MASTER'S THESIS

Using Discovered and Annotated Patterns as Compression Method for determining Similarity between Folk Songs

August 24, 2015

AUTHOR:
D. P. Boot

SUPERVISORS:
dr. A. Volk
dr. W. B. de Haas

SECOND EXAMINER:
dr. F. Wiering

THESIS NUMBER:
ICA-3344770

Universiteit Utrecht

Abstract

Repetition is a fundamental concept in music. From previous research on the classification of folk songs, we learned that repeated patterns are important for the classification in tune classes. These classes are formed based on the similarity between the songs. In this thesis, we use annotated and automatically discovered patterns as compression method for determining the similarity between folk songs. When compressing songs using repeated patterns, we expect that only characteristic information about the classes is preserved in each song and therefore the classification accuracy will be high. We analyze if this method can be used to evaluate Pattern Discovery algorithms.

We discuss a number of state-of-the-art Pattern Discovery algorithms and different measures to determine the similarity between both patterns and entire songs. We propose an experimentation framework that uses these algorithms as compression method in a classification task, to measure the coverage (ratio of notes that are remained after the compression) and the classification accuracy. The framework consists of a compression step that reconstructs a song using the patterns that are discovered and a sequence alignment step for calculating melodic similarity. The experiments are performed on MTC-ANN, a set of Dutch folk songs, and a set of Irish folk songs.

Our results show that classification of uncompressed songs already result in a high classification accuracy. Compression using the Annotated Motifs from MTC-ANN lead to a low coverage and high accuracy. None of the Pattern Discovery algorithms achieve a result comparable to the Annotated Motifs, while some naive compression approaches even outperform the Pattern Discovery algorithms.

This leads to the conclusion that annotated motifs can successfully be used for compression, but that the automatic discovery of these patterns does not lead to satisfactory results. On top of that, we cannot conclude which of the Pattern Discovery algorithms performs “best”. However, our framework leaves open specific choices one might want to make in a specific context for comparing the Pattern Discovery algorithms unambiguously. Depending on the context, the desired outcome of coverage and classification accuracy can differ. Furthermore, our method can be used to make suggestions for improvement of the output of Pattern Discovery algorithms.

Contents

1. Introduction **1**
1.1. Music Information Retrieval . . . 1
1.2. Repeated Patterns in Music . . . 3
1.3. The Discovery of Repeated Patterns . . . 4
1.4. Using Repeated Patterns for Folk Song Classification . . . 5
1.5. Research Goal . . . 5
1.6. Thesis Structure . . . 6

2. Musical Background **7**
2.1. Music representations in MIR . . . 7
2.2. Music Terminology . . . 8
2.3. Symbolic Music Representation structures . . . 11
2.4. Conclusion . . . 13

3. Pattern Discovery in Music **15**
3.1. Sequence based Pattern Discovery Algorithms . . . 16
3.2. Geometry Based Pattern Discovery algorithms . . . 22
3.3. Conclusion . . . 27

4. Similarity of Patterns and Songs **29**
4.1. Evaluation of Pattern Discovery Algorithms . . . 29
4.2. Melodic Similarity using Sequence Alignment . . . 35
4.3. Conclusion . . . 39

5. Experiment Framework **41**
5.1. Dataset . . . 42
5.2. Pattern Discovery . . . 44
5.3. Melody Reconstruction . . . 45
5.4. Similarity Calculation . . . 47
5.5. Classification . . . 47

6. Method **49**
6.1. Determining Similarity on Uncompressed Songs . . . 49
6.2. Determining Similarity using Naive Lossy Compression Strategies . . . 50
6.3. Determining Similarity using Annotated Motifs . . . 51
6.4. Determining Similarity using Pattern Discovery Algorithms . . . 51
6.5. Comparison of Reconstruction Methods . . . 54
6.6. Comparison of Similarity Calculation methods . . . 54
6.7. Comparison of the Folk Song Datasets . . . 55

6.8. Comparing the Pattern Discovery Output with the Annotated Motifs 55

7. Results 57
7.1. Determining Similarity on Uncompressed Songs 57
7.2. Determining Similarity using Naive Lossy Compression Strategies 58
7.3. Determining Similarity using Annotated Motifs 60
7.4. Determining Similarity using Pattern Discovery Algorithms 61
7.5. Comparison of Reconstruction Methods 63
7.6. Comparison of Similarity Calculation Methods 66
7.7. Comparison of the Folk Song Datasets 67
7.8. Comparing the Pattern Discovery Output with the Annotated Motifs 70

8. Discussion 75
8.1. The Influence of Compression on Folk Song Classification 75
8.2. Using Annotated and Discovered Patterns for Compression 76
8.3. Compression and Accuracy as Evaluation Measure 80

9. Conclusion 83

Acknowledgements 87

A. Dataset Contents 89
A.1. MTC-ANN 89
A.2. Irish Folk Tunes Dataset 91

B. Parameter configurations 93
B.1. Gapless Reconstruction, Note to Note Alignment 94
B.2. Gapless Reconstruction, Melodic Shape Alignment 98
B.3. First Pattern Occurrence Reconstruction, Note to Note Alignment 102
B.4. First Pattern Occurrence Reconstruction, Melodic Shape Alignment 106

C. Enlarged Graphs 111

D. Raw Results 119
D.1. Selecting a Consecutive Subsequence 119
D.2. Random Note Selection 122
D.3. Random Pattern Selection 128
D.4. Song Pattern Discovery on MTC-ANN 136
D.5. Class Pattern Discovery on MTC-ANN 147
D.6. Song Pattern Discovery on the Irish Folk Tunes Dataset 157
D.7. Class Pattern Discovery on the Irish Folk Tunes Dataset 168

Bibliography 177

Introduction 1

Music is everywhere around us. With the growth of the Internet in the past years, large music collections have been made publicly available for everyone. This makes it possible to listen to our favorite type of music everywhere and anytime. Structuring this data in a way that is searchable can be a challenging task. We often do not only want to search for songs by a particular artist, but also listen to songs that are *similar* to the ones we like. In this thesis, we will study if we can determine how similar songs are by using an important property of many musical styles: *repetition*. We will use algorithms that find these repetitions automatically, both repeating within songs and repetitions that occur in multiple songs. We propose a method to use these repetitions as a compression method and classify the compressed songs into tune classes. This way, we can study the importance of repetition in the determination of similarity between songs.

More specifically, we will focus on the determination of similarity between *folk songs*. Folk music is *"the product of a musical tradition that has been evolved through the process of oral transmission"* (Lloyd, 1967, p. 15). As a product of this oral transmission, many variations of the same song will arise. This makes folk music an interesting research topic. For example, in the field of ethnomethodology folks songs are being classified to study their (cultural) origin. In our study, we will use two different folk song datasets to classify these songs based on their "family". Both the classification and the automatic discovery of repetitions with the aid of computers can be summarized in the research field of *Music Information Retrieval*.

In section 1.1, the research field of Music Information Retrieval is introduced. Then, in section 1.2, the definition and importance of repeated patterns in music is discussed. In section 1.3, we explain in short how repeated patterns can be discovered, either by human or using an algorithm. We also state a number of challenges that Pattern Discovery algorithms have to face. We then discuss how repeated patterns are used in other research to classify folk songs in section 1.4. All of this leads to the definition of the research goal in section 1.5. Finally, an outline of the remainder of this thesis is given in section 1.6.

1.1. Music Information Retrieval

Nowadays, our main source of information in general is only a few clicks away on the Internet. To make the search for the right information easier, we use a search engine such as Google or Yahoo to get pointers to the information we are looking for without getting lost in the enormous amount of information the Internet has to offer. In Computer Science, we call this process *Information Retrieval*. To cite Manning et al. (2009), Information Retrieval can be defined as *"finding material of an unstructured nature that satisfies an information need from within large collections"*. Suppose we want to know the original artist of "Yellow Submarine" (the information need). We can use a search engine to search the World

```
From: XXXXXXXXX
Subject: Early 80's - Please identify this song! (it's *very* difficult, though)
Newsgroups: alt.music.lyrics
Date: 2000-12-14 09:42:24 PST
Hi,
thiis is so difficult because I only remember those damn FRAGMENTS of it, which can
(in combination with possible errors) make it VERY difficult to identify this song!

But I'll try my best to make myself clear as possible. This song MUST be from the
period 1979-1984, most likely 1981 or 1982.
Tempo: about 120 bpm
Sounds VERY close to a SAGA or Asia tune (maybe it is SAGA even! ;)
OK here I go...(gonna add the chords for you guitarists out there ;)
[verse 1]
F C Bb Bb C
Crazy ............... onto the ..... café
 F C Bb
I'm drinking coffee, she came away
 F C Bb Bb C
She ordered ............. precious sum of money
???
F C Bb
deedeedeedeedeedeedeedee....
 C
Ohohohoo
[(instrumental) F C Bb Bb C F C Bb]

(...)

Hope that's enough to identify this tune ... :)
```

Figure 1.1.: Example of a musical query that is hard do retrieve for a textual retrieval system. Example from (Cunningham, 2002).

Wide Web (the large collection) for web pages (material in an unstructured way) that contain the words in our query (and hopefully find the answer we are looking for[1]).

While this query concerns a musical information need, the retrieval system does not necessarily need *musical knowledge*: searching for text (*meta-data*) that matches our query is enough. But suppose now we have a question that is similar to the query presented in figure 1.1. We could try to use some words from the lyrics as search query, but regular (textual) search engines will not see the difference between the lyrics of a song or a story about a crazy night at the café with a precious sum of money. Instead, we would like to use an information retrieval system where we could supply actual musical knowledge about the song we are looking for. In our example, we could use the chords as a query.

One of the challenges here is that music is much more complex than text. We can look at music at different levels: as one entity, as various instruments playing together, as just the melody, etc. This means we often have to *extract* the right information from a musical piece in order to make music collections searchable. The field of research where such problems are studied is *Musical Information Retrieval* (MIR). To cite Downie (2004), MIR is *"a multidisciplinary research endeavor that strives to develop innovative content-based searching schemes, novel interfaces, and evolving networked delivery mechanisms in an effort to make the world's vast store of music accessible to all"*. In other words, MIR is the research field that is focused on retrieving music in a content-based manner. With content-based, we mean that we use information from *inside* the music (melody, instruments, rhythm, etc.) rather than meta-data (artist, genre, year of recording, etc.).

MIR is a very active research field. Since 2000, there is a yearly conference on MIR, organized by the International Society for Music Information Retrieval[2] (ISMIR). Besides that, there is a platform for the evaluation of open MIR challenges, the Music Information Retrieval Evaluation eXchange[3] (MIREX).

[1]The answer is The Beatles, of course.
[2]http://www.ismir.net/
[3]http://www.music-ir.org/mirex/

Examples of such problems are query by humming, music segmentation, composer classification, cover song detection and the discovery of repeated themes and sections. Each year, people can participate in one or more tasks by submitting their algorithm. At the end, for each task the participating algorithms will be ranked based on how good they performed on the specific task.

One of the central concepts in MIR is the problem of musical *similarity*. The general challenge is to determine when two (or more) musical pieces can be called similar, while they can differ in for example melodic and rhythmic variations. On top of that, the perception of hearing "something similar" heavily depends on the experience of the listener. Downie et al. (2009) therefore concludes that *"the 'similarity problem' remains a huge challenge, not least because of the difficulty of establishing 'ground-truth' in this subjective area."* Here, *ground truth* means a set of objective similarity assignments that can be used to evaluate the performance of similarity methods. One can imagine that it is very hard to come up with such a ground truth if the determination of similarity is a very subjective process. In this thesis, we want to classify folk songs based on their similarity. Because musicologists have argued that repetition is very important for determining similarity (Volk and Van Kranenburg, 2012), we will use *repeated patterns* to measure the similarity between folk songs.

1.2. Repeated Patterns in Music

From the field of musicology, we know that repetition is a fundamental concept in music. Lerdahl and Jackendorff (1985) stated that *"the importance of parallelism in musical structure cannot be overestimated. The more parallelism one can detect, the more internally coherent an analysis becomes, and the less independent information must be processed and retained in hearing or remembering a piece."* In other words, repetition (what they call "parallelism" (Mirka, 2009)) structures the music for the listener and helps in remembering the piece. Schenker (1954, p. 5) stressed the importance of repetition by saying that repetition *"is the basis of an art."* To finish, Jones (1974) said that repeated patterns are *"one of the basic composing rules"*. But also in other research areas, repetition in music is an interesting topic. In Music Psychology, experiments have shown that listeners were emotionally triggered when hearing a repetition (Livingstone et al., 2012). And in folk song Music Research, it is shown that repeated motifs can be used to identify parts of songs that remain intact over time through oral transmission (Volk and Van Kranenburg, 2012).

By definition, a repetition is something that repeats at least once. In the context of music, this repetition could be very long (such as the melody of a verse and chorus) or much shorter like a theme or motif (whereas motifs are considered shorter than themes). We will define such repetition as a repeated *pattern*: a (short) part of song with a minimal length of two notes that repeats at least once, in either the same song or in different songs. This means a pattern must have at least two occurrences. In this thesis, we will mainly focus on patterns at the size of motifs, hence we will use the terms *patterns* and *motifs* interchangeably. An example of repetition in a (folk) song is shown in figure 1.2.

Supported by the fact that repeated patterns are considered important for the structure of music, we suspect that repeated motifs can be used to describe important characteristics about the song the pattern occurs in. On top of that, we could even use patterns that occur in multiple songs to describe the similarity between these songs.

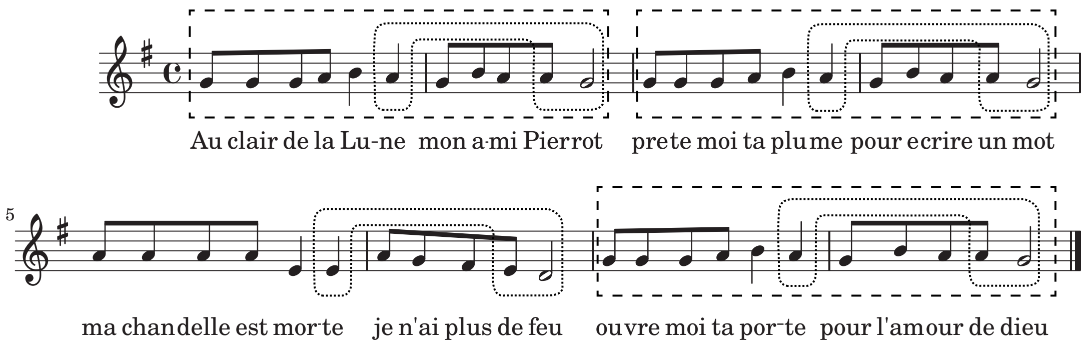

Au clair de la Lu-ne mon a-mi Pierrot prete moi ta plume pour ecrire un mot
ma chandelle est mor-te je n'ai plus de feu ouvre moi ta por-te pour l'amour de dieu

Figure 1.2.: Two repetitions in the French folk song *Au claire de la lune*. The repeated segment marked in the dashed box is easy to notice. The second pattern, bound by the dotted rectangle, is much more difficult to notice.

1.3. The Discovery of Repeated Patterns

There are two ways of finding repeated patterns in a musical piece. The first method is asking domain experts (musicologists) to manually annotate repeated motifs in a musical piece. Obviously, this is a very time consuming procedure and the results depend heavily on the experience of the annotator. This also means that these results are subjective; another listener or annotator (whether an expert or not) could hear totally different repetitions in a song.

The second way of discovering repeated patterns is letting a computer do it *automatically*, using a Pattern Discovery algorithm. These algorithms output all patterns they have discovered, given a set of (digital) songs as input. These songs could be either audio recordings or more descriptive (symbolic) music, like sheet music. Compared to the manual annotation method, the use of a computer for this process will increase the speed of discovery drastically. However, there are a number of challenges the current state-of-the-art Pattern Discovery algorithms have to overcome:

Output size. Compared with patterns annotated by human, most Pattern Discovery algorithms will return a significantly larger number of repeated motifs (Cambouropoulos, 1998). Presumably most of these repeated patterns are not particularly important (or characteristic) and could be discarded in the output. The problem here is that it is hard for an algorithm to determine how *salient* a discovered pattern is.

Determination of importance. In order to limit the output size of a Pattern Discovery algorithm, one has to determine the *importance* of a discovered pattern in order to accept or reject the motif in the resulting output. With importance, we mean how noticeable a pattern is for a listener to recognize.

Choice of musical features. Music itself is very complex. Besides the fact that everyone perceives music differently, we also need to choose which musical features we want to use in order to discover repetitions. Here, we mean specific information about melody, rhythm, sound, etc. The more features an algorithm uses for pattern discovery, the more complex the problem becomes.

Evaluation. In order to evaluate the *performance* of a Pattern Discovery algorithm, the discovered patterns are often compared to a *ground truth*, a set of patterns discovered by human which are assumed as important to the musical piece it occurs in. There is a MIREX task about Pattern Discovery, which uses this kind of evaluation method. However, this method of evaluation is not without challenges and problems. First, developing a ground truth set of patterns is a very time consuming task since this is a manual process. Second, like we stated at the beginning of this section, annotating patterns is grounded on subjectivity: since everyone perceives music differently, different people may also detects different repetitions. Therefore, it is very hard to come up with a set of patterns that can be used as a *golden standard*. This means that the quality of the ground truth depends on the consistency of the discovered patterns between multiple annotators and on the musical background of the annotators: it is likely that trained musicians are able to discover more complex repetitions than untrained listeners. However, this does not mean that simple patterns are less perceptually relevant than more complex repetitions.

In the case of the ground truth that is used in the MIREX task, Meredith (2015) argued that it is questionable if an algorithm should be evaluated by the number of "important" patterns it finds, or just how well the resulting patterns correspond to the ground truth. In his words: *"there is some question as to whether we are strictly justified in claiming that a pattern is 'important' without specifying an objectively evaluable task that the pattern helps us to perform more successfully."*

In this study, we will primarily focus on the subjectivity problem of the pattern evaluation. Since repeated patterns play an important role for human annotators in the classification process of folk songs, we hypothesize that we can also *objectively evaluate* the quality of the patterns returned by a Pattern Discovery algorithm by the role they play in determining the similarity between the songs.

1.4. Using Repeated Patterns for Folk Song Classification

Like we stated in the previous sections, we will primarily focus on the problem of *folk song classification* using repeated patterns. Related research regarding the classification of folk songs has shown that sequence alignment techniques yield high classification results when they are used as similarity measure between folk songs (Hillewaere et al., 2012; Van Kranenburg et al., 2009).

Not restricted to folk songs only, repeated patterns have been used in different studies. For example, Conklin (2009) used discovered repeated patterns as feature sets for (geographic) folk song classification. Lin et al. (2004) used the output of a Pattern Discovery algorithm that finds *Significant Repeated Patterns* as feature for the classification of music into seven different genres. Karydis et al. (2006) derived melodic and rhythmic features from repeated patterns, in order to classify classical music based on their genre. Meredith (2014) used Pattern Discovery as compression measure, where similarity between folk songs was calculated by the degree of compression in each song using the *Normalized Compression Distance* (Li et al., 2004).

1.5. Research Goal

As stated at the beginning of the introduction, we will use Pattern Discovery Algorithms as a compression method to determine the similarity between folk songs. In other words, the repeated patterns will be used as a *compression strategy*. Because these motifs are considered important and characteristic for a musical piece, we expect that after compression only important and characteristic parts of the songs are remained. The compressed songs are then used to classify folk songs based on their similarity, such that we can study the effect of compression on the classification accuracy. We expect that despite the compression, the selection of characteristic motifs yields a classification accuracy that is comparable with the classification accuracy on songs without compression. In other words, we expect that we can evaluate the quality of the discovered patterns by the resulting classification accuracy after compressing a song using the discovered repeated patterns. This way, we introduce a different way of evaluating Pattern Discovery algorithms as opposed to comparing the discovered patterns to an annotated ground truth (like the MIREX task we mentioned earlier).

The difference between our method and the approach taken by Meredith (2014) is that in his study, the similarity between pairs of songs is calculated solely on the degree of *lossless* compression (i.e. no information is lost, the original song can be reconstructed from the compressed song) between a pair of songs. In the method we propose, the discovered patterns are used for *lossy* compression (i.e. information is removed and therefore lost, the original song cannot be reconstructed from the compressed song) which potentially removes (perceptually) non-interesting parts of a song for classification.

The aim of this research is twofold. On one hand, we want to test whether repeated patterns can be used as compression method to determine the similarity between folk songs. For this, we will compare both patterns that are annotated by human experts and motifs that are automatically extracted by a Pattern Discovery algorithm. On the other hand, we want to know, driven by the first research question, if we can solve the problems with the evaluation of Pattern Discovery algorithms we mentioned earlier, by defining an evaluation measure based on the retrieval performance of musical similarity. This leads to the following research goal:

Research Goal
Study the performance of using Pattern Discovery Algorithms as compression method in determining similarity between folk songs.

We split this research goal into three research questions:

Research Question 1.
How is the accuracy of classification based on similarity influenced by compressing folk songs?

Research Question 2.
Is there a difference in classification accuracy between automatically discovered patterns and annotated patterns?

Research Question 3.
Are compression and classification accuracy a good measure for evaluating Pattern Discovery algorithms?

1.6. Thesis Structure

The remainder of this thesis is structured as follows: in chapter 2, we will discuss the basic music theory and terminology in order to understand the musical concepts we will use in the remaining of this study. Then, in chapter 3, the implementation of various pattern discovery methods is discussed. In chapter 4, we will explain two different similarity approaches: one for defining similarity between the discovered patterns and one for defining similarity between songs. Then we will discuss the experiment framework we developed for using Pattern Discovery algorithms as compression and similarity technique in chapter 5. An overview of the experiments we have performed based on this framework is given in chapter 6 and the results are presented in chapter 7. Last, we will discuss the results in chapter 8 and present our conclusion in chapter 9.

Musical Background 2

The research in this thesis is about music, and folk music in particular. In this chapter, we will make the (computer science) reader familiar with some basic concepts of music (research), as well as musical representations we will use in the rest of this thesis. In section 2.1, we will discuss two different music representations: the audio representation and the symbolic representation. Then, we introduce and explain the necessary musical theory and terminology in section 2.2. Using this knowledge, we discuss two different structures to represent symbolic music in section 2.3, that are also used by Pattern Discovery algorithms we discuss in the next chapter.

2.1. Music representations in MIR

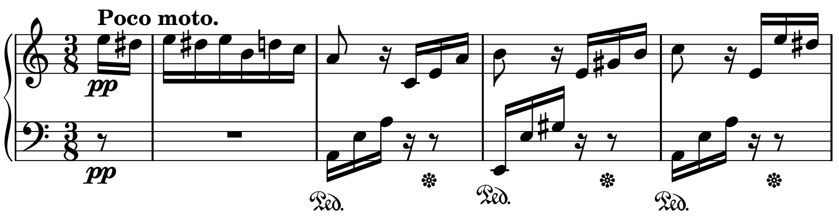
In MIR, we often need access to (a large collection of) digital music. Since the most essential part in music is the vibration of air reaching our ears, we need to create a *digital* representation of this music in order to process it digitally. We can roughly divide all representations into two different classes: audio and symbolic music.

The audio representation of a song is the most straightforward for the regular listener. A song is recorded once and digitalized in such a way that we can play it on our computer or MP3 player. Because the actual transition of *analog* sound waves to *digital* signals is outside the scope of this thesis, we refer the reader to (Pohlmann, 2010, chapter 2) for a background of the digitalization process of audio signals. This does not mean that all audio is already *music*. Music can be seen as an *interpretation* of audio by the human mind. For example, the sound of water drops falling onto a surface can be interpreted as regular noise, but when the drops are falling rhythmically and creating different sounds, one can interpret these sounds as *music*. In other words, the human mind plays an important role in the perception of audio as music.

The symbolic representation of music, as opposed to the audio representation, does not contain audio signal information. Instead, the music is represented by describing what *needs to be played*. A simple example to illustrate this difference: a composer writes down his composed music in a *symbolic* format (such as sheet music), that contains information about which notes have to be played by which instruments. When we listen to a performance of this musical piece performed by an orchestra, we hear the same song but now in the audio representation. This can also be seen in figure 2.1. It is important to notice that we need a *performer* to transforms symbolic music into something we can listen to, while the audio representation already contains the sound. This means that the actual sounds that follow from the symbolic representation are dependent on the performer and the (type of) instrument that is used.

In the light of MIR, the differences between these representations are also important. Because of the cognitive component of perceiving audio as music, one can hear parts of a song that are not actually played by a single instrument, but are the product of multiple instruments playing. This is something

(a) Audio representation: the audio signal



(b) Symbolic reprentation: the score. It contains information about the notes that have to be played, the tempo/style (*poco moto*) and the dynamics/loudness (*pp*, *pianissimo*). It even tells the performer when to hold and release the sustain pedal by the *Ped* and * symbols.

Figure 2.1.: First four measures of Bagatelle No. 25 in A minor ("Für Elise") by Ludwig van Beethoven, both in audio (a) and symbolic form (b).

that cannot be detected easily in the symbolic format of music, since each part (instrument) is described differently. The other way around can also happen: an instrument with a very subtle contribution cannot be distinguished in the audio representation, where the symbolic representation contains information about every note that has to be played despite the contribution to the entire musical piece. The differences between notated and perceived music makes MIR a very challenging research area. One of the challenges is that in the audio representation of music, we cannot easily distinguish parts or *elements* from the song, such as the different instruments or voices. By training and experience, humans can distinguish the sound of a trumpet from a piano, but this is not a straightforward process in MIR. In this thesis, we use the *symbolic* representation of music as input for Pattern Discovery.

The most common format for digital symbolic music is the MIDI (Musical Instrument Digital Interface) standard. MIDI consists of messages (*events*) that contain information about the note that has to be played (the MIDI note number), the velocity of the note, if the key is pressed or released, etc. The strength of MIDI is that it can be played by a digital (MIDI-compatible) instrument, which transforms the symbolic music into audio. Of course, there are many more digital symbolic formats. Besides MIDI, we use music in the ABC notation and in the Humdrum `**kern` data format (Huron, 1997) in this study. Another notable format is MusicXML (Good, 2001), which is an open XML standard for digital sheet music. For a thorough overview of digital music representations, we refer the reader to (Selfridge-Field, 1997).

2.2. Music Terminology

In this section, we explain the basic terminology used in this thesis in order to understand the methodology we use further on. We primarily focus on *symbolic* music. In musical notation, one of the most fundamental elements is a *note*. In general, a note has a *pitch* and a *duration*.

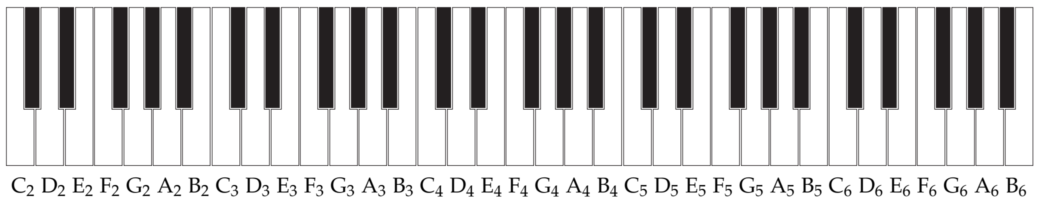

C2 D2 E2 F2 G2 A2 B2 C3 D3 E3 F3 G3 A3 B3 C4 D4 E4 F4 G4 A4 B4 C5 D5 E5 F5 G5 A5 B5 C6 D6 E6 F6 G6 A6 B6

Figure 2.2.: Part of the piano keyboard. A regular piano keyboard has 88 keys, from A_0 till C_8. For the white keys, the pitch in scientific pitch notation is given below the keyboard.

2.2.1. Notes, Pitch and Scales

In general, the *pitch* of a note can be expressed by the frequency of the resulting tone or by its name. In this thesis, we will use the *Scientific Pitch Notation* (Young, 1939) for representing notes. This notation contains of two parts: a letter (A–G) and a number. The scientific pitch notation of all the white keys on a piano keyboard are shown in figure 2.2. We know from physics that when we double the frequency of a tone, the resulting tone is *perceived as similar to* the original tone. For example, the note A_4 is defined in Western music to have a frequency of 440 Hz. If we double this frequency, we get the note A_5 with a frequency of 880 Hz. We call the distance between these two notes an *octave*. We also say that these notes belong to the same *pitch class* A.

In Western music, we divide an octave into 12 *semitones*. These semitones can be illustrated by a piano keyboard: the 12 semitones correspond to both the white and the black keys. We already know the names of 7 of these semitones, namely A till G. The other five semitones are represented on the piano keyboard as the black keys. We notate these notes as follows: $C\sharp_4$ represents the note one semitone higher than C_4. On the piano keyboard, this is the key between C_4 and D_4. We can also describe this note as one semitone *lower* than D_4: $D\flat_4$. The sharp symbol ($\sharp$) means "one semitone higher" and the flat symbol ($\flat$) means "one semitone lower". The division of an octave into 12 semitones is called the *chromatic scale* and is visualized in figure 2.3. Because every octave is divided the same way, this also means we have 12 pitch classes.

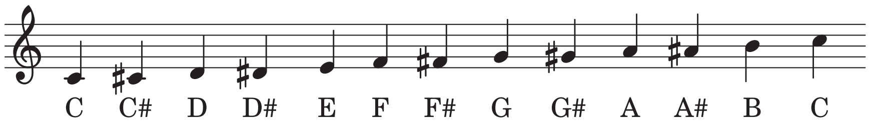


Figure 2.3.: The chromatic scale, divided into 12 pitch classes.

Another notation method we use for describing pitches is the *MIDI note number* (not surprisingly used in the MIDI format). Here, each note is represented by an integer value. For the next semitone, the MIDI note number is increased by 1. For example, the MIDI note number of the note A_4 is 69, $A\sharp_4$ is 70 (which has the same number as $B\flat_4$, see the explanation above).

We can also represent pitches in the *diatonic scale*. Unlike the chromatic scale, an octave is now divided into 7 instead of 12 steps: the number of Latin characters we use for the note notation. Back to the piano keyboard, the diatonic scale consists of all white keys of the keyboard. We call a pitch value in the diatonic scale *morphetic pitch* (Meredith et al., 2002). The morphetic pitch can also be represented by the *staff height* (the height of the note head on the staff), compare figure 2.3 and 2.4. Keeping in mind the staff height, it is immediately clear from both figures that in the diatonic scale, $C\sharp$, $C\flat$ and $C\natural$ (natural, without increase or decrease) share the same morphetic pitch value.

Like the chromatic pitch, we can also denote the morphetic pitch by a number. We follow the convention of Collins et al. (2011), where the morphetic pitch number of C_4 is 60 (the same as the MIDI note number

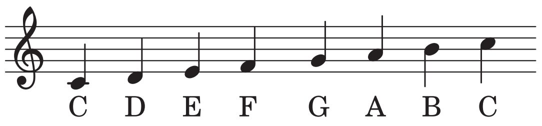


Figure 2.4.: The diatonic scale, divided into 7 pitch classes

of C_4). The morphetic pitch number of D_4 then becomes 61.

2.2.2. Pitch Intervals

The *distance* between two pitches is called a *pitch interval*. Since we can represent each note as a number (both the MIDI note number or the morphetic pitch number), we can define the pitch interval as the difference in pitch number between these two notes. For example, the chromatic pitch interval between the notes D_4 and B_4 is 9 (this can be checked in figure 2.3) whereas the diatonic pitch interval between these pitches is 5. Since in the diatonic scale sharps and flats play no role, the morphetic pitch interval between $D\sharp_4$ and $B\flat_4$ is still 5. The pitch interval can also be negative, for example between the notes F_5 and G_4 (which is a chromatic pitch interval of -10 and a morphetic pitch interval of -6).

We can do the same for pitch *classes*. Recall that all notes with the same name are grouped in the same pitch class. For pitch classes, we use the same system. We can number the 12 pitch classes C, C♯, D, . . . , A♯, B as 0, 1, 2, . . . , 11. For the morphetic pitch class, the numbers go from 0 to 6. This way, we can also calculate the pitch class intervals by subtracting the pitch class numbers of both notes.

2.2.3. Duration and Meter

Like we stated at the beginning of this chapter, two important properties of a note are pitch and *duration*. The unit of duration (or note length) is defined as a number of *beats*. For example, a note with a length of two beats has a duration exactly two times as long as a note with a length of one beat. This is illustrated in figure 2.5. The length of a half note is half the length of a whole note, the duration of a quarter notes equals half the length of a half note, etc. We can increase the duration of a note by half its original length by adding a dot after the note shape. For example, a half note with a dot has a duration of three beats. This way, any note length can be expressed by the notation used in figure 2.5.

Each song has a certain rhythmic structure: some notes are more accented than others. This is defined by the *meter* of a song. This structure is created by dividing a musical piece into *measures* with an equal number of beats. Each first beat in a measure is accented. These accents can be illustrated with a simple example. Suppose there are four beats in each measure and we count aloud from one to four in each measure. When we do this multiple times, we will hear that the "one" is more accented than the other numbers. The actual number of beats in each bar is defined by the *time signature*, that can be found at the beginning of the staff. In the excerpt presented in figure 2.5, the time signature is $\frac{4}{4}$. The upper 4 means that there are four beats in each measure. The bottom 4 represents the note type that has to be counted for one beat, in this case the quarter note. That means that in this case, four quarter notes will fill up one measure. This is also illustrated in the figure, where each measure is bounded by vertical lines.

The *onset* (or *ontime*) of a note is its position in the musical piece, measured from beginning of the song and expressed in beats. In this thesis, we choose the onset of the first note to be zero. In the excerpt in figure 2.5, the onset of the first note is therefore 0, the onset of the second note is 4 and the onset of the ninth note (the second 8th note) is 12.5.

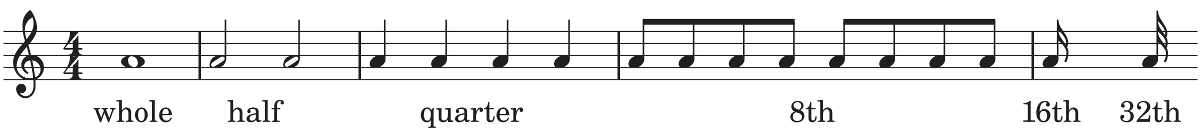


Figure 2.5.: Different note durations with their notation and name. A half note is exactly half the length of a whole note, a quarter note half the length of a half note, etc. Because the number of beats in each measure is fixed (four in this case), four quarter notes are needed to fill a measure whereas a single whole note also fills a measure (since a whole note is four times as long as a quarter note).

2.2.4. Monophonic and Polyphonic Music

Music can be *monophonic* or *polyphonic*. The main difference between these types is if there sounds (from the Ancient Greek word *phoní*) only one note at a time (*mono*) or multiple notes (*poly*). To illustrate this difference, the song in figure 1.2 is monophonic, since all notes are played sequentially. The excerpt of *Für Elise* in figure 2.1b is polyphonic, since at the first beat of the second, third and fourth bar two notes need to be played at the same time. Some instruments can only produce monophonic music, such as our own voice, a flute or a trumpet. Other instruments can play both polyphonic and monophonic, based on the musical piece. Examples of such instruments are a piano or a guitar.

2.3. Symbolic Music Representation structures

The Pattern Discovery algorithms we use in this study will find patterns in *symbolic* music. In this section, we will discuss two different representation structures that can be used as internal music representation for these algorithms: sequential structures and geometric structures.

2.3.1. Sequential structure

In a sequential (string based) structure, the melody is treated as a sequence of notes. For example, the first measure of Beethoven's *Für Elise* (figure 2.1) could be represented as

$$[\mathrm{E}_5, \mathrm{D}\sharp_5, \mathrm{E}_5, \mathrm{D}\sharp_5, \mathrm{E}_5, \mathrm{B}_4, \mathrm{D}_5, \mathrm{C}_5]$$

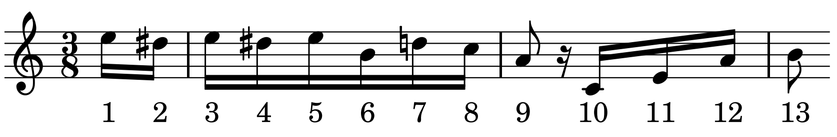
or by MIDI note number:

$$[76, 75, 76, 75, 76, 71, 74, 72].$$

An advantage of this structure is the correspondence with textual strings. In the light of Pattern Discovery, normal text mining algorithms can easily be used on this kind of structure (see (Meredith et al., 2002, section 3) for a survey of string-based Pattern Discovery algorithms). Also algorithms used in bio-informatics to find sequences in DNA strings can be used for this purpose.

Due to the linear, sequential character of this structure, the amount of information that can be represented in a single string is limited. In the above example, it is not possible to retrieve the note lengths. It is also impossible in this approach to support polyphony: multiple notes that sound at the same time.

This drawback shows the complexity of music compared to text or DNA strings. Music is more than a sequence of pitches, much more factors are of interest. To model the multidimensionality of music, Conklin and Witten (1995) introduced the concept of viewpoints. A *viewpoint* models a specific musical feature to a value. It can be seen as a function that maps a musical event (i.e. a note or a rest) to a



	1	2	3	4	5	6	7	8	9	10	11	12	13
pitch	76	75	76	75	76	71	74	72	69	60	64	69	71
ontime	−2	−1	0	1	2	3	4	5	6	9	10	11	12
duration	1	1	1	1	1	1	1	1	2	1	1	1	2
interval	⊥	−1	1	−1	1	−5	3	−2	−3	−9	4	5	2
pitch cl.	3	2	3	2	3	11	2	0	9	0	4	9	11
contour	⊥	−1	1	−1	1	−1	1	−1	−1	−1	1	1	1
fib	F	F	T	F	F	F	F	F	T	F	F	F	T

Figure 2.6.: Table of viewpoint sequences as defined by Conklin and Witten (1995). The viewpoints in the upper half are *primitive* viewpoints. The viewpoints in the bottom part are the *derived viewpoints* pitch interval, pitch class, contour and “first in bar”.

value in a specific context (e.g. pitch or contour). A sequence of notes can be represented by a *viewpoint sequence*, as can be seen in figure 2.6.

We distinguish *primitive* and *derived viewpoints*. Primitive viewpoints can be defined directly using the symbolic representation of music, like pitch and duration. Using these primitive viewpoints, we can derive new viewpoints. Examples of such *derived viewpoints* (Collins, 2011) are pitch interval and melodic contour. These derived viewpoints can in their turn be combined with other (either derived or primitive viewpoints) to create new viewpoints.

For example, the pitch viewpoint of the first note in figure 2.6 equals 76, which is the MIDI note number. Using the pitch viewpoint sequence, we can derive the pitch interval viewpoint which equals the difference between two successive notes in the sequence. For the first note, this value is undefined. Although not straightforward, viewpoints can also be used for polyphonic Pattern Discovery (Bergeron and Conklin, 2011).

2.3.2. Geometric structure

To overcome the limitations of sequential structured, Meredith et al. (2002) proposed a geometry based structure for Pattern Discovery, where notes are represented as multidimensional vectors. Compared with the viewpoint structure of Conklin and Witten (1995), each dimension can be seen as a different viewpoint. For example, we could represent the first notes of *Für Elise* as the set of 3-dimensional vectors

$$D = [\langle -2, 76, 1\rangle, \langle -1, 75, 1\rangle, \langle 0, 76, 1\rangle, \langle 1, 75, 1\rangle, \ldots]$$

Where the dimensions represent the onset time, MIDI note number and duration (in 16th notes). To reduce the complexity, we can project D to a 2-dimensional space (see figure 2.7).

One of the problems described in section 1.3 of the introduction was the *multidimensionality* of music. Music can be described by the melody or rhythm, but many other, more subtle dimensions can play an important role in the perception of a song.

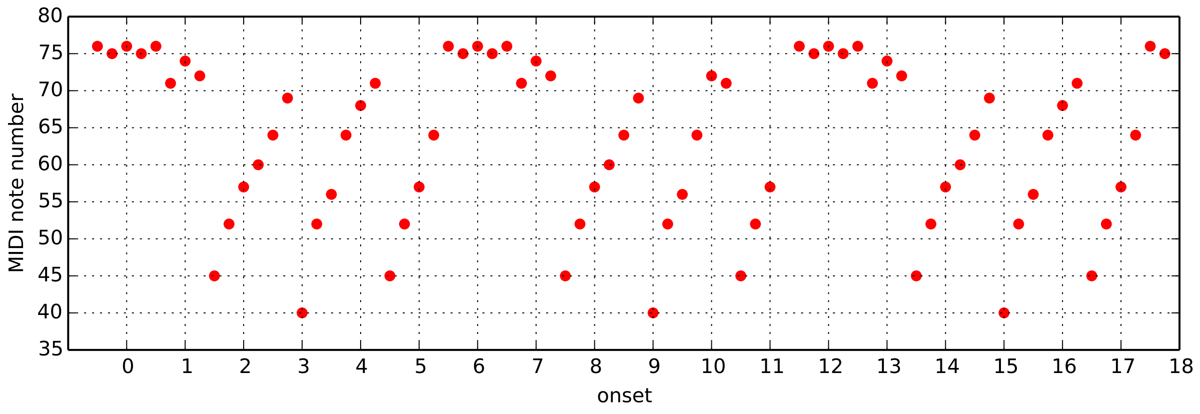


Figure 2.7.: Geometric representation of the first 4 measures of *Für Elise*. The notes are projected onto a 2D space (onset, MIDI note number).

2.4. Conclusion

In this chapter, we have seen that music can be represented in roughly two different ways: by the actual sound (the audio) or in a more descriptive manner (the symbolic representation). In this project, we will use the symbolic representation of music as input for the Pattern Discovery algorithms. This representation contains all information about which notes are played how long and at which moment in time. Also in the symbolic representation, there are many notations and formats to describe the same piece of music.

The Pattern Discovery algorithms we use in this study support the symbolic music representation as input. The underlying structures for storing this type of music can be divided into two categories: sequential structures and geometric structures. The sequential structure is based on the concept of modeling music as a sequence of notes. A disadvantage of this method is that the amount of information that can be represented by a single sequence is limited. To overcome this problem, *viewpoints* are introduced. The geometric structure models a musical piece as notes as vectors in a multidimensional space. This overcomes the problem of limited information, since dimensions can be added to each vector. In the next chapter, we will see how the different Pattern Discovery algorithms will use these underlying structures for finding repeated patterns in music.

Pattern Discovery in Music 3

The goal of a Pattern Discovery algorithm in the context of MIR is to extract repeated patterns from a musical piece. In this chapter, we will discuss different Pattern Discovery algorithms that we use to extract repeated patterns from the folk songs. Using these patterns, we will compress the songs at a later stage. For a recent overview of Pattern Discovery algorithms, we refer the reader to Janssen et al. (2013).

There are many Pattern Discovery algorithms, all of them having the same purpose. However, the underlying technique or the characteristics of the input and output differs. We therefore characterize the different algorithms by five properties:

Musical texture: monophonic versus polyphonic music. In section 2.2.4, the differences between monophonic and polyphonic music are discussed. Especially if the Pattern Discovery algorithm uses a sequential music representation, it can be challenging allowing simultaneous events in a sequential structure.

Search domain: Song versus Class Pattern Discovery. When looking for patterns, the question is if we have to look at patterns recurring in a single musical piece or looking for patterns that repeat in multiple pieces. For *Song* pattern discovery, we want to find repetitions inside a single musical piece, while for *Class* pattern discovery we want to discover patterns that exist in multiple songs from the same class. In literature, these different search domains are also called *intra-opus* for pattern discovery within songs and *inter-opus* for pattern discovery within a collection of songs. Because these terms are very similar and could be therefore confusing, we will use the terms *Song Pattern Discovery* and *Class Pattern Discovery* in the rest of this thesis.

Similarity: exact versus inexact pattern discovery. With exact pattern discovery, we want to discover exact repetitions of a pattern. Exactness does not mean that it has to be exact in *all* dimensions. Most algorithms tend to find *transposition*-invariant repetitions, which means that a repetition a few semitones higher is considered as an exact match. For inexact pattern discovery, it is allowed to have rhythmic/melodic variations.

Data structure: sequential versus geometric Like we have seen in section 2.1, we can represent a song as a sequential structure or as a geometric structure. Both have their advantages and disadvantages for pattern discovery. Geometric structures are the better choice for polyphonic music, while string based methods can benefit from sequential pattern mining solutions from other domains.

Musical representation: audio versus symbolic music Pattern discovery can be done on both audio and symbolic music, see section 2.1. We will only perform Pattern Discovery on symbolic music representations.

In the next sections, we will explain the workings of the algorithms we use for comparison in our experiment. In section 3.1, three sequential based Pattern Discovery algorithms will be discussed. Then, in section 3.2 five different geometry based Pattern Discovery algorithms are discussed.

3.1. Sequence based Pattern Discovery Algorithms

3.1.1. MGDP Algorithm

The *Maximally General Distinctive Pattern* (MGDP) algorithm (Conklin, 2010) is a monophonic Pattern Discovery algorithm that tends to discover patterns within a collection of songs (Class Pattern Discovery), based on the sequential viewpoint approach (see section 2.3.1). This means that internally, a musical piece is represented by a number of predefined viewpoints. Only exact repetitions are discovered. To limit the output size, only patterns are returned that are characteristic for the collection of songs (the *corpus*) it occurs in. To define if a pattern is characteristic for this specific corpus, each motif is being judged by the number of songs that are *not* part of the corpus, but nevertheless contain the pattern. This set of songs is called the *anti-corpus*. For example, to find distinctive patterns in a corpus of jazz pieces, we can use a set of classical music as anti-corpus. This way, patterns that are significantly represented in both corpora are discarded. Note that for each class that is assigned as corpus in the MGDP algorithm, the songs from all other classes in the dataset are used as anti-corpus.

