Comparing Pitch Spelling Algorithms on a Large Corpus of Tonal Music

David Meredith

Centre for Computational Creativity

Department of Computing

City University, London

`dave@titanmusic.com`

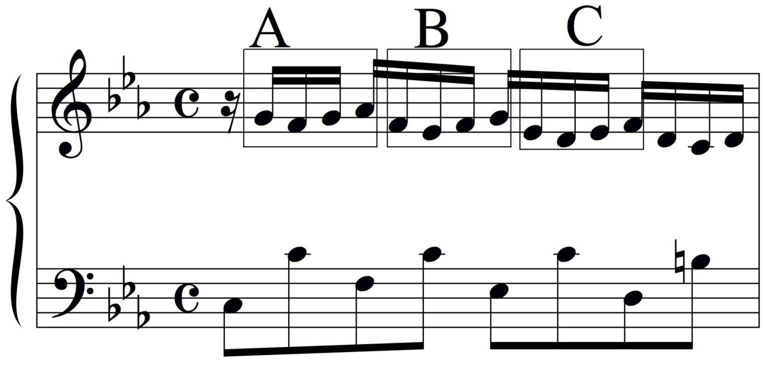
`www.titanmusic.com`

Second International Symposium on Computer Music Modeling and Retrieval (CMMR 2004)

Esbjerg, Denmark

Friday, 28 May 2004

1. The concept of a pitch spelling algorithm



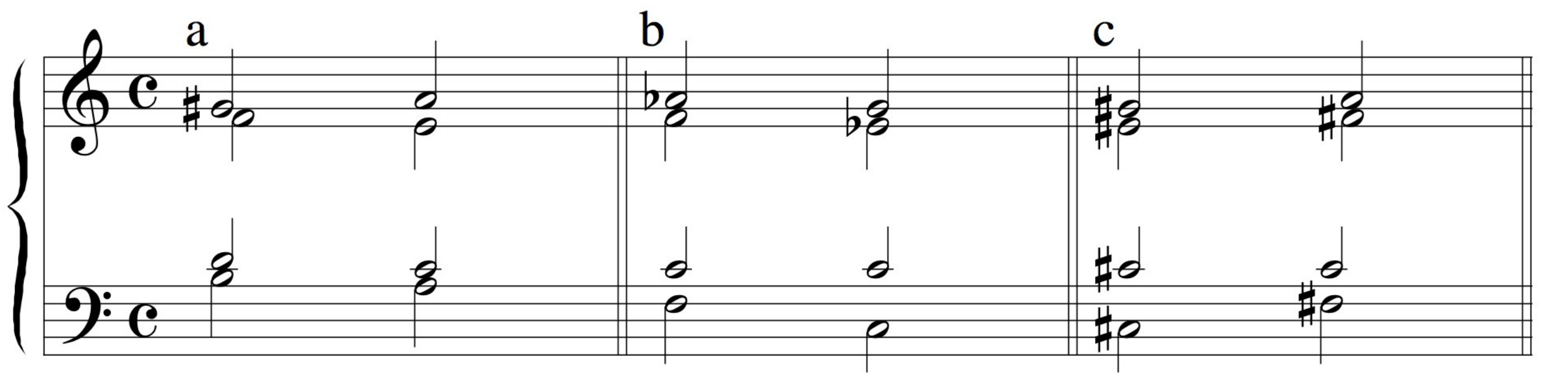
- A *pitch spelling algorithm* is an algorithm that attempts to compute the correct pitch names of the notes in a passage of tonal music, when given only the onset-time, MIDI note number and (possibly) the duration of each note.
- Practical applications of pitch spelling algorithms:

  1. Required for MIDI-to-notation transcription.
  2. Required for audio-to-notation transcription.
  3. Useful in music information retrieval and musical pattern discovery.

1. The concept of a pitch spelling algorithm

1. I'm going to talk to you about the problem of constructing and evaluating a reliable *pitch spelling algorithm*—that is, an algorithm that reliably computes 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 of each note in the passage.
2. Such algorithms are obviously required for automatically generating correctly notated scores from MIDI and audio files.
3. Less obviously, knowing the letter-names of the pitch events in a passage can be extremely useful in music information retrieval and musical pattern discovery (Meredith, Lemström, and Wiggins, 2002).
4. For example, the three patterns A, B and C here are heard as being three occurrences of the same motive even though the corresponding chromatic intervals are different in the three patterns.
5. Fast exact-matching algorithms can be used to find such occurrences if the pitch names of the notes are encoded, but not if the pitches are represented using just MIDI note numbers.

2. Pitch spelling in common practice Western tonal music



(Piston, 1978, p. 8)

2. Pitch spelling in common practice Western tonal music

1. In the vast majority of cases, the correct pitch name for a note in a passage of tonal music can be determined by considering the rôles that the note plays in the harmonic, motivic and voice-leading structures of the passage.
2. For example, when played in isolation in an equal-tempered tuning system, the first soprano note in (a) here would sound the same as the first soprano note in (b).
3. However, in (a), this note is spelt as a G♯ because it functions as a leading note in A minor; whereas in (b), the first soprano note is spelt as an A♭ because it functions as a submediant in C minor.
4. This illustrates the fact that the pitch name assigned to a note in a passage of tonal music is generally not arbitrary. In most cases, the pitch name of each note is carefully chosen so that the resulting score represents as well as possible certain important aspects of the way that the music is intended to be perceived and interpreted.

3. A comparison of three pitch spelling algorithms

- Three pitch spelling algorithms compared:
  - Cambouropoulos (1996, 1998, 2001, 2003)
  - Longuet-Higgins (1976, 1987, 1993)
  - Temperley (1997, 2001)
- Two test corpora used:
  - all pieces in first book of Bach's *Das Wohltemperirte Klavier* (BWV 846–869) (41544 notes)
  - 1655 movements from works by 9 baroque and classical composers (Corelli, Vivaldi, Telemann, Bach, Handel, B. Marcello, Haydn, Mozart and Beethoven) (1729886 notes)
- Corpora derived from MuseData collection of encoded scores (www.musedata.org).

