

ENGRAVING ORIENTED JOINT ESTIMATION OF PITCH SPELLING AND LOCAL AND GLOBAL KEYS

Augustin Bouquillard

Université Paris Dauphine, PSL
augustinus.bouquillard@orange.fr

Florent Jacquemard

Inria Paris and Cedric/CNAM
florent.jacquemard@inria.fr

ABSTRACT

We revisit the problems of pitch spelling and tonality guessing with a new algorithm for their joint estimation from a MIDI file including information about the measure boundaries. Our algorithm does not only identify a global key but also local ones all along the analyzed piece. It uses Dynamic Programming techniques to search for an optimal spelling in term, roughly, of the number of accidental symbols that would be displayed in the engraved score. The evaluation of this number is coupled with an estimation of the global key and some local keys, one for each measure. Each of the three informations is used for the estimation of the other, in a multi-steps procedure.

An evaluation conducted on a monophonic and a piano dataset, comprising 216 464 notes in total, shows a high degree of accuracy, both for pitch spelling (99.5% on average on the Bach corpus and 98.2% on the whole dataset) and global key signature estimation (93.0% on average, 95.58% on the piano dataset).

Designed originally as a backend tool in a music transcription framework, this method should also be useful in other tasks related to music notation processing.

1. INTRODUCTION

In symbolic music representations, pitches are expressed in different ways. In their simplest form, in the MIDI standard, they are encoded as integers corresponding to a number of keys on a device. The representation of pitches is much more involved in Common Western Music Notation (CWN), where the denotation of each note depends on the musical context of occurrence: the tonality (key) of the piece, the voice-leading structure (ascending or descending melodic movements), the harmonic context...

If we reason modulo octaves, every pitch class, between 0 and 11 semitons, can be denoted in several ways, using a note name, in C, D, E, F, G, A, B, and an accidental mark in \flat , \sharp , \times , acting as a pitch class modifier. *Pitch-Spelling* (PS) is the problem of choosing appropriate names to denote some given MIDI pitch values in CWN.

In the tonal system, fixing a (global) key for a piece defines some default, privileged, names and accidentals. This

rule serves two important purposes in practice for the reader of a music score. On the one hand, since the default accidentals are not printed, the number of symbols displayed on the score is reduced and hence the readability is eased. On the other hand, the key signature immediately poses a tonal context for the piece, and the presence of other (non default) accidentals constitutes an indication of the tonal function of the notes, which provides insight into the composer's intention, in particular regarding local key changes. For instance, let us consider the last chord of the 56th measure of Mozart's c minor Sonata's 1st movement (Figure 1a), which is resolved in the next bar on the second inversion of an $E\flat$ Major triad, comprising a G natural in the first voice of the left hand. By spelling the highest note of the left hand in the chord of interest as $G\flat$ instead of $F\sharp$, Mozart chooses not to comply with the principle of selecting an ascending accidental when a voice goes up, and rather to express the harmonic function of dominant of the dominant in $E\flat$, which is the new tonality he is heading to at this moment. In the same movement, measure 125 (Figure 1b), a chord composed of the exact same notes is then spelled differently: an $F\sharp$ has replaced the $G\flat$, as the tonal context is shifting back to the main tone and the harmonic function has changed to dominant of the dominant in c minor this time. This constitutes one of the numerous examples (e.g., [1], Chopin's first Ballade or Tristan's chord) of one chord being spelled differently depending on its harmonic nature or tonal function and in this case regardless of the melodic movements of its voices, which shows that pitch spelling is indeed revealing in terms of creative intent and not only useful to the readability of a piece.

There is therefore a strong interdependency between the problems of *Pitch-Spelling*, and (local and global) *Key Estimation* (KE).

A large variety of PS algorithms have been proposed, which are able to guess the spelling of reference corpora with a high degree of accuracy. Several such algorithms have been designed according to musicological criteria, selecting note spellings based on the analysis of voice-leading, interval relationships and local keys [2, 3, 4], and a principle of parsimony (minimisation of the number of accidentals) [5]. Some procedures, also based on musicological intuitions, reduce PS to optimisation problems in appropriate data structures, e.g., the Euler lattice [6] or weighted oriented graphs [7]. In other approaches, datasets of music scores are used to train models such as HMM [8] or RNN [9] for PS. The system in the latter reference, esti-

Copyright: © 2024 Augustin Bouquillard and Florent Jacquemard. This is an open-access article distributed under the terms of the [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.



(a) Mozart, Sonata no 14 in c minor, mvt 1, measures 56-57.



(b) Mozart, Sonata no 14 in c minor, mvt 1, measures 125-126.

Figure 1. Tonal cues in chord spellings in Mozart’s Sonata no 14.

mates, in addition to PS, a key signature for the input piece, with a high accuracy.

In this paper, sharing with [9] the goal of joint Pitch Spelling and Key Estimation from MIDI data, we generalise the KE problem from global key signature to global and local keys. Our approach is algorithmic, following the very simple and intuitive idea to minimise the number of accidental symbols *as they would appear on the engraved score*. Similar criteria of parsimony are also found *e.g.*, in [5]. A difference is that we are applying this principle in a tonal context: assuming a key for the piece to spell, we count the accidentals for the possible spellings according to the *rules of engraving*, *i.e.*, considering that the accidentals defined by the key signature are not printed, and that a printed accidental holds for the embedding measure.

Our procedure works in two steps. Assuming that the MIDI input is divided into measures, we first estimate, for each key in a given set, the least number of printed accidentals for all possible spellings in all measures. We use this information in order to estimate a local key for each measure, and each possible global key. Then, in a second step, the above evaluation of the quality of spellings is refined by taking into account (in addition to the number of printed accidentals) the proximity of spellings to the evaluated local keys. The estimated global key is the one giving the best evaluation (cumulated for all measures), and the spelling chosen is the one computed for this global key.

The prior division into measures is crucial in our approach because of its role in the rules of engraving (cf Section 3.1). This assumption is not required in the above cited papers, either those following algorithmic or ML approaches. The algorithmic papers rather use a sliding window of parametric size. With that respect, our procedure applies to more restricted cases. However, this assumption makes sense when dealing with quantized MIDI data, in particular in the last step of a music transcription framework, after rhythm quantization. Also, the estimated global and local keys, which are somehow a side effect of our PS procedure, may be useful, as descriptors, in other tasks of processing of MIDI data for which metrical information is known. Finally, apart from the boundaries of measures, we do not need other information such as note durations or

voice separation (see the discussion on that point in Section 5).

To summarize our contributions, we propose an original and somewhat naive approach to two old problems, which combines them and obtains very good results on challenging datasets. In Section 2, we present the preliminary notions used to state the problems of PS and KE. The reader well versed into these problems may skip this section. We detail our procedure in Section 3 and present in Section 4 its evaluation on two datasets (one monophonic and one piano), for a total of 216 464 notes, which has given very good results.