More formally, for a pattern P to be discovered and returned by the MGDP algorithm, it has to fulfill the following conditions:

P **is *supportive*** A pattern that occurs twice in the whole corpus is presumed less important than a pattern that is present in many songs. We therefore define the relative *support* of a pattern P as the *empirical probability* of P in the corpus:

$$p(P|\oplus) = \frac{c^{\oplus}(P)}{n^{\oplus}} \tag{3.1}$$

where $c^{\oplus}(P)$ is the number of songs in the corpus that contain pattern P and $n^{\oplus}$ is the total number of songs in the corpus. In other words, $p(P|\oplus)$ is the ratio of songs in the corpus that contain P. In order for a pattern P to be frequent, $p(P|\oplus)$ has to be greater than the *support threshold*.

P **is *distinctive.*** In order for a pattern to be *distinctive*, we compare the empirical probability of a pattern P in the corpus with the empirical probability of P in the *anti-corpus*. Like equation 3.1, we can define $p(P|\ominus)$ as:

$$p(P|\ominus) = \frac{c^{\ominus}(P)}{n^{\ominus}} \tag{3.2}$$

where $c^{\ominus}(P)$ is the number of songs in the anti-corpus that contain pattern P and $n^{\ominus}$ is the total number of songs in the anti-corpus. Combining equation 3.1 and 3.2, the *interest* (degree of distinctiveness) is then defined as

$$I(P) = \frac{p(P|\oplus)}{p(P|\ominus)} = \frac{c^{\oplus}(P)}{p(P|\ominus) \cdot n^{\oplus}} \tag{3.3}$$

This value represents the number of times P occurs more in the corpus than in the anti-corpus. If the distinctiveness of P is larger than a predefined *distinctiveness threshold*, the pattern is accepted as being distinctive in the corpus.

P **is *maximally general.*** We call a pattern P more *general* than a pattern Q if all occurrences of Q are also occurrences of P. For example, in figure 3.1 the sequence $\{F_4, G_4, A_4\}$ is more general than $\{F_4, G_4, A_4, C_4\}$. The MGDP algorithm only returns motives that are *maximally* general, meaning that in the returned pattern set R, no member in R is more general than another member in R:

$$\nexists P, Q \in R \quad \textsc{isMoreGeneral}(P, Q) \tag{3.4}$$

In order to find patterns that comply with these requirements, the search space is structured as a sequential tree of patterns, with the empty pattern as root. In figure 3.2, a part of the search tree is illustrated. The tree is built and traversed in a depth-first manner, such that each child node pattern is an extension of the pattern in its parent node. This can be clearly seen in the figure: the child patterns of the top node are both extensions of the top node pattern. Based on equation 3.4, the subtree of a pattern node that is *supportive*, *distinctive* and *general* has not to be explored, since all of its children are less general than the node itself.

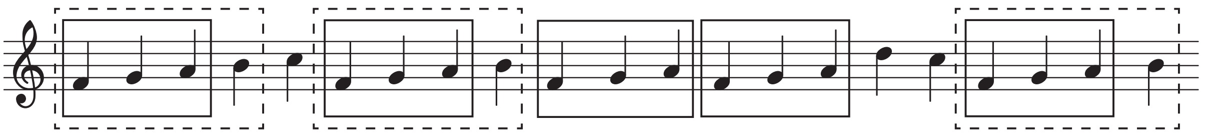


Figure 3.1.: The pattern $\{F_4, G_4, A_4\}$ is more general than $\{F_4, G_4, A_4, C_4\}$, since all occurrences of $\{F_4, G_4, A_4, C_4\}$ are also occurrences of $\{F_4, G_4, A_4\}$. It can be easily seen that this relation is not symmetric because two occurrences of the pattern $\{F_4, G_4, A_4\}$ are not occurrences of $\{F_4, G_4, A_4, C_4\}$. Suppose we allow patterns of length 2, then $\{F_4, G_4$ is even more general than $\{F_4, G_4, A_4\}$.

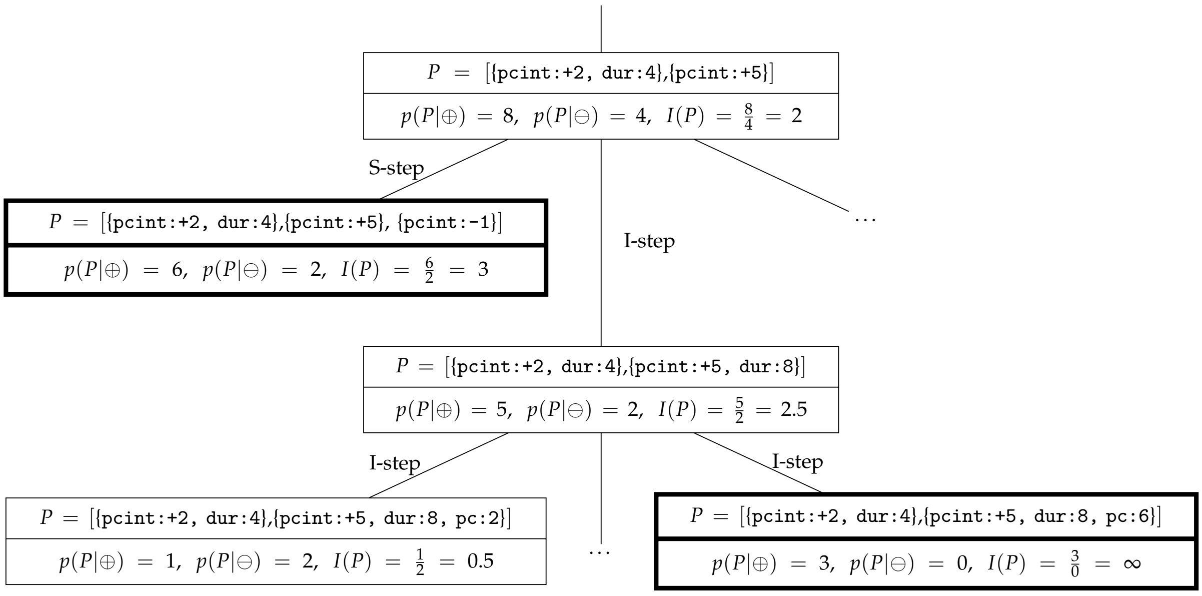
The traversal of the tree is based on the Sequential Pattern Mining algorithm (SPAM) by Ayres et al. (2002). The traversal starts in the root of the three, which is empty. Then, the pattern is extended by one of the viewpoints that are defined. For example, the empty pattern could be extended with `pcint:+2`. This means we are looking in every musical sequence for a pattern with length two of which the pitch class of the second note is two semitones higher than the first note. Due to the underlying bitmap representation of the musical pieces, finding this pattern in each song is a very efficient process. For this pattern, the support, distinctiveness and generality is being calculated. If the pattern is maximally general, the pattern is returned and the child nodes does not have to be explored further. If the pattern is not maximally general, we will extend the pattern and repeat the process.

In each iteration, the pattern of the current node can be extended in two different ways: by extending the pattern with a subsequent element (i.e. adding a note to the pattern) which we call an S-step or by extending the properties of the last element in the sequence, an I-step. Both steps are illustrated in figure 3.2. Suppose the algorithm currently extends the pattern at the top node, that represents the sequence [{`pcint:+2, dur:4`}, {`pcint:+5`}]: a quarter note which pitch class is 2 semitones higher than the previous note, followed by a note with a pitch class 5 semitones higher.

We can now extend this pattern by either an S-step or an I-step. In fact, both extensions are tried, but suppose we perform an S-step. A possible extension to the pattern is tried, for example [{`pcint:+2, dur:4`}, {`pcint:+5`}, {`pcint:-1`}]. This means that the pattern is extended with a note one semitone lower than the previous note. Since this pattern is both general and distinctive, the pattern is returned and the search path is closed. Because of this, the pattern is *maximally* general (and) distinctive. Back in the parent node, we can also perform an S-step. In the example, the second node in the pattern is made more specific by specifying the duration of the note: [{`pcint:+2, dur:4`}, {`pcint:+5, dur:8`}]. Since this pattern is not distinctive, the pattern is extended again with either S-steps or I-steps. The algorithm terminates when all maximally general distinctive patterns are found. This results in a tree where all the leaves are Maximally General and Distinctive Patterns.

Algorithm parameters

Symbol	Description
$p(P\|\ominus)$	The minimum relative support of a pattern in the corpus.
$I(P)$	The distinctiveness threshold.
n	The minimum number of notes in the pattern.



P = [{pcint:+2, dur:4},{pcint:+5}]
$p(P|\oplus) = 8,\ p(P|\ominus) = 4,\ I(P) = \frac{8}{4} = 2$
S-step
P = [{pcint:+2, dur:4},{pcint:+5}, {pcint:-1}]
$p(P|\oplus) = 6,\ p(P|\ominus) = 2,\ I(P) = \frac{6}{2} = 3$
I-step
...
P = [{pcint:+2, dur:4},{pcint:+5, dur:8}]
$p(P|\oplus) = 5,\ p(P|\ominus) = 2,\ I(P) = \frac{5}{2} = 2.5$
I-step
I-step
P = [{pcint:+2, dur:4},{pcint:+5, dur:8, pc:2}]
$p(P|\oplus) = 1,\ p(P|\ominus) = 2,\ I(P) = \frac{1}{2} = 0.5$
...
P = [{pcint:+2, dur:4},{pcint:+5, dur:8, pc:6}]
$p(P|\oplus) = 3,\ p(P|\ominus) = 0,\ I(P) = \frac{3}{0} = \infty$

Figure 3.2.: Part of the search tree of the MGDP algorithm. Three viewpoints are used: *pitch class interval* (`pcint`), *duration* (`dur`) and *pitch class* (`pc`). At each node, the algorithm can extend the pattern by either an I-step or an S-step. The distinctiveness threshold is set to $I(P) \geq 3$ and the minimum support to $p(P|\oplus) > 2$. Using these parameters, the thick bordered nodes are selected as *maximally general distinctive* patterns. The tree rooted at the bottom left node is not explored further, since the pattern occurs only once in the corpus and is therefore not general enough.

3.1.2. PatMinr

For the MGDP algorithm (section 3.1.1), we saw that the output of repeated patterns is filtered based on three properties: high support, high distinctiveness and maximally general. The PatMinr algorithm by Lartillot (2014a) uses different filtering heuristics, but they are based on the same principle as the MGDP algorithm. *PatMinr* is implemented as module within the larger *MiningSuite* framework[1] and also participated in the *Discovery of Repeated Themes and Sections* MIREX task in 2014 (Lartillot, 2014b). The algorithm works on monophonic, symbolic music and a sequential data structure is used. In contrast to the MGDP algorithm, PatMinr is only capable of finding patterns within *single songs* rather than in a corpus of songs.

PatMinr also uses the concept of *general* patterns (figure 3.1). Recall that a pattern P is more general than a pattern Q if all occurrences of Q are also occurrences of P. We can invert this statement by saying that if P is more general than Q, than Q is *more specific* than P. Where the MGDP algorithm returns the maximally general distinctive patterns (i.e. patterns that are as general as possible while being distinctive with respect to an anti-corpus), PatMinr only returns patterns that occurs more often than all patterns that are more specific. These patterns are called *closed*, an example is illustrated in figure 3.3. The reason for this is that the limitation of the output on only *maximally* general patterns filters out possible interesting (noticeable) patterns. In the example of figure 3.3, the MGDP algorihm will never return the (closed) pattern “ABCDECDE”, since “CDE” is already more general. This means that more specific patterns will not be returned, since they are already covered by the more general pattern. Although the selection of patterns is less strict than the MGDP algorithm, a combinatorial explosion

[1]https://code.google.com/p/miningsuite/

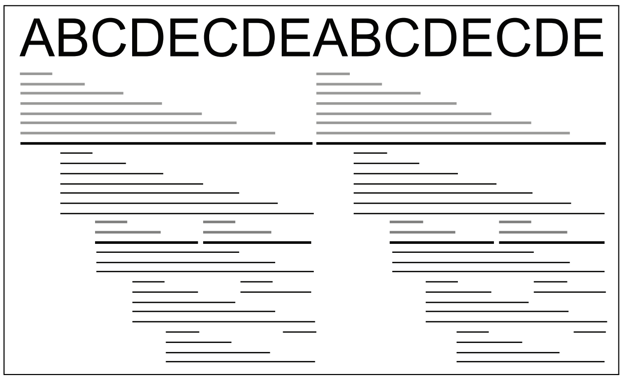


Figure 3.3.: The thick black bars represent the closed patterns. The grey lines are prefixes of the closed patterns and the thin black lines are non-closed patterns. Image from (Lartillot, 2014b).

of the output size is also prevented since most of the discovered patterns are contained in a closed pattern.

The algorithm traverses trough the musical sequence from beginning to end only once. At each new visited note, the algorithm checks whether a pattern (based on the previous notes and earlier occurrences) can be extended or that a new pattern has to be formed. To achieve this, a prefix tree is used to store the prefixes of all (candidate) closed patterns. At the end of the run, all closed patterns with at least n notes are returned. To further limit the output size, PatMinr explicitly models *pattern cyclicity*. This means that a periodic pattern will only be returned once, instead of returning all its (possibly overlapping) repetitions. To illustrate the problem of pattern cyclicity, an example is illustrated in figure 3.4. Without the modeling of pattern cyclicity, a Pattern Discovery algorithm would return all possible repeating subsequences, where PatMinr only returns "ABC".

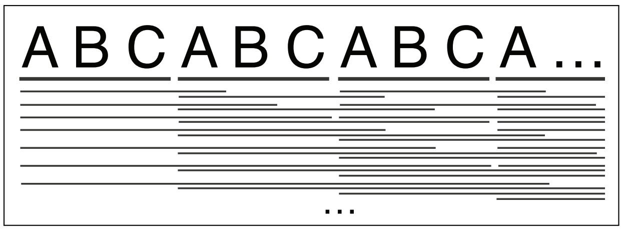
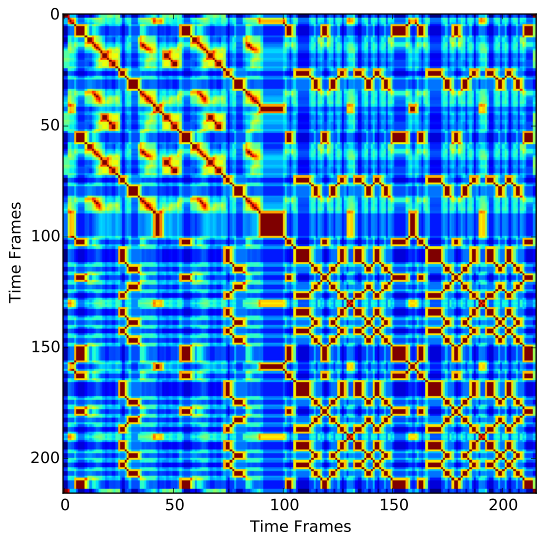


Figure 3.4.: Explicit modeling of pattern cyclicity prevents the output of countless overlapping patterns. Because of the single traversal through the piece, the repeated patterns "BCA" or "CAB" are also discarded since "ABC" is discovered first. The algorithm will therefore choose to extend the pattern starting with "A" instead of generating a new pattern starting with "B" or "C". Image from (Lartillot, 2014b).

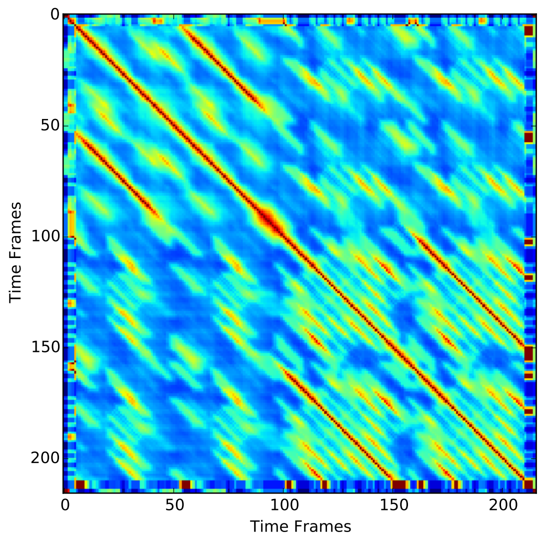
Algorithm parameters

Symbol	Description
n	The minimum number of notes in the pattern[2]

[2] In (Lartillot, 2014b), the minimum pattern length is defined as the number of nodes in the tree. Since the root is empty, the number of nodes in the pattern equals $n+1$.



(a) The transposition-invariant self-similarity matrix for symbolic music.



(b) A diagonal filter is applied to the SSM in order to make the diagonals more clear.

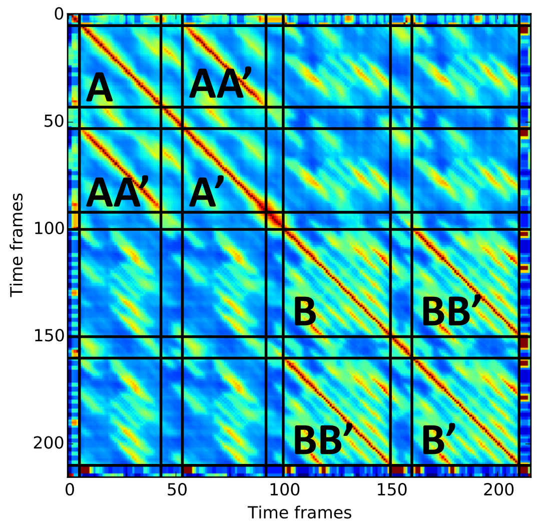
Figure 3.5.: The key transposition invariant Self Similarity Matrix used by Nieto and Farbood (2014b), based on the song of figure 3.7. The colors in the matrix indicate the similarity between the time frames. Blue pixels correspond to a low similarity between the time frames, red pixels represent a high similarity between the two frames.

3.1.3. MotivesExtractor

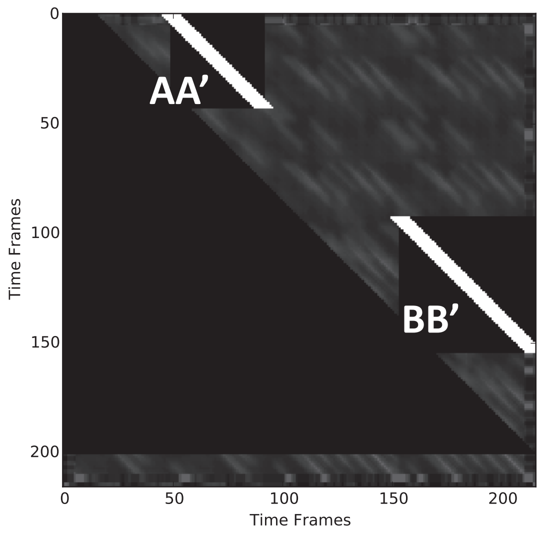
The MotivesExtractor-algorithm by Nieto and Farbood (2014a) was originally developed for extracting repeating patterns from polyphonic audio recordings, but was adapted to also support symbolic music in order to participate in all four subtasks of the 2014 *Discovery of Repeated Themes and Sections* MIREX task (Nieto and Farbood, 2014b). These four subtasks are defined by the musical texture (monophonic or polyphonic music) and the representation of music (audio or symbolic music). We will focus on the monophonic and symbolic implementation. The algorithm is developed to find both exact and inexact repetitions within *songs*.

The general idea behind this method is that repeated segments can be identified by comparing a musical piece with itself. For example, we can split a song in n time frames and calculate the similarity score between all pairs of time frames, based on pitch. This results in a $n \times n$ self-similarity matrix, where repeated patterns can be identified by diagonal lines in the matrix where the similarity score is high. This method cannot deal with repetitions that are transposed: we consider the sequence $\{\mathrm{B}3, \mathrm{G}3\}$ equal to $\{\mathrm{E}4, \mathrm{C}4\}$ since the pitch intervals are the same. Therefore, a *transposition-invariant* self-similarity matrix (Müller and Clausen, 2007) is introduced that is able to visualize repeated segments in a song even when the repetitions are transposed. This means that for all possible transpositions (12 in total) the self-similarity matrix is calculated and combined into one transposition-invariant self-similarity matrix. This leads to a matrix like figure 3.5 where the red diagonals show a high similarity between the time frames. A diagonal filter is applied to the matrix to make the diagonals more visible (figure 3.5b).

Then, a greedy tracing algorithm searches for diagonal paths with a high similarity score in the matrix above the main diagonal. This is sufficient, since the matrix is symmetric and the main diagonal represents the comparison between two identical time frames. Each diagonal path is given a score, based on the total degree of similarity of that path in the self-similarity matrix. The sharper the contour of the diagonal, the more prominent the path. This can be seen in figure 3.5b, where the yellow and



(a) The SSM can be split into four segments, of which segment A' is a repetition of A (by AA') and segment B' is a repetition of B (by BB')



(b) Two repetitions (AA' and BB') are returned by the algorithm, by searching for diagonal paths in the SSM above the main diagonal.

Figure 3.6.: Repeated segments found in the transpose-invariant SSM.

cyan diagonals are more fuzzy than the red paths. As an example, in the matrices from figure 3.5, two patterns are returned by the algorithm (figure 3.6). In figure 3.7, the original folk song is shown.

To deal with small rhythmic variations in the discovered paths, a tolerance variable ρ is introduced. The value of ρ determines how strict the tracing scoring function penalizes paths that are not perfectly diagonal. Since the self-similarity matrix is divided intro time frames, a rhythmic variation of a repetition could lead to a longer or shorter variation. This also influences the shape and orientation of the path in the self-similarity matrix. For greater values of ρ, the similarity score for nearly diagonal and more fuzzy paths increases. All paths with a similarity score greater than a threshold value θ are accepted as pattern. The last step of the algorithm is to cluster all discovered paths into patterns with groups of occurrences, where the parameter Θ is a tolerance parameter that defines how much the start and endpoints of two paths may differ in position in order to be accepted as occurrences of the same pattern.

Algorithm parameters

Symbol	Description
ρ	Defines the path tracing tolerance. For $\rho > 1$, the diagonal in the matrix does not have to be perfectly straight in order to get a good score.
θ	Scoring threshold for a path to be accepted as pattern.
Θ	Tolerance parameter for pattern clustering.
ν	The minimum number of notes in the pattern.

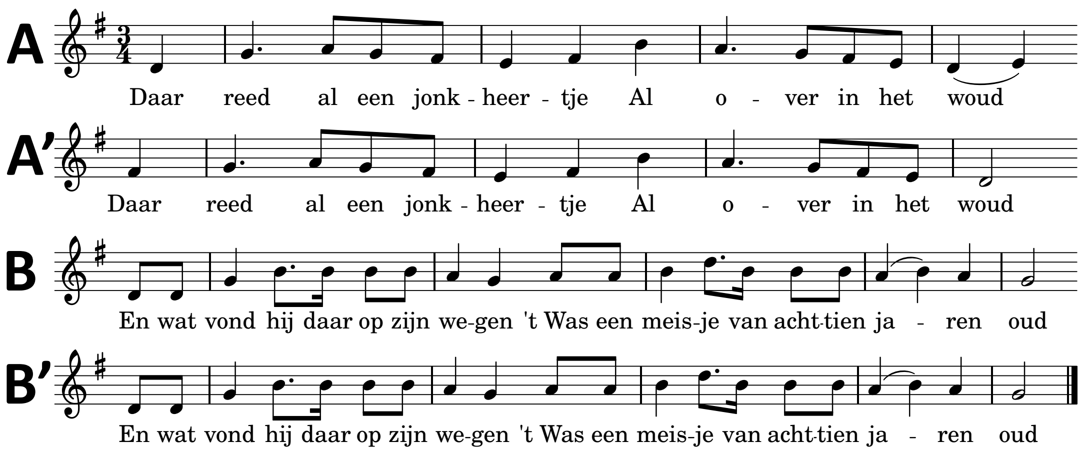

A
Daar reed al een jonk - heer - tje Al o - ver in het woud
A'
Daar reed al een jonk - heer - tje Al o - ver in het woud
B
En wat vond hij daar op zijn we-gen 't Was een meis-je van acht-tien ja - ren oud
B'
En wat vond hij daar op zijn we-gen 't Was een meis-je van acht-tien ja - ren oud

Figure 3.7.: The song *Daar reed al een jonkheertje* (Song id: NLB072883, Dutch Folksong Database) with the discovered patterns by the MotivesExtractor algorithm. Although there are slight variations, segment A' is considered a repetition of A and B' is a repetition of segment B.

3.2. Geometry Based Pattern Discovery algorithms

Unlike the algorithms discussed in the previous section, we will now look at methods that use a *geometric* representation of music in order to find repeated patterns. We have seen in section 2.3.2 that in a geometric approach, a note can be represented by a point in a multidimensional space. This will solve the problem that for sequential structures, it is hard to represent polyphonic music. In this section, we will first discuss a number of elementary approaches to geometry based pattern discovery. Then, we will discuss a number of additions to these algorithms, in order to improve the quality and (limitation of the) quantity of the output.

3.2.1. SIA & SIATEC

The first approach of a geometry based Pattern Discovery algorithm was the *Structural Induction Algorithm* (SIA) by Meredith et al. (2002). It can handle both monophonic and polyphonic symbolic music. The algorithm takes a set D as input, which is a geometric representation of the musical piece. The dataset consists of multidimensional vectors, which represent the notes in the musical piece. To reduce the complexity, the dimensionality of D is often reduced to only 2 dimensions: onset and morphetic pitch (see figure 2.7).

We define the translational vector $\mathbf{v}$ that translates an arbitrary point $\mathbf{d_1} \in D$ to another point $\mathbf{d_2} \in D$ as $\mathbf{v} = \mathbf{d_1} - \mathbf{d_2}$. For each such vector, SIA returns the *Maximal Translatable Pattern* (MTP), which is the largest pattern in D translatable by $\mathbf{v}$. More formally, we can write:

$$MTP(\mathbf{v}, D) = \{\mathbf{d} | \mathbf{d} \in D \wedge \mathbf{d} + \mathbf{v} \in D\}. \tag{3.5}$$

This results in tuples of patterns (P_1, P_2) that are *translational equivalent* (denoted $P_1 \equiv_\tau P_2$). An example of an MTP is given in figure 3.8. Since we are not just interested in *pairs* of occurrences but in *all* occurrences in a pattern, we want to group these occurrences into one pattern. The SIATEC algorithm bundles all patterns that belong to the same *Translational Equivalence Class* (TEC). That is, all patterns

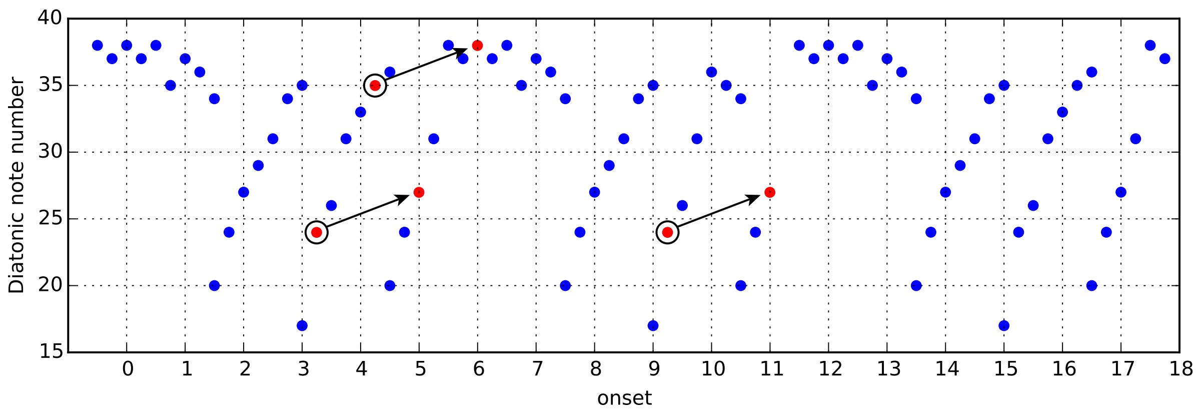


Figure 3.8.: Geometric representation of the beginning of Beethoven's *Für Elise*. The MTP for the vector $\mathbf{v} = \langle 1.75, 3 \rangle$ is highlighted.

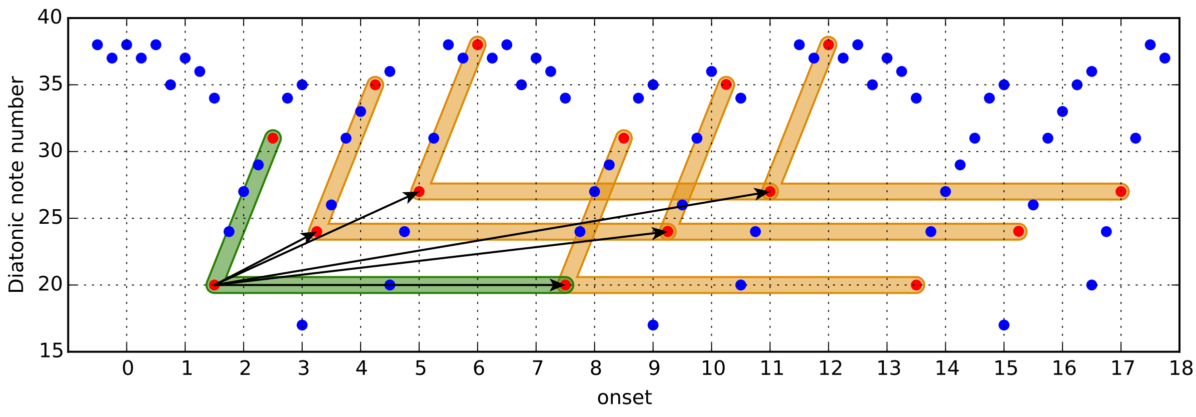


Figure 3.9.: SIATEC finds 328 Translational Equivalence Classes in total. The red points all belong to the same TEC of the three-note pattern.

that can be found by applying the same translation vector $\mathbf{v}$. For a pattern P we can define the TEC of P as

$$TEC(P, D) = \{Q | Q \equiv_\tau P \wedge Q \subseteq D\} \tag{3.6}$$

In figure 3.8, the red points belong to the same TEC with pattern $[\langle 1.5, 20\rangle, \langle 2.5, 31\rangle, \langle 7.5, 20\rangle]$ and the set of translators $V = [\langle 0, 0\rangle, \langle 1.75, 4\rangle, \langle 3.5, 7\rangle, \langle 6, 0\rangle, \langle 7.75, 4\rangle, \langle 9.5, 7\rangle]$.

3.2.2. COSIATEC

Both the SIA and SIATEC algorithm do not have any filtering method incorporated in order to limit the output size. Because of this, these algorithms suffer from one of the problems we discussed in the introduction section: an enormous amount of patterns is being discovered. To illustrate this, SIA discovers 540 MTP's in the first 82 notes of *Für Elise* in figure 3.8 and SIATEC finds 328 TEC's in the same excerpt (figure 3.9). Barlow and Morgenstern, in their *A dictionary of musical themes* (1948), have identified only one pattern in the entire song: the first three measures (see figure 2.1b).

COSIATEC (*Compressed SIATEC*, Meredith, 2006) is based on a greedy compression algorithm that iteratively filters the patterns returned by SIATEC. First, the SIATEC algorithm is run on the complete dataset. In each iteration, the best Translational Equivalence Class is selected, based on an heuristic with six *decision levels*. Each level compares a property of the TECs in order to decide which TEC is better, based on the musical intuition that large, compact patterns that repeat many times are perceptually more important than small, sparse patterns with few occurrences. If the first decision level does not result in a winner (i.e. it is a tie), then the second level is compared, and so on. These six levels/properties are, in order of importance (Meredith, 2013):

1. **Compression Ratio** Due to the properties of a TEC T, we can describe a pattern and all its occurrences as the first occurrence $\mathrm{P}(T)$ (which is a set of onset-pitch pairs) and all its occurrences by the set of translational vectors $\mathrm{V}(T)$. This way, we only have to store one occurrence, plus all of its translators to reconstruct all occurrences of T. The higher the compression ratio, the better the TEC. More formally, we can define the compression ratio as

$$\textsc{CompRatio}(T) = \frac{|\mathrm{P}(T)| + |\mathrm{V}(T)| - 1}{|\mathrm{COV}(T)|} \tag{3.7}$$

   Where $\mathrm{COV}(T)$ is the covered set of T, defined in equation 3.9. The TEC with the highest compression ratio is considered best.

2. **Compactness** We define the (bounding box) compactness of a TEC T as the number of points in the pattern of T devided by the number of points in the bounding box of P:

$$\textsc{Compactness}(T) = \frac{|\mathrm{P}(T)|}{|\mathrm{BB}(\mathrm{P}(T))|} \tag{3.8}$$

3. **Size of the covered set** A TEC T is considered better than a TEC U if the size of the *covered set* of T is larger than the size of the covered set of U: $|\mathrm{COV}(T)| > |\mathrm{COV}(U)|$. The covered set of T is defined as the union of all occurrences in T:

$$\mathrm{COV}(T) = \bigcup_{P \in T} P \tag{3.9}$$

4. **Pattern Size** The pattern size is defined as the number of elements in the first occurrence $\mathrm{P}(T)$ for a particular TEC T. At this point, a TEC is considered better when the pattern size is larger.

$$\textsc{PatternSize}(T) = |\mathrm{P}(T)| \tag{3.10}$$

5. **Pattern Width** The pattern width of a TEC T concerns the *duration* of the pattern. A patterns is considered better when the duration is shorter.

6. **Pattern Area** If all of the above measures did not select a winner, the TEC with the smallest area of the bounding box $\mathrm{BB}(\mathrm{P}(T))$ is considered to be better.

After the best TEC T is selected, $P(T)$ and all its occurrences are removed from the dataset. Then, SIATEC is run again on the smaller, compressed dataset. This process is repeated until no more patterns can be found. Because all occurrences of each pattern are stored as a translational vector, the original musical piece can be reconstructed by the discovered repeated patterns and the residue of the COSIATEC algorithm (i.e. the notes that do not belong to any pattern). The algorithm can therefore also be used as lossless compression method (Meredith, 2014).

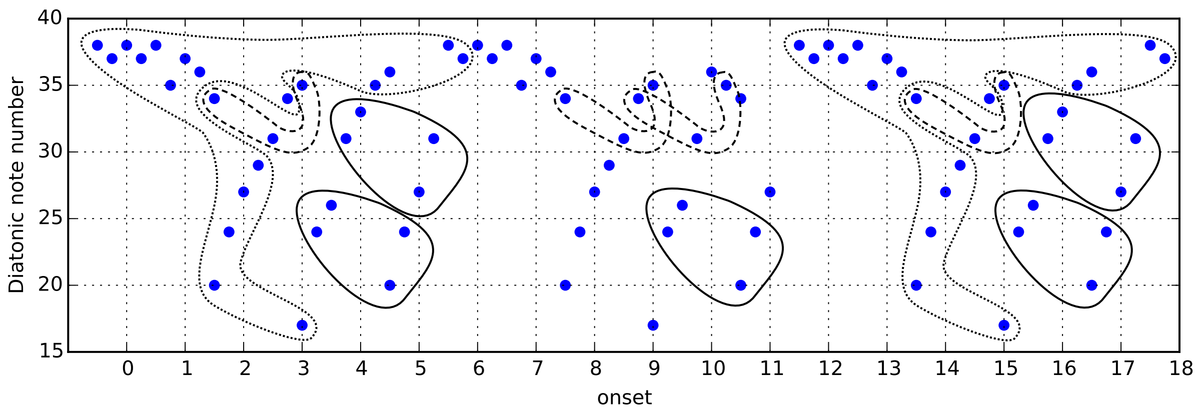


Figure 3.10.: Three repeated patterns are returned by COSIATEC, circled in the image. The pattern bounded by the continuous line is the most compact and is therefore discovered first. The dashed and dotted pattern are discovered second and third, respectively.

Due to the removal step after the best TEC is selected, each note is contained in at most one pattern. This is noticeable in the output size: on the same excerpt of *Für Elise*, COSIATEC will return only 3 repeated patterns (see figure 3.10). An improvement of this algorithm is SIATECCompress (Meredith, 2013), which only needs to run SIATEC once on the dataset. While this would not lead to the optimal compression, it runs faster on large musical pieces.

3.2.3. SIACT

In order to limit the output size of SIA or SIATEC to patterns that are *noticeable* to human listeners, Collins et al. (2010) observed that the MTP's returned by SIA had an unwanted effect on the output of the discovered patterns. Since SIA only returns *maximally* translatable patterns, it can happen that a potential noticeable pattern is contained in a much larger pattern. Because of the extra notes in this pattern, the entire sequence is not noticeable while a subset of the pattern could be. This situation is called the *problem of isolated membership*.

Collins et al. (2010) proposed a method to extract these potential interesting patterns from a much larger pattern using a *Compactness Trawler*. Since this is an addition of SIA, an algorithm that incorporates this Compactness Trawler is called SIACT (or SIACTTEC in the case of SIATEC, COSIACTTEC in the case of COSIATEC, etc.). The output of either SIA or SIATEC is analyzed by the Compactness Trawler. For each discovered pattern, the algorithm "trawls" inside the pattern to look for important (noticeable) subpatterns. A subpattern is accepted (hence returned) by the Compactness Trawler if it fulfills two conditions:

The minimum *compactness* of the pattern. See equation 3.8. In the equation, the bounding box is chosen as region around a pattern, but also other measures could be chosen (such as the convex hull). The compactness has to be larger than a threshold value a in order to be returned by the Compactness Trawler. This also filters MTP's that are not compact at all, for example the MTP in figure 3.8.

The minimum *length* of the pattern. See equation 3.10. This defines the minimum length of the pattern in order to be accepted by the Compactness Trawler.

Algorithm parameters

Symbol	Description
a	The minimum compactness value in order to be accepted by the Compactness Trawler.
b	The minimum length of a pattern returned by the Compactness Trawler.

3.2.4. SIARCT-CFP

One of the advantages of the geometric approach is the possibility to have "gaps" in the pattern, which makes it possible to detect variations in patterns, such as embellishments (Meredith et al., 2002). Lartillot and Toiviainen (2007) observed that repetitions containing rhythmic variations are hardly detected by geometric Pattern Discovery algorithms. SIARCT-CFP (SIACT with Categorization and Fingerprinting, Collins et al., 2013) is a method that can discover these kinds of *inexact* repetitions. It is also the only algorithm in the SIA family that is able to discover inexact occurrences of repeated patterns.

The R in the name of the algorithm comes from another adaption of SIA that improves the running time the algorithm (Collins, 2011). For SIA, an $n \times n$ matrix is used to store the translation vector between each pair of points in the dataset in order to calculate the Maximum Translatable Patterns. SIAR only uses the first r subdiagonals of the matrix to form the MTP's instead of all the values in the matrix. The first subdiagonals of a matrix are the values below the main diagonal, the second subdiagonals are the values below the first subdiagonal, and so forth. Like the other algorithms that are improvements of SIA, the results of SIARCT-CFP are filtered and expanded after the run of another geometric Pattern Discovery algorithm (such as SIACT). The extension of SIARCT-CFP is based on two new steps:

Fingerprinting Collins et al. (2013) observed that pattern *matching* algorithms were also not invariant to rhythmic variations, except the fingerprinting algorithm by (Arzt et al., 2012). This algorithm returns the location and similarity score of all exact and inexact matches in the dataset, given a pattern P. The fingerprinting subroutine in SIARCT-CFP uses this method to find all (partial) matches in the dataset for all discovered patterns, and adds the occurrences where the similarity score between P and the match in the dataset is greater than a threshold value c.

Categorization To further limit the number of repeated patterns, SIACT-CFP contains a *categorization* step. The goal of this step is to bundle (highly) similar patterns to one group of patterns. To do so, all discovered patterns are rated according to an empirically derived pattern importance function (Collins et al., 2011). Then, the pattern with the highest importance score is matched to all other patterns. All patterns with a similarity score higher than a threshold value m are merged with this pattern, meaning that its occurrences are now occurrences of the pattern with the highest importance score and the merged patterns are removed. Then, the pattern with the second best importance score is matched against the remaining patterns, and so on. This process is repeated until all patterns are categorized.

By this addition, SIARCT-CFP is the only algorithm in the SIA family that is capable of discovering inexact pattern occurrences.

Algorithm parameters

Symbol	Description
a	The minimum compactness value in order to be accepted by the Compactness Trawler.
b	The minimum length of a pattern returned by the Compactness Trawler.
c	The similarity threshold for two patterns to be similar for the fingerprinting algorithm.
m	The maximum distance for two patterns to be similar in the categorization step.

	Texture		**Domain**		**Matching**		**Structure**		**Representation**	
Algorithm	mono	poly	song	class	exact	inexact	seq.	geom.	audio	symb.
MGDP	x			x	x		x			x
MotivesExtractor	x	x	x			x	x		x	x
PatMinr	x		x		x		x			x
SIA/SIATEC	x	x	x		x			x		x
COSIATEC	x	x	x		x			x		x
SIACT	x	x	x		x			x		x
SIARCT-CFP	x	x	x		x	x		x		x

Table 3.1.: Properties of the Pattern Discovery algorithms discussed in this chapter.

3.3. Conclusion

Pattern Discovery algorithms are developed to find repeating sections in music. We have discussed eight different Pattern Discovery algorithms: three of them using a sequential representation of music as input and five based on the geometric representation. In the beginning of this chapter, five properties of Pattern Discovery algorithms are discussed. These properties are summarized for each algorithm in table 3.1.

The MGDP algorithm is the only method that is developed to discover repeated patterns across multiple musical pieces (Class Pattern Discovery). All other algorithms only support the discovery of repeated patterns within a single song (Song Pattern Discovery). Two of the sequential based algorithms (MGDP and PatMinr) make use of the concept of *general* patterns to filter out possibly unimportant (not-noticeable) patterns. This filters out patterns that are contained in other patterns. The third algorithm in this category (MotivesExtractor) uses a self-similarity matrix to discover repeated segments in a song. This is also the only algorithm that supports the input of audio without modifications.

For the geometry based methods, all algorithms we discussed are based on SIA(TEC) by adding improvements for filtering and categorizing patterns. COSIATEC uses SIATEC as compression step, SIACT filters patterns based on compactness properties of subpatterns *inside* the discovered patterns and SIARCT-CFP uses a matching algorithm to find inexact occurrences of a song and to combine multiple patterns. Together with MotivesExtractor, these are the only two algorithms that are capable of discovering inexact pattern occurrences.

With the use of these algorithms, a number of patterns are returned for each musical piece. In the next chapter, we discuss methods to compare and evaluate the output of Pattern Discovery algorithms, as well as methods to define the similarity between two entire songs.

Similarity of Patterns and Songs 4

In order to classify music based on similarity, we need measures that define the similarity between two (parts of) songs. Like we stated in the introduction, this is not a straightforward task. The problem is to what extend songs are perceived as similar, despite variation in melody, rhythm, etc. In this chapter, we will discuss computational methods to determine the similarity between *patterns* and methods that calculate the similarity between *songs*.

By calculating the similarity between patterns, we can easily compare the output of a Pattern Discovery algorithm with a set of annotated patterns that can be used as ground truth. This way, the output of a Pattern Discovery algorithm can be evaluated by the degree of similarity with a set of ground truth patterns. In section 4.1, we discuss different metrics to evaluate the quality of the discovered patterns, based on basic evaluation metrics used in MIR. In section 4.2, we discuss two algorithms that calculate the similarity between a pair of musical pieces. We will use the resulting similarity scores (i.e. a score of how similar both musical pieces are) later on to classify songs in groups with a high corresponding similarity.

4.1. Evaluation of Pattern Discovery Algorithms

The evaluation of the Pattern Discovery Algorithms described in chapter 3 is mostly done quantitatively (Janssen et al., 2013), comparing the output with *ground truth*. Also, the *Discovery of Repeated Themes and Sections* MIREX task evaluate the algorithms this way, using the *JKU Pattern Development Database* (JKU-PDD) as ground truth. This set contains pattern annotations from five songs, which are *"relatively uncontroversial and transparent"* (Collins, 2014a). This way, the discovered patterns are compared with a set of pattern annotations to measure the correspondence between both sets of patterns.

We will discuss the different evaluation metrics used in the MIREX task. For this, we use the following convention: $D = \{D_1, D_2, \ldots, D_N\}$ is the set of discovered patterns, $G = \{G_1, G_2, \ldots, G_M\}$ is the set of annotated patterns (the ground truth). Each pattern consists of a number of occurrences, which are denoted as $D_i = \{d_1, d_2, \ldots, d_n\}$ for a certain discovered pattern D_i. In general, a capital D or G represents a pattern, whereas a small d and g stands for a pattern *occurrence*.

4.1.1. Basic evaluation metrics in Information Retrieval

In the field of (Music) Information Retrieval, common evaluation metrics are *precision*, *recall* and *F_1-score* (Manning et al., 2009). Precision is a measure that tells us how many of the discovered patterns are relevant, i.e. also annotated in the ground truth:

$$\mathrm{P}(D,G) = \frac{|D \cap G|}{|D|}. \tag{4.1}$$

In other words, precision measures the *correctness* of the returned set D of discovered patterns, compared with the set of Ground Truth patterns in G.