3. A comparison of three pitch spelling algorithms

1. In order to get a clearer idea of the 'state of the art' in the field of pitch spelling algorithms, I compared the performance of three algorithms on two test corpora.
2. The algorithms compared were those of Cambouropoulos (1996, 1998, 2001, 2003), Longuet-Higgins (1976, 1987, 1993) and Temperley (1997, 2001).
3. I first carried out a pilot study in which I ran the algorithms on the first book of J. S. Bach's *Das Wohltemperirte Klavier* (BWV 846–869), which contains 41544 notes.
4. I then carried out a larger-scale study in which I ran the algorithms on a much larger corpus containing about 1.73 million notes and consisting of 1655 movements from works by 9 baroque and classical composers.
5. Both corpora were derived by automatic conversion from the MuseData collection of encoded scores (www.musedata.org).

5. Longuet-Higgins's (1976, 1987, 1993) algorithm

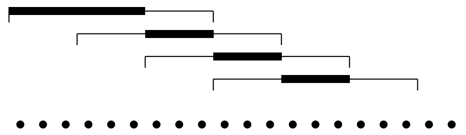
Pitch name	⋯	F♭	C♭	G♭	D♭	A♭	E♭	B♭	F	C	G	D	A	E	B	F♯	C♯	G♯	D♯	A♯	E♯	B♯	⋯
Sharpness	⋯	−8	−7	−6	−5	−4	−3	−2	−1	0	1	2	3	4	5	6	7	8	9	10	11	12	⋯

- Pitch spelling is one function of the `music.p` program.
- Designed to be used only on monophonic melodies.
- Computes value of *sharpness* for each note.
- Spells notes so they are as close as possible to tonic on line of fifths.
- Disallows consecutive chromatic intervals.
- Special rule for dealing with ascending semitones.

5. Longuet-Higgins's (1976, 1987, 1993) algorithm

1. Pitch spelling is one of the tasks performed by Longuet-Higgins's (1976, 1987, 1993) `music.p` program.
2. Longuet-Higgins (1987, p. 114) intended the `music.p` program to be used only on monophonic melodies and explicitly warns against using it on "accompanied melodies" or what he calls "covertly polyphonic" melodies (i.e., compound melodies).
3. The input to Longuet-Higgins's pitch spelling algorithm is essentially just a sequence of MIDI note numbers in the order in which they appear in the music.
4. The algorithm computes a value of "sharpness" q for each note in the input (Longuet-Higgins, 1987, p. 111). The sharpness of a note is a number indicating the position of the pitch name of the note on the line of fifths (as shown here) (Temperley, 2001, p. 117). It is therefore essentially the same as Temperley's (2001, p. 118) concept of "tonal pitch class" and Regener's (1973, p. 33) concept of *quint*.
5. Longuet-Higgins's algorithm tries to spell the notes so that they are as close as possible to the local tonic on the line of fifths (Longuet-Higgins, 1987, p. 113).
6. The algorithm also disallows consecutive chromatic intervals (Longuet-Higgins, 1987, p. 113) and incorporates a special rule for dealing with ascending semitones (Longuet-Higgins, 1987, p. 114).

6. Cambouropoulos's (1996, 1998, 2001, 2003) algorithm



- Input converted into a sequence of MIDI note numbers.
- Uses "shifting overlapping windowing technique" to improve running time and avoid errors at window boundaries.
- Allows 3 spellings for 'white note' pitch classes and 2 for 'black note' pitch classes.
- Computes all spellings for each window that do not contain both double-sharps and double-flats (128 spellings for each 9-note window).
- For each window spelling, computes penalty score based on principles of *interval optimisation* and *notational parsimony*.
- Chooses spelling with least penalty score and retains pitch names for middle third of the window.

6. Cambouropoulos's (1996, 1998, 2001, 2003) algorithm

1. The input to Cambouropoulos's method is again a sequence of MIDI note numbers in the order in which they occur in the music.
2. The algorithm uses a "shifting overlapping windowing technique" (Cambouropoulos, 2003, p. 420), as illustrated here, in which each window contains a certain number of contiguous elements in the input sequence.
3. On each step, the window position advances by a third of the window size, as you can see here.
4. Windowing improves the running time of the algorithm and overlapping the windows avoids certain types of errors at window boundaries.
5. Cambouropoulos allows 'white note' pitch classes (i.e., 0, 2, 4, 5, 7, 9 and 11) to be spelt in three ways. For example, pitch class 0 can be spelt as B♯, C♮ or D♭♭. 'Black note' pitch classes can be spelt in two ways. For example, pitch class 6 can be spelt as F♯ or G♭ (see, for example, Cambouropoulos, 1996, p. 242).
6. Given these restricted pitch name possibilities for each note, Cambouropoulos's method computes all the spellings for each window that do not contain both double-sharps and double-flats.
7. So, because the spelling for the first third of each window is retained from the previous window, there would be 128 different spellings for a 9-note window.
8. A penalty score is then computed for each of these possible window spellings. The penalty score for a given window spelling is found by computing a penalty value for each pitch interval in the window and summing these interval penalty values.

9. A given interval in a particular window spelling is penalised if it occurs infrequently in the major and minor scales, a principle that Cambouropoulos (2003, p. 421) calls *interval optimization*.
10. It is also penalised if either 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*.
11. For each window, the algorithm chooses the spelling that has the lowest penalty score and retains the pitch names for the middle third of this best window spelling.

7. Temperley's (1997, 2001) algorithm

- Searches for spelling that best satisfies following three preference rules:

  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.

- Requires duration of each note.
- Requires tempo (i.e., uses absolute performed duration).
- Cannot deal with cases where two or more notes with the same pitch start at the same time.
- Needs to compute metrical and harmonic structure in order to compute pitch names.

