Master's Thesis

Harmony Analysis based on Tonal Pitch Space

Hiroyuki Yamamoto

Supervisor Professor Satoshi Tojo

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Information Science)

February, 2021

Abstract

Tonal Pitch Space (TPS) gives us a numerical distance between two chords, and enables us to give multiple interpretations of chord sequences as to their keys and degrees. Therefore, we can find the most plausible (*i.e.* sounds natural for humans) interpretation as the shortest path for the sequence. In this thesis, we present two studies to augment/revise TPS from different aspects. They are closely related but done separately.

First, we try to improve the expressiveness by applying three extensions to TPS and the interpretation graph proposed by Sakamoto *et al.* to deal with jazz harmony well. In the first extension, we augment TPS to cover those chords commonly used in jazz. Thereafter, we propose the notion of ϵ-transition, which is a free reinterpretation of a chord, to represent a pivot chord. Also, we propose the notion of cadential shortcut, which includes multiple chords to express a cadence, given as a shorter directed path in addition to the original sequences of possible interpretations. We show our method successfully capture more information while reducing the ambiguity. Furthermore, we conduct chord reharmonization and cadence evaluation as examples of applications of our method.

Next, we address the arbitrariness in TPS. We generalize the distance calculation system and propose a framework in which we can define new distance elements freely in the form of table and train them with data. In the training procedure, a total distance of a path is converted to a path probability, then update the parameters by gradient descent based on the cross entropy loss. We compare several distance elements and show the best one can considerably improve the prediction accuracy.

Acknowledgments

I would like to express my deep gratitude and appreciation to my supervisor Professor Satoshi Tojo. Without his patient support and guidances, it would have been impossible for me to complete this master's program.

I would also like to extend my appreciation to my supervisor of minor research Professor Yasushi Inoguchi for his great instructions.

In addition, I my very thankful to my colleagues under Tojo laboratory and others for their help and friendshop.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Harmony is one of the most fundamental components of music [15], and, like natural language, there must be some kinds of syntax in harmonic sequences [1]. Harmonic syntax has been utilized in many music applications, *e.g.* audio chord transcription [3], melody harmonization [5], or meter detection [8].

Tonal Pitch Space (TPS) [7] is a music model which provides a foundation to the harmonic analysis by defining the smoothness of chord connection as the numeric distance between two chords, given their keys and degrees. When a chord name is interpreted in multiple ways as to these keys and degrees, a chord sequence also gets multiple interpretations and results in a complicated network of connections. Among which, we can regard the path of connections with the shortest distance as the most natural interpretation.

Sakamoto *et al.*[14] have proposed a method to find the most plausible interpretations of chord progressions by finding the shortest path in the interpretation graph , a directed graph whose edge weights calculated with TPS. However, this method has some limitations: (1) tetrads (*e.g.* seventh chords) are not considered, (2) natural/harmonic/melodic minor scales are not distinguished, (3) every chord has only one interpretation in an interpretation path, (4) direction is not considered, and (5) relations among more than two chords are not considered. Because these limitations tend to result in somewhat coarse analysis especially for jazz, the number of shortest paths often becomes enormous.

To overcome these limitations, in the first study we propose three extensions on TPS and interpretation graph. First, we incorporate tetrads and distinction of three minor scales (*i.e.* natural, harmonic, and melodic minor scales) in TPS to deal with the issues on (1) and (2). Second, we introduce ϵ-transition, which is a notion in automata theory, that tolerates 'free' shift in states, into the interpre-

tation graph, in which a chord progression is regarded as a state change from a vertex to another. This ϵ-transition, however, allows us to reread a chord name to another without progression. This contributes the interpretation of pivot chord, and thus solves (3). Third, we introduce the notion of cadential shortcuts to solve (4) and (5). If a certain sequence of chords matches a typical progression of cadences, we can skip the intermediate progressions and can jump to the cadences. This kind of shortcuts explicitly gives us the direction of chord progression to solve (4), and they enable us to consider tri-grams (or more) to solve (5).

Aside from this, TPS contains some arbitrariness in the structure and coefficients. The distance in TPS is composed of three elements and they are mainly based on music theory and observation, but not, strictly speaking, defined in an objective manner. Furthermore, the model is a little too simple to predict chord interpretations, resulting in a low prediction accuracy.

In the other study in this thesis, we work through these problems by a novel framework which enables us to define new distance elements and train them with data. First we rearrange the distance formula in TPS to the sum of three distance elements. Then generalize it to allow us to add other distance elements, we define these new distance elements in the form of table. Next, we introduce a novel probability formula which gives higher probability to a shorter path. Finally, by differentiating the cross entropy loss function, we calculate the gradient in parameter space and update the parameters using it.

1.2 Thesis Outline

The rest of this thesis is organized as follows: Chapter 2 provides some preliminaries to our studies. In Chapter 3 and 4, we detail our two studies on TPS respectively. Finally, in Chapter 5, we summarize our contributions and state our further directions.

Chapter 2

Preliminaries

2.1 Tonal Pitch Space

2.1.1 Overview

Tonal Pitch Space (TPS) is a music model for the quantitative harmony analysis proposed by Fred Lerdahl [7]. It is proposed to complement Lerdahl's the other music theory the Generative Theory of Tonal Music (GTTM) [6], which applies the generative grammar to extend the Schenkerian theory.

In this model, a chord is interpreted as a pair of a key and a degree in it (*e.g.* interpretations of C major triad are as follows: I/C, III/a, V/F, IV/G, VI/e, and VII/d) and a distance is defined between two chord interpretations. This distance is intended to represent the degree of the perceptual distance, *i.e.* when the distance of two chord interpretations in TPS is small, the transition between them should sound natural to us. By means of this, we can determine the most plausible interpretation pair as the pair which achieves the shortest distance.

A distance in TPS is defined as a combination of three elements explained below.

2.1.2 Regional Distance

The first element is the distance between keys, and is defined as the length of the shortest arc on the regional circle of fifths (Figure 2.1). This distance cannot be calculated between major and minor keys, so one must convert either key to the key with same key signature. For example, to calculate the regional distance between G major and A minor, we must convert G major to E minor, or A minor to C major, then we can calculate the distance, 1 in this case, from the regional circle-of-fifths.

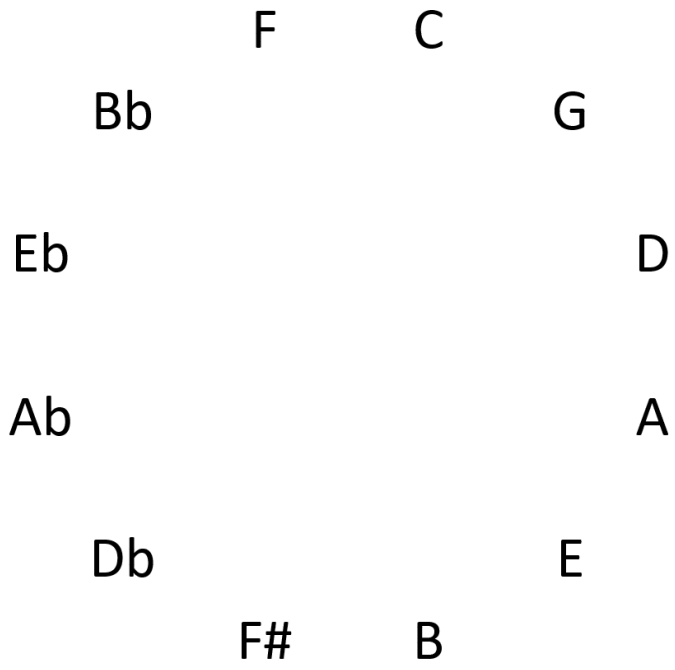

Figure 2.1: regional circle-of-fifths

2.1.3 Chordal Distance

Along with the regional circle-of-fifths, TPS defines chordal circle-of-fifths (Figure 2.2). The second element is the distance between root notes, and is defined as the length of the shortest arc on the chordal circle-of-fifths. Note that though this is also called 'circle-of-fifths', not all adjacent elements are of perfect fifths.

2.1.4 Distance based on the Basic Space

The last element is based on a structure called basic space, which is composed of 5 levels (*i.e.* root, fifth, triadic, diatonic, and chromatic) and each level contains pitch classes reflecting the chord interpretation. The levels are arranged in order of the importance of a pitch class to the chord, level a being the most important. The way how to calculate the distance on basic space is illustrated in Figure 2.3. The distance is defined as the number of missing circles from destination to source. In Figure 2.3, there are five missing circles (which are highlighted), therefore, the distance in this case is 5.

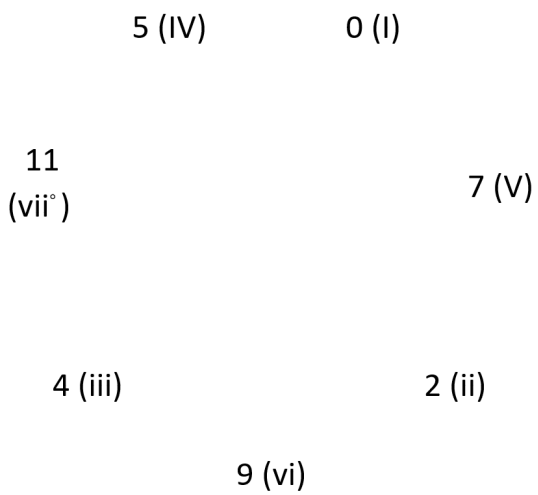

Figure 2.2: chordal circle-of-fifths

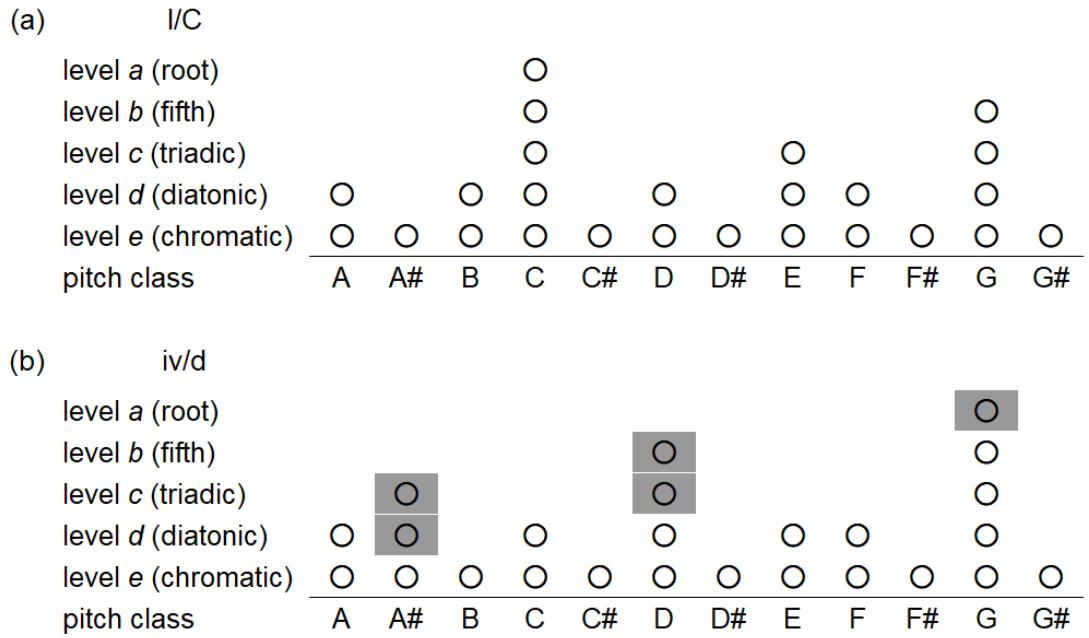

Figure 2.3: basic space from I/C to iv/d

2.1.5 Distance within Related Keys

TPS distinguishes key pairs as related or distant. Related keys are defined as follows:

$$\begin{cases} C(\mathrm{I}) = \{\mathrm{I}, \mathrm{i}, \mathrm{ii}, \mathrm{iii}, \mathrm{IV}, \mathrm{V}, \mathrm{vi}\} & \text{from major key} \\ C(\mathrm{i}) = \{\mathrm{i}, \mathrm{I}, \mathrm{bIII}, \mathrm{iv}, \mathrm{v}, \mathrm{bVI}, \mathrm{bVII}\} & \text{from minor key} \end{cases} \tag{2.1}$$

If we define a boolean function

$$f(x, y) \triangleq C(x) \text{ contains } y$$

this function satisfies $f(x, y) = f(y, x)$ for all (x, y). Key pairs which are not related are regarded as distant.

We can think of a key's related keys as the ones which share all, or all except one, pitch classes with the original key plus parallel key of the original key. Related keys are considered to have so strong structural links with the original key that the modulation between them should sound natural to humans.

The distance between chord interpretations x and y, which are related keys, is defined as a sum of the three distances explained above:

$$\delta(x, y) = region(x, y) + chord(x, y) + basicspace(x, y) \tag{2.2}$$

2.1.6 Distance across Related Keys

If we interpret a transition between distant keys as a direct modulation, it can sound unnatural to us. Instead, in TPS, it is interpreted as a combination of modulations between related keys. For example, modulation from C major to D major should be interpreted as modulations C major $\rightarrow$ G major $\rightarrow$ D major, instead of just C major $\rightarrow$ D major. Note that, there are infinite combinations to realize this, so we must choose the one that achieves the shortest total distance. Also note that, in all passing keys (*e.g.* G major, in this example) the degrees are set to be one. Therefore, the distance is defined as follows:

$$\begin{aligned} \delta(x, y) = \min(\quad & \delta(x, T_{R_1}) + \Delta(R_1, R_n) + \delta(T_{R_n}, y) \\ & |R_1 \in C(R_x), R_n \in C(R_y)) \\ \Delta(R_1, R_n) = \min(& \sum_{i=1}^{n-1} \delta(T_{R_i}, T_{R_{i+1}}) | R_{i+1} \in C(R_i)) \end{aligned} \tag{2.3}$$

As explained above, the distance within related keys (equation 2.2) is composed of the sum of three elements. Now, because equation 2.3 is the sum of quation 2.2s, the resulting distance can also be considered as the sum of the three elements. Therefore, we can rewrite the distance as follows:

$$\delta(x, y) = totalRegion(x, y) + totalChord(x, y) + totalBasicspace(x, y) \tag{2.4}$$

2.2 Former Approaches

Sakamoto *et al.*[14] have applied TPS to analyze chord sequences to find the most plausible interpretation as the shortest path based on the distances described above.