2. PRELIMINARIES

We call *part* a polyphonic sequence of notes, organized in measures (bars). Typically, it shall correspond to one staff in CWN. Every note is given by values of pitch, onset and duration. The representation of these values is described in the two next subsections, before another subsection treating the subject of keys and signatures.

2.1 Pitch Representations

There are two classical alternative representations for the note *pitch*, distinguishing the input and output of a pitch-spelling algorithm:

- a *MIDI* value, in $0..128$, which is a number of semitones [10],
- a *spelling*, made of:
 - a note *name* in A, \dots, G ,
 - a symbol of *accidental*, amongst \natural (natural), \flat (flat), $\flat\flat$ (double flat), \sharp (sharp), \times (double sharp),
 - an octave number in $-2..9$.

The lowest MIDI value 0 corresponds to $C_{\flat-1}$ ($B\sharp-2$), and the highest one, 128, is $G\sharp9$ ($A\flat9$). The 88 keys of a piano correspond to the MIDI numbers 21 ($A_{\flat}0$) to 96 ($C_{\sharp}7$). A MIDI value modulo 12, is called *pitch class*. The pitch class of a note ν is denoted by $pc(\nu)$.

Every note name is associated a unique pitch class: 0 for C to 11 for B. An accidental symbol acts as a pitch class modifier: $\flat\flat$, \flat , \natural , \sharp , \times , respectively add -2, -1, 0, 1, and 2 to the pitch class of the note name component in a spelling. In the following, the symbol \natural is sometimes omitted, *i.e.*, written as a space. Altogether, with the octave component, this principle permits to associate a unique MIDI value to a given spelling. In the other direction, there are several alternative valid spellings for a given MIDI value, as summarized in Figure 2 for the 12 pitch classes.

For every pitch class, the 2 or 3 alternative spellings of Figure 2 have different names. Hence, for the purpose of finding a spelling for a pitch of MIDI value m , it is sufficient to choose one of the 2 or 3 possible names for m modulo 12. The corresponding accidental symbol and octave number can then be deduced from the name chosen.

pitch class	spelling ₁	spelling ₂	spelling ₃
0	D $\flat\flat$	C	B \sharp
1	D \flat	C \sharp	[B \times]
2	E $\flat\flat$	D	C \times
3	[F $\flat\flat$]	E \flat	D \sharp
4	F \flat	E	D \times
5	G $\flat\flat$	F	E \sharp
6	G \flat	F \sharp	[E \times]
7	A $\flat\flat$	G	F \times
8	A \flat	G \sharp	
9	B $\flat\flat$	A	G \times
10	[C $\flat\flat$]	B \flat	A \sharp
11	C \flat	B	A \times

Figure 2. Enharmonic spellings for each pitch class.



Figure 3. Bach, Fugue in A Major BWV864, measure 33, rh.

Example 2.1. Figure 3 presents the right-hand part in measure 33 of the Fugue in A Major BWV864 of J.S. Bach. The MIDI values of the two notes on the first beat of the measure are 66, with possible spellings either F \sharp 4 or G \flat 4, and 74, with possible spellings either D5, or C \times 5, or E \flat 5. Here the chosen spelling does not induce any accidental, as F \sharp is included in the key signature. An alternative spelling to the one on the score for these two notes could thus be G \flat , D but it would generate an added accidental on the G since the signature does not contain any G \flat ; hence we observe the importance of the key signature for pitch spelling.

2.2 Time information

In this paper, we assume that the *onset* and *duration* values of notes are rational numbers, expressed in fraction of a measure. The choice of a time unit is not really relevant in this work. The informations related to time that is important in our procedure are actually:

- an ordering and equality relation on onsets, for sorting the notes in input and detecting notes starting simultaneously,
- the number of measure to which a note belongs, *i.e.*, the detection of bar changes in the flow of input notes.

We call *grace-note* a note of theoretical duration zero. We use this term generically for a note which is part of an ornament (appoggiatura, gruppetto, mordent, trill *etc.*). Two notes are called *simultaneous* if they have the same onset and are not grace notes. This definition can correspond to notes involved in the same chord, or notes in different voices and starting simultaneously.

Example 2.2. For instance, the two first notes of Figure 3 do not really constitute a chord but they are *simultaneous* since they share the same onset: 0. The time signature for

the measure is 9/8, hence the two first notes F \sharp and D \flat both have a duration of $\frac{1}{9}$ bar, whereas the next semi-quaver F \sharp 5, starting at the onset $\frac{2}{9}$, has a duration of $\frac{1}{18}$.

2.3 Key Signature and Key

A Key Signature (KS) is an integral number k between -7 and 7, which indicates that by default, $|k|$ note names shall be altered by a sharp accidental symbol (when $0 < k \leq 7$) or a flat accidental (when $-7 \leq k < 0$). The names of the notes altered are defined according to the order of fifths: F \sharp , C \sharp , G \sharp , D \sharp , A \sharp , E \sharp , B \sharp for the sharps (k positive) and B \flat , E \flat , A \flat , D \flat , G \flat , C \flat , F \flat , for the flats (k negative). In a score, a KS indication is placed at the beginning of a part and will influence the display of the spelling of notes in every measure, as explained in Section 3.1. The KS can be changed during a part.

From the notational point of view (*i.e.*, for engraving), the choice of KS will change drastically the way the notes are displayed on the score, as it influences the number of accidental symbols printed, hence the readability of the score.

From a musical point of view, a KS together with a *mode*, defines a *global key* K for a piece, which is a central notion in tonal music. Indeed, the key identifies a diatonic scale, whose first note (amongst seven notes), called the *tonic* note, represents the main tonal focus of the piece.

In Figure 4, we describe 15 key signatures and the tonic of associated keys. Keys defined by the same KS and different modes are called *relative*.

Additionally to the global key of a piece (or part), some alternative *local keys* can be identified for extracts of the piece, and might diverge from the global key through *modulations*. In this work, we shall estimate a local key for each measure in a part, and this information is used for pitch spelling (Section 3.3).

We shall consider below, in our joint Pitch Spelling and Key Estimation procedure, the *major* mode, and three minor modes: *harmonic minor*, *melodic minor* (also called *ascending minor*) and *natural minor* (also called *descending minor* or *Aeolian*). The three minor modes differ by the alteration of some degrees in the scale, notably the leading tone (or subtonic in the case of the natural minor scale) and the submediant with accidentals not in the key signatures: seventh degree (raised) for *harmonic minor*, and sixth and seventh degrees (raised) for *melodic minor*.

Only the major and harmonic minor modes are used for global Key Estimation. The melodic and natural minor modes are only used for local Key Estimation, see Section 3.3 and Section 4.3 for a discussion on occurrences of these modes in pieces used for evaluation. We shall consider a measure of distance between keys defined by Gottfried Weber in [11] (see Appendix A.1), see also [12] for other use of Weber's table in the context of Key Estimation.