Using precision as the only evaluation metrics gives an incomplete picture of the evaluation. For example, if our set of discovered patterns contains only one entry which is also present in the ground truth, the resulting precision would be 1. However, this does not say anything about the *amount* of Ground Truth patterns that is actually discovered. Precision is therefore always used in combination with *Recall*:

$$\mathrm{R}(D,G) = \frac{|D \cap G|}{|G|}. \tag{4.2}$$

As opposed to Precision, Recall gives us the ratio of Ground Truth patterns that are discovered by the Pattern Discovery algorithm. Note that recall does not penalize for extra discovered patterns on top of those which are also present in the set of Ground Truth patterns. Because both metrics are needed to fairly judge the output of the Pattern Discovery algorithm, the F_1 score combines both metrics into one equation:

$$F_1(D,G) = 2 \cdot \frac{\mathrm{P}(D,G) \cdot \mathrm{R}(D,G)}{\mathrm{P}(D,G) + \mathrm{R}(D,G)} \tag{4.3}$$

To avoid division by zero, we define the F_1 score to be 0 when both the precision and recall are 0.

4.1.2. Evaluation Metrics for Pattern Discovery

One of the problems with the precision, recall and F_1-score introduced in the previous section is that they are very strictly defined. This means that a discovered pattern $D_i \in D$ and a ground truth pattern $G_j \in G$ have to be *exactly* the same in order to be counted as match. More specifically, the occurrences of both patterns need to be exactly the same: the same number of occurrences and the exact same notes in each occurrence. In order to use these metrics in a more loose manner, three variations of the normal precision, recall and F_1-score are introduced in the *Discovery of Repeated Themes and Sections* MIREX task (Collins, 2014a). As opposed to treating each pattern “at once” in the evaluation procedure, these metrics will give us the answers to the following questions:

- How good is the algorithm at *establishing* all patterns (disregarding the number of occurrences)?
- How good is the algorithm in finding all *occurrences* of a pattern?
- How good is the algorithm in doing both?

In the following sections, we will discuss these three approaches in more detail. This is based on the work of Collins and Meredith (Collins, 2014a; Meredith, 2015).

4.1.3. Establishment Precision, Recall and F_1 score

A way of a less strict metric than the normal precision, recall and F_1 score is determining how well an algorithm can find *approximately* the same patterns as the ground truth patterns. While in the normal precision, recall and F_1 score a discovered pattern could be either in the set of ground truth patterns or not, we are now interested in the question how *similar* each pattern in one set is to the most similar pattern in the other set. Where the normal precision returned the ratio of ground truth patterns discovered, the *establishment* precision returns an average score of how *similar* each discovered pattern is to the most similar ground truth pattern. For the establishment *recall* metric, it is exactly the other way around: the average similarity score for each ground truth pattern with the most similar discovered pattern. This way, each pattern is matched to the *best* (but not necessarily the exact same) pattern in the other set, which is less strict than the metrics described in section 4.1.

For a given discovered pattern $D_i = \{d_1, d_2, \ldots, d_n\}$ with n occurrences and a given ground truth pattern $G_j = \{g_1, g_2, \ldots, g_m\}$ with m occurrences, we introduce a scoring function $s(d_i, g_j)$ that returns a similarity score between an occurrence of D and an occurrence of G. For this, we follow the MIREX implementation by using the *cardinality score* defined as

$$s(d_i, g_j) = \frac{|d_i \cap g_j|}{\max\left(|d_i|, |g_j|\right)} \tag{4.4}$$

This will give us the ratio of notes that are contained in both occurrences: the overlap. The reason for choosing the cardinality score as similarity measure is that this metric is very straightforward and intuitive, although also other, more complex similarity metrics could be used (for example the fingerprinting algorithm by Arzt et al. (2012) that is used for SIARCT-CFP in section 3.2.4). To calculate the similarity score between *all* pairs of occurrences (d_i, g_j), we will build an $n \times m$ *score matrix* $S(D, G)$.

$$S(D_i, G_j) = \begin{bmatrix} s(d_1, g_1) & s(d_1, g_2) & \cdots & s(d_1, g_m) \\ s(d_2, g_1) & s(d_2, g_2) & \cdots & s(d_2, g_m) \\ \vdots & \vdots & \ddots & \vdots \\ s(d_n, g_1) & s(d_n, g_2) & \cdots & s(d_n, g_m) \end{bmatrix} \tag{4.5}$$

We will define $S_{\max}(D_i, G_j)$ as the maximum similarity score between any of the occurrences of pattern D_i and pattern G_j, which is the maximum score in the matrix $S(D_i, G_j)$:

$$S_{\max}(D_i, G_j) = \max\left(S(D_i, G_j)\right) \tag{4.6}$$

Since the cardinality score returns a value in the range $[0, 1]$, the value of $S_{\max}(D_i, G_j)$ will be in the same range. This value gives notion about the maximum similarity score between *any* of the occurrences of D_i and *any* of the occurrences of G_j. If $S_{\max}(D_i, G_j)$ is (almost) one, we know that at least one occurrence of both patterns are (almost) the same, which is enough in this case. We will now define the *establishment precision* and *establishment recall*, which is the average over all maximum similarity scores for each pattern in the set of discovered patterns and the set of ground truth patterns, respectively. We can do so by taking the maximum value of the scoring matrix $S(D_i, G_j)$, for each discovered pattern $D_i \in D$ ($0 < i \leq N$) and ground truth pattern $G_j \in G$ ($0 < j \leq M$):

$$\mathrm{P}_{\mathrm{est}}(D, G) = \frac{1}{N} \cdot \sum_{i=1}^{N} \max_{j=1}^{M} \left(S_{\max}\left(D_i, G_j\right)\right) \tag{4.7}$$

In other words, the establishment precision is the average of each discovered pattern to the *most similar* ground truth pattern. For the establishment recall, we want to know how well every ground truth pattern is represented in the set of discovered patterns. We therefore take the average similarity score for each ground truth pattern with the *most similar* discovered pattern:

$$\mathrm{R}_{\text{est}}(D,G) = \frac{1}{M} \cdot \sum_{j=1}^{M} \max_{i=1}^{N} \left(S_{\max} \left(D_i, G_j \right) \right) \tag{4.8}$$

Like in section 4.1, N is the total number of discovered patterns and M is the total number of ground truth patterns. The *establishment* F_1*-score* can be calculated the same way as in equation 4.3, replacing the normal precision and recall functions by the established precision and recall.

4.1.4. Occurrence Precision, Recall and F_1 score

In the previous section, we have seen that a more flexible variant of the normal precision, recall and F_1 score can be defined by only looking at the most similar occurrences of a discovered and ground truth pattern rather than checking if all occurrences are exactly the same. However, this does not tell us anything about how many of the *occurrences* of a ground truth pattern are actually discovered. We therefore introduce a second set of evaluation metrics: *occurrence* precision, recall and F_1 score. Instead of measuring the performance by the number of ground truth patterns that are *established* in the algorithm's output, we will now check how many occurrences of a ground truth pattern are also returned by a Pattern Discovery algorithm.

We know from the scoring matrix in equation 4.5 that, if a value in the matrix is close to 1, the discovered pattern occurrence and the ground truth occurrence are very similar. In order to also allow slight variations in the occurrences, we choose a threshold value c such that if $\max\left(S\left(D,G\right)\right) \geq c$, we assume that the discovered pattern D and ground truth pattern G have at least one occurrence in common (that may be an *inexact* occurrence if we choose $c < 1$).

For the occurrence precision, we want to know how many occurrences of each discovered pattern are also present in any of the ground truth patterns. Therefore, we have to match all occurrences from each ground truth pattern D_i with all occurrences of G_j where there is at least one similar occurrence between the 2 patterns. In other words, when $S_{\max}(D_i, G_j) \geq c$. We build an $N \times M$ precision matrix P, where we calculate how well the occurrences between the patterns D_i and G_j are similar:

$$P_{i,j} = \begin{cases} \frac{1}{n} \cdot \sum\limits_{k=1}^{n} \max\limits_{l=1}^{m} \left(s\left(d_k, g_l\right) \right) & \text{if } S_{\max}(D_i, G_j) \geq c \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

For each pattern occurrence in D_i, the highest similarity score of the most similar occurrence in G_j is returned. Here, n is the number of occurrences of pattern D_i and m is the number of occurrences of G_j. The final value is the average over all (maximum) similarity scores. In other words, this value represents how well the occurrences of D_i are also present in G_j. If there is no significant match between any of the occurrences from both sets (i.e. $S_{\max}(D_i, G_j) < c$), the value in the matrix is zero.

For the occurrence *recall*, we want to know the exact opposite: how many of the occurrences from a ground truth pattern G_j are also present in a discovered pattern D_i? We therefore also introduce an

$N \times M$ recall matrix R with

$$R_{i,j} = \begin{cases} \frac{1}{m} \cdot \sum_{l=1}^{m} \max_{k=1}^{n} \left(s\left(d_k, g_l\right)\right) & \text{if } S_{\max}(D_i, G_j) \geq c \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

For the occurrence precision, we find for each discovered pattern the most similar ground truth pattern in terms of similarity between the occurrences of both patterns using the precision matrix P. The results are being averaged for all discovered patterns in order to get the occurrence precision. If none of the ground truth patterns have a similar enough occurrence for a certain discovered pattern D_i (i.e. the ith column in the matrix P only contains zero's), the pattern is discarded in calculating the average. Like in the previous cases, the occurrence recall is calculated by finding the most similar *discovered* pattern for each *ground truth* pattern.

$$\mathrm{P}_{\mathrm{occ}} = \frac{1}{N'} \cdot \sum_{i=1}^{N} \max_{j=1}^{M} \left(P_{i,j}\right) \tag{4.11}$$

$$\mathrm{R}_{\mathrm{occ}} = \frac{1}{M'} \cdot \sum_{j=1}^{M} \max_{i=1}^{N} \left(R_{i,j}\right) \tag{4.12}$$

Where N' is the number of non-zero columns in matrix P and M' is the number of non-zero rows in matrix R. As with the established precision and recall, the occurrence F_1 score can be calculated using equation 4.3.

4.1.5. Three-layer precision, recall and F_1 score

Up to now, we have seen three sets of evaluation metrics to compare a set of discovered patterns with a set of ground truth patterns: the *normal* precision, recall and F_1 score, the *establishment* precision, recall and F_1 score and the *occurrence* precision, recall and F_1 score. We will now introduce a fourth set of metrics, that combine the previous three: the *three-layer* precision, recall and F_1 score. Like the name suggests, the calculation of these metrics consists of three steps:

1. Calculate precision, recall and F_1 score on a discovered pattern occurrence and a ground truth pattern occurrence.

2. Calculate precision, recall and F_1 score on a discovered pattern and a ground truth pattern.

3. Calculate precision, recall and F_1 score on the set of discovered patterns and the set of ground truth patterns.

In other words, in the first layer two pattern *occurrences* are being compared. In the second layer, two *patterns* are compared, based on the results of the first layer. Finally, the third layer returns the evaluation metrics based on the complete sets of patterns, based on the results of the second layer. Thus, the similarity is calculated from the most detailed level to the similarity of the entire sets of patterns. We will now explain the calculation of these three layers in more detail.

First Layer: Comparison of Pattern Occurrences

In the first layer, a pattern occurrence of a discovered pattern is being compared with a pattern occurrence of a ground truth pattern. Remember that *precision* will give us the ratio of discovered patterns that are also contained in the set of ground truth patterns, while recall returns the ratio of ground truth patterns that are contained in the set of discovered patterns. For a discovered pattern occurrence $d_i \in D_i$ and a ground truth pattern occurrence $g_j \in G_j$. The first-layer precision and recall are defined as:

$$\mathrm{P}_1(d_i, g_j) = \frac{|d_i \cap g_j|}{|d_i|}. \tag{4.13}$$

$$\mathrm{R}_1(d_i, g_j) = \frac{|d_i \cap g_j|}{|g_j|}. \tag{4.14}$$

Note that these equations are the same as the *normal* precision and recall, despite the fact that now only pattern occurrences are being compared. The equations also shows a high similarity with the cardinality score from equation 4.4. The first layer precision returns the ratio of notes in the discovered pattern that overlaps with the ground truth pattern, where the first layer recall returns the ratio of notes in the ground truth pattern that overlaps with the discovered pattern. Just like the normal F_1 score, the first-layer F_1 score is defined as

$$F_1(d_i, g_j) = 2 \cdot \frac{\mathrm{P}_1(d_i, g_j) \cdot \mathrm{R}_1(d_i, g_j)}{\mathrm{P}_1(d_i, g_j) + \mathrm{R}_1(d_i, g_j)} \tag{4.15}$$

Second Layer: Comparison of Patterns

We will now evaluate the similarity between *patterns* instead of single occurrences. For this, we can use the equations of the first layer, because the similarity between two patterns is determined by the similarity between its occurrences. The second layer precision and recall are calculated the same way as the occurrence precision and recall, by finding the most similar pattern occurrence from the other set (i.e. the pattern occurrence with the highest similarity score) and average over all occurrences. We will use the F_1 score from the first layer to calculate the second-layer precision and recall for a given discovered pattern $D_i = \{d_1, d_2, \dots, d_n\}$ with n occurrences and a given ground truth pattern $G_j = \{g_1, g_2, \dots, g_m\}$ with m occurrences:

$$\mathrm{P}_2(D_i, G_j) = \frac{1}{n} \cdot \sum_{k=1}^{n} \max_{l=1}^{m} \left(F_1(d_k, g_l)\right) \tag{4.16}$$

$$\mathrm{R}_2(D_i, G_j) = \frac{1}{m} \cdot \sum_{l=1}^{m} \max_{k=1}^{n} \left(F_1(d_k, g_l)\right) \tag{4.17}$$

The second-layer F_1 score (which we will call F_2) is defined the same way as equation 4.15, but now with the second layer precision and recall instead of the first layer:

$$F_2(D_i, G_j) = 2 \cdot \frac{\mathrm{P}_2(D_i, G_j) \cdot \mathrm{R}_2(D_i, G_j)}{\mathrm{P}_2(D_i, G_j) + \mathrm{R}_2(D_i, G_j)} \tag{4.18}$$

Third Layer: Comparison of the Complete Sets of Patterns

Analog to the second layer where we used the occurrence-wise similarity of the first layer to calculate the pattern-wise similarity of the second layer, we can now use the results from the second layer to compare the complete set of discovered patterns with the complete set of ground truth patterns. Remember that we have a set of discovered patterns $D = \{D_1, D_2, \ldots, D_N\}$ and a set of Ground Truth patterns $G = \{G_1, G_2, \ldots, G_M\}$. The third layer precision and recall are then defined as:

$$\mathrm{P}_3(D, G) = \frac{1}{N} \cdot \sum_{i=1}^{N} \max_{j=1}^{M} \left(F_2(D_i, G_j) \right) \tag{4.19}$$

$$\mathrm{R}_3(D, G) = \frac{1}{M} \cdot \sum_{j=1}^{M} \max_{i=1}^{N} \left(F_2(D_i, G_j) \right) \tag{4.20}$$

And the third layer F_1 score (F_3) then becomes:

$$F_3(D, G) = 2 \cdot \frac{\mathrm{P}_3(D, G) \cdot \mathrm{R}_3(D, G)}{\mathrm{P}_3(D, G) + \mathrm{R}_3(D, G)} \tag{4.21}$$

These equations are the three-layer precision, recall and F_1 score, respectively. Like in the original meaning of precision and recall, the three-layer precision averages over the maximum F_2 scores for each discovered pattern with any ground truth pattern. For the three-layer precision, the average is taken over the maximum F_2 scores for each ground truth pattern with any discovered pattern.

4.2. Melodic Similarity using Sequence Alignment

To calculate the similarity between two melodies, we will use a technique called *sequence alignment*. The reason for this is that sequence alignment has been proven to be an effective similarity measure for monophonic (folk) music (Hillewaere et al., 2012). Sequence alignment is a method that originates from computational biology to find (partial) matches in DNA protein strings. The idea is that each character of two strings is piecewise compared and based on a scoring function, the optimal alignment can be found.

A very simple distance function is the *Levenshtein distance*, which allows three operations for each character comparison step when both characters are not equal:

- The *addition* of a character
- The *deletion* of a character
- The *substitution* of a character

Suppose that each operation leads to an addition to the distance of 1, then the Levenshtein distance between the words `BEAR` and `BARS` is 2 since the `E` of `BEAR` needs to be added to `BARS` (getting `BEARS`) and the `S` of `BARS` needs to be removed.

More formally, the optimal alignment of two songs can be calculated using the Needleman-Wunsch algorithm (Needleman and Wunsch, 1970). This is a dynamic programming based solution, by iteratively traversing a matrix D. Let $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_m\}$ be two musical pieces and x

and y be notes. The optimal alignment between these two pieces can then be found by calculating all values in a matrix D:

$$D_{i,j} = \max \begin{cases} D_{i-1,j-1} + S(x_i, y_j) \\ D_{i-1,j} - \gamma \\ D_{i,j-1} - \gamma \end{cases} \tag{4.22}$$

Here, $S(x_i, y_i)$ is the similarity scoring function for the notes x_i and y_j and γ is the penalty score for leaving a gap in the alignment. A very simple scoring function could be $\gamma = 1$ and

$$S(x_i, y_j) = \begin{cases} 1 & \text{if } \Phi(x_i) = \Phi(y_j) \\ -1 & \text{otherwise} \end{cases} \tag{4.23}$$

where $\Phi(x)$ is a transformation function that transforms a note x to a single value. For example, if we want to align two musical sequences based on pitch, we will use $\Phi(x) \equiv \text{PITCH}(x)$. The similarity score between X and Y is equal to the value $D_{n,m}$ (the bottom right value in the matrix). Note that a penalty adds a negative value to the similarity score, so the higher the similarity score, the more similar the songs.

An example of the alignment calculation with the matrix D can be seen in figure 4.1. At first, the scores of the first row and column are calculated. Since each character is being compared with "nothing", the first row and column are populated by the penalty score. Then, the matrix is filled from the top left to the bottom right using equation 4.22. It follows from this equation that in order to calculate the score of a particular cell, the scores of its left, upper left and upper neighbor already need to be known. After determining the maximum score for the cell, the score is being stored in the matrix, as well as a pointer to the neighbor responsible for this maximum score.

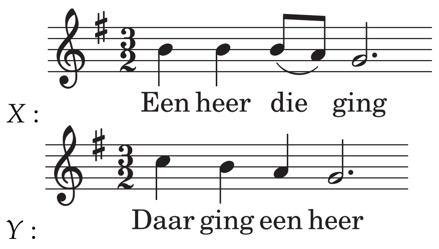
As an example, assume that we want to calculate the value of $D_{2,2}$, the matching of the second note in both melodies. The first possibility is matching, which results in a score of $(D_{1,1} + S(x_2, y_2)) = 0$, since we have a match. The scores for leaving a gap in either X or Y are $(D_{1,2} - \gamma) = -1$ and $(D_{2,1} - \gamma) = -3$, respectively. The best choice here is a match, and the score of 0 is stored together with a pointer to $D_{1,1}$. When the bottom right cell of the matrix is reached, the optimal alignment can be found by *backtracking* to the top left cell using the stored pointers in each cell. A diagonal path means matching the two characters, whereas a horizontal or vertical path represent a gap in the alignment. For example, the highlighted optimal alignment of X and Y in figure 4.1 is

$$\begin{array}{rlllll} X = & \{b_3, & b_3, & b_3, & a_3, & g_3\} \\ Y = & \{c_3, & b_3, & & a_3, & g_3\}. \end{array}$$

We can easily verify that this leads to a similarity score of 1, since the alignment involves three matches, one mismatch and one gap.

4.2.1. Note to Note Alignment

Van Kranenburg et al. (2013) have shown that note to note alignment is a good similarity measure for classifying the folk songs from the *Annotated Corpus*. They use the Needleman-Wunsch-Gotoh alignment algorithm (Needleman and Wunsch, 1970; Gotoh, 1982), which is an extension of the general Needleman-Wunsch algorithm explained in the previous section. Instead of a single penalty score, a



(a) The first six beats of two songs from the tune family "Daar ging een heer 1". When converting the scores into sequences of scientific pitch notation, we get $X = \{b_3, b_3, b_3, a_3, g_3\}$ and $Y = \{c_4, b_3, a_3, g_3\}$.

$$\begin{tikzcd}
 & & b_3 & b_3 & b_3 & a_3 & g_3 \\
 & 0 & -1 \arrow[l] & -2 & -3 & -4 & -5 \\
c_4 & -1 & -1 \arrow[ul, red] & -2 \arrow[ul] \arrow[l] & -3 & -4 & -5 \\
b_3 & -2 & 0 & 0 \arrow[ul, red] & -1 \arrow[ul] \arrow[l, red] & -2 & -3 \\
a_3 & -3 & -1 & -1 & -1 & 0 \arrow[ul, red] & -1 \\
g_3 & -4 & -2 & -2 & -2 & -1 & 1 \arrow[ul, red]
\end{tikzcd}$$

(b) Path of the best alignment by the Needleman-Wunsch algorithm. There are multiple best alignments, one is highlighted in red.

Figure 4.1.: An example of the similarity calculations of two melodies using sequence alignment. The matrix is built using the similarity scoring function of equation 4.23 and with a gap penalty of $\gamma = 1$.

different penalty for gap openings and gap extensions is introduced. This transforms the calculation of matrix D in equation 4.22 to:

$$D_{i,j} = \max \begin{cases} D_{i-1,j-1} + S(x_i, y_j) \\ E_{i-1,j} + S(x_i, y_j) \\ F_{i,j-1} + S(x_i, y_j) \end{cases} \tag{4.24}$$

where

$$E_{i,j} = \max \begin{cases} D_{i,j-1} - \gamma_{open} \\ E_{i,j-1} - \gamma_{ext} \end{cases}$$
$$F_{i,j} = \max \begin{cases} D_{i-1,j} - \gamma_{open} \\ E_{i-1,j} - \gamma_{open} \\ F_{i-1,j} - \gamma_{ext} \end{cases} \tag{4.25}$$

Here, γ_{open} and γ_{ext} are defined as gap opening penalty and gap extension penalty, respectively. The matrices E and F are needed to keep track of the gap length of songs X and Y, respectively.

By choosing $S(x_i, y_i)$ as a combination of pitch band, metric weight (a measure for how much the note is important for the rhythm) and phrase position (the position of a note in the musical phrase), Van Kranenburg et al. achieved a positive classification ratio of 0.99. To reduce the chance of over-fitting on the dataset and to make the effect of the note removal more visible, we use a slightly simpler similarity scoring function, which only takes the pitch interval into account:

$$S(x_i, y_j) = \begin{cases} 1 & \text{if } \textsc{interval}(x_i) = \textsc{interval}(y_j) \\ -1 & \text{if } \textsc{interval}(x_i) \neq \textsc{interval}(y_j) \end{cases} \tag{4.26}$$

Where $\textsc{interval}(x_i) = \textsc{pitch}(x_i) - \textsc{pitch}(x_{i-1})$ and $\textsc{interval}(x_0) = \textsc{interval}(y_0) = 0$. All resulting distances are normalized to fit into a 0 to 1 range.

4.2.2. Melodic Shape Alignment

Another way to calculate melodic similarity with a sequence alignment algorithm is to use more high level information about the piece, such as the *melodic shape*. This has the advantage over note to note alignment that small variations between tunes are not penalized very hard if the contour is roughly the same. This concept is used by Urbano (2013), who developed a global sequence alignment algorithm using B-Splines (De Boor, 1972) to model the melodic shape of a song. B-splines are curves that can be described by a set of control points (the red points in figure 4.2b to 4.2e). The resulting curve can be described as a (combination) of polynomial functions. The Melodic Shape Alignment method is the best performing implementation of the *Symbolic Melodic Similarity* MIREX task for five years (2010–2014).

Besides the Note to Note Alignment approach, we will use the ShapeH algorithm as described in (Urbano, 2013). The general idea behind the algorithm is that a musical piece is described by a set of curves, rather than a set of notes. To achieve this, each musical piece is split into *3-grams* (groups of 3 consecutive notes). These 3 notes are then projected into a 2D (time, pitch) space. All information about note duration is discarded, so the distances between the 3 notes in the time dimension are always the same. Then, a uniform B-Spline with 3 control points (which is a B-Spline of degree 3) is constructed where the 3 notes are used as control points. This spline can be represented a second degree polynomial function that results in a smooth shape. This shape represents the contour of the 3-gram.

In each 3-gram, the slope of the begin and end points of the spline are stored for comparison. Since the contour can be represented as a second degree polynomial function, we can use the derivative to identify the direction of the shape in those points. For example, the 3-gram from figure 4.2b can be represented by the tuple $(\uparrow, \downarrow)$ since the spline is going upwards at the beginning and downwards at the end. Instead of the previous penalty values, ShapeH uses a penalty function $\gamma(x_i, y_j)$ that penalizes the alignment depending on the similarity in direction of the begin and end points between the compared 3-grams:

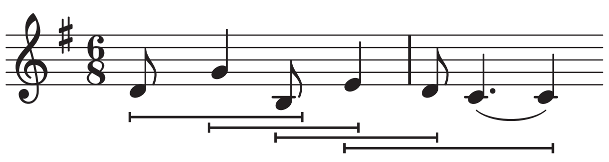
$$\gamma(x_i, y_j) = \begin{cases} \gamma_{match} & \text{if } (x_{i,0} = y_{j,0} \wedge x_{i,1} = y_{j,1}) \\ \gamma_{partial} & \text{if } (x_{i,0} = y_{j,0} \wedge x_{i,1} \neq y_{j,1}) \text{ or } (x_{i,0} \neq y_{j,0} \wedge x_{i,1} = y_{j,1}) \\ \gamma_{mismatch} & \text{if } (x_{i,0} \neq y_{j,0} \wedge x_{i,1} \neq y_{j,1}) \end{cases} \tag{4.27}$$

Here, $x_{i,0}$ is the first element of the tuple and $x_{i,1}$ is the second element. The three penalty values γ_{match}, $\gamma_{partial}$ and $\gamma_{mismatch}$ are all predefined constants, penalizes a match (the directions of both begin and end points are the same), a partial match (the direction of either the begin or end points is the same, the other one is different) or a mismatch (both directions of the begin and end points are different), respectively. For the similarity score $S(x, y)$, the concept of *inverse document frequency* is used. This means that the score of a match decreases when the matched 3-gram is more frequent in the corpus (defined by the function $f(n)$). The similarity score then looks like this:

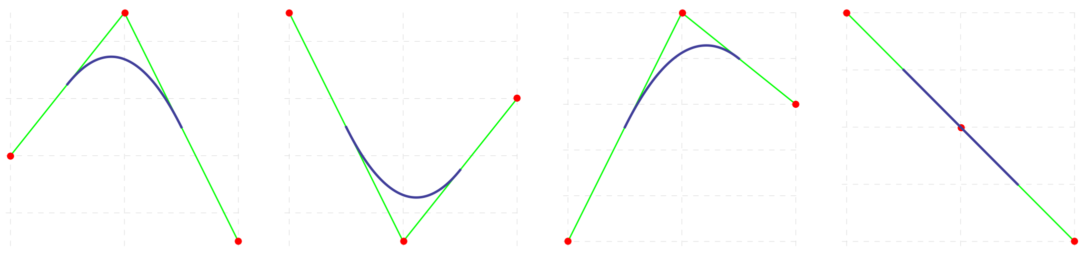
$$S(x_i, y_j) = \begin{cases} 1 - f(n) & \text{if } x_i = y_j \\ -(1 - f(n)) & \text{otherwise} \end{cases} \tag{4.28}$$

Combining the previous two equations with the matrix from equation 4.22, we get the following matrix definition of D:

$$D_{i,j} = \max \begin{cases} D_{i-1,j-1} + S(x_i, y_j) - \gamma(x_i, y_j) \\ D_{i-1,j} + S(x_i, y_j) - \gamma(x_{i-1}, y_j) \\ D_{i,j-1} + S(x_i, y_j) - \gamma(x_i, y_{j-1}) \end{cases} \tag{4.29}$$



(a) Beginning of the song of NLB074038_01.



(b) $x_0 = (\uparrow, \downarrow)$ **(c)** $x_1 = (\downarrow, \uparrow)$ **(d)** $x_2 = (\uparrow, \downarrow)$ **(e)** $x_3 = (\downarrow, \downarrow)$

Figure 4.2.: Four 3-gram spines can be constructed from the melody of (a). For each spline, we store the direction of the begin and end point as a tuple.

Instead of the regular alignment method where the value of $D_{n,m}$ is the similarity score, SHAPEH employs a hybrid approach (hence the "H" in the name of the algorithm) where not the final score from $D_{n,m}$ is used for the overall similarity score, but the largest value in the matrix. This way, only gaps and mismatches at the beginning of the song are penalized.

4.3. Conclusion

We have seen that different evaluation metrics can be used to evaluate the output of a Pattern Discovery algorithm against a set of ground truth patterns. The regular Information Retrieval evaluation measures *precision*, *recall* and F_1 *score* are too strict for this purpose, because every pattern occurrence from a discovered pattern and a ground truth pattern has to be exactly the same in order to be counted as *match*. Therefore, we described three custom measures that are also used in the *Discovery of Repeated Themes and Sections* MIREX task. These measures as based on establishment (can an algorithm at least detect one ground truth pattern occurrence), occurrences (can an algorithm detect most ground truth pattern occurrences) or a combination of both. All these metrics are less strict than the normal precision, recall and F_1 score.

To calculate the similarity between monophonic musical pieces, a *sequence* alignment algorithm has been proven to be an effective similarity measure (Hillewaere et al., 2012). We discussed the Note to Note Alignment approach by Van Kranenburg et al. (2009), which aligns two songs on a note to note basis using the pitch information of each note. The second measure we discussed is the Melodic Shape Alignment algorithm by Urbano (2013) that calculates the similarity between two songs based on the melodic contour of each group of three successive notes. These similarity measures will be used as a step in our experiment framework, which we will discuss in the next chapter.

Experiment Framework 5

In order to test what the impact of Pattern Discovery algorithms is for determining the similarity between folk songs, we will introduce an experiment framework. This framework is visualized in figure 5.1. The framework is developed in a way that every stage can be changed independently of the configuration of the other stages. In this chapter, we will give an overview and explanation of the various stages.

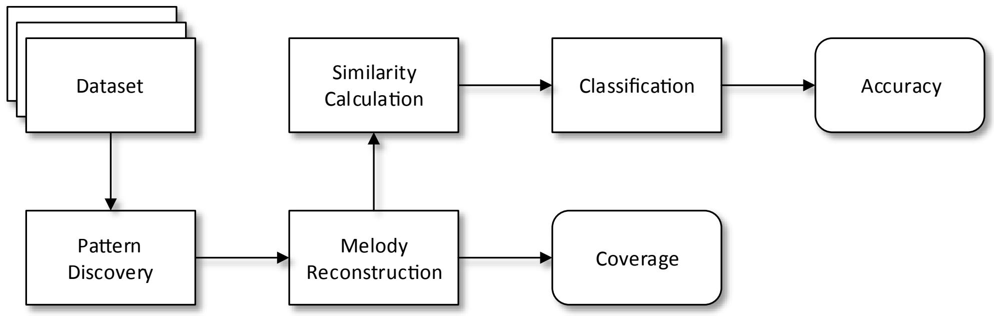


Figure 5.1.: The proposed framework for using the output of Pattern Discovery algorithms for similarity calculation between songs.

As can be seen in the flowchart in figure 5.1, the experimental process consists of a number of components which we can easily vary:

Dataset The first step is about choosing the dataset. In our experiments, we use two different folk song datasets: the dataset MTC-ANN which is part of the Meertens Tune Collections (Van Kranenburg et al., 2014), and a dataset of Irish folk tunes we created for this study, extracted from the website www.thesession.org. The tunes are converted to a standardized format before pattern discovery is performed on the pieces. More information about the datasets and the conversion is given in section 5.1.

Pattern Discovery The Pattern Discovery step can either be a human process (using pattern annotations) or an automatic operation (using Pattern Discovery algorithms). The algorithms described in chapter 3 are used in the second case. The parameter values we use for these algorithms are described in section 5.2.

Melody reconstruction After the pattern discovery, we reconstruct each song based on the patterns that are found. For instance, we reconstruct a song with only the notes of the original piece that belong to any of the discovered patterns. In this step, the ratio of notes that is still in the songs after the reconstruction (the *coverage*) is calculated. We use different strategies for this, as described in section 5.3.

Melodic similarity calculation In order to classify the tunes, we need a similarity measure that tells us how *similar* the pieces are. For this, we calculate the distance between all pairs of songs using two different sequence alignment methods: Note to Note Alignment and Melodic Shape Alignment. The precise implementation is explained in section 5.4.

Classification Based on the similarity matrix from the previous step, the actual classification is done using a k-Nearest Neighbor Classifier. In this step, the ratio of pieces that are correctly classified (the *classification accuracy*) is calculated. This is described in section 5.5.

It is important to realize that each experiment will output two important values: the *coverage* and *accuracy*. We will use these values to draw conclusions about the importance of the discovered patterns in the classification step: what happens with the accuracy when we lose information (i.e. the coverage drops)?

In the rest of this chapter, we will use the following notations for datasets, songs and notes: a dataset of songs C consists of one or more songs $s_1, s_2, \ldots s_n$. Here, n equals the number of songs in the corpus, which is often denoted as $|C|$. Both songs are from the same class if $\textsc{Class}(s_x) = \textsc{Class}(s_y)$. The length of song s equals the number of notes in the song: $|s|$.

5.1. Dataset

5.1.1. MTC-ANN

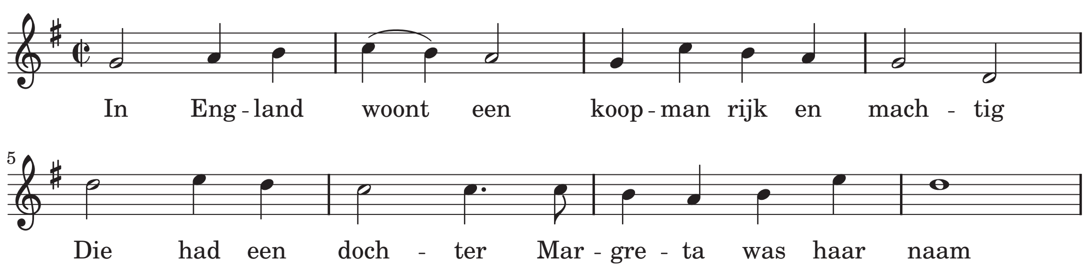
The Meertens Tune Collections (Van Kranenburg et al., 2014) are a bundle of datasets that cover a total number of 4,830 Dutch folk songs in the Humdrum `**kern` data format (Huron, 1997). Most of the songs are labeled with a tune family, a *"group of melodies that presumably descent from a single source but can vary by variation, imitation and assimilation"* (Bayard, 1950). For this project, we use a much smaller dataset of the MTC, called MTC-ANN[1]. This collection consists of 360 songs, divided into 26 tune families (see appendix A.1 for the distribution of the songs over the tune families). All songs are transcriptions of sung recordings and are therefore monophonic pieces. Each piece is transposed to the key of G. An excerpt of two songs that are part of the same tune class in the dataset is presented in figure 5.2.

The classification of these folk songs into tune families has been a manual process. We already stated that in a study performed by Volk and Van Kranenburg (2012), musicology experts were asked to annotate important motifs of the songs in MTC-ANN, of which they considered to be important for the tune family classification. In other words, they annotated motifs that they considered important for the similarity between the songs belonging to one tune family. Note that this not necessarily means that *all* repeating motifs were annotated, only the important ones. For this study, we use an updated set with annotated motifs where a number of repetitions are added that were missed in the original study.

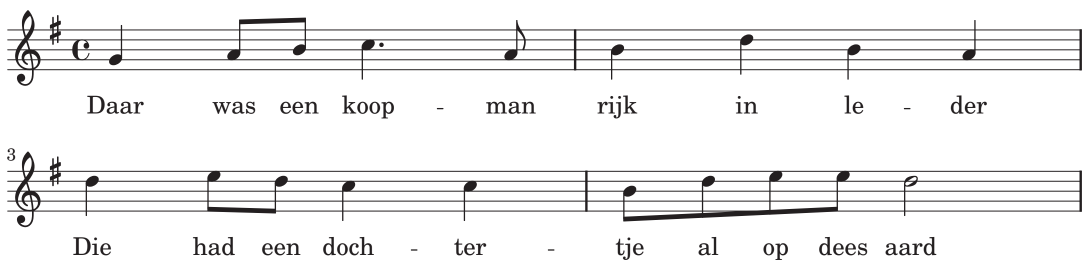
5.1.2. Irish folk songs from TheSession.org

As a second dataset, we composed a collection of traditional Irish folk songs from the website "The Session" (www.thesession.org). This sites hosts a large collection of folk tunes, as well as musical sessions and events of Irish folk song performances. Users can add their own variations or performances of a tune to the collection in ABC notation (Walshaw, 2011). This way, the site also hosts a large collection of folk tune variations. This makes this dataset very interesting together with MTC-ANN, because this set

[1]In other studies, this dataset is also called the *Annotated Corpus*


In Eng - land woont een koop - man rijk en mach - tig
5
Die had een doch - ter Mar - gre - ta was haar naam

(a) First two phrases of record 70079, strophe 1


Daar was een koop - man rijk in le - der
3
Die had een doch - ter - tje al op dees aard

(b) First two phrases of record 73788, strophe 1

Figure 5.2.: Two excerpts from MTC-ANN: both songs are part of the tune class *Daar was een koopman rijk en machtig*.

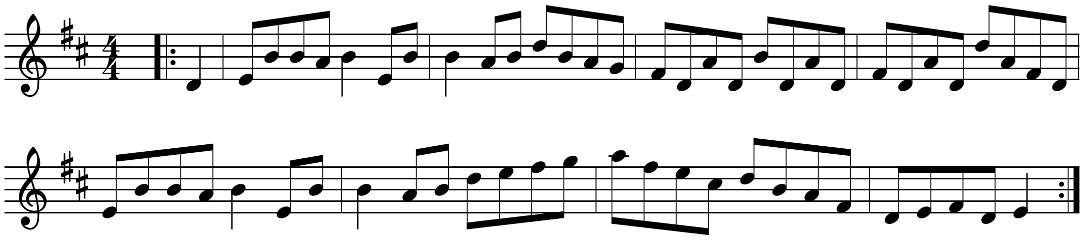
has also class labels related to variations and origin. Excerpts of two contributions are shown in figure 5.3. Further on, we will call this collection the *Irish Folk Tunes Dataset*.

Our goal is to compose a dataset that is similar in size and class distribution as MTC-ANN. From the site, all tunes with at least 10 and at most 20 annotated variations are considered as candidates for the dataset. This way, the classes will not be too small or too large compared with MTC-ANN. First, all stylistic information such as grace notes and repetition signs were removed, together with all songs with less than 20 notes (mostly containing only small textual notices about other variations in the same class). User contributions with more than one annotated variation in one contribution were split into multiple variations. Also, songs that were significantly longer than the other tunes in the same class were removed to keep the dataset balanced in terms of song length. From all classes that still contain 10 songs, 30 classes are randomly picked to form the dataset.

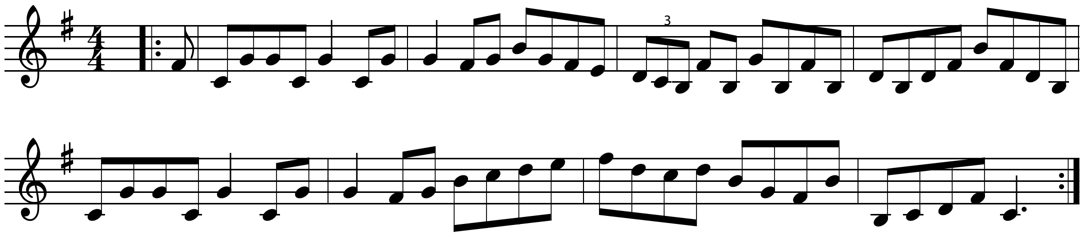
The last pruning step of the dataset involved a piecewise similarity calculation using the Note to Note Alignment algorithm. This method is thoroughly explained in section 4.2.1. From all duplicate songs, one was removed from the dataset. To ensure each song is monophonic, we only kept the highest note in cases where polyphony occurred. This technique is also known as the skyline method (Uitdenbogerd and Zobel, 1999). To meet the same specifications as MTC-ANN, all tunes are transposed to the key of G. This gives us a set of 366 songs divided into 30 classes (see appendix A.2).

5.1.3. Data format

The songs from both datasets are converted from the Humdrum `**kern` format and ABC notation to a data format that is supported by all algorithms. This way, we ensure that we do not suffer from conversion and rounding errors in the generation of different input formats. Since most algorithms are also participating in the *Discovery of Repeated Themes and Sections* MIREX task, we will use the *Collins Lisp format*, a symbolic music notation that is introduced by Collins (2014a) as input format for this MIREX task. For each note in a piece, the onset, chromatic pitch number, morphetic pitch number, duration

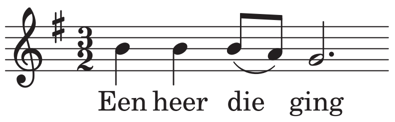


(a) *Cooley's*, tune ID 1



(b) *Cooley's*, tune ID 12342

Figure 5.3.: An example from TheSession.org: the first two phrases of two variations of the song *Cooley's*.



(a)

```
(0 71 66 1 0)
(1 71 66 1 0)
(2 71 66 1/2 0)
(5/2 69 65 1/2 0)
(3 67 64 3 0)
...
```

(b)

Figure 5.4.: Example of a song (a) in Collins Lisp Format (b).

and voice part are saved. Since only monophonic music is used in this study, the voice part will always be one. All rational duration and onset values (see section 2.2.3) are expressed as fractions to prevent initial rounding issues (see figure 5.4). The Python toolkit `music21` (Cuthbert and Ariza, 2010) is used for the conversion.

5.2. Pattern Discovery

The Pattern Discovery algorithms described in chapter 3 are used for the pattern discovery step in this experiment framework. We have seen that many algorithms have a number of parameters. To make sure we do not favor any of the algorithms and to get an insight in the optimal parameter configuration, we run each algorithm with all possible combinations of parameter values, presented in table 5.1. Since the SIA and SIATEC algorithms without additional pattern filtering return so many patterns, it makes no sense to use these algorithms as compression method. We will therefore discard these algorithms in our experiments, and add a Compactness Trawler to the SIATEC algorithm instead (SIACTTEC). We also added the compactness trawler to the COSIATEC, which is called COSIACTTEC. Recall that we had two different Pattern Discovery tasks, dependent on whether the patterns are discovered in a single song (Song Pattern Discovery) or in a group of songs (Class Pattern Discovery).

All parameters are from the original implementations, except for the enforcement of a minimum number

		Parameters	
Algorithm	**Music dimensions**	**Description**	**Value**
SIACTTEC	Morphetic pitch, onset	Minimum pattern size	$\{3, 4, 5, 6, 7\}$
		Minimum compactness	$\{0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$
COSIACTTEC	Morphetic pitch, onset	Minimum pattern size	$\{3, 4, 5, 6, 7\}$
		Minimum compactness	$\{0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$
MotivesExtractor	Chromatic pitch, onset, duration	Minimum pattern size	$\{3, 4, 5, 6, 7\}$
		Scoring theshold	$\{0.1, 0.3, 0.5, 0.7\}$
		Clustering tolerance	$\{0, 2, 4, 6, 8\}$
		Path tracing tolerance	$\{2, 3, 4\}$
PatMinr	Chromatic pitch (interval), morphetic pitch (interval)	Minimum pattern size	$\{3, 4, 5, 6, 7\}$
SIARCT-CFP	Morphetic pitch, onset	Minimum pattern size	$\{3, 4, 5, 6, 7\}$
		Minimum compactness	$\{0.5, 0.7, 0.9, 1.0\}$
		Minimum exactness	$\{0.3, 0.5, 0.7, 0.9\}$
		Min. similarity in category	$\{0.7, 0.9, 1.0\}$
MGDP	Duration, pitch class, pitch class interval, morphetic pitch interval	Minimum relative support	$\{0.5, 0.7, 0.9\}$
		Distinctiveness threshold	$\{10, 20, 26, 30$[2]$, 40\}$
		Minimum pattern length	$\{3, 4, 5, 6, 7\}$
		Minimum patterns per class[3]	$\{0, 5, 10\}$
		Enforce minimum patterns	$\{\text{True}, \text{False}\}$

Table 5.1.: List of algorithms used in the experiment framework, with their possible parameter values.

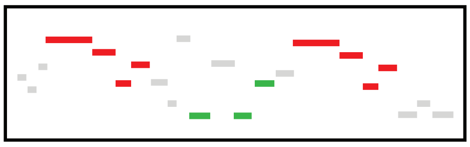
of patterns per class in the MGDP algorithm. Since the MDGP algorithm can discover patterns in groups of songs only, this method will not be used in the Song Pattern Discovery task. Because the classes are very different (i.e. the songs in one class are more repetitive than songs in other classes), it can happen that for some distinctiveness thresholds many classes do not contain useful patterns, whereas for other classes a large number of patterns are returned. To solve this, we introduce two extra parameters for the MGDP algorithm, that enables the algorithm to enforce a minimum number of returned patterns by changing the other parameters. When *enforce minimum patterns* is set to *True*, the algorithm will decrease each round the distinctiveness threshold by one until the defined minimum number of patterns is reached or until the distinctiveness threshold is 0. For all algorithms except the MGDP algorithm, the original implementations by the corresponding authors are used, with slight modifications to fit this particular evaluation framework.

Besides a standardized input, all algorithms also use the same output format. Every pattern consists of a list of occurrences and each occurrence is stored as a list of tuples, containing onset and pitch information.

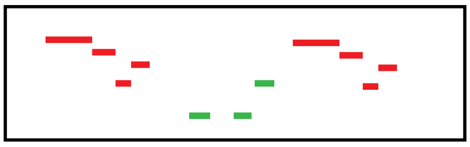
5.3. Melody Reconstruction

After the Pattern Discovery step, we have acquired a set of discovered patterns for each song in the dataset. Like we stated in the previous section, each Pattern Discovery algorithm outputs a discovered pattern as a set of occurrences. For each occurrence, we have a set of (ontime, pitch) pairs that describe the notes in the occurrence.

