



# A Graphical Theory of Musical Pitch Spelling

## Link

<http://nrs.harvard.edu/urn-3:HUL.InstRepos:38779539>

## Terms of use

This article was downloaded from Harvard University's DASH repository, and is made available under the terms and conditions applicable to Other Posted Material (LAA), as set forth at

<https://harvardwiki.atlassian.net/wiki/external/NGY5NDE4ZjgzNTc5NDQzMGIzZWZhMGFIOWI2M2EwYTg>

## Accessibility

<https://accessibility.huit.harvard.edu/digital-accessibility-policy>

## Share Your Story

The Harvard community has made this article openly available.  
Please share how this access benefits you. [Submit a story](#)

## Acknowledgements

Throughout this process, David Parkes has provided the perfect alchemy of everything one would want from an advisor – an open mind, an attentive ear, insightful commentary. It has been an honor and a privilege to have the opportunity to discuss these ideas with him.

I am grateful to James Bean, who first asked me the innocent-seeming question, “what’s the best way to spell eighth tones?”, and who has been a motivating figure for my wanting to pursue this topic and put pen to paper.

I would like to thank Hridesh Kedia for all the well-timed words of encouragement and for reassuring me in my tremulous steps through academic writing.

I am indebted to my family still standing, and my mother in loving memory, for getting me this far, and with any luck, a good deal further.

Finally (not to mention *initially* and *continuously*) I would be nowhere without Laila Smith’s unwavering belief.

# Contents

|                                          |           |
|------------------------------------------|-----------|
| <b>Abstract</b>                          | <b>i</b>  |
| <b>Acknowledgements</b>                  | <b>ii</b> |
| <b>List of Definitions</b>               | <b>vi</b> |
| <b>1 Prelude</b>                         | <b>1</b>  |
| 1.1 The Pitch-Spelling Problem . . . . . | 1         |
| 1.1.1 Evaluating Spellings . . . . .     | 6         |
| 1.2 Responding to Related Work . . . . . | 9         |
| 1.2.1 Parsimony . . . . .                | 9         |
| 1.2.2 Accuracy . . . . .                 | 9         |
| 1.2.3 Generalizability . . . . .         | 10        |
| 1.2.4 Search Spaces . . . . .            | 11        |
| 1.2.5 Intervals . . . . .                | 12        |
| 1.2.6 Cognitive Plausibility . . . . .   | 12        |
| 1.2.7 Windowing . . . . .                | 14        |
| 1.3 The Minimum Cut Problem . . . . .    | 14        |
| 1.3.1 Maximum Flow . . . . .             | 15        |
| 1.4 Approach and Outline . . . . .       | 16        |
| 1.5 Contributions . . . . .              | 17        |
| <b>2 Graphical Representation</b>        | <b>19</b> |
| 2.1 Parsimony Pivot . . . . .            | 19        |

|          |                                              |           |
|----------|----------------------------------------------|-----------|
| 2.2      | Segmentation . . . . .                       | 21        |
| 2.3      | Closeness . . . . .                          | 24        |
| 2.3.1    | Events . . . . .                             | 24        |
| 2.3.2    | Notes . . . . .                              | 27        |
| 2.4      | Flow Network . . . . .                       | 28        |
| 2.4.1    | Nodes . . . . .                              | 28        |
| 2.4.2    | Arcs . . . . .                               | 29        |
| <b>3</b> | <b>Forward Problem</b>                       | <b>33</b> |
| 3.1      | The Minimum Cut Problem Revisited . . . . .  | 33        |
| 3.2      | Interpreting the Linear Program . . . . .    | 37        |
| 3.2.1    | Encoding Spellings . . . . .                 | 37        |
| 3.2.2    | Checking Validity . . . . .                  | 38        |
| 3.2.3    | Specifying Sigma . . . . .                   | 39        |
| 3.2.4    | Dual Interpretation . . . . .                | 42        |
| <b>4</b> | <b>Inverse Problem</b>                       | <b>43</b> |
| 4.1      | The Maximum Flow Problem Revisited . . . . . | 43        |
| 4.2      | Characterizing Omega . . . . .               | 46        |
| 4.3      | Exact Inverse . . . . .                      | 50        |
| 4.3.1    | Exact Inverse Theorem . . . . .              | 52        |
| 4.4      | Empirical Robustness . . . . .               | 54        |
| 4.4.1    | Approximate Inverse Theorem . . . . .        | 58        |
| <b>5</b> | <b>The Microtonal Context</b>                | <b>60</b> |
| 5.1      | Quarter Tones . . . . .                      | 60        |
| 5.1.1    | Constructing Omega-sub-2 . . . . .           | 62        |
| 5.1.2    | Constructing Sigma-sub-2 . . . . .           | 63        |
| 5.2      | Eighth Tones . . . . .                       | 65        |
| 5.2.1    | Flow Approach . . . . .                      | 66        |
| 5.2.2    | LP Approach . . . . .                        | 67        |
| 5.2.3    | Hybrid Approach . . . . .                    | 70        |

*CONTENTS*

v

**6 Coda**

**71**

**Glossary: Musical Terms**

**73**

**Bibliography**

**75**

# List of Definitions

|     |                                                     |    |
|-----|-----------------------------------------------------|----|
| 2.1 | Definition (Parsimony Pivot) . . . . .              | 19 |
| 2.2 | Definition (Time-interval chain) . . . . .          | 21 |
| 2.3 | Definition (Event chain partitioning) . . . . .     | 22 |
| 2.4 | Definition (Entry-wise multiplication) . . . . .    | 24 |
| 2.5 | Definition (Matrix pullback) . . . . .              | 24 |
| 2.6 | Definition (Pitch class function) . . . . .         | 28 |
| 2.7 | Definition (Coding and decoding) . . . . .          | 28 |
| 3.1 | Definition (MAINLP linear program) . . . . .        | 36 |
| 3.2 | Definition (Spell-check) . . . . .                  | 38 |
| 3.3 | Definition (Binding to spellings) . . . . .         | 39 |
| 4.1 | Definition (Lifted $s$ - $t$ cut) . . . . .         | 50 |
| 4.2 | Definition (EXACTINVERSE linear program) . . . . .  | 51 |
| 4.3 | Definition (Consistent Corpus) . . . . .            | 51 |
| 4.4 | Definition (Approximate Consistency) . . . . .      | 54 |
| 4.5 | Definition (APPROXINVERSE linear program) . . . . . | 56 |
| 5.1 | Definition (MAINLP linear program) . . . . .        | 67 |
| 5.2 | Definition (UPDOWNLP linear program) . . . . .      | 68 |

# Chapter 1

## Prelude

### 1.1 The Pitch-Spelling Problem

Let us dive into a musical example. The three passages of music depicted in Figure 1.1 have the same sound, but as can be seen to even a musically untrained eye, they look different. Yet most formally trained musicians will have a preference for the passage depicted in Figure 1.1a. What



Figure 1.1: Different spellings of the same passage of music.

makes these notated passages different is the way they are *spelled*. The pitch-spelling problem is

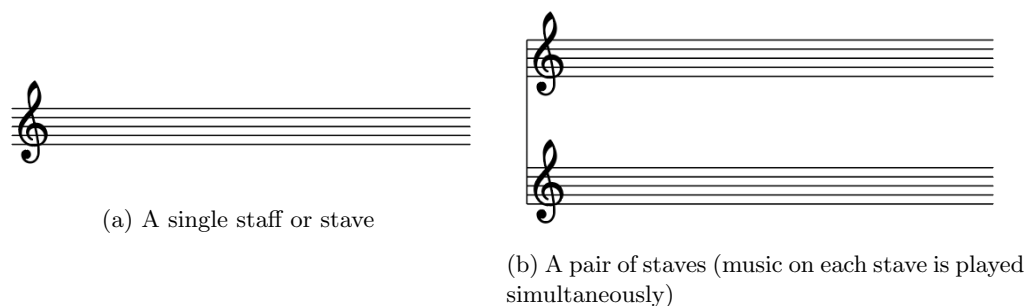


Figure 1.2

then the problem of generating the best spelling for a passage of music.

Before we delve into what spellings are possible and what makes them different (even evaluable), let us take stock of some of the commonalities between the different score<sup>1</sup> excerpts. Each excerpt consists of a pair of staves (Figure 1.2) that sound at the same time; each passage is represented using the same rhythmic notation (Table 1.1); finally, each passage has the same the sounding pitch, which can be verified through the use of Figure 1.3. The method of verifying that the passages represented have the same sounding pitch involves checking that all the corresponding combinations of accidental-notehead pairs<sup>2</sup> differently notated passages have the same pitch, or in other words, are on the same vertical line in Figure 1.3. For example, the first accidental-notehead-pair in the bottom part of the different passage representations is either  $E\flat$  or  $D\sharp$  (second from the left, top set of spellings in Figure 1.3). These spellings are indeed on the same vertical line in the diagram, which means that they represent the same pitch, namely the pitch that lies between  $D$  and  $E$  (as is indicated by the dotted arrow collinear with  $D\sharp$  and  $E\flat$  in Figure 1.3).

The method of verification just described is similar to the way a musician trained to read music would verify that the passages depicted are equivalent, except that the “lookup phase” would be less onerous, since they would not have to consult Figure 1.3! For an experienced “sight-reader”<sup>3</sup>, each accidental-notehead-pair might be immediately associated with a physical and/or aural cognitive reaction that makes the verification very quick indeed. What has been internalized by this hypothetical expert sight-reader is the *meaning* of the accidentals in context. Each accidental acts on the pitch represented when placed directly in front of a notehead (as in  $\sharp \bullet$ , in the context of

<sup>1</sup>A score is the notation of a whole piece of music (in the vein of Figure 1.1). This term and others feature in a glossary at the back.

<sup>2</sup>Noteheads are the following symbols:  $\bullet$ ,  $\circ$ ,  $\ominus$ . Accidentals are the following symbols:  $\sharp$ ,  $\flat$ ,  $\natural$ ,  $\sharp\sharp$ ,  $\flat\flat$ ,  $\times$ . Hence, accidental-notehead pairs are, for example,  $\sharp \bullet$ .

<sup>3</sup>reader of music

|                                                                                                                                                                                 | Relative<br>Length<br>(time) |                                                                                           |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------|-------------------------------------------------------------------------------------------|
| sounding  or  | $1/8$                        |  silence |
|                                                                                                | $1/4$                        |          |
|                                                                                                | $3/8$                        |                                                                                           |
|                                                                                                | $1/2$                        |          |
|                                                                                                | 1                            |                                                                                           |

Table 1.1: Musical time notation. Rhythms are read from left to right within a staff as time is passing, with a steady pulse (setting what is meant by “relative time”). The underlying pulse could be fast or slow, unless specified, in which case the passage of music could be over in a flash, or dragged out.

the staff) or directly after a letter name (as in  $D\sharp$ ). A “sharp” ( $\sharp$ ) raises the pitch of the associated notehead by one half-step. The full description of the action of the basic accidentals is given in Table 1.2 ( $\times$ ,  $\sharp$ ,  $\natural$ ,  $\flat$ ,  $\flat\flat$ ).

What we verified, either through lookup operations in Figure 1.3 or through learned knowledge of music notation, is that the passages represented have *equivalent spellings*, which is to say that they are *valid spellings of the same passage of music*. The (recursive) definition of a spelling is then the following:

A spelling is an assignment of individual spellings to the pitched events<sup>4</sup> (notes and chords<sup>5</sup>) of a passage of music. At the “base” of the recursion, we say that the spelling of a pitched event is the assignment of accidentals and vertical position in the staff to each of its noteheads.

Equivalently, each notehead is assigned an accidental, which determines its vertical position in the staff (and hence its letter name). The pitch-spelling problem is then the question of how to choose such an assignment, given a passage of music.

<sup>4</sup>The category pitched event is similar in concept to the `NotRest` class, from which `Note` and `Chord` inherit, in Michael Scott Cuthbert’s `music21` package [8, 9]. It is close both in name and function to the `Event` class in Robert Rowe’s C++ implementation accompanying his book [21].

<sup>5</sup>Events containing several noteheads, .



Figure 1.3: All the pitches in an octave and all their spellings, across two octaves (left and right). In the center, the ‘scale’ of pitches over the span of one octave, as it is represented on a piano. The namings  $\{B^\times, E^\times, F^{\flat\flat}, C^{\flat\flat}\}$ , that is, spellings of black notes that use double accidentals, come up so little in classical harmony as to be omitted by some authors in their complete list of spellings (see [13]). We include them for completeness of the notational system, as generated by the set of rules in Figure 1.4.

For  $p$  a pitch class and  $\mathcal{L} \in \{C, D, E, F, G, A, B\}$ :

$p$  permits a spelling  $\mathcal{L}\sharp\sharp \leftrightarrow p$  is two half-steps above  $\mathcal{L}$

$p$  permits a spelling  $\mathcal{L}\sharp \leftrightarrow p$  is one half-step above  $\mathcal{L}$

$p$  permits a spelling  $\mathcal{L} \leftrightarrow p$  is identified with  $\mathcal{L}$

$p$  permits a spelling  $\mathcal{L}\flat \leftrightarrow p$  is one half-step below  $\mathcal{L}$

$p$  permits a spelling  $\mathcal{L}\flat\flat \leftrightarrow p$  is two half-steps below  $\mathcal{L}$

Figure 1.4: The above collection of rules is necessary and sufficient to generate the set of spellings in Figure 1.3. The shortest distance between E and F, and between B and C is a half-step. Otherwise, the distance from a white note to the nearest black note is a half-step.

|                |     |              |
|----------------|-----|--------------|
| $\sharp\sharp$ | +2  | Double Sharp |
| $\sharp$       | +1  | Sharp        |
| $\natural$     | +0  | Natural      |
| $\flat$        | - 1 | Flat         |
| $\flat\flat$   | - 2 | Double Flat  |

Table 1.2: Abbreviated version of Figure 1.4 with the names of the accidentals included. The number in the middle of the table is the number of half-steps by which each accidental raises a pitch (negative values indicate that the pitch is instead lowered).

### 1.1.1 Evaluating Spellings

I have claimed that different spellings may be preferable to others. In certain contexts, there really can be a “right answer” to the question of how to spell a passage in the best possible way. Specifically, there are deterministic spellings that can be arrived at through an application of the understanding of the (classical) Tonal Harmony that is taught in music theory curricula. It is in the context of learned analytical tools that Robert Rowe approaches the question of computational pitch-spelling – within his framework of “Machine Musicianship”:

We labor mightily to make a computer program perform the analysis required of a freshman music student. Once the work is done, however, the program can make analyses more reliably and certainly much more quickly than the freshman.<sup>6</sup>

The first example (Figure 1.1a) demonstrates how musicianship training provides tools for choosing a good spelling. This example has a good spelling as it expresses the harmonic function of the notes and chords in relation to one another *as the passage sounds* (in the ears of certain classically trained beholders); that is, the passage *sounds* like an “Augmented-Sixth progression”, which would be found in most elementary books on classical or romantic harmony, and even without a broader context of where where this passage would sit in a particular score, there is a strong argument that this is the right way to spell it.

There is a somewhat strong, but in many cases completely reasonable, assumption in use here: the idea that we would want to reflect the wisdom of our Tonal Harmony textbook or teacher in the spellings that we (a) prefer and (b) write for ourselves. If we relax this assumption, it is conceivable we could have a different set of assumptions motivating our decisions and preferences. There might be a preference among some musicians with a jazz background for seeing the spelling in Figure 1.5b. Notice that only one note (as part of a chord about a quarter of the way from the left) is spelled differently. In the original example, it is spelled as a  $C^\sharp$ , while in the alternative spelling, it is written as a  $D\flat$ .

The difference in the spelling choices (Figures 1.5a and 1.5b) can be attributed to a set of “forces” acting on the note that changes, pulling it towards one option or the other. The note is drawn towards  $D\flat$  by the spelled notes in the first bar (before the vertical line separator, or barline.<sup>7</sup>). Notice that

---

<sup>6</sup>8, Section 1.3 [21]

<sup>7</sup>Incidentally, the barline also cancels the influence of any accidentals, hence the  $D$  on the other side of the barline is a  $D^\natural$  and not a  $D\flat$



Figure 1.5: Revisiting the first example.

the choice of a  $\flat$  accidental next to more  $\flat$  signs looks more congruent – as a rule of thumb, this makes for better spellings. Preferring that the notehead in question be spelled in keeping with the noteheads before the barline is a *harmonic* concern given that the first bar can be heard as having a uniform sonic identity or color. Indeed, a standard way to conceptualize jazz and to notate jazz leadsheets is to notate chords with letter based “chord symbols” on a per bar basis; for instance, the first bar would be labeled  $E\flat 7$ . By contrast, the desire to make the note a  $C\sharp$  is due to a *melodic*, or *voice-leading*, concern.<sup>8</sup> The  $C\sharp$  is spelled as such because it is “*leading*” to a  $D$  at the start of the next bar. Conversely, it is *undesirable* to have a  $D\flat$  and a  $D\sharp$  consecutively for reasons of voice-leading. The repetition of the  $D$ -letter name and equivalently, the notation of these two notes on the same line of the staff, fails to reflect the important psychoacoustic effect of the pitch being *raised*, which is conversely expressed by having  $C\sharp$  going up to  $D\sharp$ .

The hypothetical disagreement over  $C\sharp$  versus  $D\flat$  suggests the potential for spelling choices as an *expressive* device, through which the scribe (potentially a composer) can express something about pitch relationships that they wish to emphasize to the reader (potentially a performer). If I see the  $D\flat$ , I am seeing (and hearing) the pitch as belonging to the collection of notes before the barline, and if I see the  $C\sharp$ , I am hearing the note as pushing into the next bar.<sup>9</sup>

Although there may not necessarily be spelling that everyone can agree on, there *are* choices of spelling that wilfully obscure. It is worth asking what makes this spelling, reprised in Figure 1.6, quite so “bad”. There are three key issues that make this spelling difficult to read.

1. The distance between pitches as they are heard is not reflected by the spacings between pitches

<sup>8</sup>Indeed, this is the way that the conventional spelling of an Augmented-Sixth progression would be justified or explained.

<sup>9</sup>On some instruments, the difference between the  $C\sharp$  and  $D\flat$  could be more than just psychological, with the  $C\sharp$  being tuned just a fraction higher in pitch than the  $D\flat$ . These distinctions in pitch are not possible on all instruments (take, for example, the piano), and depend a lot on context. In this thesis, we will assume “equal temperament”, the equivalence in *sound* of what we refer to as equivalent spellings. See [15] for a discussion of tuning systems.



Figure 1.6: Bad spelling from the first example.

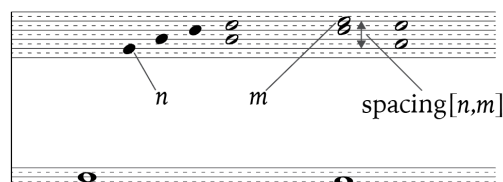


Figure 1.7: The spacing invariant in the spellings of Figure 1.1a and Figure 1.5a.

as they are written. For example, the movement of pitch in the second part actually appears to be *upward* even though we hear descending pitch (which we can confirm is reflected in the other spelling we have seen so far). In fact, most of the spacings are distorted (if not quite flipped). A good mix of spacings (invariant across Figures 1.1a and 1.1b) is shown in Figure 1.7.

2. There is a mix of flats and sharps in close proximity, in ways that are not justifiable with explanations involving harmonic concerns or voice-leading.
3. There is unnecessary use of double accidentals (♯'s, double sharps, and ♭'s, double flats). Double accidentals tend to only be employed when forced by the necessity to yield a certain spacing between notes (as in the first item of this list). Consider the second spelling from the first example (reprinted in Figure 1.8). As mentioned, it maintains the spacings of the first example, and so it is “good” on the first count (item one). But in order to preserve proper spacings, double-sharps must be introduced. Certainly among beginning readers of music, and even for more advanced musicians, it is more difficult to read passages that contain a lot of double-sharps. Hence, they are introduced more often as a fallback than an immediate choice.

These negative issues provide an insight into what *positive* criteria might be looked for in a pitch-spelling algorithm.

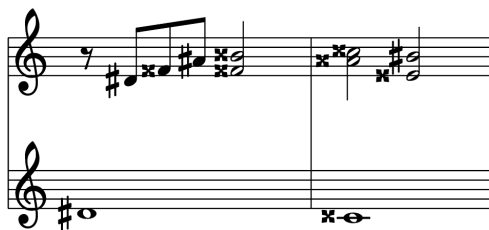


Figure 1.8: Second spelling from the first example.

## 1.2 Responding to Related Work

Honingh has put forward a number of criteria for evaluating pitch spelling algorithms, these being “parsimony”, “spelling accuracy”, “generalizability” and “cognitive plausibility” [13]. I will here address how these criteria have been addressed in the literature and which criteria I wish to emphasize in my own work.

### 1.2.1 Parsimony

There is a good deal of consensus in the literature about, for example, point 3 in the list of issues with Figure 1.6 in the previous part; it is mostly agreed that it is desirable to minimize the number of double accidentals used in a spelling. Relatedly, it is agreed that where possible, we should want to minimize the number of accidentals used at all.<sup>10</sup> These principles are encompassed by the term, introduced by Cambouropoulos, of notational *parsimony* [4].

### 1.2.2 Accuracy

The most common method of evaluating pitch spelling algorithms in the literature has been to measure its “spelling accuracy” which has generally meant to measure its success in replicating the spelling choices made by composers (and/ or editors) over a large corpus of works, and then to return the accuracy as a percentage success rate [18, 17, 6, 16]. In most cases, the test corpus is populated with pieces of a certain era, or more often a single composer (e.g. Bach [13, 16]), and hence with a specific kind of rule based tonal harmonic language. I will not be interested in this measure of correctness, although it has been a strong way to make direct comparisons between the

<sup>10</sup>Recall that a notehead with no accidental indicates that the pitch should be understood as a white note or plain letter-name. Alternatively, accidentals carry over to any noteheads of the same vertical position that appear before the next barline.

work of different researchers [16].

While it is more or less possible to view Bach’s spelling choices as “correct”, there is not an obvious way to use a spelling accuracy method of appraisal in a more modern or subjective spelling context. There is even less of a guarantee for dealing with extended sets of accidentals (at half and a quarter the granularity, for example), which I will refer to as the “Microtonal Context” (Chapter 5), since these spellings are certainly not found in canonical corpora. As we have seen in Section 1.1, reasonable differences in preference rules and priorities can result in different results in spelling. Hence, notions of “accuracy” and “correctness” are unstable when we open up to spelling priorities that are not necessarily in keeping with the spelling rules “you learned in musicianship class”. Recall that, for these kinds of problems, we have seen how different spellings can *communicate* different priorities to the reader or performer. If we accept that these priorities and communication thereof may be subject to change, we will prefer to view the outcome of a spelling algorithm as a “good default” rather than objectively correct.

Instead of testing against corpora, I give provable results in Chapter 4 about *fitting to corpora*, as well as developing a language of corpus-consistency - a means of describing how intervallic relationships differ between corpora.