In theory, the list of key signatures can be extended on the right and on the left, respectively through double sharps and double flats. For instance, with $k = 8$ (G \sharp major), F is altered with a double sharps (F \times), with $k = 9$ (D \sharp major), F and C are altered with double sharps (F \times , C \times), *etc.* We do not consider the case of extended KS in this work, as they are very rarely found.

key signature	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
major mode	C \flat	G \flat	D \flat	A \flat	E \flat	B \flat	F	C	G	D	A	E	B	F \sharp	C \sharp
minor modes	A \flat	E \flat	B \flat	F	C	G	D	A	E	B	F \sharp	C \sharp	G \sharp	D \sharp	A \sharp
s_0 :															
C	b	b													
D	b	b	b	b											
E	b	b	b	b	b	b									
F	b														
G	b	b	b												
A	b	b	b	b	b										
B	b	b	b	b	b	b	b								

Figure 4. Key signatures and keys.

In both major and minor modes, the keys associated with key signatures respectively -7 and 5, -5 and 7, and -6 and 6 have tonics with the same pitch class, but different names. These keys are called *enharmonic*. They correspond to the 3 first and 3 last columns in Figure 4. Melodies written in either of two enharmonic keys cannot be distinguished by ear (in equal temperaments) as changing a key for its enharmonic preserves not only the intervals but also the pitch class of every note. Therefore, spelling in one or the other of enharmonic keys is essentially a matter of choice. Usually, the keys with KS 5 (5 sharps, e.g., B major) or KS -5 (5 flats, e.g., D \flat major) are preferred over their enharmonic equivalents with respectively KS -7 (e.g., C \flat major) and KS 7 (e.g., C \sharp major). But it is not always the case, for instance the Prelude and Fugue, BWV 848, of Bach’s Well-Tempered Clavier are in C \sharp major.

3. ALGORITHMS

We present in this section the procedure that we are using to estimate, in the same process, best spellings, a global key and one local key per measure, for a given sequence of input notes. Roughly, it works measure by measure, by comparing the number of accidentals in all spellings for all candidate global keys in a given set. This exhaustive comparison is done through dynamic programming techniques, based on the principle used to decide which accidentals must be printed or not in CWN (presented in next Section 3.1). The estimated local keys are used to refine the selection of spellings in each measure, in case of ties.

3.1 Notational Convention Modulo

In order to lighten the notations, and ease the readability, some accidental symbols are not printed in scores in CWN. Following a principle of parsimony, the notational conventions are roughly that: the accidentals in the key signature are omitted by default, and the other accidentals need not be repeated in the same measure. There is moreover a restriction to this rule, summarised as follows in [13]: *It applies only to the pitch at which it is written: each additional octave requires a further accidental.*

Let us present below a relaxed version of this convention, without the above restriction for octaves. Our approach amounts in some way to reasoning modulo 12, replacing the notion of pitch by the notion of pitch class. This application of the principle of parsimony is more suitable to the

problem of Pitch Spelling [5], which is more concerned in *counting* the accidentals than in *printing* them.

Formally, the definition of the relaxed convention is based on a notion of *spelling state*, or state for short. Such a state s is a mapping from the note names in $\{A, \dots, G\}$ into accidental symbols in $\{\flat, \sharp, \natural, \times\}$. Let us assume a key signature $k \in \{-7, \dots, 7\}$ and a measure containing a sequence ν_1, \dots, ν_p of p notes, enumerated by increasing onsets. An initial state s_0 for the measure is built from k as expected (see Figure 4): for $k = 0$, $s_0(n) = \natural$ for all name n , for $k = 1$, $s_0(F) = \sharp$ and $s_0(n) = \natural$ for all other name n , etc. For $1 \leq i \leq p$, the state s_i is computed by updating the previous state s_{i-1} as follows, where n and a are the name and accidental of ν_i :

- i. if $s_{i-1}(n) = a$, then $s_i = s_{i-1}$, and a is *not printed*,
- ii. otherwise, $s_i(n) = a$, $s_i(n') = s_{i-1}(n')$ for all $n' \neq n$, and a is *printed*.

Example 3.1. In Figure 3, for instance, at all onsets before $\frac{5}{9}$, the spelling state is composed of F \sharp , C \sharp , G \sharp with every other note of the scale being natural. On onset $\frac{5}{9}$ however, the state changes for the first time in the measure and now contains A \sharp instead of A \natural .

The next onset also induces a change in the state with G \natural being replaced by G \sharp , which is a note present in the ascending minor melodic mode of B. It is interesting to remark that Bach used it in a descending motion, therefore a pitch spelling process relying too much on motion direction between notes would have failed here.

The spelling state then stays the same until onset $\frac{15}{18}$ with the return of A \natural (belonging to the natural minor mode of B) and finally G \natural at onset $\frac{16}{18}$, hence the last states of the measure are the same as the one it started with.

3.2 Processing One Measure in a Key

Let K be a key with key signature $k \in \{-7 \dots 7\}$ and let $\bar{\nu} = \nu_1, \dots, \nu_\ell$ be a sequence of notes inside a measure, with known MIDI values and unknown spellings. In order to find the spellings for $\bar{\nu}$ with the least number of printed accidentals, according to the conventions of Section 3.1, it is fortunately not necessary to enumerate all the $O(3^\ell)$ possible spellings of $\bar{\nu}$. Indeed, for that purpose, it is sufficient to perform a shortest path search, using the state structure of Section 3.1. Before presenting the details of the search method, let us formulate an additional hypothesis, that is



Figure 5. Beethoven, Sonata 21 "Waldstein", measures 1-3, lh.

relevant musically and turned out to be important from a combinatoric point of view:

(h) *two simultaneous notes in the same pitch class must have the same name.*

The hypothesis (h) is not a strict notational convention, although counter examples are very rare. However, we do not assume the other direction (simultaneous notes with same name must be in the same pitch class), as it can occur fairly frequently in tonal music, at least from the end of the nineteenth century, for instance in a dominant ninth chord with an appoggiatura on the ninth (e.g., D F# A C F# in G major), much appreciated by Ravel, among others.

Example 3.2. Without the hypothesis (h) stated above, the presence of numerous chords inside of one measure made the complexity of PSE explode in cases such as the beginning of the Waldstein Sonata, displayed in Figure 5. The treatment of simultaneous notes allows the algorithm not to treat all possible spellings for each apparition of doubled notes inside a chord (in this example one of the C's in the repeated C major chord need not be treated once the other C's spelling has been determined).

In order to ensure the hypothesis (h), we consider another state variable $c \in \mathcal{C} = \{A, \dots, G\}^{\{0, \dots, 11\}}$ which is a partial mapping of pitch classes into note names. It is used to memorize the name associated to a pitch class when processing a subsequence of simultaneous notes in \bar{v} . The spelling of a note with pitch class p and note name n is called *compatible* with c iff $c(p)$, if defined, is equal to n .