7. Temperley's (1997, 2001) algorithm

1. Temperley's (1997, 2001) pitch spelling algorithm is implemented in the **`harmony`** program which forms one component of his and Sleator's *Melisma* system.
2. The input to the **`harmony`** program must be in the form of a "note list" (Temperley, 2001, pp. 9–12) giving the MIDI note number of each note together with its onset time and duration in milliseconds.
3. This note list must be accompanied by a representation of the metrical structure of the passage in the form of a "beat-list" of the type generated by Temperley and Sleator's **`meter`** program.
4. Temperley's (2001, pp. 115–136) pitch spelling algorithm searches for the spelling that best satisfies three "preference rules".
   (a) The first of these rules stipulates that the algorithm should "prefer to label nearby events so that they are close together on the line of fifths" (Temperley, 2001, p. 125). This rule bears some resemblance to the basic principle underlying Longuet-Higgins's algorithm (see above).
   (b) The second rule expresses the principle that if two tones are separated by a semitone and the first tone is distant from the key centre, then the interval between them should preferably be spelt as a diatonic semitone rather than a chromatic one (Temperley, 2001, p. 129). This rule is also very similar to one of the rules used in Longuet-Higgins's algorithm.
   (c) The third preference rule steers the algorithm towards spelling the notes so that what Temperley calls a "good harmonic representation" results (Temperley, 2001, p. 131).
5. Note, however, that Temperley's algorithm requires more information in its input than the other algorithms. In particular, it needs to know the duration of each note and the tempo at each point in the

passage. It also needs to perform a full analysis of the metrical and harmonic structure of the passage in order to generate a high quality result.

6. Also, it cannot deal with cases where two or more notes with the same pitch start at the same time.

8. Results of running algorithms on first book of J. S. Bach's *Das Wohltemperirte Klavier*

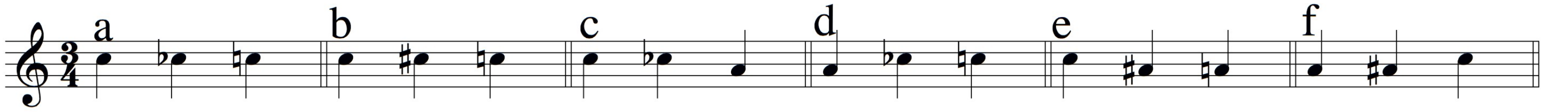
Algorithm	*% notes correct*	*Number of errors*
Cambouropoulos	93.74	2599
Longuet-Higgins	99.36	265
Temperley	99.71	122

Total number of notes in corpus = 41544.

8. Results of running algorithms on first book of J. S. Bach's *Das Wohltemperirte Klavier*

1. When these three algorithms were run on the first book of J. S. Bach's *Das Wohltemperirte Klavier*, the results obtained were as shown here.

9. The *ps13* algorithm



Step 1 For each note n and each pitch class p, compute $CNT(p,n)$ which is the number of times that p occurs in a context surrounding n that includes K_{pre} notes preceding n and K_{post} notes following n.

Step 2 For each note n and each pitch class p, compute the letter name $L(p,n) \in \{\text{A,B,C,D,E,F,G}\}$ that n would have if p were the tonic at the point where n occurs (assuming that the notes are spelt as they are in the harmonic chromatic scale on p).

Step 3 For each note n and each letter name ℓ, compute the set of tonic pitch classes, $X(n,\ell)$, that would lead to n having the letter name ℓ.

Step 4 For each note n and each letter name ℓ, compute the sum, $N(\ell,n)$, of the values of $CNT(p,n)$ for all the tonic pitch classes $p \in X(n,\ell)$.

Step 5 Make the letter name of n equal to that value of ℓ for which $N(\ell,n)$ is a maximum.

In Stage 2, neighbour-note and passing-note errors corrected.

9. The *ps13* algorithm

1. Having done this comparison and gained a better idea of the ‘state of the art’ in the field, I attempted to construct a new algorithm that improved on Temperley’s.

2. I experimented with about 30 different algorithms and I’ll now briefly describe the one that performed best, an algorithm that I call *ps13*.[1]

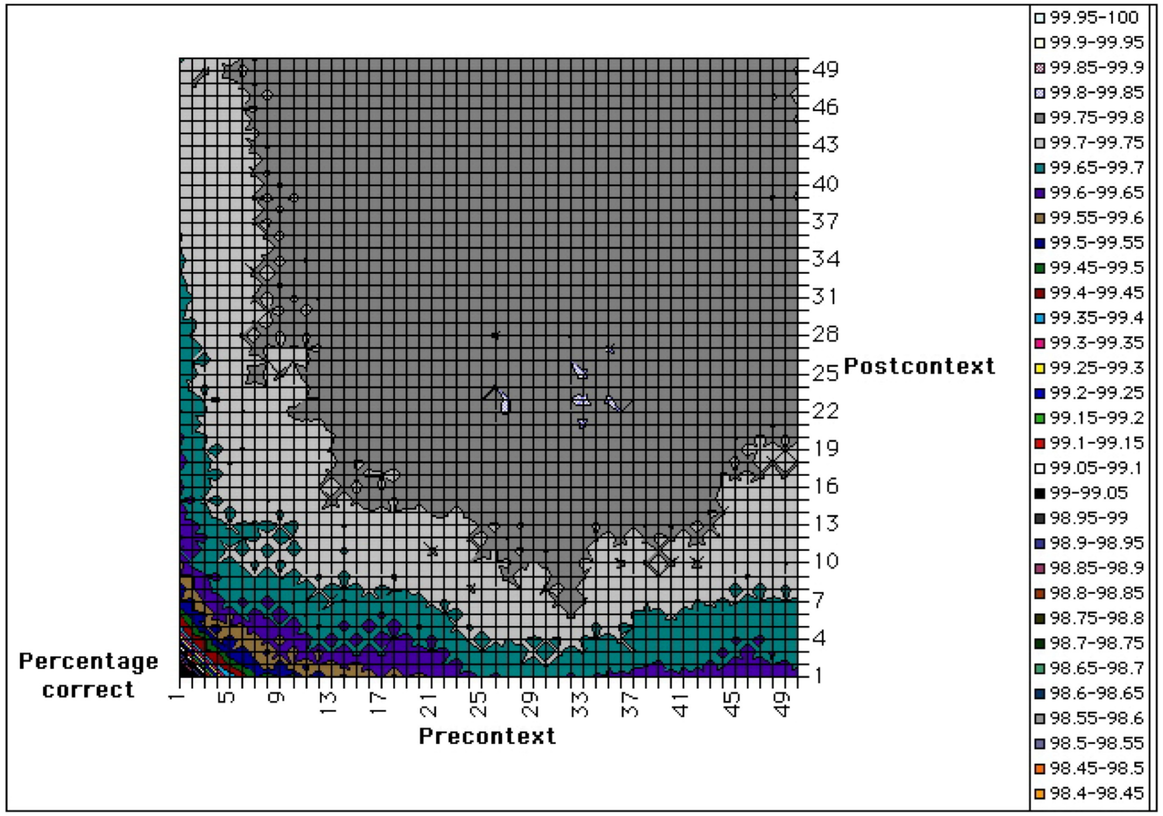
3. At the highest level of description, *ps13* can be broken down into two stages, which I’ll call Stage 1 and Stage 2.