### 1.2.3 Generalizability

One of the new criteria put forward by Honingh is “generalizability”, the power of the tools to be turned to other problems. Several of the sets of tools have been shown to be of use in other domains of computational music analysis and music cognition [5],[14], [13, 12]. I would add further important aspects of generalizability that have not been explicitly addressed in the literature and that will be of particular interest to me in this thesis. First, generalizability could include the potential to account for spelling in non-tonal contexts, for instance in certain modern composition practices that are not even necessarily *atonal*, but equally, not functionally tonal in the sense that is expected in the classical corpora against which spelling algorithms have usually been tested. Second, generalizability could include the potential to account for spelling in microtonal contexts (with finer granularity of accidentals). Along the lines of generalizability, I add “flexibility”, the ability to modify priorities, which ties into the demonstration of two reasonable spellings according to different priorities even in the context of two *tonal* systems of spelling (Section 1.1).

### 1.2.4 Search Spaces

Before addressing “cognitive plausibility”, let us briefly take a more detailed view of the algorithms themselves found in the literature. Most of the algorithms arrange the set of all pitch-classes in some formation – a line [17, 21, 24], a three-dimensional coiled line [6], or a two-dimensional planar arrangement [13] (Figure 1.9). The initial aim of each the algorithm is then to minimize spread in

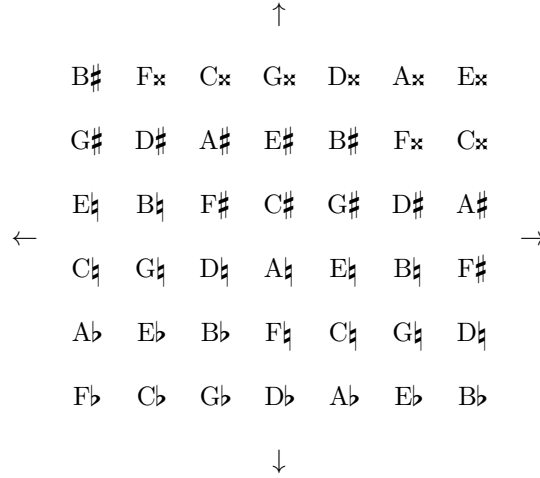


Figure 1.9: The planar arrangement of pitch-class names used in Honigh’s work

the space while picking out pitch-class names that are valid spellings for notes in the score. Hence, Honingh’s statement of the approach is a search for compact orientations in the Euler-Lattice, namely the planar arrangement of pitch class names in Figure 1.9. Most of the other approaches aim to minimize proximity of “the next note” to a center of mass computed from a group of the preceding notes [6, 5, 18, 14]. Meredith’s algorithm differs in that it looks at the distances in the Line-of-Fifths a distribution of surrounding pitches; it then adds a second run through to correct for voice-leading errors [17].

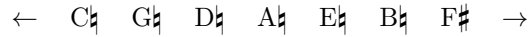


Figure 1.10: The Line-of-Fifths. The “interval” between adjacent pitch class names is called a “perfect fifth”, which is considered to be the second most consonant interval after unison (or the octave) [15]. See Section 1.2.5 for a description of musical intervals.

In each of the algorithms surveyed, there is strong agreement about the use of certain primitive in each of the one-dimensional, two- and three-dimensional formations in the search space. The use of the “Line-of-Fifths” primitive is due to Temperley [24], and is also central to the setups proposed

by other authors. Each row of Honigh’s Euler-Lattice (Figure 1.9) is in fact a “Line-of-Fifths” and Meredith has proved that the three-dimensional coiling approach has the same effect of using a Line-of-Fifths primitive [19]. Instead of using the Line-of-Fifths *explicitly*, Cambouropoulos seeks to maximize the frequency of certain spacings: the spacings that are most common in classical music – within major and minor scales [3, 4] – which in turn result in pitch class names that are closest together in the Line-of-Fifths. The music theoretical term for the spacings, the distance between pairs of pitches (with a few technicalities), is “interval”.

### 1.2.5 Intervals

Examples of intervals (as the term is used in the musical context) include “perfect unison” and “perfect octave”, which can each be thought of as a mathematical “identity interval” used to describe the distance between two pitches with the same pitch class names (e.g.  $D\flat$ -to- $D\flat$ ). If we raise the second pitch (yielding an interval of  $D\flat$ -to- $D\sharp$ ), we refer to the interval as augmented, e.g. “an *augmented* octave”. If we lower the second pitch (yielding an interval of  $D\flat$ -to- $D\flat$ ), we get a *diminished* octave. In either case, the interval is still an *octave* or *unison*, because the two letter names are  $D$ .

There is a name for every inter-letter-name distance (on the piano keyboard, or in the alphabet *mod* 7), shown in Table 1.3. The set of valid modifiers to letter-name intervals is given in Table 1.4. The set of interval spellings available for each set of half-step distances is given in Table 1.3. The preparation stage of Cambouropoulos’s algorithm involves obtaining a distribution on intervals based on a weighted sum over the prevalence of the intervals in major and minor scales [3]. This distribution is thresholded to form four priority groups, with the intervals with the highest incidence rate having the highest priority in the spelling phase.

### 1.2.6 Cognitive Plausibility

A cognitively plausible pitch-spelling algorithm is defined to be one that closely reflects the way that people come up with spellings [13]. For example, Longuet-Higgins’s 1976 computational approach to pitch-spelling was motivated by the question of how musicians take melodic dictation [14]. As far as more recent efforts are concerned, at least in the non-tonal context, my experience of pitch-spelling as a composer has been closer to the intervallic approach. As a musician, if I am reading a piece of

|        |                 |
|--------|-----------------|
| C to C | unison          |
| C to D | second          |
| C to E | third           |
| C to F | fourth          |
| C to G | fifth           |
| C to A | sixth           |
| C to B | seventh         |
| C to C | octave          |
| C to D | compound second |
|        | ⋮               |

Table 1.3: Letter name intervals

| Letter name interval | Variants |            |            |           |
|----------------------|----------|------------|------------|-----------|
| unison               | Perfect  | Diminished | Augmented  |           |
| second               | Major    | Minor      | Diminished | Augmented |
| third                | Major    | Minor      | Diminished | Augmented |
| fourth               | Perfect  | Diminished | Augmented  |           |
| fifth                | Perfect  | Diminished | Augmented  |           |
| sixth                | Major    | Minor      | Diminished | Augmented |
| seventh              | Major    | Minor      | Diminished | Augmented |
| octave               | Perfect  | Diminished | Augmented  |           |

Table 1.4: Ways of varying the letter name intervals (e.g. “major third”). Double-augmented, and double-diminished intervals are also possible but we leave them out for simplicity.

| # half-steps | Options            |                   |                  |
|--------------|--------------------|-------------------|------------------|
| 0            | perfect unison     | diminished second |                  |
| 1            | augmented unison   | minor second      |                  |
| 2            |                    | major second      | diminished third |
| 3            |                    | augmented second  | minor third      |
| 4            | diminished fourth  |                   | major third      |
| 5            | perfect fourth     |                   | augmented third  |
| 6            | augmented fourth   | diminished fifth  |                  |
| 7            |                    | perfect fifth     | diminished sixth |
| 8            |                    | augmented fifth   | minor sixth      |
| 9            | diminished seventh |                   | major sixth      |
| 10           | minor seventh      |                   | augmented sixth  |
| 11           | major seventh      | diminished octave |                  |
| 12           | augmented seventh  | perfect octave    |                  |

Table 1.5: Naming options for different *sounding* intervals a certain number of half-steps’ distance which has a certain sound regardless of starting pitch.

music with harmony that is not close to classical, my greatest priority is to see “sensible” intervals, so I can hear relationships more easily at a glance. In that sense, I will join Cambouropoulos in making pairwise pitch relationships the emphasis of my algorithm rather than more absolute pitch criteria. Unlike in Cambouropoulos’s work, I will allow the emphasis placed on certain intervals to differ based on the pitch classes involved.

As with generalizability, I propose an extension to the criterion of cognitive plausibility. As I am composing music, I try to optimize the intervals I am introducing with each new note that I write, but I understand that as I add to my score, the introduction of a new note could have a back-propagating influence on the notes that have been spelled before. In very complex situations, it can be hard to have the vision to see what chain of changes could be implemented to improve the overall spelling of the notes that have been already written down. The method I propose accommodates back-propagation corrections, while still operating on the same principles as the cognitively plausible case, and hence, can be viewed as useful and at the same time comprehensible to the working musician.

### 1.2.7 Windowing

All of the algorithms involve windowing, whereby, the algorithm has a “view-finder” with a fixed width that performs local optimizations within its view before proceeding (forward) through the score. The use of windowing seems to respond to the idea that spelling should be happening in real-time in response to an input stream of pitches. My approach does away with windowing, and with the idea that spelling needs to be done by progressing incrementally through the score. I show that the Pitch-Spelling Problem can be modelled in such a way that there is a holistic goal and that the problem can be approached with the whole picture always in view.

## 1.3 The Minimum Cut Problem

The basic essence of my contribution is to introduce a graphical model of the score as a flow network that allows us to recast the problem of pitch spelling in terms of the Minimum Cut Problem.

In Figure 1.11, I give a depiction that I keep in mind when thinking about Minimum Cut. We make a  $\{0, 1\}$ -assignment over all the nodes in some graph (with directed edges), where there are two nodes in the graph that are preassigned values 0 and 1 respectively. We only incur a cost between

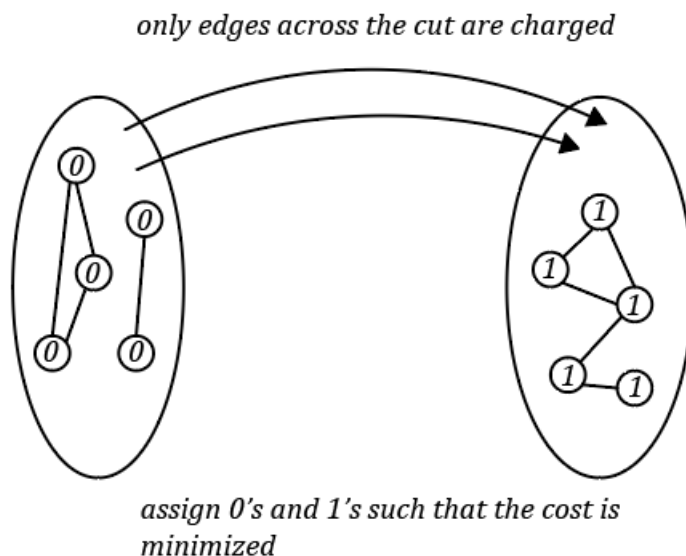


Figure 1.11: The minimum cut problem

edges with start-node 0 and end-node 1. The goal is to minimize the total cost (the sum over all edges).

Strictly speaking, the above description of Minimum Cut differs from the formal description of the minimum cut problem, wherein each assignment only needs to be in the unit interval  $[0, 1]$  rather than purely binary. A remarkable property of the minimum cut problem is that there are always  $\{0, 1\}$ -assignments that are optimal solutions and that can be found very efficiently by built-for-purpose algorithms like Hochbaum’s Pseudoflow algorithm [10, 11] or the widely used Simplex algorithm, which is known to be very fast in practice at solving linear programs [2, 22, 23]. Linear programs are minimization or maximization problems with linear objectives (e.g.  $\min x + 3y$ ) and affine inequality constraints (e.g.  $x - 3y + 3 \leq 0$ ).<sup>11</sup>

### 1.3.1 Maximum Flow

Hochbaum’s algorithm is actually proposed to solve the “maximum flow” problem, but *on the way* finds a minimum cut. It turns out that max flow and min cut are very closely linked. Indeed, in most sources, minimum cut is only introduced because of its relation to max flow, which tends to have many more applications. In the maximum flow problem, we are given a directed graph,

<sup>11</sup>For an introduction to linear programming, see [2]

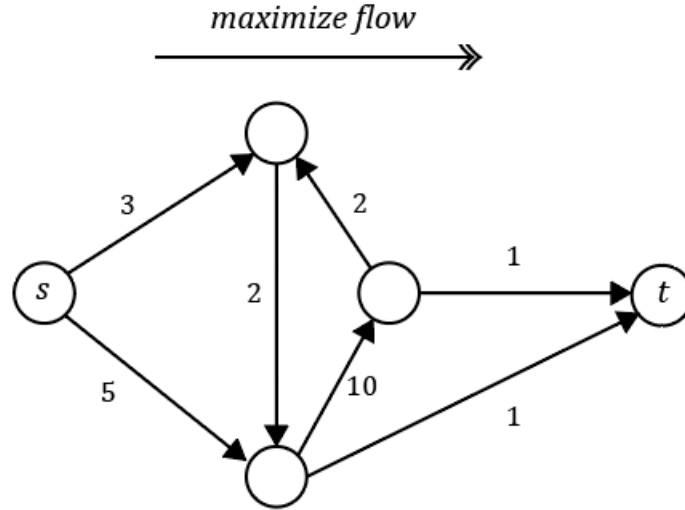


Figure 1.12: A maximum flow problem

or flow network. The weight of each edge is interpreted as a capacity on flow. In the example in Figure 1.12, the maximum flow is 2, because 2 units of flow can be pushed through the whole network without exceeding any capacity constraints, and this must be an upper bound, because the sum of the weights of the edges on the right hand side is 2. This hints at a key result in the field, called the *max flow/min cut theorem*, which states that the maximum flow of a network is equal to a minimum cut defined on the same network [2]. The minimum cut is achieved in Figure 1.12 when the first four nodes are assigned the value 0 and the node  $t$  is assigned value 1.

*A sketch of a proof of the max flow/min cut theorem.* In the maximum flow setting, the minimum cut is the bottleneck for flow – the set of arcs between the source and the sink that saturate first as we push arcs to their capacity; the flow across this edge-set constrains the most flow that can be pushed from  $s$  to  $t$ , but equally, this flow is attained as there is no more binding set of edge constraints that separates  $s$  from  $t$  (by the minimality of the Minimum Cut).  $\square$

## 1.4 Approach and Outline

Generally, we will view notes in a passage of music as related to nodes in a flow network. Connections will be drawn between notes that are close to each other in time in the passage of music, corresponding to proximity spatially in the score. Edge-weights will be assigned to reflect impor-

tance that the interval between the notes be spelled in an ideal way. The assignments found by a minimum cut finding algorithm will correspond to the assignment of *spellings* to each of the notes in the resulting score.

The outline:

- In Chapter 2, I give a precise formulation of the graphical model for a given passage of music, in terms of two uncertain matrices of parameters.
- In Chapter 3, I state the forward minimum cut problem in terms of this graph and resolve the uncertainty of one of the unknown matrices.
- In Chapter 4, I exploit the *min cut/max flow theorem* to extract the other matrix of parameters from a corpus of musical scores in an inverse formulation of the problem setup.
- In Chapter 5, I give the extension of the model and algorithm to the microtonal context.

## 1.5 Contributions

In this thesis, I make the following contributions:

1. A graphical model of pitch notation featuring a novel approach to pitch spelling that is agnostic with respect to the arrow of time through the score and does not require time-windowing.
2. A formulation of the pitch-spelling problem as a minimum cut search in the flow network – equally, providing a rare instance of a minimum cut *primal* problem rather than a mere dual of maximum flow.
3. A notion of a “parsimony pivot”, an artificially added pitch (adjacent to all notes in a passage) introduced as a means of accounting for the problem of parsimony in pitch-spelling.
4. A mechanism referred to as an “adjacency matrix pullback” used to account for the effects of different musical parameters – at different indexing scales.
5. Decomposition of the problem into modular parts via matrix manipulation. The use of geometric series and exponential decay terms to lightly effect the priorities of the pitch-spelling algorithm such that the mechanism of the algorithm is unchanged.

6. An inverse formulation designed to help populate the parameters of the forward problem formulation (run on a large corpus of musical scores).
7. Two theorems regarding the potential of the inverse formulation to derive parameters that produce “the right” spellings when used as parameters for the forward problems. A concrete way of comparing the spelling priorities of different corpora in terms of the inverse problem and a notion of “consistency” between corpora and sets of parameters.
8. An extension of the theory to the microtonal context (unattempted in the literature) featuring two new forward problem formulations and three algorithmic approaches for solving them.

## Chapter 2

# Graphical Representation

We will begin with a discussion of musical scores, which we will first model in terms of parallel chains of events to derive some information about closeness and adjacency, and then as a flow network. The final flow network will be home to our (minimum cut) algorithm.

### 2.1 Parsimony Pivot

A simple way to enforce parsimony is by introducing a dummy note held for the duration of the passage of music  $\mathcal{M}$ . Recall that parsimony is the “Ockam’s Razor” of pitch-spelling, as Honing puts it [13] – the desire for a certain simplicity of final representation. For us this means, the avoidance of double accidentals and the avoidance of accidentals altogether if the spelling can be expressed with plain letter-names.

**Definition 2.1** (Parsimony Pivot). *Let  $\mathcal{M}$  be a passage of music that we wish to spell. We define the parsimony pivot for  $\mathcal{M}$ ,  $p_{\mathcal{M}}$ , to be a note, artificially added to the score<sup>1</sup>, with predetermined spelling, that lasts for the duration of  $\mathcal{M}$ .*

In Chew’s framework, a good candidate for the parsimony pivot could be a note close to the Center of Effect<sup>2</sup> of a given passage (or segment) of music [6]. In harp music, a good choice could be  $\mathbb{C}\flat$ , a pitch class name that is uniquely appreciated by harpists, because it is the natural way of referring to the lowest note on the harp (and its most resonant key)! In most cases, however, we

---

<sup>1</sup>I owe the idea of adding a dummy note to a score to settle parsimony to James Bean [1].

<sup>2</sup>essentially a center of mass over the pitch classes arranged in a “spiral array” or coiled “Line-of-Fifths”

will take  $p_{\mathcal{M}}$  to be of pitch class D and with spelling  $D\sharp$ . D is a natural choice of parsimony pivot for several reasons. Compellingly, if one considers the most natural spellings of all pitch classes in the context of just  $D\sharp$  we get the outcome laid out in Table 2.1. Indeed, this is the distribution of spellings if one minimizes distance between spellings in the Euler-Lattice, the Spiral Array or the Line-of-Fifths, or if one chooses from the preferred set of intervals in Cambouropoulos’s modeling approach [13, 5, 4, 24, 21]. The distribution of spellings in Table 2.1 is, moreover, very parsimonious, from a qualitative perspective. It has an even split of  $\sharp$ ’s and  $\flat$ ’s, thus not introducing a bias one way or the other, and it makes use of  $\sharp$ ’s wherever permissible – recall that a  $\sharp$ -modified letter name is equivalent to a plain letter name (refer back to Table 1.2). As is desirable, the set of spellings in 2.1 completely avoids double accidentals ( $\times$ ,  $\flat\flat$ ). These accidentals should be permissible in a spelling system, but there should be a bias against using them, in the interests of parsimony.

| Pitch Class | 0         | 1         | 2         | 3        | 4         | 5         | 6         | 7         | 8                | 9         | 10       | 11        |
|-------------|-----------|-----------|-----------|----------|-----------|-----------|-----------|-----------|------------------|-----------|----------|-----------|
| Spelling    | $C\sharp$ | $C\sharp$ | $D\sharp$ | $E\flat$ | $E\sharp$ | $F\sharp$ | $F\sharp$ | $G\sharp$ | $G\sharp/A\flat$ | $A\sharp$ | $B\flat$ | $B\sharp$ |

Table 2.1: Most natural spellings with respect to a parsimony pivot of  $D\sharp$ .

The parsimony pivot should be no more than a tie-breaker in choosing between say the examples in Figures 1.1a and 1.1b. Recall that their set of intervals is identical, and so the only difference between them is a shift of letter-names accompanied by a shift of accidentals into a territory with more  $\times$ ’s than are strictly necessary. The light tug of an invisible  $D\sharp$ , would pull the spelling towards the more parsimonious collection of notes in Table 2.1. Of course, the influence of the parsimony pivot should not be too strong, or it could wind up distorting the spelling chosen. Consider a passage with just two notes, one from pitch class 1 and the other from pitch class 3. If the tug of the parsimony pivot is too great, it might overpower the intervallic force between the two pitches, which would result in a  $C\sharp$ ,  $D\sharp$  combination or a  $D\flat$ ,  $E\flat$  combination. A strongly weighted  $D\sharp$  will instead yield a  $C\sharp$ ,  $E\flat$  combination, by no means a good spelling in isolation.

The significance of  $p_{\mathcal{M}}$  lasting for the whole of  $\mathcal{M}$  will be that it is adjacent to every note in  $\mathcal{M}$ , in a sense that will be made clear in the following sections. Being adjacent to all notes in  $\mathcal{M}$ , the parsimony pivot will exert a light tug on all notes in  $\mathcal{M}$ .  $p_{\mathcal{M}}$  will also provide the basis for the source and sink vertices (as  $s$  and  $t$  are in Figure 1.12) in the eventual minimum cut problem setup.

## 2.2 Segmentation

All pitch spelling algorithms in the literature rely on a segmentation scheme, a way of splitting a score into smaller chunks. While the algorithm presented relies on less rigid rules of segmentation than many other approaches in the literature, there is still a need to break up the local pitch context in certain moments of a score. For instance, if a musician has a 5 minute rest between two subpassages in the score that they are responsible for, it is reasonable to let the two subpassages be spelled independently of each other, since the musician has 5 minutes between the completion of one note and the preparation of the next.

Our general approach is to connect all events in the same part in a chain. But we will break the chain in instances such as the “long rest”<sup>3</sup> scenario mentioned above. There are other reasons that the chain may be cut, for instance if we run into a double barline, which is a musical semantic of indicating “new section”. We will call these score elements that refresh the local pitch context (“long rests”, double barlines etc.) “*chain-breakers*”.

Recall from the introduction that a passage of music  $\mathcal{M}$  is divided into *parts*  $\mathcal{M}(i)$ ,  $i \in \mathcal{P}(\mathcal{M})$ . A string quartet has four parts (*violin I*, *violin II*, *viola*, *'cello*), so for  $\mathcal{M}$  a passage from a string quartet, we have  $\mathcal{P}(\mathcal{M}) = \{0, 1, 2, 3\}$ , and the parts  $\mathcal{M}(0)$ ,  $\mathcal{M}(1)$ ,  $\mathcal{M}(2)$ ,  $\mathcal{M}(3)$  corresponding to the *violin I*, *violin II*, *viola* and *'cello* parts respectively.

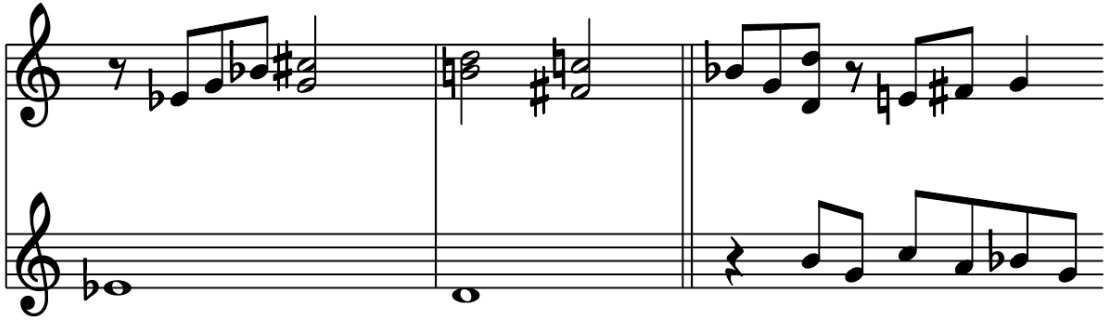


Figure 2.1: Passage with two parts.