Let $\mathcal{S} = \{\flat, \flat, \natural, \sharp, \times\}^{\{A, \dots, G\}}$ be the set of possible values for the state variable s defined in Section 3.1, $I = \{1 \leq i < \ell \mid \nu_i \text{ and } \nu_{i+1} \text{ are simultaneous}\}$, $\bar{I} = \{1, \dots, \ell\} \setminus I$, and let us consider the following set of configurations:

$$V = \{\langle s_0, 0 \rangle\} \cup \mathcal{S} \times \bar{I} \cup \mathcal{S} \times \mathcal{S} \times \mathcal{C} \times I.$$

The configuration $\langle s_0, 0 \rangle$ is initial, where s_0 is the initial state associated to K as in Section 3.1. In configurations of the form $\langle s, i \rangle \in \mathcal{S} \times \bar{I}$ we have one state s for a note index $i \in \bar{I}$. These configurations are dedicated to the processing of single notes. The other configurations, of the form $\langle s, t, c, i \rangle \in \mathcal{S} \times \mathcal{S} \times \mathcal{C} \times I$ contain additional information in t and c for processing a sub-sequence of simultaneous notes, ensuring in particular hypothesis (h).

We consider a set of transitions $E \subset V \times \mathbb{N} \times V$, containing the *weighted edges* of one of the following forms:

$$\langle s, i - 1 \rangle \xrightarrow[n, a]{w} \langle s', i \rangle \quad (1)$$

$$\langle s, i - 1 \rangle \xrightarrow[n, a]{w} \langle s', s, c', i \rangle \quad (2)$$

$$\langle s, t, c, i - 1 \rangle \xrightarrow[n, a]{w} \langle s', t, c', i \rangle \quad (3)$$

$$\langle s, t, c, i - 1 \rangle \xrightarrow[n, a]{w} \langle s', i \rangle \quad (4)$$

Every above case means that there exists a spelling of ν_i with name n and accidental a , which is moreover compatible with c in cases (3) and (4). In transitions (1)-(4), s' is the update of s as defined in Section 3.1 (cases i, ii). However, since the purpose of state s is no more score engraving like in Section 3.1, but the the search of a pitch spelling, we shall distinguish below, for the definition of the weight value w , the cases when the accidental a is *counted* from the cases when it is *not counted* (and not anymore *printed* or *not printed* like in Section 3.1).

The transition (1) processes the single note ν_i which is not simultaneous with ν_{i+1} . The condition for counting a is the same as the condition for *printing* in Section 3.1 (cases i, ii):

- if $s(n) = a$, then $s' = s$ and a is *not counted*,
- otherwise, $s'(n) = a$, $s'(n') = s(n')$ for all $n' \neq n$, and a is *counted*.

The transition (2) initiates the processing of a subsequence of two or more simultaneous notes, when ν_i is simultaneous with ν_{i+1} . It makes a copy of the current state s in the second component, and create $c' = \{\langle p, n \rangle\}$, with $p = pc(\nu_i)$. The accidental a is *counted* or *not*, and s updated into s' , under the same conditions as in (1).

The transition (3) performs one step of the processing of a subsequence of simultaneous notes, again when ν_i is simultaneous with ν_{i+1} . It propagates the frozen copy of state t (without modifying it) and updates c into c' as follows:

- if $c(p)$ is defined and equal to n , then $c' = c$, $s' = s$ and a is *not counted*,
- if $c(p)$ is undefined, then $c' = c \cup \{\langle p, n \rangle\}$, moreover, if $t(n) \neq a$, s is updated into s' with $\langle n, a \rangle$ and a is *counted*, otherwise, $s' = s$ and a is *not counted*,

The transition (4) terminates the processing of a subsequence of simultaneous notes, when ν_i is not simultaneous with ν_{i+1} . The conditions for counting a and updating s into s' are the same as in (1).

Finally, the weight value of each transition in (1)-(4) is:

1. $w = 0$ if a is *not counted*, or if K is in *harmonic minor* or *melodic minor* mode, and ν_i corresponds to an altered (raised) degree in the corresponding scale and a is the accidental for this degree in the scale.
2. otherwise, $w = 1$ if $a \in \{\flat, \natural, \sharp\}$, and $w = 2$ if $a \in \{\flat\flat, \times\}$.

The purpose of the above exception for minor modes is to help the estimation of keys with minor modes in the next section.

The oriented graph $\mathcal{G} = \langle V, E \rangle$ is acyclic. The vertex $\langle s_0, 0 \rangle$ is called the *source* of \mathcal{G} (it has no incoming edge) and the vertices of $\mathcal{S} \times \{\ell\}$ are called *targets* of \mathcal{G} (they have no outgoing edge).

Intuitively, a path starting from the source vertex $\langle s_0, 0 \rangle$ and ending with a target vertex of V , and following the edges of E , describes a spelling of the sequence of notes \bar{v} in input. The cumulated sum of the weight values w of edges involved in such a path amounts to a count of the accidental symbols printed.

The problem of searching for spellings of the notes of \bar{v} with a minimal number of printed accidentals reduces to the search in \mathcal{G} of a path with a minimal cumulated weight, from the source vertex into a target vertex of V . That can be done in time $O(|V| + |E|)$ with a greedy Viterbi algorithm [14], tagging the vertices of V with cumulated weight values. It proceeds in a forward way, starting from the source vertex, such that the tag of a vertex v is the weight of a minimal path leading to v .

For efficiency, the graph \mathcal{G} is built on-the-fly, starting from the source vertex $\langle s_0, 0 \rangle$, and adding vertices along with the computation of their tagging, following edges satisfying the above conditions. This strategy ensures a pruning of unnecessary search branches, making the approach efficient for a use on practical cases, as described in Section 4.

3.3 Processing One Part

We assume given a sequence of m measures containing notes with known MIDI values and unknown spellings. Let us consider a fixed array of keys $\bar{K} = K_1, \dots, K_n$. Our Pitch Spelling approach works by filling a table of dimensions $n \times m$ with best spellings for every key in \bar{K} and every measure, using variants of the algorithm of Section 3.2 for each cell. Then, one estimated global key is selected in \bar{K} , according to the table content, and the spellings in the corresponding row of the table can be applied to the input.

We proceed in several steps, building actually several tables, with several variants for the domain of weight values.

3.3.1 Step 1: Computation of a First Spelling Table.

In the first step, we compute a table T of dimension $n \times m$, such that the cell $T[i, j]$ contains the cumulated weight (in \mathbb{N}) of a best spelling, according to the algorithm of Section 3.2, with $K = K_i$ and \bar{v} are the notes of measure j .

3.3.2 Step 2: Estimation of Candidate Global Keys

We compute the sum of each row in T , and save in a list L of candidate global keys the keys of \bar{K} in *major* and *harmonic or natural minor* modes with a smallest sum (there may be ties).