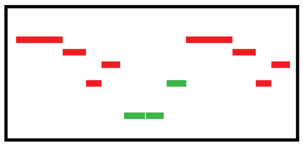
[2]26 for MTC-ANN, 30 for the Irish Folk Tunes Dataset
[3]Only if *Enforce minimum patterns* is set to *true*



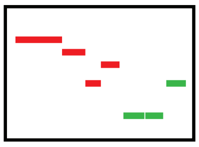
(a) Representation of a song where two patterns are discovered (red and green). The gray notes will be removed, since they are not part of a pattern.



(b) Onset Preserved Reconstruction



(c) Gapless Reconstruction



(d) First Pattern Occurrence Reconstruction

Figure 5.5.: The resulting pieces after the reconstruction procedure from the first piece of figure 5.5a. From this figure we can see that the original song length is 22 notes, so the coverage after the note filtering step is $11/22 = 0.5$ for figure 5.5b and 5.5c and $7/22 \approx 0.32$ for figure 5.5d.

In the Melody Reconstruction step, we actually compress the songs using a *lossy compression* strategy. Recall that lossy compression means that we throw away information without the possibility to restore the original song. We will *only* use the information contained in the pattern occurrences to reconstruct each song. This means that a reconstructed song only contain notes that are present in at least one discovered pattern. We will now introduce three different melody reconstruction approaches we will use in the Evaluation Framework, also visualized in figure 5.5:

Onset Preserved Reconstruction This is the most simple strategy: all notes not belonging to any pattern are replaced by a rest of the same duration. This way, the onset time of all notes is preserved in the reconstructed song.

Gapless Reconstruction Instead of leaving gaps in parts of the pieces where notes are removed, we shift all succeeding notes to the onset time of the removed note. This way, all notes that are present in a pattern are "stitched together" without any gaps caused by the note removal. Gaps caused by a rest in a pattern will not be removed.

First Pattern Occurrence Reconstruction To recall the problems with the output of Pattern Discovery algorithms in section 1.3, the number of discovered patterns is often very high with respect to the length of the song. This can eventually cause the two previous strategies to give uninteresting results, because nearly all notes can be in any pattern. This leads to the resulting filtered song being nearly the same as the original song. To prevent this, only the first occurrence of each repeated pattern is saved instead of all occurrences. For the note removal, the same strategy is used as the Gapless Reconstruction method, omitting any gaps caused by the melody reconstruction.

Let $\textsc{Reconstruct}(s)$ be the function that reconstructs song s using one of the approaches we introduced. Doing this for all songs in the dataset will give us a new collection of songs C_{recons}, such that all songs $s \in C_{recons}$ contain only notes that are present in at least one discovered pattern

$$C_{recons} = \{\textsc{Reconstruct}(s) \mid \in C \land |\textsc{Reconstruct}(s)| > 0\} \tag{5.1}$$

From equation 5.1, we see that songs that contain no more notes after the melody reconstruction step are discarded and will not be added to the set C_{recons}. This means that a song of which no patterns are found will not be in C_{recons}. Therefore, the size of C does not have to be the same as C_{recons}. With this information, we can now define the *coverage* of a song s as:

$$\textsc{Coverage}(s) = \frac{|\textsc{Reconstruct}(s)|}{|s|} \tag{5.2}$$

In other words, the *coverage* of a song represents the fraction of notes that are covering the original song. Since we are interested in the evaluation of Pattern Discovery on a complete dataset rather than a single song, we will define the resulting coverage of the dataset as the average of song coverages:

$$\textsc{Coverage}(C, |C_{recons}|) = \frac{1}{C_{recons}} \cdot \left(\sum_{\{s \in C \,|\, \textsc{Reconstruct}(s) \in C_{recons}\}} \textsc{Coverage}(s) \right) \tag{5.3}$$

This way, the coverage gives us a measure of how much information is left in the song or the dataset.

5.4. Similarity Calculation

For the melodic similarity calculation, we will use the two sequence alignment algorithms as described in section 4.2. For both algorithms, we use the same penalty values as the authors proposed in their original implementation (Van Kranenburg et al., 2013; Urbano, 2013) because they already yield good results:

Note to Note Alignment

- Gap opening penalty $\gamma_{open} = 0.8$
- Gap extension penalty $\gamma_{ext} = 0.2$

Melodic Shape Alignment

- Match penalty $\gamma_{match} = 0.5$
- Partial match penalty $\gamma_{partial} = 1$
- Mismatch penalty $\gamma_{mismatch} = 8$

This results in a similarity matrix filled with distance values between every pair of songs. The smaller the distance, the more similar these songs are.

5.5. Classification

All tunes in the corpus are piecewise compared using one of the sequence alignment methods described in section 4.2. For each piece, the distances to all other pieces are ranked in such a way that the top N results contain the N closest (most similar) pieces. We then use the k-Nearest-Neighbor classification rule (Cover and Hart, 1967) to assign each song in the corpus to the class of its closest neighbor. Because we choose $k = 1$, it is sufficient to only look at the closest neighbor. However, it can happen that there are more than one "closest" neighbors when multiple tunes are at the same closest distance from the queried song. In this case, the majority class of the closest neighbors wins. In case of a tie, the class of a randomly selected closest neighbor is assigned.

To calculate the classification on the set of compressed songs C_{recons}, we use a cross validation technique called *leave-one-out* cross-validation. This means that we select a piece $s \in C_{recons}$ and assign it to the class of its nearest neighbor. Since we know the *true* class label of s from the original dataset, we can immediately check whether the classification is correct (i.e. it is a true positive (TP)) or not:

$$\text{TP}(s) = \begin{cases} 1 & \text{if } \textsc{Class}(s) = \textsc{Class}(\textsc{NearestNeighbor}(s)) \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

This procedure is repeated for all songs in C_{recons}. We can now define the classification accuracy of the output as follows:

$$\text{Accuracy} = \frac{\sum_{s \in C_{recons}} \text{TP}(s)}{|C|} \tag{5.5}$$

In other words, the classification accuracy is the ratio of songs that is correctly classified. To prevent a too positive accuracy score, we divide by $|C|$ instead of $|C_{recons}|$. This means that all songs $s \in (C \setminus C_{recons})$ (i.e. all songs in which no pattern is found) are treated as wrongly classified.

After this step, both the coverage and classification accuracy are known for the chosen algorithms and components of the framework. Because all the steps in the framework can be easily replaced by other algorithms and methods, the framework is very suitable for comparing different parameter configurations or algorithms. When changing a method in one of the stages, the results of all preceding stages does not need to be calculated again if they are already calculated once. This means that only the changed stage and all following stages need to be calculated in order to retrieve the coverage and classification accuracy. In the next chapter, we describe how this framework is used for the various experiments we will perform.

Method 6

In the previous chapter, we have discussed the experiment framework that combines the Pattern Discovery algorithms described in chapter 3 with the similarity calculation methods from section 4.2. In this chapter, we will precisely describe the experiments we have performed, mostly based on this framework. The output of the experiments will contribute to the answering of the research questions we have stated in the Introduction.

In order to draw conclusions about the effect of using discovered patterns as compression method, we perform a number of experiments to see how compression affects the classification accuracy. We therefore calculate the classification accuracy on the uncompressed folk songs (section 6.1), on songs compressed by naive lossy compression strategies (section 6.2), on song compressed using the annotated motifs from MTC-ANN (section 6.3) and finally on the folk songs compressed by using Pattern Discovery algorithms (both Song Pattern Discovery and Class Pattern Discovery) in section 6.4. We expect that none of the compression techniques lead to a higher accuracy than the classification on the uncompressed songs since the results of Van Kranenburg et al. (2009) show that an alignment algorithm already results in a very high classification accuracy. However, we are interested to see if we can get comparable results by using compression. Because of the considered importance of repeated patterns for tune classification, we expect that the Pattern Discovery algorithms and annotated motifs result in a higher classification accuracy than the naive lossy compression methods for the same coverage value.

To evaluate the effect of the different possible methods in the experiment framework we proposed in chapter 5, we will also perform a number of experiments that change one of the steps in the framework. In section 6.5, we will compare different reconstruction methods to see what the effect is of leaving gaps in the compressed songs, as well as removing all occurrences but one of each pattern. The next experiment in section 6.6 varies the sequence alignment algorithm we use for determining the similarity between two songs, because we want to know if another alignment algorithm would also perform well in determining the similarity between folk songs. In this light, we will compare the results of all experiments performed on MTC-ANN with another dataset, the Irish Folk Tunes Dataset, in section 6.7. Using these results, we can conclude if the results so far are specific for MTC-ANN, or that this is at least characteristic for multiple folk song datasets. In section 6.8, we will compare the discovered and annotated patterns more directly using the evaluation metrics defined in section 4.1.

6.1. Determining Similarity on Uncompressed Songs

This experiment is used as a baseline, where we use the framework presented in figure 6.1. Compared to the evaluation framework from figure 5.1, we will calculate the similarity between the complete pieces. This means no Pattern Discovery is performed in this experiment. Note that, since we do not compress the songs, this experiment will not return a coverage value (because the coverage is always 1.0 is this case).

We will run this experiment on both datasets (MTC-ANN and the Irish Folk Tunes Dataset) and on both similarity calculation algorithms (Note to Note Alignment and Melodic Shape Alignment).

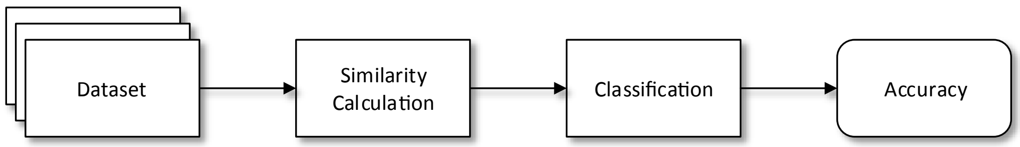


Figure 6.1.: Evaluation framework without Pattern Discovery: the complete songs are used for the similarity calculation.

6.2. Determining Similarity using Naive Lossy Compression Strategies

In order to be able to draw conclusions from the output of the experiment framework described in this chapter, we have to perform a number of experiments *without* the use of discovered repeated patterns. We will therefore introduce a number of naive lossy compression strategies.

6.2.1. Selecting a Consecutive Subsequence

To compare the results of the alignment of short concatenated segments from Pattern Discovery algorithms with longer consecutive sequences, we perform 3 different experiments, where we select a predefined number of consecutive notes in the note filtering step, instead of using patterns. This method is based on the observations of Xia et al. (2014), that the first and last n notes is a good feature for determining similarity (in their case, n was 200 because they used it to compress very large songs). The three approaches are, for each piece in the corpus:

- Select the *first* n notes
- Select the *last* n notes
- Select the *middle* n notes

Here, "selecting" means that these notes are kept and all other notes are removed. So, in the first approach the last $|s| - n$ notes are removed from each song (where $|s|$ is the number of notes in piece s). By repeating the experiments for all values of n in the range $[2, \max_{s \in C} |s|]$, we can easily calculate the relation between coverage and accuracy. In the case that $n > |s|$ for a particular piece s, we keep all notes in s. The pieces are classified using this selection of notes.

6.2.2. Random Note Selection

We can randomly select (groups of) notes before the note filtering takes place. In each piece, we randomly select a number of notes until a predefined coverage ratio is reached. We can vary the desired coverage ratio, as well as the length l of each random selection. This means that, after a note is randomly selected, all $l - 1$ notes that follow directly are also selected. If one of these notes is already present in another selection, this selection is rejected and a new note is selected. For each coverage value in the range $[0.1, 0.9]$, we average over 20 samples. In each sample, the randomly selected notes are used as patterns for the classification. The coverage values are chosen with intervals of 0.1 and $l = 3$.

6.2.3. Random Pattern Selection

A downside of the previous method is that the note selection is completely random and does not replicate the effect of repeating patterns. To fix this, we generate a mask $M = \{0, 0, \ldots, 0\}$ with the length of the largest song in the corpus. Then, the same random selection as described above is used to populate the mask with ones, until a coverage ratio is reached. Then, this mask is used to select notes in all pieces of the corpus. This way, the position of the note selections is the same for all pieces in the corpus.

Suppose we have a mask $M = \{0, 1, 1, 1, 0, 0, 0, 1, 1\}$ and two songs $s_1 = \{\mathrm{C}_4, \mathrm{E}_4, \mathrm{G}_4, \mathrm{C}_5, \mathrm{G}_4, \mathrm{E}_4, \mathrm{C}_4\}$ and $s_2 = \{\mathrm{G}_3, \mathrm{A}_3, \mathrm{B}_3, \mathrm{C}_4, \mathrm{D}_4, \mathrm{E}_4, \mathrm{F\#}_4, \mathrm{G}_4, \mathrm{G}_3\}$. After applying M to both songs, we get

$$\begin{aligned} M(s_1) &= \{\mathrm{E}_4, \mathrm{G}_4, \mathrm{C}_5, \} \\ M(s_2) &= \{\mathrm{A}_3, \mathrm{B}_3, \mathrm{C}_4, \mathrm{G}_4, \mathrm{G}_3\} \end{aligned} \tag{6.1}$$

The resulting coverage for this mask is 5/9. This experiment is performed 200 times for random coverage values. The resulting patterns from each run are used for classifying the songs. Since the coverage is chosen to be random in each run, the coverage values are binned into intervals of 0.05. The values in each bin are averaged to get one accuracy value for each coverage bin.

6.3. Determining Similarity using Annotated Motifs

In this experiment, we will test the hypothesis that repeated patterns are important for classifying songs in tune classes. Compared to the evaluation framework, we skip the Pattern Discovery stage and directly feed the annotated motifs from MTC-ANN to the Melodic Reconstruction stage (see figure 6.2). We will vary both the melodic reconstruction methods (Onset Preserved Reconstruction, Gapless Reconstruction, First Pattern Occurrence Reconstruction) and the similarity calculation algorithms (Note to Note Alignment and Melodic Shape Alignment), to see which of the combinations performs best.

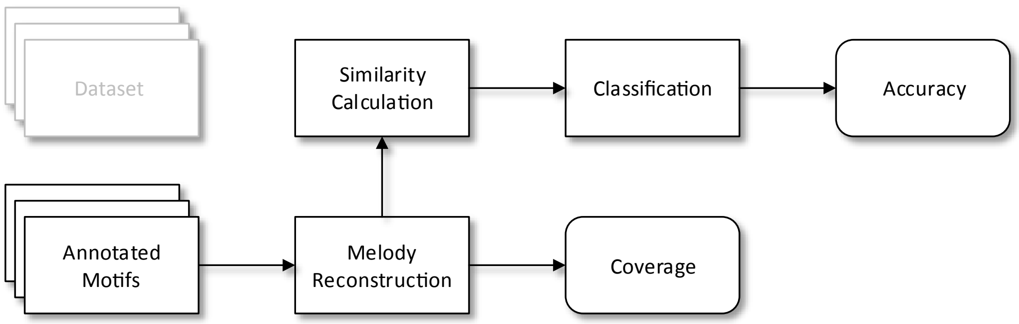


Figure 6.2.: Evaluation framework using Annotated Motifs.

6.4. Determining Similarity using Pattern Discovery Algorithms

We have defined two different Pattern Discovery tasks: Song Pattern Discovery and Class Pattern Discovery. Recall that the Song Pattern Discovery task involves the discovery of repeated patterns withing a single song and the Class Pattern Discovery task involves the discovery of repeated patterns in a collection of songs. This collection is in our case a tune class.

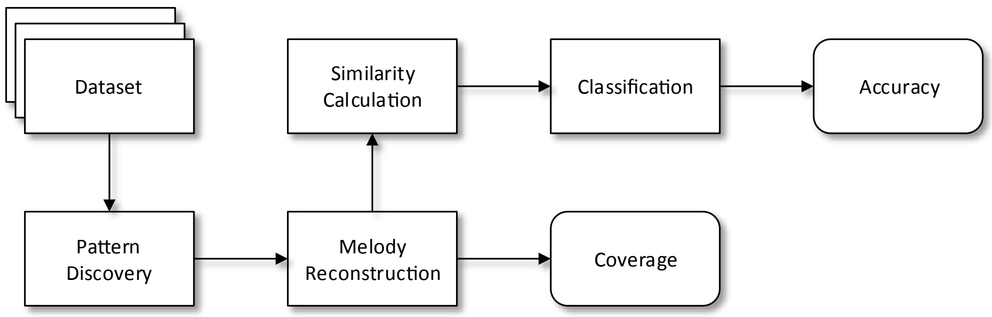


Figure 6.3.: Experiment setup for Song Pattern Discovery.

6.4.1. Song Pattern Discovery

This experiment is the first that makes full use of the complete framework introduced in chapter 5. The precise flowchart for this task is illustrated in figure 6.3. In order limit the number of variables in each experiment, we use one dataset, one reconstruction method and one alignment algorithm. In later experiments, we will compare these variables separately. The songs from MTC-ANN are used as input for five different Pattern Discovery algorithms, namely:

- COSIACTTEC
- MotivesExtractor
- PatMinr
- SIARCT-CFP
- SIACTTEC

The framework is used for all parameter configurations from table 5.1. In this experiment, patterns are found that repeat *inside* a single song. This will test the hypothesis that knowing one song from a tune class is enough to classify all other songs belonging to the same class. In order to keep the number of variables as small as possible in each experiment, we will use Note to Note Alignment for the similarity calculation and Gapless Reconstruction as melody reconstruction method.

6.4.2. Class Pattern Discovery

In contrast to the previous experiment, we will now use the Pattern Discovery algorithms to find repetitions inside *tune classes*. Because most algorithms we use are specifically designed for pattern discovery inside single songs, the framework is slightly adapted for this experiment (see figure 6.4). Before the Pattern Discovery step, all songs from the same tune class are concatenated before being analyzed by the algorithm. After the discovery, the songs are split into individual pieces again, discarding pattern occurrences that lie on the boundary of two adjacent songs in the concatenated piece (see figure 6.5).

We will run this experiment with the following algorithms:

- COSIACTTEC
- MGDP
- MotivesExtractor

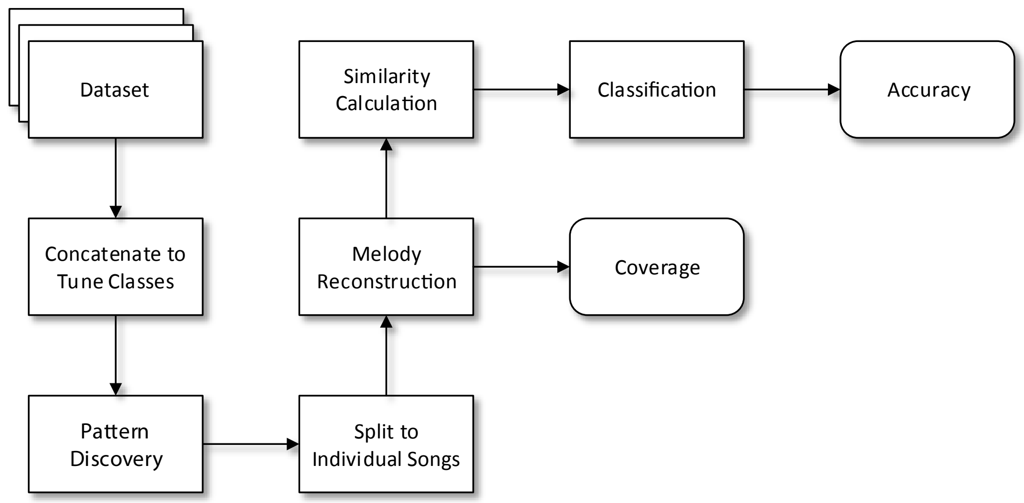


Figure 6.4.: Experiment setup for Song Pattern Discovery. All the songs belonging to the same classes are concatenated before the pattern discovery step.

- PatMinr
- SIACTTEC

Since the MGDP algorithm cannot be used for Pattern Discovery within songs, this method is only used in the Class Pattern Discovery experiment. Because this is the only algorithm that is specifically developed for Class Pattern Discovery, the concatenation and splitting step can be skipped (figure 6.4). Instead, a *corpus* (containing the pieces of the class of which the patterns need to be discovered) and an *anti-corpus* (all the other pieces from the dataset) need to be supplied (see also section 3.1.1).

The SIARCT-CFP algorithm is only used in the Song Pattern Discovery task, because the algorithm turned out to be too slow on the Class Pattern Discovery task to test a representative number of parameter values. Like the previous experiment, Gapless Reconstruction and Note to Note Alignment are used.

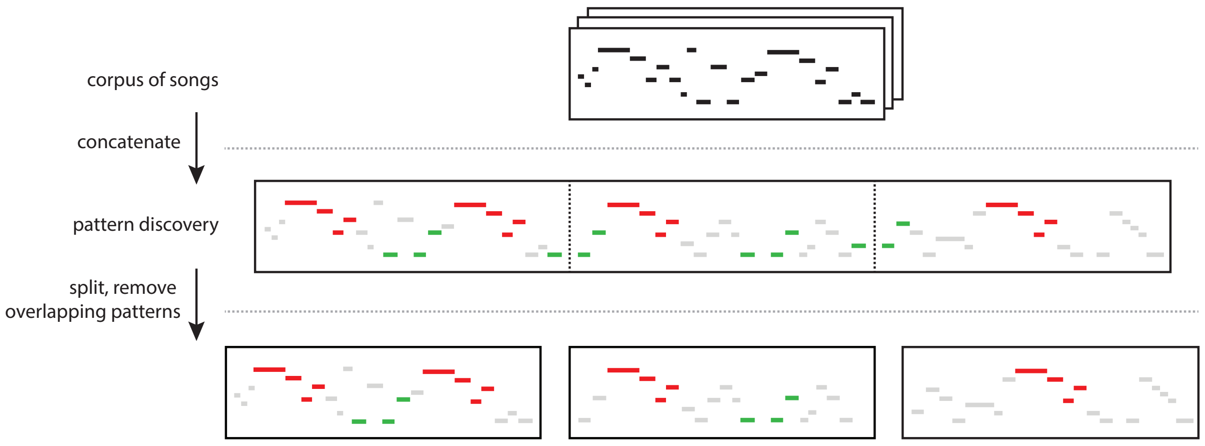


Figure 6.5.: Schematic overview of the procedure for Class Pattern Discovery using algorithms only designed for Song Pattern Discovery. All songs in each tune class are concatenated and fed as input to the discovery algorithms. After the pattern discovery, all overlapping patterns (in this example the second and fourth green pattern) are removed and the concatenated tunes are being split back into the original pieces.

6.5. Comparison of Reconstruction Methods

In this experiment, we want to measure the effect of using different reconstruction methods. Recall from section 5.3 that we have introduced three reconstruction methods:

- Onset Preserved Reconstruction
- Gapless Reconstruction
- First Pattern Occurrence Reconstruction

We will vary the reconstruction method, while keeping the other variables in the framework constant. We therefore test the different reconstruction mode on the dataset MTC-ANN, using Note to Note Alignment. We will compare the results of both Song Pattern Discovery and Class Pattern Discovery with all the algorithms that are suitable for these tasks (section 5.2).

6.6. Comparison of Similarity Calculation methods

From the work of Van Kranenburg et al. (2013), we know that using Note to Note Alignment for determining the similarity between folk songs is a good strategy. To verify whether other alignment methods will give better or worse results, we will use the Melodic Shape Alignment algorithm in the Similarity Calculation step, instead of the Note to Note Alignment algorithm we used in the previous experiments. In other words, we will vary the following methods:

- Note to Note Alignment
- Melodic Shape Alignment

Like in the previous experiment, we will perform this experiment for both Pattern Discovery tasks (Song Pattern Discovery and Class Pattern Discovery) to see which effect both alignment algorithms have on both tasks.

6.7. Comparison of the Folk Song Datasets

All experiments up to now are performed on MTC-ANN. We know from previous research that sequential alignment works very well for tune classification. On top of that, we also know that patterns play are important for the classification of the folk songs of MTC-ANN. To test if these observations are limited to this particular dataset, we will compare the output of the experiments described in section 6.4.1 and 6.4.2 with the same Pattern Discovery experiments on the Irish Folk Tunes Dataset. Like the Dutch folk songs, the Irish folk songs in this dataset are also grouped into classes based on similarity and variation.

6.8. Comparing the Pattern Discovery Output with the Annotated Motifs

To test the correspondence between the Annotated Motifs and the output of the Pattern Discovery algorithms we use in our experiments, we will evaluate the discovered patterns from the algorithms the same way as in the *Discovery of Repeated Themes and Sections* MIREX task. This means that we use the evaluation metrics from the MIREX task, treating the motif annotations from MTC-ANN as Ground Truth. We will evaluate the output using the metrics described in section 4.1.2:

- Establishment precision, establishment recall and establishment F_1-score
- Occurrence precision, occurrence recall and occurrence F_1 score
- Three layer precision, three-layer recall and three-layer F_1 score

We will not use the "normal" precision, recall and F_1 score in this experiment, since these measures are too strict for the purpose of the evaluation. Instead, we use these three sets of evaluation metrics, that measure the ability of an algorithm to discover (establish) a pattern, to discover all occurrences and to do both, respectively. For the inexactness threshold c, we choose $c = 0.75$ to also allow variations in the matched occurrences of the Occurrence precision, recall and F_1 score.

We have to support some special cases when either the set of ground truth patterns G or the set of the discovered patterns D are empty for a song. We will therefore define the following rules, regarding the precision, recall and F_1 score for all three evaluation metrics:

- If $D = \emptyset$ and $G = \emptyset$, the precision, recall and F_1 score for all evaluation metrics will be 1.
- If either $D = \emptyset$ and $G \neq \emptyset$ or $D \neq \emptyset$ and $G = \emptyset$, the precision, recall and F_1 score for all evaluation metrics will be 0.

Results 7

In this chapter, we will show the results of the experiments described in the previous chapter. In section 7.1, we will present the results of classification on the entire songs without compression by the discovery of repeated patterns. Then, in section 7.2, the results of naive approaches to compress songs are presented. In section 7.3 we show the performance of the classification and compression using patterns annotated by experts. We will then give the results of using Pattern Discovery algorithms for the tune classification task. In section 7.4, we present the accuracy and coverage of the Pattern Discovery algorithms in both the discovery task inside songs and inside classes. In section 7.5 and 7.6 , we compare the results between different reconstruction methods and similarity calculation algorithms, respectively. In section 7.7, we present the results on the Irish Folk Tunes Dataset. Finally, the discovered patterns from MTC-ANN are compared with the Annotated Motifs using the evaluation metrics introduced in the *Discovery of Repeated Themes and Sections* MIREX task in section 7.8.

7.1. Determining Similarity on Uncompressed Songs

In order to see what the effect of compression is on the classification accuracy, we will first classify the folk songs of both datasets without compression (section 6.1). Recall that we measure the classification accuracy by the ratio of songs that is classified in the right tune class, according to the musicologists who classified these songs manually. We will also compare the influence of Note to Note Alignment and Melodic Shape Alignment on the classification accuracy. The results of this experiment are presented in table 7.1.

Dataset	**Accuracy**	
	Note to Note	**Melodic Shape**
MTC-ANN	0.92	0.87
Irish Folk Song Dataset	0.96	0.92

Table 7.1.: Classification accuracy of the entire songs from both datasets.

We see that both alignment methods lead to a high classification accuracy for both datasets. Although the differences are small, Note to Note Alignment performs better than Melodic Shape Alignment for both datasets. We can also see that the songs from both datasets can be successfully compared using an alignment algorithm, with the worst classification accuracy of 0.87 using Melodic Shape Alignment on MTC-ANN. The result of Note to Note Alignment on MTC-ANN is in correspondence with the findings of Van Kranenburg et al. (2013) using Note to Note Alignment on pitch interval sequences. These results will give a good reference for all other experiments, since we want to know if we could get a comparable classification accuracy with less information by compression. At least, we are interested what influence compression has on the classification accuracy. On top of that, we are interested in compression methods that lead to a *low* coverage (which corresponds to a high compression) and a *high* classification accuracy. This way, only the most important information for classification about each song is left.

7.2. Determining Similarity using Naive Lossy Compression Strategies

In section 6.2, we have introduced a number of naive, lossy compression methods. Compared with *lossless* compression, we cannot completely reconstruct a compressed song back to the original. Because we throw out (what we expect to be) unimportant information, this process is irreversible. We will compare these naive compression strategies with the output of the Pattern Discovery algorithms in the next section. We introduced the following methods:

- Selecting consecutive subsequences (first, middle, last *n* notes)
- Randomly selecting notes
- Randomly selecting patterns

The results are presented in figure 7.1. For clarification purposes, the classification accuracy on the uncompressed songs is represented by a line instead of a single point. Recall that the coverage represents the ratio of notes from the compressed song that *covers* the original song. In the case of the classification on the uncompressed songs, the coverage will be 1.0 since all notes are used for the similarity calculation and classification. The raw data of these experiments can be found in appendix D.1, D.2 and D.3, respectively.

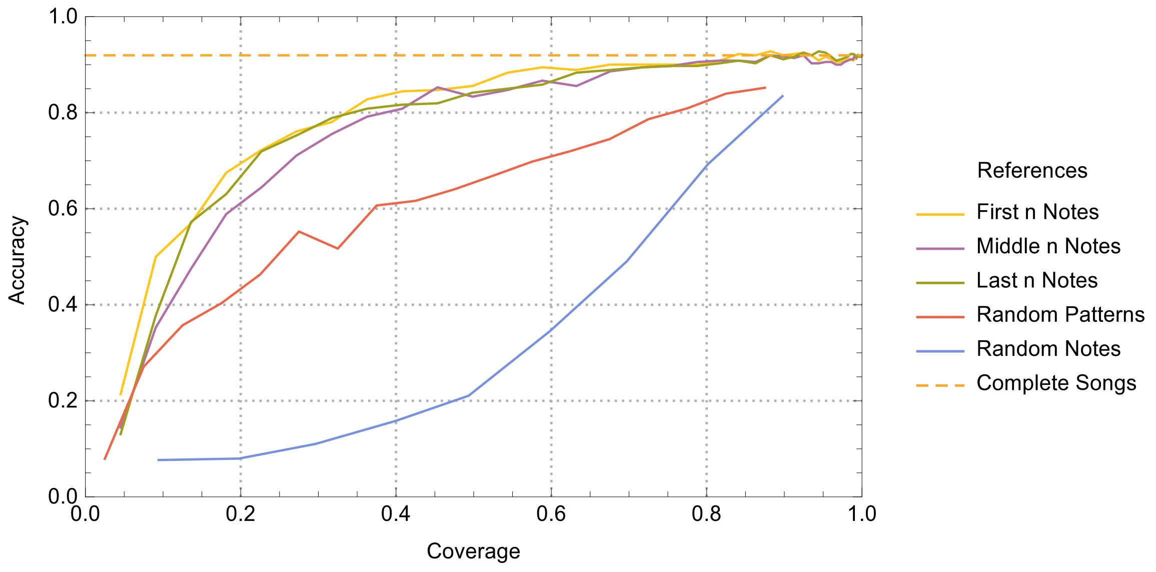


Figure 7.1.: Results of the reference experiments on MTC-ANN using Note to Note Alignment

We see that the Random Notes selection leads to the lowest accuracy for all compression methods. We suspect that this is caused by the fact that this method creates many small gaps in the songs, which affects the performance of the sequence alignment algorithm. If these algorithms cannot calculate sensible similarity scores between the songs, the resulting classification accuracy will also be low. When random patterns are selected (i.e. larger sequences of notes), we see that the accuracy is already much better compared to the Random Notes selection method.

The last three methods select a large consecutive sequence of notes starting at the beginning, middle and end, respectively. We will call these experiments the selection of the first, middle and last n notes. Here, $2 \leq n \leq \max_{s \in C} |s|$ where C represents the set of songs in the dataset. We see that these three approaches lead to the highest accuracy. Unlike the Random Note selection, the accuracy increases fast at a low coverage. The selection of the last n notes even outperforms the classification on the complete songs, although the differences are very small.

Conclusion

From the results in section 7.1, we have seen that our classification Framework using sequence alignment to calculate the similarity between the songs already yields high classification accuracy results. The efforts of the naive lossy compression strategies in section 7.2 have shown that the selection of large consecutive sequences is a good approach for compression. Even for coverage values around 0.6, the accuracy is already near the classification accuracy on the uncompressed songs. The random selection of notes and patterns yield worse results, although for high coverage values the loss of information is small enough for the alignment algorithms to successfully calculate the similarity scores between the songs.

7.3. Determining Similarity using Annotated Motifs

Before we compare the output of the tested Pattern Discovery algorithms, we will first look at the performance of the *Annotated Motifs* from MTC-ANN as compression method. In other words, we want to know how many songs are classified correctly, when we reconstruct each song with only the notes that are in one or more annotated patterns. Unlike the previous results, this will directly give us notion about the effect of using repeated patterns as compression method. On top of that, we can test the claim that the annotated patterns are expected to be important in a classification task and link this to the hypothesis that using these patterns for compression, the most important information in a song is still present for classification. Since we know that these repeated patterns are annotated because they play an important role in the classification of the songs into tune classes, we can use this set of patterns as a baseline for the output of the Pattern Discovery algorithms. Of the 360 songs in the dataset, 354 of them have one or more annotated motifs. According to the maintainers of the dataset, the experts who were in charge of annotating the motifs did not find any motif in these six songs that were important for the tune classification. On average, there are 3.4 repeated patterns annotated for each song.

Since we only have a set of annotated patterns from MTC-ANN, we will only look at this dataset in this section. The results of the tune classification using the annotated motifs of MTC-ANN are presented in table 7.2. The first thing to notice is that the resulting classification accuracy using Onset Preserved Reconstruction/Gapless Reconstruction and Note to Note Alignment (0.89) is only slightly worse than the performance on the uncompressed songs. However, on average only 40% of the notes of the uncompressed songs are used in the classification task. This means that after a large compression, the similarity between the songs can still be determined successfully.

Reconstruction method	**Coverage**	**Accuracy**	
		Note to Note	**Melodic Shape**
Onset Preserved Reconstruction	0.40	0.89	0.81
Gapless Reconstruction	0.40	0.89	0.81
First Pattern Occurrence Reconstruction	0.30	0.86	0.78

Table 7.2.: Classification accuracy and coverage of compressed songs from MTC-ANN by the annotated motifs

We see the same differences in accuracy between the two alignment methods as in table 7.1: the use of Melodic Shape Alignment leads to a slightly lower accuracy than using the Note to Note Alignment approach. Onset Preserved Reconstruction and Gapless Reconstruction yield higher accuracy values than First Pattern Occurrence Reconstruction, although the differences are only 0.03 for both alignment algorithms. Compared with the lossy compression methods (see previous section), using the annotated motifs for compression works better than any of the lossy compression methods we proposed.

As expected, the resulting coverage when using the First Pattern Occurrence Reconstruction method is lower than for the other two reconstruction approaches: 0.30. This is expected because only the first occurrence of each pattern is used in the reconstruction of a song. We already noted that this influenced the classification accuracy, but the decrease in coverage (9 percentage points) is larger than the decrease in accuracy (3 percentage points) in the case of Note to Note Alignment. We see that the coverage and accuracy for the Onset Preserved Reconstruction and Gapless Reconstruction methods are identical; we suspect that both the Note to Note Alignment and the Melodic Shape Alignment algorithms ignore gaps in the songs for the sequence alignment.

To conclude, compared with the classification accuracy on the uncompressed songs (table 7.1), the accuracy has only slightly decreased using the annotated motifs for similarity and compression, with only a coverage of 40%. Using the First Pattern Occurrence Reconstruction, the reconstructed songs even cover only 30% of the original songs but also with a cost of a slightly lower classification accuracy.

7.4. Determining Similarity using Pattern Discovery Algorithms

We will now look at the accuracy and coverage of the classification experiments based on the *discovered* repeated patterns for compression. In this section, we will present the results of the experiment framework described in chapter 6. For this, the dataset MTC-ANN is used. We will compare the two pattern discovery tasks used for compression: first we will give the results of Song Pattern Discovery, then the results of using Class Pattern Discovery.

7.4.1. Song Pattern Discovery of MTC-ANN

The results for the Song Pattern Discovery task are shown in figure 7.2 using the Gapless Reconstruction method and Note to Note Alignment[1]. Recall that with Song Pattern Discovery task, the Pattern Discovery algorithms will discover repeated patterns within each song separately. We run each algorithm for every combination of parameter values (see section 5.2). The resulting coverage and accuracy for all these experiments are shown in the graph. The output of the classification on the complete songs and the naive lossy compression methods (which we will call the *reference experiments* from now on), as well as the results on the annotated motifs are also plotted. A graphical representation of the effect of the parameters for the various algorithms is shown in appendix B and the raw data of this experiment can be found in appendix D.4.

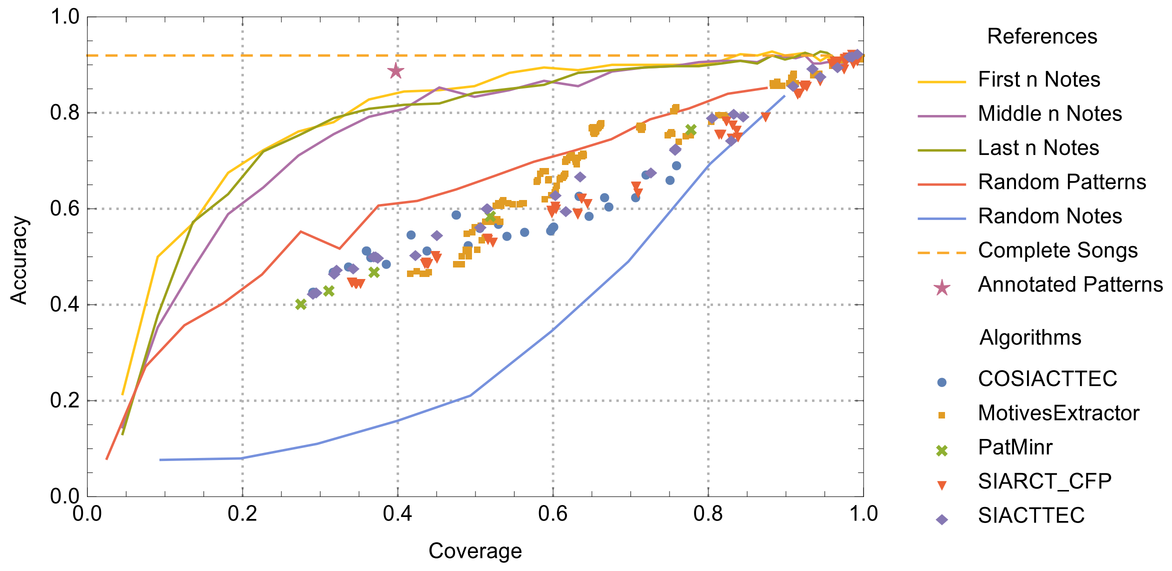


Figure 7.2.: Song Pattern Discovery of MTC-ANN using the Gapless Reconstruction method and Note to Note Alignment. Each point in the plot represents a parameter configuration.

It can be seen from both the algorithm output and the reference experiments that there exists a positive relation between the coverage and accuracy. This is not very surprising, since the classification on the

[1]Due to the large amount of information in this graph, especially at high coverage values the information could be a bit cluttered. None of the Pattern Discovery algorithms exceed the classification accuracy of the uncompressed songs, although it looks from the graph that this is not the case. In Appendix C, larger images of the graphs can be found.

complete songs performs already very well. When the coverage decreases, the amount of information (the number of notes) to successfully calculate the similarity between the songs also decreases. So the higher the coverage, the better the alignment methods works and the higher the classification accuracy will be. Since we want to study the effect of compression by Pattern Discovery algorithms, we define a "good" performing algorithm as one that results in a low coverage, but also with a high classification accuracy (the top-left part of the graphs). In contrast, algorithms that return too many patterns perform less according to our viewpoint because this leads to a coverage of almost 1.0 and therefore a very low compression.

Almost every pattern discovery output performs better than the Random Notes selection, except from a few parameter configurations of SIARCT-CFP at a coverage of 0.77. On the other hand, the selection of the first, middle or last *n* notes works better than any Pattern Discovery algorithm. By selecting only the first 18 notes of each song (corresponding to an average coverage of 0.41) more than 84% of the pieces are classified correctly. Only the annotated motifs show better results, with an accuracy of 0.89 (see section 7.3).

The selection of Random Patterns outperforms almost all pattern discovery output, only a few parameter configurations of MotivesExtractor yield better results. This means that selecting the same notes in each song (i.e. selecting note 3–6, 10–14, ...) works better in this case than selecting notes based on repetition within each song. However, note that in this pattern discovery task, only patterns inside *songs* are found, which are not necessarily the same patterns that are characteristic for the whole tune class it belongs to.

For a number of configurations of MotivesExtractor, SIARCT-CFP and SIACTTEC, the coverage is very close to 1.0. This means that almost every note is present in a pattern discovered by one of these algorithms. Because of this, the resulting accuracy is also comparable with the classification on the uncompressed songs. The discovered patterns of COSIACTTEC and PatMinr cover less notes; the highest coverage of these algorithms is about 0.78. In summary, all Pattern Discovery output performs worse than most of the lossy compression methods, except the Random Note Selection. Even the selection of Random Patterns works better than most Pattern Discovery output.

7.4.2. Class Pattern Discovery of MTC-ANN

The results of the pattern discovery task within *tune classes* are presented in figure 7.3 (the raw data can be found in appendix D.5). Here, the patterns that are discovered occur in multiple songs of the same tune class, as opposed to the Song Pattern Discovery task, where we only look at repetition within a song. In these experiment, the results of the reference experiments are the same as for the Song Pattern Discovery task, because they are only influenced by the similarity measure we choose. Compared to the Song Pattern Discovery task, the resulting coverage and accuracy of the algorithm output are more spread. Some parameter configurations lead to a very low coverage and accuracy.

The positive relation between coverage and accuracy that we observed in the previous section is also visible in this task. Also more algorithms return very few patterns, which results in a low coverage and accuracy. This is the case for a number of parameter configurations of MotivesExtractor and MGDP. For some parameter values of MotivesExtractor, the classification accuracy and coverage is 0. This means that either no repeated patterns were found in the tune classes, or that each discovered pattern occurrence overlapped at least two songs. The latter situation leads to the removal of these occurrences in the splitting process (section 6.4.2). Compared to the Song Pattern Discovery task, the total maximum coverage is lower: around 0.9 for some configurations of SIACTTEC against nearly 1.0 for the Song Pattern Discovery task.

Overall, the results are better than the Song Pattern Discovery. Many algorithm configurations now lead to a higher accuracy than the Random Pattern reference experiment. The only method that performs

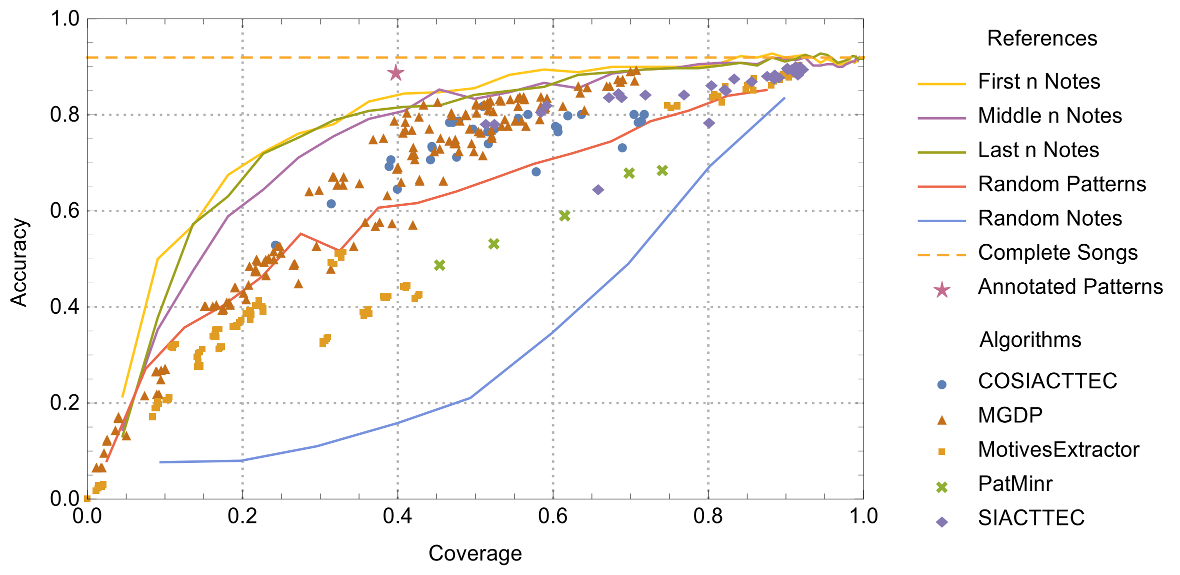


Figure 7.3.: Class Pattern Discovery of MTC-ANN using the Gapless Reconstruction reconstruction method and Note to Note Alignment. Each point in the plot represents a parameter configuration.

worse is the PatMinr algorithm. The selection of the first n notes works still better than any Pattern Discovery algorithm configuration. For some parameter configurations, the MGDP algorithm comes close to the performance of the selection of the middle n and last n notes. This is an expected result, since the MGDP algorithm only discovers patterns within a tune class that are *distinctive* compared to the other tune classes. This is the only method in our comparison that uses so-called *anti-corpora* for pattern discovery (see section 3.1.1). Observe that in the output of the MotivesExtractor algorithm, there are no results in the coverage interval between 0.45 and 0.75. This is the only algorithm that has such a "gap" in the results.