**Definition 2.2** (Time-interval chain).  $T_{\mathcal{M}(p)}$  is a countable collection of disjoint time intervals<sup>4</sup>

<sup>3</sup>The notion of what we consider a long rest is contextual. If the tempo of a piece of music is very fast (there are a larger number of beats per minute), then the same rest (notated in terms of the number of beats it lasts) can vary wildly in terms of its *absolute time* duration (see Table 1.1 – by convention, the 1/4 length elements are generally said to last 1 beat).

<sup>4</sup>Disjoint in the sense that their intersection have 0 width.

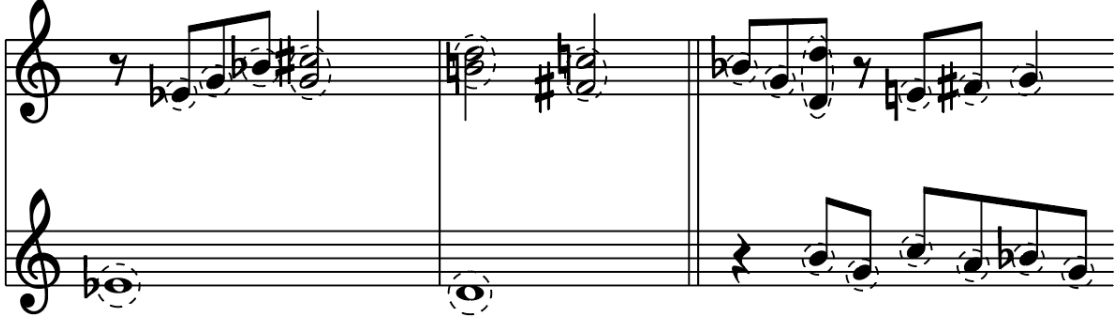


Figure 2.2: Events (chords and individual noteheads) circled.

associated with the part  $p$  in the passage of music  $\mathcal{M}$ . We will write  $T_{\mathcal{M}(p)}(i)$  to mean the  $i$ th time-interval of  $T_{\mathcal{M}(p)}$ . We allow “instantaneous” time-intervals, namely time-intervals where the start of the interval is equal to its end point.

*Example 2.1.* The time-interval chain for the top part of the passage in Figure 2.1 is given by

$$\begin{aligned} \mathcal{T}_{\mathcal{M}(0)} = & ([0.5, 1], [1, 1.5], [1.5, 2], [2, 4], [4, 6], [6, 8]) \\ & \circ ([8, 8]) \\ & \circ ([8, 8.5], [8.5, 9.0], [9.0, 9.5], [10, 10.5], [10.5, 11], [11, 12]) \end{aligned} \quad (2.1)$$

We split up the chain into sublists to emphasize the separation of events before the double barline (a chain-breaker) and after it.

In a similar way, the time-interval chain for the bottom part of the passage in Figure 2.1 is given by

$$\begin{aligned} \mathcal{T}_{\mathcal{M}(1)} = & ([0, 4], [4, 8]) \\ & \circ ([8, 8]) \\ & \circ ([9, 9.5], [9.5, 10], [10, 10.5], [10.5, 11], [11, 11.5], [11.5, 12]) \end{aligned} \quad (2.2)$$

Notice that

The above example motivates the next definition.

**Definition 2.3** (Event chain partitioning).  $\mathbf{EX}_{\mathcal{M}(p)}$  is an ordered list of “events” and “chain-

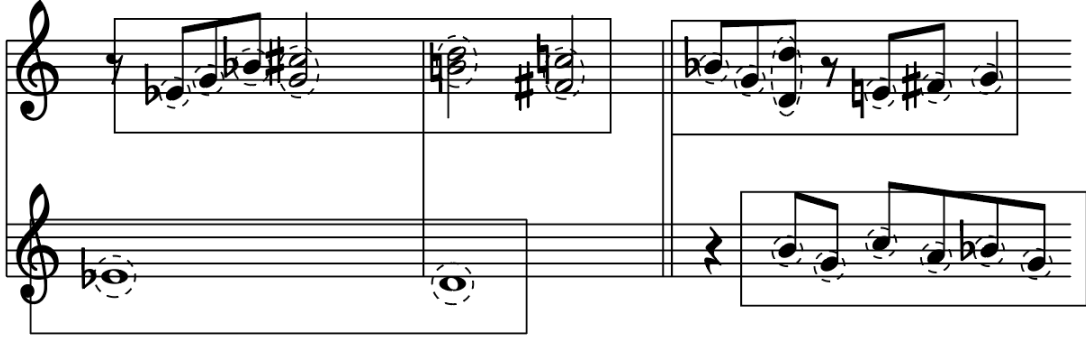


Figure 2.3: Sub-parts indicated. The double barline in the middle is a “chain-breaker”

breakers” indexed by  $T_{\mathcal{M}(p)}$ . If  $\tau \in T_{\mathcal{M}(p)}$ , we write  $\mathbf{EX}_{\mathcal{M}(p)}(\tau)$ , to mean the event or chain-breaker indexed by time-interval  $\tau$ . Let  $(\tau_0, \dots, \tau_N)$  be the list of time intervals that index chain-breakers, in order, and let  $(\chi_0, \dots, \chi_N)$  be the list of associated chain-breakers. We count the start and end of the passage as chainbreakers  $\chi_0$  and  $\chi_N$  respectively. The disjoint Event Chain Partitioning of  $\mathbf{EX}_{\mathcal{M}(p)}$  is a series of lists “cut-out” by removing the chain-breakers:

$$\mathbf{E}_{\mathcal{M}(p)}^1, \mathbf{E}_{\mathcal{M}(p)}^2, \dots, \mathbf{E}_{\mathcal{M}(p)}^N \quad (2.3)$$

with

$$\chi_0 \circ \mathbf{E}_{\mathcal{M}(p)}^1 \circ \chi_1 \circ \mathbf{E}_{\mathcal{M}(p)}^2 \circ \chi_2 \circ \dots \circ \chi_{N-1} \circ \mathbf{E}_{\mathcal{M}(p)}^N \circ \chi_N = \mathbf{EX}_{\mathcal{M}(p)} \quad (2.4)$$

The  $\mathbf{E}_{\mathcal{M}(p)}^l$  are purely made up of events and not chain-breakers. We define  $\mathbf{E}_{\mathcal{M}(p)}$  to completely omit all the chain-breakers. Hence

$$\mathbf{E}_{\mathcal{M}(p)} = \mathbf{E}_{\mathcal{M}(p)}^1 \circ \mathbf{E}_{\mathcal{M}(p)}^2 \circ \dots \circ \mathbf{E}_{\mathcal{M}(p)}^N \quad (2.5)$$

We call  $\{\mathbf{E}_{\mathcal{M}(p)}^k\}$  the set of sub-parts.

*Remark 2.1* (A note on the ordering of events and chain-breakers). Even when there is a tie of events and chain-breakers with the same instantaneous time, we can defer to an ordering on “types” of events and chain-breakers, which is nicely implemented in `music21` in the language of objected-oriented-programming-style “class hierarchy” [8, 9].

## 2.3 Closeness

### 2.3.1 Events

We can enumerate the sub-parts, of  $\mathcal{M}$  so that each one is associated with a different integer. Then we let  $\mathcal{X}(e)$  equal the integer of the sub-part to which the event  $e$  belongs. The position of  $e$  is fully determined by  $\mathcal{X}(e)$  and  $\mathcal{N}(e)$ , which we take to mean the position of  $e$  in the order of events within sub-part  $\mathcal{X}(e)$ .

$\mathcal{N}$  provides a way of characterizing distance in the score in a crude way. We let  $\mathbf{M}_k = (m_{i,j}^k)$  be the  $k$ -close matrix, logging all pairs of events a distance  $k$  apart in the sense of  $\mathcal{N}$ .

$$m_{i,j}^k = \begin{cases} 1 & \text{if } |\mathcal{N}(i) - \mathcal{N}(j)| = k \\ 0 & \text{otherwise} \end{cases} \quad (2.6)$$

We will enhance our definition of closeness using the following two definitions.

**Definition 2.4** (Entry-wise multiplication). *We notate entry-wise multiplication of matrices with the symbol  $\cdot \cdot$ .*

*Usage example:*

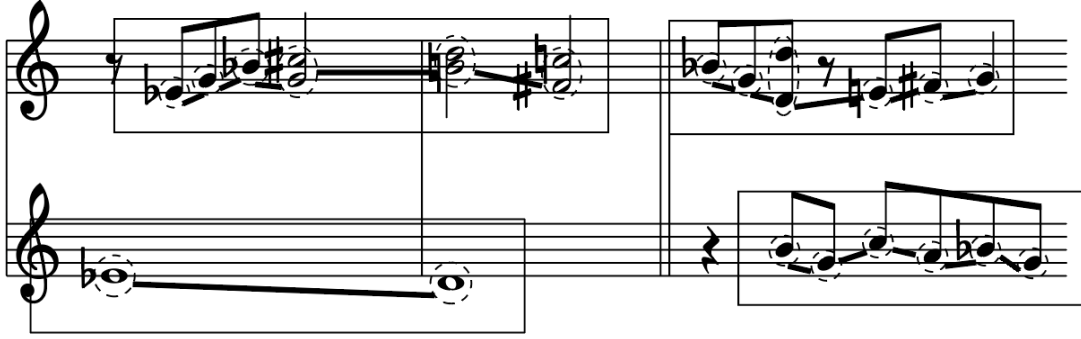
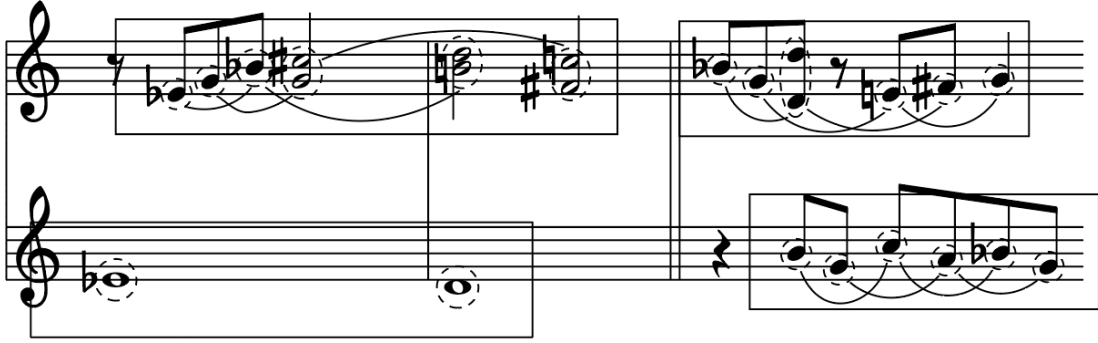
$$A \cdot \cdot B = (a_{i,j} \cdot b_{i,j}) \quad (2.7)$$

**Definition 2.5** (Matrix pullback). *Let  $G = (V, E)$  be a graph with adjacency matrix  $A$ . Fix a map  $f : N \rightarrow V$ . We notate the pullback adjacency matrix of  $A$  with respect to  $f$  as  $f^{-1}(A) = (b_{i,j})_{i,j \in |N|}$  where we use  $f$  to determine the value of  $b_{i,j}$  by looking at the value of  $A$  at index  $(f(i), f(j))$ .*

$$b_{i,j} = a_{f(i), f(j)} \quad (2.8)$$



Figure 2.4: Part with 12 events and 2 sub-parts.

Figure 2.5: Connections drawn in according to the adjacency structure of  $\mathbf{A}_1$ .Figure 2.6: Connections drawn in according to the adjacency structure of  $\mathbf{A}_2$ .

*Example 2.2.* Let  $\mathbf{I}$  be a  $2 \times 2$  identity matrix. Acting on the passage shown in Figure 2.4, we obtain:

$$\chi^{-1}(\mathbf{I}) = \begin{pmatrix} \mathbf{I}_6 & \mathbf{0}_6 \\ \mathbf{0}_6 & \mathbf{I}_6 \end{pmatrix} \quad (2.9)$$

where  $\mathbf{I}_6$  is a  $6 \times 6$  identity matrix and  $\mathbf{0}_6$  is a  $6 \times 6$  zero matrix. The interpretation of the outcome is that the first 6 notes are in the same sub-part and the second set of 6 notes are in the same distinct sub-part.

Using these definitions, we can take

$$\mathbf{A}_k = \mathbf{M}_k \cdot \chi^{-1}(\mathbf{I}) \quad (2.10)$$

where  $\mathbf{I}$  is an identity matrix with dimension the number of sub-parts.  $\mathbf{A}_k$  is equal to  $\mathbf{M}_k$  except that it only connects events that are in the same sub-part.

We also want to have a notion of adjacency between parts as well as within sub-parts. Events in separate parts will be adjacent if their associated time intervals overlap. We will let  $\mathcal{T}(e)$  denote the time-interval associated with an event  $e$  (the time period during which  $e$  sounds). Let  $\mathbf{H} = (h_{i,j})$  be the matrix that logs pairs of overlapping events.

$$h_{i,j} = \begin{cases} 1 & \text{if } \mu(\mathcal{T}(i) \cap \mathcal{T}(j)) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2.11)$$

where  $\mu$  measures the width of an interval.

$$\mathbf{B} = \mathbf{H} \cdot \cdot \overline{\mathcal{X}^{-1}(\mathbf{I})} \quad (2.12)$$

where  $\overline{\mathbf{A}}$  denotes the complementary matrix of a  $\{0,1\}$ -matrix  $\mathbf{A} = (a_{i,j})$ , namely with  $\overline{\mathbf{A}} = (1 - a_{i,j})$ . Hence,  $\mathbf{B}$  is equal to  $\mathbf{H}$  except that it only connects events that are in different sub-parts. We could also have taken  $\mathbf{B} = \mathbf{H} \cdot \cdot \overline{\mathcal{P}^{-1}(\mathbf{I})}$ , where  $\mathcal{P}(e)$  gives the part of event  $e$ , since overlapping events are in separate *sub-parts* if and only if they are in separate *parts*.

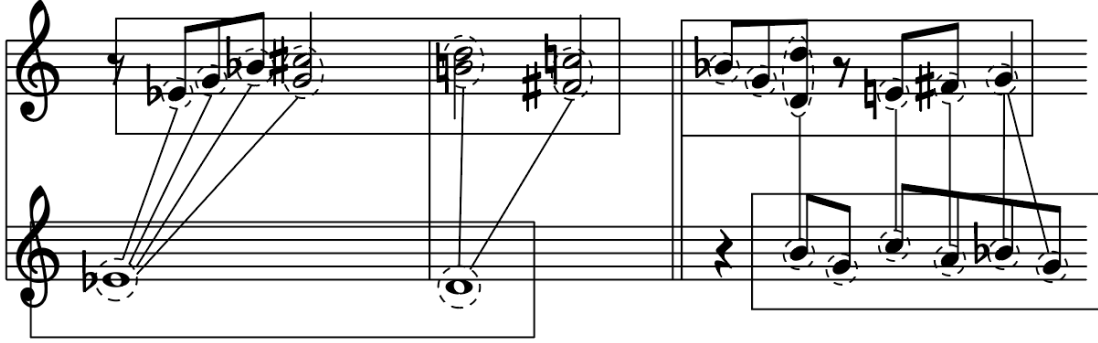


Figure 2.7: Connections drawn in according to the adjacency structure of  $\mathbf{B}$  (between parts).

*Remark 2.2* (Parsimony Pivot connects to everything). Recall that the Parsimony Pivot was defined to overlap in time with every note in the passage  $\mathcal{M}$  (Definition 2.1). By the definition of  $\mathbf{B}$ , this means that the Parsimony Pivot is “close” to *every* event in the passage. This lays the groundwork for our final flow network, where the Parsimony Pivot will be split into a source half and a sink half, with each of these halves connected to every node in the network (loosely, every note in the score).

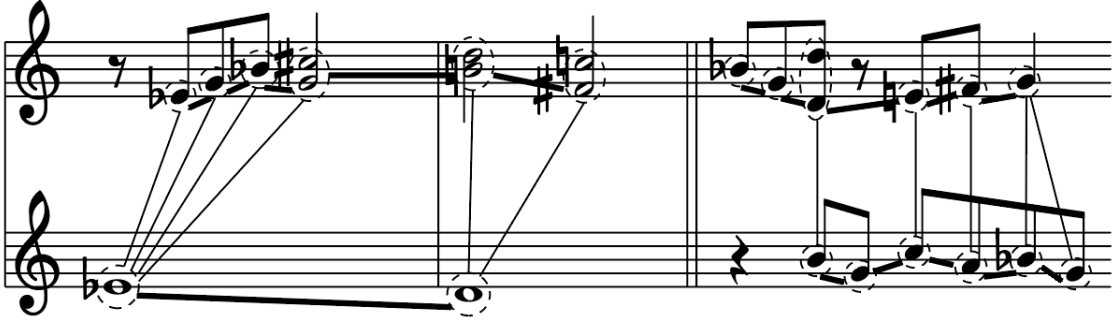


Figure 2.8: Adjacency structure given by  $\alpha \mathbf{A}_1 + \beta \mathbf{B}$  (in other words  $L = 1$ ), for some  $\alpha, \beta$ .  $\alpha \mathcal{E}^{-1}(\mathbf{A}_1) + \beta \mathcal{E}^{-1}(\mathbf{B})$  would be the same but with individualized edges added between events for each of notes that are in the events.

### 2.3.2 Notes

In a similar vein to the work we have already done, we will use a pullback matrix operation applied to a map from a large set onto a smaller set, so that the larger set can inherit the adjacency information of the smaller set. For a note  $n$ , we let  $\mathcal{E}(n)$  denote the event that contains  $n$ .

For free, we get that  $\mathcal{E}^{-1}(\mathbf{A}_k)$  is the matrix that denotes note that are  $k$ -close in the same sub-part, and similarly  $\mathcal{E}^{-1}(\mathbf{B})$  provides connections between notes that are overlapping, in different parts.

For the purposes of the algorithm, we will want to dampen the influence on each other of notes that are further from each other, and we can define an adjacency matrix accordingly. Let  $L$  be the limit of closeness that we will choose to care about (the largest  $k$  such that we define  $k$ -closeness in the score).

$$\mathbf{A}^{\alpha, \beta} = \sum_{k=0}^L \alpha^k \mathcal{E}^{-1}(\mathbf{A}_k) + \beta \mathcal{E}^{-1}(\mathbf{B}) \quad (2.13)$$

where  $0 < \alpha < 1$  and  $\beta$  is a suitably chosen constant to properly characterize the relative importance of close notes in the same parts versus close notes in different parts.

*Remark 2.3.* Recall from Remark 2.2 that all events will be connected to the parsimony pivot in terms of the adjacency structure we have been developing here. The pullback construction in assignment (2.13) ensures that all *noteheads* connect to the Parsimony Pivot.

## 2.4 Flow Network

**Definition 2.6** (Pitch class function). *For every note  $n$  in  $\text{Score}_{\mathcal{M}}$ , let  $\mathcal{C}(n)$  be the pitch class of  $n$ . There are 12 pitch classes,  $\{[C], [C\sharp], [D], \dots, [B]\}$  and we will generally take  $\mathcal{C}(n)$  to be an integer in  $\mathbb{Z}_{12}$ .*

We will always assume that  $\mathcal{M}$  contains its parsimony pivot (definition 2.1).

*Example 2.3.* For the natural choice of  $p_{\mathcal{M}}$  discussed in section 2.1, namely, setting the parsimony pivot to be  $D$  with spelling  $D\flat$ , we have

$$\mathcal{C}(p_{\mathcal{M}}) = 2$$

Since  $[D]$  is third in the 0-indexed list  $([C], [C\sharp], [D], \dots, [B])$ .

### 2.4.1 Nodes

**Definition 2.7** (Coding and decoding). *Let  $V_{\mathcal{M}}$  (think “vertices”) be the set of notes in  $\text{Score}_{\mathcal{M}}$ . We define a set of “nodes”  $N_{\mathcal{M}}$  in terms of  $V_{\mathcal{M}}$  and the following coding and decoding functions (to go back and forth between notes and nodes).*

*Let  $v \in V_{\mathcal{M}}$  and  $b \in \mathbb{Z}_2$ . Then  $n = \langle v, b \rangle$  is the corresponding member of  $N_{\mathcal{M}}$ .*

$$\langle v, b \rangle = 2v + b \tag{2.14}$$

*and conversely, given  $n \in N_{\mathcal{M}}$ ,*

$$(v, b) = (\lfloor n/2 \rfloor, n \bmod 2) \tag{2.15}$$

*fully determining the (bijective) correspondence between  $V_{\mathcal{M}}$  and  $N_{\mathcal{M}}$ , independent of  $\mathcal{M}$ .*

*Remark 2.4.* Hence, we can view every note as being split into two nodes (including the Parsimony Pivot) (see Figure 2.9).



Figure 2.9: Two notes added per node according to the coding and decoding functions.

### 2.4.2 Arcs

We define  $\pi_1$  to be the decoding function composed with a projection in the first coordinate.

$$\pi_1 : \begin{cases} N_{\mathcal{M}} \rightarrow V_{\mathcal{M}} \\ n \mapsto v \quad \text{for } n = \langle v, b \rangle. \end{cases} \quad (2.16)$$

We define  $\pi_2$  similarly in terms of the second coordinate.

$$\pi_2 : \begin{cases} N_{\mathcal{M}} \rightarrow \mathbb{Z}_2 \\ n \mapsto b \quad \text{for } n = \langle v, b \rangle. \end{cases} \quad (2.17)$$

Now we can define  $\langle \mathcal{C} \circ \pi_1, \pi_2 \rangle$  using an analogous coding function to the function  $\langle \rangle : V_{\mathcal{M}} \times \mathbb{Z}_2 \rightarrow N_{\mathcal{M}}$  that we have already seen (definition 2.7). This new coding-decoding pair connects  $\mathbb{Z}_{12} \times \mathbb{Z}_2$  and  $\mathbb{Z}_{24}$  but the same arithmetic works.

$$\langle \mathcal{C} \circ \pi_1, \pi_2 \rangle : \begin{cases} N_{\mathcal{M}} \rightarrow \mathbb{Z}_{12} \times \mathbb{Z}_2 \rightarrow \mathbb{Z}_{24} \\ n \mapsto (\mathcal{C}(v), b) \mapsto \langle \mathcal{C}(v), b \rangle \quad \text{for } n = \langle v, b \rangle. \end{cases} \quad (2.18)$$

We use these functions to gain pulled back adjacency information (see Definition (2.5)).