Given a chord sequence, at first, their method extends each chord to its interpretations and constructs a graph whose edges have weights that correspond to the distances on TPS. Then it applies the Viterbi algorithm [16] to find the shortest interpretation paths from the start to the goal. Figure 2.4 shows an interpretation graph for chord sequence C → F → G → C, and one of the shortest interpretation paths is I/C → IV/ C → V/C → I/C.

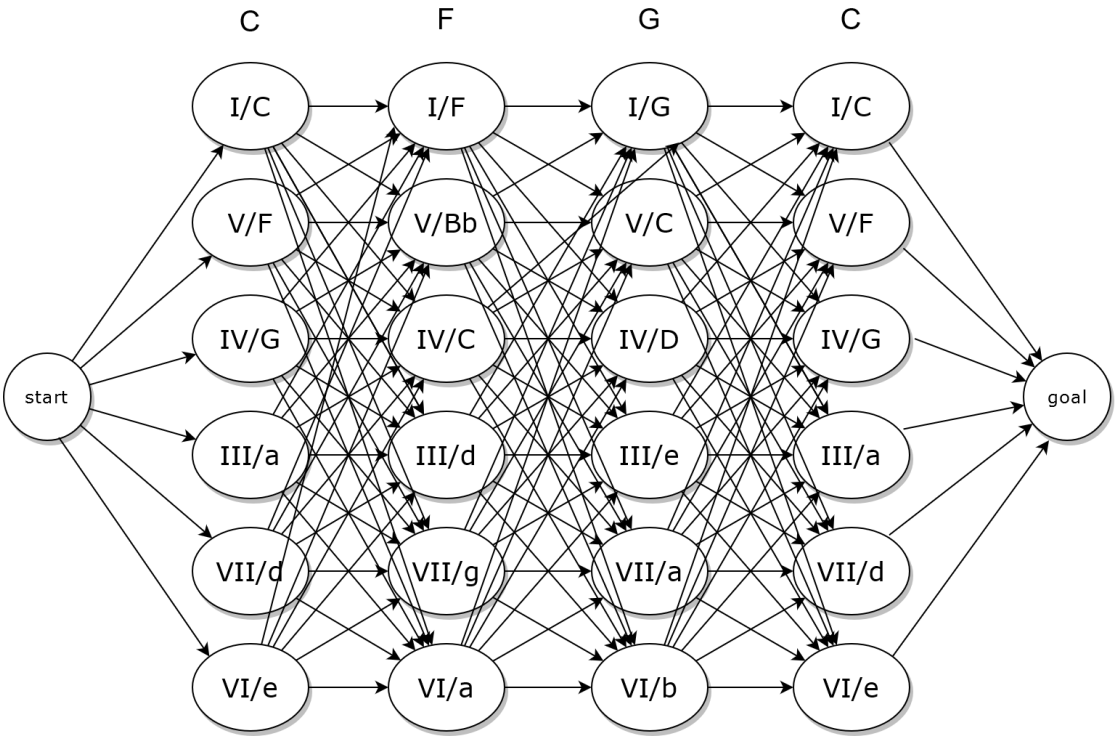

Figure 2.4: interpretation graph

However, because this method is based on TPS, there are several limitations derived from this method. First, TPS basically is a theory for the harmony of period of common practice [11] and only consider major triads and minor triads and does not distinguish three minor scales, but it is insufficient to analyze jazz harmony which belongs to the period beyond common practice and for the most part composed of tetrads and is often characterized by scale-conscious performances. Also, TPS only defines the distances between two chord interpretations, so, for example,

some chord sequences found very often in jazz like II[1] $\rightarrow$ V $\rightarrow$ I cannot be dealt directly with it. Moreover, the distance becomes symmetric (*i.e.* $\delta(x, y) = \delta(y, x)$ for all x and y), but it seems not appropriate for harmony analysis; every chord progression should hold its own meaning, and the reverse order often does not make sense. Finally, the interpretation graph of [14] does not allow double interpretations on a chord in an interpretation path, therefore it cannot express key modulations by pivot chords, which is common in jazz. As a consequence, the original TPS cannot distinguish the subtle differences in the jazz chords, and the analysis results in a number of the shortest paths with the same value.

Additionally, in the effort to improve cadence detection, Matsubara *et al.*[19] have proposed to restrict the minor scale to harmonic one to avoid the ambiguity in chord interpretation, and to revise the candidates of chord interpretations for each Berklee chord name. Also, Yamaguchi *et al.*[20] have proposed to extend the basic space with an additional layer particularly designed to express seventh chord.

[1]We henceforth simplify the degree notation to use only upper case letters and omit accidentals.

Chapter 3

Three Extensions on TPS

3.1 Overview

TPS [7] and the method of Sakamoto *et al.* [14] have several limitations especially when we analyze jazz harmony, so we propose three extensions to handle these limitations. We explain these extensions in detail in section 3.2. Then we conduct experiments to confirm if our approach successfully improve the expressiveness of the model and reduce the ambiguity of the results in section 3.3. And finally, in section 3.4, we show two applications utilizing added information and structure by the extensions.

3.2 Three Extensions

3.2.1 Extension 1: Tetrads and Scales

TPS basically is a theory for the harmony of the period of common practice [11] and only consider triads and does not distinguish three minor scales, but it is insufficient to analyze jazz harmony which belongs to the period beyond common practice and for the most part composed of tetrads[1] and is often characterized by scale-conscious performances. So, as the first extension, we extend chord types and scales as in Table 3.1, which is obtained from [9]. This includes commonly used chord types, available scales, degrees, and chord functions. Some of them are non-diatonic chords and they are penalized according to the number of out-of-scale notes. The penalty cost used here is a parameter (we set this 1 for each

[1] in fact, tetrads are mentioned a little in the literature [7] but we employ Table 3.1 in consideration of the subsequent extensions.

out-of-scale notes[2]). Note that not all of the chord interpretations appeared in the original TPS are included in the table (*e.g.* second-degree triads of major keys).

3.2.2 Extension 2: ϵ-transitions

In the former approach, it is impossible to assign two interpretations at a same time to a chord. Then secondly, we introduce ϵ-transition to express paths which contain interpretation changes within a chord. For example, given a chord sequence A7 $\rightarrow$ Dm7 $\rightarrow$ G7 $\rightarrow$ CM7, we may consider A7 $\rightarrow$ Dm7 is in D (harmonic) minor, and Dm7 $\rightarrow$ G7 $\rightarrow$ CM7 is in C major, then Dm7 is a pivot chord, which bridges the key modulation and needs to be interpreted in both keys. We call this bridging edges ϵ-transitions. To bring this into the calculation, we modify the interpretation graph, as shown in Figure 3.1, duplicating every chord interpretation candidate so that each chord can have at most two interpretations at once without losing applicability of Viterbi algorithm. In Figure 3.1 all diagonal arrows within rectangles are ϵ-transition, for example the white arrow inside the Dm7's rectangle is the one described above. The cost of this interpretation change is a parameter (we set this cost 0.5).

[2]We have chosen this value through our experiments, and the same goes for the other parameters in this chapter.

chord type	**scale**	**degree**	**chord function**
major triad	maj	I	T
	nat	bIII	T
	maj, mel	IV	SD
	maj, har, mel	V	D
dominant seventh	maj*	I	T
	maj*, nat*, har*, mel*	bII	D
	mel	IV	SD
	maj, har, mel	V	D
	nat*, har*, mel*, maj*	bVI	SDM
	nat*, har*, mel*	VII	SD
major seventh	maj	I	T
	nat*, har*, mel*, maj*	bII	SDM
	nat	bIII	T
	maj	IV	SD
	nat, har	bVI	SDM
minor triad	nat, har, mel	I	T
	nat, har	IV	SDM
minor seventh	nat	I	T
	maj, mel	II	SD
	maj	III	T
	nat, har	IV	SDM
	nat	V	D
	maj	VI	T
minor-major seventh	har, mel	I	T
diminished seventh	har	VII	D
half-diminished seventh	nat, har	II	SDM
	maj*	#IV	T
	mel	VII	D

'maj', 'nat', 'har', and 'mel' mean major, natural minor, harmonic minor, and melodic minor scales respectively. '*' is added if the chord is non-diatonic.

Table 3.1: Extended Chord Types

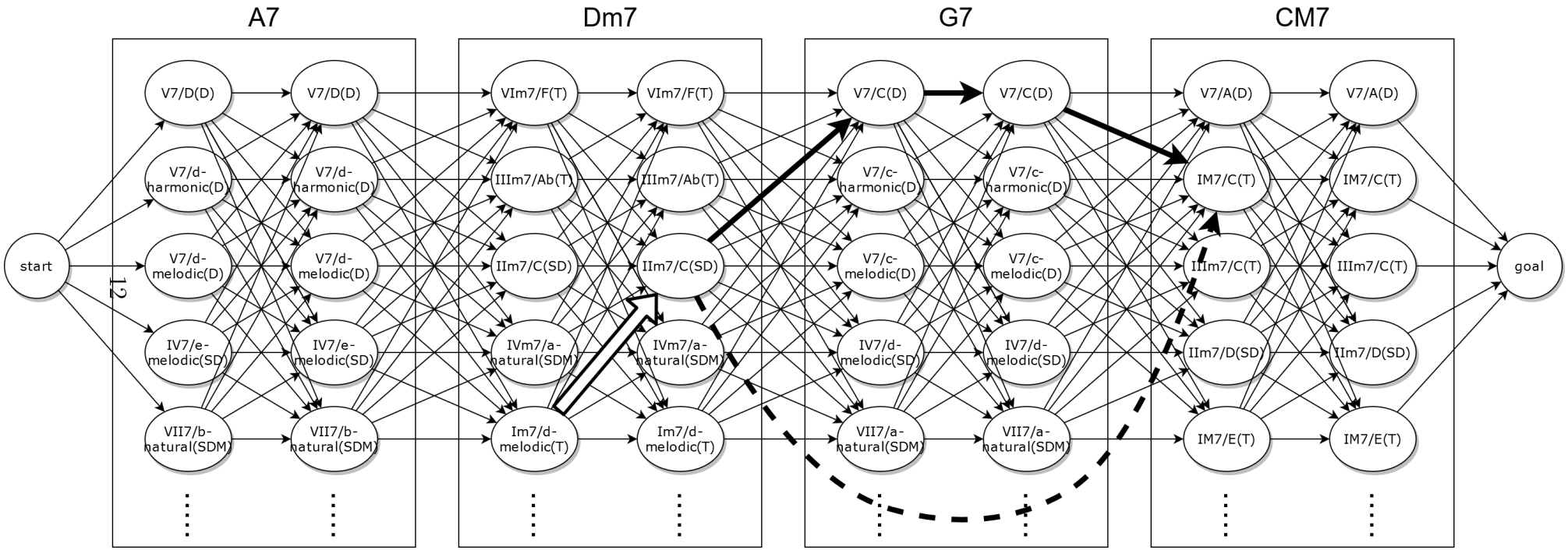

Figure 3.1: extended interpretation graph

3.2.3 Extension 3: Cadential Shortcuts

Thirdly, to handle chord sequences of more than two chords, we extend the interpretation graph further to add new edges, which we call cadential shortcuts. Based on [9], we define the sequence patterns as certain sequences of chord functions and are listed in Table 3.2. These patterns are defined as sequences of chord functions within the same keys and their parallel keys, and the consecutive same functions can be wrapped into one. For example, pattern 5 (SD $\rightarrow$ D $\rightarrow$ T) can be applied to Fm7 $\rightarrow$ Dm7 $\rightarrow$ G7 $\rightarrow$ Cm7. This method also enables us to take the direction into consideration and to assign different costs to reversed chord sequences. We search for all the patterns in Table 3.2 in the graph and for every identical pattern, we add cadential shortcuts connecting from the start node to the end node of the identical patterns and set the cost as that of the original path times 0.5 (this is also a parameter). In Figure 3.1, there found a pattern 5 (SD $\rightarrow$ D $\rightarrow$ T, shown by thick arrows) and added a cadential shortcut (shown by a dashed arrow). Our statistical analysis will be shown later in the next section. This modification also retains the applicability of the Viterbi algorithm to find the shortest paths (for convenience of explanation, we expressed the cadential shortcuts as edges, but they are actually combinations of edges and nodes).

pattern	function sequence
1	D $\rightarrow$ T
2	SD $\rightarrow$ T
3	SDM $\rightarrow$ T
4	SD $\rightarrow$ SDM $\rightarrow$ T
5	SD $\rightarrow$ D $\rightarrow$ T
6	SDM $\rightarrow$ D $\rightarrow$ T
7	SD $\rightarrow$ SDM $\rightarrow$ D $\rightarrow$ T

Table 3.2: Cadence Patterns

3.3 Experiments

3.3.1 Data and Method

We test if our approach can successfully disambiguate the chord interpretations by reducing the number of the shortest paths. We use JazzCorpus [4], which is an annotated corpus of tonal jazz chord sequences of 76 pieces totaling roughly

3,000 chords, and for each music piece we extract every successive nine[3] chords in the sliding window and then apply the analysis described above and calculate the averages of generated graph complexity (in terms of node count and edge count) and the numbers of shortest paths. To compensate those chords which are not included in Table 3.1, we substitute them for similar chords (e.g. Csus4,7 with C7, C#5,7 with C7).