To conclude, the Pattern Discovery results are overall better than for the Song Pattern Discovery task with respect to the reference experiments, but the selection of the first n notes and the annotated motifs are still superior in terms of classification accuracy.

7.5. Comparison of Reconstruction Methods

We will now study the effect of changing the Reconstruction Method (see section 5.3). Recall that this step in the framework is responsible for reconstructing each song using the patterns that are discovered. Changing the reconstruction method can compress the songs even more, or change the input for the alignment algorithms. In our proposed experiment framework, three different reconstruction methods were introduced:

- Onset Preserved Reconstruction
- Gapless Reconstruction
- First Pattern Occurrence Reconstruction

We already observed from the results on the annotated motifs in section 7.3 that there is no difference in accuracy and coverage between using Onset Preserved Reconstruction and Gapless Reconstruction, because both alignment algorithms seem to ignore gaps in the sequences. Since it turned out that this is also the case in these results, we will discard the first reconstruction method and only compare the results between the Gapless Reconstruction and First Pattern Occurrence Reconstruction method.

The output of the First Pattern Occurrence Reconstruction method is shown in figure 7.4 for the Song Pattern Discovery task and in figure 7.5 for the Class Pattern Discovery task. To make the comparison between this reconstruction method and the Gapless Reconstruction clearer, the results of the latter method are also shown in the graph with a light gray color. Note that these results are the same as in figure 7.2 and 7.3. In both discovery tasks, the most results have a slightly smaller coverage compared to the Gapless Reconstruction method. The results of the reference experiments are still unchanged, but the classification performance using the annotated motifs has decreased (see section 7.3).

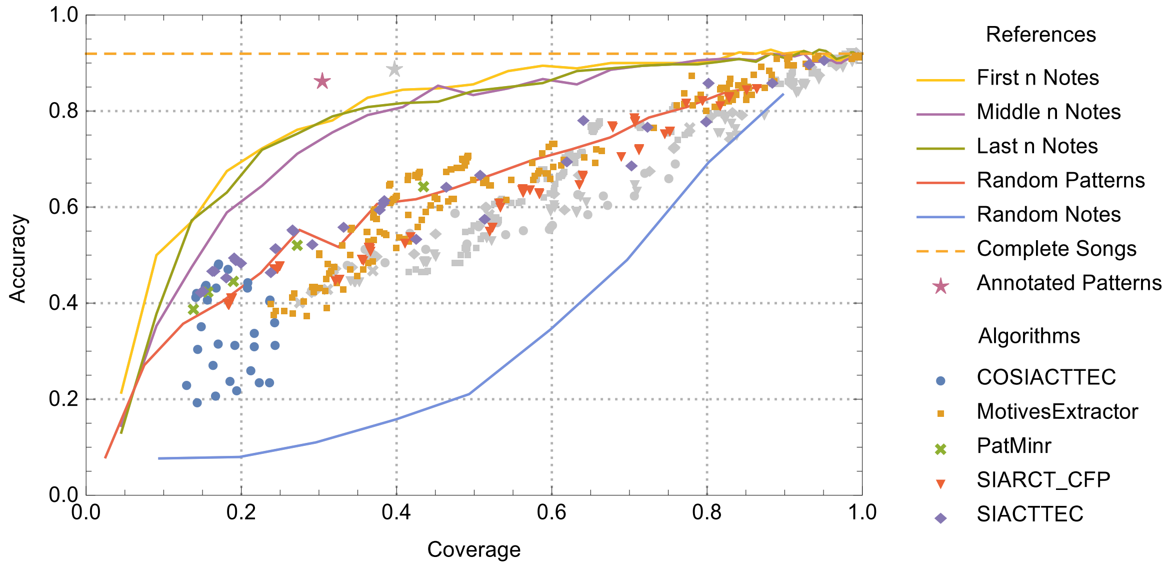


Figure 7.4.: Song Pattern Discovery of MTC-ANN using the First Pattern Occurrence Reconstruction method and Note to Note Alignment. In gray the results from figure 7.2 using the Gapless Reconstruction method. Each point in the plot represents a parameter configuration.

To see what happens with the individual algorithms, we compared the average coverage and classification accuracy for both reconstruction methods in table 7.3 for Song Pattern Discovery and table 7.4 for Class Pattern Discovery. In order to test if there is a significant difference in accuracy or coverage between the two reconstruction method, we will test this statistically. Since the data is not normally distributed, we use the non-parametric Wilcoxon signed-rank test (Wilcoxon, 1946) with a significance level $\alpha = 0.01$ to check individually if the decrease or increase of either the coverage or accuracy is statistically significant. For the Class Pattern Discovery task, the decrease in accuracy of SIACTTEC and PatMinr is not statistically significant ($p = 0.24$ and $p = 0.79$, respectively). Especially for the PatMinr algorithm, this can also be observed from the graph. None of the changes in coverage and accuracy for PatMinr are statistically significant. This has to do with the limited number of parameter configurations we tested, such that a significance level of 0.01 is too low. If we choose α to be 0.10 (the smallest possible significance level for a test with 5 samples), the differences *are* statistically significant ($p = 0.058$) except for the decrease in accuracy for the Class Pattern Discovery.

For the Song Pattern Discovery task, the most noticeable change is the performance of the COSIACTTEC algorithm. Because this algorithm allows every note to be in at most one pattern, the songs are highly compressed. Therefore, the resulting accuracy for this algorithm is very low. We can see from table 7.3 that the average coverage dropped from 0.51 to 0.19 compared with the Gapless Reconstruction method. Although the average classification accuracy is decreased for all algorithms, there are more parameter configurations that lead to a higher accuracy than the Random Pattern selection. However,

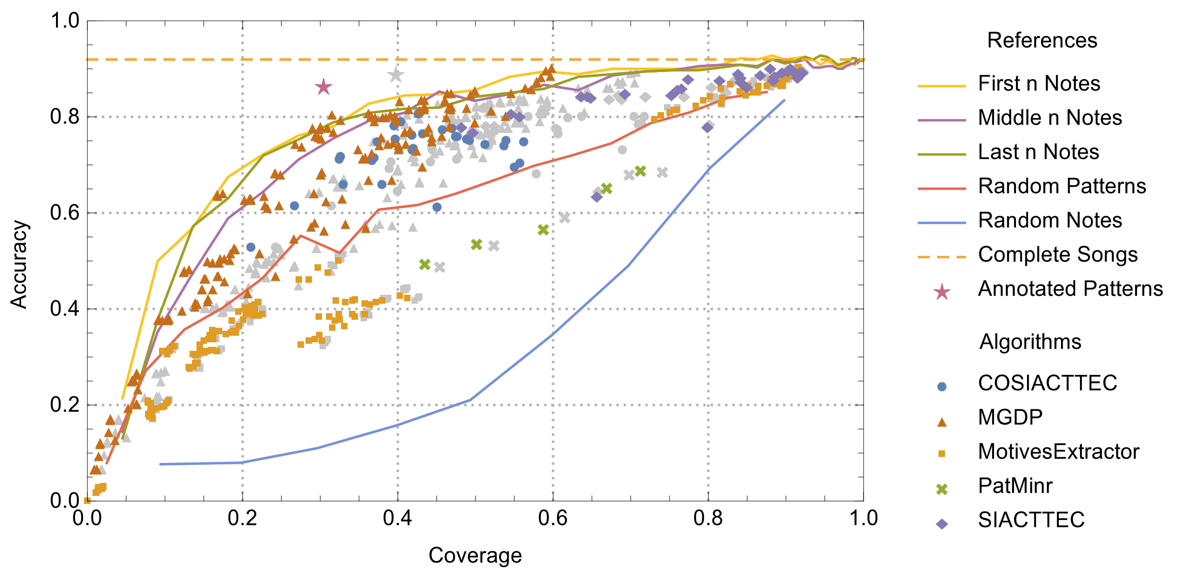


Figure 7.5.: Class Pattern Discovery of MTC-ANN using the First Pattern Occurrence Reconstruction method and Note to Note Alignment. In gray the results from figure 7.3 using the Gapless Reconstruction method. Each point in the plot represents a parameter configuration.

Algorithm	**Average Coverage**			**Average Accuracy**		
	GR	**FPOR**	**Difference**	**GR**	**FPOR**	**Difference**
SIACTTEC	0.64	0.44	−0.20	0.67	0.61	−0.06
COSIACTTEC	0.51	0.19	−0.32	0.56	0.35	−0.21
MotivesExtractor	0.84	0.74	−0.10	0.81	0.77	−0.04
PatMinr	0.45	0.24	−0.21*	0.53	0.49	−0.04*
SIARCT-CFP	0.74	0.52	−0.22	0.71	0.62	−0.09

Table 7.3.: Differences between the averages of accuracy and coverage using Song Pattern Discovery for the Gapless Reconstruction method (GR) and the First Pattern Occurrence Reconstruction method (FPOR). Values marked with an asterisk are not statistically significant ($p > 0.01$).

compression using the first, middle or last n notes is still superior to any Pattern Discovery algorithm output in terms of accuracy. The PatMinr algorithm gives for most configurations better results than the Random Pattern selection, which is an improvement over the Gapless Reconstruction method. Still, a lot of parameter configurations of MotivesExtractor and SIACTTEC lead to a high coverage and accuracy. This means that these algorithms return either a large number of patterns, or just very large patterns. To conclude, removing repetitions in a single song lead to a lower coverage and accuracy. However, this decrease is not linear: the decrease in coverage is larger than the decrease in classification accuracy. Due to this shift, more algorithms perform better than the Random Pattern selection compared to the Gapless Reconstruction method.

The Class Pattern Discovery task using First Pattern Occurrence Reconstruction shows the same differences as the Song Pattern Discovery task. However, the results of the PatMinr algorithm are not as much influenced by the First Pattern Occurrence Reconstruction method than in the Song Pattern Discovery task. In fact, there is no significant decrease in accuracy for this algorithm. We also see that the COSIACTTEC algorithm is not as much influenced by the First Pattern Occurrence Reconstruction than in the other pattern discovery task. According to table 7.4, the average decrease in coverage is also larger than the average decrease in accuracy. A lot of configurations of the MGDP algorithm return a

Algorithm	Average Coverage			Average Accuracy		
	GR	FPOR	Difference	GR	FPOR	Difference
SIACTTEC	0.80	0.77	−0.03	0.86	0.85	−0.01*
COSIACTTEC	0.53	0.43	−0.10	0.75	0.73	−0.02
MotivesExtractor	0.23	0.23	0	0.29	0.29	0
PatMinr	0.61	0.58	−0.03*	0.60	0.58	−0.02*
MGDP	0.36	0.27	−0.09	0.61	0.59	−0.02

Table 7.4.: Differences between the averages of accuracy and coverage using Class Pattern Discovery for the Gapless Reconstruction method (GR) and the First Pattern Occurrence Reconstruction method (FPOR). Values marked with an asterisk are not statistically significant ($p > 0.01$).

higher accuracy than the selection of the first, middle or last n notes. At a coverage of 60%, the MGDP Algorithm comes very close to the accuracy of the classification on the complete songs.

In summary, Onset Preserved Reconstruction and Gapless Reconstruction lead to the same results because gaps are omitted by the similarity calculation algorithms. Using First Pattern Occurrence Reconstruction results in a smaller coverage for nearly all experiments, but as a result this also decreases accuracy for some algorithms (especially for COSIACTTEC). Because the accuracy for both pattern discovery approaches is not decreased much compared to the decrease in coverage in the case of First Pattern Occurrence Reconstruction, we conclude that the First Pattern Occurrence Reconstruction yields better results than Onset Preserved Reconstruction: the classification does not suffer much from the large decrease in coverage.

7.6. Comparison of Similarity Calculation Methods

All presented results up to now have used the Note to Note Alignment algorithm for calculating the similarity between the melodies. We will now change this method to the Melodic Shape Alignment algorithm to see how it influences the accuracy. We already know from the results in section 7.1 and 7.3 that Note to Note Alignment resulted in a higher classification accuracy for these experiments. In this section, we will study the effect of changing the similarity calculation method on the resulting coverage and accuracy using the Pattern Discovery algorithms. The results of the pattern discovery task within *tune classes* are shown in figure 7.6. Because the observations and conclusions are the same as for the Song Pattern Discovery, we will show the graph for one task only. In table 7.5 , the average coverage and accuracy for each algorithm and for both similarity measures are presented. We also use the Wilcoxon signed-rank test if the average decrease in coverage or accuracy is statistical significant ($\alpha = 0.01$). Like in the results of section 7.5, we observe that the differences for the PatMinr algorithm are not significant for this significance level because the number of parameter configurations is too low ($p = 0.058$).

It is clear that the results are worse with respect to the accuracy of the Note to Note Alignment method. For the reference experiments, it seems that the Melodic Shape Alignment algorithm performs as good as the Note to Note Alignment method for songs with a low coverage. Between a coverage of 0.4 and 0.6, the Note to Note Alignment method performs definitely better than the Melodic Shape Alignment. The selection of the first, middle and last n notes starts at higher coverage, because the Melodic Shape Alignment needs at least 3 notes in order to interpolate a b-spline instead of 2 for the Note to Note Alignment (which uses the interval between two consecutive notes as similarity measure). The performance of the annotated motifs come very close to the results of the first and last n notes. When selecting the last 16 notes of each song, the accuracy is almost as high as for the annotated motifs with a coverage of only 0.36.

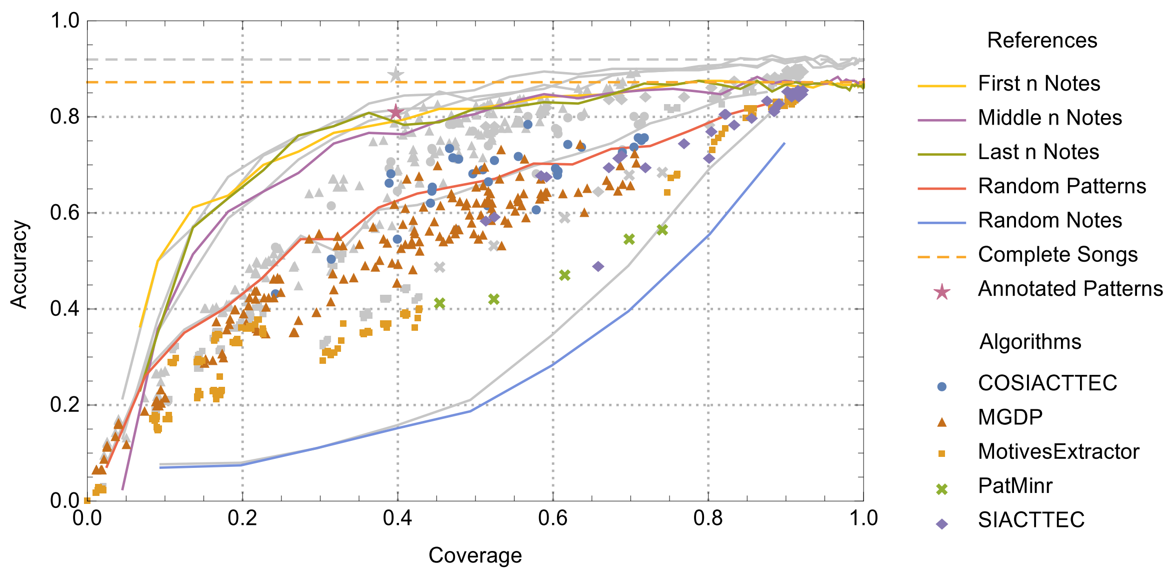


Figure 7.6.: Class Pattern Discovery of MTC-ANN using the Gapless Reconstruction method and Melodic Shape Alignment. In gray the results from figure 7.3 using the Note to Note Alignment algorithm. Each point in the plot represents a parameter configuration.

Algorithm	Average Coverage	Average Accuracy		
		Note to Note	Melodic Shape	Difference
SIACTTEC	0.80	0.86	0.77	−0.09
COSIACTTEC	0.53	0.75	0.68	−0.07
MotivesExtractor	0.23	0.29	0.26	−.0.03
PatMinr	0.61	0.60	0.48	−0.12*
MGDP	0.36	0.61	0.48	−0.13

Table 7.5.: Differences between averages of accuracy and coverage using Class Pattern Discovery. Values marked with an asterisk are not statistically significant ($p > 0.01$).

For the Pattern Discovery algorithms, we also see that the accuracy is lower compared to the Note to Note Alignment algorithm.

7.7. Comparison of the Folk Song Datasets

So far, we compared the different variables in the proposed framework using the Dutch folk song dataset MTC-ANN. To check if the behavior of the algorithms and naive compression methods are specific for this dataset, we will compare these results by using Pattern Discovery as compression method on another dataset: the Irish Folk Tunes Dataset. Note that we do not have a set of annotated motifs for this dataset, hence we could not compare with such expert knowledge. The results of the Pattern Discovery algorithms and reference experiments on the Irish Folk Tunes Dataset are shown in figure 7.7 and figure 7.8 for the Song Pattern Discovery and Class Pattern Discovery, respectively. A graphical representation of the effect of the parameters for the various algorithms is shown in appendix B. The raw data can be found in appendix D.6 and D.7.

As we have seen before in section 7.1, performing Note to Note Alignment on the uncompressed songs gives already a very high accuracy of 0.96. For the Song Pattern Discovery task, the selection of the

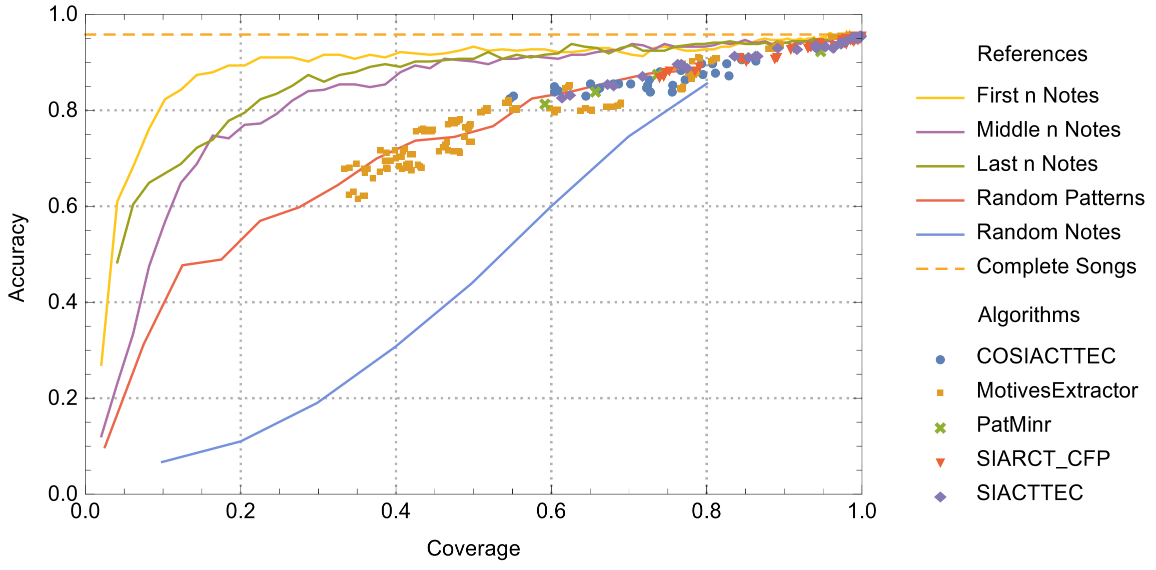


Figure 7.7.: Song Pattern Discovery of the Irish Folk Tunes Dataset using the Gapless Reconstruction method and Note to Note Alignment. Each point in the plot represents a parameter configuration.

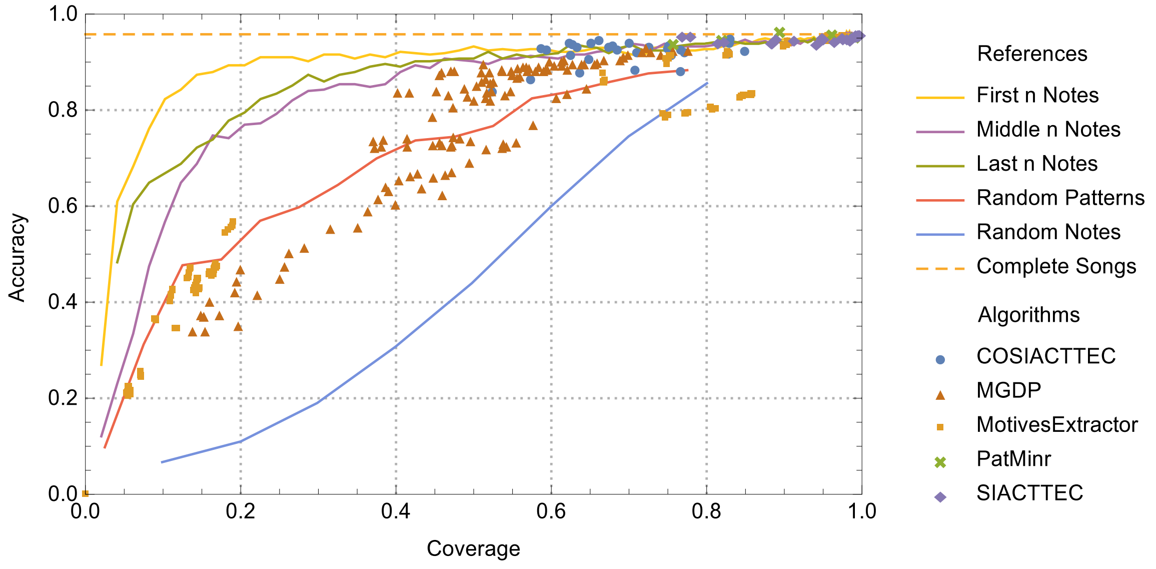


Figure 7.8.: Class Pattern Discovery on the Irish Folk Tunes Dataset using the Gapless Reconstruction method and Note to Note Alignment. Each point in the plot represents a parameter configuration.

first, middle and last n notes outperform also on this dataset every configuration of a Pattern Discovery algorithm. Compared to the results of MTC-ANN, the accuracy for these naive compression methods increase faster with respect to the coverage. For example, at a coverage of 0.2 the accuracy when selecting the first n notes is already more than 90%, whereas on MTC-ANN this happens at a coverage of 0.6. We expect that this is the result of the fact that the songs of the the Irish Folk Tunes Dataset are longer than the songs from MTC-ANN. Since coverage is a relative measure, a coverage of 0.2 for the the Irish Folk Tunes Dataset (20 notes) corresponds with a coverage of 0.45 for MTC-ANN. We have already seen that selecting the first, middle or last *n* notes is good enough for the alignment algorithm to successfully calculate the similarity (even if *n* is small or it relatively covers only a small part of the song). The output of the Pattern Discovery algorithms is more dense and is situated very close to the results of the Random Pattern selection. MotivesExtractor is the only algorithm that return songs with an average coverage of less than 0.55.

The resulting accuracies for the Class Pattern Discovery task are also higher than MTC-ANN. Both COSIACTTEC, SIACTTEC and PatMinr outperform for some parameter configurations the first, middle and last *n* notes selection and PatMinr even reaches the same accuracy as the classification on the complete songs at a coverage of 0.9. The gap in the output of MotivesExtractor is also visible on this dataset. For a few parameter configurations of MotivesExtractor, the compressed songs perform even worse than the Random Notes selection at a coverage and accuracy around 0.8.

To conclude, also on this dataset, classification using alignment yields a very high retrieval performance. For MTC-ANN, using sequence alignment for calculating the similarity between folk songs was already proven to be successful. From these results, we can conclude that this is also the case for the Irish Folk Tunes Dataset. Most of the algorithms show the same behavior as in the classification on MTC-ANN, but the accuracy increases very fast at small coverage values. This shows that the classification results on MTC-ANN are not an isolated case, because we see the same characteristics for the Irish Folk Tunes Dataset.

Conclusion

We have seen from the compression using the annotated motifs (section 7.3) that the classification accuracy is only slightly worse than the accuracy on the uncompressed songs. This supports the claim that these patterns are important in the tune classification process and supports our hypothesis that annotated or discovered patterns could be used for compression for determining the similarity between folk songs.

The use of pattern discovery within songs (section 7.4.1) showed that many algorithms were performing worse than the Random Pattern selection. In the light of this compression and classification task, we can conclude that repeated patterns inside a song contain not enough information to classify the song in the right tune class. None of the algorithms reached the coverage and accuracy of the annotated motifs. This is also the case for the Class Pattern Discovery (section 7.4.2). While these results are overall better than for the Song Pattern Discovery task, the selection of the first *n* notes perform still better. We can see that the input of extra information about the tune classes pays of: some parameter configurations of the MGDP algorithm show comparable results as the selection of the middle or last *n* notes.

Changing the reconstruction method (section 7.5) to First Pattern Occurrence Reconstruction instead of the Gapless Reconstruction approach that was used in the previous experiments showed that this lead to a lower average coverage for all algorithms. As an effect, the classification accuracy has also lowered. Both Onset Preserved Reconstruction and Gapless Reconstruction lead to exactly the same coverage and classification accuracy. Both alignment algorithms we use in the framework seem to ignore gaps in the sequences.

The effect of these two alignment algorithms of the classification accuracy is presented in section 7.6. For both pattern discovery approaches, the Melodic Shape Alignment resulted in a lower accuracy than the Note to Note Alignment. The last variation we made was the change of the dataset. In section 7.7, we have seen that the algorithms and reference experiments show the same behavior as with MTC-ANN.

All in all, we can conclude that the annotated motifs work better as compression method than any of the Pattern Discovery algorithm parameter configuration we tested. In the next section, we will study the comparison of the annotated motifs with the discovered patterns using the evaluation metrics from the *Discovery of Repeated Themes and Sections* MIREX task.

7.8. Comparing the Pattern Discovery Output with the Annotated Motifs

In the previous sections, we have seen that no set of patterns discovered by a Pattern Discovery algorithm yield better results than the annotated motifs from MTC-ANN. Recall that for the "goodness" of an algorithm output, we want the resulting coverage to be low and the classification accuracy to be high. In this section, we will compare the output of the Pattern Discovery algorithms on MTC-ANN with the annotated motifs. For this, we will use the evaluation metrics that are introduced in the *Discovery of Repeated Themes and Sections* MIREX task (see section 4.1). We assume that the annotated motifs can be used as set of *ground truth* patterns in this case. Because we do not have such a ground truth for the Irish Folk Tunes Dataset, we will only look at the performance of both the Song and Class Pattern Discovery of the MTC-ANN dataset.

The average number of patterns per song for each algorithm is shown in table 7.6. For most of the algorithms, there is a large difference between the Song Pattern Discovery and Class Pattern Discovery task with respect to the average number of discovered patterns per song. This is caused by the fact that there can be found much more patterns *between* songs than *within* a song. Also in these results, the compression strategy of COSIACTTEC is very clear: with 4.70 discovered patterns on average in each song, this algorithm comes closest to the average of 3.40 annotated patterns in each song for the Class Pattern Discovery task.

In figure 7.9, the three evaluation metrics are plotted for all parameter configurations in the Song Pattern Discovery task. For all metrics, the output of Pattern Discovery algorithms are most similar to the annotated motifs when the precision is high (i.e. the ratio of discovered patterns that are also present in the annotated motifs) and the recall is high (i.e. the ratio of annotated motifs that are discovered by the Pattern Discovery algorithm). We see that in the best case, we achieve an establishment precision and recall of almost 0.6 for the MotivesExtractor algorithm. This *establishment* measure mentions how many patterns are found by the algorithm, not taking into account the *number* of occurrences. We can see from the graph that both the MotivesExtractor algorithm and SIACTTEC yield a best precision of about 0.6. Since the SIACTTEC algorithm returns five times as many patterns per song on average, we see that this influences the recall score: compared to MotivesExtractor, many discovered patterns are not part of the annotated motifs.

For the *occurrence* precision and recall, we see for all algorithms that the occurrence precision is quite low compared with the establishment precision: the maximum value is 0.35. This means that for many patterns, more occurrences are discovered than are annotated. We see that the occurrence precision and recall for the SIARCT-CFP output is extremely low: at most 0.02. Since a lot of annotated motifs contains inexact repetitions, we just would suspect that the SIARCT-CFP algorithm would perform very

Algorithm/method	**Average number of patterns per song**	
	Song Pattern Discovery	**Class Pattern Discovery**
COSIACTTEC	1.80	4.70
MGDP	–	47.04
MotivesExtractor	2.53	39.28
PatMinr	2.95	50.64
SIARCT-CFP	19.60	–
SIACTTEC	16.41	313.24

Table 7.6.: Average number of patterns discovered in a single song. For the annotated motifs, the average number of discovered patterns per song is 3.40.

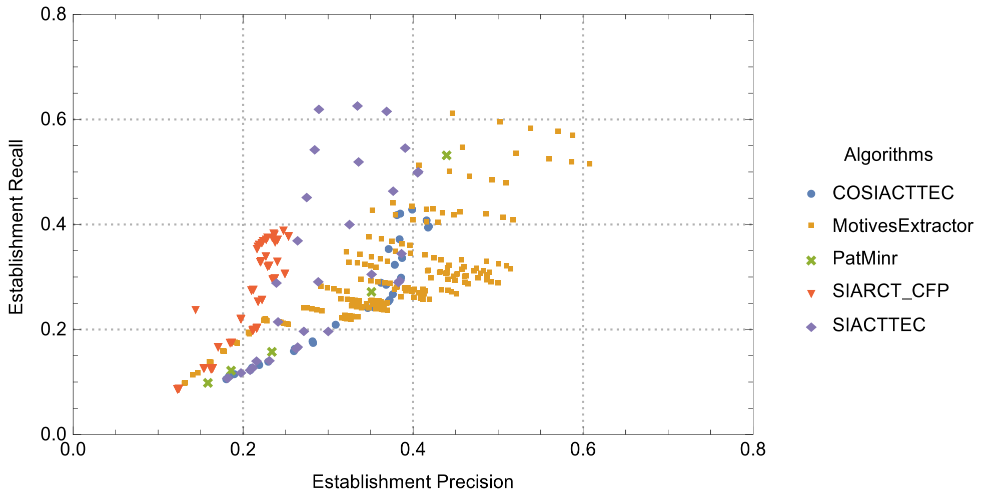
well in this evaluation measure because the algorithm is supposed to discover inexact occurrences. This could mean that there is a problem with the parameters we chose in the experiments.

We did the same evaluation using Class Pattern Discovery task. In other words, the discovered patterns will now be found in a set of songs instead of only one song. We see in general that the resulting precision and recall scores show the same behavior as in the Song Pattern Discovery task: for the establishment and three-layer precision and recall, the results per algorithm are clustered, whereas for the occurrence precision and recall the results per algorithm are more line-shaped.

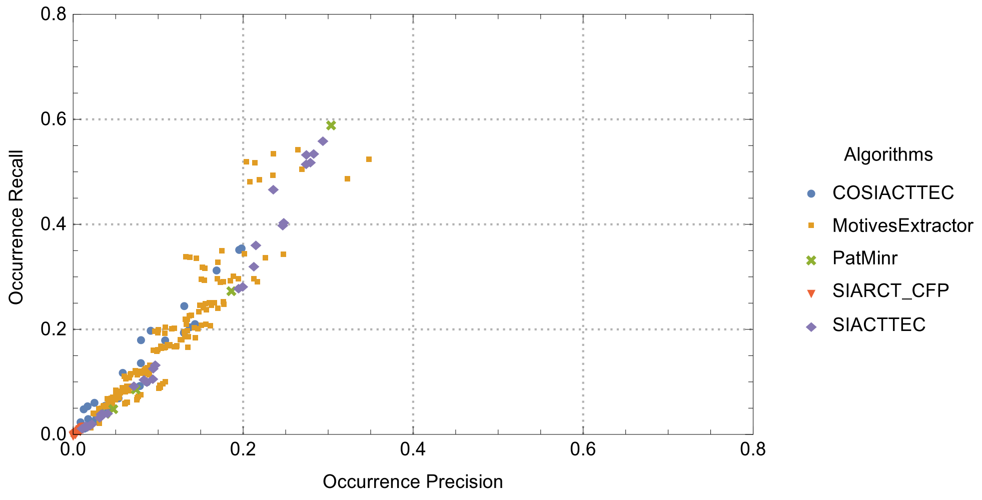
Compared to the Song Pattern Discovery task, the maximum establishment precision is lower than for the Song Pattern Discovery task. We suspect that this has to do with the enormous amount of patterns most algorithms return in this task. The compression approach of COSIACTTEC does not work out well here: while the number of discovered patterns per song is low, many annotated motifs are not discovered by the algorithm. The MGDP algorithm shows also his strength here: it leads to the highest establishment and three-layer precision.

One would expect that the average score for the Class Pattern Discovery task would be higher than for the Song Pattern Discovery task, because the set of ground truth patterns contain only patterns that are characteristic for the complete tune class instead of only for the single pieces. If we compare the results in figure 7.9 and 7.10, we see that overall the establishment precision is lower for the Class Pattern Discovery task then for the Song Pattern Discovery task. For the occurrence precision, it is exactly the other way around. For the three-layer precision and recall, the results of the Class Pattern Discovery are more spread than for the Song Pattern Discovery.

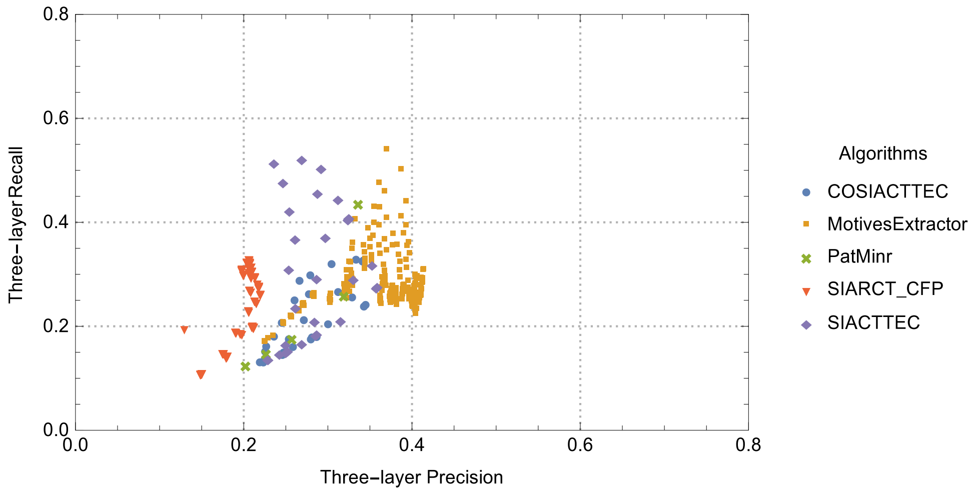
To summarize, we have seen that for both the Song Pattern Discovery and Class Pattern Discovery task, most algorithms return a lot more patterns than the annotated motifs. According to the evaluation metrics used in this experiment, there is no algorithm that discovers all (occurrences of) the annotated motifs. In order to improve these results, the algorithms have to be adapted for this specific compression task.



(a) Establishment Precision and Recall.

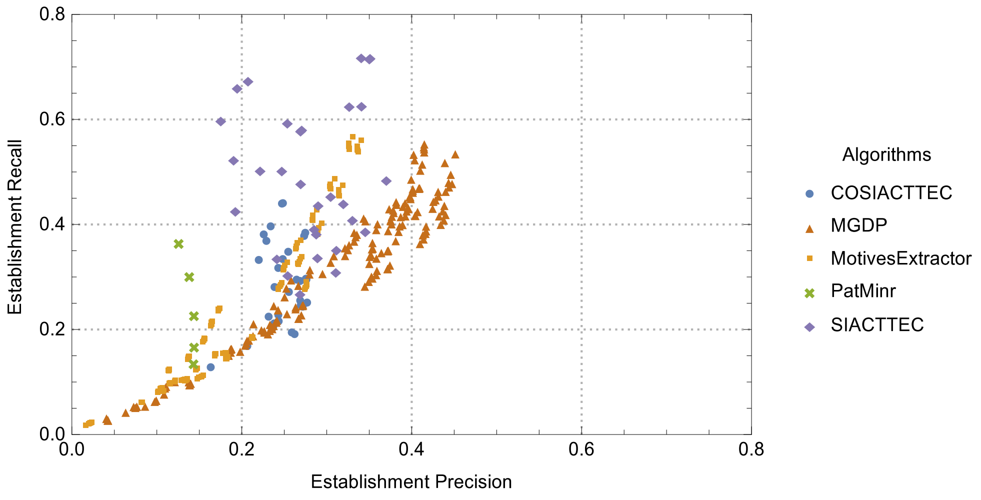


(b) Occurrence Precision and Recall.

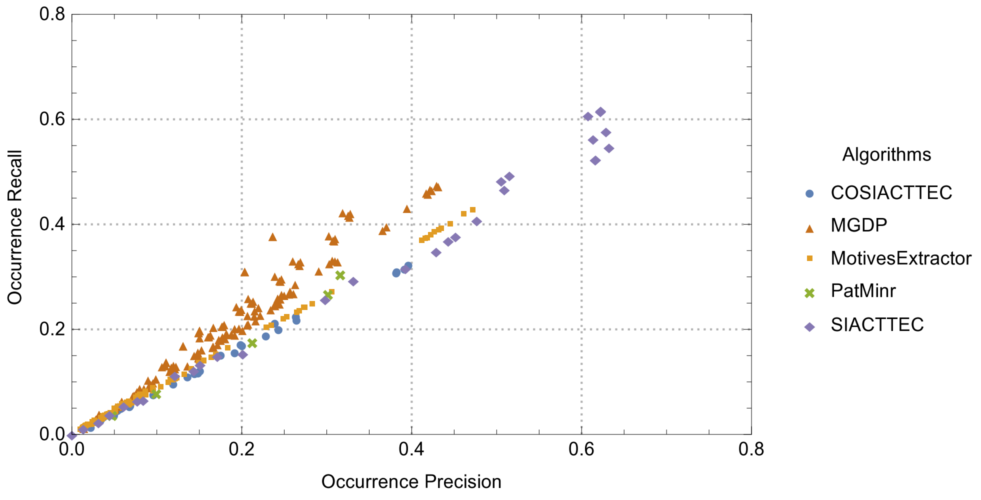


(c) Three-layer precision and recall.

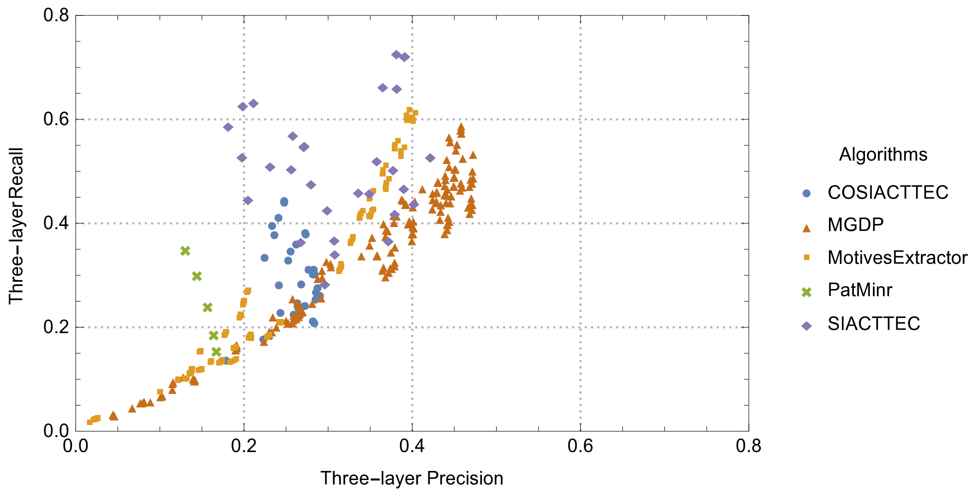
Figure 7.9.: Three types of precision and recall of the Song Pattern Discovery. Each point in the plot represents a parameter configuration.



(a) Establishment Precision and Recall.



(b) Occurrence Precision and Recall.



(c) Three-layer Precision and Recall.

Figure 7.10.: Three types of precision and recall of the Class Pattern Discovery. Each point in the plot represents a parameter configuration.

Discussion 8

In the introduction, we stated three research questions regarding the use of Pattern Discovery algorithms as compression measure in order to classify folk songs. In this section, we will discuss the results we obtained in the previous section based on these research questions:

Research Question 1.
How is the accuracy of classification based on similarity influenced by compressing folk songs?

Research Question 2.
Is there a difference in classification accuracy between automatically discovered patterns and annotated patterns?

Research Question 3.
Are compression and classification accuracy a good measure for evaluating Pattern Discovery algorithms?

In each section, we give the answer to the corresponding research question based on the results from the previous chapter, and put these results in a broader context.

8.1. The Influence of Compression on Folk Song Classification

In order to compare and evaluate the effects of compression using Pattern Discovery algorithms, we need to know the general effects of compression on this type of songs, as well as the performance of tune classification without compression.

We see from the results of section 7.1 that for both datasets we used in this study, the classification accuracy is already very high for the uncompressed songs. These results correspond to related studies that use sequence alignment methods to classify folk songs (Hillewaere et al., 2012; Van Kranenburg, 2010). We therefore want to know if compressing the songs gives *comparable* results with respect to the classification on songs without compression. The results in section 7.2 show that the selection of the first, middle or last n notes is a very simple, yet powerful compression method. The performance of the other two proposed compression strategies, randomly selecting notes and randomly selecting patterns, perform worse than the selection of the first, middle or last n notes. We suspect that the alignment methods we use for calculating the similarity score between the songs works much better on consecutive parts of a song rather than on a number of small fragments.

In a study by Xia et al. (2014), the first and the last part of MIDI songs are selected for similarity measuring to speed up the search in a large song collection. They argued that these parts of the songs are the most constant between pieces and are therefore a good choice for this kind of compression.

While our results support their reasoning, we have also shown that on our datasets, it does not matter much whether we select the first, middle or last part of each song.

To answer the first research sub question, we can say that performing compression on folk songs decreases the power to successfully determine the similarity between these songs. It is also clear that naive compression techniques that leave a large part of a song intact result in a higher classification accuracy than compression methods that lead to many gaps in the compressed songs for the same coverage value. For future research, it would be good to study the effect of the alignment algorithms we used in the experiment framework (chapter 5) in more detail. We would therefore propose to also test other similarity measures, to see if the results are not too much biased by the fact that alignment on folk songs is already a very powerful method. The research of Hillewaere et al. (2012) can be a good starting point for this, since they compared multiple string based classification measures for folk songs.

8.1.1. Generalization to other Musical Domains

In the Introduction, we explicitly narrowed the scope of this project to the classification of (compressed) folk songs into tune classes. The question is how the results we have seen in chapter 7 can be generalized to other musical domains, or other classification purposes (such as classification on musical genre or composer). In the current implementation of the framework, only monophonic music can be analyzed and compressed. The reason for this is that not all Pattern Discovery algorithms support polyphonic music and also both sequential alignment algorithms are designed for monophonic music only. This means that the current implementation cannot be used for music that is mostly polyphonic such as many classical and popular music pieces, without extracting a monophonic melody first. However, the framework is designed in such a way that other Pattern Discovery algorithms or similarity measures can be easily added, to increase the support for polyphonic music.

The second generalization is the type of classification task. In this thesis, we focused on the classification of songs in tune classes, based on the similarity between the songs. In early experiments, where folk songs were classified into (geographic) origin and rhythm instead of similarity, the results showed that there was not a clear relation between the coverage of a compressed song and the classification accuracy. On top of that, the classification accuracy on the uncompressed songs was also very low. Since other research has shown that it is possible to classify (uncompressed) folk songs in geographic origin and rhythm (Hillewaere et al., 2012) or using repeated patterns (Conklin and Anagnostopoulou, 2011; Lin et al., 2004), this is something that needs to be explored further. We believe that the removal of notes that do not belong to a pattern is a good strategy in the case of classification based on similarity, but that this method needs improvement to also support other classification types. In other words, our proposed framework works particularly well in the case where the classes are based on *explicit* melodic similarity between the pieces. For other classification types, there is improvement possible. A possible explanation for this is that the discovered patterns that were found in the pieces were not similar enough *within* the classes and not distinctive enough *between* the classes in these cases.

8.2. Using Annotated and Discovered Patterns for Compression

We see from the results in section 7.3 that the Annotated Motifs from MTC-ANN lead to both a high accuracy and a low coverage. This supports the statements by Volk and Van Kranenburg (2012) that repeated patterns contain relevant information about the tune class of the song it occurs in, such that this information can be used for classification. Using the annotated motifs, the songs from MTC-ANN are nearly 60% compressed, while the accuracy is only slightly decreased in comparison to the classification on the uncompressed songs. Based on this result, we can conclude that at least the *annotated* motifs could be used as compression method in determining similarity between the folk songs in MTC-ANN.

The results in section 7.4 show that all Pattern Discovery algorithms we have tested as compression method, lead to a worse classification accuracy compared to the annotated motifs. In the following sections, we will discuss this observation in more detail.

8.2.1. Output size of Pattern Discovery algorithms

In the introduction, we stated that many Pattern Discovery algorithms suffer from a too large number of discovered patterns. Many algorithms solve this problem by adding a filtering step after the pattern discovery (Janssen et al., 2013). Based on a heuristic (like the heuristics we have described for COSIACT-TEC in section 3.2.2), patterns are removed or merged with other patterns. We have learned from the annotated motifs that, with (relatively) few patterns per song, the classification accuracy could still be high. A possible reason why Pattern Discovery algorithms perform worse is that these algorithms return large number of *non-important* (i.e. not characteristic for the tune class) patterns, that cancel out the distinctive power of the *characteristic* patterns (like those of the Annotated Motifs).