Let  $\mathbf{\Sigma}$  (“Sigma”) and  $\mathbf{\Omega}$  (“Omega”) be matrices with dimension  $\mathbb{Z}_{24} \times \mathbb{Z}_{24}$ . They will each carry information about arc-weights that depend on pitch classes. We pullback their weight information to add to our arc-weight and adjacency scheme in a way that is pitch-class dependent.

Let  $\{\mathbf{e}_{i,j}\}_{i,j \in N_{\mathcal{M}}}$  be the set of matrix units (1 in the  $(i,j)$ th entry and 0 everywhere else). Define

$$\mathbf{I}_m^c = c \cdot \mathbf{e}_{m,m} + \sum_{i \in N_{\mathcal{M}}: i \neq m} \mathbf{e}_{i,i} \quad (2.19)$$

(Nearly the identity matrix, but for one  $c$ -scaled entry on the diagonal).

By way of example, if  $|N_{\mathcal{M}}| = 4$ , we have

$$\mathbf{I}_2^c = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2.20)$$

$\mathbf{I}_m^c$  is an elementary matrix that performs row scaling. Specifically, as a left multiplier, it scales the  $m$ th row of a matrix by  $c$ . As a right multiplier, it scales the  $m$ th column.

$\mathbf{C}_{\mathcal{M}}$  below is the adjacency matrix of the flow network.

$$\mathbf{C}_{\mathcal{M}} = \mathbf{S}[\mathbf{A} \cdot \cdot \mathbf{W}] \mathbf{T} \mathbf{Z} \quad (2.21)$$

$$\mathbf{S} = \mathbf{I}_{\langle p_{\mathcal{M}}, 1 \rangle}^\varepsilon \quad (2.22)$$

$$\mathbf{A} = \pi_1^{-1}(\mathbf{A}^{\alpha, \beta}) \quad (2.23)$$

$$\mathbf{W} = \langle \mathcal{C} \circ \pi_1, \pi_2 \rangle^{-1}(\mathbf{\Omega}) \cdot \cdot \overline{\pi_1^{-1}(\mathbf{I})} + \langle \mathcal{C} \circ \pi_1, \pi_2 \rangle^{-1}(\mathbf{\Sigma}) \cdot \cdot \pi_1^{-1}(\mathbf{I}) \quad (2.24)$$

$$\mathbf{T} = \mathbf{I}_{\langle p_{\mathcal{M}}, 0 \rangle}^\varepsilon \quad (2.25)$$

$$\mathbf{Z} = \prod_{i \in N_{\mathcal{M}}} \mathbf{I}_i^{\exp(\mathcal{T} \circ \mathcal{E} \circ \pi_1(i)/K)} \quad (2.26)$$

where  $\varepsilon > 0$  is a small constant and  $K$  is a large constant much greater than the largest value that  $\mathcal{T}$  takes on events in  $\mathcal{M}$ .

$\mathbf{A}$  pulls back the adjacency structure developed in Section 2.3 to the scale of arcs between nodes (recall there are twice as many nodes as notes).

$\mathbf{S}$  and  $\mathbf{T}$  capture the limited but non-zero influence of the parsimony pivot in breaking ties, first

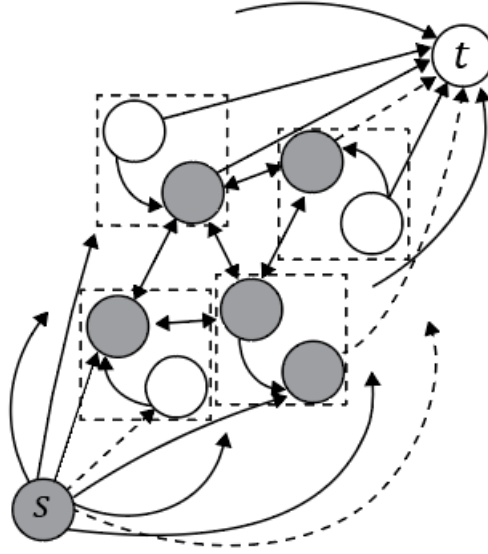


Figure 2.10: The model. Arrows returning to  $s$  and returning from  $t$  are present, but will not be relevant to the minimum cut algorithms addressed later and hence are omitted, for clarity.

discussed after Definition 2.1. This is why we want  $\varepsilon$  to be small but non-zero. We now have the two “halves” of the Parsimony Pivot (labeled  $s$  and  $t$ ) connected to all the internal nodes of the graph as depicted in Figure 2.10. Some nodes are white and some are grey to indicate the different states (0 and 1) that associated decision variables can take in the algorithmic developments that follow (see Chapters 3 and 5).

$\mathbf{Z}$  can also be written as a diagonal matrix, with entry  $(i, i)$  given by  $\exp(\mathcal{T} \circ \mathcal{E} \circ \pi_1(i)/K)$ . From the Taylor series of  $e^x$ , we know that  $e^x \approx 1 + x$ , hence this term nudges the weight of arcs that correspond to later pairs of events upwards. The idea is that these later arcs will have a greater tug than “earlier” arcs in the graph. The idea with this term is to reflect Cambouropoulos’s recommendation, and the third of three preference rules for optimizing over pitch class intervals: “Prefer a sequence in which the *higher ‘quality’ intervals appear last*” – going on to cite music perception studies that have shown that listeners tend to hear the last note of an interval as more prominent, and the example of  $G \ G^\sharp \ A$ , which is a spelling preferable to  $G \ A^\flat \ A^\sharp$  for reasons of voice-leading, but which also places the minor second *last* in the sequence<sup>5</sup>.

---

<sup>5</sup>243 in [3]

$\mathbf{A}$  can be decomposed into the same linearly independent pieces as  $\mathbf{W}$ .

$$\mathbf{A} = \pi_1^{-1}(\mathbf{A}^{\alpha, \beta}) \quad (2.27)$$

$$= \pi_1^{-1} \left( \sum_{k=0}^L \alpha^k \mathcal{E}^{-1}(\mathbf{A}_k) + \beta \mathcal{E}^{-1}(\mathbf{B}) \right) \quad (2.28)$$

$$= (\mathcal{E} \circ \pi_1)^{-1} \left( \sum_{k=0}^L \alpha^k \mathbf{A}_k + \beta \mathbf{B} \right) \cdot \cdot \overline{\pi^{-1}(\mathbf{I})} + (\mathcal{E} \circ \pi_1)^{-1}(\mathbf{A}_0) \cdot \cdot \pi^{-1}(\mathbf{I}) \quad (2.29)$$

In line (2.29) we use the fact that the pullback operation is linear and composes like a regular inverse.  $(\mathcal{E} \circ \pi_1)^{-1}(\mathbf{A}_0)$  ends up in the coefficient of  $\pi^{-1}(\mathbf{I})$  and the coefficient of  $\overline{\pi^{-1}(\mathbf{I})}$ . Since  $(\mathcal{E} \circ \pi_1)^{-1}(\mathbf{A}_0) \cdot \cdot \pi^{-1}(\mathbf{I}) = \pi^{-1}(\mathbf{I})$ , we can rewrite  $\mathbf{A} \cdot \cdot \mathbf{W}$  as the following sum of linearly independent terms.

$$\begin{aligned} \mathbf{A} \cdot \cdot \mathbf{W} &= \left[ (\mathcal{E} \circ \pi_1)^{-1} \left( \sum_{k=0}^L \alpha^k \mathbf{A}_k + \beta \mathbf{B} \right) \cdot \cdot \langle \mathcal{C} \circ \pi_1, \pi_2 \rangle^{-1}(\boldsymbol{\Omega}) \right] \cdot \cdot \overline{\pi_1^{-1}(\mathbf{I})} \\ &\quad + \langle \mathcal{C} \circ \pi_1, \pi_2 \rangle^{-1}(\boldsymbol{\Sigma}) \cdot \cdot \pi^{-1}(\mathbf{I}) \end{aligned} \quad (2.30)$$

## Chapter 3

# Forward Problem

In chapter 2 we defined a weighted directed graph  $\mathbf{Net}_{\mathcal{M}}$  (encoding information present in a passage of music  $\mathcal{M}$ ). In this chapter we present the idea that a spelled score of  $\mathcal{M}$ , which we will call  $\mathbf{Score}_{\mathcal{M}}$ , can be obtained by solving the  $s$ - $t$  minimum cut problem in  $\mathbf{Net}_{\mathcal{M}}$  where  $s = \langle p_{\mathcal{M}}, 1 \rangle$  and  $t = \langle p_{\mathcal{M}}, 0 \rangle$ .<sup>1</sup>

### 3.1 The Minimum Cut Problem Revisited

In the first chapter, we gave the minimum cut problem a somewhat informal introduction. Here we present the problem in one<sup>2</sup> of its linear program forms. Let  $G = (V, E)$  be a flow network ( $V$  is the set of nodes, and  $E$  is the edge relation). We will assume the cost vector (denoting the edge-weights)

---

<sup>1</sup> $p_{\mathcal{M}}$ , the parsimony pivot (definition 2.1), and the natural choice of  $\langle \rangle$  (definition 2.7), are also presented in chapter 2.

<sup>2</sup>The minimum cut problem can also be defined over paths (390 [2]) rather than edges and vertices.

is non-negative, and indexed by edges in  $E$ . Then the MINCUT linear program is defined as follows.

$$\text{MINCUT} = \min_{y, \mathbf{x}} \sum_{e \in E} c_e y_e \quad (3.1)$$

$$\text{subject to} \quad x_j \leq y_{s,j} \quad \forall j : (s, j) \in E \quad (3.2)$$

$$1 - x_i \leq y_{i,t} \quad \forall i : (i, t) \in E \quad (3.3)$$

$$x_j - x_i \leq y_{i,j} \quad \forall (i, j) \in E, i \neq s, j \neq t \quad (3.4)$$

$$x_i \leq 1 \quad \forall i \in V \quad (3.5)$$

$$y_{i,j}, x_i \geq 0 \quad \forall i, j \in V \quad (3.6)$$

Notice, as mentioned that there is no binary restriction on the  $x_i$ . In general, integer programs – linear programs with integrality constraints – are hard to solve efficiently, but the minimum cut problem is special in that it is guaranteed to yield integral solutions even when the explicit condition of integrality of solutions is relaxed [2]. Linear programs can be solved very efficiently in general with the simplex algorithm (with the exception of a few pathological cases), and minimum cut can be solved, in particular, very efficiently with built-to-measure algorithms like Pseudoflow [10, 11].

The fact that optimal solutions will be  $\{0, 1\}$ -vectors even when the integrality conditions of the linear program are relaxed means that we are justified in viewing the problem as one of partitioning the vertices into two sets, and only charging those with arcs from the 0-set to the 1-set (Figure 3.1).

At the minimum, each  $y_{i,j}$  variable will push down as far as it can, binding to the  $x_j - x_i$  constraint or the non-negativity constraint on  $y_{i,j}$ . When  $x_j$  takes value 1 and  $x_i$  takes value 0,  $y_{i,j}$  is forced to take value 1, which means that the cost of edge  $(i, j)$  is incurred in the sum (3.1). For all other assignments to  $x_j$  and  $x_i$ ,  $y_{i,j}$  can drop to 0 in which case the cost of the edge is not incurred. Hence, the equivalence of the linear programming statement of minimum cut and the picture in Figure 3.1 with the caveat that the source  $s$  and the sink  $t$  of the network flow can be thought of as having preassigned value 0 and 1 respectively, resulting in constraints (3.2) and (3.3) respectively.

The following equivalent formulation of minimum cut reveals that  $x_s$  and  $x_t$  can be thought of as decision variables that are fixed in advance of the main algorithm. This should call to mind the definition of a parsimony pivot (Definition 2.1), which has a fixed spelling; this fixed spelling will, in turn, fix the assignment of two  $x_i$ -type decision variables.

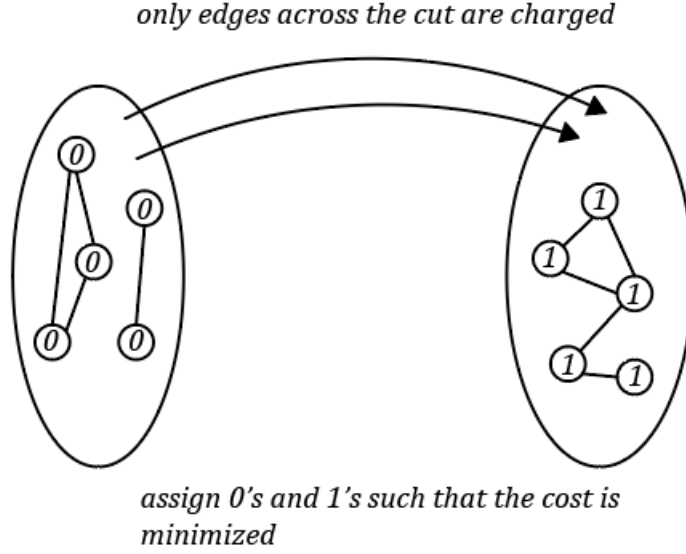


Figure 3.1: Recall this general picture of min cut.

$$\text{Mincut}' = \min_{\mathbf{y}, \mathbf{x}} \sum_{e \in E} c_e y_e \quad (3.7)$$

$$\text{subject to} \quad x_s = 0 \quad \forall j : (s, j) \in E \quad (3.8)$$

$$x_t = 1 \quad \forall i : (i, t) \in E \quad (3.9)$$

$$x_j - x_i \leq y_{i,j} \quad \forall (i, j) \in E \quad (3.10)$$

$$x_i \leq 1 \quad \forall i \in V \quad (3.11)$$

$$y_{i,j}, x_i \geq 0 \quad \forall i, j \in V \quad (3.12)$$

In Figure 3.2, we reprise the definition of  $\mathbf{C}_{\mathcal{M}}$  from Chapter 2.

The matrix  $\mathbf{C}_{\mathcal{M}} = (c_{i,j})$  acts as the cost vector of our linear program, an instantiation of minimum cut. We now introduce the decision variables.

Let  $\mathbf{Y} = (y_{i,j})_{i,j \in N_{\mathcal{M}}}$  be a matrix indexed by pairs of nodes and let  $\mathbf{x} = (x_i)_{i \in N_{\mathcal{M}}}$  be a vector indexed by single nodes. The set of  $y_{i,j}$  differ from the edge based decision variables of minimum cut in the general setting in that  $y_{i,j}$  is defined even when there is no arc  $(i, j) \in A_{\mathcal{M}}$ ; in these cases,  $c_{i,j} = 0$ , and so  $y_{i,j}$  does not contribute to the cost, which means its value is irrelevant to the value

$$\mathbf{C}_{\mathcal{M}} = \mathbf{S}[\mathbf{A} \cdot \cdot \mathbf{W}] \mathbf{T} \mathbf{Z} \quad (3.13)$$

$$\mathbf{S} = \mathbf{I}_{\langle p_{\mathcal{M}}, 1 \rangle}^{\varepsilon} \quad (3.14)$$

$$\begin{aligned} \mathbf{A} \cdot \cdot \mathbf{W} = & \left[ (\mathcal{E} \circ \pi_1)^{-1} \left( \sum_{k=0}^L \alpha^k \mathbf{A}_k + \beta \mathbf{B} \right) \cdot \cdot \langle \mathcal{C} \circ \pi_1, \pi_2 \rangle^{-1}(\boldsymbol{\Omega}) \right] \cdot \cdot \overline{\pi_1^{-1}(\mathbf{I})} \\ & + \langle \mathcal{C} \circ \pi_1, \pi_2 \rangle^{-1}(\boldsymbol{\Sigma}) \cdot \cdot \pi^{-1}(\mathbf{I}) \end{aligned} \quad (3.15)$$

$$\mathbf{T} = \mathbf{I}_{\langle p_{\mathcal{M}}, 0 \rangle}^{\varepsilon} \quad (3.16)$$

$$\mathbf{Z} = \prod_{i \in N_{\mathcal{M}}} \mathbf{I}_i^{\exp(\mathcal{T} \circ \mathcal{E} \circ \pi_1(i)/K)} \quad (3.17)$$

Figure 3.2: Cost matrix, as derived in Chapter 2

of the cut. Hence, the two statements of the problem are equivalent.  $\mathbf{e}$  is an  $|N_{\mathcal{M}}|$  sized vector of 1's, which allows us to wrap the cost into a single sum. As hinted, we take source  $s = \langle p_{\mathcal{M}}, 1 \rangle$  and sink  $t = \langle p_{\mathcal{M}}, 0 \rangle$ .

**Definition 3.1** (MAINLP linear program).

$$\text{MAINLP} = \min_{\mathbf{Y}, x} \mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y} \mathbf{e} \quad (3.18)$$

$$\text{subject to} \quad x_j \leq y_{s,j} \quad \forall j \neq s, j \neq t \quad (3.19)$$

$$1 - x_i \leq y_{i,t} \quad \forall i \neq s, i \neq t \quad (3.20)$$

$$x_j - x_i \leq y_{i,j} \quad \forall c_{(i,j)} > 0, i \neq s, j \neq t \quad (3.21)$$

$$x_i \leq 1 \quad \forall i \in N_{\mathcal{M}} \quad (3.22)$$

$$y_{i,j}, x_i \geq 0 \quad \forall i, j \in N_{\mathcal{M}} \quad (3.23)$$

Recall that in Definition 2.1, we specified that a parsimony pivot should have *definite* predetermined spelling. For concreteness, we proposed that  $p_{\mathcal{M}}$  have pitch class 2, and spelling  $D\sharp$ . As we will see, this is the interpretation of the fixing of  $x_s = 0$  and  $x_t = 1$  in the linear program.

**Proposition 3.1.** *Every  $x \in \mathbb{Z}_2^{|N_{\mathcal{M}}|}$  can be extended to a feasible solution to MAINLP.*

A feasible solution is a solution that is not necessarily optimal but that satisfies all constraints of a linear program. Hence, in the context of MAINLP, a feasible solution is equivalent to an  $s$ - $t$  cut that is not necessarily minimal.

*Proof of Proposition 3.1.* For every  $(i, j)$  take  $y_{i,j} = \max\{0, x_j - x_i\}$ . Hence, constraints (3.19), (3.20), (3.21), (3.22) and (3.23) are all satisfied. The only constraints on the  $x_i$  is that they are in  $[0, 1]$ , which is true of  $x_i \in \mathbb{Z}_2$  (just notation for  $x \in \{0, 1\}$ ).  $\square$

In general, we will use the notation  $\mathbb{Z}_k$  to denote the set of integers  $\{0, 1, 2, \dots, k-1\}$ .

## 3.2 Interpreting the Linear Program

### 3.2.1 Encoding Spellings

There are at most three spellings per note in a score, since every note has a pitch in one of the twelve pitch classes, and, as Figure 1.3 shows, there are at most three classes of spellings for each pitch class. We can categorize pitch classes by the spellings they permit (see Table 3.1). We can view the selection of an accidental  $a$  set from the  $i$ th spelling set  $\mathcal{S}_j$  (which determines a spelling), as an assignment of a pair of binary predicates  $\phi_0^j$  and  $\phi_1^j$ . Various sets of predicates are possible to capture the variety of spellings. Our approach will be to select  $\Phi = (\phi_0^0, \phi_1^0, \phi_0^1, \phi_1^1, \dots, \phi_0^4, \phi_1^4, \phi^5)$  such that 1's generally correspond to sharp spellings and 0's generally correspond to flat spellings. We can render this as a formal minimization as in objective 3.24.  $\phi^5$  is a special case, because category 5 notes only permit two possible spellings (see Table 3.1).

$$\max_{\Phi} \sum_{j \in [5]} \left[ \sum_{a \in \mathcal{S}_j \cap \text{SHARPS}} \phi_0^j(a) + \phi_1^j(a) - \sum_{a \in \mathcal{S}_j \cap \text{SHARPS}^C} \phi_0^j(a) + \phi_1^j(a) \right] + \phi^5(\sharp) - \phi^5(\flat) \quad (3.24)$$

where  $\text{SHARPS} = \{\sharp, \times\}$  and  $\text{SHARPS}^C = \{\flat, \flat\flat, \flat\flat\flat\}$ . In a tie, we look for spellings such that  $\phi_0^j = \phi_1^j$  when the spelling is a double sharp or a double flat.

$$\max_{\Phi} \sum_{j \in [5]} \left[ \sum_{a \in \mathcal{S}_j \cap \text{DOUBLES}} |\phi_0^j(a) - \phi_1^j(a)| \right] \quad (3.25)$$

where  $\text{DOUBLES} = \{\flat\flat, \times\}$ . in fact, there is a maximizer for objective 3.25 that is also a maximizer for 3.24, so we could equally have combined them into a single objective (the sum) and obtained the same result. One such maximizer<sup>3</sup> is outlined in Table 3.2.

<sup>3</sup>There are still some degrees of freedom in choosing  $\Phi$ , whereby, for example, we could exchange the roles of  $\phi_0^0$  and  $\phi_1^0$ , as they are given in Table 5.2c and still achieve a maximum. For all remaining discussions, we will assume that we have stuck to the assignment laid out in 3.2.

| Spelling-Set                                | Pitch Classes | Category |
|---------------------------------------------|---------------|----------|
| $\mathcal{S}_0 = \{\sharp, \#, \flat\}$     | 0, 5          | 0        |
| $\mathcal{S}_1 = \{\#, \flat, \times\}$     | 1, 6          | 1        |
| $\mathcal{S}_2 = \{\times, \flat, \sharp\}$ | 2, 7, 9       | 2        |
| $\mathcal{S}_3 = \{\flat, \#, \flat\}$      | 3, 10         | 3        |
| $\mathcal{S}_4 = \{\flat, \sharp, \times\}$ | 4, 11         | 4        |
| $\mathcal{S}_5 = \{\#, \flat\}$             | 8             | 5        |

Table 3.1: Partitioning pitch classes by permissible spelling set

|                                                                                                                                                              |                                                                                                                                                                  |                                                                                                                                                                  |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $ \begin{array}{c c c} a \in \mathcal{S}_0 & \phi_0^0 & \phi_1^0 \\ \hline \sharp & 0 & 1 \\ \# & 1 & 1 \\ \flat & 0 & 0 \end{array} $ <p>(a) Category 0</p> | $ \begin{array}{c c c} a \in \mathcal{S}_1 & \phi_0^1 & \phi_1^1 \\ \hline \# & 1 & 0 \\ \flat & 0 & 0 \\ \times & 1 & 1 \end{array} $ <p>(b) Category 1</p>     | $ \begin{array}{c c c} a \in \mathcal{S}_2 & \phi_0^2 & \phi_1^2 \\ \hline \times & 1 & 1 \\ \flat & 0 & 0 \\ \sharp & 1 & 0 \end{array} $ <p>(c) Category 2</p> |
| $ \begin{array}{c c c} a \in \mathcal{S}_3 & \phi_0^3 & \phi_1^3 \\ \hline \flat & 0 & 0 \\ \# & 1 & 1 \\ \flat & 0 & 1 \end{array} $ <p>(d) Category 3</p>  | $ \begin{array}{c c c} a \in \mathcal{S}_4 & \phi_0^4 & \phi_1^4 \\ \hline \flat & 0 & 0 \\ \sharp & 1 & 0 \\ \times & 1 & 1 \end{array} $ <p>(e) Category 4</p> | $ \begin{array}{c c} a \in \mathcal{S}_5 & \phi^5 \\ \hline \# & 1 \\ \flat & 0 \end{array} $ <p>(f) Category 5</p>                                              |

Table 3.2: The set of predicates that minimizes objectives 3.24 and 3.25, divided up by category (see Table 3.1).