3.3.2 Results

The result is shown in Table 3.3(a). It shows that while our extensions increase the complexity of the interpretation graphs the number of resulting shortest paths is reduced, especially when cadential shortcuts are applied.

This result is of arbitrary nine chords within pieces, so that there exist a lot of unnatural chord sequences to which hardly any cadence patterns can be applied. Then, we apply the same calculations, but this time the whole piece instead of nine chords sliding window, to several well-known jazz standards. The results are shown in Table 3.3(b). Though each piece has different characteristics from the others, it can be seen that in all cases our proposed method can reduce the number of shortest paths.

Figure 3.2 shows the first eight measures of ‘ Fly Me to the Moon ’ along with the results of (a) original method [14] and (b) our proposed method. White arrows represent ϵ-transitions and consecutive dashed arrows represent cadential shortcuts. As can be seen, the result of the original method contains many forks with the same costs, and in this example, there are $2^4 = 16$ paths. On the other hand, our method can find one shortest path. This is especially because the cadential shortcuts prefer those plausible cadences to the other equally costed forks.

[3]We have chosen this number in terms of processing time.

	#nodes	**#edges**	**#shortest paths**
original [14]	56	300	227.49
+ tetrads, 4 scales	55.01	287.22	172.76
+ ϵ-transitions	108.01	617.43	450.27
+ cadential shortcuts	131.58	657.06	74.18

(a) average

	#chords	**original** [14]	**proposed approach**
Fly Me to the Moon	34	8,495,104	384
Autumn Leaves	31	262,144	48
I Got Rhythm	53	over 1B	41,472
Giant Steps	24	193,536	3,888
Afro Blue	18	786,432	8
Blue Monk	13	3,072	2,560
I Remember Clifford	75	58,392,576	768

(b) shortest paths counts for each piece

Table 3.3: Graph Complexity and Shortest Paths Counts

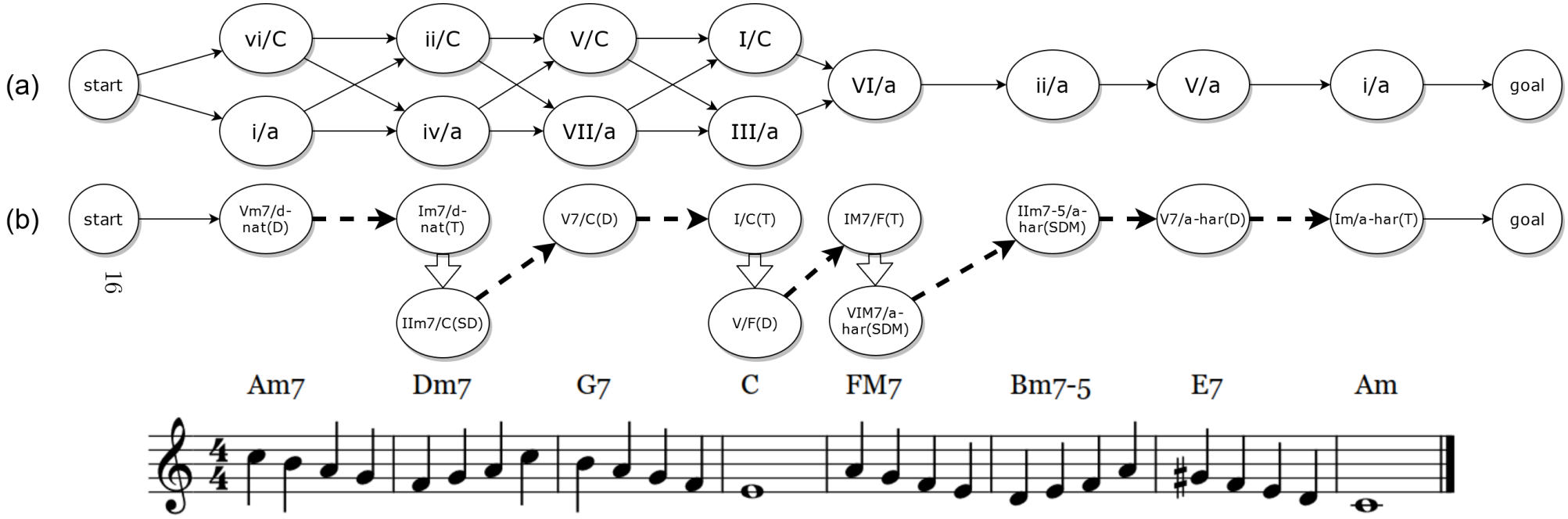

Figure 3.2: analysis of 'Fly Me to the Moon' (a) original method (b) proposed method

3.3.3 Tree Representation

Music have some hierarchical structure, among which, the cadence and modulation between keys are fundamental basis of harmonic aspect. There are a lot of computational approach to obtain the harmonic structures [6, 13, 10, 2]. These studies mainly focus on the generative grammars based on the music theory of the chord function, the cadence and the key. In contract to these approaches, by utilizing the fact that, when our method successfully narrow down the candidate paths, the resulting paths usually contain a lot of cadence shortcuts which are connected with ϵ-transitions, we can generate a tree with a bottom up manner.

Specifically, the steps to generate a tree are as follows:

1. Generate a tree for each cadential shortcut.
    - The trees can be n-ary according to the length of cadential shortcuts, but we adopt (left-brancing) binary tree after the convention of preceding studies.
    - However, every repetition of same functions is grouped under a parent node at first (n-arys are allowed here).
    - The caption of each leaf node is the corresponding function and key in the cadential shortcut, of each internal node is copied from its right most child node's. We omit the scale information here for the sake of ease.
2. In case an ϵ-transion connects two cadential shortcuts, corresponding trees should be merged by one, where the root node of the previous tree as a child of the left most leaf node of the subsequent tree.
3. Generate a tree for each chord which is not assigned to any cadential shortcuts (the tree is therefore composed of only one node). Captions are blank.
4. Line up the trees generate above.

Figure 3.3 shows two examples of generated trees. Double lines represent ϵ-transitions, dashed lines represent cadential shortcuts, and dotted lines represent repetitions of same functions. In the case of Figure 3.3(a), the leftmost tree (which represents the result shown in Figure 3.2(b)) is composed of four cadences interconnected by three modulations. The second and third one have an identical structure with two cadences and one modulations. The last one consists of just one cadence. In the case of Figure 3.3(b), all trees have just one cadence because no ϵ-transition is detected. Overall, our method seems to detect modulation sequences which finish in either relative keys.

Although the structures expressed by these trees are still very coarse, they seem to capture some basic structural facts. So we believe our three extensions especially ϵ-transitions and cadential shortcuts can actually improve the expressiveness.

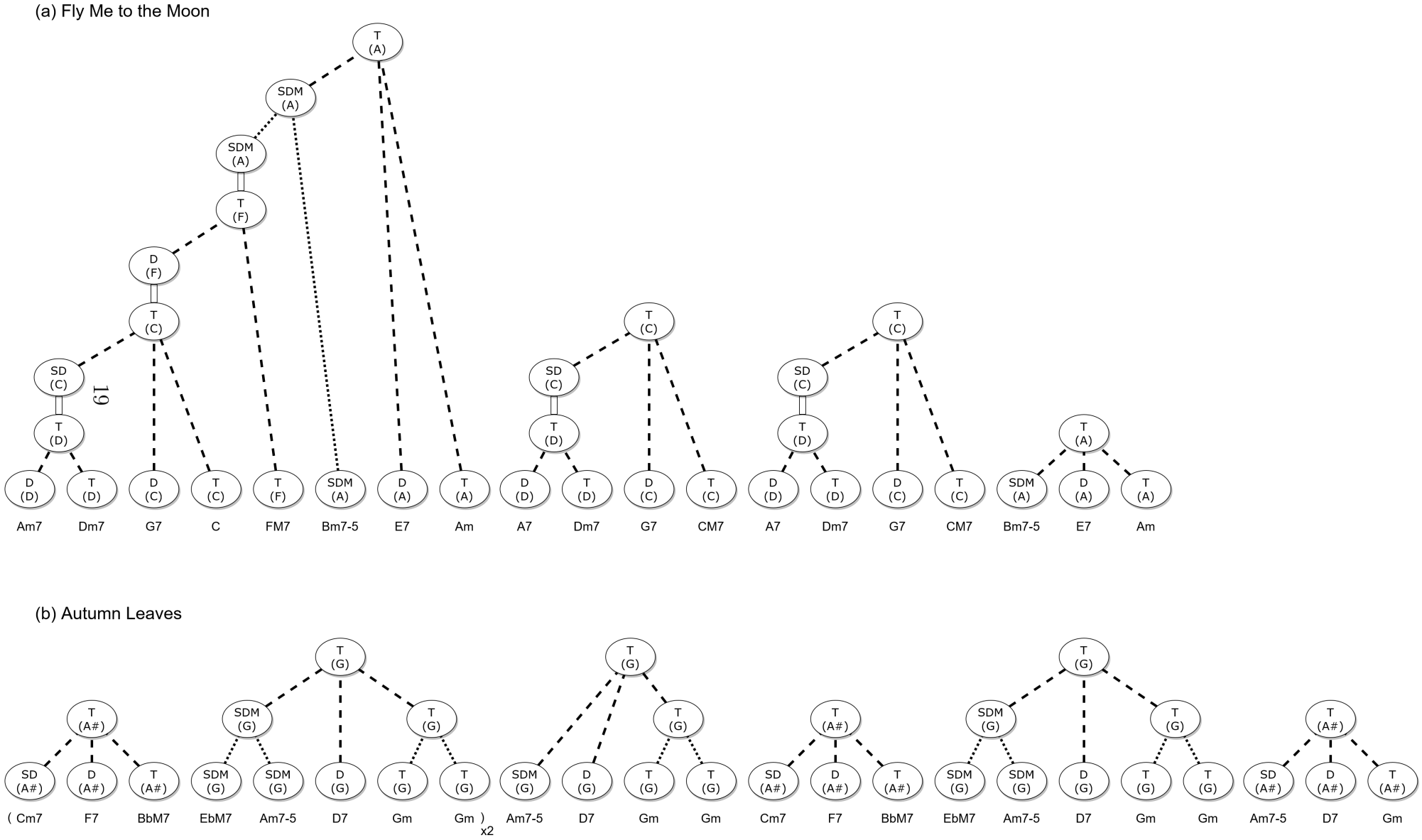

Figure 3.3: generated trees

3.4 Applications

3.4.1 Overview

In this section, we present two applications which utilize the extensions explained above. The first one utilizes the additional information obtained by our method, and the second one the structure of the third extension.

3.4.2 Reharmonization

Overview

As a first application, we propose a method to replace chords based on the information of chord functions. Because cadential shortcuts assign chord functions to each constituent chord, it is possible to replace some chords retaining the rough harmonical meaning.

Method

If a chord is given a chord function by a cadential shortcut, we replace the chord with the chord of the same key, scale and function from Table 3.1 at random (*e.g.* if C is interpreted as T of C major, the candidate set for replacement is {C7, CM7, Em7, Am7, F#m7-5}). If a chord is given two chord functions by two cadential shortcuts interconnected by an ϵ-transition, we search all candidate chords for each chord function and replace the original chord with the intersection of the two candidate chords at random (*e.g.* if F is interpreted both T of D natural minor and SD of C major, the candidate chords for each chord function are {F7, Dm, Dm7} and {F7, Dm7} respectively. Then the intersection is {F7, Dm7}. So we can replace F with F7 or Dm7).

Example

Figure 3.4 shows an (a) before and (b) after example of applying this method. Replaced chords are highlighted by red rectangles. When we listen to the two chord sequences above, we should notice that there is not a significant difference in impression between these two. That is because this method does not change any chord functions, *i.e.* the meaning in harmony. Strictly speaking, it is possible that the method explained in section 3.2 simply reduces the shortest paths completely ignoring human perception. But if the reharmonized chord sequence sounds similar to the original one, which is the case in Figure 3.4, this fact can be seen as a collateral evidence of the effectiveness of our method.