4. Stage 1 involves carrying out the following steps: [SEE SLIDE]

5. 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. That is, the second stage of the algorithm corrects errors like these shown on the staff at the top here.

[1]Patent pending (Meredith, 2003).

10. Results of running *ps13* on the first book of J. S. Bach's *Das Wohltemperirte Klavier*

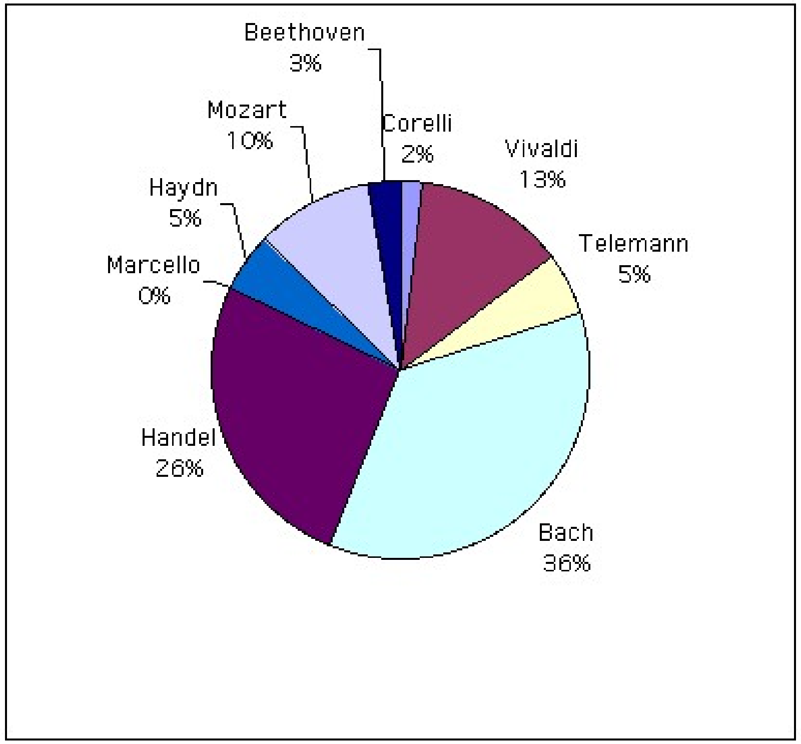


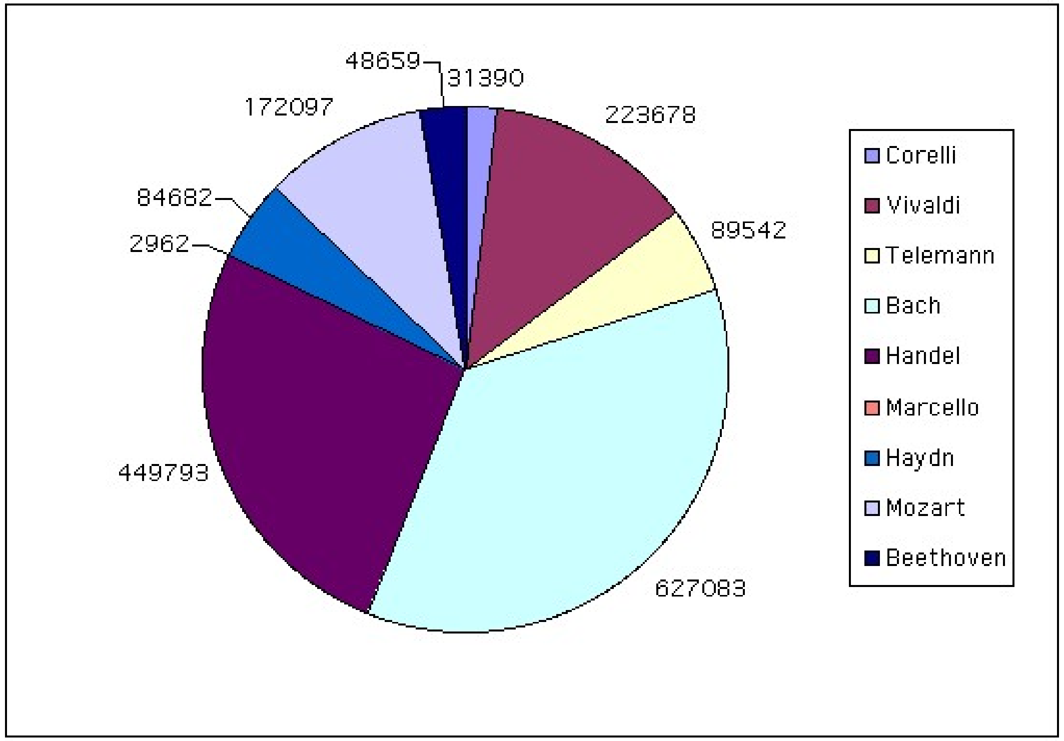
- When $K_{\text{pre}} = 33$ and $K_{\text{post}} \in \{23, 25\}$, *ps13* makes 81 mistakes on this corpus (i.e., 99.81% notes spelt correctly).