### 3.2.2 Checking Validity

We want  $\mathbf{x}$  to be interpreted in the sense that would allow us to use the information in tables 3.2 and 3.1. The following procedure can be thought of as a formal description of how one might check the validity of  $x$  using the predicate tables and all the information we have built up so far.

**Definition 3.2** (Spell-check). *We give a procedure for checking with  $x$  corresponds to a valid spelling.*

*For each  $v \in V_{\mathcal{M}}$  such that  $\mathcal{C}(v) \neq 8$ :*

*$k \leftarrow$  the pitch class category of  $\mathcal{C}(v)$  from table 3.1.*

*$b \leftarrow \text{Check if } (x(\langle v, 0 \rangle), x(\langle v, 1 \rangle)) \text{ is a row of table 3.2, category } k.$*

*If  $b$  is TRUE: continue. Else: answer: “no”*

*end For.*

*answer: “yes”*

The steps of the procedure are a little more involved than we would like. Fortunately, there is a simple enough closed form statement that we can use henceforward to verify that an assignment  $x$

is meaningful as a spelling.

**Definition 3.3** (Binding to spellings). *Partition the vertices of  $\text{Adj}_{\mathcal{M}}$  by pitch class. Namely, let  $V_{\mathcal{M}}^i = \{v \in V_{\mathcal{M}} : \mathcal{C}(v) = i\}$  for each  $i \in \mathbb{Z}_{12}$ .*

*$\mathbf{x} \in \mathbb{Z}_2^{|N_{\mathcal{M}}|}$  binds to a valid spelling if  $\mathbf{x}$  satisfies the following formula:*

$$\bigwedge_{v \in V, \mathcal{C}(v) \in I} x(\langle v, 0 \rangle) \rightarrow x(\langle v, 1 \rangle) \quad \wedge \quad \bigwedge_{v \in V, \mathcal{C}(v) \in J} x(\langle v, 1 \rangle) \rightarrow x(\langle v, 0 \rangle) \quad (3.26)$$

where  $I = \{0, 3, 5, 10\}$  and  $J = \{1, 2, 4, 6, 7, 9, 11\}$ .

*In such a case, we say that  $\mathbf{x}$  is spell-binding.*

**Proposition 3.2.**  *$\mathbf{x} \in \mathbb{Z}_2^{|N_{\mathcal{M}}|}$  is spell-binding if and only if the spell-check procedure ends in “yes”.*

*Proof.* Formula (3.26) evaluates to false when there exists  $v \in V$  with  $(x(\langle v, 0 \rangle), x(\langle v, 1 \rangle)) = (1, 0)$  for  $\mathcal{C}(v) \in I$  or  $(x(\langle v, 0 \rangle), x(\langle v, 1 \rangle)) = (0, 1)$  for  $\mathcal{C}(v) \in J$ . These are precisely the rows missing from the Tables that represent pitch classes from  $I$  and pitch classes from  $J$  respectively.  $\square$

The spell-check procedure is still useful in that it can be used to tell us the precise interpretation of the value of the  $x_i$  as *spellings*, by reading off what the interpretation of each pair of predicates means in the appropriate table. The spell-binding criterion meanwhile is a more tangible means of determining whether or not the  $x_i$  are meaningful as a spelling at all.

*Remark 3.1* (The Parsimony Pivot yields a source and a sink). Let us use the table for category 2, to interpret the binary encoding of the parsimony pivot’s spelling  $D\sharp$ . According to the table, this should correspond to having  $x(\langle p_{\mathcal{M}}, 0 \rangle) = 1$  and  $x(\langle p_{\mathcal{M}}, 1 \rangle) = 0$ , hence, the interpretation of  $\langle p_{\mathcal{M}}, 1 \rangle$  as a source node and  $\langle p_{\mathcal{M}}, 0 \rangle$  as a sink (the nodes’ for which the binary assignment is initially fixed), bearing in mind our discussion of MINCUT’, (3.7) - (3.12).

### 3.2.3 Specifying Sigma

As Equation (3.15) shows, the  $24 \times 24$  matrix  $\Sigma$  determines controls the arc-weights between nodes that are pulled back from the same note i.e. on arcs  $(n, m)$  where  $n = \langle u, b_0 \rangle$ ,  $m = \langle v, b_1 \rangle$  and  $u = v$ . Whenever  $u = v$  in this way,  $\mathcal{C}(u) = \mathcal{C}(v)$  so we only need to define  $\Sigma$  along its block diagonal, with indices of the form  $(\langle c, b_0 \rangle, \langle c, b_0 \rangle)$ , where  $c$  is a pitch class (in  $\mathbb{Z}_{12}$  and  $b_0, b_1 \in \mathbb{Z}_2$ ).

We can characterize  $\Sigma$  “casewise” using a pullback by a function  $\mathcal{K} : \mathbb{Z}_{12} \rightarrow \mathbb{Z}_3$ , which we define below. We need to assign large arc-weights to the flow network on arcs that should be strongly

avoided in the search for a minimum cut, in order for the resulting assignment to be definitiely spell-binding. We follow the standard linear programming convention of denoting such a large cost parameter with the letter  $M$  (“big  $M$ ”). The size of  $M$  needed used will depend in practice on the other values of the cost vector (yet to be defined). Alternatively,  $M$  can be defined as a formal value that “wins” in every  $\geq$ -comparison [2].

$$\mathcal{K} : \begin{cases} i \mapsto 0 & i \in \{0, 3, 5, 10\} \\ j \mapsto 1 & j \in \{1, 2, 4, 6, 7, 9, 11\} \\ 8 \mapsto 2 \end{cases} \quad (3.27)$$

Hence, using the notational convention from Chapter 2,  $\langle \mathcal{K} \circ \pi_1, \pi_2 \rangle$  has a specific meaning as a function.

$$\langle \mathcal{K} \circ \pi_1, \pi_2 \rangle(7) = \langle \mathcal{K} \circ \pi_1, \pi_2 \rangle(\langle 3, 1 \rangle) \quad (3.28)$$

$$= \langle \mathcal{K}(3), 1 \rangle \quad (3.29)$$

$$= \langle 0, 1 \rangle \quad (3.30)$$

$$= 1 \quad (3.31)$$

and so on, similarly for other inputs in  $\mathbb{Z}_{24}$ . Thus, we can take a pullback using  $\langle \mathcal{K} \circ \pi_1, \pi_2 \rangle$ , as in Figure 3.3.

**Proposition 3.3.** *If  $(x^*, \mathbf{Y}^*)$  is a basic feasible optimal solution to MAINLP and  $M$  is sufficiently large then  $x^*$  is spell-binding.<sup>4</sup>*

*Proof.* Assume that  $x^*$  is not spell-binding. We will show that it cannot be optimal.

There must be an atomic formula in equation (3.26) that evaluates to 0. Without loss of generality assume it comes from the set of  $v$  such that  $\mathcal{C}(v) \in I$  (the left term) and let  $v$  be the witnessing vertex. So we have  $x^*(\langle v, 0 \rangle) = 1$  but  $x^*(\langle v, 1 \rangle) = 0$ . Then if  $j = \langle v, 0 \rangle$ ,  $i = \langle v, 1 \rangle$ , we have  $c_{i,j} = M$  and  $y_{i,j}^* \geq 1$ . Hence,  $c_{i,j}$  is charged in the cost  $\mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y}^* \mathbf{e}$  and a change to the value of  $x^* \rightarrow x'$  at  $\langle v, 1 \rangle$  will strictly lower the cost ( $M$  is as big as it needs to be to ensure this is true). Since  $x''$  can definitely be extended to a feasible solution to MAINLP,  $(x'', \mathbf{Y}'')$  (Proposition (3.1)),  $(x'', \mathbf{Y}'')$  has

---

<sup>4</sup>We will assume that there are no arcs of weight remotely close to  $M$  specified by  $\Omega$ . We can guarantee this by putting a bound on these arc-weights of say 1.

$$\Sigma = \langle \mathcal{K} \circ \pi_1, \pi_2 \rangle^{-1} \begin{pmatrix} 0 & 0 \\ M & 0 \\ & 0 & M \\ & 0 & 0 \\ & & 0 & 0 \\ & & 0 & 0 \end{pmatrix} \quad (3.32)$$

$$= \begin{pmatrix} \begin{matrix} 0 & 0 \\ M & 0 \end{matrix} & & & & & & & & & & \\ & \begin{matrix} 0 & M \\ 0 & 0 \end{matrix} & & & & & & & & & \\ & & \begin{matrix} 0 & M \\ 0 & 0 \end{matrix} & & & & & & & & \\ & & & \begin{matrix} 0 & 0 \\ M & 0 \end{matrix} & & & & & & & \\ & & & & \begin{matrix} 0 & M \\ 0 & 0 \end{matrix} & & & & & & \\ & & & & & \begin{matrix} 0 & 0 \\ M & 0 \end{matrix} & & & & & \\ & & & & & & \begin{matrix} 0 & M \\ 0 & 0 \end{matrix} & & & & \\ & & & & & & & \begin{matrix} 0 & M \\ 0 & 0 \end{matrix} & & & \\ & & & & & & & & \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} & & \\ & & & & & & & & & \begin{matrix} 0 & M \\ 0 & 0 \end{matrix} & \\ & & & & & & & & & & \begin{matrix} 0 & M \\ 0 & 0 \end{matrix} \end{pmatrix} \quad (3.33)$$

Figure 3.3

strictly lower cost than  $(x^*, \mathbf{Y}^*)$  and is feasible. Thus  $(x^*, \mathbf{Y}^*)$  was not optimal after all.  $\square$

The big  $M$  edges then, can be viewed as enforcing the spell-binding condition in the search for an optimal solution to the MAINLP problem.

### 3.2.4 Dual Interpretation

In the approach to pitch spelling we take in this thesis, we have a problem that motivates minimum cut as a *primal* problem. In fact, whereas the maximum flow between two nodes in a network usually has a natural interpretation - in the present context, the meaning of the maximum flow of the network is not so clear! An interpretation of the maximum flow, could, however, provide deeper insights into the structure of the pitch-spelling problem in future work.

## Chapter 4

# Inverse Problem

The setup of the inverse problem is as follows. We have a large set of musical passages that are well-spelled as scores. We call the large set of scores **corpus**. From the **corpus** set, we wish to obtain a good estimate for what arc-weights would have resulted in the spellings of the scores in the **corpus** set. Recall that  $\mathbf{C}_{\mathcal{M}}$  is defined in terms of a pullback on the  $24 \times 24$  matrix  $\mathbf{\Omega}$ , as yet undefined. The outcome of the following linear program (once solved), will then be a concrete rendering of  $\mathbf{\Omega}$ .

### 4.1 The Maximum Flow Problem Revisited

In the first chapter, we gave an informal introduction to the maximum flow problem. In Chapter 3 we gave the formal definition of the minimum cut problem. The picture of a maximum flow is recalled here (Figure 4.1). Here we do the same for the maximum flow problem. Let  $G(V, E)$  be a flow network ( $V$  is the set of nodes, and  $E$  is the edge relation). We will assume the capacity vector (denoting edge-weights) is non-negative, and indexed by edges in  $E$ . Then the MAXFLOW linear program is defined as follows.<sup>1</sup>

---

<sup>1</sup>In addition to the Pseudoflow algorithm [10, 11], there are a number of well-known algorithms for solving max-flow as a primal problem [7, 2]

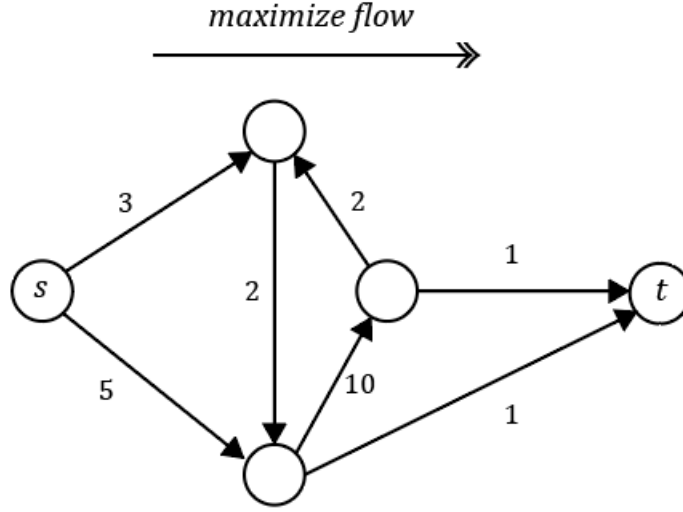


Figure 4.1: Recall: an instantiation of the max flow problem

$$\text{MAXFLOW} = \max_f \sum_{j:(s,j) \in E} f_{(s,j)} \quad (4.1)$$

$$\text{subject to} \quad \sum_{i:(i,v) \in E} f_{(i,v)} = \sum_{j:(v,j) \in E} f_{(v,j)} \quad \forall v \in V \setminus \{s, t\} \quad (4.2)$$

$$f_e \leq c_e \quad \forall e \in E \quad (4.3)$$

$$f_e \geq 0 \quad \forall e \in E \quad (4.4)$$

Recall that we wish to maximize the flow from node  $s$  to node  $t$ . Flow is a conserved quantity through the network, hence, we insist that the amount of flow entering a node is the same as the amount of flow exiting a node (for all internal nodes). This is what condition (4.2) enforces. Another outcome of the flow conservation is that we can describe the flow from  $s$  to  $t$  equivalently as the amount of flow exiting  $s$  or the amount of flow entering  $t$ . Hence the objective (4.1) is stated as the amount of flow exiting  $s$ . Constraint (4.3) enforces the capacity interpretation of edge weights, whereby the flow across an edge cannot exceed the weight of the edge. Finally enforces the nonnegativity constraint that no flow across an edge can be negative.

Notice that we can apply the *min cut/max flow theorem* (Section 1.3.1) to state an equivalent linear program (LP). Let  $(x^*, y^*)$  be a solution to the MINCUT linear program (Section 3.1). The

objective is redundant, so we just write the trivial objective of maximizing a constant, which has no effect on the outcome of the optimization. Any *feasible* solution to the following linear program will be a maximum flow in  $G$ , since any feasible solution is a flow equal to the minimum cut.

$$\max_f 0 \tag{4.5}$$

$$\text{subject to} \quad \sum_{j:(s,j) \in E} f_{(s,v)} = \sum_{e \in E} c_e y_e^* \tag{4.6}$$

$$\sum_{i:(i,v) \in E} f_{(i,v)} = \sum_{j:(v,j) \in E} f_{(v,j)} \quad \forall v \in V \setminus \{s, t\} \tag{4.7}$$

$$f_e \leq c_e \quad \forall e \in E \tag{4.8}$$

$$f_e \geq 0 \quad \forall e \in E \tag{4.9}$$

Now let  $(x, y)$  be any feasible solution to the MinCut linear program (not necessarily optimal). Optimal solutions to the following linear program are also maximum flows. The objective of the linear program is to minimize the “duality gap” between a feasible flow and a feasible cut. By the *max flow/min cut theorem*, this duality gap must be nonnegative. It is minimized when the flow is maximized since the cost of the cut is fixed.

$$\min_{f, \Delta} \Delta \tag{4.10}$$

$$\text{subject to} \quad \Delta + \sum_{j:(s,j) \in E} f_{(s,v)} = \sum_{e \in E} c_e y_e \tag{4.11}$$

$$\sum_{i:(i,v) \in E} f_{(i,v)} = \sum_{j:(v,j) \in E} f_{(v,j)} \quad \forall v \in V \setminus \{s, t\} \tag{4.12}$$

$$f_e \leq c_e \quad \forall e \in E \tag{4.13}$$

$$f_e \geq 0 \quad \forall e \in E \tag{4.14}$$

These versions of the maximum flow problem will be useful as we construct new linear programs in this section that enable us to solve the “inverse problem” (how do we populate the matrix  $\Omega$ ?).

## 4.2 Characterizing Omega

Recall from the previous chapters that  $\Omega$  is a matrix whose  $(i, j)$ th entry expresses the pull that  $i$  and  $j$  exert on each other in the search for a minimum cut, where  $i$  and  $j$  are associated with a specific pitch class. We have seen before how we can look at a reduced set of categories of pitch classes and then use a pullback to gain information on a larger scale. We do the same thing here. Our goal is to have a  $k \times k$  matrix  $\mathbf{M}$ , where  $k < 24$ , so that through a pullback in terms of the categorizer  $\mathcal{R}$ , we can obtain  $\Omega$ . There are a few additional mysterious terms in the following equation, which we will explain shortly.

$$\Omega = \langle \mathcal{R} \circ \pi_1, \pi_2 \rangle^{-1}(\mathbf{M}) \cdot \overline{\pi_1^{-1}(\Delta_6)} \quad (4.15)$$

where  $\Delta_6$  is the matrix with entry  $i, j$  equal to 1 if  $i - j \equiv_{12} 6$  and 0 otherwise (a cyclic permutation of the identity matrix).

*Remark 4.1* (A discussion of the more mysterious terms in assignment (4.16)). First, the pullback is in terms of the function  $\langle \mathcal{R} \circ \pi_1, \pi_2 \rangle$  rather than  $\mathcal{R}$ , because we want to keep the binary index constant in the pullback (see Section 2.4.2 for a description of what is meant by the notation featuring projections  $\pi_1, \pi_2$  and angle-brackets  $\langle \rangle$ ). Second, there are good reasons to argue that pitch classes 6 half-steps apart should have no particular pull on each other's spellings. A good indication of this comes from Cambouropoulos's hierarchy of intervals, which gives equal weight to "augmented fourths" and "diminished fifths" (the two most plausible 6 half-step intervals – see Table 1.5), in terms of their incidence in major and minor scales [4, 3]. Similar arguments would be possible in terms of any of the models or "Search Spaces" addressed in the first chapter.

In fact, we will not expect  $M$  to pull back and yield  $\Omega$  exactly. Rather, we will use the construction in assignment (4.16) to generate an upper bound on  $\Omega$ , which we will call  $\Gamma$  (Gamma).

$$\Gamma = \langle \mathcal{R} \circ \pi_1, \pi_2 \rangle^{-1}(\mathbf{M}) \cdot \overline{\pi_1^{-1}(\Delta_6)} \quad (4.16)$$

Now with  $\Omega = (\omega_{i,j})$  and  $\Gamma = (\gamma_{i,j})$ , we will insist that each entry of  $\Omega$  are bounded by the corresponding entry of  $\Gamma$ :

$$\omega_{i,j} \leq \gamma_{i,j} \quad (4.17)$$

for each  $i, j$ .

The motivation for this system of upper bounds is that we would actually like to obtain  $\Omega$  as a

maximization over some linear constraints.  $\mathbf{\Gamma}$  has two roles.

1. It fixes  $\mathbf{\Omega}$  at a certain magnitude, where in general, any scalar multiple of a given optimal solution might also be optimal.
2. It clamps certain entries of  $\mathbf{\Omega}$  to 0, when we can be certain ahead of time, through our knowledge of the musical notation system, that such an entry *must* be 0. An example of this point follows.

*Example 4.1* (An example of the second point). The pitch class 7, spelled as  $G$ , “wants” the pitch class 5 to be spelled as  $F$ , and not  $E^\sharp$  or  $G^\flat$ . Once again, this can be argued with the help of any of the search spaces in the first chapter, or through an application of musical “common sense”.

Suppose now that the entry of  $\mathbf{\Omega}$  at  $(\langle 7, 0 \rangle, \langle 5, 0 \rangle)$  is nonzero; suppose, for the sake of argument, that it is in fact quite large. Now let us say we have a score with a note  $v$ , such that its pitch class  $\mathcal{C}(v) = 5$  and a note  $u$  such that its pitch class  $\mathcal{C}(u) = 7$ , and suppose that  $u$  and  $v$  are close to each other so that they are connected in the flow network of the passage of music,  $\text{Net}_{\mathcal{M}}$ . In the context of the passage, the cost matrix  $\mathbf{C}_{\mathcal{M}}$ , will have a large value at  $(\langle u, 0 \rangle, \langle v, 0 \rangle)$  and likewise at  $(\langle v, 0 \rangle, \langle u, 0 \rangle)$  by the construction of  $\mathbf{C}_{\mathcal{M}}$  (given in Section 2.4.2).

Hence, our solver of minimum cut will seek to exclude  $c_{\mathcal{M}}(\langle v, 0 \rangle, \langle u, 0 \rangle)$  and  $c_{\mathcal{M}}(\langle u, 0 \rangle, \langle v, 0 \rangle)$  from the cost of the cut, by pushing  $y_{\mathcal{M}}(\langle u, 0 \rangle, \langle v, 0 \rangle)$  and  $y_{\mathcal{M}}(\langle v, 0 \rangle, \langle u, 0 \rangle)$  to 0, or in other words, pushing  $x_{\mathcal{M}}(\langle u, 0 \rangle)$  and  $x_{\mathcal{M}}(\langle v, 0 \rangle)$  to be the same, 1 or 0. If  $x_{\mathcal{M}}(\langle u, 0 \rangle) = 1$ , the one spelling available for  $u$  is  $E^\sharp$ , while  $x_{\mathcal{M}}(\langle v, 0 \rangle) = 1$  leaves  $v$  with  $F^\times$  and  $G^\flat$  as options. If  $x_{\mathcal{M}}(\langle v, 0 \rangle) = 0$ , the one spelling available for  $u$  is  $A^\flat$ , while  $x_{\mathcal{M}}(\langle u, 0 \rangle) = 0$  leaves  $u$  with  $F$  and  $G^\flat$  as options.

Although some of the options are passable in that the spellings are both  $\flat$ -type in one case and  $\times$ -type in the other, it is obviously undesirable that these are promoted at the expense of the natural parsimonious spelling of  $F$  with  $G$ . Even if the entry of  $\mathbf{\Omega}$  at  $(\langle 7, 0 \rangle, \langle 5, 0 \rangle)$  is not large, but still nonzero, it will still exert a pull away from the natural choice for the spelling of these two pitch classes.

So we conclude that the entries of  $\mathbf{\Omega}$  at  $(\langle 7, 0 \rangle, \langle 5, 0 \rangle)$  and  $(\langle 5, 0 \rangle, \langle 7, 0 \rangle)$  must be 0. We can show through a similar argument that the entries of  $\mathbf{\Omega}$  at  $(\langle 7, 1 \rangle, \langle 5, 1 \rangle)$  and  $(\langle 5, 1 \rangle, \langle 7, 1 \rangle)$  should also be 0. Therefore, we use  $\mathbf{\Gamma}$  to clamp  $\mathbf{\Omega}$  to 0 at these entries.