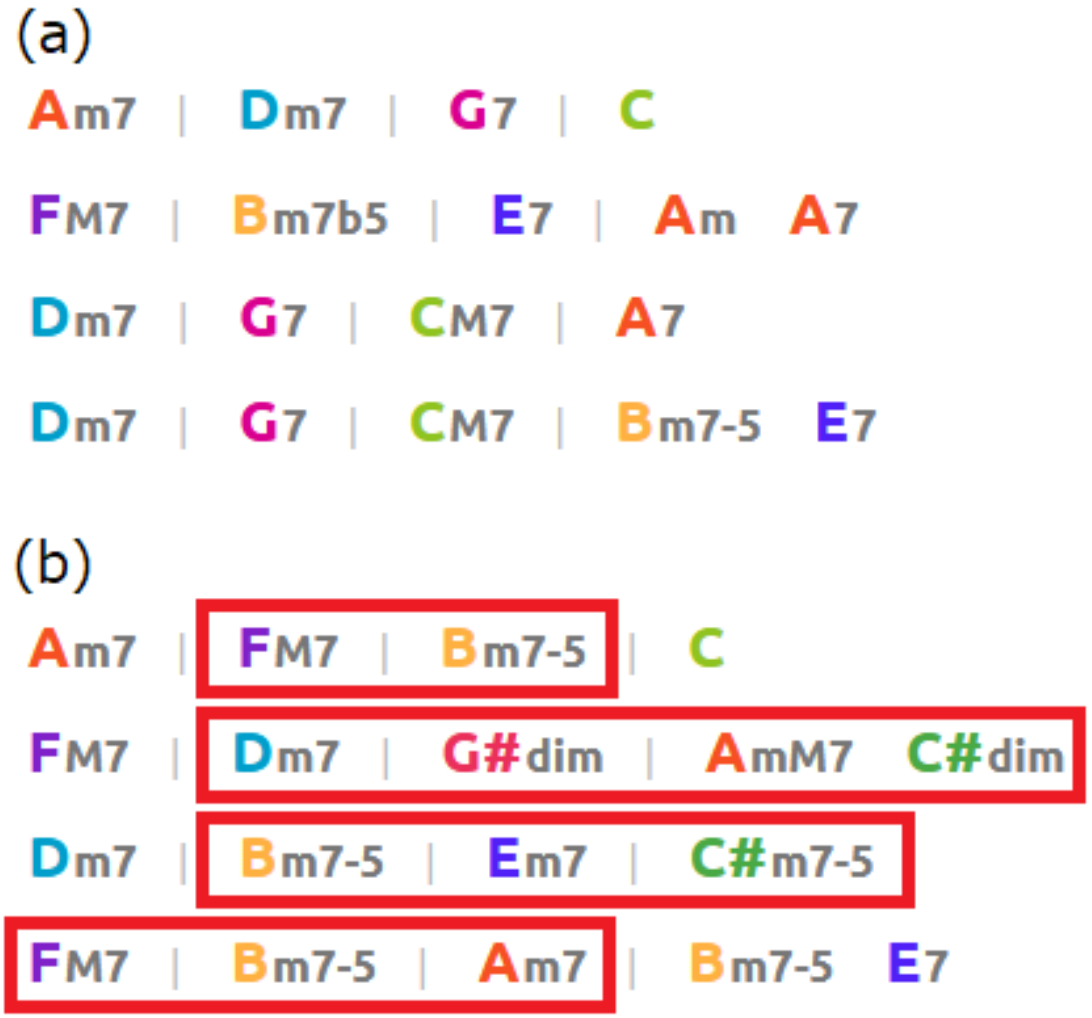

Figure 3.4: analysis of 'Fly Me to the Moon' (a) original method (b) proposed method

3.4.3 Cadence Pattern Evaluation

Overview

In this application, we conduct an experiment utilizing cadential shortcuts in an opposite manner to evaluate cadence patterns themselves. In the foregoing section, we adopted the cadence patterns in Table 3.2 as de-facto standards, but here we drop it and examine all bi-gram and tri-gram patterns. Moreover, we include patterns of degrees (e.g. II $\rightarrow$ V $\rightarrow$ I).

Method

We use nine-chord sliding window again, but this time we replace the chord at the middle of each window with all possible chord candidates (*i.e.* 8 chord types times 12 keys). Figure 3.5 shows an example of (a) the original nine-chord sequence and (b) the modified chord sequence whose fifth chord (*i.e.* FM7) is replaced by all possible chords. We create an interpretation graph from the modified chord sequence and search for the shortest paths. Then we can calculate a chord prediction accuracy from the shortest paths by counting the paths which pass through one of the interpretations of the correct chord (*i.e.* FM7).

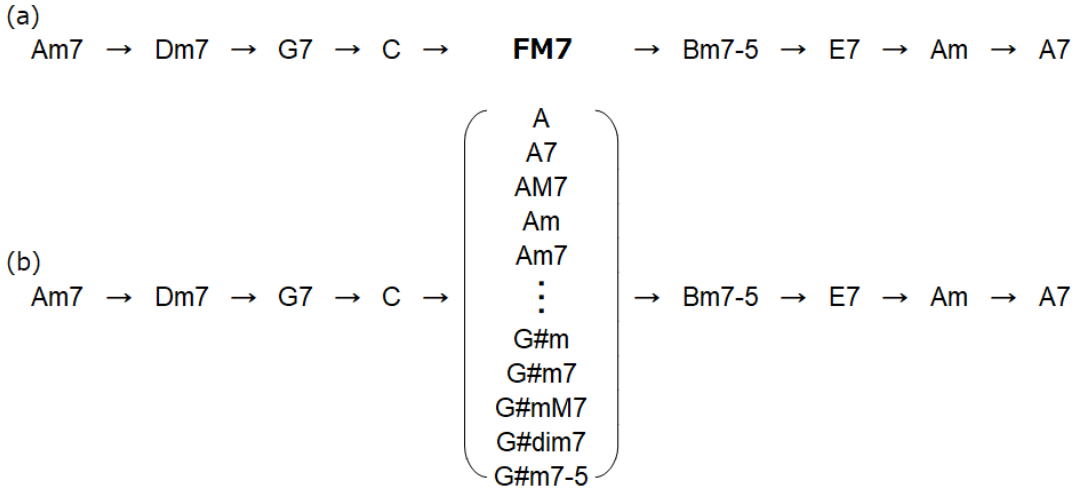

Figure 3.5: an example of nine-chord sequence (a) original sequence (b) modified sequence

We evaluate arbitrary cadence patterns by how much gain (or loss) should them yield through cadential shortcuts compared to the accuracy without any cadential shortcuts (18.8%). Intuitively, better patterns should be able to guide the shortest paths more strongly to the true chord, and vice versa.

For this experiment, we use JazzCorpus[4].

Results

The result is shown in Table 3.4. Common progressions like II → V → I and their relative siblings (*i.e.* IV → VII → III) can be found in the best patterns. And we can find IV → VII → III is better than both IV → VII and VII → III, likewise, II → V → I is better than both II → V and V → I, SD → D → T is better than both SD → D and D → T. This result confirms the limitations of the former approach we mentioned above, that is, the insufficiency of bi-gram. Also, the necessity of distinguishing the directions is corroborated by the fact that D → T is located at the best 11th while T → D at the very worst.

There are some other interesting features we can find in the table for which we have not given explanations yet. For example, there are many third-degree ascensions (*e.g.* I → III, VI → I) in the worst patterns, but some of their reversed patterns can be found in the best patterns.

	best patterns	**acc gain**		**worst patterns**	**acc gain(%)**
1	IV → VII → III	55.85%	1	T → D	-24.47%
2	II → V → I	55.48%	2	I → III	-23.35%
3	II → V	49.41%	3	III → V	-23.19%
4	IV → VII	43.88%	4	VI → I	-22.61%
5	VII → III	28.83%	5	II → IV	-22.29%
6	SD → D → T	27.98%	6	SD → D → SDM	-21.22%
7	SD → D	27.39%	7	D → SDM → T	-21.12%
8	V → I	17.34%	8	IV → VI	-21.06%
9	SD → SDM	17.02%	9	V → II → IV	-20.85%
10	SDM → T	16.76%	10	IV → I	-20.69%
11	D → T	15.00%		T → D → T	-20.69%
12	III → IV	12.82%	12	I → IV	-20.27%
13	I → VI → II	11.17%	13	VI → III	-20.21%
14	VI → II → V	10.74%	14	D → SDM → SD	-20.11%
15	I → II	9.20%	15	II → VI	-20.00%
16	VII → III → IV	8.78%	16	V → VII → IV	-19.31%
17	SDM → SD	7.77%	17	SD → T	-18.78%
18	VII → III → I	7.71%	18	D → SDM	-18.72%
19	I → VI	5.85%	19	T → SD → T	-18.35%
20	IV → II	5.59%	20	IV → II → III	-18.09%
	III → IV → III	5.59%	21	IV → VI → III	-17.66%
22	V → III → VI	4.47%	22	II → VII → I	-17.55%
23	IV → VII → V	4.10%	23	II → VI → I	-17.39%
24	III → I	4.04%		II → IV → I	-17.39%
25	III → VI → II	3.19%	25	IV → V → III	-17.34%
26	SDM → T → SDM	2.87%		II → III → I	-17.34%
27	VI → II	2.66%		D → T → D	-17.34%
	V → I → VI	2.66%	28	IV → I → III	-16.60%
29	SD → SDM → SD	2.18%	29	IV → VII → I	-14.52%
	VII → V	2.18%	30	V → III → IV	-14.36%

Table 3.4: 30 Best and Worst Patterns (**acc gain** represents accuracy gain)

Chapter 4

Distance Learning Model

4.1 Overview

As explained in chapter 2, TPS distance is composed of three elements (*i.e.* region, chord, and basicspace). Although these elements are inspired by both music theory and actual experience, they do not have enough evidence to support the specific settings. Also, they seem to be a little too simple when it comes to predicting chord interpretations.

In this chapter, we propose a novel distance model which generalize the settings in TPS using various distance elements and a method to learn the parameters from human-annotated data. Then, by an experiment, we show our model can successfully achieve higher predicting accuracy than that of TPS.

4.2 Proposed Model

We define symbols and functions as follows:

$\mathcal{X} \triangleq \{\mathrm{I/A}, \mathrm{ii/A}, \mathrm{iii/A}, \cdots, \mathrm{v/g\#}, \mathrm{VI/g\#}, \mathrm{VII/g\#}\}$: the set of chord interpretations

$\mathcal{I} \triangleq \{1, 2, 3, 4.1, 4.2, \cdots, |\mathcal{I}|\}$: the set of distance element indices

$scale : \mathcal{X} \to \{0, 1\}$: the function which maps a chord interpretation to its scale[1] (*e.g.* $scale(iii/A) = 0$, $scale(III/c) = 1$)

$tonic : \mathcal{X} \to \{n \in \mathbb{Z} | 0 \leq n \leq 11\}$: the function which maps a chord interpretation to its tonic note (*e.g.* $tonic(iii/A) = 1$, $tonic(III/c) = 0$)

[1]Here, we only consider major (= 0) and minor (= 1) scales.

$majorTonic(x) \triangleq \begin{cases} tonic(x) \text{ if } scale(x) = 0 \\ (tonic(x) + 3) \ mod \ 12 \text{ otherwise} \end{cases}$ for all $x \in \mathcal{X}$ (*e.g.* $majorTonic(iii/A) = 9$, $majorTonic(III/c) = 3$)

From now on, we use x and y as arbitrary elements in $\mathcal{X}$.

$root : \mathcal{X} \to \{n \in \mathbb{Z} | 0 \leq n \leq 11\}$: the function which maps a chord interpretation to its root note (*e.g.* $root(iii/A) = 1$, $root(III/c) = 3$)

$degree : \mathcal{X} \to \{n \in \mathbb{Z} | 1 \leq n \leq 7\}$: the function which maps a chord interpretation to its degree (*e.g.* $degree(iii/A) = 3$, $degree(III/c) = 3$)

$distanceElement_i : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$: the function which maps a chord interpretation pair to their distance based on the distance element of index $i \in \mathcal{I}$

$b : \mathcal{I} \to \{0, 1\}$: the function which specifies the activation of each distance element

The distance on TPS can be thought of the sum of three distance elements as in equation 2.4. Now we rearrange this equation as a sum of all (active) distance elements.

$$GTPS(x, y) = \sum_{i \in \mathcal{I}} b(i) \cdot distanceElement_i(x, y) \tag{4.1}$$

Then, we add several new distance elements which are defined in the form of tables. With those distance elements, we can freely define which feature to distinguish or ignore to calculate the distance. Also with $b(i)$ term in equation 4.1, we can use any combinations of distance elements.

Note that we allow the distance to be zero or negative, because they do not prevent us from calculating the shortest paths in interpretation graphs. If it is uncomfortable to think a distance to be negative, it is possible to add some constant values to every distance without changing the shortest path as illustrated in Figure 4.1.

4.2.1 Distance Element 1: TPS Region

This ellement corresponds to the *totalRegion* in equation 2.4.

$$distanceElement_1(x, y) \triangleq totalRegion(x, y) \tag{4.2}$$

4.2.2 Distance Element 2: TPS Chord

This ellement corresponds to the *totalChord* in equation 2.4.