10. Results of running *ps13* on the first book of J. S. Bach's *Das Wohltemperirte Klavier*

1. In the first step of Stage 1, *ps13* counts how many times each pitch class occurs within some specified context surrounding a particular note, defined by the values of K_{pre} and K_{post}.
2. This value is supposed to give an approximate measure of how tonally stable the pitch class is in that context.
3. In order to find the optimal values of K_{pre} and K_{post}, I ran the algorithm 2500 times on the first book of Bach's 48, each time using a different pair of values for the K_{pre} and K_{post} chosen so that both were between 1 and 50.
4. I found that *ps13* performed better than Temperley's algorithm for just over 80% of the $\left\langle K_{\text{pre}}, K_{\text{post}} \right\rangle$ pairs tested.
5. *ps13* performed best on the test corpus when K_{pre} was set to 33 and K_{post} was set to either 23 or 25. With these parameter values, *ps13* made only 81 errors on the test corpus—that is, it correctly predicted the pitch names of 99.81% of the notes in the test corpus.
6. Over all 2500 $\left\langle K_{\text{pre}}, K_{\text{post}} \right\rangle$ pairs tested, *ps13* made on average only 109 note spelling errors (that is, it spelt 99.74% of the notes correctly).
7. Note that this average value was better than the result obtained by Temperley's algorithm for this test corpus.

11. Structure of the larger test corpus





- Total number of notes in corpus = 1729886
- Total number of movements = 1655

11. Structure of the larger test corpus

1. I then ran all four algorithms on a much larger corpus containing about 1.7 million notes and consisting of 1655 movements from works by 9 baroque and classical composers (Corelli, Vivaldi, Telemann, Bach, Handel, B. Marcello, Haydn, Mozart and Beethoven).
2. The values of K_{pre} and K_{post} for *ps13* were set to 33 and 23, respectively, these being the values that produced the best results when the algorithm was run on the smaller corpus in the pilot study.
3. These pie charts show the percentage and number of notes in this larger corpus in works by each composer.
4. As you can see, about 80% of the music in the corpus is baroque and the remaining 20% is classical. So the corpus is not very stylistically varied—it contains music written between about 1675 and 1825.
5. Also, note that the corpus is very unevenly distributed between the nine composers.
6. Unfortunately, a more varied and balanced collection of encoded scores of a comparable size does not seem to be currently available.

12. Comparison of algorithms with respect to number of notes spelt correctly:

Number of notes spelt incorrectly

	Cam	LH	ps13	Tem
Corelli	148	120	30	6
Vivaldi	1816	5518	1497	4900
Telemann	604	665	481	111
Bach	13347	13022	3450	1166
Handel	1929	3342	2339	1309
Marcello	1	4	14	5
Haydn	1497	6174	823	6391
Mozart	2300	10147	2250	19832
Beethoven	600	1703	727	6525
Complete test corpus	22242	40695	11611	40245

12. Comparison of algorithms with respect to number of notes spelt correctly: Number of notes spelt incorrectly

1. This table shows the number of notes spelt incorrectly by each algorithm for each composer in the corpus.
2. The bottom row in this table gives the total number of notes in the corpus spelt incorrectly by each algorithm.
3. As you can see, *ps13* made about half as many errors as Cambouropoulos's algorithm which made about half as many errors as the algorithms of Longuet-Higgins and Temperley.

13. Comparison of algorithms with respect to number of notes spelt correctly:

Percentage of notes spelt correctly

	Cam	LH	ps13	Tem
Corelli	99.53%	99.62%	99.90%	99.98%
Vivaldi	99.19%	97.53%	99.33%	97.81%
Telemann	99.33%	99.26%	99.46%	99.88%
Bach	97.87%	97.92%	99.45%	99.81%
Handel	99.57%	99.26%	99.48%	99.71%
Marcello	99.97%	99.86%	99.53%	99.83%
Haydn	98.23%	92.71%	99.03%	92.45%
Mozart	98.66%	94.10%	98.69%	88.48%
Beethoven	98.77%	96.50%	98.51%	86.59%
Complete test corpus	98.71%	97.65%	99.33%	97.67%

13. Comparison of algorithms with respect to number of notes spelt correctly: Percentage of notes spelt correctly

1. This table shows the *percentage* of notes spelt correctly by each algorithm for each composer.
2. The bottom row shows the total percentage of notes in the corpus spelt correctly by each algorithm.
3. As you can see, *ps13* did best with 99.33% of the notes spelt correctly.