The same argument does not apply for the value of  $\mathbf{\Omega}$  at either  $(\langle 7, 0 \rangle, \langle 5, 1 \rangle)$  and  $(\langle 5, 1 \rangle, \langle 7, 0 \rangle)$ , or  $(\langle 7, 1 \rangle, \langle 5, 0 \rangle)$  and  $(\langle 5, 0 \rangle, \langle 7, 1 \rangle)$ . When  $x_{\mathcal{M}}(\langle v, 1 \rangle) = 1$ ,  $v$  can take spellings  $E^\sharp$  or  $F^\flat$ . When

$c_{\mathcal{M}}(\langle u, 0 \rangle) = 1$ ,  $u$  can take spellings  $G\sharp$  and  $F\sharp$ , which means there is always a way to preserve the major second interval, regardless of the spelling of  $v$  (see Table 1.4 for a reminder of musical intervals). When  $x_{\mathcal{M}}(\langle v, 1 \rangle) = 0$ ,  $v$  must take spelling  $G\flat$ , while when  $x_{\mathcal{M}}(\langle u, 0 \rangle) = 0$ ,  $u$  must take spelling  $A\flat$ , which, again preserves the major second interval.

Hence the upper bound matrix  $\mathbf{\Gamma}$  should allow  $\mathbf{\Omega}$  to take non-negative values at  $(\langle 7, 0 \rangle, \langle 5, 1 \rangle)$ ,  $(\langle 5, 1 \rangle, \langle 7, 0 \rangle)$ ,  $(\langle 7, 1 \rangle, \langle 5, 0 \rangle)$  and  $(\langle 5, 0 \rangle, \langle 7, 1 \rangle)$ .

A similar set of arguments to those given in the above example can be applied for all pairs of pitch classes. There are of course a lot of cases to go through, but the principles are the same in each case. We can cut the number of pairs to check down by breaking the pitch classes into subsets and using a pullback over the partitioning. As a starting point, let us argue that 0 and 5 can be treated the same way in building up  $\mathbf{\Gamma}$ . Suppose we replaced 5 with 0 in the above argument. The reasoning would be almost exactly the same, since 0 has the same permissible spelling set, which I called  $\mathcal{S}_0$  in Table 3.1 ( $\mathcal{S}_0 = \{\sharp, \flat, \natural\}$ ). In fact, we could replace 7 in the above argument with any pitch class except 11 and 6, and the reasoning would be the same for 5 and 0. 11 and 6 are special because  $5 + 6 = 11$ , and  $0 + 6 = 6$ , hence the  $\overline{\pi_1^{-1}(\Delta_6)}$  term in (4.16), will completely zero out the terms of  $\mathbf{\Gamma}$  at  $(\langle 5, b_1 \rangle, \langle 11, b_2 \rangle)$  (for  $b_1, b_2 \in \mathbb{Z}_2$ ) and likewise for edges of the form  $(\langle 0, b_1 \rangle, \langle 6, b_2 \rangle)$ . Therefore we need only 1 row and 1 column of  $\mathbf{M}$  (in (4.16)) that will account for the upper bounds over indices featuring pitch class 5 and pitch class 0.

We can go through and group elements with the same spelling set that are separated by 5 half-steps *modulo* 12. So that the same reasoning applies. There will be two pitch classes left over: as ever, 8 is the odd one out, and 7 is a special case, because it comes from a spelling set with 3 and not 2 elements.

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (4.20)$$

Figure 4.2

We obtain the following function  $\mathcal{R}$  with which to arrange a pullback.

$$\mathcal{R}(i) = \begin{cases} 0 & i \in \{0, 5\} \\ 1 & i \in \{1, 6\} \\ 2 & i \in \{2, 9\} \\ 3 & i \in \{7\} \\ 4 & i \in \{3, 11\} \\ 5 & i \in \{4, 10\} \\ 6 & i \in \{8\} \end{cases} \quad (4.18)$$

Through repeated application of the arguments in the Example 4.1, and letting  $\mathbf{\Gamma}$  take value 1 when it is not 0, we arrive at the matrix  $\mathbf{M}$  (Figure 4.2). Notice, for example, the fourth<sup>2</sup> of the  $2 \times 2$  matrices in the top row, and the fourth of the  $2 \times 2$  matrices in the first column, which both look as shown:

$$\begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array} \quad (4.19)$$

We have a zero where both binary indices are 0 (within this  $2 \times 2$  matrix), a zero where both binary indices are 1, and 1's elsewhere, just as was argued in Example 4.1.

---

<sup>2</sup>0-indexed third, because  $\mathcal{R}(7) = 3$ .

Hence, we obtain  $\mathbf{\Gamma}$  in Figure 4.3.

$$\mathbf{\Gamma} = \langle \mathcal{R} \circ \pi_1, \pi_2 \rangle^{-1}(\mathbf{M}) \cdot \overline{\pi_1^{-1}(\mathbf{\Delta}_6)} \quad (4.21)$$

$$= \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & \mathbf{0} & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ \mathbf{0} & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ \mathbf{0} & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & \mathbf{0} & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & \mathbf{0} & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & \mathbf{0} & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \dots & \dots & \dots & \dots & \dots & 0 & 0 & \dots & \dots & \dots & \mathbf{0} & \dots & \dots & \mathbf{0} \\ 0 & 1 & 1 & 0 & 1 & 0 & \mathbf{0} & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & \mathbf{0} & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & \mathbf{0} & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & \mathbf{0} & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \mathbf{0} & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \quad (4.22)$$

Figure 4.3

### 4.3 Exact Inverse

We now go about constructing the linear program that promises to extract values of  $\mathbf{\Omega}$ . We will need to interpret the data of a pre-spelled score in terms of the statement of MAINLP (Definition 3.1).

**Definition 4.1** (Lifted  $s$ - $t$  cut). *Suppose  $\text{Score}_{\mathcal{M}}$  is the spelled score of a passage of music  $\mathcal{M}$ . Let  $x \in \mathbb{Z}_2^{|\mathcal{N}_{\mathcal{M}}|}$  be the vector that binds to the spelling of  $\text{Score}_{\mathcal{M}}$ , then define  $Y_{\mathcal{M}}^{\text{score}} = (y_{i,j})$  by  $y_{i,j} = x_j - x_i$ . We say  $Y_{\mathcal{M}}^{\text{score}}$  is the  $s$ - $t$  cut lifted from  $\text{Score}_{\mathcal{M}}$ .*

Suppose for every  $\text{Score}_{\mathcal{M}}$  in corpus,  $\mathbf{Y}_{\mathcal{M}}^{\text{score}}$  is defined to be the  $s$ - $t$  lifted from  $\text{Score}_{\mathcal{M}}$ . Then we

can define the following linear program. We notate the matrices as  $\mathbf{\Gamma} = (\gamma_{i,j})$  (known),  $\mathbf{\Omega} = (\omega_{i,j})$ ,  $\mathbf{F}_{\mathcal{M}} = (f_{\mathcal{M}}(i,j))$ ,  $\mathbf{C}_{\mathcal{M}} = (c_{\mathcal{M}}(i,j))$  (variables). As usual,  $\mathbf{e}$  denotes a column vector of only 1's.

**Definition 4.2** (EXACTINVERSE linear program).

$$\text{EXACTINVERSE} = \max_{\mathbf{\Omega}, \{\mathbf{C}_{\mathcal{M}}\}, \{\mathbf{F}_{\mathcal{M}}\}} \mathbf{e}' \mathbf{\Omega} \mathbf{e} \quad (4.23)$$

subject to

$$\sum_{i:(i,k) \in A_{\mathcal{M}}} f_{\mathcal{M}}(i,k) = \sum_{j:(k,j) \in A_{\mathcal{M}}} f_{\mathcal{M}}(k,j) \quad k \in \pi_1^{-1}(V_{\mathcal{M}} \setminus \{p_{\mathcal{M}}\}) \quad (4.24)$$

$$f_{\mathcal{M}}(i,j) \leq c_{\mathcal{M}}(i,j) \quad (i,j) \in A_{\mathcal{M}} \quad (4.25)$$

$$f_{\mathcal{M}}(i,j) \geq 0 \quad (i,j) \in A_{\mathcal{M}} \quad (4.26)$$

$$\sum_{j:(s_{\mathcal{M}},j) \in A_{\mathcal{M}}} f_{\mathcal{M}}(s_{\mathcal{M}},j) = \mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y}_{\mathcal{M}}^{score} \mathbf{e} \quad \mathcal{M} \in \text{corpus} \quad (4.27)$$

$$0 \leq \omega_{i,j} \leq \gamma_{i,j} \quad i, j \in \mathbb{Z}_{24} \quad (4.28)$$

$$\begin{aligned} \mathbf{C}_{\mathcal{M}} = \mathbf{I}_{\langle p_{\mathcal{M}}, 1 \rangle}^{\varepsilon} & \left[ \right. \\ & (\mathcal{E} \circ \pi_1)^{-1} \left( \sum_{k=0}^L \alpha_{\mathcal{M}}^k \mathbf{A}_k^{\mathcal{M}} + \beta_{\mathcal{M}} \mathbf{B}^{\mathcal{M}} \right) \\ & \cdot \cdot \cdot \langle \mathcal{C} \circ \pi_1, \pi_2 \rangle^{-1}(\mathbf{\Omega}) \cdot \cdot \cdot \overline{\pi_1^{-1}(\mathbf{I})} \\ & + \langle \mathcal{C} \circ \pi_1, \pi_2 \rangle^{-1}(\mathbf{\Sigma}) \cdot \cdot \cdot \pi^{-1}(\mathbf{I}) \\ & \left. \right] \mathbf{I}_{\langle p_{\mathcal{M}}, 0 \rangle}^{\varepsilon} \\ & \cdot \prod_{i \in N_{\mathcal{M}}} \mathbf{I}_i^{exp(\mathcal{T} \circ \mathcal{E} \circ \pi_1(i)/K)} \end{aligned} \quad (4.29)$$

As shown,  $\alpha_{\mathcal{M}}$  and  $\beta_{\mathcal{M}}$  (known parameters) can depend on  $\mathcal{M}$ , a reasonable guideline might be to fix constants (especially if the **corpus** is reasonably uniform in style). In general, the contents of the parentheses, in the line  $(\mathcal{E} \circ \pi_1)^{-1} \left( \sum_{k=0}^L \alpha_{\mathcal{M}}^k \mathbf{A}_k^{\mathcal{M}} + \beta_{\mathcal{M}} \mathbf{B}^{\mathcal{M}} \right)$  can be any matrix that an interactive user deems reflects adjacency structure of the corpus passages well. As an example, if it is known that different points, different musicians should be tuning to one another to a greater extent, then these parts can be emphasized in the  $\mathbf{B}$  matrix, while other relationships can even be omitted in these regions.

**Definition 4.3** (Consistent Corpus). *Let  $\Omega$  be a  $24 \times 24$  matrix with non-negative entries. We say **corpus** is consistent with  $\Omega$  if, for all  $\text{Score}_{\mathcal{M}} \in \text{corpus}$ , when  $\mathbf{C}_{\mathcal{M}}$  (providing arc-weights) is defined in terms of  $\Omega$ , the cost of the  $s$ - $t$  cut lifted from  $\text{Score}_{\mathcal{M}}$  is minimal.*

$$\mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y}_{\mathcal{M}}^{\text{Score}} \mathbf{e} = \mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y}_{\mathcal{M}}^{\rightarrow} \mathbf{e} \quad (4.30)$$

where  $\mathbf{Y}_{\mathcal{M}}^{\text{Score}}$  is the  $s$ - $t$  cut lifted from  $\text{Score}_{\mathcal{M}}$  and  $\mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y}_{\mathcal{M}}^{\rightarrow} \mathbf{e}$  is the weight of the minimum  $s$ - $t$  cut (found by solving MAINLP).

We will simply also say that **corpus** is consistent if there exists a non-trivial matrix  $\Omega$  such that **corpus** is consistent with  $\Omega$ .

### 4.3.1 Exact Inverse Theorem

**Exact Inverse Theorem.** *Suppose that **corpus** is consistent. Then EXACTINVERSE is a linear program with non-trivial optimal solution. Let  $(\Omega^*, \{\mathbf{C}_{\mathcal{M}}\}, \{\mathbf{F}_{\mathcal{M}}\})$  be an optimal solution to EXACTINVERSE instantiated in terms of **corpus**. Then in particular, **corpus** is consistent with  $\Omega^*$ .*

*Proof.* We can confirm that the matrix of constraints expressed by (4.29) are affine (linear up to a constant term) and hence that EXACTINVERSE is a valid linear program. The other explicitly notated as affine constraints. The (4.29) constraints are of the following form.<sup>3</sup>

$$\begin{aligned} c_{\mathcal{M}}(i, j) = & \mathbf{i}_{\langle p_{\mathcal{M}}, 1 \rangle}^{\varepsilon}(i, j) \cdot \mathbf{i}_{\langle p_{\mathcal{M}}, 0 \rangle}^{\varepsilon}(i, j) \cdot \mathbf{d}_{\mathcal{M}}(j, j) \\ & \cdot \left[ \left( \sum_{k=0}^L \alpha_{\mathcal{M}}^k \mathbf{a}_{\mathcal{M}}^k(\mathcal{E} \circ \pi_1(i), \mathcal{E} \circ \pi_1(j)) + \beta_{\mathcal{M}} \mathbf{b}^{\mathcal{M}}(\mathcal{E} \circ \pi_1(i), \mathcal{E} \circ \pi_1(j)) \right) \right. \\ & \cdot [1 - \mathbf{i}(\pi_1(i), \pi_1(j))] \cdot \omega(\langle \mathcal{C} \circ \pi_1(i), \pi_2(i) \rangle, \langle \mathcal{C} \circ \pi_1(j), \pi_2(j) \rangle) \\ & \left. + \mathbf{i}(\pi_1(i), \pi_1(j)) \cdot \sigma(\langle \mathcal{C} \circ \pi_1(i), \pi_2(i) \rangle, \langle \mathcal{C} \circ \pi_1(j), \pi_2(j) \rangle) \right] \end{aligned} \quad (4.31)$$

where the lower case bold Roman letters indicate entries of their uppercase counterparts – with the exception of  $\mathbf{d}$ ;  $\mathbf{d}_{\mathcal{M}}(\mathbf{i}, \mathbf{j})$  indicates the  $i, j$ th entry of the diagonal matrix  $\prod_{i \in N_{\mathcal{M}}} \mathbf{I}_i^{\exp(\mathcal{T} \circ \mathcal{E} \circ \pi_1(i)/K)}$ , which, as a right-multiplier, scales the  $j$ th column by  $\mathbf{d}_{\mathcal{M}}(j, j)$ . The only unknown terms are  $c_{\mathcal{M}}$  on the left hand side and  $\omega$  on the right hand side and the only operations are addition and scalar multiplication. Hence, this constraint is affine and EXACTINVERSE is a valid linear program.

<sup>3</sup>The opacity of communication of this expression demonstrates the benefit of the pullback abstraction!

The fact that **corpus** is consistent guarantees the existence of a non-trivial solution to EXACT-INVERSE. Constraints (4.24) through (4.26) are the flow constraints for a flow over the network described by  $\mathbf{C}_{\mathcal{M}}$  (for each  $\mathcal{M}$ ). The left-hand side of constraint (4.27) is the objective of the maximum flow, while the right hand side is the objective of a minimum cut problem over the same network. By strong duality,

$$\sum_{j:(s_{\mathcal{M}},j) \in A_{\mathcal{M}}} f_{\mathcal{M}}(s_{\mathcal{M}}, j) = \mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y}_{\mathcal{M}}^{\rightarrow} \mathbf{e} \quad (4.32)$$

where  $\mathbf{Y}_{\mathcal{M}}^{\rightarrow}$  is the  $s$ - $t$  minimum cut associated with  $\mathbf{C}_{\mathcal{M}}$ , found through the instantiation of the forward problem (part of the output of MAINLP). Since **corpus** is consistent, there exists a non-trivial matrix  $\mathbf{\Omega}^{\mathbf{w}}$  from which  $\mathbf{C}_{\mathcal{M}}$  is derived (through pullbacks), such that

$$\mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y}_{\mathcal{M}}^{\text{Score}} \mathbf{e} = \mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y}_{\mathcal{M}}^{\rightarrow} \mathbf{e} \quad (4.33)$$

$\mathbf{\Omega}^*$  is part of a feasible solution and hence, (4.27) is satisfied, which in turn means that  $\mathbf{\Omega}^*$  witnesses the consistency of **corpus**. It remains to be argued that  $\mathbf{\Omega}^*$  is non-trivial.

If there is a non-trivial matrix  $\mathbf{\Omega}^{\mathbf{w}}$  that witnesses the consistency of **corpus**, then  $\mathbf{\Omega}^*$  will also be non-trivial, since the sum of entries is strictly greater than the original witness.

$$\mathbf{e}' \mathbf{\Omega}^* \mathbf{e} \geq \mathbf{e}' \mathbf{\Omega}^{\mathbf{w}} \mathbf{e} \quad (4.34)$$

by the definition of the objective of EXACTINVERSE. □

*Remark 4.2* (The role of  $\mathbf{\Gamma}$ ). The Exact Inverse Theorem would still hold if we were to simply set  $\mathbf{\Gamma}$  to be  $\mathbf{E}$ , the matrix full of 1's. The more precise formulation obtained in Section 4.2 provides a robustness to the effect of feasible but *unhelpful* solutions emerging from the linear program. It is possible, indeed likely, that without the clamping of terms that should be 0, certain entries of the  $\mathbf{\Omega}$  will creep into a positive existence on account of the maximization objective. Hence, the introduction of  $\mathbf{\Gamma}$  can be thought of as an important heuristic for keeping the parameters that emerge from EXACTINVERSE sensible musically, and thus, of greater use when plugged back into a forward problem applied to a *different* passage of music.

## 4.4 Empirical Robustness

In practice, the “exactitude” of EXACTINVERSE is a little optimistic. Up to this point, we have implicitly made the strong assumption that every **corpus** has a non-trivial preimage of parameters, such that the parameters gives rise to a Forward Problem that maps back to the spelling of any one score in the **corpus** set, at least with a certain adjacency interpretation of the scores fixed. Even in instances where the **corpus** is known to only contain scores of a canonized classical composer, “errors” or idiosyncrasies in spelling are known to arise. We introduce a definition of “Approximate Consistency” to provide some practical robustness and to provide a means of measuring “how” consistent a **corpus** is.<sup>4</sup>

**Definition 4.4** (Approximate Consistency). *Let  $\Omega$  be a  $24 \times 24$  matrix with non-negative entries and let  $\omega$  be the largest entry of  $\Omega$ . We say **corpus** is  $\Delta$ -shy of consistency with  $\Omega$  if, for all  $\text{Score}_{\mathcal{M}} \in \text{corpus}$ , when  $\mathbf{C}_{\mathcal{M}}$  (providing arc-weights) is defined in terms of  $\Omega$ , the cost of the  $s$ - $t$  cut lifted from  $\text{Score}_{\mathcal{M}}$  is at most  $\Delta \cdot \omega$  shy of minimal.*

$$\mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y}_{\mathcal{M}}^{\text{Score}} \mathbf{e} - \mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y}_{\mathcal{M}}^{\rightarrow} \mathbf{e} \leq \Delta \cdot \omega \quad (4.35)$$

where  $\mathbf{Y}_{\mathcal{M}}^{\text{Score}}$  is the  $s$ - $t$  cut lifted from  $\text{Score}_{\mathcal{M}}$  and  $\mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y}_{\mathcal{M}}^{\rightarrow} \mathbf{e}$  is the weight of the minimum  $s$ - $t$  cut (found by solving MAINLP).

We will also say **corpus** is  $\Delta$ -consistent if there exists a non-trivial matrix  $\Omega$  such that **corpus** is  $\Delta$ -shy of consistency with  $\Omega$ .

The following proposition motivates the form of the definition.

**Proposition 4.1.** *Let  $\Omega$  be a  $24 \times 24$  entrywise non-negative matrix and let  $\hat{\Omega} = (\hat{\omega}_{i,j})$  be the normalized version of  $\Omega$ , namely with  $\hat{\omega}_{k,l} = \omega_{k,l} / \max_{i,j} \omega_{i,j}$ . **corpus** is  $\Delta$ -shy of consistency with  $\Omega$  if and only if **corpus** is  $\Delta$ -shy of consistency with  $\hat{\Omega}$ .*

*Proof.* Let  $\mathbf{C}_{\mathcal{M}} = (c_{\mathcal{M}}(i, j))$  be the cost matrix pulled back from  $\Omega$  and let  $\hat{\mathbf{C}}_{\mathcal{M}} = (\hat{c}_{\mathcal{M}}(i, j))$  be the cost matrix pulled back from  $\hat{\Omega}$ . By the definition of approximate consistency,

$$\mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y}_{\mathcal{M}}^{\text{Score}} \mathbf{e} - \mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y}_{\mathcal{M}}^{\rightarrow} \mathbf{e} \leq \Delta \cdot \omega \quad (4.36)$$

---

<sup>4</sup>I thank Yannis Paschalidis for directing me to a similar mechanism in his own work in computational biology, which is an important inspiration for the approximate perspective we adopt here [25].

where  $\omega = \max \omega_{i,j}$ ,  $\mathbf{Y}_{\mathcal{M}}^{\text{Score}} = (y_{\mathcal{M}}^{\text{Score}}(i, j))$  is the  $s$ - $t$  cut lifted from  $\text{Score}_{\mathcal{M}}$  and  $\mathbf{e}'\mathbf{C}'_{\mathcal{M}}\mathbf{Y}_{\mathcal{M}}^{\rightarrow}\mathbf{e}$  is the weight of the minimum  $s$ - $t$  cut (found by solving MAINLP).

The vector  $x^{\text{score}} \in N_{\mathcal{M}}$  that gives rise to  $\mathbf{Y}_{\mathcal{M}}^{\text{Score}}$  is spell-binding, because it came from a spelled score. There can therefore be no pair  $(i, j)$  such that  $y_{\mathcal{M}}^{\text{Score}}(i, j) = 1$  and  $c_{\mathcal{M}}(i, j) \neq 0$  since by the constructions in Chapter 3, the assignment of “big  $M$ ” edges via  $\Sigma$  is such that the cost of this edge is incurred only when  $x^{\text{score}}$  is not spell-binding. By Proposition 3.3, the vector  $x^{\rightarrow}$  that gives rise to the  $\mathbf{Y}_{\mathcal{M}}^{\rightarrow}$  is also spell-binding, and so the same argument applies in showing that  $y_{\mathcal{M}}$  takes value 0 wherever  $c_{\mathcal{M}}$  takes value  $M$ .