$$distanceElement_2(x, y) \triangleq totalChord(x, y) \tag{4.3}$$

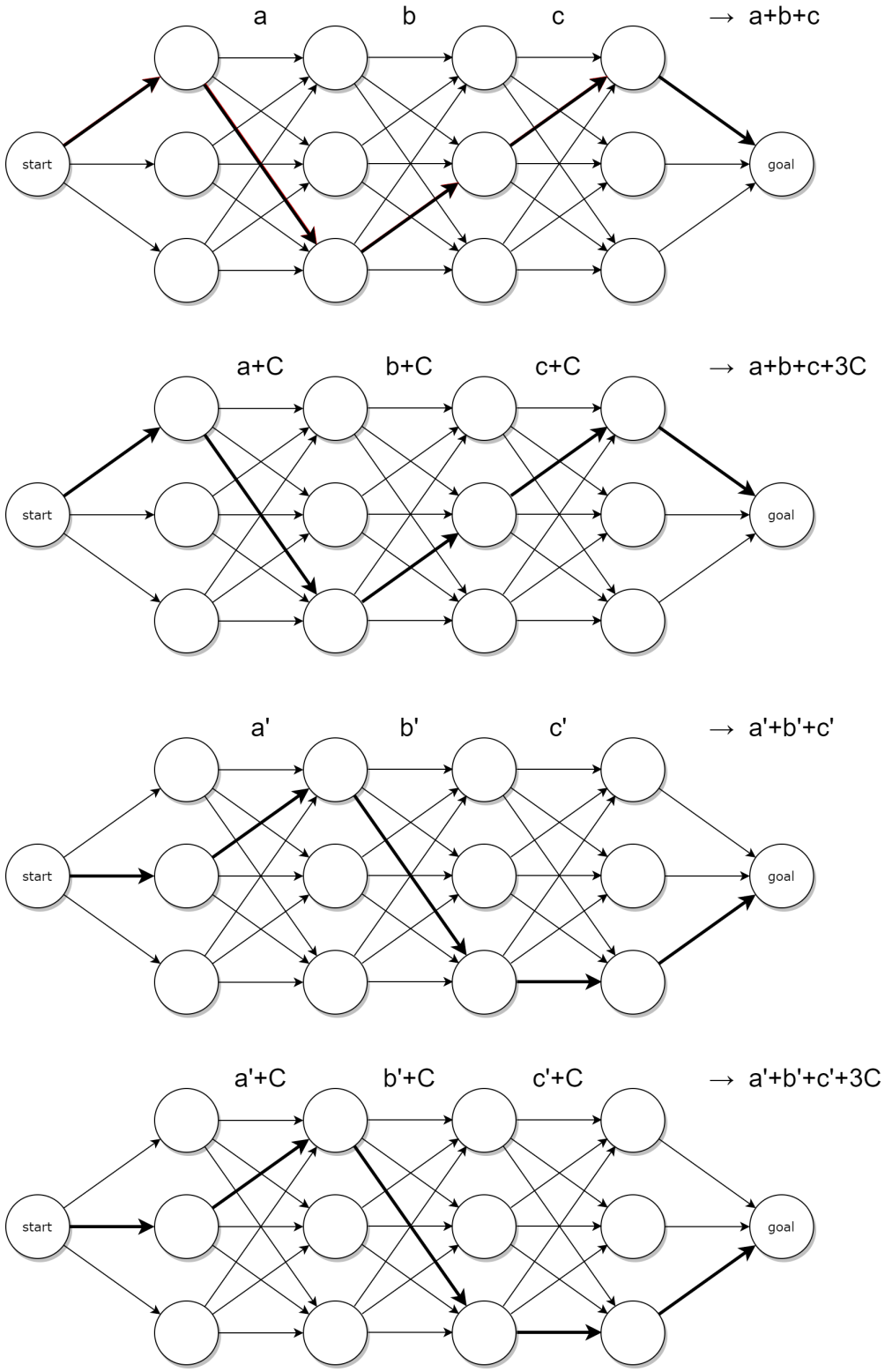

Figure 4.1: an example of adding a constant value C to every distance
If $a + b + c$ is greater than $a' + b' + c'$, $a + b + c + 3C$ is always greater than $a' + b' + c' + 3C$.

4.2.3 Distance Element 3: TPS Basic Space

This ellement corresponds to the *totalBasicspace* in equation 2.4.

$$distanceElement_3(x, y) \triangleq totalBasicspace(x, y) \tag{4.4}$$

(We do not define alternatie distance elements of basicspace. Because, in interpretation graphs, the levels other than diatonic level, which is almost identical to TPS region, do not reflect the difference between each interpretation. Of course $totalBasicspace(x, y)$ can be different when x and y are not in relative keys, but we do not think it have a significance, and, even if it does, we can cover the operation by key-degree distance.)

4.2.4 Distance Element 4: Scale Distance

Distance elements for scale transitions. We define two variants as follows:

DE 4.1: Scale Distance

$$\begin{aligned} &M_{4.1} \in \mathbb{R}^2 \\ &distanceElement_{4.1}(x, y) \triangleq M_{4.1}\left[(scale(y) - scale(x))\ mod\ 2\right] \end{aligned} \tag{4.5}$$

This distant element distinguishes three patterns (*i.e.* major → major, minor → minor, and major → minor or minor → major).

DE 4.2: Asymmetric Scale Distance

$$\begin{aligned} &M_{4.2} \in \mathbb{R}^{2\times 2} \\ &distanceElement_{4.2}(x, y) \triangleq M_{4.2}\left[scale(x), scale(y)\right] \end{aligned} \tag{4.6}$$

This distant element distinguishes the direction (*i.e.* major → minor and minor → major).

4.2.5 Distance Element 5: Tonic Distance

Distance elements for tonic transitions, which generalize *totalRegion* in equation 2.4. We define six variants as follows:

DE 5.1: Relative Tonic Distance

$$\begin{aligned} &M_{5.1} \in \mathbb{R}^7 \\ &distanceElement_{5.1}(x, y) \\ &\triangleq M_{5.1} \left[min \begin{pmatrix} (majorTonic(y) - majorTonic(x)) \ mod \ 12, \\ (majorTonic(x) - majorTonic(y)) \ mod \ 12 \end{pmatrix} \right] \end{aligned} \tag{4.7}$$

Among all variants, this one is conceptually closest to the *totalRegion*.

DE 5.2: Parallel Tonic Distance

$$M_{5.2} \in \mathbb{R}^7$$

$$distanceElement_{5.2}(x, y) \triangleq M_{5.2} \left[min \begin{pmatrix} (tonic(y) - tonic(x)) \ mod \ 12, \\ (tonic(x) - tonic(y)) \ mod \ 12 \end{pmatrix} \right] \tag{4.8}$$

Unlike the relative tonic distance, this one identifies parallel keys (*e.g.* C major and C minor), instead of relative keys (*e.g.* C major and A minor).

DE 5.3: Asymmetric Relative Tonic Distance

$$M_{5.3} \in \mathbb{R}^{12}$$

$$distanceElement_{5.3}(x, y) \triangleq M_{5.3} \left[(majorTonic(y) - majorTonic(x)) \ mod \ 12 \right] \tag{4.9}$$

This one distinguishes the direction of tonic transition over *majorTonic*s.

DE 5.4: Asymmetric Parallel Tonic Distance

$$\begin{aligned} &M_{5.4} \in \mathbb{R}^{12} \\ &distanceElement_{5.4}(x, y) \triangleq M_{5.4} \left[(tonic(y) - tonic(x)) \ mod \ 12 \right] \end{aligned} \tag{4.10}$$

This one distinguishes the direction of tonic transition over *tonic*s.

4.2.6 Distance Element 6: Key Distance

Distance elements for scale-tonic transitions, which can handle both scale transitions and tonic transitions at once. One can calculate them using both scale distances and tonic distances and adding the resulting distances, but this assumes the independence of the transitions of scales and that of tonics. By contrast, key distances can consider the interactions of scales and tonics.

DE 6.1: Symmetric Key Distance

$$
\begin{aligned}
&M_{6.1} \in \mathbb{R}^{2\times 7} \\
&distanceElement_{6.1}(x,y) \\
&\triangleq M_{6.1}\left[(scale(y)-scale(x))\ mod\ 2, min\begin{pmatrix}(tonic(y)-tonic(x))\ mod\ 12,\\ (tonic(x)-tonic(y))\ mod\ 12\end{pmatrix}\right]
\end{aligned}
\tag{4.11}
$$

DE 6.2: Asymmetric Key Distance

$$
\begin{aligned}
&M_{6.2} \in \mathbb{R}^{2\times 2\times 12} \\
&distanceElement_{6.2}(x,y) \triangleq M_{6.2}\left[scale(x), scale(y), (tonic(y)-tonic(x))\ mod\ 12\right]
\end{aligned}
\tag{4.12}
$$

This one distinguishes the direction of scales and tonics.

4.2.7 Distance Element 7: Root-Degree Distance

Distance elements for root note transitions from each degree, which roughly generalize *totalChord* in equation 2.4. We define two variants as follows:

DE 7.1: Symmetric Degree-Root Distance

$$
\begin{aligned}
&M_{7.1} \in \mathbb{R}^{7\times 7} \\
&distanceElement_{7.1}(x,y) \\
&\triangleq M_{7.1}\left[degree(x), min\begin{pmatrix}(root(y)-tonic(x))\ mod\ 12,\\ (tonic(x)-root(y))\ mod\ 12\end{pmatrix}\right]
\end{aligned}
\tag{4.13}
$$

This one calculates distances according to the relative positions of *root*s for each (before) *degree*.

DE 7.2: Asymmetric Degree-Root Distance

$$
\begin{aligned}
&M_{7.2} \in \mathbb{R}^{7\times 12} \\
&distanceElement_{7.2}(x,y) \triangleq M_{7.2}\left[degree(x), (root(y)-tonic(x))\ mod\ 12\right]
\end{aligned}
\tag{4.14}
$$

This one distinguishes the direction in addition to the features above.

4.2.8 Distance Element 8: Key-Degree Distance

Distance elements for key-degree transitions, which can handle both key (*i.e.* scale and tonic) transitions and degree transitions at once. These can distinguish all the combinations TPS can distinguish, thus, they subsume $totalBasicspace$ in equation 2.4.

DE 8.1: Symmetric Key-Degree Distance

$$
\begin{aligned}
&M_{8.1} \in \mathbb{R}^{2\times7\times7\times7} \\
&distanceElement_{8.1}(x, y) \\
&\triangleq M_{8.1}\,[\quad (scale(y) - scale(x))\ mod\ 2, degree(x), \\
&\qquad\qquad degree(y), min\begin{pmatrix}(tonic(y) - tonic(x))\ mod\ 12, \\ (tonic(x) - tonic(y))\ mod\ 12\end{pmatrix}]
\end{aligned}
\tag{4.15}
$$

Note that this one deals with source and destination degrees separately as in asymmetric case.

DE 8.2: Asymmetric Key-Degree Distance

$$
\begin{aligned}
&M_{8.2} \in \mathbb{R}^{2\times7\times2\times12\times7} \\
&distanceElement_{8.2}(x, y) \\
&\triangleq M_{8.2}\,[scale(x), degree(x), scale(y), degree(y), (tonic(y) - tonic(x))\ mod\ 12]
\end{aligned}
\tag{4.16}
$$

4.3 Learning Strategy

We define additional symbols and functions as follows:

G: an interpretation graph with T layers

$G_{s:t}$: from sth layer to tth layer of G ($G_{s:s}$ can be abbreviated as G_s). As a simplified notation, a node in the sth layer can be written as $x \in G_s$, likewise, $x \in G_{s:t}$ be a path from the sth layer to the tth layer, and $x \in G_{s:t-1}||x_t$ be a path from sth layer to the $(t-1)$th layer and added x_t to be the last node.

$x_{s:t}$: from sth element to tth element of an interpretation path x ($x_{s:s}$ can be abbreviated as x_s[2])

[2]in the previous section, we used just x or y to mean them

$x^*_{0:T}$: the ground truth interpretation path

$GTPS_{path}(x_{s:t}) \triangleq \sum_{u=s}^{t-1} GTPS(x_u, x_{u+1})$

We want the calculated distances to allow us to estimate the true interpretation path as a shortest path in the interpretation graph. So we need to learn the parameters to give true interpretation path a shorter total distance than the other interpretation paths.

For that purpose, we first define the path probability formula and then train the parameters by using the gradients on the parameter spaces.

4.3.1 Path Probability

We define the path probability from start node to sth chord interpretation as below:

$$\begin{aligned}
&\mathrm{P}(X_0 = x_0|G_0) \triangleq 1 \\
&\mathrm{P}(X_{0:s} = x_{0:s}|G_{0:s}) \triangleq \prod_{t=0}^{s-1} \frac{\exp(-GTPS(x_t, x_{t+1}))}{denom(G,t)} \\
&\text{where } denom(G,t) \triangleq \sum_{l \in G_t} \sum_{m \in G_{t+1}} \mathrm{P}(X_t = l|G_{0:t}) \exp(-GTPS(l,m))
\end{aligned} \tag{4.17}$$

We can calculate a whole interpretation path as $\mathrm{P}(X_{0:T} = x_{0:T}|G_{0:T})$.

This probability is designed to give higher values to the interpretation paths with shorter total distances.

Theorem 1 (order accordance)**.** *In an interpretation graph G, $GTPS_{path}(x_{0:s})$ is smaller than $GTPS_{path}(x'_{0:s})$ if and only if $\mathrm{P}(X_{0:s} = x_{0:s}|G_{0:s})$ is greater than $\mathrm{P}(X_{0:s} = x'_{0:s}|G_{0:s})$.*

Proof.

$$
\begin{aligned}
&GTPS_{path}(x_{0:s}) < GTPS_{path}(x'_{0:s}) \\
&\Leftrightarrow \exp(-GTPS_{path}(x_{0:s})) > \exp(-GTPS_{path}(x'_{0:s})) \\
&\Leftrightarrow \exp(-\sum_{t=0}^{s-1} GTPS(x_t, x_{t+1})) > \exp(-\sum_{t=0}^{s-1} GTPS(x'_t, x'_{t+1})) \\
&\Leftrightarrow \prod_{t=0}^{s-1} \exp(-GTPS(x_t, x_{t+1})) > \prod_{t=0}^{s-1} \exp(-GTPS(x'_t, x'_{t+1})) \\
&\#\text{divide both sides by the same (positive) value} \\
&\Leftrightarrow \frac{\prod_{t=0}^{s-1} \exp(-GTPS(x_t, x_{t+1}))}{\prod_{t=0}^{s-1} denom(G,t)} > \frac{\prod_{t=0}^{s-1} \exp(-GTPS(x'_t, x'_{t+1}))}{\prod_{t=0}^{s-1} denom(G,t)} \\
&\Leftrightarrow \prod_{t=0}^{s-1} \frac{\exp(-GTPS(x_t, x_{t+1}))}{denom(G,t)} > \prod_{t=0}^{s-1} \frac{\exp(-GTPS(x'_t, x'_{t+1}))}{denom(G,t)} \\
&\#\text{from equation 4.17} \\
&\Leftrightarrow \mathrm{P}(X_{0:s} = x_{0:s} | G_{0:s}) > \mathrm{P}(X_{0:s} = x'_{0:s} | G_{0:s})
\end{aligned}
$$

□

We can calculate the node probability $\mathrm{P}(X_s = x_s | G_{0:s})$ as below:

$$
\begin{aligned}
\mathrm{P}(X_s = x_s | G_{0:s}) &= \sum_{x_{0:s} \in G_{0:s-1} || x_s} \mathrm{P}(X_{0:s} = x_{0:s} | G_{0:s}) \\
&= \sum_{x_{0:s} \in G_{0:s-1} || x_s} \prod_{t=0}^{s-1} \frac{\exp(-GTPS(x_t, x_{t+1}))}{denom(G,t)} \\
&= \sum_{x_{0:s} \in G_{0:s-1} || x_s} \left(\prod_{t=0}^{s-2} \frac{\exp(-GTPS(x_t, x_{t+1}))}{denom(G,t)} \right) \frac{\exp(-GTPS(x_{s-1}, x_s))}{denom(G,t)} \\
&= \sum_{x_{0:s} \in G_{0:s-1} || x_s} \mathrm{P}(X_{0:s-1} = x_{0:s-1} | G_{0:s-1}) \frac{\exp(-GTPS(x_{s-1}, x_s))}{denom(G,t)} \\
&= \sum_{x_{s-1} \in G_{s-1}} \mathrm{P}(X_{s-1} = x_{s-1} | G_{0:s-1}) \frac{\exp(-GTPS(x_{s-1}, x_s))}{denom(G,t)}
\end{aligned}
$$

As we can see, this process has a recursive structure, and, by calculating in a sequential manner from the start node, we can get the node probability with the time complexity linear to the s. Note that, $\mathrm{P}(X_s = x_s | G_{0:s}) = \mathrm{P}(X_s = x_s | G_{0:T})$ is not always the case.