Since we compress by reconstructing each song with only the notes that are contained in at least one discovered pattern, a large output will lead to a high coverage. Without modifying the Pattern Discovery algorithms, most parameter configurations we tested are a bad choice for compression, especially when so many patterns are returned that the coverage reaches 1.0. We have shown that reconstructing each song with only one occurrence per pattern in each song (section 7.5) decreases the coverage for most of the parameter configurations we tested, while the accuracy did not decrease much. Apparently, multiple occurrences of a pattern can be reduced to one occurrence without the loss of much information for the similarity calculations.

Another unwanted effect of the pattern extraction on the compressed songs is that for some parameter configurations, no repetitions are found in many songs. This happens in both the Song Pattern Discovery and Class Pattern Discovery task. This situation is in correspondence with the Annotated Motifs, since six songs from MTC-ANN do not have a repetition annotated in the Annotated Motifs. This problem is caused by the fact that the *degree of repetition* is different for all songs. This means that if we want to limit the output size by changing the algorithm parameters in order to be more *strict* in the discovery process (e.g. increasing the minimum pattern length or increasing the minimum compactness of a pattern), the number of songs without any discovered patterns increases. As a consequence, this influences the classification accuracy. The problem could be solved by modifying the algorithm parameters during the discovery process. We already did this for the MGDP algorithm, because this method suffered much from this problem. Since we wanted to study the effect of the algorithms without any strict modification, this could be further studied in future research.

8.2.2. Analysis of the Annotated Patterns

In order to better understand the discriminative power of the annotated patterns, we have plotted the positions of the annotated patterns for two tune classes. The pattern positions of the songs in the tune class "En er waren eens twee zoeteliefjes" are presented in figure 8.1. We can immediately see that the *position* of the repeated patterns are very consistent across the songs. This makes us expect that position is a good heuristic for finding pattern occurrences. This is in correspondence with the observations of Van Kranenburg (2010), who stated that *phrase position* (i.e. the position of a note in a phrase) is a good feature for Note to Note Alignment. Pattern occurrences that are annotated based on their position, are often *inexact* occurrences in terms of melodic and rhythmic similarity because location of the pattern is assumed more important than the exact similarity of melody or rhythm. This emphasizes the need of more Pattern Discovery algorithms that are able to discover inexact pattern occurrences, or discovering repetitions primarily based on their position. In a related research field, it is shown that positional

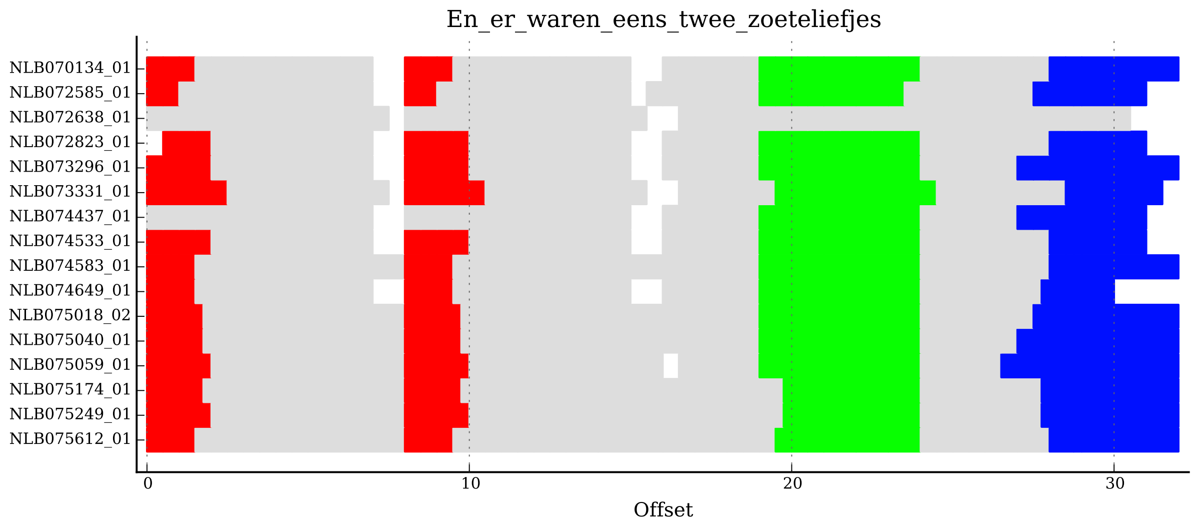


Figure 8.1.: Pattern plot for the tune class "En er waren eens twee zoeteliefjes". Every motif class has its own color.

information of repetitions is also an important feature for melodic segmentation (Rodríguez-López and Volk, 2015).

Not all tune classes from MTC-ANN show a clear and consistent positioning of the annotated motifs like in figure 8.1. To illustrate this, in figure 8.2 the annotated motifs of the tune class "Zolang de boom zal groeien" are shown. Although the patterns are not as nice aligned as in the example of "En er waren eens twee zoeteliefjes", the order in which the patterns occur is consistent over all songs.

Another reason for the differences between the Annotated Motifs and the output of the Pattern Discovery algorithms is presented in figure 8.3. Here, a number of occurrences from the first repeated pattern of the tune class "Zolang de boom zal bloeien" is shown. Although all fragments are categorized as occurrences of the same pattern, the occurrences vary in both melody and rhythm. We therefore expect that these variations can also be a reason why Pattern Discovery algorithms did not find the same pattern occurrences as those that are annotated, since many of the tested Pattern Discovery algorithms cannot discover inexact repetitions.

8.2.3. Differences between Song and Class Pattern Discovery

Most Pattern Discovery algorithms are specifically developed for the discovery of repeated patterns within a single song in order to find patterns that are *noticeable* for a human listener. Driven by the importance of repeated motifs in the classification of folk songs, we have implemented a way to adapt Song Pattern Discovery algorithms to also support the discovery of repeated motifs within a class of songs. The reason for implementing and testing both discovery approaches is twofold. At first, we have tested the hypothesis that the discovery of repeated patterns within one song could be used to also successfully classify the other songs from the same tune family. However, this assumes that there are also repetitions inside a song that are important for determining the similarity of a larger group of songs. Since most of these algorithms do not even perform better than the random pattern reference experiment (see section 7.4.1), we conclude that the discovery of repeated patterns inside songs is not a good way of determining the similarity between folk songs.

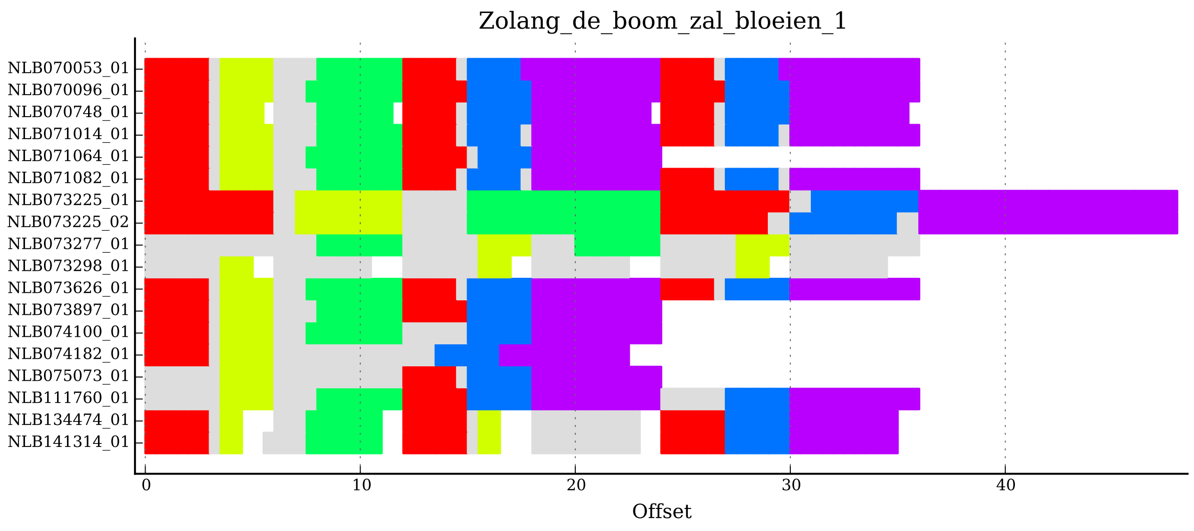


Figure 8.2.: Pattern plot for the tune family "Zolang de boom zal bloeien". Every motif class has its own color.

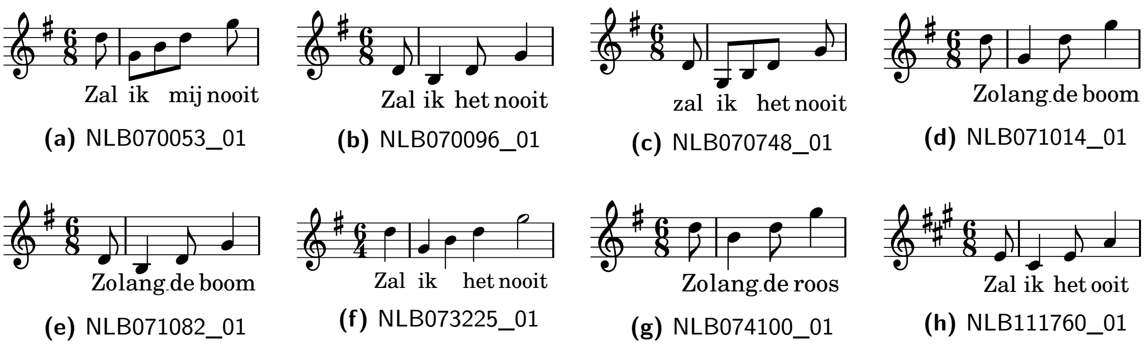


Figure 8.3.: Eight different occurrences from the same annotated motif class, the first (red) pattern in the tune class "Zolang de boom zal bloeien" (figure 8.2). Within the pattern, both rhythmic and melodic variations occur.

The second reason for comparing both approaches is to see the effect of adapting the algorithms to support Class Pattern Discovery. In section 7.4.2, we see that this approach results in higher accuracies with lower coverage values. Overall, we can conclude that Class Pattern Discovery performs better than Song Pattern Discovery. The reason for this is that in the Class Pattern Discovery task, we specifically search for patterns repeating in an entire tune class, rather than in a single song. This makes the classification easier, since there is a higher correspondence between the compressed songs from the same class when only repeated segments that occur within that class are used for classification.

This leads to the question if it would not be better to perform pattern discovery within the *complete* dataset, rather than within a single song or a tune class. Especially in the case of Class Pattern Discovery, the resulting accuracy is positively biased due to the fact that we already know the class labels beforehand. We use this information when the tune classes are concatenated (or divided into a corpus and anti-corpus in the case of MGDP). This bias could be solved by performing pattern discovery on the complete dataset, where patterns will be discovered across all pieces in the dataset despite their original tune class. We did some attempts to implement this pattern discovery approach using our framework, but due to the computational complexity in terms of both time and storage we could not run this experiment on the complete dataset and for all parameter configurations. Because the adaption of the algorithms falls outside the scope of this research, this is something that could be studied further in future research. Collins (2014b) has already done an analysis of using SIARCT-CFP for Class Pattern Discovery on Beethoven's piano sonatas.

Despite this possible bias, the Class Pattern Discovery task simulates the same pattern discovery procedure as followed by Volk and Van Kranenburg (2012) in the experiment where musicologists had to annotate the important motifs for tune classification. Here, the musicologists also knew the true class labels of the songs before annotating the important motifs.

8.3. Compression and Accuracy as Evaluation Measure

We compared many algorithms and methods to study the effect of repeated patterns for compression in determining similarity between songs. In this section, we will discuss if we could use the coverage and accuracy we retrieved from the experiments as evaluation measure. We will compare this method to the evaluation using a set of ground truth patterns which is the case in the *Discovery of Repeated Themes and Sections* MIREX task. Then, we will discuss if coverage and accuracy can be used to say which algorithm performs better than another algorithm.

8.3.1. Comparison with the MIREX Evaluation Task

Like we stated in the introduction, one of the current challenges in the study and development of Pattern Discovery algorithms is the way we evaluate them. In fact, we want to know how "well" an algorithm performs in retrieving repeated motifs. The usual way to evaluate the output of a Pattern Discovery algorithms is by comparing them with reference data, i.e. ground truth patterns. We already pointed out some problems with this approach. First, the quality of the Ground Truth depends on the experience of the annotator. Second, we are not sure that comparing the output of Pattern Discovery algorithms with a ground truth is the best way of evaluating such algorithms, since ground truth can be considered subjective.

We proposed an experiment framework to analyze the effect of the use of Pattern Discovery algorithms as compression method. Based on this, we could also evaluate an algorithm based on the resulting coverage and accuracy. This way, we do not *directly* measure the quality of a Pattern Discovery algorithm's output by comparing these patterns with a ground truth, but instead we measure the quality *indirectly* by using these patterns as compression measure in a classification task. This mostly solves the problem of subjectiveness, since the only subjective information we use is the class label assignment for each song.

Meredith (2015) studied the ground truth patterns of the *JKU Patterns Development Database* (JKU-PDD), used in the *Discovery of Repeated Themes and Sections* MIREX task in more detail. He argues that if musicologists cannot agree upon which patterns are *important* (noticeable), this kind of evaluation is subjective and incomplete. Besides, patterns that are important in this light do not necessarily have to play an important role in an *objectively evaluable task*. In this study, we build a framework to achieve such an objectively evaluable task. By using compression and classification, we have evaluated Pattern Discovery algorithms based on the characteristic patterns they discover, with respect to either a single song or a complete tune class.

On top of that, Meredith (2015) argues that we do not know if the ground truth patterns (from the JKU-PDD in this case) are also "important" in such an *objectively evaluable task*. We have clearly shown in section 7.3 that using the annotated motifs from MTC-ANN in the objectively evaluable task of folk song classification, these motifs lead to a higher accuracy than when using the output of Pattern Discovery algorithms. In other words, these ground truth patterns *are* important for this objectively evaluable task.

8.3.2. Evaluating Pattern Discovery algorithms based on Compression and Accuracy

So far, we have not compared the performance of the Pattern Discovery algorithms with each other. We saw in section 7.8 that we could compare the discovered patterns of the algorithms on MTC-ANN with the annotated motifs, where many of the annotated motifs and its occurrences were not found by most of the algorithms. In fact, we evaluated in this experiment the ability of each algorithm to find exactly the same patterns as the ones that are annotated. As we stated in the previous section, evaluating these algorithms using the coverage and accuracy we retrieved from our experiment framework could give a measure of how well an algorithm is capable of finding distinctive patterns for classification.

The problem we encountered when performing such an evaluation on the results of chapter 7, is that the *quality* of the different algorithms heavily depends on the discovery task we choose, as well as the reconstruction and similarity measure in the framework. On top of that, an evaluation measure has to be defined that combines coverage and accuracy in one evaluation function. This is not an arbitrary process, since there are multiple ways of judging a set of patterns as *better* than another set of patterns. This depends on the context of the evaluation task the algorithms are evaluated in: do we only want a high accuracy, a low coverage, or both? What we have seen in our results is that there is always a tradeoff between coverage and accuracy. Depending on how much you allow the accuracy to drop in order to get a low coverage, the *best* algorithm can differ. This means that based on the framework we presented and the algorithms we compared, we cannot make a statement which algorithm performs better than another algorithm. This is something that could be studied further, as well as more directly analyzing the differences between the annotated and the automatically discovered patterns.

Looking at these arguments and placing them in a broader perspective alongside with the criticism of Meredith (2015) about the JKU-PDD, we have to argue if a *golden evaluation standard* could exist for Pattern Discovery algorithms. In fact, the way the MIREX task is organized right now tests essentially the quality of the algorithms to find exactly the same patterns as those that are annotated. Our method is based on the fact that we can classify folk songs based on less information than the entire song. Since a number of the algorithms we used in our experiments are specifically adapted (and evaluated) for the MIREX task, we could expect that (without significant modifications) these algorithms would also perform well in the evaluation task we proposed. After all, the annotated motifs are, like the ground truth patterns from the JKU-PDD, considered to be *important*.

Conclusion 9

We have developed a framework to use Pattern Discovery algorithms as compression method to determine the similarity between folk songs. The songs are originally grouped into tune classes, of which we assume to be originating from the same song, or are variations of each other. We have used this framework to study the impact of using Pattern Discovery algorithms as a compression method for the classification of these songs. The compression is achieved by removing all notes that are not part of a pattern. The actual similarity calculation is done by an alignment algorithm.

Without compression, the classification on the entire songs already yields an accuracy between 0.87 and 0.96. Using a naive lossy compression approach, selecting the first, middle or last *n* notes in each song, the accuracy is still high even for lower coverage values. The selection of random patterns and notes perform worse. With these results, we can conclude that without the use of patterns, the classification accuracy is already very high. For some naive compression strategies, the classification accuracy still remains high for lower coverage values.

To study the effect of repeated patterns for compression and tune classification, we used our proposed framework on both the Pattern Discovery algorithms and a set of Annotated Motifs from the set of Dutch folk songs, MTC-ANN. We have shown that, using the Annotated Motifs, the classification accuracy was 0.89 using Gapless Reconstruction and Note to Note Alignment, with only a coverage of 0.4. This means that with only 40% of the notes left in each song on average, the classification accuracy has dropped only a few percentage points compared to the classification on the entire songs.

For the Pattern Discovery algorithms, we both studied the effect on the classification accuracy using Song Pattern Discovery and Class Pattern Discovery. In the latter case, we modified our framework such that algorithms specifically designed for Song Pattern Discovery could also be used for Class Pattern Discovery. For every algorithm, we tried a large set of parameter values. None of the parameter values lead to a better result in terms of coverage and accuracy than the annotated motifs from MTC-ANN. The selection of the first, middle or last *n* notes were also superior to most of the algorithm runs. We have compared the output of the Pattern Discovery algorithms on the dataset MTC-ANN with the evaluation metrics used in the *Discovery of Repeated Themes and Sections* MIREX task. Using these metrics, we can conclude that the best algorithm yields a precision and recall score of at most 0.7. The comparison of the Pattern Discovery output with the Annotated Motifs suggest that the *position* of a pattern occurrence in the piece is important for the pattern.

To conclude, repeated patterns can be used for compression and classification, but there is a large gap in performance between the classification accuracy using the annotated motifs and the state-of-the-art Pattern Discovery algorithms. We have learned from the Annnotated Motifs that the *position* of a pattern in the song can be important for filtering and categorizing pattern occurrences, as well as supporting the discovery of inexact pattern occurrences. Alongside with the MIREX task on the Discovery of Repeated Patterns, compression and classification accuracy can be used as evaluation measure on the output of Pattern Discovery algorithms, although the determination of the best performing algorithm depends on the context the algorithms are evaluated in. For future research, more Pattern Discovery algorithms, compression methods and similarity measures can be added to the framework to support music from other domains and to further investigate the role of patterns in compression and classification. Because the output size of some algorithms is still large, a reconstruction method that discards patterns with a low distinctiveness can be a valuable improvement. Our proposed evaluation method can help to develop Pattern Discovery algorithms that discover patterns in a broader context than just testing how *noticeable* a pattern is against a set of Ground Truth patterns. This opens the door to further improving and developing Pattern Discovery algorithms.

Acknowledgments

Although writing a thesis is something you have to do on your own, many people have provided me with suggestions, advice and feedback that have helped me to write this thesis. First of all, I would like to express my gratitude to my supervisors Anja Volk and Bas de Haas for all their feedback and all the new things I have learned. During our meetings, I became convinced that having two "first" supervisors is a privilege: Anja always came up with relevant ideas and related work. And if I got lost in all my ideas about new possibilities and interesting things that could be worth researching further, Bas got me back on track. I am also thankful for the possibility to attend two days of the Lorentz Workshop on Music Similarity in Leiden, where I could discuss this project with other researchers in the field. I also want to thank Frans Wiering for the supervision at the very beginning of this project and the valuable feedback on this thesis as second examiner.

Second, I want to thank Berit Janssen and Peter van Kranenburg from the Meertens Institute for their feedback and ideas, and for providing the Annotated Motifs from MTC-ANN. I also want to thank Tom Collins for the fruitful feedback at the beginning of this project and the clarifications of the PattDisc Matlab code. I like to thank Marcelo Rodríguez López and Dimitrios Bountouridis for the discussions we had and the suggestions and ideas they provided me with. Thanks to Rick Barneveld and Gijs Boosten for proof reading this thesis and providing me with valuable feedback.

Finally, I want to thank a number of people very special to me. First, I would like to thank my parents for their advice and support. Not only in this project, but also in the past seven years. And last but not least, I want to thank my girlfriend Jolien for all her advice and motivation and always being there for me.

fine

Dataset Contents A

A.1. MTC-ANN

Tune class (family)	# Songs	Song IDs
Daar ging een heer 1	16	072587_01, 072587_02, 072774_02, 073046_01, 073588_01, 073672_01, 073681_01, 073743_01, 073822_01, 074004_01, 074048_02, 074227_01, 075551_01, 076625_01, 076632_01, 144072_01
Daar reed een jonkheer 1	12	070801_01, 072154_01, 072912_01, 072920_01, 072946_01, 073426_01, 073929_01, 074028_01, 111656_01, 112210_01, 141251_01, 144042_01
Daar was laatstmaal een ruiter 2	17	070493_01, 072003_01, 072015_01, 072627_01, 072690_01, 072708_01, 073287_01, 073287_02, 073337_02, 073483_01, 073628_01, 073639_01, 073990_01, 074328_01, 074427_01, 074552_01, 076211_01
Daar zou een maagdje vroeg opstaan 2	10	015569_01, 071441_01, 071666_01, 072299_01, 072306_01, 072311_01, 072886_01, 072886_02, 073150_01, 138219_01
Een lindeboom stond in het dal	13	073120_01, 073269_02, 073324_01, 073709_01, 074308_01, 074378_01, 074452_01, 074547_02, 075273_01, 075635_01, 076426_01, 076740_01, 124573_01
Een Soudaan had een dochtertje 1	9	070089_01, 070141_01, 070144_01, 070740_01, 071958_01, 073804_02, 073866_01, 074754_01, 141648_01
En er waren eens twee zoeteliefjes	16	070134_01, 072585_01, 072638_01, 072823_01, 073296_01, 073331_01, 074437_01, 074533_01, 074583_01, 074649_01, 075018_02, 075040_01, 075059_01, 075174_01, 075249_01, 075612_0
Er reed eens een ruiter 1	27	070996_01, 071957_03, 072553_01, 072559_01, 072813_01, 072837_01, 072851_01, 072851_02, 072862_01, 072883_01, 072895_01, 072898_01, 072898_02, 073076_01, 073333_01, 074246_01, 074246_02, 074349_01, 074433_01, 074575_01, 075176_01, 075184_01, 075325_01, 075325_02, 145525_01, 146608_01, 162684_01
Er was een herderinnetje 1	11	070238_01, 070521_01, 071016_01, 072497_01, 073339_01, 075309_02, 075318_01, 112115_01, 141407_01, 141649_01, 146728_01
Er was een koopman rijk en machtig	17	070079_01, 070122_01, 070475_01, 070606_01, 072441_01, 072967_01, 073031_01, 073146_01, 073788_01, 073998_01, 074948_01, 075013_01, 075313_01, 075431_01, 146699_01, 151180_01, 152778_01
Er was een meisje van zestien jaren 1	15	072355_01, 072355_02, 072355_12, 072356_02, 072357_01, 072358_01, 072359_01, 072360_01, 072457_01, 073775_02, 074260_01, 074260_02, 074334_01, 074336_01, 147463_01
Er woonde een vrouwtje al over het bos	12	073862_01, 074309_01, 074443_01, 074593_01, 075739_03, 075742_01, 075771_02, 075848_01, 075881_02, 076076_01, 076258_01, 076271_01
Femmes voulez-vous éprouver	13	070526_01, 072450_01, 072871_01, 074286_01, 074390_01, 074470_01, 075034_01, 075906_01, 076118_01, 076495_01, 111478_01, 125421_01, 146731_01
Heer Halewijn 2	11	072254_01, 072255_01, 072257_01, 072378_01, 073750_01, 073754_01, 073994_01, 074156_03, 074261_01, 074426_02, 074613_02
Heer Halewijn 4	11	072248_01, 072250_01, 072253_01, 072256_01, 073326_01, 074003_01, 074216_01, 074333_01, 074603_01, 143240_01, 147912_01
Het vrouwtje van Stavoren 1	8	070078_01, 070125_01, 070360_01, 070535_01, 070693_01, 071227_01, 072237_01, 072500_01

(continued from previous page)

Tune class (family)	# Songs	Song IDs
Het was laatst op een zomerdag	17	072482_01, 072505_01, 072567_01, 072664_01, 072754_01, 073516_01, 073777_01, 073946_01, 074104_01, 074166_01, 074342_01, 074938_01, 074956_01, 075065_01, 075074_01, 075532_01, 075616_01
Het was op een driekoningenavond 1	12	070033_01, 071669_01, 071974_02, 072382_02, 072614_01, 072647_01, 072881_01, 073486_01, 074276_01, 074277_01, 075063_01, 075367_01
Ik kwam laatst eens in de stad	18	070137_01, 071478_01, 072721_01, 073685_01, 073803_01, 073879_02, 073895_01, 073958_01, 074077_02, 074672_01, 074769_01, 074769_02, 074840_01, 074840_02, 074860_01, 075191_01, 075831_01, 076303_01
Kom laat ons nu zo stil niet zijn 1	11	072085_01, 073404_01, 073562_01, 073939_01, 075021_01, 075035_01, 075057_01, 075156_01, 075167_01, 075379_01, 075525_01
Lieve schipper vaar me over 1	15	070411_01, 070463_01, 070532_01, 070839_01, 071237_01, 072897_01, 073374_01, 076128_01, 111484_01, 112233_01, 125427_01, 134480_01, 135273_01, 145856_01, 167193_01
O God ik leef in nood	8	072624_01, 072968_01, 074007_01, 074038_01, 075079_01, 075307_03, 146741_01, 152784_01
Soldaat kwam uit de oorlog	17	072103_01, 072283_01, 072284_01, 072285_01, 072286_01, 072287_01, 072288_01, 072289_01, 073311_01, 073393_01, 073888_01, 073992_01, 074157_01, 074161_01, 074234_01, 074468_01, 074954_01
Vaarwel bruidje schoon	11	070732_01, 071369_01, 072499_01, 073210_01, 075158_01, 076130_07, 112123_01, 134389_01, 144100_01, 162519_01, 162526_01
Wat zag ik daar van verre 1	15	070492_01, 071944_01, 072565_01, 072665_01, 072688_01, 072691_01, 072714_01, 073286_01, 073304_01, 073771_02, 073991_02, 073997_01, 074962_01, 075064_01, 075068_01
Zolang de boom zal bloeien 1	18	070053_01, 070096_01, 070748_01, 071014_01, 071064_01, 071082_01, 073225_01, 073225_02, 073277_01, 073298_01, 073626_01, 073897_01, 074100_01, 074182_01, 075073_01, 111760_01, 134474_01, 141314_01

A.2. Irish Folk Tunes Dataset

Tune class (song title)	# Songs	Tune ID	Session IDs
Cooley's	10	1	1, 12342, 12343, 20796, 20960, 21423, 22061, 23915, 24552, 25140
This Is My Love, Do You Like Her	16	6	6, 1680, 1791, 8734, 12117, 12355, 12356, 12357, 12358, 12359, 12360, 12361, 15237, 23497, 24613, 24614
Merrily Kissed The Quaker	9	70	70, 12535, 12536, 12538, 12539, 12540, 24522, 24550, 25966
Miss McLeod's	17	75	75, 12552, 12553, 12554, 12555, 12556, 12557, 12558, 12560, 21434, 23677, 24548, 24681, 25452, 25695, 26044, 26249
Rakish Paddy	9	86	86, 12601, 12602, 12604, 12605, 12606, 12607, 22967, 22975
Humours Of Trim, The	12	88	88, 12618, 12619, 12620, 12621, 12622, 12623, 12624, 12625, 20918, 26362, 26363
Irish Washerwoman, The	12	92	92, 12633, 12635, 12636, 12637, 12638, 12639, 12640, 21778, 25137, 25138, 25139
Out On The Ocean	14	108	108, 12682, 12683, 12684, 12685, 12686, 20578, 21011, 21012, 21013, 21044, 21048, 25910, 25911
Wind That Shakes The Barley, The	14	116	116, 12707, 12708, 12709, 12710, 12712, 12713, 12714, 12715, 12716, 12717, 22309, 23706, 25485
Star Of Munster, The	10	197	197, 12854, 12855, 12856, 12857, 12858, 12859, 24524, 26183, 26184
Tam Lin	14	248	248, 12953, 12954, 12958, 12959, 12960, 12961, 12963, 12964, 20856, 20857, 20858, 21721, 23449
Jenny Dang The Weaver	10	380	380, 13203, 13204, 23195, 23196, 23198, 23199, 23200, 23907, 24106
Pigeon On The Gate, The	13	517	517, 13448, 13449, 13450, 13451, 13452, 13453, 13454, 13455, 13456, 23096, 23100, 24503
Garrett Barry's	9	544	544, 13502, 13503, 13504, 13505, 13506, 21269, 25988, 26003
Tap Room, The	13	711	711, 13772, 13773, 13774, 13775, 13776, 13777, 13778, 13779, 13780, 13781, 20956, 24969
Paddy O'Rafferty	10	741	741, 13824, 13825, 13826, 13827, 13828, 24994, 25048, 26002, 26388
Fisher's	11	872	872, 1322, 14043, 14044, 14045, 14046, 14047, 14048, 20909, 24434, 26307
Off She Goes	13	1133	1133, 14400, 14401, 23310, 24472, 24473, 24474, 24475, 24476, 24477, 24478, 24483, 24608
Drag Her Round The Road	12	1148	1148 (2x), 14414, 14415, 14416, 14417, 14418, 14419, 14420, 22048, 25020, 25022
If We Hadn't Any Women In The World	12	1376	1376, 14732, 14733, 14734, 14735, 14736, 14737, 14738, 14739, 14740, 14741, 21628
Road To Skye, The	11	1709	1709, 15130, 15131, 15132, 15133, 15134, 15135, 15136, 15137, 15138, 24825
Shoe The Donkey	15	2320	2320, 15686, 15687, 15688, 15689, 15690, 15691, 15692, 15693, 15694, 15695, 15696, 25176, 25748, 25748
Flowers Of Edinburgh, The	9	2549	2549, 15820, 15821, 15822, 15823, 15824, 15825, 15826, 15828
Wren, The	10	2828	2828, 16033, 16034, 16035, 16036, 16037, 16038, 16039, 16040, 16041
Pearl O'Shaughnessy's	14	4321	4321, 17016, 17017, 17018, 17019, 17020, 20845, 20846, 20847, 20848, 20849, 20850, 20851, 20852
Lochiel's Awa' Tae France	10	6887	6887, 18462, 18463, 18464, 18465, 18467, 18468, 18469, 18470, 26295
Starry Nights Of Shetland, The	8	7357	7357, 18873, 18874, 18875, 18876, 18877, 18878, 18879
Lovely Nancy	16	7423	7423, 18908, 18909, 18910, 18911, 18912, 18913 (2x), 18914, 18915, 18917 (6x)
She Hasn't The Thing She Thought She Had	12	7467	7467, 18948, 18949, 18950, 18951, 18952, 18953, 18954, 18955, 18956, 18957, 26227
Planxty Burke	11	10039	10039, 20171, 20172, 20173, 20175, 20176, 20177, 20178, 20179, 20180, 20182

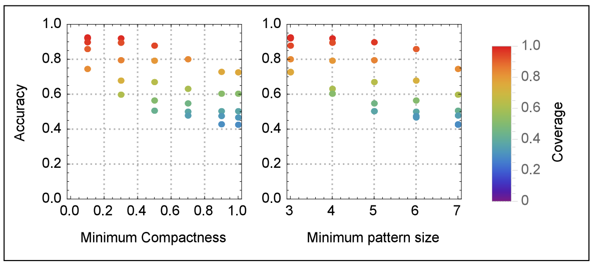
Parameter configurations B

In this appendix, the effect of the various algorithm parameters are visualized for all combinations of pattern discovery tasks, filtering methods, alignment algorithms and datasets. In each graph, the parameter values are plotted against the accuracy. The color of each dot represents the corresponding coverage. Note that combinations of parameters can be compared by looking at the other graphs at the same height and color. This raw data where these plots are based on can be found in appendix B.

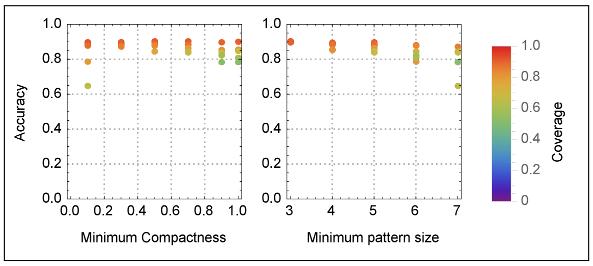
For each combination of reconstruction method and similarity method, the graphs are plotted. Since the output of the Gapless Reconstruction and Onset Preserved Reconstruction is exactly the same, we will only use the name of the Gapless Reconstruction method. This appendix is structured as follows:

- Gapless Reconstruction and Note to Note Alignment: section B.1
- Gapless Reconstruction and Melodic Shape Alignment: section B.2
- First Pattern Occurrence Reconstruction and Note to Note Alignment: section B.3
- First Pattern Occurrence Reconstruction and Melodic Shape Alignment: section B.4

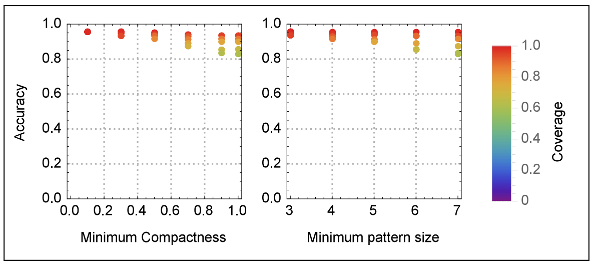
B.1. Gapless Reconstruction, Note to Note Alignment



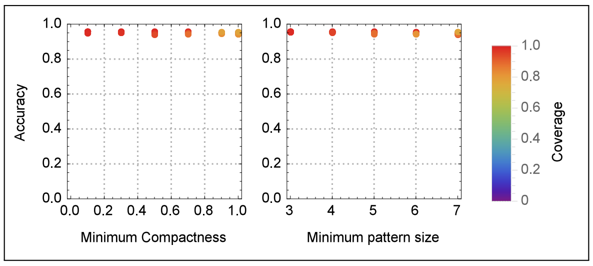
(a) Song Pattern Discovery on MTC-ANN



(b) Class Pattern Discovery on MTC-ANN

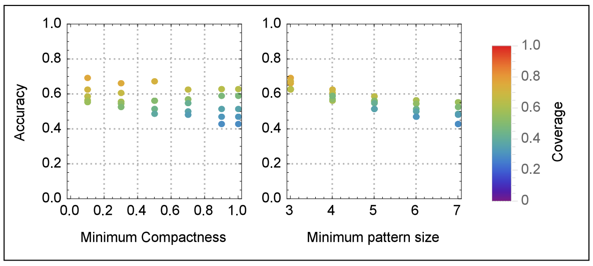


(c) Song Pattern Discovery on the Irish Folk Tunes Dataset

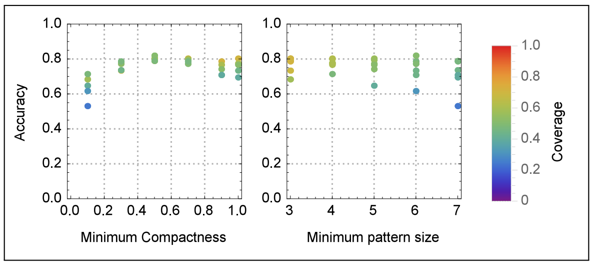


(d) Class Pattern Discovery on the Irish Folk Tunes Dataset

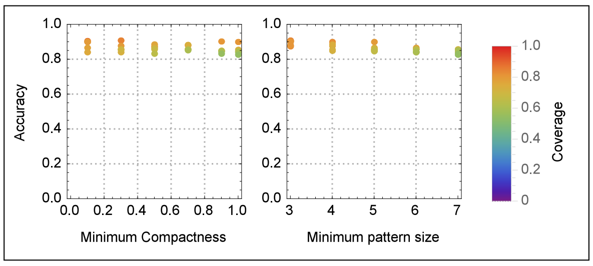
Figure B.1.: SIACTTEC



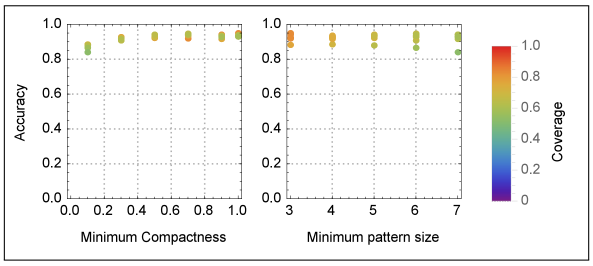
(a) Song Pattern Discovery on MTC-ANN



(b) Class Pattern Discovery on MTC-ANN

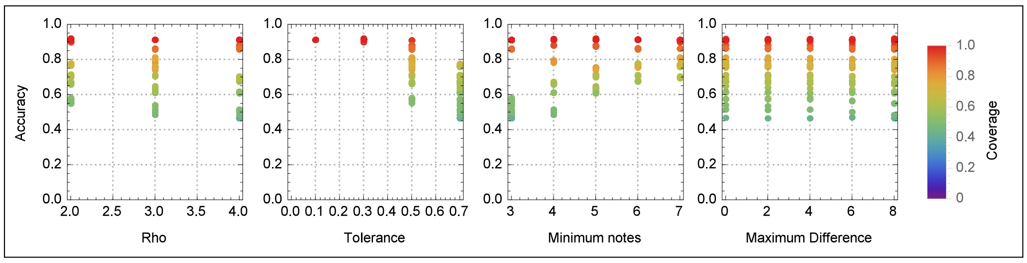


(c) Song Pattern Discovery on the Irish Folk Tunes Dataset

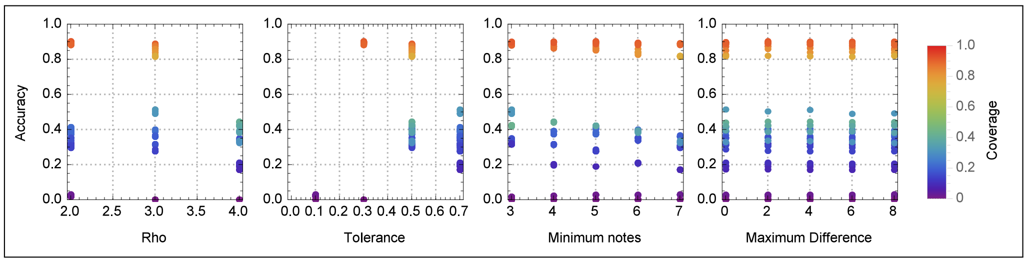


(d) Class Pattern Discovery on the Irish Folk Tunes Dataset

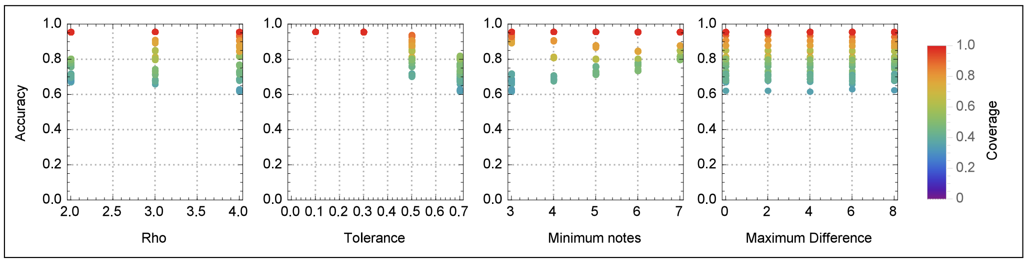
Figure B.2.: COSIACTTEC



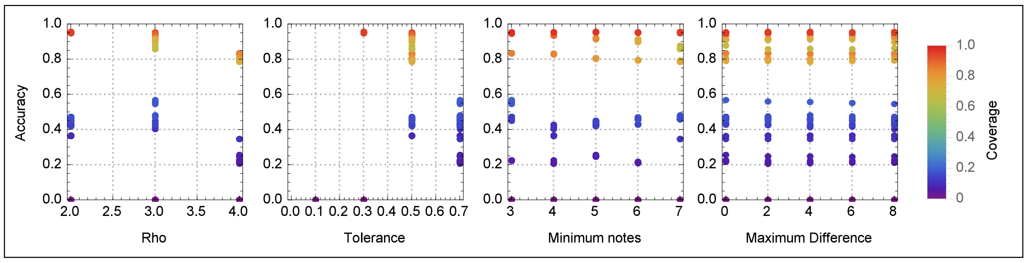
(a) Song Pattern Discovery on MTC-ANN



(b) Class Pattern Discovery on MTC-ANN

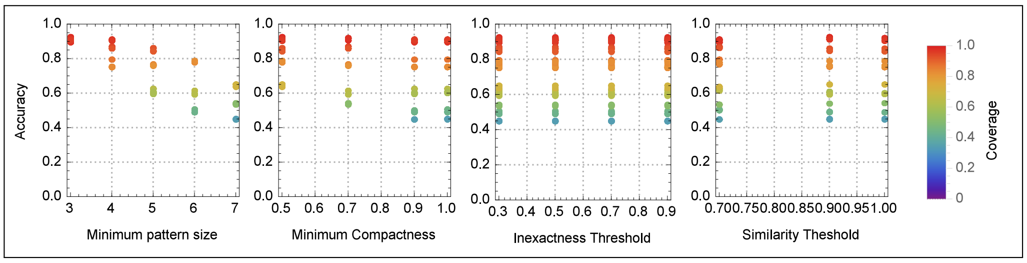


(c) Song Pattern Discovery on the Irish Folk Tunes Dataset

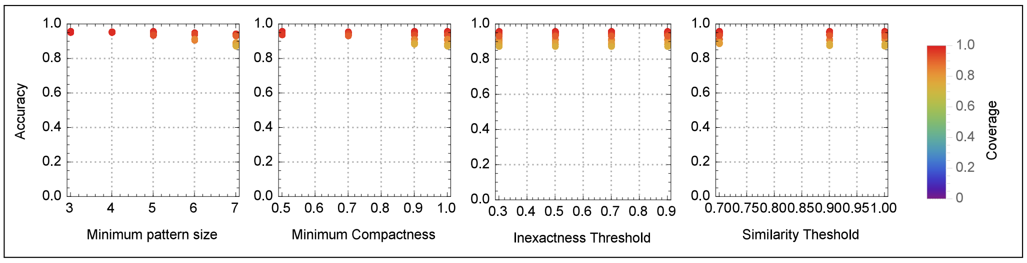


(d) Class Pattern Discovery on the Irish Folk Tunes Dataset

Figure B.3.: MotivesExtractor

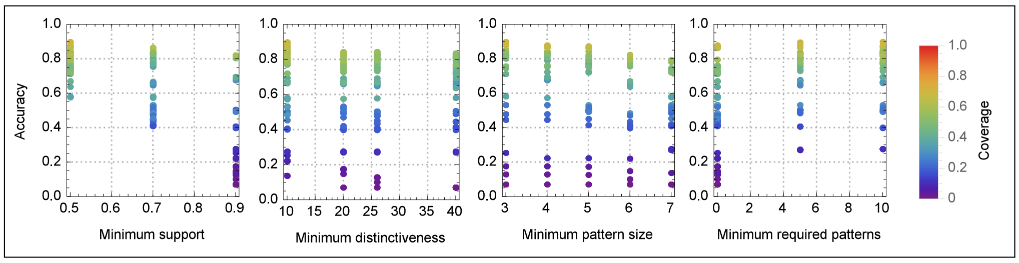


(a) Song Pattern Discovery on MTC-ANN

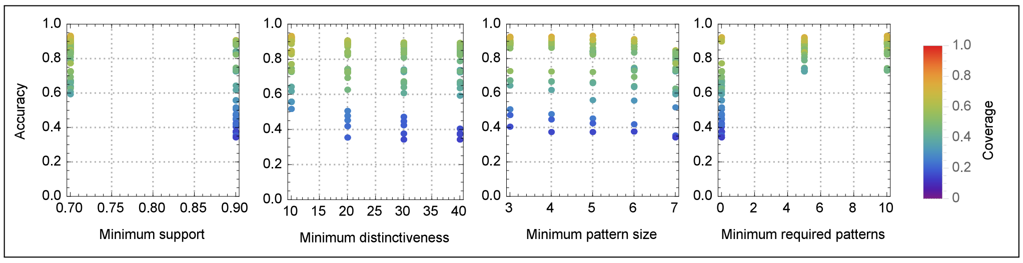


(b) Song Pattern Discovery on the Irish Folk Tunes Dataset

Figure B.4.: SIARCT-CFP

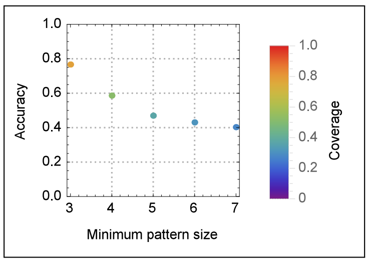


(a) Class Pattern Discovery on MTC-ANN

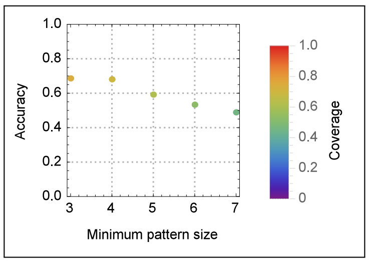


(b) Class Pattern Discovery on the Irish Folk Tunes Dataset

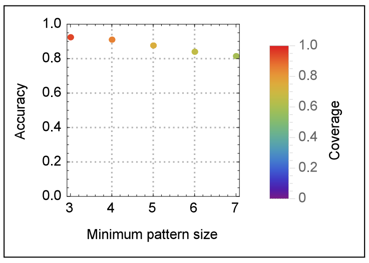
Figure B.5.: MGDP Algorithm



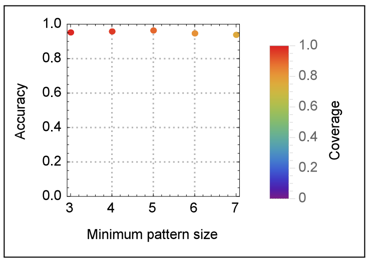
(a) Song Pattern Discovery on MTC-ANN



(b) Class Pattern Discovery on MTC-ANN



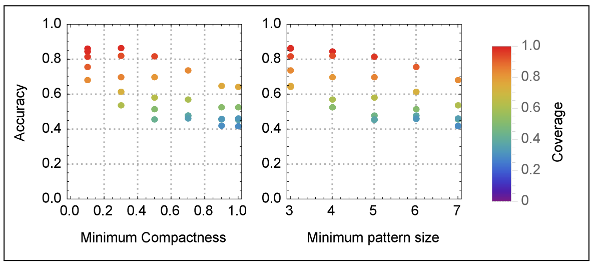
(c) Song Pattern Discovery on the Irish Folk Tunes Dataset