Therefore, wherever  $y_{\mathcal{M}}^{\text{score}}(i, j) = 1$  or  $y_{\mathcal{M}}^{\rightarrow}(i, j) = 1$ , we have  $c_{\mathcal{M}}(i, j)$  of the following form.

$$c_{\mathcal{M}}(i, j) = \mathbf{i}_{\langle p_{\mathcal{M}}, 1 \rangle}^{\varepsilon}(i, j) \cdot \mathbf{i}_{\langle p_{\mathcal{M}}, 0 \rangle}^{\varepsilon}(i, j) \cdot \mathbf{d}_{\mathcal{M}}(j, j) \quad (4.37)$$

$$\begin{aligned} & \cdot \left( \sum_{k=0}^L \alpha_{\mathcal{M}}^k \mathbf{a}_{\mathcal{M}}^{\mathcal{M}}(\mathcal{E} \circ \pi_1(i), \mathcal{E} \circ \pi_1(j)) + \beta_{\mathcal{M}} \mathbf{b}^{\mathcal{M}}(\mathcal{E} \circ \pi_1(i), \mathcal{E} \circ \pi_1(j)) \right) \\ & \cdot [1 - \mathbf{i}(\pi_1(i), \pi_1(j))] \cdot \omega(\langle \mathcal{C} \circ \pi_1(i), \pi_2(i) \rangle, \langle \mathcal{C} \circ \pi_1(j), \pi_2(j) \rangle) \end{aligned} \quad (4.38)$$

$$= a_{i,j} \cdot \omega(\langle \mathcal{C} \circ \pi_1(i), \pi_2(i) \rangle, \langle \mathcal{C} \circ \pi_1(j), \pi_2(j) \rangle) \quad (4.39)$$

for a scalar  $a_{i,j}$ . Equivalently, at the same  $(i, j)$

$$\hat{c}_{\mathcal{M}}(i, j) = a_{i,j} \cdot \omega(\langle \mathcal{C} \circ \pi_1(i), \pi_2(i) \rangle, \langle \mathcal{C} \circ \pi_1(j), \pi_2(j) \rangle) / \omega \quad (4.40)$$

$$= c_{\mathcal{M}}(i, j) / \omega \quad (4.41)$$

Hence, taking  $\oplus$  to be an “exclusive or” operation.

$$(\mathbf{e}'\mathbf{C}'_{\mathcal{M}}\mathbf{Y}_{\mathcal{M}}^{\text{Score}}\mathbf{e} - \mathbf{e}'\mathbf{C}'_{\mathcal{M}}\mathbf{Y}_{\mathcal{M}}^{\rightarrow}\mathbf{e}) / \omega = \mathbf{e}'\mathbf{C}'_{\mathcal{M}}(\mathbf{Y}_{\mathcal{M}}^{\text{Score}} - \mathbf{Y}_{\mathcal{M}}^{\rightarrow})\mathbf{e} / \omega \quad (4.42)$$

$$= \sum_{y_{i,j}^{\rightarrow} \oplus y_{i,j}^{\text{score}} = 1} c_{\mathcal{M}}(i, j) / \omega \quad (4.43)$$

$$= \sum_{y_{i,j}^{\rightarrow} \oplus y_{i,j}^{\text{score}} = 1} \hat{c}_{\mathcal{M}}(i, j) \quad (4.44)$$

$$= \mathbf{e}'\hat{\mathbf{C}}_{\mathcal{M}}(\mathbf{Y}_{\mathcal{M}}^{\text{Score}} - \mathbf{Y}_{\mathcal{M}}^{\rightarrow})\mathbf{e} \quad (4.45)$$

$$= \mathbf{e}'\hat{\mathbf{C}}_{\mathcal{M}}\mathbf{Y}_{\mathcal{M}}^{\text{Score}}\mathbf{e} - \mathbf{e}'\hat{\mathbf{C}}_{\mathcal{M}}\mathbf{Y}_{\mathcal{M}}^{\rightarrow}\mathbf{e} \quad (4.46)$$

and so we have the desired inequality:

$$\mathbf{e}'\hat{\mathbf{C}}_{\mathcal{M}}\mathbf{Y}_{\mathcal{M}}^{\text{score}}\mathbf{e} - \mathbf{e}'\hat{\mathbf{C}}_{\mathcal{M}}\mathbf{Y}_{\mathcal{M}}^{\rightarrow}\mathbf{e} \leq \Delta \quad (4.47)$$

This inequality is enough, provided  $\mathbf{Y}_{\mathcal{M}}^{\rightarrow}$  is minimal for  $\hat{\mathbf{C}}_{\mathcal{M}}$  as well as  $\mathbf{C}_{\mathcal{M}}$ , which we now show. First,  $\mathbf{Y}_{\mathcal{M}}^{\rightarrow}$  is still feasible, since changing the cost vector can only take away the optimality, not feasibility of a solution. If  $\mathbf{Y}_{\mathcal{M}}^{\rightarrow}$  was not optimal for  $\hat{\mathbf{C}}_{\mathcal{M}}$ , we could use a similar argument to show that the optimal solution with respect to  $\hat{\mathbf{C}}_{\mathcal{M}}$  was optimal and strictly less than the cut weight of  $\mathbf{Y}_{\mathcal{M}}^{\rightarrow}$  with respect to  $\mathbf{C}_{\mathcal{M}}$  contradicting the assumption that  $\mathbf{Y}_{\mathcal{M}}^{\rightarrow}$  was a minimum cut found by MAINLP.

The converse result follows the same argument. □

Now we can construct the approximate equivalent of EXACTINVERSE.

**Definition 4.5** (APPROXINVERSE linear program).

$$\text{APPROXINVERSE} = \max_{\mathbf{\Omega}, \mathbf{C}, \mathbf{F}, \delta, \Delta, z} \mathbf{e}' \mathbf{\Omega} \mathbf{e} - \lambda \Delta \quad (4.48)$$

subject to

$$\sum_{i:(i,k) \in A_{\mathcal{M}}} f_{\mathcal{M}}(i, k) = \sum_{j:(k,j) \in A_{\mathcal{M}}} f_{\mathcal{M}}(k, j) \quad k \in \pi_1^{-1}(V_{\mathcal{M}} \setminus \{p_{\mathcal{M}}\}) \quad (4.49)$$

$$f_{\mathcal{M}}(i, j) \leq c_{\mathcal{M}}(i, j) \quad (i, j) \in A_{\mathcal{M}} \quad (4.50)$$

$$f_{\mathcal{M}}(i, j) \geq 0 \quad (i, j) \in A_{\mathcal{M}} \quad (4.51)$$

$$y_{\mathcal{M}}(i, j) = x_{\mathcal{M}}(j) - x_{\mathcal{M}}(i) \quad (i, j) \in A_{\mathcal{M}} \quad (4.52)$$

$$y_{\mathcal{M}}(i, j) = 0 \quad (i, j) \notin A_{\mathcal{M}} \quad (4.53)$$

$$\sum_{j:(s_{\mathcal{M}}, j) \in A_{\mathcal{M}}} f_{\mathcal{M}}(s_{\mathcal{M}}, j) = \mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y}_{\mathcal{M}} \mathbf{e} - \delta_{\mathcal{M}} \quad \mathcal{M} \in \text{corpus} \quad (4.54)$$

$$\Delta \geq \delta_{\mathcal{M}} \quad \mathcal{M} \in \text{corpus} \quad (4.55)$$

$$0 \leq \omega_{i,j} \leq \gamma_{i,j} \quad i, j \in \mathbb{Z}_{24} \quad (4.56)$$

$$\begin{aligned} \mathbf{C}_{\mathcal{M}} = & \mathbf{I}_{\langle p_{\mathcal{M}}, 1 \rangle}^{\varepsilon} \left[ \right. \\ & (\mathcal{E} \circ \pi_1)^{-1} \left( \sum_{k=0}^L \alpha_{\mathcal{M}}^k \mathbf{A}_k^{\mathcal{M}} + \beta_{\mathcal{M}} \mathbf{B}^{\mathcal{M}} \right) \\ & \cdot \cdot \langle \mathcal{C} \circ \pi_1, \pi_2 \rangle^{-1}(\mathbf{\Omega}) \cdot \cdot \overline{\pi_1^{-1}(\mathbf{I})} \\ & + \langle \mathcal{C} \circ \pi_1, \pi_2 \rangle^{-1}(\mathbf{\Sigma}) \cdot \cdot \pi^{-1}(\mathbf{I}) \\ & \left. \right] \mathbf{I}_{\langle p_{\mathcal{M}}, 0 \rangle}^{\varepsilon} \\ & \cdot \prod_{i \in N_{\mathcal{M}}} \mathbf{I}_i^{\exp(\mathcal{T} \circ \mathcal{E} \circ \pi_1(i)/K)} \end{aligned} \quad (4.57)$$

Choosing the right value of  $\lambda$  in APPROXINVERSE is a fine balancing act. If  $\lambda$  is too large, then the optimal solution could be the trivial choice of  $\mathbf{\Omega}$  (all 0's), which is feasible with 0 duality gap between flows and  $s$ - $t$  cuts (but, by our definition, the trivial matrix is not a suitable witness of approximate or complete consistency).

On the other hand, if  $\Delta$  is fixed as some  $\Delta^{\mathbf{F}}$ , then the objective reduces to that of APPROXINVERSE, and hence we are guaranteed to find a non-trivial witness of  $\Delta^{\mathbf{F}}$ -consistency provided

one exists (subject to the proof of the following theorem). Alternatively, if the optimal solution to APPROXINVERSE is trivial, we conclude that **corpus** is not  $\Delta^F$ -consistent.

#### 4.4.1 Approximate Inverse Theorem

**Approximate Inverse Theorem.** *Suppose that **corpus** is  $\Delta^F$ -consistent for some  $\Delta^F$ . Let APPROXINVERSE( $\Delta^F$ ) be the LP obtained when  $\Delta$  is fixed with  $\Delta = \Delta^F$ . Then APPROXINVERSE( $\Delta^F$ ) is a linear program with non-trivial optimal solution.*

*Suppose further that we may also fix an entry of  $\mathbf{\Omega}^*$  beforehand, with  $\omega_{i,j} = 1$  for some  $(i,j)$ . Call the statement of this linear program, APPROXINVERSE( $\Delta^F, i, j$ ). Let  $(\mathbf{\Omega}^*, \{\mathbf{C}_{\mathcal{M}}\}, \{\mathbf{F}_{\mathcal{M}}^*\}, \delta^*)$ , be an optimal solution to APPROXINVERSE( $\Delta^F, i, j$ ). There is a choice of  $(i, j)$  such that **corpus** is  $\Delta^F$ -shy of consistency with  $\mathbf{\Omega}^*$ .*

*Proof.* The proof that the definition of  $\mathbf{C}_{\mathcal{M}}$  is suitably affine is the same as that in the proof of the Exact Inverse Theorem that we just saw. All the remaining constraints are notated in a straightforward way, and so we conclude that APPROXINVERSE( $\Delta^F$ ) is a linear program.

The fact that **corpus** is  $\Delta^F$ -consistent guarantees the existence of a non-trivial solution to APPROXINVERSE( $\Delta^F$ ). Constraints (4.24) through (4.26) are the flow constraints for a flow over the network described by  $\mathbf{C}_{\mathcal{M}}$  (for each  $\mathcal{M}$ ). The left-hand side of constraint (4.27) is the objective of the maximum flow, while the right hand side is the objective of a minimum cut problem over the same network. By strong duality,

$$\sum_{j:(s_{\mathcal{M}}, j) \in A_{\mathcal{M}}} f_{\mathcal{M}}(s_{\mathcal{M}}, j) = \mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y}_{\mathcal{M}}^{\rightarrow} \mathbf{e} \quad (4.58)$$

where  $\mathbf{Y}_{\mathcal{M}}^{\rightarrow}$  is the  $s$ - $t$  minimum cut associated with  $\mathbf{C}_{\mathcal{M}}$ , found through the instantiation of the forward problem (part of the output of MAINLP). Since **corpus** is  $\Delta^F$ -consistent, there exists a non-trivial matrix  $\mathbf{\Omega}^w$  from which  $\mathbf{C}_{\mathcal{M}}$  is derived (through pullbacks), such that

$$\mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y}_{\mathcal{M}}^{\text{Score}} \mathbf{e} - \mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y}_{\mathcal{M}}^{\rightarrow} \mathbf{e} \leq \Delta^F / \omega^w \quad (4.59)$$

where  $\omega^w = \max_{i,j} \omega_{i,j}^w$ . Let  $(k, l)$  be such that  $\omega_{k,l}^w = \omega^w$  and let  $\omega_{k,l} = 1$  be fixed before solving APPROXINVERSE( $\Delta^F, k, l$ ). By the proof of Proposition 4.1, there exists a non-trivial optimal solution with  $\omega_{k,l} = 1$  that witnesses the  $\Delta^F$ -consistency of **corpus**.  $\square$

*Remark 4.3.* The APPROXINVERSE linear program not only extracts the  $\Omega$  parameters from a score, but also gives a guarantee on the appropriateness of the model in reflecting the rules of spelling that are enacted in this corpus. The  $\Delta$ -witness could therefore be used as a metric for describing and comparing different groups of scores in future studies. Moreover, obtaining a low value of  $\Delta$  on a corpus of Tonal music would give a good indication that the graphical model presented in this thesis is adaptable not only in a general setting, but also to spelling classical scores by specific composers to a high “accuracy” (see Section 1.2.2).

## Chapter 5

# The Microtonal Context

Microtones, the notes with pitches “in-between” the keys of the piano, are a feature of the language of some more modern composers. I will here extend the forward problem to the microtonal context.

### 5.1 Quarter Tones

There are a number of notational conventions for representing microtones. We will use that in Figure 5.1 to denote quarter-tones (or *half* half-steps). For eighth-tones, we will add up-hats above or down-checks below, as shown in Figure 5.2. In Figure 5.2, the  $+1/2$  column comprises the quarter-tone separated additions. The  $+1/4$  and  $+3/4$  columns are made up of the spellings from a quarter-tone higher and a quarter-tone lower with the appropriate arrow indication (hat or check, meaning pitch-up or pitch-down). The principles used in modifying the model in this particular notational context will be applicable to other microtonal notational conventions.

|      |      |      |      |    |
|------|------|------|------|----|
| -2   | -1   | +0   | +1   | +2 |
| bb   | b    | q    | #    | x  |
| db   | db   | q#   | ##   |    |
| -1.5 | -0.5 | +0.5 | +1.5 |    |

Figure 5.1: A new set of “in-between” accidentals

We will construct a statement of the forward problem that makes sense in this particular mi-



Figure 5.2: The full complement of pitch classes and their spellings (at a granularity of eighth-tones ( $\frac{1}{4}$ 's of half-steps)). An extension of the rules in Figure 1.4 can be used to generate the above diagram.

crotonal context. In just the same way as before, we will want to construct two pitch-class indexed matrices (which we will call  $\Omega_2$  and  $\Sigma_2$  after  $\Omega$  in Chapter 4 and  $\Sigma$  in Chapter 3) that have an important influence on the weighting of the graphical model via matrix pullbacks (as seen in Section 2.4.2). I will spare the reader many of the formal details in constructing the equivalent of the big cost matrix definition (3.13), but this work can be done without too much trouble.

### 5.1.1 Constructing Omega-sub-2

Generally speaking, our approach in constructing  $\Omega_2$  will be to view a pitch-class with a value that ends in .5 as “behaving like” the pitch class above or below it. This will enable us to construct a pullback function that leaves the relationships between the original pitch classes the same, and duplicates the various entries of the original  $\Omega$  matrix for all the entries that should “behave like” the indexing pitch classes of the original matrix.

In most cases, the pitch-class above or below that is chosen as a proxy for the quarter-tone pitch class should be the closest black note, since none of the new “in-between” accidentals (see Figure 5.1) is equivalent to a natural sign. The case regarding the pitch classes 4.5 and 11.5 is more complicated, since these two pitch classes lie between two *white* notes. Our solution in this case is to have a representative variable from the variables associated with each of the surrounding white notes in the standard model, developed in Chapter 3. Take the pitch class 4.5 as an example; we will borrow the 0th variable of pitch class 4 and the 0th variable of pitch class 5 to represent its edgewise interactions with other pitch classes. If the variable borrowed from pitch class 5 would be happy<sup>1</sup> spelled as an  $F\sharp$  or  $G\flat$  and the variable borrowed from pitch class 4 would be happy spelled as an  $F\flat$ , we will take 4.5 to be spelled as  $F$ -quarter-flat (this is the case when both variables take value 0). If the variable borrowed from pitch class 5 would be happy spelled as an  $E\sharp$  and the variable borrowed from pitch class 4 happy spelled as an  $E\sharp$  or  $D\sharp$ , we will take 4.5 to be spelled as  $E$ -quarter-sharp (this is the case when both variables take value 1).<sup>2</sup>

To obtain the quarter tone version of  $\Omega$ , we should therefore produce a pullback based on the

---

<sup>1</sup>Incurring a minimal cost

<sup>2</sup>To enforce the sameness of the two variables, we will use big  $M$  edges in the usual way. See Section ??.

following inheritance structure.

$$\mathcal{B}(\langle n, b \rangle) = \begin{cases} \langle n, b \rangle & n \in \mathbb{Z}_{12} \\ \langle 1, b \rangle & n \in \{0.5, 1.5\} \\ \langle 3, b \rangle & n \in \{2.5, 3.5\} \\ \langle 4, 0 \rangle & \langle n, b \rangle = \langle 4.5, 0 \rangle \\ \langle 5, 0 \rangle & \langle n, b \rangle = \langle 4.5, 1 \rangle \\ \langle 6, b \rangle & n \in \{5.5, 6.5\} \\ \langle 8, b \rangle & n \in \{7.5, 8.5\} \\ \langle 10, b \rangle & n \in \{9.5, 10.5\} \\ \langle 11, 0 \rangle & \langle n, b \rangle = \langle 11.5, 0 \rangle \\ \langle 0, 0 \rangle & \langle n, b \rangle = \langle 11.5, 1 \rangle \end{cases} \quad (5.1)$$

Now if  $\Omega_2$  is the extension of  $\Omega$  to a quarter-tone scale, we can take

$$\Omega_2 = \mathcal{B}^{-1}(\Omega) \quad (5.2)$$

Judging from Figure 5.2, the pitch classes  $\{1.5, 2.5, 6.5, 7.5, 8.5, 9.5\}$  are like 8 in that they only have two available spellings and only one decision variable is needed to capture the full variety of the spelling. Specifically, the predicate that controls double-flatness or double-sharpness (the relevant tables are recalled as Tables 5.1 and 5.2) in pitch classes 1,2,6 and 9 is redundant and so its arc-weights need not be inherited. Hence, we can more economically zero-out the (ir)relevant rows and columns, with

$$\Omega_2 = \prod_{i \in I} \mathbf{I}_{\langle i, 1 \rangle}^0 \mathcal{B}^{-1}(\Omega) \prod_{i \in I} \mathbf{I}_{\langle i, 1 \rangle}^0 \quad (5.3)$$

where  $I = \{1.5, 2.5, 6.5, 7.5, 8.5, 9.5\}$  (recall that each  $\mathbf{I}_{\langle i, 1 \rangle}^0$  multiplies a row or column by 0).

### 5.1.2 Constructing Sigma-sub-2

We will want to introduce a similar pullback to generate  $\Sigma_2$ , and in the same way, we can zero-out all the  $2 \times 2$  submatrices along the diagonal that correspond to  $i \in I$ , just as we did with pitch class

| Spelling-Set                                | Pitch Classes | Category |
|---------------------------------------------|---------------|----------|
| $\mathcal{S}_0 = \{\sharp, \#, \flat\}$     | 0, 5          | 0        |
| $\mathcal{S}_1 = \{\#, \flat, \times\}$     | 1, 6          | 1        |
| $\mathcal{S}_2 = \{\times, \flat, \sharp\}$ | 2, 7, 9       | 2        |
| $\mathcal{S}_3 = \{\flat, \#, \flat\}$      | 3, 10         | 3        |
| $\mathcal{S}_4 = \{\flat, \sharp, \times\}$ | 4, 11         | 4        |
| $\mathcal{S}_5 = \{\#, \flat\}$             | 8             | 5        |

Table 5.1: Recall: partitioning pitch classes by permissible spelling set

| $a \in \mathcal{S}_0$ | $\phi_0^0$ | $\phi_1^0$ |
|-----------------------|------------|------------|
| $\sharp$              | 0          | 1          |
| $\#$                  | 1          | 1          |
| $\flat$               | 0          | 0          |

(a) Category 0

| $a \in \mathcal{S}_1$ | $\phi_0^1$ | $\phi_1^1$ |
|-----------------------|------------|------------|
| $\#$                  | 1          | 0          |
| $\flat$               | 0          | 0          |
| $\times$              | 1          | 1          |

(b) Category 1

| $a \in \mathcal{S}_2$ | $\phi_0^2$ | $\phi_1^2$ |
|-----------------------|------------|------------|
| $\times$              | 1          | 1          |
| $\flat$               | 0          | 0          |
| $\sharp$              | 1          | 0          |

(c) Category 2

| $a \in \mathcal{S}_3$ | $\phi_0^3$ | $\phi_1^3$ |
|-----------------------|------------|------------|
| $\flat$               | 0          | 0          |
| $\#$                  | 1          | 1          |
| $\flat$               | 0          | 1          |

(d) Category 3

| $a \in \mathcal{S}_4$ | $\phi_0^4$ | $\phi_1^4$ |
|-----------------------|------------|------------|
| $\flat$               | 0          | 0          |
| $\sharp$              | 1          | 0          |
| $\times$              | 1          | 1          |

(e) Category 4

| $a \in \mathcal{S}_5$ | $\phi^5$ |
|-----------------------|----------|
| $\#$                  | 1        |
| $\flat$               | 0        |

(f) Category 5

Table 5.2: Recall: the set of predicates that correspond to spelling assignments up by category (see Table 5.1).

8 in Chapter 3. Once again, however, the pitch classes in  $\{4.5, 11.5\}$  must be treated as a special case. We need big  $M$  edges that enforce that  $x(\langle 4.5, 0 \rangle) = x(\langle 4.5, 1 \rangle)$  and likewise for 11.5. This can be achieved simply with two big  $M$  edges, so that the corresponding submatrix is

$$\begin{array}{cc} 0 & M \\ M & 0 \end{array} \tag{5.4}$$

The fact that this pair of big  $M$  edges works to ensure that the two connected decision variables take the same value in the minimum cut optimal solution is along the lines of Proposition 3.3 in tandem with the equivalence of  $\overline{\phi \oplus \psi}$  (the **not** of an **xor**) and  $\phi \rightarrow \psi \wedge \psi \rightarrow \phi$ .

We now construct the formal pullback for  $\Sigma_2$ . We will signify the quarter-tone pitch classes by

$\mathbb{Z}_{12}^{+0.5}$ . Define  $\mathcal{K}_2 : \mathbb{Z}_{12} \cup \mathbb{Z}_{12}^{+0.5} \rightarrow \mathbb{Z}_4$  by

$$\mathcal{K}_2(i) = \begin{cases} 0 & i \in \{0, 3, 5, 10\} \cup \{3.5, 10.5\} \\ 1 & i \in \{1, 2, 4, 6, 7, 9, 11\} \cup \{0.5, 5.5\} \\ 2 & i \in \{8\} \cup \{1.5, 2.5, 6.5, 7.5, 8.5, 9.5\} \\ 3 & i \in \{4.5, 11.5\} \end{cases} \quad (5.5)$$

We can now extend the definition of  $\Sigma$  slightly (compare with Chapter 3). We add in the zeroing-out terms on the left and right as well.

$$\Sigma_2 = \prod_{i \in I} \mathbf{I}_{\langle i, 1 \rangle}^0 \cdot \langle \mathcal{K}_2 \circ \pi_1, \pi_2 \rangle^{-1} \begin{pmatrix} 0 & 0 & & & & \\ M & 0 & & & & \\ & & 0 & M & & \\ & & 0 & 0 & & \\ & & & 0 & 0 & \\ & & & 0 & 0 & \\ & & & & 0 & M \\ & & & & M & 0 \end{pmatrix} \cdot \prod_{i \in I} \mathbf{I}_{\langle i, 1 \rangle}^0 \quad (5.6)$$

Other than the changes we have described to  $\Omega_2$  and  $\Sigma_2$ , all of the techniques from previous chapters carry over. We will see in the following section, that when eighth tones are introduced, some complications ensue.