4.3.2 Loss and Gradient

We define a cross entropy loss function as follows:

$$Loss(x_{0:T}|G_{0:T}) \triangleq \sum_{x_{0:T} \in G_{0:T}} -\mathrm{P}^*(X_{0:T} = x_{0:T}) \ln \mathrm{P}(X_{0:T} = x_{0:T}|G_{0:T}) \tag{4.18}$$

Here, P^* is the probability function which only responds to the ground truth:

$$\mathrm{P}^*(X_{0:T} = x_{0:T}) \triangleq \begin{cases} 1 \text{ if } x_{0:T} = x^*_{0:T} \\ 0 \text{ otherwise} \end{cases}$$

We can get the gradient by differentiating *Loss* (4.18) with respect to the parameters, then apply stochastic gradient descent algorithm to update the parameters to minimize the value of *Loss* (4.18), which results in maximizing the path probability for the ground truth path.

4.3.3 Accuracy

We evaluate our model based on how accurately it can predict each chord interpretation by specifying the shortest path in the interpretation graph. If there are more than one shortest paths, we calculate a weighted average for each node in proportion to how many paths go through the node[3] as in Figure(4.2).

We show the algorithm to calculate the path accuracy in Algorithm 1.

4.4 Experiments

4.4.1 Data and Method

We use the dataset annotated in *rntxt* format [17], published at [17, 18]. The dataset is composed of 360 pieces (1,691 phrases, 76,341 chords) and we regard every phrase as a unit (*i.e.*, to which we predict the interpretation path) but when a phrase exceeds 50 chords we divide it into units each of which does not exceed 50 chords, resulting in 2,472 phrases. Then use 1,976 phrases to the training, and 248 phrases to the validation, and remaining 248 phrases to the test

Rntxt format contains a lot of information other than degree/key, but in this study we utilize only key and degree information. About secondary/tertiary chords, we employ local keys (*e.g.*, V/V/V on C major key is interpreted as V on D major key).

[3] this proportion is different from the node probability

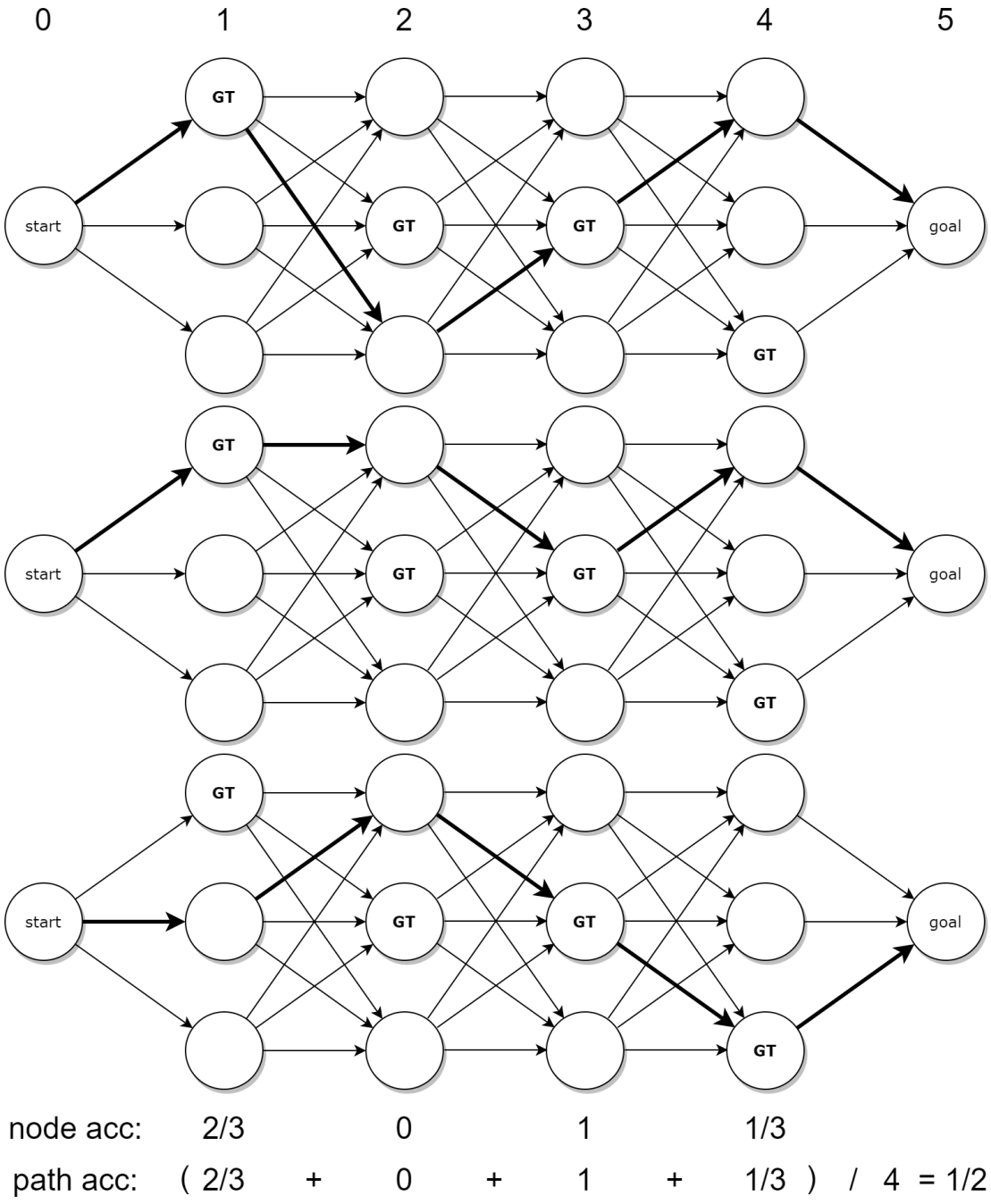

Figure 4.2: sample calculation of path accuracy
Note that, if we look at the last three layers, there are just two unique paths. But the accuracy of the fourth layer is 1/3 instead of 1/2.

Algorithm 1 Algorithm for calculating the path accuracy

Input: a # a list of ground truth interpretations for layer indices
G # an interpretation graph, which is a table of interpretations for layer indices 0 to T-1 and node indices
b # a backward link list calculated by means of Viterbi algorithm (b[i][j] is the list of node indeces in layer i-1 to which there is a link from node index j in layer i)

Output: accuracy

```
c ← zeroList(sizeOf(G)) # a list of the number of paths from each node to the goal node. Initially zero
p ← zeroList(sizeOf(G)) # a list of the probabilities of each node in each layer. Initially zero
c[T - 1][0] ← 1 # path count from goal node to goal node
for i ← T-1 to 1 step -1 do
    for j in G[i] do
        for k in b[i][j] do
            c[i-1][k] ← c[i-1][k] + c[i][j]
p[0][0] ← 1 # node probability of start node
for i ← 0 to T-1 do
    for j in G[i] do
        d ← sum(c[i+1][k] where b[i+1][k] contains j)
        for k where b[i+1][k] contains j do
            p[i+1][k] ← p[i+1][k] + p[i][j] × c[i+1][k] / d
accuracy ← 0
for i ← 1 to T-1 do # exclude start and goal layers
    accuracy ← accuracy + p[i][a[i]]
accuracy ← accuracy / T
```

We set all initial parameter values to be zero and train the models by mini-batch stochastic gradient descent with batch size=100 and learning rate=0.001. We continue training until no accuracy update in validation set for ten epochs in a row, then pick the parameter which gives the highest validation accuracy..

4.4.2 Results

We compare the performances of each distance element and some combinations. The result is shown in the table 4.1.

exp 0 is without any distance elements, just for information.

exp 1 is the original TPS. This one successfully double the accuracy (*i.e.*, narrow down the candidate interpretation by half) from **exp 0**. We consider this one to be the baseline.

We also conduct ablation patterns of TPS (**exp 2-7**). When used alone (**exp 2-4**), $totalBasicspace$ is the best performance (**exp 4**) and achieved almost the same accuracy as the full TPS (**exp 1**). We consider the reason why $totalBasicspace$ is a little better than $totalRegion$ (**exp 2**) is that basic space can exress region distance by the diatonic level and also other levels can give additional information. Seeing the result of **exp 3**, however, $totalChord$ do not improve accuracy well. That is also the case when used two of them together (**exp 5-7**).

In **exp 8-19**, we test each proposed distance elements by themselves. DE 5.1 can accomplish almost the same accuracy as the full TPS (**exp 1, 10**), although it has only seven parameters. DE 4.x cannot improve accuracy at all without distinguishing directions (**exp 8,9**), but surprisingly, for the other distance elements, it turns out that there is very little or no accuracy gain by distinguishing the direction from the comparisons **exp 10** to **exp 12**, **exp 11** to **exp 13**, **exp 14** to **exp 15**, **exp 16** to **exp 17**, and **exp 18** to **exp 19**. We also test tonic distances in which parallel keys are identified (**exp 11, 13**), but they are significantly worse than those of relatie keys (**exp 10, 12**). DE 8.x, being the most complex distance elements, can achieve over 86% accuracy.

In **exp 20-26**, we test some combinations of proposed distance elements. The combinations are selected so that involved distance elements complement each other though not exhaustive. The combination of **exp 23** can achieve 83% with only 58 parameters, likewise, that of **exp 25** and **exp 26** can achieve 85.5% and 86% with a little more parameters. Therefore, it seems that taking the interactions of all scale, tonic, and degree into account is not so important considering the huge parameter size. Also, it is interesting that DE 4.1 have meaningful contribution in **exp 23** here although it does not make difference at all by itself (**exp 8**).

Lastly, we test some combinations of TPS element and distance tables (**exp 27-29**). Root table can be benefited from the elements from TPS (**exp 28**), but in the other combinations, there are not so obvious accuracy gains.

exp	DE 1	DE 2	DE 3	DE 4.1	DE 4.2	DE 5.1	DE 5.2	DE 5.3	DE 5.4	DE 6.1	DE 6.2	DE 7.1	DE 7.2	DE 8.1	DE 8.2	prms	mean	stdev
0																-	0.1900	0.0257
1	○	○	○													-	0.3847	0.1023
2	○															-	0.3780	0.1034
3		○														-	0.1930	0.0288
4			○													-	0.3842	0.1023
5	○	○														-	0.3770	0.1052
6	○		○													-	0.3850	0.1025
7		○	○													-	0.3841	0.1023
8				○												2	0.1900	0.0257
9					○											4	0.2522	0.1432
10						○										7	0.3983	0.1006
11							○									7	0.2908	0.2415
12								○								12	0.3974	0.1003
13									○							12	0.2870	0.2408
14										○						14	0.4249	0.1739
15											○					48	0.5017	0.3380
16												○				49	0.5646	0.1640
17													○			84	0.5741	0.1628
18														○		686	0.8625	0.1780
19															○	2,352	0.8690	0.1717
20				○		○										9	0.3978	0.1015
21					○	○										11	0.5131	0.3402
22						○						○				56	0.7627	0.1585
23				○		○						○				58	0.8301	0.1869
24					○	○						○				60	0.8226	0.1814
25										○		○				63	0.8495	0.1775
26											○		○			132	0.8601	0.1681
27		○	○							○						14	0.4210	0.2318
28	○		○									○				49	0.7309	0.1566
29			○							○		○				63	0.8308	0.1887

Table 4.1: performances of distance elements
prms, **mean**, and **stdev** represent model parameters, mean accuracies, and standard deviations of accuracies

Chapter 5

Conclusion and Future Work

5.1 Conclution

In this thesis, we exhibit two studies on TPS.

Three extensions

In order to overcome some limitations of TPS and interpretation graph, we have proposed three extensions. At first, we extend the original TPS to include tetrads and various minor scales, and thereafter, we have introduced ϵ-transition, that is a distance-free transition in two chords, and cadential shortcut, into the network of possible interpretations of chord progression. Inevitably, we have also considered the tri-grams and the explicit direction in them. As a result, we could reduce the number of the shortest paths and could obtain the efficient algorithm to find the most plausible interpretation, which results in a more reliable method to analyze jazz chord sequence.