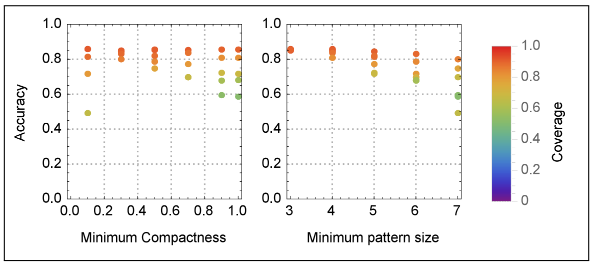
(d) Class Pattern Discovery on the Irish Folk Tunes Dataset

Figure B.6.: PatMinr

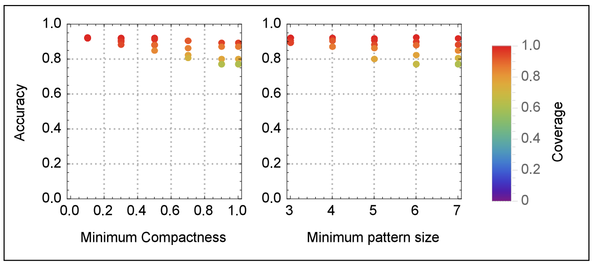
B.2. Gapless Reconstruction, Melodic Shape Alignment



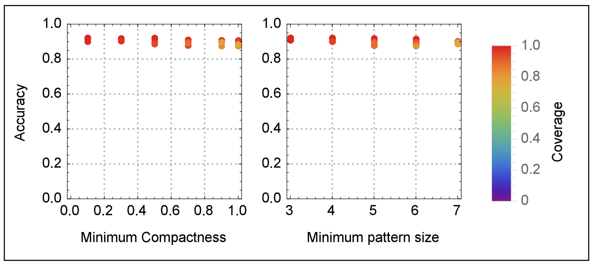
(a) Song Pattern Discovery on MTC-ANN



(b) Class Pattern Discovery on MTC-ANN

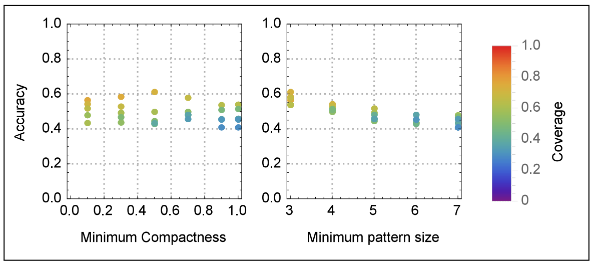


(c) Song Pattern Discovery on the Irish Folk Tunes Dataset

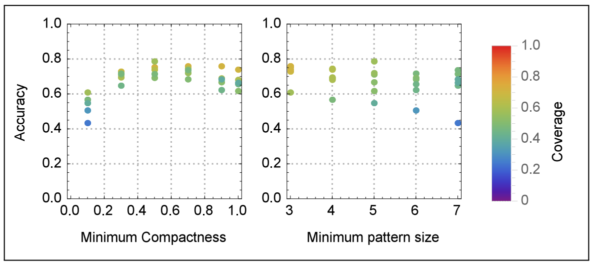


(d) Class Pattern Discovery on the Irish Folk Tunes Dataset

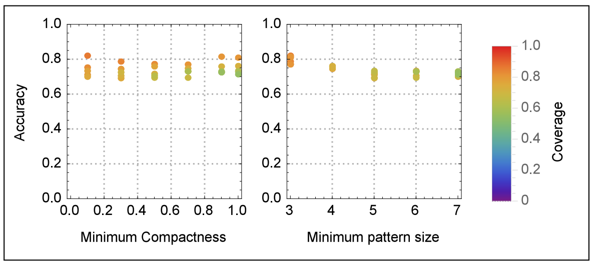
Figure B.7.: SIACTTEC



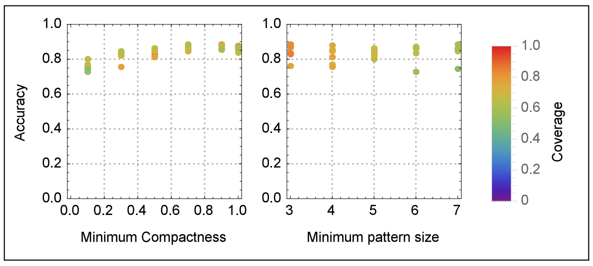
(a) Song Pattern Discovery on MTC-ANN



(b) Class Pattern Discovery on MTC-ANN

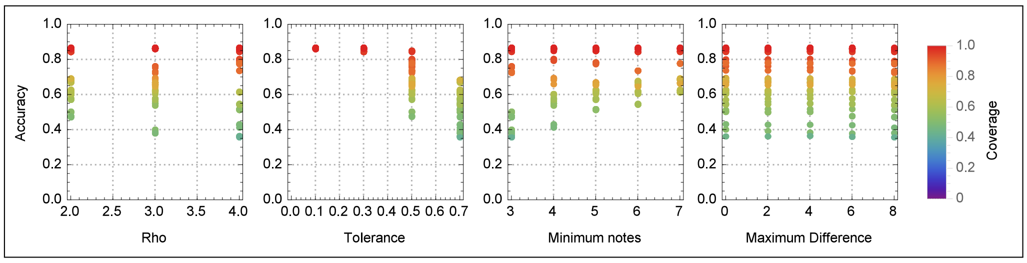


(c) Song Pattern Discovery on the Irish Folk Tunes Dataset

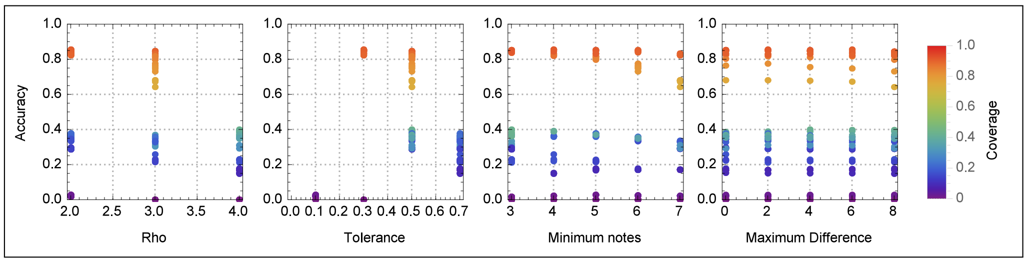


(d) Class Pattern Discovery on the Irish Folk Tunes Dataset

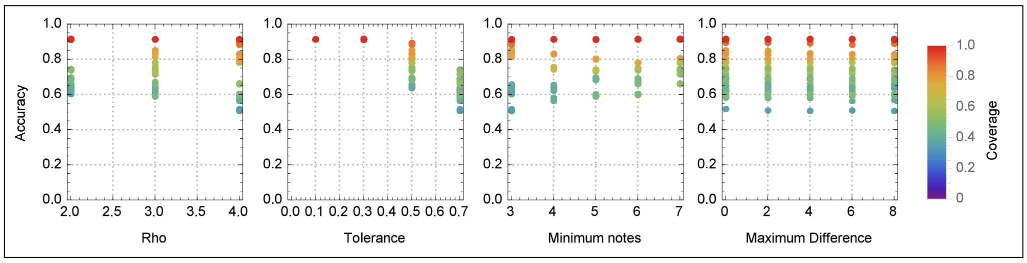
Figure B.8.: COSIACTTEC



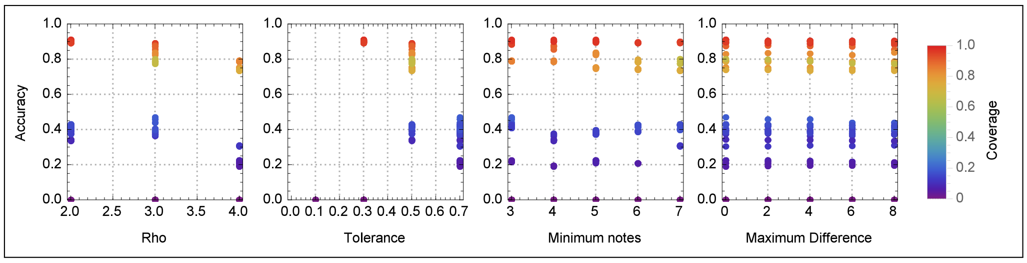
(a) Song Pattern Discovery on MTC-ANN



(b) Class Pattern Discovery on MTC-ANN

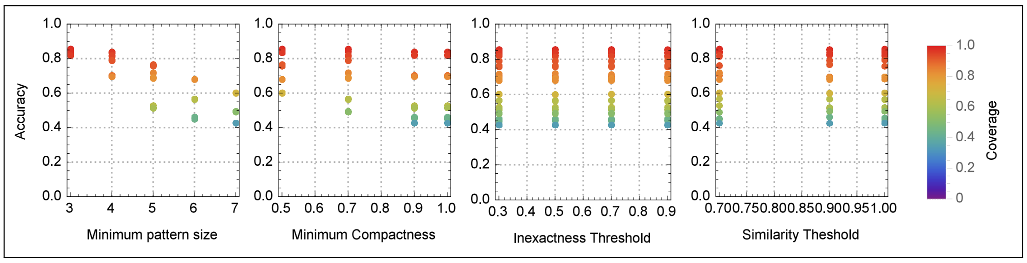


(c) Song Pattern Discovery on the Irish Folk Tunes Dataset

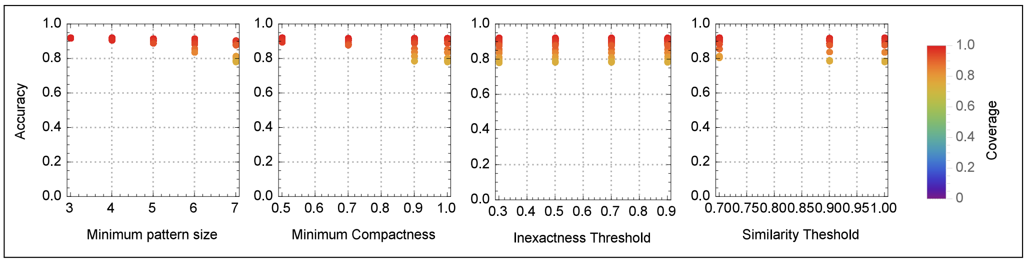


(d) Class Pattern Discovery on the Irish Folk Tunes Dataset

Figure B.9.: MotivesExtractor

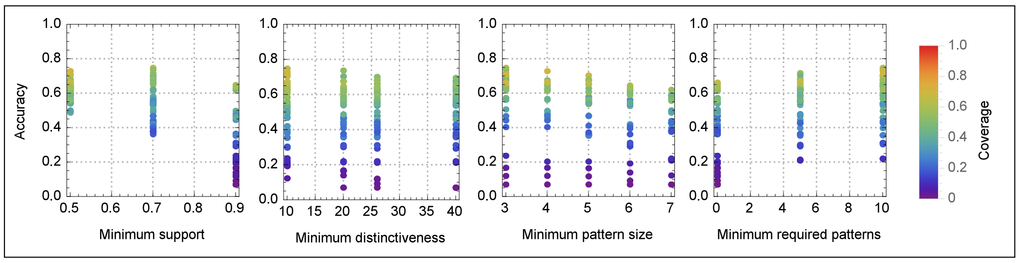


(a) Song Pattern Discovery on MTC-ANN

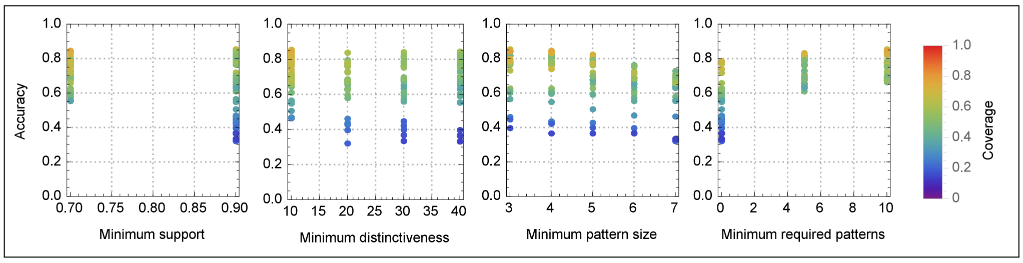


(b) Song Pattern Discovery on the Irish Folk Tunes Dataset

Figure B.10.: SIARCT-CFP

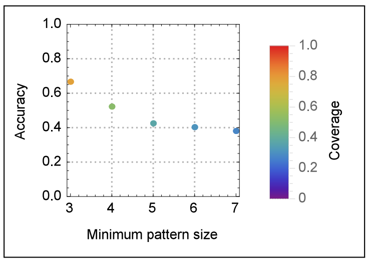


(a) Class Pattern Discovery on MTC-ANN

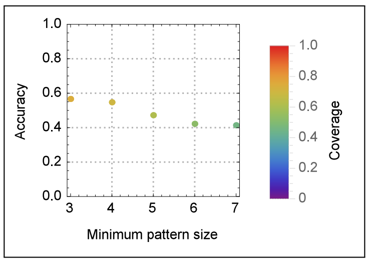


(b) Class Pattern Discovery on the Irish Folk Tunes Dataset

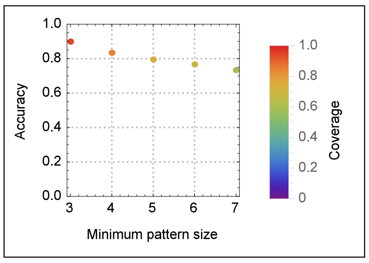
Figure B.11.: MGDP Algorithm



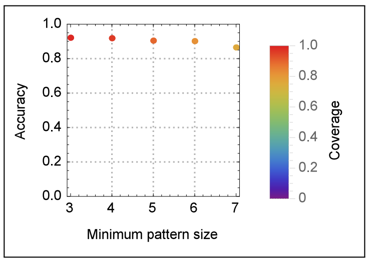
(a) Song Pattern Discovery on MTC-ANN



(b) Class Pattern Discovery on MTC-ANN



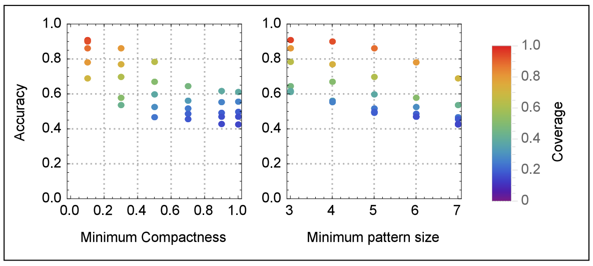
(c) Song Pattern Discovery on the Irish Folk Tunes Dataset

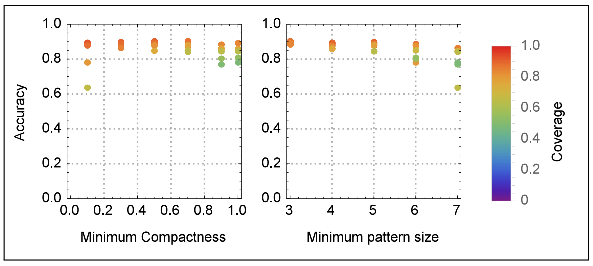


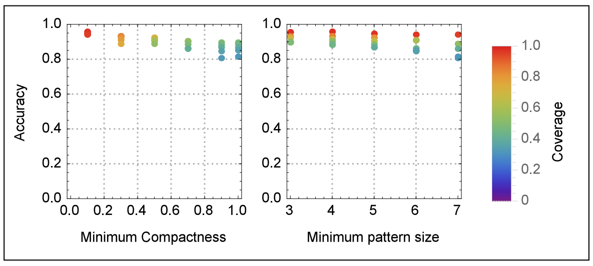
(d) Class Pattern Discovery on the Irish Folk Tunes Dataset

Figure B.12.: PatMinr

B.3. First Pattern Occurrence Reconstruction, Note to Note Alignment


(a) Song Pattern Discovery on MTC-ANN


(b) Class Pattern Discovery on MTC-ANN


(c) Song Pattern Discovery on the Irish Folk Tunes Dataset

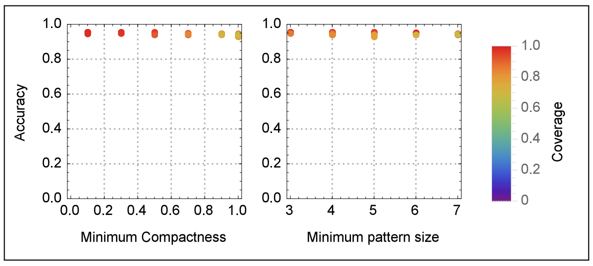
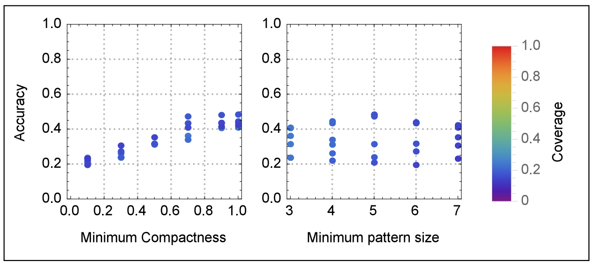
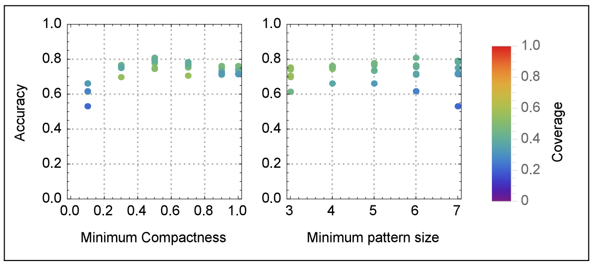
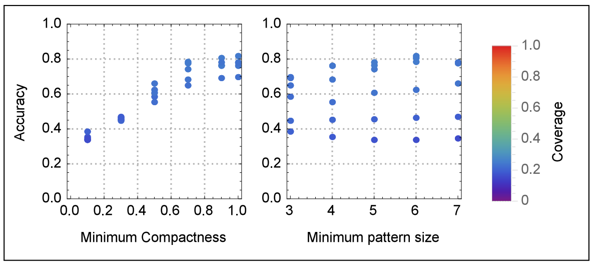

(d) Class Pattern Discovery on the Irish Folk Tunes Dataset

Figure B.13.: SIACTTEC


(a) Song Pattern Discovery on MTC-ANN


(b) Class Pattern Discovery on MTC-ANN


(c) Song Pattern Discovery on the Irish Folk Tunes Dataset

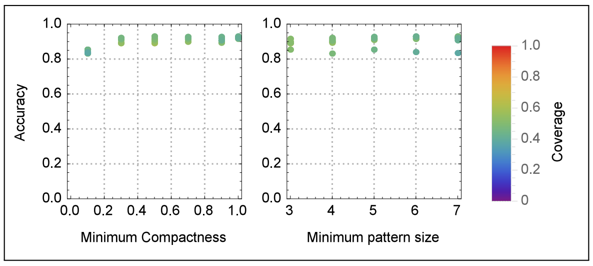
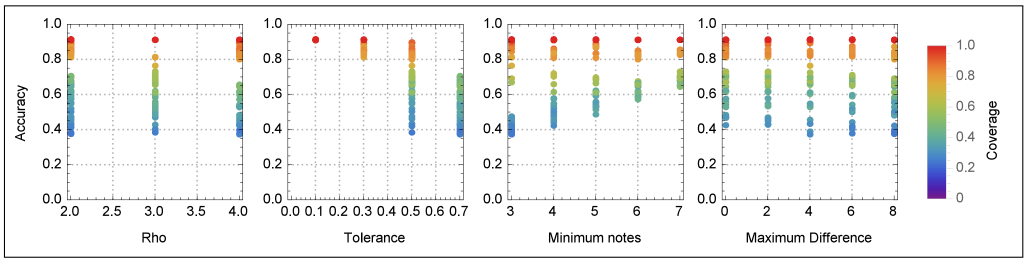
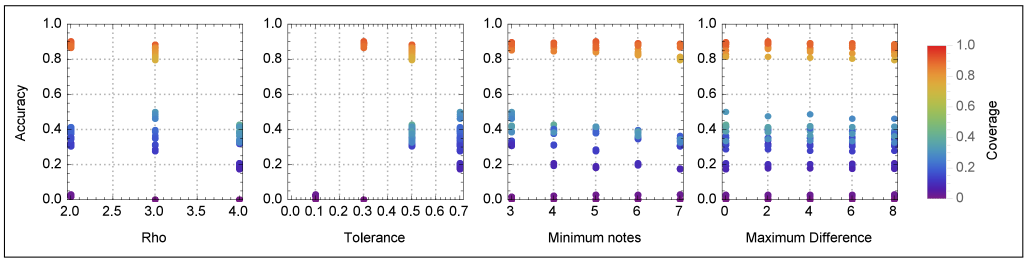

(d) Class Pattern Discovery on the Irish Folk Tunes Dataset

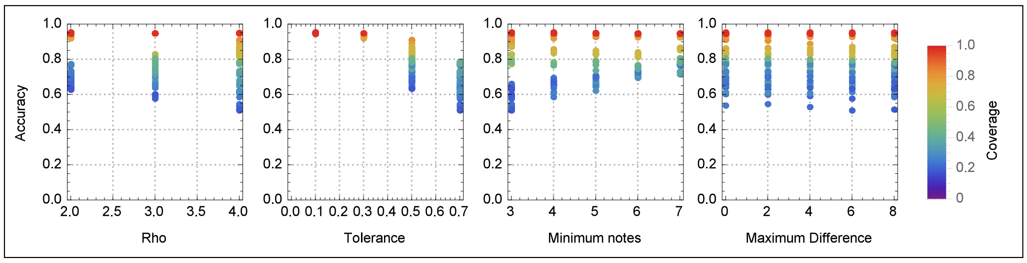
Figure B.14.: COSIACTTEC



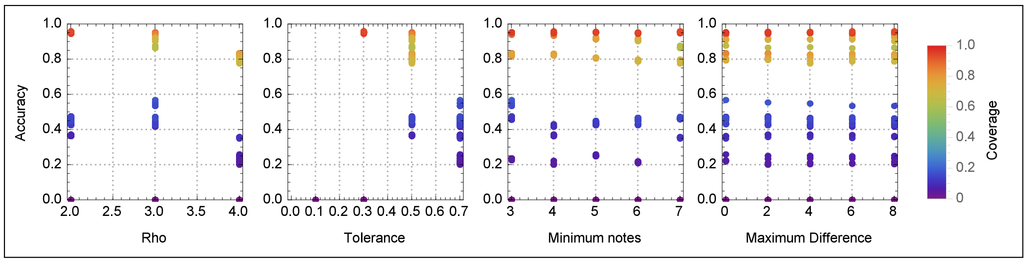
(a) Song Pattern Discovery on MTC-ANN



(b) Class Pattern Discovery on MTC-ANN

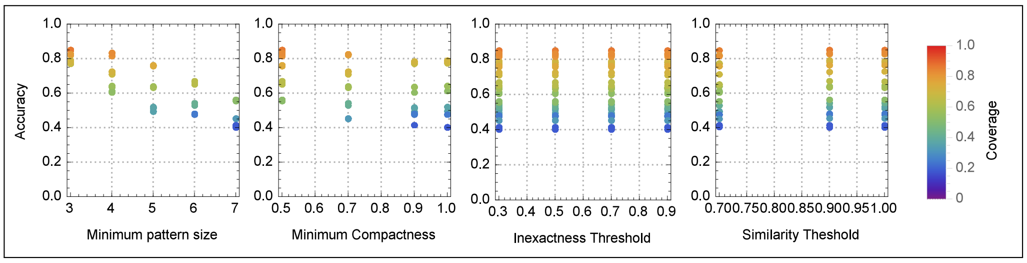


(c) Song Pattern Discovery on the Irish Folk Tunes Dataset

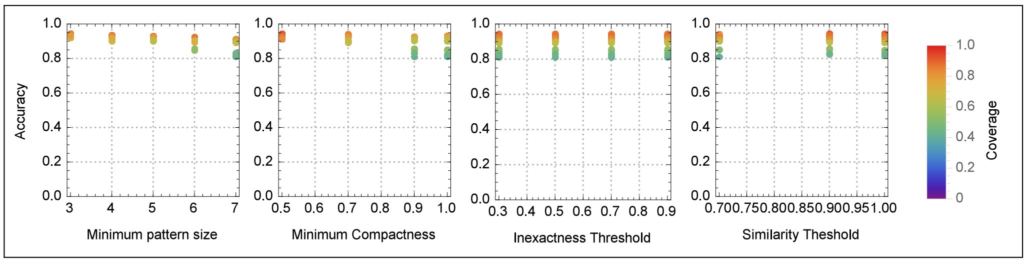


(d) Class Pattern Discovery on the Irish Folk Tunes Dataset

Figure B.15.: MotivesExtractor

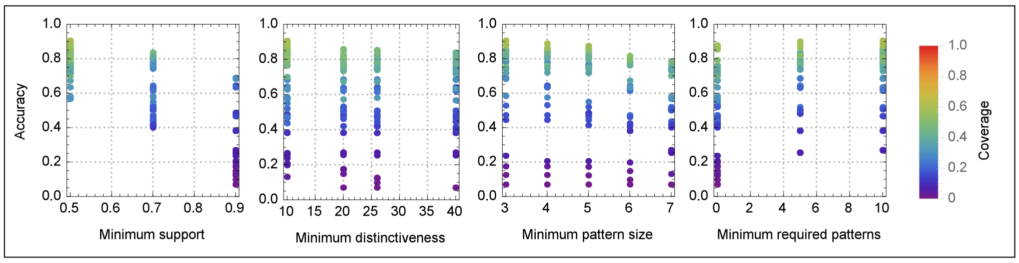


(a) Song Pattern Discovery on MTC-ANN

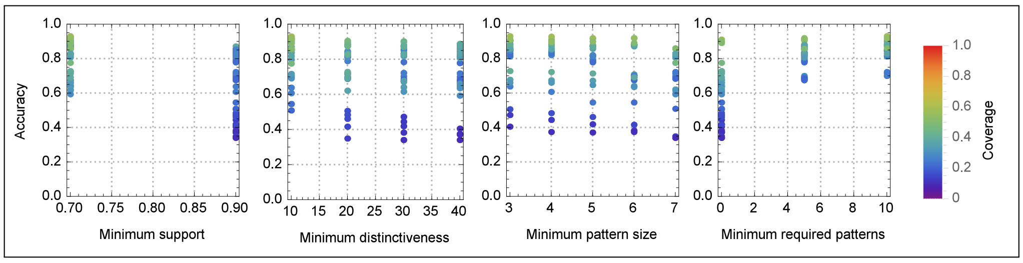


(b) Song Pattern Discovery on the Irish Folk Tunes Dataset

Figure B.16.: SIARCT-CFP

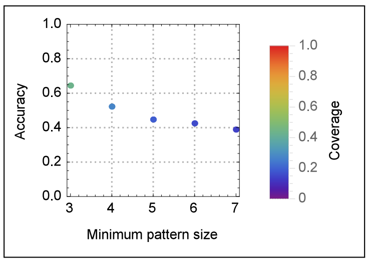


(a) Class Pattern Discovery on MTC-ANN

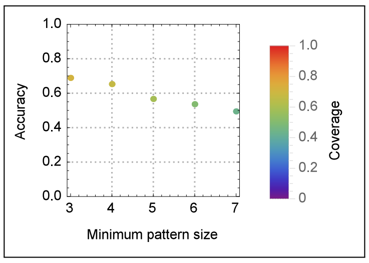


(b) Class Pattern Discovery on the Irish Folk Tunes Dataset

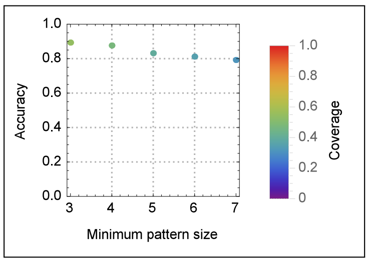
Figure B.17.: MGDP Algorithm



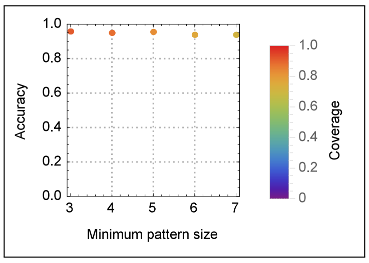
(a) Song Pattern Discovery on MTC-ANN



(b) Class Pattern Discovery on MTC-ANN



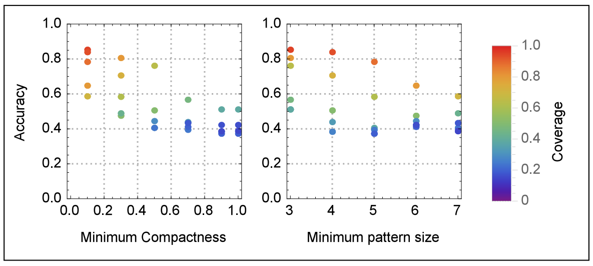
(c) Song Pattern Discovery on the Irish Folk Tunes Dataset



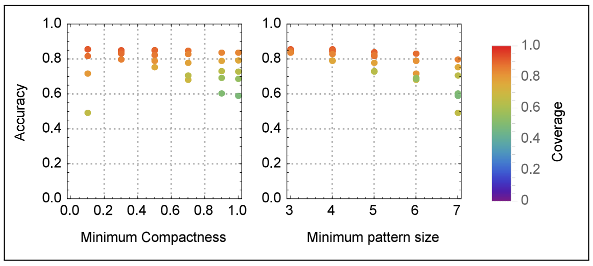
(d) Class Pattern Discovery on the Irish Folk Tunes Dataset

Figure B.18.: PatMinr

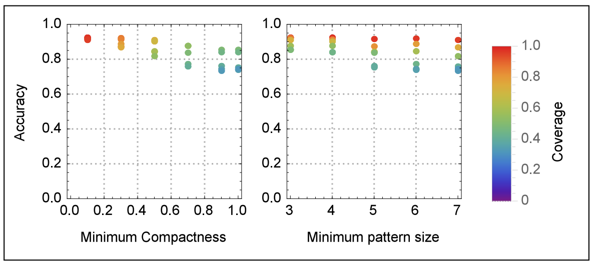
B.4. First Pattern Occurrence Reconstruction, Melodic Shape Alignment



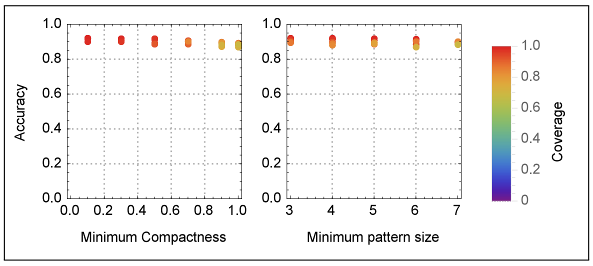
(a) Song Pattern Discovery on MTC-ANN



(b) Class Pattern Discovery on MTC-ANN

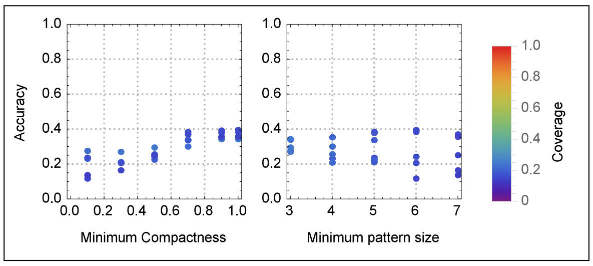


(c) Song Pattern Discovery on the Irish Folk Tunes Dataset

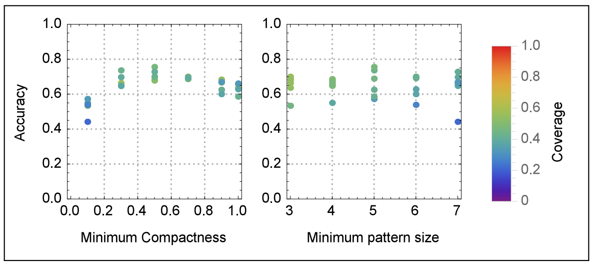


(d) Class Pattern Discovery on the Irish Folk Tunes Dataset

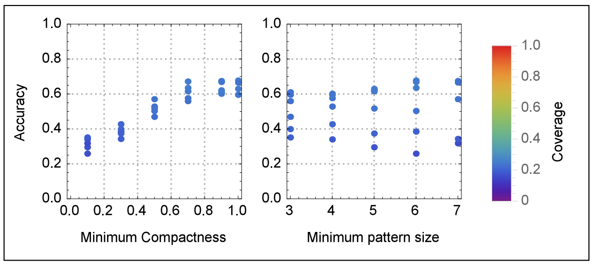
Figure B.19.: SIACTTEC



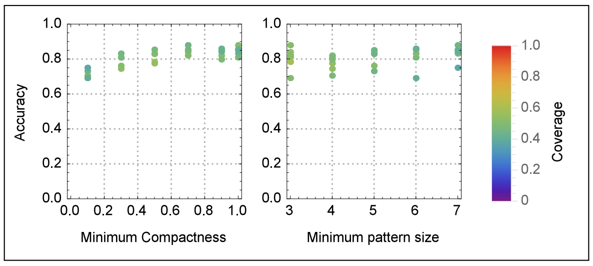
(a) Song Pattern Discovery on MTC-ANN



(b) Class Pattern Discovery on MTC-ANN

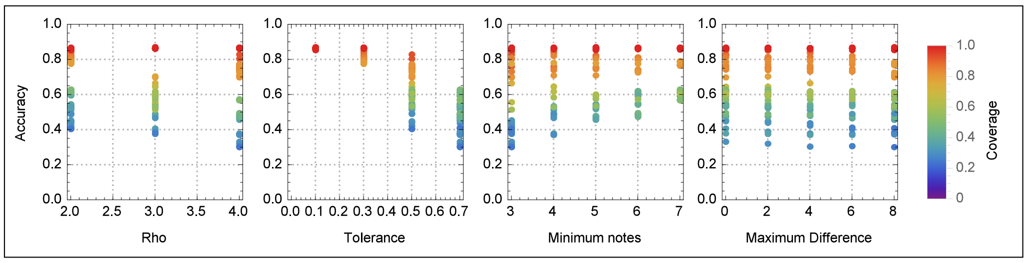


(c) Song Pattern Discovery on the Irish Folk Tunes Dataset

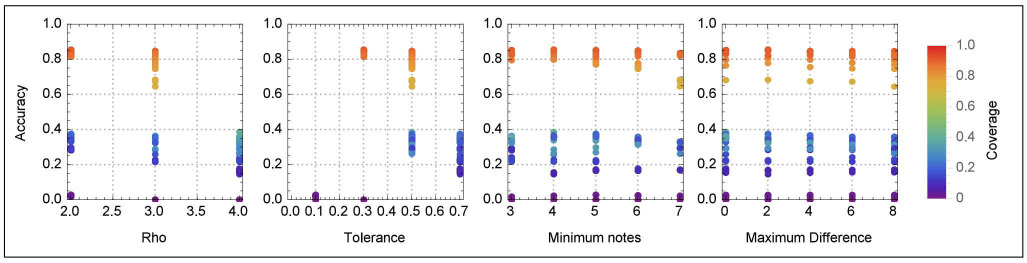


(d) Class Pattern Discovery on the Irish Folk Tunes Dataset

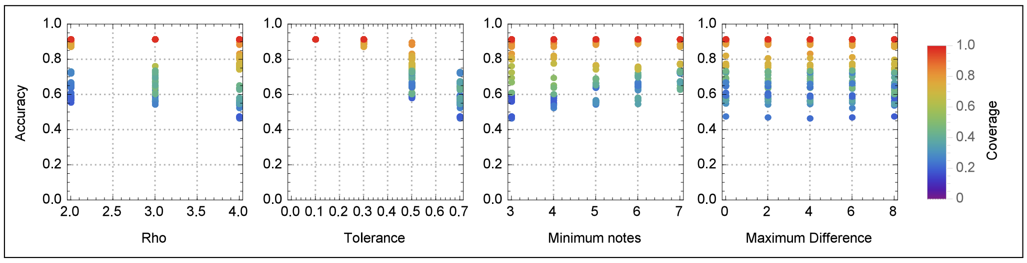
Figure B.20.: COSIACTTEC



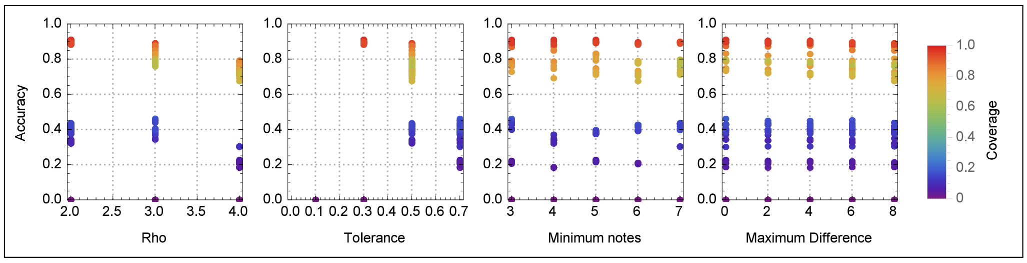
(a) Song Pattern Discovery on MTC-ANN



(b) Class Pattern Discovery on MTC-ANN

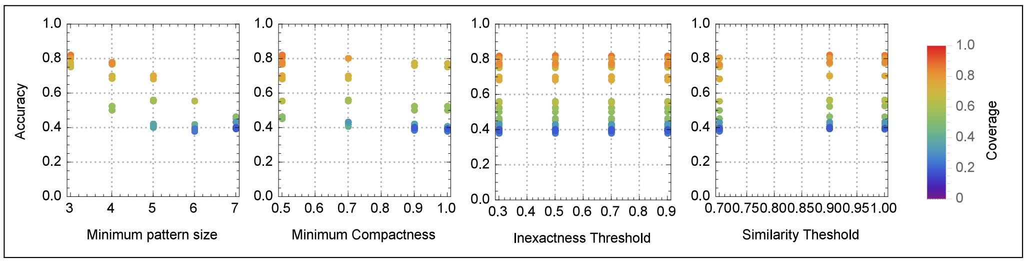


(c) Song Pattern Discovery on the Irish Folk Tunes Dataset

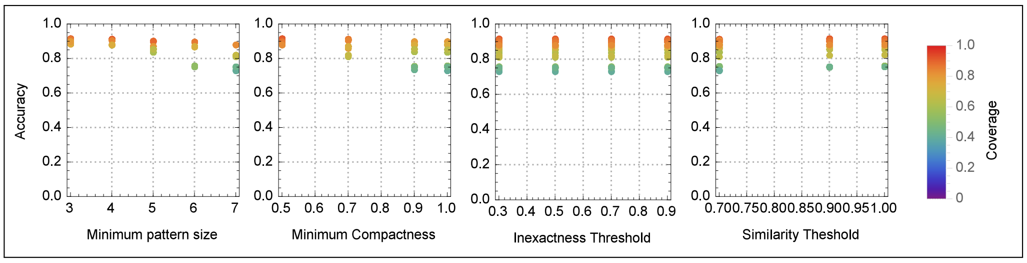


(d) Class Pattern Discovery on the Irish Folk Tunes Dataset

Figure B.21.: MotivesExtractor

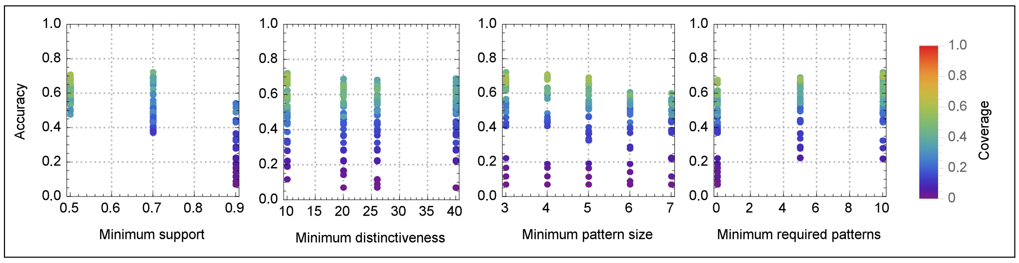


(a) Song Pattern Discovery on MTC-ANN

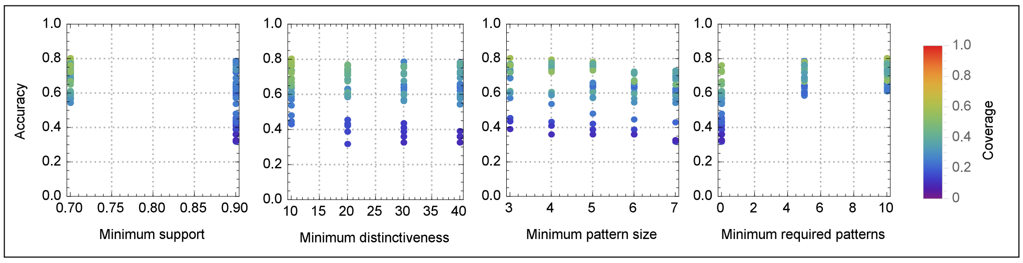


(b) Song Pattern Discovery on the Irish Folk Tunes Dataset

Figure B.22.: SIARCT-CFP

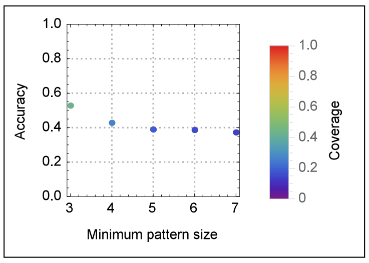


(a) Class Pattern Discovery on MTC-ANN

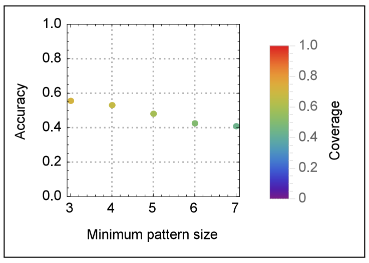


(b) Class Pattern Discovery on the Irish Folk Tunes Dataset

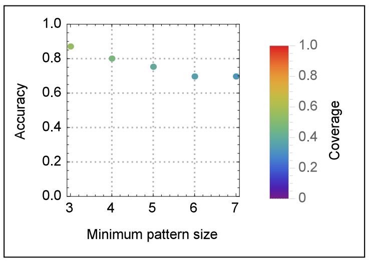
Figure B.23.: MGDP Algorithm



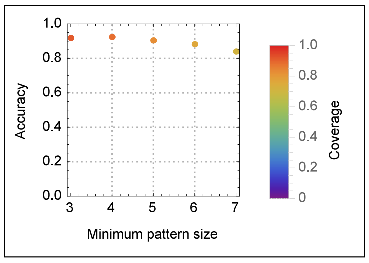
(a) Song Pattern Discovery on MTC-ANN



(b) Class Pattern Discovery on MTC-ANN



(c) Song Pattern Discovery on the Irish Folk Tunes Dataset



(d) Class Pattern Discovery on the Irish Folk Tunes Dataset

Figure B.24.: PatMinr

Enlarged Graphs C

Despite the graphs in chapter 7 show all the necessary results, due to the high number of parameter configurations the point clouds are dense in some graph areas. Therefore, all the coverage-accuracy plots are enlarged in this Appendix to cover one page each. In the caption of the figures, the reference to the original figure in the results chapter is given.