3.3.3 Step 3: Computation of a Grid of Local Keys

In a second table G of dimensions $n \times m$, we shall store estimated local keys. The cell $G[i, j]$ shall contain the local

key estimation for the measure number j , assuming that K_i is the global key. The estimation is done column by column (*i.e.*, measure by measure).

For each measure number $0 \leq j \leq m$, we compute a ranking R_j^w of the keys in \bar{K} , according to the values (in \mathbb{N}) of $T[0, j], \dots, T[n, j]$, the smallest weight value being the best.

Then, for the same j and for each $0 \leq i \leq n$ we compute two other rankings $R_{j,i}^p$ and $R_{j,i}^g$ of \bar{K} , according to respectively the Weber distance to the previous estimated local key and the assumed global key (the smallest distance is the best). More precisely, the value associated to the key $K_{i'} \in \bar{K}$, with $0 \leq i' \leq n$, for computing its rank in $R_{j,i}^p$ is (see Appendix A.1 for the Weber table):

- the Weber distance between $K_{i'}$ and K_i if $j = 0$,
- the Weber distance between $K_{i'}$ and the previous local key in the same row $G[i, j - 1]$, if $j > 0$.

And the value associated to the key $K_{i'}$ when computing its rank in $R_{j,i}^g$ is the Weber distance between $K_{i'}$ and K_i . Finally, for j and i , we aggregate the three rankings R_j^w , $R_{j,i}^p$, and $R_{j,i}^g$ into a unique ranking $R_{j,i}$, by comparing, for each $0 \leq i' \leq n$, the mean of its three ranks from all the rankings – see [15] about this method. The estimated local key in $G[i, j]$ is the one with the best rank in $R_{j,i}$.

In practice, the computation of G can be restricted to the rows present in the list L extracted at *step 2*.

3.3.4 Step 4: Computation of a Second Spelling Table

In this final step, we compute a last table U of dimension $n \times m$ (or *length of L* $\times m$ since the computation of U can also be restricted to the rows present in the list L), using the same algorithm as in *step 1*, but with more involved weight values.

These weight values, used to replace the $w \in \mathbb{N}$ in Section 3.2, are tuples of integers, with the following components, for $U[i, j]$:

1. the value $w \in \mathbb{N}$ of Section 3.2,
2. the number of accidentals which do not occur in the scale associated to the estimated local key, for the notes \bar{v} of measure j and the global key $K = K_i$,
3. the number of spellings not in the chromatic harmonic scale [1] of the estimated local key,
4. the number of accidentals with a color different from the global key signature k_i of K_i (*i.e.*, \flat or \natural when $k_i > 0$ and \sharp or \times when $k_i < 0$),
5. the number of $C\flat$, or $B\sharp$, or $F\flat$, or $E\sharp$.

The components (2) and (3) will second the former weight value (1) in order to refine the search of best spelling thanks to the information gained with the local key estimation in *step 3*. The two last components (4) and (5) have been added for the purpose of tie-breaking.

We consider two orderings on the domain of above weight values:

W_{lex} : the lexicographic comparison of the tuplets with the 5 components (1)-(5),

W_+ : the lexicographic comparison of the 4-uplets with, for first component the sum of (1) and (2), and for next components (3), (4), and (5) respectively.

Using the same technique as in *step 2*, we extract from L one unique estimated global key $K_i \in \bar{K}$, using the above refined weight values, and apply the spellings found in the row i of the table U .

Example 3.3. A printout of the first table (*step 1*) computed when treating the Fugue BWV 864 (see the extract in Figure 3) can be found in Appendix A.2.

Here, 2 global candidates are obtained (*step 2*): A major (3 sharps) and F# minor (3 sharps) as their cumulated row costs are close and inferior to all the others. Only their corresponding rows will then be computed in the second table (*step 4*), this time taking into account local tonal analysis (performed during *step 3*) to refine the choice between candidate best paths. The second table is also displayed in Appendix A.2.

Only 1 global candidate remains at the end: F# minor (3 sharps). Even if the piece is in A major instead of F# minor, the process found the correct global key signature, which is what impacts the pitch spelling. This information, along with the local tonalities computed in each measure and used in the second table to choose between paths with the same number of accidents, was critical in finally selecting a spelling. Indeed, for this piece, the algorithm reached an accuracy of 100% compared to the groundtruth¹.

3.4 Rewriting Passing Notes

After choosing a spelling with the algorithm of Section 3.3, we apply local corrections by rewriting the passing notes, using a slight generalisation of the rewrite rules proposed by D. Meredith in the original PS13 Pitch-Spelling algorithm [4], step 2.

Every rule applies to a trigram of notes ν_0, ν_1, ν_2 , and rewrites the middle note ν_1 , by changing its name. In Figure 6, we present the rules for particular cases of notes. In general however, the rules are defined by patterns comparing the respective note names of ν_0, ν_1 , and ν_2 (without the accidentals), and the difference between their pitch (in number of semitones). For instance, in the left-hand-side $C C \flat C$ of the first rule *broderie down*, ν_0, ν_1 , and ν_2 all have the same note name C, the difference, in semitones, between ν_0 and ν_1 is -1 and the difference between ν_1 and ν_2 is $+1$. This rule rewrites the middle $C \flat$ (ν_1) into B.

The rewrite rules are applied from left to right to the sequence spelled notes. Note that at each rewrite step, at most one rule can be applied.

3.5 Deterministic Variant

We propose a variant of the algorithm presented in Sections 3.2 and 3.3, which is more efficient but less exhaustive. This algorithm, called PS13b (as in "PS13 with bar

¹ See Appendices A.2 and A.3 for more details on the processing of Bach's Fugue BWV 864 (Example 3.3).

broderie down	$C C \flat C \rightarrow C B C$
broderie up	$C C \sharp C \rightarrow C D \flat C$
descending ₁₁	$C C \flat A \rightarrow C B A$
descending ₁₂	$C C \flat \flat A \flat \rightarrow C B \flat A \flat$
descending ₂₁	$C A \sharp A \rightarrow C B \flat A$
descending ₂₂	$C A \sharp A \flat \rightarrow C B \flat A \flat$
ascending ₁₁	$A A \sharp C \rightarrow A B \flat C$
ascending ₁₂	$A \flat A \sharp C \rightarrow A \flat B \flat C$
ascending ₂₁	$A C \flat C \rightarrow A B C$
ascending ₂₂	$A C \flat C \sharp \rightarrow A B C \sharp$

Figure 6. Rewrite rules for passing notes (particular cases).

info"), is very similar to PS13 [4], except that it uses the information on measures, which is assumed available in this paper but not in [4], in order to estimate global and local keys. In [4], that estimation is done (implicitly) by counting the number of occurrences of the (assumed) tonic note in a window whose optimal size was evaluated manually.