Distance Learning Model

In order to tackle the arbitrary nature of TPS and improve prediction accuracies, we generalized the distance calculation in TPS and introduced a new framework to freely define additional distance elements and train them with data. We have defined several experimental distance elements and shown the results, and found a combination of two tables achieved over 80% accuracy with only 77 parameters.

5.2 Future Work

So far, the studies descibed above are, though closely related to each other, conducted independently. Thus, the main direction for our future work is to integrate

them. But there are several more issues to be considered, for example, (1) distinction between long-term and short-term modulations, (2) chords' relative importance, especially related to their lengths or relationships with beats, (3) repititions in chord sequences, and (4) local and global structure.

Bibliography

[1] J. Bharucha, C. Krumhansl: *"The representation of harmonic structure in music: Hierarchies of stability as a function of context"*, *Cognition*, vol. 13, pp. 63-102, 1983

[2] M. Granroth-Wilding, M Steedman: *"A robust parser-interpreter for jazz chord sequences"*, *Journal of New Music Research*, 43(4), pp. 355-374, October 2014

[3] W. bas de Haas, J. P. Magalhães, F. Wiering, *"Improving audio chord transcription by exploiting harmonic and metric knowledge"*, *International Society for Music Information Retrieval Conference (ISMIR)*, 295-300, 2012

[4] *"JazzCorpus"* http://jazzparser.granroth-wilding.co.uk/JazzCorpus.html, 2013

[5] H. V. Koops, J. P. Magalhães, W. bas de Haas, *"A functional approach to automate melody harmonization"* in *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling & design - FARM'13* , p.47, ACM Press, 2013

[6] F. Lerdahl, R. Jackendoff: *"A Generative Theory of tonal music"*, Cambridge, MA, 1983

[7] F. Lerdahl: *"Tonal Pitch Space"*, Oxford University Press, 2001

[8] A. McLeod, M. Steedman, *"Meter detection in symbolic music using a lexicalized PCFG"*, in *Proceedings of the 14th Sound and Music Computing Conference*, 2017

[9] Musashino Academia Musicae: *"Jazz theory workshop"*, ISBN: 978-4990194116, 2005

[10] M. Neuwirth, M. Rohrmeier: *"Towards a syntax of the classical cadence"*, What is a Cadence, pp. 287-338, 2015

[11] W. Piston: *"Harmony"*, W.W. Norton, 1948

[12] H. Riemann, *"Harmony simplified, or The theory of the tonal functions of chords"*, Augener Ltd., 1895

[13] M. Rohrmeier: *"Towards a generative syntax of tonal harmony"*, *Journal of Mathematics and Music*, 5(1), pp. 35-53, March 2011

[14] S. Sakamoto, S. Arn, M. Matsubara, S. Tojo: *"Harmonic analysis based on tonal pitch space"*, in *Proceedings of the 8th International Conference on Knowledge and Systems Engineering (KSE)*, pp. 230-233, 2016

[15] C. Stumpf: *"The origins of music"*, Oxford University Press, Oxford, UK, 2012, First published in 1911, translated by David Trippett

[16] A. Viterbi: *"Error bounds for convolutional codes and an asymptotically optimum decoding algorithm." IEEE transactions on Information Theory*, 13.2: pp. 260-269, 1967

[17] D. Tymoczko, M. Gotham, M. S. Cuthbert, C. Ariza: *"The romantext format: a flexible and standard method for representing roman numeral analyses"*, *International Society for Music Information Retrieval Conference (ISMIR)*, 123-129, 2019

[18] M. S. Cuthbert, C. Ariza: *"Music21: A toolkit for computer-aided musicology and symbolic music data"* in *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, 637-642, 2010

[19] M. Matsubara, T. Kodama, S. Tojo: *"Revisiting cadential retention in GTTM"* in *2016 Eighth international conference on knowledge and systems engineering (KSE)*, 218-223, 2016

[20] N. Yamaguchi, N. Sugamura: *"Improving TPS to tackle non key constituent note"* in 情報処理学会研究報告*, vol.2011-MUS-89 No.10, 1-6, 2011*

Appendix A

Properness of the Path Probability

Theorem 2 (properness)**.** *The path probability function (4.17) is a proper probability mass function.*

Proof. It is trivial that the function (4.17) always returns zero or more, so we show a proof of it being normalized.

I. Base case: $s = 0$

$$\sum_{x_{0:0} \in G_{0:0}} \mathrm{P}(X_{0:0} = x | G_{0:0})$$

\# becuase layer 0 only contains the start node

$$= \mathrm{P}(X_0 = x_0 | G_0) = 1$$

II. Induction case:

Assume the probability for $x_{0:s}$ is normalized:

$$\sum_{x_{0:s} \in G_{0:s}} \mathrm{P}(X_{0:s} = x_{0:s} | G_{0:s}) = 1$$

It follows that:

$$\sum_{x_{0:s+1} \in G_{0:s+1}} \mathrm{P}(X_{0:s+1} = x_{0:s+1} | G_{0:s+1})$$

$$= \sum_{x_{0:s+1} \in G_{0:s+1}} \prod_{t=0}^{s} \frac{\exp(-GTPS(x_t, x_{t+1}))}{denom(G, t)}$$

$$= \sum_{x_{0:s+1} \in G_{0:s+1}} \left(\prod_{t=0}^{s-1} \frac{\exp(-GTPS(x_t, x_{t+1}))}{denom(G, t)} \right) \frac{\exp(-GTPS(x_s, x_{s+1}))}{denom(G, s)}$$

$$= \sum_{x_{0:s+1} \in G_{0:s+1}} \mathrm{P}(X_{0:s} = x_{0:s} | G_{0:s}) \frac{\exp(-GTPS(x_s, x_{s+1}))}{denom(G, s)}$$

$$= \sum_{x_{0:s} \in G_{0:s}} \mathrm{P}(X_{0:s} = x_{0:s} | G_{0:s}) \sum_{x_{s+1} \in G_{s+1}} \frac{\exp(-GTPS(x_s, x_{s+1}))}{denom(G, s)}$$

\# from the assumption

$$= \mathbb{E}_{x_{0:s} \sim \mathrm{P}(X_{0:s} = x_{0:s} | G_{0:s})} \left[\sum_{x_{s+1} \in G_{s+1}} \frac{\exp(-GTPS(x_s, x_{s+1}))}{denom(G, s)} \right]$$

$$= \mathbb{E}_{x_s \sim \mathrm{P}(X_s = x_s | G_s)} \left[\sum_{x_{s+1} \in G_{s+1}} \frac{\exp(-GTPS(x_s, x_{s+1}))}{denom(G, s)} \right]$$

$$= \frac{\mathbb{E}_{x_s \sim \mathrm{P}(X_s = x_s | G_s)} \left[\sum_{x_{s+1} \in G_{s+1}} \exp(-GTPS(x_s, x_{s+1})) \right]}{denom(G, s)}$$

$$= \frac{\sum_{x_s \in \text{nodes in } G_s} \mathrm{P}(X_s = x_s | G_{0:s}) \left(\sum_{x_{s+1} \in G_{s+1}} \exp(-GTPS(x_s, x_{s+1})) \right)}{denom(G, s)}$$

$$= \frac{\sum_{x_s \in \text{nodes in } G_s} \sum_{x_{s+1} \in \text{nodes in } G_{s+1}} \mathrm{P}(X_s = x_s | G_{0:s}) \exp(-GTPS(x_s, x_{s+1}))}{denom(G, s)}$$

$$= \frac{denom(G, s)}{denom(G, s)}$$

$$= 1$$

□

Appendix B

Differentiating the Loss Function

We update the parameters in the distance tables based on the gradient on the parameter space of loss function (4.18). The partial differentiation of *Loss* (4.18) with respect to a parameter i can be calculated as below:

$$
\begin{aligned}
&\frac{\partial}{\partial i} Loss(x_{0:T}|G_{0:T}) \\
&= \frac{\partial}{\partial i} \sum_{x_{0:T} \in G_{0:T}} -\mathrm{P}^*(X_{0:T} = x_{0:T}) \ln \mathrm{P}(X_{0:T} = x_{0:T}|G_{0:T}) \\
&= -\frac{\partial}{\partial i} \ln \mathrm{P}(X_{0:T} = x^*_{0:T}|G_{0:T}) \\
&= \sum_{t=0}^{T-1} \left(\frac{\partial}{\partial i} GTPS(x^*_t, x^*_{t+1}) + \frac{\frac{\partial}{\partial i} denom(G,t)}{denom(G,t)} \right) \\
&= \sum_{t=0}^{T-1} \left(\frac{\partial}{\partial i} GTPS(x^*_t, x^*_{t+1}) \right. \\
&\quad \left. + \frac{\frac{\partial}{\partial i} \sum_{l \in G_t} \sum_{m \in G_{t+1}} \mathrm{P}(X_t = l|G_{0:t}) \exp(-GTPS(l,m))}{denom(G,t)} \right) \\
&= \sum_{t=0}^{T-1} \left(\frac{\partial}{\partial i} GTPS(x^*_t, x^*_{t+1}) \right. \\
&\quad \left. + \frac{\sum_{l \in G_t} \sum_{m \in G_{t+1}} \left(\begin{array}{l} \left(\frac{\partial}{\partial i} \mathrm{P}(X_t = l|G_{0:t})\right) \exp(-GTPS(l,m)) \\ +\mathrm{P}(X_t = l|G_{0:t})(-\frac{\partial}{\partial i} GTPS(l,m)) \exp(-GTPS(l,m)) \end{array} \right)}{denom(G,t)} \right)
\end{aligned}
$$

Here, $\mathrm{P}(X_t = l|G_{0:t})$ and $denom(G,t)$ appear in the calculation process of path probability, so we can reuse the results. Also, $GTPS(x^*_t, x^*_{t+1})$ and $\frac{\partial}{\partial i} GTPS(x^*_t, x^*_{t+1})$

can be calculated easily, because $GTPS(4.1)$ is composed of simple sums and products. Lastly, $\frac{\partial}{\partial i}\mathrm{P}(X_t = l|G_{0:t})$ in $\frac{\partial}{\partial i}denom(G,s)$ can be calculated as below:

$$
\begin{aligned}
&\frac{\partial}{\partial i}\mathrm{P}(X_s = l|G_{0:s}) \\
&= \frac{\partial}{\partial i} \sum_{x_{0:s} \in G_{0:s-1}||l} \mathrm{P}(X_{0:s} = x_{0:s}|G_{0:s}) \\
&= \frac{\partial}{\partial i} \sum_{x_{0:s} \in G_{0:s-1}||l} \prod_{t=0}^{s-1} \frac{\exp(-GTPS(x_t, x_{t+1}))}{denom(G,t)} \\
&= \frac{\partial}{\partial i} \sum_{x_{0:s} \in G_{0:s-1}||l} \left(\prod_{t=0}^{s-2} \frac{\exp(-GTPS(x_t, x_{t+1}))}{denom(G,t)} \right) \frac{\exp(-GTPS(x_{s-1}, x_s))}{denom(G,s)} \\
&= \frac{\partial}{\partial i} \sum_{x_{0:s} \in G_{0:s-1}||l} \mathrm{P}(X_{0:s-1} = x_{0:s-1}|G_{0:s-1}) \frac{\exp(-GTPS(x_{s-1}, x_s))}{denom(G,s-1)} \\
&= \frac{\partial}{\partial i} \sum_{x_{s-1} \in G_{s-1}} \mathrm{P}(X_{s-1} = x_{s-1}|G_{0:s-1}) \frac{\exp(-GTPS(x_{s-1}, x_s))}{denom(G,s-1)} \\
&= \sum_{x_{s-1} \in G_{s-1}} \left(\left(\frac{\partial}{\partial i} \mathrm{P}(X_{s-1} = x_{s-1}|G_{0:s-1}) \right) \frac{\exp(-GTPS(x_{s-1}, l))}{denom(G,s-1)} \right. \\
&\quad + \mathrm{P}(X_{s-1} = x_{s-1}|G_{0:s-1}) \frac{-\frac{\partial}{\partial i} GTPS(x_{s-1}, l) \exp(-GTPS(x_{s-1}, l))}{denom(G,s-1)} \\
&\quad \left. + \mathrm{P}(X_{s-1} = x_{s-1}|G_{0:s-1}) \exp(-GTPS(x_{s-1}, l)) \left(-\frac{\frac{\partial}{\partial i} denom(G,s-1)}{(denom(G,s-1))^2} \right) \right)
\end{aligned}
$$

And, $\frac{\partial}{\partial i}\mathrm{P}(X_0 = x_0|G_0) = 0$. This calculation also has the recurssive structure, and then it can be done with the time complexity linear to s.