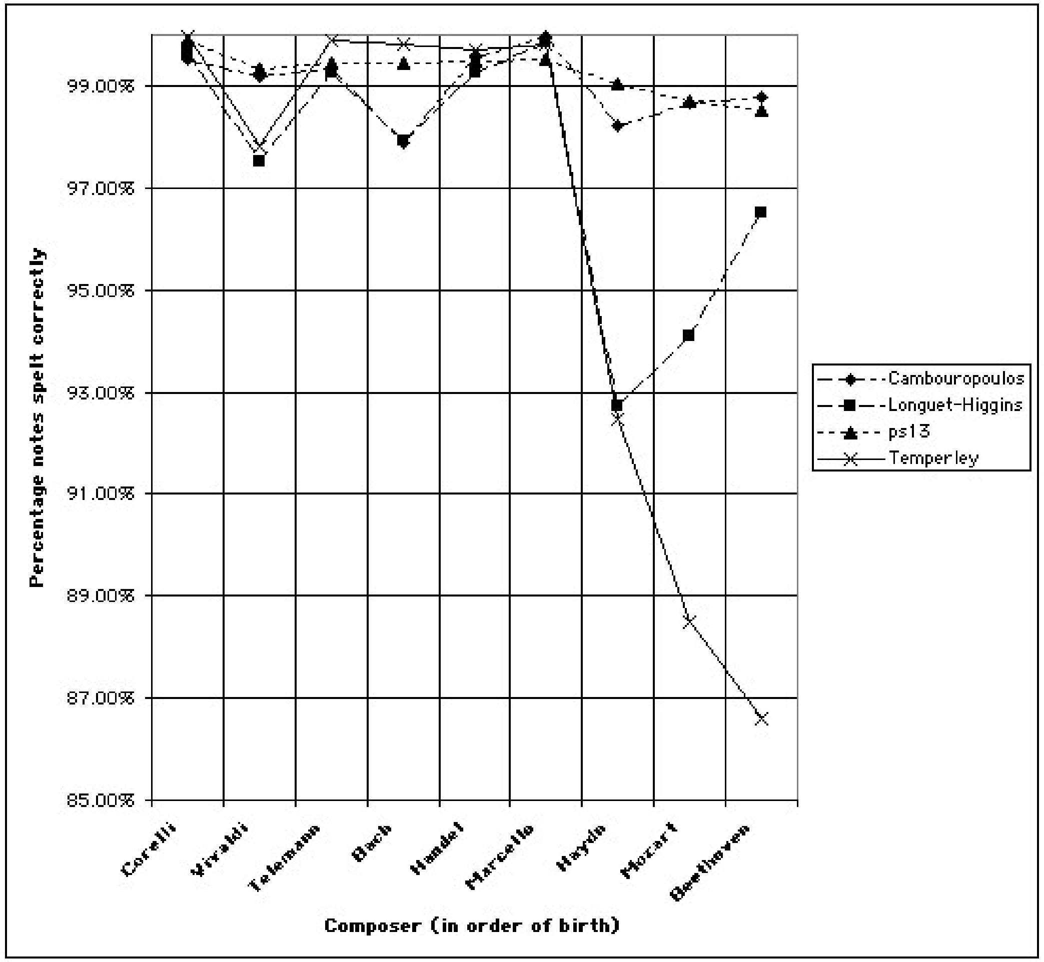
14. Comparison of algorithms with respect to number of notes spelt correctly: Significance of differences between numbers of errors

Composer	Best	p-value	2nd best	p-value	3rd best	p-value	Worst
Corelli	Tem	< 0.0001	ps13	< 0.0001	LH	0.0765	Cam
Vivaldi	ps13	< 0.0001	Cam	< 0.0001	Tem	< 0.0001	LH
Telemann	Tem	< 0.0001	ps13	< 0.0001	Cam	0.0736	LH
Bach	Tem	< 0.0001	ps13	< 0.0001	LH	0.0352	Cam
Handel	Tem	< 0.0001	Cam	< 0.0001	ps13	< 0.0001	LH
Marcello	Cam	0.1797	LH	0.7388	Tem	0.0389	ps13
Haydn	ps13	< 0.0001	Cam	< 0.0001	LH	0.0348	Tem
Mozart	ps13	0.3665	Cam	< 0.0001	LH	< 0.0001	Tem
Beethoven	Cam	< 0.0001	ps13	< 0.0001	LH	< 0.0001	Tem
Complete test corpus	ps13	< 0.0001	Cam	< 0.0001	Tem	0.0954	LH

14. Comparison of algorithms with respect to number of notes spelt correctly: Significance of differences between numbers of errors

1. I then used McNemar's test (Dietterich, 1998; McNemar, 1969) to determine whether the differences between the scores achieved by the algorithms were statistically significant.
2. The results of this analysis are shown here.
3. In this table, each value in a column headed 'p-value' gives the statistical significance (expressed as a probability computed using McNemar's test) of the difference in performance between the algorithms to the left and right of the value in the table.
4. For example, the value in the seventh column and first row of this table indicates that if the algorithms of Longuet-Higgins and Cambouropoulos would actually make an equal number of note errors when run over the whole population of works represented by the works by Corelli in the test corpus, then the probability of getting a difference in scores at least as great as that observed in this study would be 0.0765.
5. In psychological experiments, a p-value less than 0.05 is usually considered significant.
6. The bottom row of this table shows that, overall, *ps13* spelt very significantly more notes correctly than Cambouropoulos's algorithm ($p < 0.0001$), which in turn spelt very significantly more notes correctly than Temperley's algorithm ($p < 0.0001$). However, the percentage of notes spelt correctly by Temperley's algorithm (97.67%) and Longuet-Higgins's algorithm (97.65%) did not differ significantly ($p = 0.0954$).

15. Comparison of algorithms with respect to number of notes spelt correctly: Accuracy depends on style of music analysed



15. Comparison of algorithms with respect to number of notes spelt correctly: Accuracy depends on style of music analysed

1. This graph shows the percentage of notes spelt correctly by each algorithm for each composer, the composers being placed along the horizontal axis in order of birth year.
2. This graph suggests that the algorithms of Temperley and Longuet-Higgins perform significantly worse on the classical composers (Haydn, Mozart and Beethoven) than they do on the baroque composers.
3. The graph also seems to show that *ps13* performs more consistently across the different composers and styles than the other algorithms.