In the algorithm PS13b, the choice of the spelling with name n and accidental a for the input note ν_i (Section 3.2, transition rules (1)-(4)), is forced to the (unique) spelling in the chromatic harmonic scale of the current key K [1]. Hence, the transitions are deterministic, and there is no need to search for best spelling in a measure because there is only one. The rest of the algorithm works as described in Section 3.3. The complexity of the table construction in this case is $O(n \times p)$ where p is the total number of notes in input and n is the number of keys considered. This complexity is significantly better than the one of the exhaustive algorithm in Sections 3.2 and 3.3. In counterpart, some potentially correct spellings will be missed (see Section 4.3).

4. EVALUATION

4.1 Implementation

The algorithms PSE (of Section 3.3) and PS13b (of Section 3.5) have been implemented² in C++20. This language was chosen for the sake of efficiency and for integration into larger systems. The implementation is object oriented, with general classes for pitches, keys, *etc.*, and data structures specific to the algorithm, such as states, bags of best paths and tables.

The input must be provided by a note enumerator, associating to each natural number a midi pitch, a bar number, and a flag of simultaneity (with the next note). Therefore, our algorithm can be integrated in a larger project. This has been done for a MIDI-to-score transcription framework, where the timings (in particular the bar boundaries) are computed before pitch spelling.

A Python binding, based on pybind11 [16], was also written and used for evaluation. It offers calls (in Python) to the methods of the C++ implementation, for the various procedures and steps presented in Section 3.

For the evaluation, we used the Music21 toolkit [17], in association with the above Python binding. Music21

² The C++ code as well as the Python evaluation scripts are publicly accessible at <https://gitlab.inria.fr/pse/pse>.

parses the MusicXML files in the evaluation datasets (see Section 4.2) and extracts for each note the information needed by the algorithm:

- the MIDI key value,
- the number of the measure the note belongs to,
- a flag telling whether the note is simultaneous with the next one.

This information is fed to the PS procedure, and the spellings computed are compared to the ones in the original scores. Moreover, some scores are produced in output that highlight the errors of our procedure with colour codes, and mark each measure with the estimated local key³.

4.2 Datasets

Two main datasets were used for performance evaluation: a monophonic (complex) one, originated from the Lamarque-Goudard rhythm textbook [18] *D'un Rythme à l'Autre*, and the ASAP piano dataset [19].

Performances of both algorithms described in Section 3 (PSE and PS13b) were assessed on the integrality of the Lamarque-Goudard dataset, containing 250 excerpts, as MusicXML files, from pieces of extremely various styles, from Bach and Scarlatti to Wolf, Duparc, Debussy, Ibert...

Evaluation was also executed on 5 separate corpora from the 222 pieces of the ASAP piano dataset, also in MusicXML format. All Bach preludes and fugues from the Well Tempered Clavier present in ASAP were used, except Preludes BWV 856 and 873 for technical reasons. All sonata movements by Mozart and Beethoven included in ASAP were also tested, as well as the K 475 Fantaisie by Mozart. Every one of the 13 Chopin Etudes contained in ASAP, from both opus 10 and 25, was used, as well as the 8 Rachmaninov preludes present, from both opus 23 and 32. The cumulated total of notes spelled by our two algorithms for this evaluation reaches a value of 216 464.

4.3 Results and Discussion

Experimentations were conducted for several combinations of the weight domains in $\{W_{lex}, W_+\}$, with the best results obtained when using the weights of W_+ . The execution time is about 1.68s on average per piece of the evaluation corpus (subset of Bach WTC), with the exhaustive algorithm PSE presented in Section 3.3, whereas it is only 0.04s on average per piece with the deterministic variant PS13b presented in Section 3.5, with results less accurate by more than 1%.

A summary of the evaluation results is presented in Table 1. The detailed results are also accessible online³. They are organised by corpus, each comprising a folder for each of the algorithm PSE and PS13b. Results for alternative versions of the algorithms corresponding to different ways of combining weights to compute costs (either lexicographically or additively) are also included. Every folder contains a table summarizing the results obtained

³ See <https://github.com/florento/PSEval/> for the evaluation results, including annotated scores and tabular summaries.

on all the pieces in the considered corpus. In addition to the tables, folders relating to the Bach, Beethoven and Lamarque-Goudard corpora include the annotated MusicXML scores of the treated pieces, where the spelling errors are annotated with color codes, and green notes indicate an initial error corrected by the final rewriting. The annotations also include the global key estimation, and local key estimations for each measure. The score obtained after the execution of PSE on Bach's Fugue BWV 893 in b minor is included in Appendix A.3 as an example.

Regarding global and local tonality estimation, our algorithm achieves very good results when we compare key signatures together but tends to prefer minor tones to their major relatives. This can be explained by the large number of notes a minor tonality possesses in our acceptance, as we accept spelling from both harmonic and natural minor modes, as well as the ascendant melodic one. The only error of global tone estimation on our whole Well-Tempered Clavier dataset is directly due to this tendency: the presence of natural B's in the BWV 870 prelude (C major of book 2), in a piece where flat Bs are also numerous due to modulations to F major, D minor *etc.*, did not prevent our algorithm from estimating the piece as written in D minor, because these natural B's were interpreted as part of the ascendant melodic minor mode of D, instead of indicators of a C major context.

About enharmonic tones, which are absolutely impossible to distinguish when only pitches and durations of notes are given, if the algorithm only proposed the correct global tone or its enharmonic counterpart among its global candidates at the end of the first pass, and if its final estimated global key is enharmonic to the correct one, then the piece is renamed in the enharmonic rival global tone and errors are computed on this version. This way of treating that issue is in accordance with the definition of a well spelled piece by [6], also shared with [4], and [2], relying on correctness of the intervals of the piece.

4.4 Comparison With Other Systems

On the task of global tonality guessing (KE), we have compared our algorithms' performances to the ones obtained with the Krumhansl-Schmuckler (KS) model for key determination, as implemented in the Music21 Python library [17]. This famous key-finding algorithm computes for every major and minor tonality a correlation coefficient between profile values of the tested key and total durations of their corresponding pitch class in the musical piece considered. It then chooses the best tonality according to the calculated correlation coefficients. It is interesting to note that our algorithms only need to know measure delimitations and not note durations whereas KS uses note durations and does not care about measures. On the whole corpus (both ASAP and Lamarque-Goudard) we attain a 93% correctness of key signature determination on average, while KS obtains a 75% accuracy in total.

The results for global tonality guessing (KE) on the Lamarque-Goudard (LG) dataset are rather low, in comparison to ASAP. The LG corpus, extracted from a rhythm textbook, consists of 250 short excerpts of longer pieces.

