD = 1.13, 1.16$) as well as being independent of tempo. *ps1303* made even fewer errors than *ps13*, spelling 99.43% of the notes in the corpus correctly ($SD = 0.54$).

5 CONCLUSIONS

Various versions of the pitch spelling algorithms of Longuet-Higgins, Cambouropoulos, Temperley and Sleator, Chew and Chen and Meredith were compared by running them on a test corpus containing 195972 notes, equally divided between eight classical and baroque composers. Meredith's *ps1303* algorithm performed best, spelling 99.43% of the notes in the corpus correctly. The next best algorithm was Meredith's *ps13* which spelt 99.31% of the notes correctly, only slightly more than the 99.30% spelt correctly by the best version of Temperley and Sleator's algorithm (MH2P). However, MH2P only achieved this note accuracy when the music in the corpus was at half-speed and a sudden enharmonic change in one movement in the corpus was ignored. Furthermore, Temperley and Sleator's algorithms were very sensitive to tempo and considerably more dependent on style than those of Meredith which are unaffected by tempo. An attempt was made to find optimal versions of the Cambouropoulos and Chew-Chen algorithms. The best versions of these algorithms tested spelt 99.15% of the notes correctly and were slightly less dependent on style than Meredith's algorithms.

It would be interesting to compare the algorithms tested here with other pitch spelling algorithms described in the literature (e.g., that of Stoddard et al., 2004). It would also be useful to devise an appropriate statistical method for measuring the significance of the difference between two note accuracies achieved on the same test corpus. It would also be worth exploring the extent to which pitch spelling algorithms can be used to improve the performance of key-finding algorithms and content-based music information retrieval systems. Finally, it would be interesting to compare the algorithms studied here on a large, high-quality test corpus containing encodings of post-classical tonal works (e.g., 19th century romantic music, jazz). Unfortunately, such a collection of encodings does not yet exist in the public domain.

ACKNOWLEDGEMENTS

The research reported here was funded by EPSRC grant GR/S17253/02.

REFERENCES

S. A. Abdallah and M. D. Plumbley. Polyphonic transcription by non-negative sparse coding of power spectra. In *Proceedings of the Fifth International Conference on Music Information Retrieval (ISMIR 2004)*, Universitat Pompeu Fabra, Barcelona, Spain, 2004.

E. Cambouropoulos, M. Crochemore, C. S. Iliopoulos, L. Mouchard, and Y. J. Pinzon. Computing approximate repetitions in musical sequences. *International Journal of Computer Mathematics*, 79(11):1135–1148, 2002.

E. Cambouropoulos. A general pitch interval representation: Theory and applications. *Journal of New Music Research*, 25(3):231–251, 1996.

E. Cambouropoulos. Automatic pitch spelling: From numbers to sharps and flats. In *VIII Brazilian Symposium on Computer Music (SBC&M 2001)*, Fortaleza, Brazil, 2001.

E. Cambouropoulos. Pitch spelling: A computational model. *Music Perception*, 20(4):411–429, 2003.

E. Chew and Y.-C. Chen. Determining context-defining windows: Pitch spelling using the spiral array. In *Fourth International Conference on Music Information Retrieval (ISMIR 2003) (October 26–30)*, Baltimore, MD., 2003.

E. Chew and Y.-C. Chen. Real-time pitch spelling using the Spiral Array. *Computer Music Journal*, 29(2):61–76, 2005.

E. Chew. *Towards a Mathematical Model of Tonality*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA., 2000.

Z. Galil. On improving the worst case running time of the Boyer-Moore string matching algorithm. *Communications of the ACM*, 22(9):505–508, 1979.

D. E. Knuth, J. H. Morris, and V. R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6: 323–350, 1977.

C. L. Krumhansl. *Cognitive Foundations of Musical Pitch*, volume 17 of *Oxford Psychology Series*. Oxford University Press, New York and Oxford, 1990.

H. C. Longuet-Higgins. The perception of melodies. In H. C. Longuet-Higgins, editor, *Mental Processes: Studies in Cognitive Science*, pages 105–129. British Psychological Society/MIT Press, London, England and Cambridge, Mass., 1987.

Q. McNemar. *Psychological Statistics*. John Wiley and Sons, New York, 4th edition, 1969.

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

D. Meredith. Pitch spelling algorithms. In R. Kopiez, A. C. Lehmann, I. Wolther, and C. Wolf, editors, *Proceedings of the Fifth Triennial ESCOM Conference (ESCOM5) (8-13 September 2003)*, pages pp. 204–207, Hanover University of Music and Drama, Hanover, Germany, 2003.

D. Meredith. Comparing pitch spelling algorithms on a large corpus of tonal music. In U. K. Wiil, editor, *Computer Music Modeling and Retrieval, Second International Symposium, CMMR 2004, Esbjerg, Denmark, May 26–29, 2004, Revised Papers*, volume 3310 of *LNCS*, pages 173–192, Berlin, 2005. Springer.

J. Stoddard, C. Raphael, and P. E. Utgoff. Well-tempered spelling: A key-invariant pitch spelling algorithm. In *Proceedings of the Fifth International Conference on Music Information Retrieval (ISMIR 2004) (October 10–14)*, Universitat Pompeu Fabra, Barcelona, Spain, 2004.

D. Temperley. *The Cognition of Basic Musical Structures*. MIT Press, Cambridge, MA, 2001.