## 5.2 Eighth Tones

As mentioned, our convention for representing eighth tones will be to add up-arrows or down-checks to each of the accidentals involved (consult Figure 5.2). Clearly the two bit encoding of spellings is now insufficient and we need a new decision variable or predicate for each pitch class indicating “up” or “down”.

There are complications, however. Switching between *up* and *down* can entirely change how the note is “behaving”. For example, via our pullback approach given in the previous section, we had arcs incident to the pitch class 0.5 being assigned arc-weight information according to the values pitch-class-1-incident arcs; the set of arc-weights incident to pitch class 0.5 is therefore different from those incident to pitch class 0. So, the question of how to assign arc-weights to 0.25-pitch-

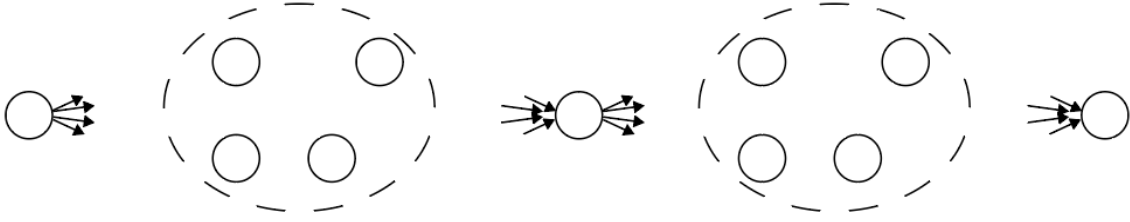


Figure 5.3: For every connected component of  $\text{Net}_{\mathcal{M}} \setminus \{\langle p_{\mathcal{M}}, 1 \rangle, \langle p_{\mathcal{M}}, 0 \rangle\}$ , add an intermediate source/sink

class-incident edges is then a challenging one. When the corresponding up-down variable is in an *up* state, it should have incident edges with weights along the lines of a arcs of a 0 pitch class note. Conversely, when the variable is in a *down* state, it should have incident edges with weights along the lines of arcs of a 1 pitch class note, which should be *different*. To capture this edge-switching property, we need to modify our model.

We propose two updates to the model, and with these, three algorithmic approaches to solving the Pitch-Spelling Problem extended to eighth tones.

### 5.2.1 Flow Approach

A key observation is that the interval between adjacent notes is very hard to parse when one is spelled up and the other is spelled down. Under the assumption that we should, therefore, constrain all close notes to have the same upness, we can introduce a *single* up-down variable for all the notes in a single connected component in the sense of the section on segmentation strategies (Section 2.2). In what follows, we will restrict our attention to a single connected component of the graph.<sup>3</sup>

Fix the up-variable in *up* state. Call the resultant flow network (or really a single connected component of it)  $G_{\uparrow}$ . Now fix the up-variable in *down* state and call the resultant flow network  $G_{\downarrow}$ . Merge the source  $s_{\downarrow}$  of  $G_{\downarrow}$  into the sink of  $t_{\uparrow}$  so they are a single node  $r$  incident to all the arcs that would flow out of  $s_{\downarrow}$  or into  $t_{\uparrow}$  respectively. Associate the up-down variable to the node  $r$ . The resulting flow network is depicted in Figure 5.3, with  $r$  corresponding to the node in the center of the diagram.

The outcome of rearranging the pair of flow networks  $G_{\uparrow}$  and  $G_{\downarrow}$  as in Figure 5.3 is that the up-down variable  $u_r$  associated with  $r$ , has the effect of “switching off” one subgraph or the other. At an optimal solution found by Pseudoflow or simplex [10, 11, 2],  $u_r$  will be either up or down (1

<sup>3</sup>By connected component, I in fact mean connected component when we ignore the fact that the parsimony pivot is connected to everything - hence we drop the source and sink of the network when we refer to connected components.

or 0 respectively) since it is part of a minimum cut problem. Suppose  $u = 0$ ; then if all the nodes in  $G_\uparrow$  take value 0, this whole subgraph contributes nothing to the total cost, which must therefore be an optimal (sub)assignment. Hence, the cost is reduced to the cost over edges in  $G_\downarrow$ . Conversely, when  $u = 1$ , the total cost over the whole (connected component of the) network is reduced to the cost over  $G_\uparrow$ . In other words, when  $u = 0$  only down-state edges are active, and when  $u = 1$  only up-state edges are active, which is precisely what we wanted. In interpreting the solution, we can check the value of  $u$  to determine which set of nodes to ignore wholesale.

### 5.2.2 LP Approach

If we wish to allow for some leeway with regards to close pairs of notes in different up-down states, rather than insisting that all connected up-down-variable nodes take the same value, we can make modifications directly to MAINLP. The result is no longer a minimum cut problem, but still an LP, which can be solved very efficiently using the simplex method. We first reprise the statement of MAINLP for reference.

**Definition 5.1** (MAINLP linear program).

$$\text{MAINLP} = \min_{\mathbf{Y}, \mathbf{x}} \mathbf{e}' \mathbf{C}'_{\mathcal{M}} \mathbf{Y} \mathbf{e} \quad (5.7)$$

$$\text{subject to} \quad x_j \leq y_{s,j} \quad \forall j \neq s, j \neq t \quad (5.8)$$

$$1 - x_i \leq y_{i,t} \quad \forall i \neq s, i \neq t \quad (5.9)$$

$$x_j - x_i \leq y_{i,j} \quad \forall c_{(i,j)} > 0, i \neq s, j \neq t \quad (5.10)$$

$$x_i \leq 1 \quad \forall i \in N_{\mathcal{M}} \quad (5.11)$$

$$y_{i,j}, x_i \geq 0 \quad \forall i, j \in N_{\mathcal{M}} \quad (5.12)$$

where  $s = \langle p_{\mathcal{M}}, 1 \rangle$  and  $t = \langle p_{\mathcal{M}}, 0 \rangle$ .

Take  $N_{\mathcal{M}} = V_{\mathcal{M}} \times \mathbb{Z}_2 \cup V_{\mathcal{M}}$ . The first term in the union is no different from the usual set of decision variables. The second term corresponds to the up-down state variable. For a note  $n$ , we will refer to the up-down variable as  $u_n$ . Hence for a node  $x_i$ , where  $i = \langle n, b \rangle$ , the associated up down variable is  $u_{\pi_1(i)}$  or equivalently,  $u_n$ . Rather than having a single arc variable  $y_{i,j}$  for each pair of nodes, we introduce four arc-wise variables,  $y_{i,j}^{\uparrow,\uparrow}$ ,  $y_{i,j}^{\downarrow,\uparrow}$ ,  $y_{i,j}^{\uparrow,\downarrow}$  and  $y_{i,j}^{\downarrow,\downarrow}$ .  $y_{i,j}^{\uparrow,\downarrow}$  can only be nonzero when  $u_{\pi_1(i)} = 1$  (up),  $u_{\pi_1(j)} = 0$  (down), and so on. Correspondingly we have entries  $c_{i,j}^{\uparrow,\uparrow}$ ,  $c_{i,j}^{\downarrow,\uparrow}$ ,  $c_{i,j}^{\uparrow,\downarrow}$  and

$c_{i,j}^{\downarrow,\downarrow}$  in the cost vector, which are incurred only when  $y_{i,j}^{\uparrow,\uparrow}$ ,  $y_{i,j}^{\downarrow,\uparrow}$ ,  $y_{i,j}^{\uparrow,\downarrow}$  or  $y_{i,j}^{\downarrow,\downarrow}$  respectively take nonzero value.

To enforce the switching on and off of  $y_{i,j}^{\uparrow,\downarrow}$  and so on, we add the following constraints instead of constraints of the form of (??):

$$x_j - x_i \leq y_{i,j}^{\downarrow,\downarrow} + u_{\pi_1(i)} + u_{\pi_1(j)} \quad (5.13)$$

$$x_j - x_i \leq y_{i,j}^{\uparrow,\downarrow} + (1 - u_{\pi_1(i)}) + u_{\pi_1(j)} \quad (5.14)$$

$$x_j - x_i \leq y_{i,j}^{\downarrow,\uparrow} + u_{\pi_1(i)} + (1 - u_{\pi_1(j)}) \quad (5.15)$$

$$x_j - x_i \leq y_{i,j}^{\uparrow,\uparrow} + (1 - u_{\pi_1(i)}) + (1 - u_{\pi_1(j)}) \quad (5.16)$$

When the sum of the last two terms of the right hand side is greater than or equal to 1 for any of these constraints, the corresponding  $y$  variable finds the value 0 at an optimum. This  $y$  variable gets zeroed-out, because taking a non-zero value is not necessary for constraint satisfaction, but *does* cause the total cost to increase (hence it can no longer be part of an optimum).

We also add a matrix variables  $\mathbf{W} = (w_{i,j})$  for edges between up-down nodes as well as the vector of up-down variables  $\mathbf{u}$ . We let  $\mathbf{C}_{\mathcal{M}}^u$  denote the matrix of arc-weights over arcs between up-down nodes. Note that there are no connections between up-down nodes and the  $x_i$  so we have enough cost matrices to characterize the whole problem:

**Definition 5.2** (UPDOWNLP linear program).

$$\text{UPDOWNLP} = \min_{\mathbf{W}, \mathbf{Y}, \mathbf{x}, \mathbf{u}} \mathbf{e}' \left[ \mathbf{C}_{\mathcal{M}}^{\uparrow, \uparrow'} \mathbf{Y}^{\uparrow, \uparrow} + \mathbf{C}_{\mathcal{M}}^{\uparrow, \downarrow'} \mathbf{Y}^{\uparrow, \downarrow} + \mathbf{C}_{\mathcal{M}}^{\downarrow, \uparrow'} \mathbf{Y}^{\downarrow, \uparrow} + \mathbf{C}_{\mathcal{M}}^{\downarrow, \downarrow'} \mathbf{Y}^{\downarrow, \downarrow} \right] \mathbf{e} + \mathbf{e}' \mathbf{C}_{\mathcal{M}}^u \mathbf{W} \mathbf{e} \quad (5.17)$$

$$\text{subject to} \quad x_j - x_i \leq y_{i,j}^{\downarrow, \downarrow} + u_{\pi_1(i)} + u_{\pi_1(j)} \quad \forall(i, j) : c_{i,j}^{\downarrow, \downarrow} > 0 \quad (5.18)$$

$$x_j - x_i \leq y_{i,j}^{\uparrow, \downarrow} + (1 - u_{\pi_1(i)}) + u_{\pi_1(j)} \quad \forall(i, j) : c_{i,j}^{\uparrow, \downarrow} > 0 \quad (5.19)$$

$$x_j - x_i \leq y_{i,j}^{\downarrow, \uparrow} + u_{\pi_1(i)} + (1 - u_{\pi_1(j)}) \quad \forall(i, j) : c_{i,j}^{\downarrow, \uparrow} > 0 \quad (5.20)$$

$$x_j - x_i \leq y_{i,j}^{\uparrow, \uparrow} + (1 - u_{\pi_1(i)}) + (1 - u_{\pi_1(j)}) \quad \forall(i, j) : c_{i,j}^{\uparrow, \uparrow} > 0 \quad (5.21)$$

$$u_n - u_m \leq w_{m,n} \quad \forall(m, n) : c_{m,n}^u > 0 \quad (5.22)$$

$$x_{\langle p, \mathcal{M}, 1 \rangle} = 0 \quad (5.23)$$

$$x_{\langle p, \mathcal{M}, 0 \rangle} = 1 \quad (5.24)$$

$$\mathbf{x}, \mathbf{u} \leq \mathbf{e} \quad (5.25)$$

$$\mathbf{Y}, \mathbf{x}, \mathbf{u}, \mathbf{W} \geq \mathbf{0} \quad (5.26)$$

There is no immediate way of populating the matrix  $\mathbf{C}_{\mathcal{M}}^u$  through an inverse problem in the vein of Chapter 4. Given what we have already said in Section 5.2.1, however, we know that adjacent up-down state variables should be the same *with high priority*. We can therefore populate  $\mathbf{C}_{\mathcal{M}}^u$  by way of a pullback from  $k\mathbf{E}$  (where  $\mathbf{E}$  is the matrix of all 1's and  $k$  is a suitably large scalar) instead of the usual  $\mathbf{\Omega}$  or  $\mathbf{\Omega}_2$ .

**Proposition 5.1.** *The optimal solution to UPDOWNLP found by the Simplex algorithm is spell-binding.*

*Proof.* The  $u_n$  are only constrained  $0 \leq u_n \leq 1$  and so the extreme points of the feasible region of UPDOWNLP have  $u_n = 0$  or  $u_n = 1$ . By a fundamental result of linear programming, if there are optimal solutions to an LP, there are optimal solutions that are extreme points of the feasible region, and these are the optimal solutions found by the simplex algorithm [2]. Furthermore, UPDOWNLP is bounded below by 0, and so it must have a finite optimum. By another fundamental result of linear programming, any bounded LP must achieve its *infimum* [2]. We have therefore, that the simplex algorithm must find an optimum with the  $u_n$  taking values 0 or 1.

Fix  $\mathbf{u}^*$  that is part of an optimal solution and fix  $w_{m,n} = u_n - u_m$  for each  $(n, m)$  such that

$c_{m,n}^u > 0$ .<sup>4</sup> The remaining constraints that are active (in the sense that the associated  $y$  entry can be nonzero) form a minimum cut problem statement and so we know that simplex will find an integer (spell-binding) solution to them.  $\square$

### 5.2.3 Hybrid Approach

Phase I of the simplex algorithm involves finding an initial basic feasible solution (an extreme point of the feasible set); in Phase II, the algorithm produces “pivots” between adjacent extreme points in a manner that improves the cost or holds it constant [2]. Instead of the usual simplex-based Phase I, we can solve the flow approach version of the problem using the Pseudoflow algorithm [10, 11] (Section 5.2.1). The solution found in the flow approach will be a basic feasible solution of UPDOWNLP when converted to the right format. Not only, that, we expect the solution found by Pseudoflow to be optimal in UPDOWNLP, since UPDOWNLP relaxes the condition that adjacent up-down states must be equal, but at a high cost. Hence, we expect the simplex algorithm to make no pivots, and merely to provide a certificate of the correctness of the solution found in the flow approach.

---

<sup>4</sup>In particular we have  $c_{m,n}^u = 1$  for these  $(m, n)$  by the proposed construction of  $\mathbf{C}_{\mathcal{M}}^u$  described in the paragraph following the definition of UPDOWNLP.

## Chapter 6

# Coda

I have presented a graphical model of the score, which gives rise to a family of pitch-spelling algorithms. I have shown that the modeling strategy is adaptable to different notational contexts, from the typical 12-tone scale, to scales with quarter-tones, first, and then to scales with eighth-tones. In the former two cases, I have shown that the pitch-spelling problem – under the cognitive assumption that interval optimization is the fundamental force in the proper spellings of music outside of the purview of Tonal Harmony – reduces to the minimum cut problem from the study of network flows. In the latter case (the eighth tone system), I have demonstrated how modifications to the model can handle the complications that arise. With a simplifying assumption, that connected up-down variables should always be the same, I demonstrate how an alteration to the graph can reveal another minimum cut reduction. I also provide a linear programming implementation *without* such simplifying assumption, which I show *also* results in spell-binding (meaningful) solutions; further, I show that the two approaches to solving the eighth-tone version of the problem can be hybridized – the first approach can replace the usual Phase I of the simplex method, to obtain a probably optimal initial basic feasible solution.

The methods employed require no use of windowing, hence avoiding risks equivalent to “aliasing”<sup>1</sup>, whereby a window could fall on the wrong part of a score and miss a key detail. Instead, a full graph structure is developed and the Pitch-Spelling Problem reduced to a form that can easily be stated in terms of existing graph algorithms.

In the heart of the thesis, I develop a theory surrounding an inverse optimization problem that

---

<sup>1</sup>The term from signal processing [20]

reverses the process of the pitch-spelling problem, extracting *parameters* from a corpus of spelled scores. These parameters can, in turn, be used to instantiate a forward problem.

Along the way, several key gadgets are introduced:

- The representation of spellings with binary encodings.
- The use of a Parsimony Pivot, which provides a powerful way of ensuring parsimony conditions are met, while also evolving into a source and sink structure.
- The Pullback Adjacency Matrix structure, which enables properties on many levels of the score structure to be combined through matrix multiplication and entrywise multiplication, and which also allows key components of the algorithmic mechanism to be modularized into separate matrix contributions.
- Minimum Cut as a *primal* modeling tool.

# Musical Terms

**accidental** symbol to the left of a notehead (or to the right of a letter name) that modifies its pitch by a certain number of half-steps in the scale.

**black note** note with the same pitch as a black key on the piano keyboard.

**chord**  pitched event (with rhythmic/timing information), made up of several sounding pitches (hence featuring more than one notehead).

**double accidental** double sharp  or double flat  accidentals..

**half-step** the smallest unit of distance in pitch between notes on the piano keyboard (between white notes and black notes, from *E* to *F* and from *B* to *C*).

**interval** a way of referring to the distance between the spellings of two pitches. More detail is given in Section 1.2.5.

**letter name** the names given to the white notes of the piano: *C, D, E, F, G, A, B*. Equivalent to *Do, Re, Mi, Fa, So, La, Ti*.

**note**  pitched event (with rhythmic/timing information), made up of a single sounding pitch (hence featuring only one notehead).

**notehead** one of the following symbols: , , . Its vertical position on the stave indicates its letter name, or its pitch up to the action of an accidental.

**octave** the *modulus* of the scale, a period of 12 half-steps or 7 letter names. Pitches separated by octaves sound psychoacoustically very similar and are said to have the same pitch class.

**pitch** frequency of the sound-wave, how “high” or “low” a sound is.

**pitch class** pitch *modulo* octaves. In other words, pitches separated by octaves have the same pitch class. This equivalence reflects the psychoacoustic similarity of these pitches.

**scale** the ordered set of half-step-separated pitch classes from  $C$  up to  $B$ .

**score** a document of notation representing a whole piece of music.

**spelling** an assignment of letter-name-accidental combinations to the pitches of a passage of music.

We will say that a spelling is valid, if it actually represents the passage of music it is supposed (that is that the letter-name-accidental combinations have the same pitch class as the note they are supposed to be representing). Notes of the same pitch class permit the same set of spellings.

**staff** same as stave (plural *staves*).

**stave** five horizontal line grid against which the height of noteheads (hence their pitch) is measured.

**white note** note with the same pitch as a white key on the piano keyboard.

# Bibliography

- [1] James Bean. Personal communication.
- [2] Dimitris Bertsimas. *Introduction to linear optimization*. Athena Scientific series in optimization and neural computation. Athena Scientific, Belmont, Mass., 1997.
- [3] Emilios Cambouropoulos. A general pitch interval representation: Theory and applications. *Journal of New Music Research*, 25(3):231–251, September 1996.
- [4] Emilios Cambouropoulos. Pitch Spelling: A Computational Model. *Music Perception: An Interdisciplinary Journal*, 20(4):411–429, June 2003.
- [5] Elaine Chew. *Mathematical and computational modeling of tonality : theory and applications*. International series in operations research & management science ; 204. Springer, New York, 2014.
- [6] Elaine Chew and Yun-Ching Chen. Real-Time Pitch Spelling Using the Spiral Array. *Computer Music Journal*, 29(2):61–76, June 2005.
- [7] Thomas H. Cormen et al. *Introduction to algorithms*. Cambridge, Mass. : MIT Press, c2009., 2009.
- [8] Michael Scott Cuthbert and Christopher Ariza. music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data. In J. Stephen Downie and Remco C. Veltkamp, editors, *Proceedings of the 11th International Society for Music Information Retrieval Conference, IS-MIR 2010, Utrecht, Netherlands, August 9-13, 2010*, pages 637–642. International Society for Music Information Retrieval, 2010.

- [9] Michael Scott Cuthbert et al. music21: a Toolkit for Computer-Aided Musicology. Open-source software.
- [10] Dorit S. Hochbaum. The pseudoflow algorithm: A new algorithm for the maximum-flow problem. *Operations Research*, 56(4):992–1009, August 2008.
- [11] Dorit S. Hochbaum and James B. Orlin. Simplifications and speedups of the pseudoflow algorithm. *Networks*, 61(1):40–57, January 2013.
- [12] Aline Honingh and Rens Bod. Convexity and the well-formedness of musical objects. *Journal of New Music Research*, 34(3):293–303, November 2005.
- [13] Aline K Honingh. Compactness in the Euler-Lattice: A Parsimonious Pitch Spelling Model. *Musicae Scientiae*, 13(1):117–138, March 2009.
- [14] H. C. Longuet-Higgins. Perception of melodies. *Nature*, 263(5579), October 1976.
- [15] D. Gareth Loy. *Musimathics : the mathematical foundations of music*. MIT Press, Cambridge, Mass. ; London, 2006.
- [16] David Meredith. Comparing pitch spelling algorithms on a large corpus of tonal music. *Computer Music Modeling And Retrieval*, 3310:173–192, 2005.
- [17] David Meredith. The ps13 pitch spelling algorithm. *Journal of New Music Research*, 35(2):121–159, June 2006.
- [18] David Meredith. Optimizing Chew and Chen’s Pitch-Spelling Algorithm. *Computer Music Journal*, 31(2):54–72, June 2007.
- [19] David Meredith. Proof of the equivalence of the spiral array and the line of fifths in Chew and Chen’s pitch-spelling algorithm. *Computer Music Journal*, 31(4):5–7, 2007.
- [20] Alan V. Oppenheim. *Signals and systems*. Prentice-Hall signal processing series. Prentice-Hall, Englewood Cliffs, N.J., 1983.
- [21] Robert Rowe. *Machine musicianship*. MIT Press, Cambridge, Mass., 2001.
- [22] Daniel Spielman and Shang-Hua Teng. Smoothed analysis: an attempt to explain the behavior of algorithms in practice. *Communications of the ACM*, 52(10):76–84, October 2009.

- [23] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. November 2001.
- [24] David Temperley. *The cognition of basic musical structures*. MIT Press, Cambridge, Mass., 2001.
- [25] Q. Zhao, A. Stettner, E. Reznik, D. Segrè, and I. C. Paschalidis. Learning cellular objectives from fluxes by inverse optimization. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 1271–1276, Dec 2015.