Table 1. Accuracy results of both pitch spelling algorithms on pieces from widely different styles and correctness of their global key signature estimation, compared to Krumhansl-Schmuckler algorithm’s performances (Music 21 implementation).

	number of spelled notes	pitch spelling PSE	pitch spelling PS13b	key estimation PSE	key estimation PS13b	key estimation Krumhansl-Schmuckler
Bach WTC ASAP corpus	55530	99.50%	98.27%	99.09%	98.29%	87.27%
5 movements from Mozart Sonatas present in ASAP	10043	99.11%	97.30%	100%	100%	80%
Fantaisie K. 745 plus 5 movements from Mozart Sonatas	13830	97.65%	95.97%	80%	80%	60%
33 movements from Beethoven Sonatas	87292	97.64%	95.65%	92.32%	95.71%	66.15%
13 Etudes by Chopin	25103	96.71%	96.03 %	96.15%	96.15 %	84.62%
4 Rachmaninov Preludes	7022	98.76%	97.49%	100%	100%	100%
Lamarque-Goudard	27687	98.46%	98.23%	76.90%	74.30%	50.60%

The pitches and duration of notes appearing in extracts are not necessarily representative of the whole pieces, which explains why KS performs rather poorly on it, since its correlation coefficients with tonal profiles become erroneous. Our algorithms, mainly relying on accidentals number minimization to infer the tonality, therefore prove significantly more robust when it comes to shorter extracts.

We do not provide a comparison table on Pitch Spelling results for several reasons. First, we assume given measure information, unlike the algorithms PS13 [4], CIV [8] or PKSpell [9]; to this respect, a comparison would not be fair. Second, most of the former evaluations of pitch spelling algorithms used as a benchmark the Musedata dataset proposed by D. Meredith [4]. We could not evaluate our algorithms on this dataset because it does not include the measure boundary information required by our procedures. Since note durations are included in Musedata, it would be possible to conduct an evaluation on this dataset by manually providing a time signature for each of its 216 pieces. Nevertheless, with success rates (for PS) of 99.41% with PS13 [4], 99.82% with CIV [8] and 99.87% with PKSpell [9], the remaining room for improvements on this benchmark is rather marginal, and we preferred to focus on larger datasets like ASAP to evaluate and improve our algorithms.

The recent system PKSpell [9], for which it is reported a 0.13% error rate on MuseData (the best results so far), has also been evaluated on 33 pieces of the challenging piano dataset ASAP [19]. It shows on this dataset an accuracy of 96.50% for the pitch spelling task and 90.30% for key signature estimation. Since the identities of the 33 pieces for evaluation are not disclosed in [9], it is not feasible for this paper to report performances on the exact same pieces. However, with an accuracy of 98.19% on average for pitch spelling on 110 pieces (by 5 different composers ranging from Bach to Rachmaninov) from the same ASAP dataset, as reported on Table 1, and 95.58% for global key signature estimation, it is likely that the proposed PSE algorithm

(and PS13b) should at least have similar performances to PKSpell, if not better.

5. CONCLUSION

We have presented two algorithms, PSE and PS13b for joint pitch spelling and estimation of global and local keys, from MIDI data including information on measure boundaries. Originally thought to be integrated in a transcription framework, these procedures could also be used in various tasks of music notation processing. Since PS13b has proven to be very efficient, it could also be used for displaying music notation from MIDI data in real-time.

The evaluation on challenging datasets has shown robust results both for pitch spelling and key signature estimation. Regarding the estimation of keys, there is currently a bias towards minor tonalities which are often preferred to their major relative tones that should be corrected. However it does not impact the chosen key signature.

Several directions can be explored in order to improve the current approach, such as a refinement of the weight domain for taking into account note durations and metric weight (strong or weak beats) when computing the best path in a given measure. Moreover, in order to improve the accuracy of the tonal analysis, some subtler musical criteria could be implemented, such as cadence detection or chord classification as well as a process to detect justified key signature changes.

As outlined in Section 2.2, the only parameters related to durations that we are considering in our algorithms are the division into measures and knowledge about the simultaneity of notes (which in our case only depends on their onsets). The results of our experiments are already solid, although the seemingly important parameter of *individual note durations* is ignored. We shall try to include this parameter to see how it can include the results, however, it is not clear to us what weighting to give to durations in

the search of best spellings presented in Section 3.2, while keeping our model independent of the corpora considered.

Another way to extend our approach to other genres, perhaps less tonal, would be to integrate new *modes* into the computation of the PS table. For instance, one may consider the integration of jazz modes (*e.g.*, Ionian, Dorian *etc*) in order to tackle the problem of pitch spelling for jazz, which has not been studied at lot in the literature. This could be of interest in particular for the notation of jazz soli, improvisations, and bass lines for instance.

6. REFERENCES

- [1] J. Nagel, “The chromatic modal scale: Proper spelling for tonal voice-leading,” *JOMAR Press*, 2007.
- [2] D. Temperley, *The cognition of basic musical structures*. MIT press, 2004.
- [3] E. Chew and Y.-C. Chen, “Real-time pitch spelling using the spiral array,” *Computer Music Journal*, vol. 29, no. 2, pp. 61–76, 2005.
- [4] D. Meredith, “The PS13 pitch spelling algorithm,” *Journal of New Music Research*, vol. 35, no. 2, pp. 121–159, 2006.
- [5] E. Cambouropoulos, “Pitch spelling: A computational model,” *Music Perception*, vol. 20, no. 4, pp. 411–429, 2003.
- [6] A. K. Honingh, “Compactness in the Euler-lattice: A parsimonious pitch spelling model,” *Musicae Scientiae*, vol. 13, no. 1, pp. 117–138, 2009.
- [7] B. Wetherfield, “The minimum cut pitch spelling algorithm: Simplifications and developments,” in *TENOR*, 2020.
- [8] G. Teodoru and C. Raphael, “Pitch spelling with conditionally independent voices,” in *Proc. conf. of the Int. Society for Music Information Retrieval (ISMIR)*, 2007, pp. 201–206.
- [9] F. Foscarin, N. Audebert, and R. Fournier-S’Niehotta, “PKSpell: Data-driven pitch spelling and key signature estimation,” in *Proc. 22nd conf. of the Int. Society for Music Information Retrieval (ISMIR)*, 2021.
- [10] *Official MIDI Specifications*, MIDI Association, 2023. [Online]. Available: <https://www.midi.org/specifications/midi-2-0-specifications>
- [11] G. Weber, *Versucht einer geordneten Theorie der Tonsetzkunst*. B. Schott’s Söhne, 1818.
- [12] L. Feisthauer, L. Bigo, M. Giraud, and F. Levé, “Estimating keys and modulations in musical pieces,” in *Sound and Music Computing Conference (SMC)*, 2020.
- [13] E. Gould, *Behind Bars: The definitive guide to music notation*. Faber Music, 2011.
- [14] L. Huang, “Advanced dynamic programming in semiring and hypergraph frameworks,” in *COLING*, 2008.
- [15] J. Demšar, “Statistical comparisons of classifiers over multiple data sets,” *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
- [16] J. Wenzel. pybind11 – Seamless operability between C++11 and Python. [Online]. Available: <https://pybind11.readthedocs.io>
- [17] M. S. Cuthbert and C. Ariza, “music21: A toolkit for computer-aided musicology and symbolic music data,” in *Proc. conf. of the Int. Society for Music Information Retrieval (ISMIR)*, 2010.
- [18] E. Lamarque and M.-J. Goudard, *D’un Rythme à l’autre*. Lemoine, 1997, vol. 1-4.
- [19] F. Foscarin, A. Mcleod, P. Rigaux, F. Jacquemard, and M. Sakai, “ASAP: A dataset of aligned scores and performances for piano transcription,” in *Proc. conf. of the Int. Society for Music Information Retrieval (ISMIR)*, 2020, pp. 534–541.