16. Comparison of algorithms with respect to number of intervals spelt correctly:

Number of intervals spelt incorrectly

	Cam	LH	ps13	Tem
Corelli	288	92	60	12
Vivaldi	2064	1264	1937	941
Telemann	727	450	788	204
Bach	12697	5852	4507	2074
Handel	2518	1703	2030	822
Marcello	2	8	26	8
Haydn	1750	1700	1298	1579
Mozart	2596	2734	2955	2874
Beethoven	736	772	695	933
Complete test corpus	23378	14575	14296	9447

16. Comparison of algorithms with respect to number of intervals spelt correctly: Number of intervals spelt incorrectly

1. When the lists of errors generated by the algorithms were examined, it was observed that, in some cases, many errors were the result of large segments of the music simply being transposed up or down by a diminished second.
2. In other words, a single incorrect interval between two notes resulted in a whole segment of notes following the incorrect interval being spelt incorrectly.
3. The algorithms were therefore also compared with respect to the number of *intervals* spelt correctly.
4. This table shows the number of intervals spelt incorrectly by each algorithm for each composer.
5. As you can see, Temperley's algorithm performed best followed by *ps13* and Longuet-Higgins algorithm which both made about 50% more errors. Cambouropoulos's algorithm performed worst making more than twice as many interval errors as Temperley's algorithm.

17. Comparison of algorithms with respect to number of intervals spelt correctly:

Percentage of intervals spelt correctly

	Cam	LH	ps13	Tem
Corelli	99.08%	99.71%	99.81%	99.96%
Vivaldi	99.08%	99.43%	99.13%	99.58%
Telemann	99.19%	99.50%	99.12%	99.77%
Bach	97.97%	99.07%	99.28%	99.67%
Handel	99.44%	99.62%	99.55%	99.82%
Marcello	99.93%	99.73%	99.12%	99.73%
Haydn	97.93%	97.99%	98.47%	98.13%
Mozart	98.49%	98.41%	98.28%	98.33%
Beethoven	98.49%	98.41%	98.57%	98.08%
Complete test corpus	98.65%	99.16%	99.17%	99.45%

17. Comparison of algorithms with respect to number of intervals spelt correctly: Percentage of intervals spelt correctly

1. This table shows the percentage of intervals spelt correctly by each algorithm for each composer.
2. As you can see, Temperley's algorithm correctly spelt 99.45% of the intervals correctly and *ps13*'s score differed from that of Longuet-Higgins's algorithm by only 0.01%.

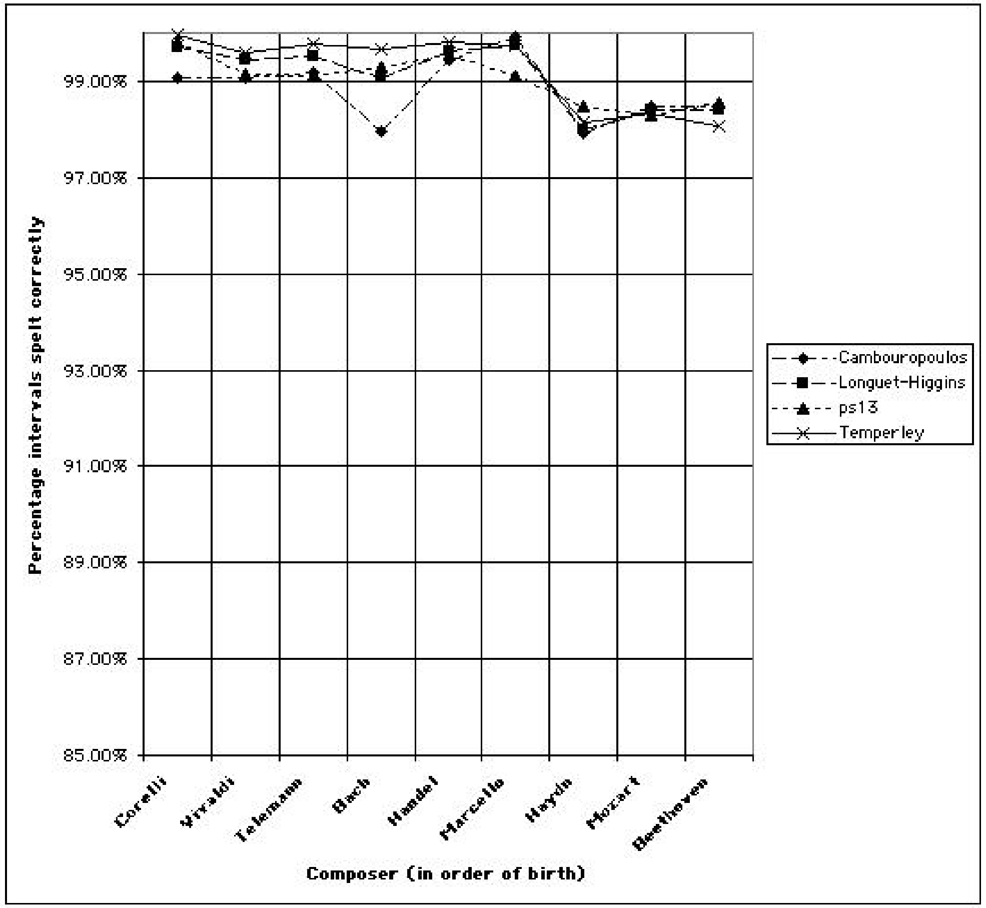
18. Comparison of algorithms with respect to number of intervals spelt correctly: Significance of differences between numbers of errors

Composer	Best	p-value	2nd best	p-value	3rd best	p-value	Worst
Corelli	Tem	< 0.0001	ps13	0.0068	LH	< 0.0001	Cam
Vivaldi	Tem	< 0.0001	LH	< 0.0001	ps13	0.1077	Cam
Telemann	Tem	< 0.0001	LH	< 0.0001	Cam	0.0029	ps13
Bach	Tem	< 0.0001	ps13	< 0.0001	LH	< 0.0001	Cam
Handel	Tem	< 0.0001	LH	< 0.0001	ps13	< 0.0001	Cam
Marcello	Cam	0.0577, 0.0577	LH,Tem	0.001, 0.002	ps13		
Haydn	ps13	< 0.0001	Tem	0.0028	LH	0.2416	Cam
Mozart	Cam	0.0584	LH	0.0143	Tem	0.2459	ps13
Beethoven	ps13	0.041	Cam	0.0252	LH	< 0.0001	Tem
Complete test corpus	Tem	< 0.0001	ps13	0.0637	LH	< 0.0001	Cam

18. Comparison of algorithms with respect to number of intervals spelt correctly: Significance of differences between numbers of errors

1. Again, the statistical significance of the differences between the performances of the algorithms were analysed using McNemar's test and the results are shown here.
2. The bottom line of this table reveals that, with respect to the number of intervals spelt correctly, Temperley's algorithm performs very significantly better on this test corpus than the other three algorithms.