Appendix C

Learned Tables

We show some learned tables here along with the experiment index in table 4.1.

exp 8 (mean accuracy: 0.1900)

same	**different**
-9.1420	9.1420

Table C.1: Distance Element 4.1

exp 9 (mean accuracy: 0.2522)

m → m	**m → M**	**M → m**	**M → M**
-0.0805	2.1005	2.1065	-4.1265

Table C.2: DE 4.2 Asymmetric Scale Distance

M and **m** represent major and minor respectively.

exp 10 (mean accuracy: 0.3983)

0	**1**	**2**	**3**	**4**	**5**	**6**
-3.5671	1.7843	0.1090	0.5144	0.4938	-0.7986	1.4642

Table C.3: DE 5.1 Symmetric Relative Tonic Distance

exp 11 (mean accuracy: 0.2908)

0	1	2	3	4	5	6
-3.1773	1.0981	0.3198	0.3420	0.4696	-0.4395	1.3873

Table C.4: DE 5.2 Symmetric Parallel Tonic Distance

exp 12 (mean accuracy: 0.3974)

0	1	2	3	4	5	6	7	8	9	10	11
-3.9559	1.7830	-0.3763	0.5761	0.1307	-1.3396	2.4011	-1.0155	0.5497	-0.1429	0.0119	1.3777

Table C.5: DE 5.3 Asymmetric Relative Tonic Distance

exp 13 (mean accuracy: 0.2870)

0	1	2	3	4	5	6	7	8	9	10	11
-3.1325	1.2794	0.3784	-0.0616	0.3652	-0.7204	1.0724	-0.7232	0.3251	-0.0661	0.2336	1.0496

Table C.6: DE 5.4 Asymmetric Parallel Tonic Distance

exp 14 (mean accuracy: 0.4249)

	0	1	2	3	4	5	6
same scale	-4.5291	1.4668	-0.7550	-0.7202	0.3703	-1.9399	1.5414
different scale	0.5231	2.2363	0.0065	0.0185	0.5357	-0.6334	1.8790

Table C.7: DE 6.1 Symmetric Key Distance

exp 15 (mean accuracy: 0.5017)

	0	1	2	3	4	5	6	7	8	9	10	11
m → m	-4.0805	0.5347	0.6193	0.6912	0.7210	1.5947	0.3386	0.8221	0.7171	0.6523	0.8196	0.4392
m → M	0.1487	0.6921	0.0095	-0.8318	0.7684	-1.1241	0.9943	-1.5331	-0.2327	0.5193	0.2843	1.0473
M → m	-0.1208	1.0329	-0.1088	0.5299	0.0316	-1.4335	0.6568	-1.2028	0.6335	-0.7179	0.0266	0.9415
M → M	-4.1080	0.7495	-0.1991	-0.2233	0.7713	-1.5880	0.8108	-1.4034	0.9633	-0.7715	-0.4024	0.5202

Table C.8: DE 6.2 Asymmetric Key Distance

exp 16 (mean accuracy: 0.5646)

	0	1	2	3	4	5	6
I	-2.1364	-0.2428	-1.2300	-0.9671	-0.8545	-1.9446	-0.9838
II	-1.4643	-0.4901	0.8658	1.7406	1.1741	-2.1727	-0.5462
III	0.7865	1.9792	1.5660	1.2300	1.8880	0.4950	2.1298
IV	-0.4006	0.0876	-0.1672	0.5432	0.7467	-0.8113	0.4326
V	-3.6247	-0.5544	0.0197	-0.0175	-0.4645	-1.8448	0.3572
VI	-0.6037	1.3931	-0.2668	1.5488	1.1821	-0.1407	0.3900
VII	-0.9196	0.3194	0.1389	0.9728	-0.4951	0.8346	0.5217

Table C.9: DE 7.1 Symmetric Root-Degree Distance

exp 17 (mean accuracy: 0.5741)

	0	1	2	3	4	5	6	7	8	9	10	11
I	-1.9201	-0.3975	-2.2787	-0.3055	-0.6012	-1.0834	-1.5807	-2.6999	-0.4727	-0.9351	-1.0649	-1.4854
II	-1.4865	1.5211	-0.6794	2.0620	0.1544	0.2403	-0.1252	-2.0222	1.7608	0.2322	1.4291	-0.9605
III	0.4270	1.8276	0.4380	2.1739	1.2958	0.4127	2.1799	0.8899	2.6460	1.3481	1.2776	1.3829
IV	-1.2916	1.3897	-0.6210	0.5828	0.5750	0.0990	-0.1995	-1.6577	0.9013	0.3302	1.1225	-1.1953
V	-2.8522	-0.2712	-0.0217	-0.1476	-0.0296	-1.2315	-0.3548	-1.7013	-1.6602	-0.6881	-0.8627	-0.6053
VI	-0.1232	3.0267	-1.1130	1.7791	-0.1045	0.4096	0.1881	-0.7840	2.1779	0.2436	0.7574	-0.1274
VII	-1.4845	0.0222	0.8162	1.1288	-0.7300	0.9453	0.2520	-0.2731	-1.0152	0.8859	0.6409	-0.7285

Table C.10: DE 7.2 Asymmetric Root-Degree Distance

exp 20 (mean accuracy: 0.3978)

same	different
-1.4857	1.4856

Table C.11: DE 4.1 Symmetric Scale Distance

0	1	2	3	4	5	6
-5.8407	1.8882	-0.1762	0.5447	0.4020	-0.2239	3.4059

Table C.12: DE 5.1 Symmetric Relative Tonic Distance

exp 21 (mean accuracy: 0.5131)

m → m	m → M	M → m	M → M
7.9310	0.7848	-0.1483	-8.5676

Table C.13: DE 4.2 Asymmetric Scale Distance

0	1	2	3	4	5	6
-3.7589	2.0313	-0.3118	-0.0394	0.2855	-1.3292	3.1225

Table C.14: DE 5.1 Symmetric Relative Tonic Distance

exp 22 (mean accuracy: 0.7627)

0	1	2	3	4	5	6
-4.2126	2.0075	-0.1807	0.3806	0.6485	-0.8862	2.2429

Table C.15: DE 5.1 Symmetric Relative Tonic Distance

	0	1	2	3	4	5	6
I	-1.4555	0.5248	-0.9321	-0.9356	-0.1057	-1.0672	-1.4584
II	-0.9990	0.9874	-0.0976	1.4389	0.9192	-0.8180	0.2644
III	0.5509	1.1457	2.7326	2.0825	1.8152	1.0904	0.5746
IV	-0.9032	0.4555	0.6506	0.3071	0.9307	-0.6415	0.4745
V	-3.0040	-1.0570	-1.2147	-0.7959	-1.0415	-1.5849	-1.3887
VI	-1.0994	1.0170	0.7277	1.2165	0.8760	0.0874	0.0735
VII	-1.2973	-0.1743	0.9229	0.8209	-0.4374	-0.3001	0.1224

Table C.16: DE 7.1 Symmetric Root-Degree Distance

exp 23 (mean accuracy: 0.8301)

same	different
-1.3284	1.3284

Table C.17: DE 4.1 Symmetric Scale Distance

0	1	2	3	4	5	6
-4.7035	2.3838	-0.3904	0.1168	0.1830	-0.7972	3.2076

Table C.18: DE 5.1 Symmetric Relative Tonic Distance

	0	1	2	3	4	5	6
I	-1.1163	0.8278	-0.8286	-0.2953	-0.2934	-0.2274	-1.2769
II	-0.5993	1.2294	0.3006	1.7796	1.4394	0.0974	0.6483
III	0.6651	1.8481	0.3599	0.6069	1.4840	0.6341	0.9252
IV	-0.6827	0.8744	0.1457	0.1562	-0.0750	-0.6584	1.1782
V	-2.6282	-1.5102	-0.6740	-0.7721	-1.0519	-0.9889	-2.5869
VI	-0.2773	1.0749	-0.2257	0.7386	0.8021	0.1763	-0.3795
VII	-1.5505	0.1360	-0.6268	0.8390	-0.3952	-0.0479	0.8014

Table C.19: DE 7.1 Symmetric Root-Degree Distance

exp 24 (mean accuracy: 0.8226)

m → m	m → M	M → m	M → M
-1.4968	2.0810	0.9054	-1.4897

Table C.20: DE 4.2 Asymmetric Scale Distance

0	1	2	3	4	5	6
-4.7924	2.8432	-0.2835	0.2135	-0.1444	-0.6909	2.8546

Table C.21: DE 5.1 Symmetric Relative Tonic Distance

	0	1	2	3	4	5	6
I	-1.4962	-0.0177	-1.5910	-1.2736	-0.5644	-2.4248	-1.2053
II	-0.2567	2.3961	0.8835	3.5469	2.7362	0.0857	0.7835
III	0.4598	2.0948	0.8605	2.7362	3.9246	0.5282	0.8618
IV	-2.8761	0.4976	-0.6746	-0.2275	0.0450	-0.9408	-0.1462
V	-4.6433	-1.2227	-2.6818	-1.4713	-2.0131	-1.3421	-2.7926
VI	-0.7158	2.5107	-0.9377	1.5766	2.2842	0.2702	0.0889
VII	-2.7079	0.7241	0.1875	3.0367	1.0382	-0.8398	0.9053

Table C.22: DE 7.1 Symmetric Root-Degree Distance

exp 25 (mean accuracy: 0.8495)

	0	1	2	3	4	5	6
same scale	-4.3479	1.1185	-0.3396	-0.2549	0.3644	-1.5989	1.0589
different scale	0.5935	1.8052	0.3880	-0.1346	0.4425	-0.1841	1.0889

Table C.23: DE 6.1 Symmetric Key Distance

	0	1	2	3	4	5	6
I	-0.6811	0.6933	-0.4004	-0.4410	-0.1498	-0.5408	-0.9726
II	-0.5044	0.8811	0.0271	1.1171	0.5425	0.0053	0.2244
III	0.5616	0.4865	1.1036	1.0527	1.2224	1.0255	0.5324
IV	-0.9107	0.9280	0.1347	0.1765	0.6473	0.1758	-0.1181
V	-2.7061	-1.0805	-0.8100	-0.6738	-1.0829	-0.6197	-0.9695
VI	0.0597	1.2621	-0.0252	0.7898	0.2470	0.1686	-0.2000
VII	-1.1013	-0.0228	-0.2167	0.7258	-0.5920	-0.0604	0.0891

Table C.24: DE 7.1 Symmetric Root-Degree Distance

exp 26 (mean accuracy: 0.8601)

	0	1	2	3	4	5	6	7	8	9	10	11
m → m	-4.6702	0.6296	1.0156	0.3040	0.7798	-0.0841	0.4645	-0.0546	0.7311	0.4758	0.5950	0.7532
m → M	0.4585	0.7105	0.4104	-0.2051	0.5977	0.2976	0.6990	-0.7562	-0.0385	0.7601	0.3898	0.7186
M → m	0.6366	0.6263	0.0118	0.5935	0.1331	-0.2731	0.7573	-0.6503	0.6228	-0.1717	0.4816	0.7297
M → M	-4.2386	0.6887	-0.8995	-0.6776	0.3238	-2.0665	0.8252	-1.8150	0.6786	-1.0249	-0.6993	0.4256

Table C.25: DE 6.2 Asymmetric Key Distance

	0	1	2	3	4	5	6	7	8	9	10	11
I	-0.7846	0.2012	-1.1181	-0.6599	-0.0854	-1.2274	-0.5275	-0.9145	0.0018	-0.1719	-0.3207	0.6099
II	-0.0888	0.3121	-0.1132	1.3319	0.6036	0.1743	0.3521	0.1246	0.3674	0.4958	0.0565	0.7932
III	0.5100	0.3069	0.7551	0.8806	0.7171	0.6353	0.4982	0.8885	1.4533	1.0428	1.2905	0.9273
IV	-1.0089	-0.3286	0.0088	0.2255	-0.4282	-0.2808	0.2926	-0.1518	0.3864	-0.0159	1.1027	0.7816
V	-3.0607	-0.6654	-1.1435	-1.0418	-0.5856	-1.0516	-0.9890	-0.7324	-0.9687	-1.0162	-0.7775	-0.6387
VI	-0.0050	0.4755	-0.2038	0.5977	0.1136	0.5360	0.0029	0.4213	0.7213	0.1303	0.9610	0.6919
VII	-1.8355	-0.0802	0.1853	0.9261	-0.6274	-0.3199	0.0685	0.3828	-0.4321	-0.0005	0.0746	-0.0145

Table C.26: DE 7.2 Asymmetric Root-Degree Distance

exp 27 (mean accuracy: 0.4210)

	0	1	2	3	4	5	6
same scale	-2.6892	0.1660	-1.2418	-0.4851	0.1629	-0.9612	0.0357
different scale	1.4732	0.6458	0.9513	1.6348	1.3342	-1.3819	0.3555

Table C.27: DE 6.1 Symmetric Key Distance

exp 28 (mean accuracy: 0.7309)

	0	1	2	3	4	5	6
I	-1.7124	-0.2856	-1.0896	-0.7193	-0.4216	-1.1785	-0.4627
II	-0.8944	0.2632	0.0719	1.0884	0.6311	-1.2397	0.0343
III	0.6594	0.4868	2.1172	1.5341	1.5659	1.4235	0.1188
IV	-0.8169	0.8430	0.1607	0.6808	0.4742	-0.7168	0.3717
V	-3.0319	-1.0239	-1.0646	-0.3079	-0.3995	-1.7175	-0.5103
VI	-0.0950	1.0500	0.5974	1.4117	1.1828	0.3587	0.2392
VII	-1.0077	-0.0867	0.3995	1.4172	-0.2055	-0.1742	-0.0192

Table C.28: DE 7.1 Symmetric Root-Degree Distance

exp 29 (mean accuracy: 0.8308)

	0	1	2	3	4	5	6
same scale	-2.9213	0.7873	-1.3205	-0.9826	-0.3925	-0.6543	0.2823
different scale	1.9547	1.2910	0.7344	0.8642	0.8754	-0.9530	0.4349

Table C.29: DE 6.1 Symmetric Key Distance

	0	1	2	3	4	5	6
I	-0.5894	-0.1715	-0.5039	-0.7799	-0.6642	-0.6184	-1.2058
II	-0.4217	0.8921	0.2129	0.8705	0.7474	-0.0335	0.7388
III	0.4562	1.8898	0.5690	0.4528	1.1606	1.0905	0.4266
IV	-0.6044	1.0036	0.3485	0.6186	-0.3626	-0.0723	3.3166
V	-2.9783	-2.2100	-0.8053	-1.4102	-2.0756	-0.5896	-2.5831
VI	0.1019	1.6858	0.4895	1.5451	1.2057	0.5585	0.0056
VII	-1.4729	0.3084	0.1233	0.3992	-0.8177	-0.3257	0.0789

Table C.30: DE 7.1 Symmetric Root-Degree Distance