A. APPENDIX

In this appendix, we present some details about the procedure (Section A.1) and two samples of evaluation results, for illustration. The complete evaluation results may be found at <https://github.com/florento/PSEval/>.

A.1 Table of Weber

The table of relationship of keys defined in [11], and used in Section 3.3, step 3, is displayed below. Keys in major mode are uppercase, keys in minor mode are lowercase.

KS	C ^b	G ^b	D ^b	A ^b	E ^b	B ^b	F	C	G	D	A	E	B	F [#]	C [#]	G [#]	D [#]	A [#]	E	B	F	C	G	D	A	E	B	F [#]	C [#]	G [#]	D [#]	A [#]																			
-7	0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	11																			
-6	1	0	1	2	3	4	5	6	7	8	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	
-5	2	1	0	1	2	3	4	5	6	7	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10		
-4	3	2	1	0	1	2	3	4	5	6	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10			
-3	4	3	2	1	0	1	2	3	4	5	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10				
-2	5	4	3	2	1	0	1	2	3	4	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10					
-1	6	5	4	3	2	1	0	1	2	3	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10						
0	7	6	5	4	3	2	1	0	1	2	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10							
1	8	7	6	5	4	3	2	1	0	1	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10								
2	9	8	7	6	5	4	3	2	1	0	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10									
3	10	9	8	7	6	5	4	3	2	1	0	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10								
4	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	
5	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10		
6	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10			
7	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10				
-7	0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10
-6	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	
-5	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10		
-4	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10			
-3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10				
-2	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10					
-1	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10						
0	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10							
1	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10								
2	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10									
3	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10										
4	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	
5	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10		
6	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10			
7	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10				

A.2 Execution Tables for the Fugue BWV 864

We display below the verbatim of the tables computed when processing the Fugue BWV 864, see Example 3.3.

First table computed for processing the Fugue BWV 864 (Example 3.3):

spelling 813 notes

PSE: first table

Row Costs:

Gbmajor	(6b)	cost	accid=199	dist=0	chromarm=188	color=10	cflat=25
Dbmajor	(5b)	cost	accid=251	dist=0	chromarm=237	color=19	cflat=46
Abmajor	(4b)	cost	accid=291	dist=0	chromarm=280	color=35	cflat=44
Ebmajor	(3b)	cost	accid=279	dist=0	chromarm=274	color=123	cflat=24
Bbmajor	(2b)	cost	accid=252	dist=0	chromarm=258	color=246	cflat=8
Fmajor	(1b)	cost	accid=219	dist=0	chromarm=222	color=274	cflat=0
Cmajor	(0)	cost	accid=169	dist=0	chromarm=175	color=344	cflat=0
Gmajor	(1#)	cost	accid=123	dist=0	chromarm=130	color=3	cflat=4
Dmajor	(2#)	cost	accid=71	dist=0	chromarm=78	color=3	cflat=4
Amajor	(3#)	cost	accid=38	dist=0	chromarm=43	color=2	cflat=4
Emajor	(4#)	cost	accid=69	dist=0	chromarm=81	color=1	cflat=4
Bmajor	(5#)	cost	accid=113	dist=0	chromarm=124	color=1	cflat=4
F#major	(6#)	cost	accid=154	dist=0	chromarm=165	color=0	cflat=0
Ebminor	(6b)	cost	accid=179	dist=0	chromarm=205	color=23	cflat=19
Bbminor	(5b)	cost	accid=220	dist=0	chromarm=257	color=30	cflat=34
Fminor	(4b)	cost	accid=255	dist=0	chromarm=283	color=76	cflat=29
Cminor	(3b)	cost	accid=229	dist=0	chromarm=279	color=159	cflat=12
Gminor	(2b)	cost	accid=204	dist=0	chromarm=262	color=293	cflat=4
Dminor	(1b)	cost	accid=167	dist=0	chromarm=222	color=293	cflat=0
Aminor	(0)	cost	accid=130	dist=0	chromarm=175	color=344	cflat=0
Eminor	(1#)	cost	accid=110	dist=0	chromarm=130	color=3	cflat=4
Bminor	(2#)	cost	accid=67	dist=0	chromarm=78	color=3	cflat=4
F#minor	(3#)	cost	accid=33	dist=0	chromarm=43	color=2	cflat=0
C#minor	(4#)	cost	accid=69	dist=0	chromarm=81	color=1	cflat=4
G#minor	(5#)	cost	accid=111	dist=0	chromarm=124	color=1	cflat=5
D#minor	(6#)	cost	accid=145	dist=0	chromarm=165	color=0	cflat=0

Second table computed for processing the Fugue BWV 864 (Example 3.3):

Row Costs:

Amajor	(3 sharps)	cost	accid=38	dist=35	chromarm=3	color=4	cflat=1
F#minor	(3 sharps)	cost	accid=33	dist=21	chromarm=0	color=0	cflat=0

A.3 Annotated Score of the Fugue BWV 893

The Fugue BWV 864, in the previous section and Example 3.3, was spelled with PSE with an accuracy 100%, see

https://github.com/florento/PSEval/blob/941d7e10731fefdc28cdac1c53a58cd28d132f2/Results_ASAP/Bach/PSE/BWV864_Fugue.musicxml

Each time that one our algorithms makes some spelling errors, our evaluation script produce a copy of the original (musicXML) score with the mistakes annotated. For instance, the four errors in the spelling obtained with PSE for the Fugue BWV 893 from the ASAP dataset are highlighted in red in the following score:

https://github.com/florento/PSEval/blob/941d7e10731fefdc28cdac1c53a58cd28d132f2/Results_ASAP/Bach/PSE/BWV893_Fugue.musicxml

or, for a PDF version:

https://github.com/florento/PSEval/blob/941d7e10731fefdc28cdac1c53a58cd28d132f2/Results_ASAP/Bach/PSE/BWV893_Fugue.pdf

All scores processed with our method for the evaluation presented in Section 4, for the two datasets of Section 4.2 (see Table 1 for the list), with annotations as well as tables of results, are available at

<https://github.com/florento/PSEval/>