19. Comparison of algorithms with respect to number of intervals spelt correctly:
Dependence of accuracy on musical style



19. Comparison of algorithms with respect to number of intervals spelt correctly: Dependence of accuracy on musical style

1. This graph shows the percentage of intervals spelt correctly by each algorithm for each composer, with the composers arranged along the horizontal axis in order of birth year.
2. From this graph it is clear that the algorithms of Longuet-Higgins and Temperley were noticeably better at spelling *intervals* correctly than they were at spelling *notes* correctly.
3. In particular, this graph shows that most of the note spelling errors made by Longuet-Higgins and Temperley's algorithms on the music of Haydn, Mozart and Beethoven were due to whole segments being spelt a diminished second away from the correct spelling.
4. Nonetheless, we can see that, even in terms of the number of intervals spelt correctly, all the algorithms perform worse on the classical music than on the baroque music.

20. Conclusions and further work

Notes	*ps13* (99.33%)	$>$	Camb (98.71%)	$>$	Temp (97.67%)	$\simeq$	LH (97.65%)
Intervals	Temp (99.45%)	$>$	*ps13* (99.17%)	$\simeq$	LH (99.16%)	$>$	Camb (98.65%)

- The algorithms of Temperley and Longuet-Higgins mis-spelt many more notes in the classical music than the other algorithms.
- Need to test these and other algorithms (e.g. Chew and Chen, 2003) on a stylistically more varied corpus.
- Could also build complete pitch-spelling algorithms from various different key-finding algorithms (e.g. Holtzmann, 1977; Krumhansl, 1990; Vos and van Geenen, 1996).

20. Conclusions and further work

1. To sum up, four pitch spelling algorithms were run on a large corpus of baroque and classical music.
2. When the algorithms were evaluated in terms of the number of notes they spelt correctly, it was found that my *ps13* algorithm performed best, correctly spelling 99.33% of the notes in the corpus correctly.
3. While all four algorithms performed well on the baroque music, it was found that the algorithms of Temperley and Longuet-Higgins performed noticeably less well than the other algorithms on the classical music in the corpus.
4. However, when the algorithms were evaluated in terms of the number of *intervals* spelt correctly, it was found that all the algorithms performed very well across all styles, with Temperley's algorithm performing best, correctly spelling 99.45% of the intervals in the corpus correctly.
5. It would be interesting to extend this study by testing these and other algorithms (e.g. Chew and Chen, 2003) on other corpora containing works in a wider variety of tonal styles including, e.g., romantic, impressionist, rock and jazz.
6. Krumhansl (1990, p. 79) claims that "once a key (or key region) has been determined, the correct spellings of the tones will be able to be determined in most cases".
7. *ps13* and the algorithms of Temperley and Longuet-Higgins all perform something a bit like key-finding as part of the pitch-spelling process. However, the algorithms give significantly different results when run on the same test corpora. This demonstrates that there are various plausible ways of using the key-structure of a passage to determine pitch names and it is not at all obvious which of these methods will give the best results.

8. Krumhansl's claim needs to be tested by building complete pitch spelling algorithms based on various key-finding algorithms (e.g. Holtzmann, 1977; Krumhansl, 1990; Vos and van Geenen, 1996) and comparing the performance of these algorithms with those that I've just described.

References

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

Cambouropoulos, E. (1998). *Towards a General Computational Theory of Musical Structure*. Ph.D. thesis, University of Edinburgh. Available online at `http://users.auth.gr/~emilios/englishpage/phd.html`.

Cambouropoulos, E. (2001). Automatic pitch spelling: From numbers to sharps and flats. In *VIII Brazilian Symposium on Computer Music (SBC&M 2001)*, Fortaleza, Brazil. Available online at `ftp://ftp.ai.univie.ac.at/papers/oefai-tr-2001-12.pdf`.

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

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

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, **10**(7), 1895–1924.

Holtzmann, S. R. (1977). A program for key determination. *Interface*, **6**, 26–56.

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

Longuet-Higgins, H. C. (1976). The perception of melodies. *Nature*, **263**(5579), 646–653.

Longuet-Higgins, H. C. (1987). 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.

Longuet-Higgins, H. C. (1993). The perception of melodies. In S. M. Schwanauer and D. A. Levitt, editors, *Machine Models of Music*, pages 471–495. M.I.T. Press, Cambridge, Mass.

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

Meredith, D., Lemström, K., and Wiggins, G. A. (2002). Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, **31**(4), 321–345. Draft available online at `http://www.titanmusic.com/papers/public/siajnmr_submit_2.pdf`.

Meredith, D. (2003). Method of computing the pitch names of notes in MIDI-like music representations. Patent filing submitted to UK Patent Office on 11 April 2003. Application number 0308456.3. Draft available online at `http://www.titanmusic.com/papers.html`.

Piston, W. (1978). *Harmony*. Victor Gollancz Ltd., London. Revised and expanded by Mark DeVoto.

Regener, E. (1973). *Pitch Notation and Equal Temperament: A Formal Study*. University of California Press, Berkeley, CA.

Temperley, D. (1997). An algorithm for harmonic analysis. *Music Perception*, **15**(1), 31–68.

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

Vos, P. G. and van Geenen, E. W. (1996). A parallel-processing key-finding model. *Music Perception*, **14**, 185–224.