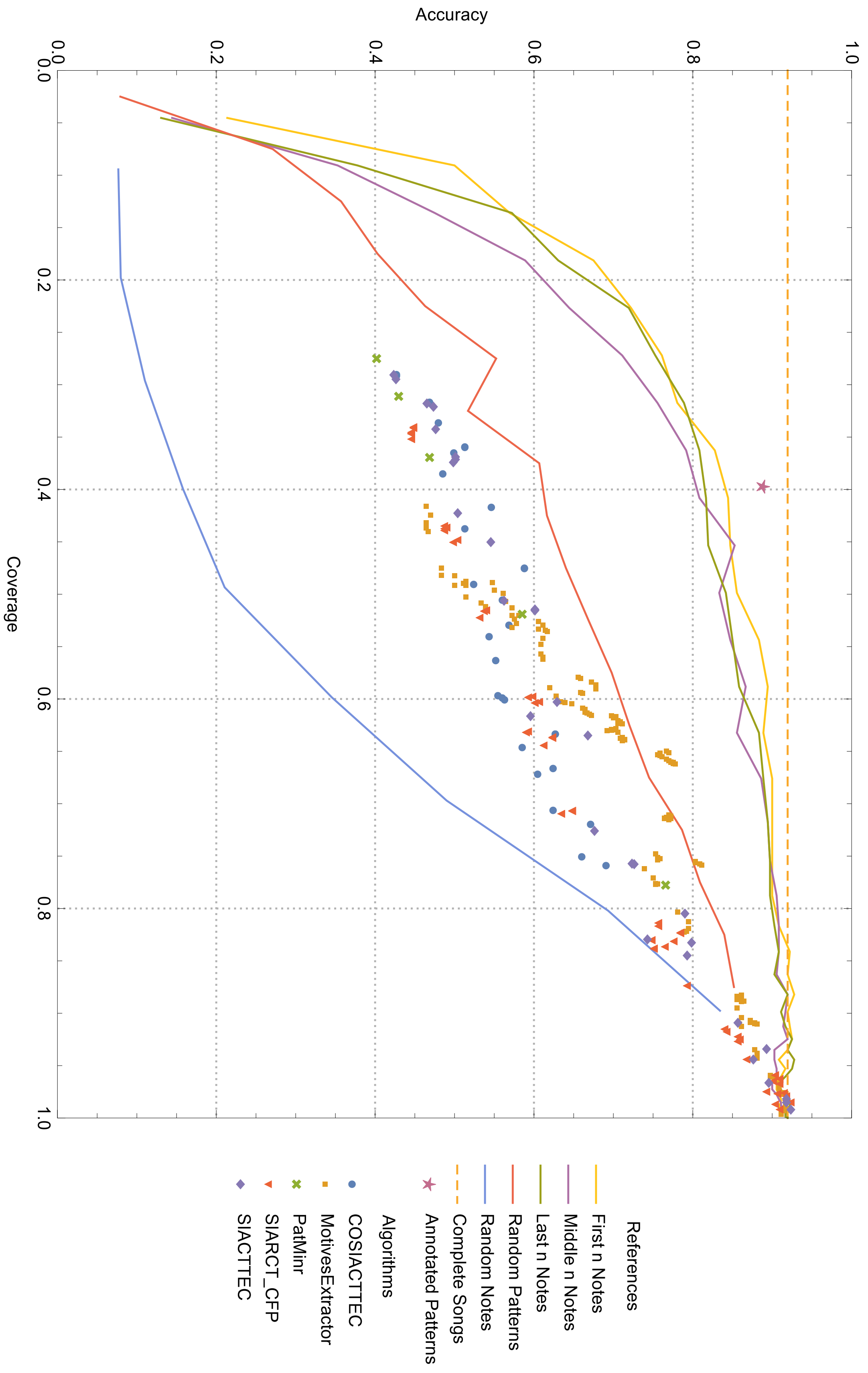


Figure C.1.: Song Pattern Discovery of MTC-ANN using the Gapless Reconstruction method and Note to Note Alignment. Each point in the plot represents a parameter configuration. This is a larger version of figure 7.2

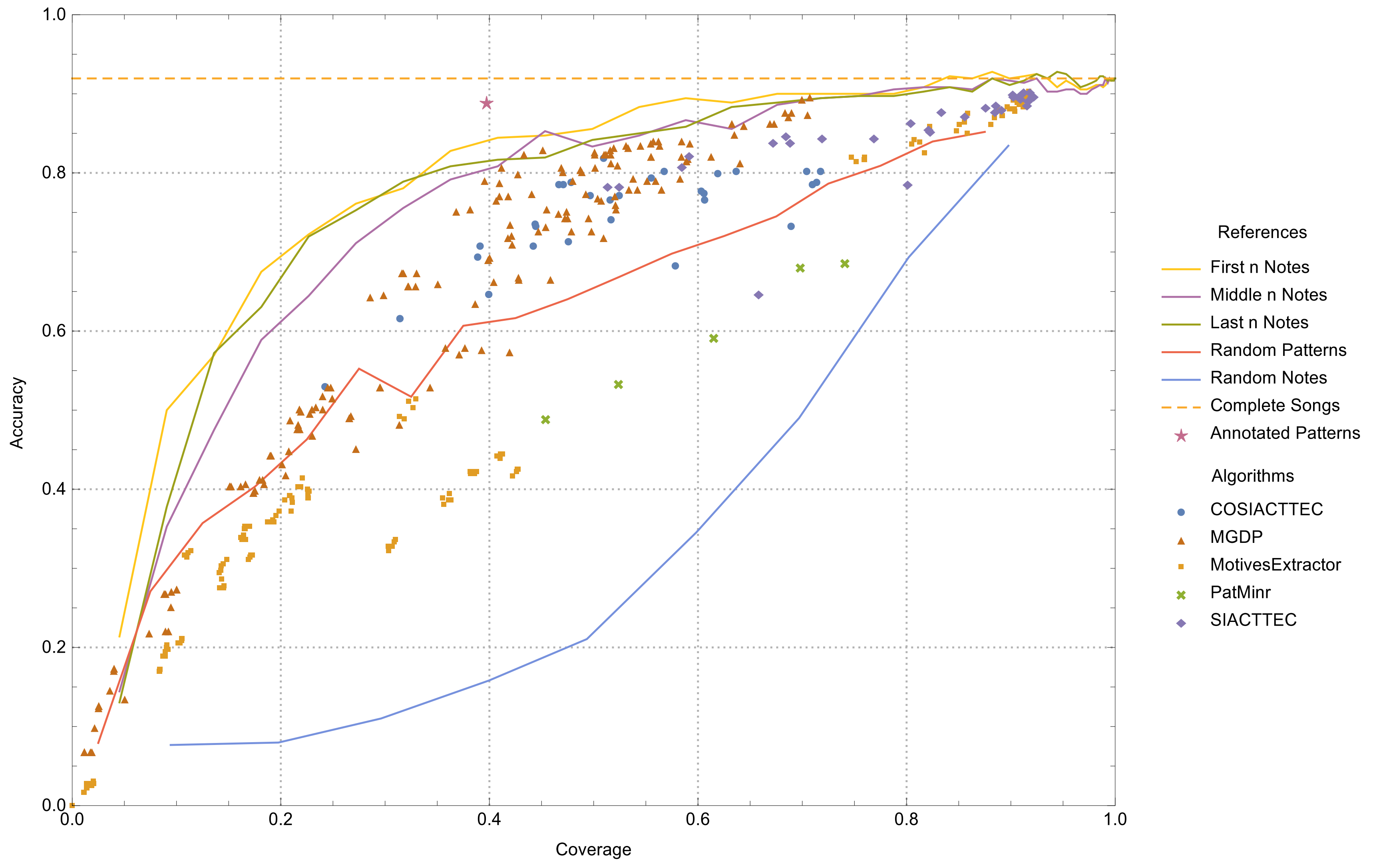


Figure C.2.: Class Pattern Discovery of MTC-ANN using the Gapless Reconstruction reconstruction method and Note to Note Alignment. Each point in the plot represents a parameter configuration.This is a larger version of figure 7.3

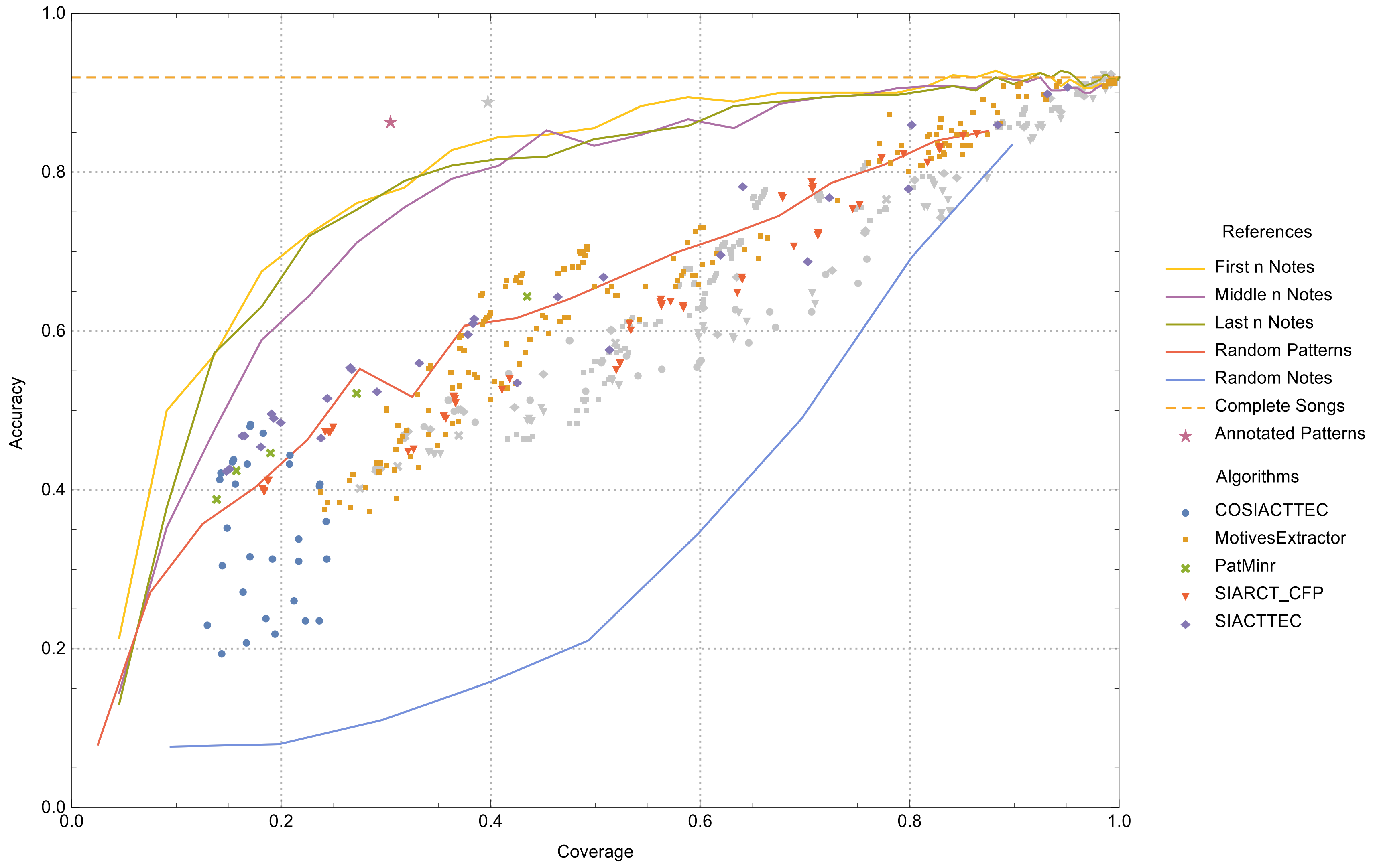


Figure C.3.: Song Pattern Discovery of MTC-ANN using the First Pattern Occurrence Reconstruction method and Note to Note Alignment. In gray the results from figure 7.2 using the Gapless Reconstruction method. Each point in the plot represents a parameter configuration.This is a larger version of figure 7.4

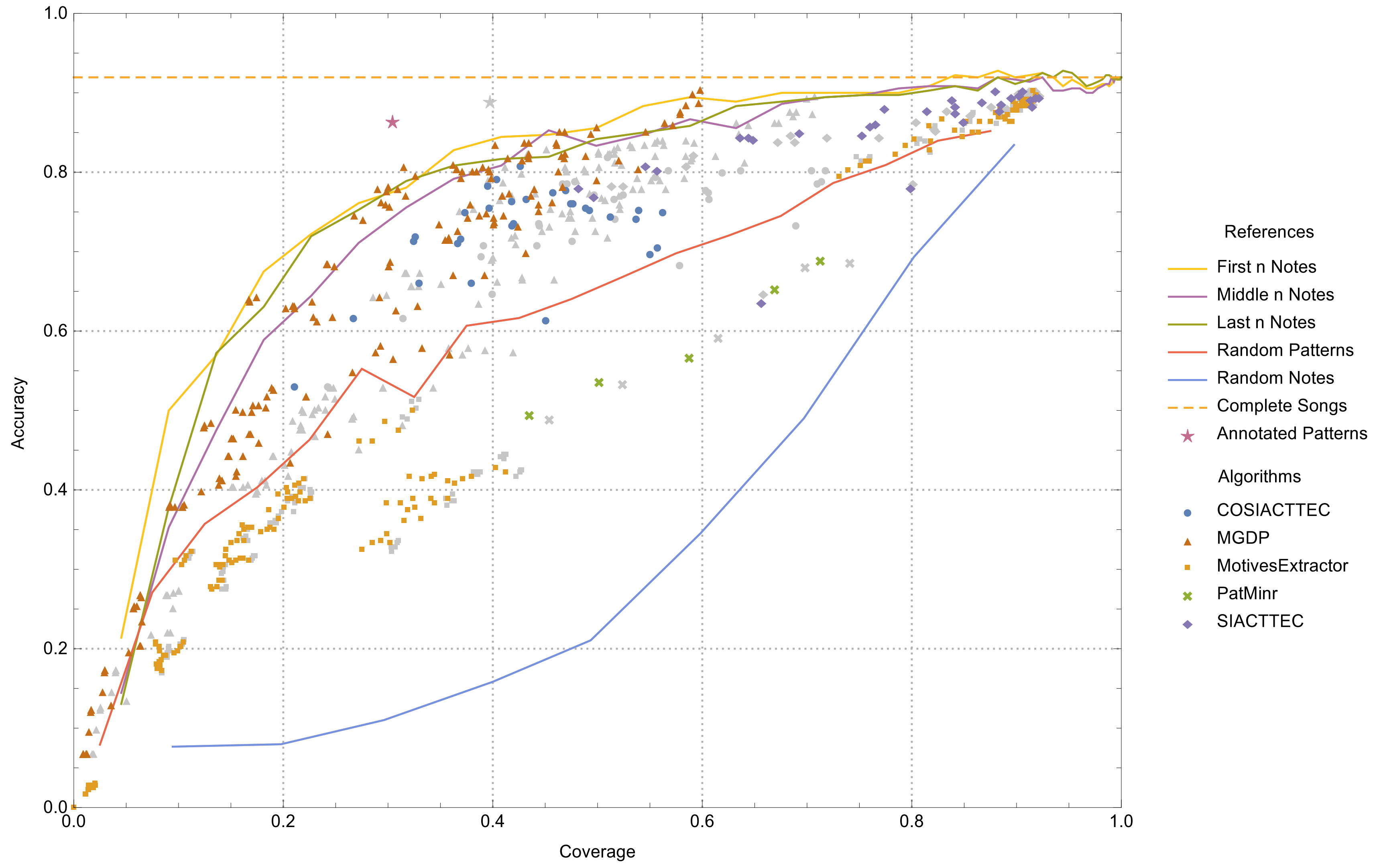


Figure C.4.: Class Pattern Discovery of MTC-ANN using the First Pattern Occurrence Reconstruction method and Note to Note Alignment. In gray the results from figure 7.3 using the Gapless Reconstruction method. Each point in the plot represents a parameter configuration.This is a larger version of figure 7.5

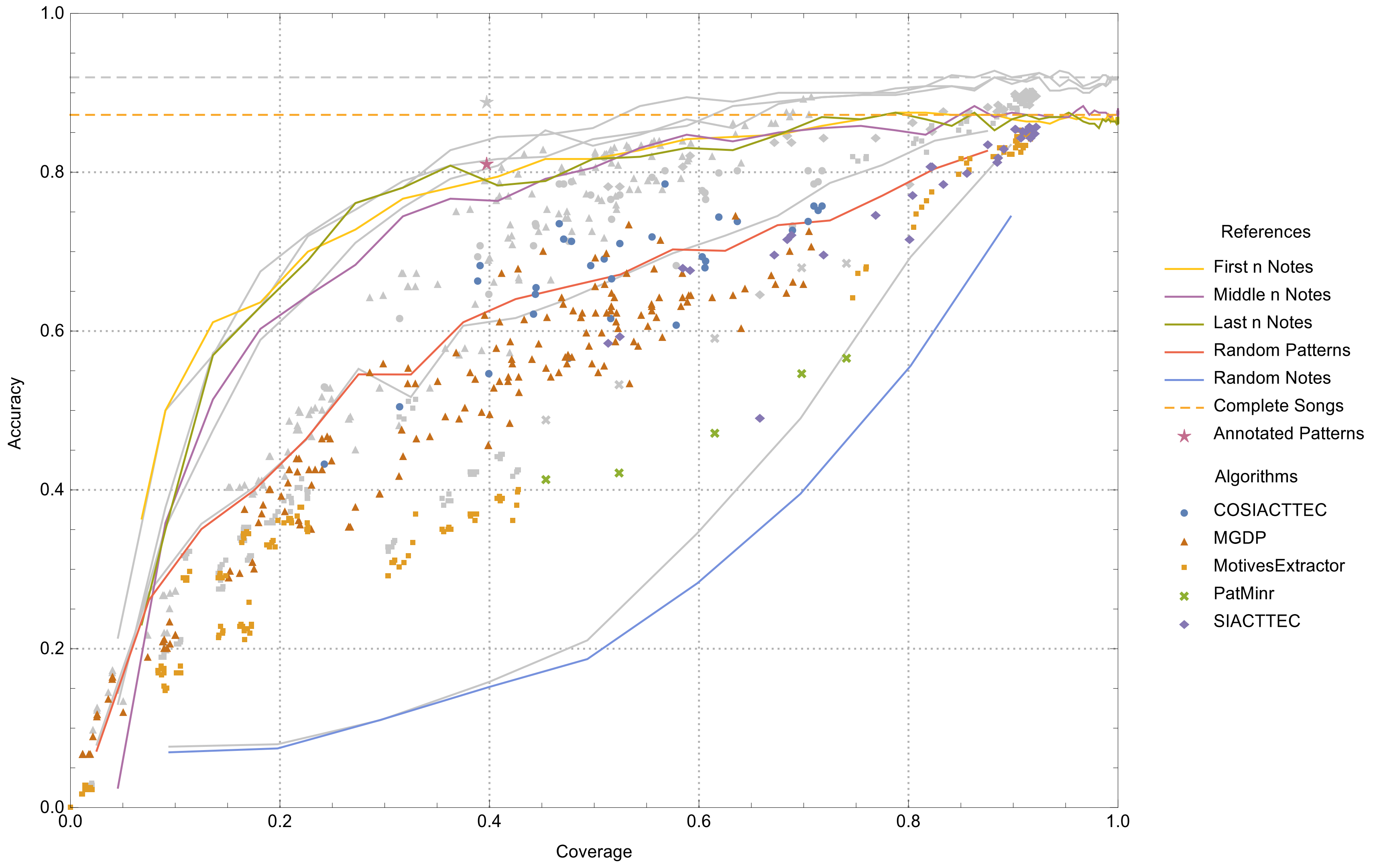


Figure C.5.: Class Pattern Discovery of MTC-ANN using the Gapless Reconstruction method and Melodic Shape Alignment. In gray the results from figure 7.3 using the Note to Note Alignment algorithm. Each point in the plot represents a parameter configuration.This is a larger version of figure 7.6

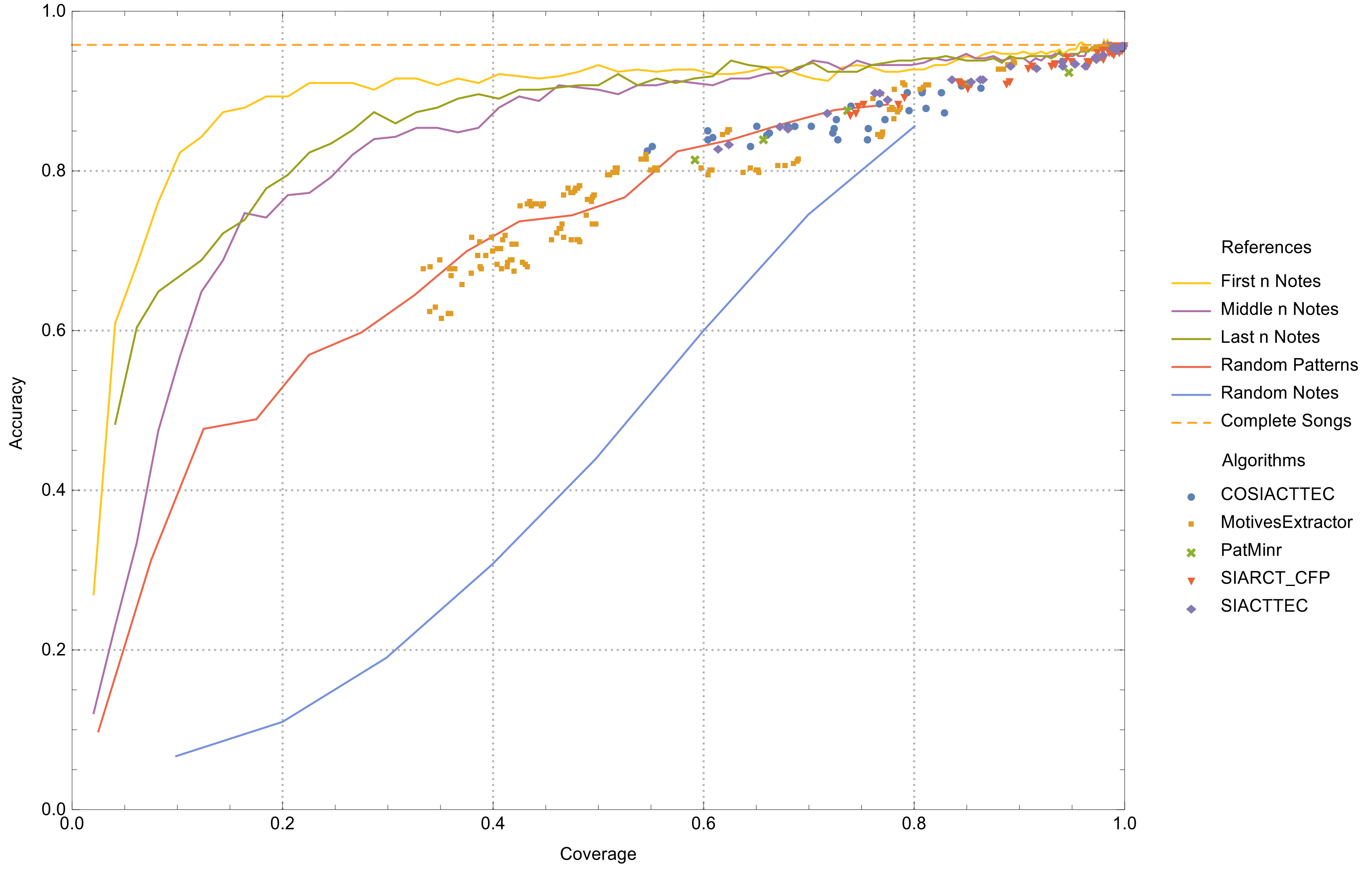


Figure C.6.: Song Pattern Discovery of the Irish Folk Tunes Dataset using the Gapless Reconstruction method and Note to Note Alignment. Each point in the plot represents a parameter configuration.This is a larger version of figure 7.7

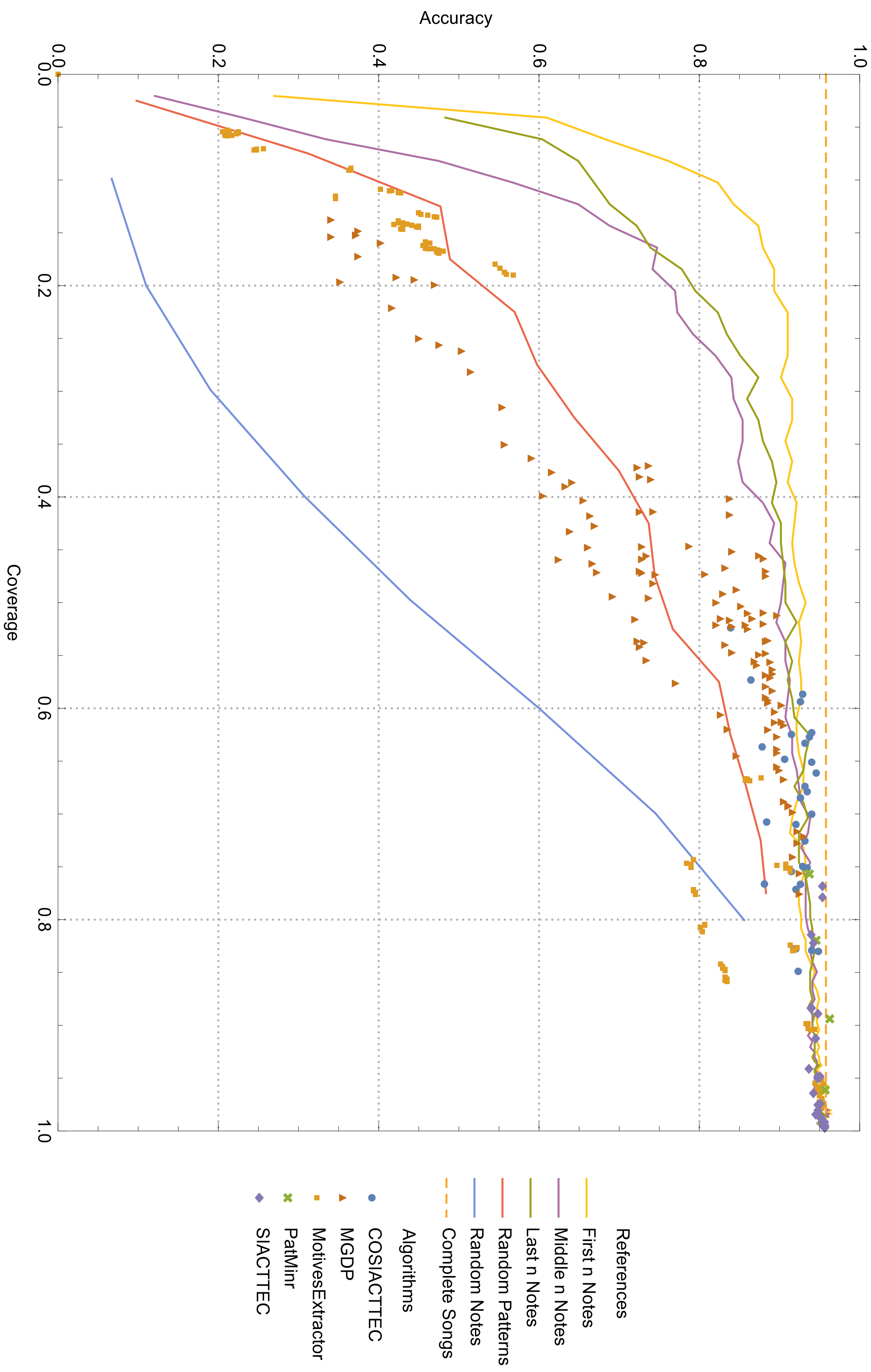


Figure C.7.: Class Pattern Discovery on the Irish Folk Tunes Dataset using the Gapless Reconstruction method and Note to Note Alignment. Each point in the plot represents a parameter configuration. This is a larger version of figure 7.8.

Raw Results D

In this appendix, all raw data retrieved from the experiments described in chapter 6 are shown, except for the experiments where all results are already presented in tabular form in chapter 7. In order to save space, the names of the reconstruction methods and alignment algorithms are abbreviated:

GR Gapless Reconstruction

FPOR First Pattern Occurrence Reconstruction

NNA Note to Note Alignment

MSA Melodic Shape Alignment

When combinations of these methods are used, there is a dash between the abbreviations. For example, the combination of Gapless Reconstruction and Note to Note Alignment is denoted as "GR-NNA". The exact meaning of each parameter can be found in chapter 3. For some algorithms, the parameter symbols are explained in the caption of the table.

In the first three sections, the raw results of the naive lossy compression methods are shown: the results of the consecutive subsequence selection is given in section D.1, the random note selection in section D.2 and the random pattern selection in section D.3. Then, the raw results of the Pattern Discovery algorithms are given. The results are categorized per dataset, pattern discovery approach and algorithm. For MTC-ANN, the raw results of the Song Pattern Discovery task are presented in section D.4 and Class Pattern Discovery task in section D.5. For the Irish Folk Tunes Dataset, the results of the Song Pattern Discovery task are shown in section D.6. The Class Pattern Discovery results are shown in section D.7.

D.1. Selecting a Consecutive Subsequence

D.1.1. MTC-ANN

		Accuracy first n		Accuracy middle n		Accuracy last n	
n (notes)	Coverage	NNA	MSA	NNA	MSA	NNA	MSA
2	0.045	0.214		0.144		0.131	
4	0.091	0.500	0.500	0.353	0.358	0.378	0.350
6	0.136	0.569	0.611	0.475	0.514	0.572	0.569
8	0.181	0.675	0.636	0.589	0.603	0.631	0.631
10	0.227	0.722	0.700	0.644	0.644	0.719	0.689
12	0.272	0.761	0.728	0.711	0.683	0.753	0.761
14	0.317	0.781	0.767	0.756	0.744	0.789	0.781
16	0.363	0.828	0.781	0.792	0.767	0.808	0.808

(continued from previous page)

		Accuracy First n		Accuracy Middle n		Accuracy Last n	
n (notes)	Coverage	NNA	MSA	NNA	MSA	NNA	MSA
18	0.408	0.844	0.794	0.808	0.764	0.817	0.783
20	0.453	0.847	0.817	0.853	0.792	0.819	0.789
22	0.499	0.856	0.817	0.833	0.806	0.842	0.817
24	0.544	0.883	0.828	0.847	0.831	0.850	0.819
26	0.588	0.894	0.842	0.867	0.847	0.858	0.831
28	0.632	0.889	0.844	0.856	0.839	0.883	0.828
30	0.676	0.900	0.847	0.886	0.850	0.889	0.847
32	0.717	0.900	0.858	0.894	0.856	0.894	0.869
34	0.755	0.900	0.867	0.897	0.858	0.897	0.867
36	0.788	0.900	0.875	0.906	0.853	0.897	0.875
38	0.816	0.908	0.875	0.908	0.847	0.903	0.867
40	0.841	0.922	0.872	0.908	0.867	0.908	0.858
42	0.863	0.919	0.872	0.906	0.883	0.903	0.875
44	0.882	0.928	0.872	0.919	0.869	0.919	0.853
46	0.899	0.919	0.867	0.917	0.875	0.911	0.867
48	0.913	0.922	0.864	0.914	0.872	0.917	0.872
50	0.925	0.925	0.864	0.919	0.872	0.925	0.867
52	0.935	0.919	0.861	0.903	0.869	0.919	0.869
54	0.944	0.908	0.867	0.903	0.869	0.928	0.869
56	0.953	0.917	0.869	0.906	0.869	0.925	0.875
58	0.960	0.911	0.867	0.906	0.878	0.917	0.869
60	0.967	0.906	0.869	0.900	0.883	0.908	0.864
62	0.973	0.906	0.867	0.900	0.872	0.911	0.861
64	0.978	0.908	0.867	0.906	0.878	0.914	0.861
66	0.982	0.911	0.867	0.908	0.875	0.917	0.856
68	0.985	0.911	0.867	0.911	0.875	0.922	0.867
70	0.988	0.908	0.867	0.911	0.875	0.922	0.864
72	0.991	0.911	0.872	0.919	0.872	0.919	0.867
74	0.993	0.917	0.864	0.914	0.869	0.917	0.864
76	0.995	0.917	0.869	0.919	0.872	0.919	0.864
78	0.996	0.917	0.864	0.917	0.872	0.917	0.864
80	0.997	0.919	0.864	0.917	0.872	0.917	0.864
82	0.999	0.919	0.869	0.917	0.872	0.917	0.861
84	0.999	0.919	0.864	0.919	0.872	0.919	0.864
86	0.999	0.919	0.864	0.919	0.878	0.919	0.869
88	1.000	0.919	0.864	0.919	0.875	0.919	0.861

Table D.1.: Raw results of the selection of the first, middle or last n notes on MTC-ANN.

D.1.2. Irish Folk Tunes Dataset

		Accuracy first n		Accuracy middle n		Accuracy last n	
n (notes)	Coverage	NNA	MSA	NNA	MSA	NNA	MSA
2	0.020	0.270		0.121		0.143	
4	0.041	0.610	0.632	0.230	0.236	0.483	0.480
6	0.061	0.683	0.719	0.334	0.334	0.604	0.573
8	0.082	0.761	0.756	0.475	0.447	0.649	0.632
10	0.102	0.823	0.803	0.567	0.497	0.669	0.615
12	0.123	0.843	0.829	0.649	0.551	0.688	0.654
14	0.143	0.874	0.854	0.688	0.581	0.722	0.649
16	0.164	0.879	0.865	0.747	0.626	0.739	0.666
18	0.184	0.893	0.885	0.742	0.649	0.778	0.711
20	0.205	0.893	0.888	0.770	0.697	0.795	0.716
22	0.225	0.910	0.893	0.772	0.713	0.823	0.753
24	0.246	0.910	0.893	0.792	0.736	0.834	0.756
26	0.266	0.910	0.893	0.820	0.750	0.851	0.806

(continued from previous page)

		Accuracy First *n*		**Accuracy Middle** *n*		**Accuracy Last** *n*	
n **(notes)**	**Coverage**	**NNA**	**MSA**	**NNA**	**MSA**	**NNA**	**MSA**
28	0.287	0.902	0.899	0.840	0.736	0.874	0.801
30	0.307	0.916	0.902	0.843	0.758	0.860	0.817
32	0.327	0.916	0.904	0.854	0.775	0.874	0.812
34	0.347	0.907	0.904	0.854	0.789	0.879	0.798
36	0.366	0.916	0.907	0.848	0.781	0.890	0.817
38	0.386	0.910	0.907	0.854	0.778	0.896	0.812
40	0.406	0.921	0.907	0.879	0.792	0.890	0.823
42	0.425	0.919	0.907	0.893	0.789	0.902	0.823
44	0.444	0.916	0.910	0.888	0.806	0.902	0.826
46	0.462	0.919	0.910	0.907	0.803	0.904	0.826
48	0.481	0.924	0.910	0.904	0.834	0.907	0.803
50	0.500	0.933	0.913	0.902	0.826	0.907	0.826
52	0.519	0.924	0.913	0.896	0.812	0.921	0.806
54	0.537	0.927	0.916	0.907	0.823	0.907	0.798
56	0.555	0.924	0.919	0.907	0.809	0.916	0.826
58	0.573	0.927	0.921	0.913	0.812	0.910	0.826
60	0.591	0.927	0.924	0.910	0.812	0.916	0.837
62	0.609	0.921	0.924	0.907	0.820	0.919	0.843
64	0.626	0.921	0.924	0.916	0.843	0.938	0.860
66	0.643	0.924	0.924	0.916	0.834	0.933	0.851
68	0.659	0.930	0.924	0.921	0.826	0.930	0.843
70	0.674	0.930	0.924	0.924	0.829	0.919	0.831
72	0.689	0.921	0.924	0.927	0.823	0.930	0.846
74	0.704	0.916	0.924	0.938	0.826	0.935	0.843
76	0.718	0.913	0.924	0.935	0.829	0.924	0.837
78	0.732	0.930	0.927	0.927	0.812	0.924	0.837
80	0.746	0.933	0.927	0.938	0.829	0.924	0.843
82	0.759	0.930	0.927	0.933	0.823	0.933	0.848
84	0.773	0.924	0.927	0.933	0.831	0.935	0.837
86	0.785	0.924	0.927	0.933	0.837	0.938	0.840
88	0.797	0.927	0.927	0.933	0.840	0.938	0.831
90	0.809	0.927	0.927	0.935	0.829	0.941	0.834
92	0.820	0.933	0.924	0.941	0.829	0.941	0.837
94	0.830	0.933	0.924	0.938	0.837	0.944	0.854
96	0.840	0.938	0.924	0.941	0.846	0.941	0.860
98	0.849	0.944	0.924	0.947	0.843	0.938	0.854
100	0.858	0.941	0.924	0.941	0.843	0.938	0.843
102	0.867	0.947	0.924	0.941	0.848	0.938	0.857
104	0.875	0.949	0.924	0.944	0.851	0.941	0.857
106	0.883	0.947	0.924	0.938	0.848	0.935	0.843
108	0.891	0.947	0.924	0.941	0.860	0.944	0.860
110	0.898	0.947	0.924	0.941	0.843	0.941	0.865
112	0.904	0.949	0.924	0.941	0.851	0.941	0.868
114	0.910	0.947	0.924	0.935	0.857	0.944	0.860
116	0.915	0.947	0.924	0.941	0.848	0.944	0.854
118	0.921	0.949	0.921	0.938	0.848	0.944	0.868
120	0.926	0.947	0.921	0.944	0.862	0.944	0.871
122	0.930	0.949	0.921	0.944	0.860	0.941	0.876
124	0.934	0.949	0.921	0.947	0.874	0.944	0.899
126	0.938	0.952	0.921	0.947	0.879	0.949	0.899
128	0.941	0.947	0.921	0.944	0.876	0.944	0.902
130	0.945	0.949	0.921	0.947	0.888	0.944	0.902
132	0.948	0.952	0.921	0.944	0.879	0.947	0.882
134	0.951	0.952	0.921	0.944	0.890	0.947	0.885
136	0.954	0.952	0.921	0.947	0.888	0.947	0.876
138	0.956	0.958	0.921	0.944	0.890	0.947	0.882
140	0.958	0.961	0.921	0.944	0.885	0.949	0.876
142	0.960	0.961	0.921	0.944	0.888	0.949	0.885
144	0.962	0.958	0.921	0.944	0.890	0.949	0.890
146	0.964	0.952	0.921	0.949	0.890	0.949	0.885

(continued from previous page)

		Accuracy First n		Accuracy Middle n		Accuracy Last n	
n (notes)	Coverage	NNA	MSA	NNA	MSA	NNA	MSA
148	0.966	0.952	0.921	0.949	0.890	0.949	0.888
150	0.967	0.952	0.921	0.952	0.882	0.952	0.890
152	0.969	0.952	0.921	0.955	0.890	0.952	0.893
154	0.971	0.955	0.921	0.955	0.890	0.952	0.890
156	0.972	0.955	0.921	0.952	0.890	0.952	0.882
158	0.974	0.955	0.921	0.955	0.896	0.952	0.890
160	0.975	0.955	0.921	0.955	0.907	0.952	0.882
162	0.976	0.958	0.921	0.955	0.899	0.952	0.890
164	0.978	0.958	0.921	0.958	0.902	0.952	0.893
166	0.979	0.958	0.921	0.958	0.910	0.952	0.899
168	0.980	0.963	0.921	0.961	0.907	0.952	0.902
170	0.982	0.961	0.921	0.958	0.910	0.952	0.902
172	0.983	0.961	0.921	0.958	0.904	0.952	0.907
174	0.984	0.963	0.921	0.961	0.910	0.955	0.907
176	0.985	0.958	0.921	0.958	0.910	0.955	0.913
178	0.986	0.958	0.921	0.961	0.919	0.958	0.913
180	0.987	0.961	0.921	0.961	0.913	0.958	0.902

Table D.2.: Raw results of the selection of the first, middle or last n notes on the Irish Folk Tunes Dataset.

D.2. Random Note Selection

D.2.1. MTC-ANN

			Accuracy	
Sample	Goal Coverage	Coverage	Note to Note Alignment	Melodic Shape Alignment
0	0.1	0.094	0.053	0.033
1	0.1	0.094	0.069	0.047
2	0.1	0.094	0.050	0.061
3	0.1	0.094	0.075	0.050
4	0.1	0.094	0.089	0.072
5	0.1	0.094	0.050	0.072
6	0.1	0.094	0.078	0.083
7	0.1	0.094	0.078	0.056
8	0.1	0.094	0.078	0.072
9	0.1	0.094	0.097	0.103
10	0.1	0.094	0.106	0.075
11	0.1	0.094	0.106	0.094
12	0.1	0.094	0.083	0.075
13	0.1	0.094	0.086	0.058
14	0.1	0.094	0.078	0.058
15	0.1	0.094	0.089	0.067
16	0.1	0.094	0.058	0.067
17	0.1	0.094	0.072	0.083
18	0.1	0.094	0.061	0.072
19	0.1	0.094	0.078	0.092
0	0.2	0.198	0.069	0.056
1	0.2	0.198	0.106	0.067
2	0.2	0.198	0.058	0.069
3	0.2	0.198	0.092	0.083
4	0.2	0.198	0.044	0.072
5	0.2	0.198	0.100	0.081
6	0.2	0.198	0.111	0.089
7	0.2	0.198	0.072	0.058

(continued from previous page)

			Accuracy	
Sample	**Goal Coverage**	**Coverage**	**Note to Note Alignment**	**Melodic Shape Alignment**
8	0.2	0.198	0.064	0.078
9	0.2	0.198	0.053	0.056
10	0.2	0.198	0.092	0.067
11	0.2	0.198	0.067	0.078
12	0.2	0.198	0.058	0.064
13	0.2	0.198	0.078	0.086
14	0.2	0.198	0.097	0.100
15	0.2	0.198	0.078	0.089
16	0.2	0.198	0.106	0.108
17	0.2	0.198	0.069	0.072
18	0.2	0.198	0.067	0.047
19	0.2	0.198	0.114	0.069
0	0.3	0.296	0.078	0.108
1	0.3	0.296	0.092	0.097
2	0.3	0.296	0.111	0.072
3	0.3	0.296	0.089	0.103
4	0.3	0.296	0.133	0.128
5	0.3	0.296	0.111	0.097
6	0.3	0.296	0.103	0.158
7	0.3	0.296	0.122	0.139
8	0.3	0.296	0.114	0.089
9	0.3	0.296	0.103	0.106
10	0.3	0.296	0.125	0.117
11	0.3	0.296	0.067	0.108
12	0.3	0.296	0.092	0.125
13	0.3	0.296	0.133	0.119
14	0.3	0.296	0.144	0.122
15	0.3	0.296	0.119	0.106
16	0.3	0.296	0.131	0.108
17	0.3	0.296	0.108	0.092
18	0.3	0.296	0.128	0.103
19	0.3	0.296	0.100	0.111
0	0.4	0.400	0.144	0.153
1	0.4	0.400	0.119	0.153
2	0.4	0.400	0.133	0.133
3	0.4	0.400	0.133	0.131
4	0.4	0.400	0.206	0.139
5	0.4	0.400	0.153	0.133
6	0.4	0.400	0.133	0.139
7	0.4	0.400	0.153	0.183
8	0.4	0.400	0.206	0.189
9	0.4	0.400	0.147	0.119
10	0.4	0.400	0.147	0.164
11	0.4	0.400	0.156	0.122
12	0.4	0.400	0.139	0.167
13	0.4	0.400	0.208	0.156
14	0.4	0.400	0.172	0.158
15	0.4	0.400	0.144	0.156
16	0.4	0.400	0.164	0.194
17	0.4	0.400	0.175	0.144
18	0.4	0.400	0.175	0.175
19	0.4	0.400	0.156	0.128
0	0.5	0.493	0.189	0.147
1	0.5	0.493	0.219	0.183
2	0.5	0.493	0.197	0.186
3	0.5	0.493	0.172	0.153
4	0.5	0.493	0.231	0.197
5	0.5	0.493	0.192	0.150
6	0.5	0.493	0.242	0.211
7	0.5	0.493	0.217	0.200

(continued from previous page)

			Accuracy	
Sample	**Goal Coverage**	**Coverage**	**Note to Note Alignment**	**Melodic Shape Alignment**
8	0.5	0.493	0.225	0.194
9	0.5	0.493	0.228	0.217
10	0.5	0.493	0.192	0.192
11	0.5	0.493	0.247	0.200
12	0.5	0.493	0.219	0.211
13	0.5	0.493	0.225	0.183
14	0.5	0.493	0.225	0.186
15	0.5	0.493	0.203	0.172
16	0.5	0.493	0.197	0.197
17	0.5	0.493	0.192	0.178
18	0.5	0.493	0.208	0.225
19	0.5	0.493	0.192	0.156
0	0.6	0.598	0.350	0.283
1	0.6	0.598	0.325	0.297
2	0.6	0.598	0.358	0.306
3	0.6	0.598	0.333	0.286
4	0.6	0.598	0.328	0.219
5	0.6	0.598	0.356	0.258
6	0.6	0.598	0.333	0.314
7	0.6	0.598	0.336	0.308
8	0.6	0.598	0.344	0.236
9	0.6	0.598	0.347	0.250
10	0.6	0.598	0.328	0.283
11	0.6	0.598	0.364	0.283
12	0.6	0.598	0.336	0.339
13	0.6	0.598	0.378	0.269
14	0.6	0.598	0.289	0.253
15	0.6	0.598	0.372	0.289
16	0.6	0.598	0.361	0.297
17	0.6	0.598	0.364	0.306
18	0.6	0.598	0.364	0.292
19	0.6	0.598	0.344	0.281
0	0.7	0.697	0.475	0.397
1	0.7	0.697	0.514	0.367
2	0.7	0.697	0.461	0.375
3	0.7	0.697	0.492	0.436
4	0.7	0.697	0.503	0.397
5	0.7	0.697	0.475	0.369
6	0.7	0.697	0.483	0.400
7	0.7	0.697	0.517	0.394
8	0.7	0.697	0.481	0.397
9	0.7	0.697	0.511	0.367
10	0.7	0.697	0.481	0.400
11	0.7	0.697	0.450	0.383
12	0.7	0.697	0.483	0.389
13	0.7	0.697	0.500	0.431
14	0.7	0.697	0.492	0.408
15	0.7	0.697	0.486	0.386
16	0.7	0.697	0.514	0.367
17	0.7	0.697	0.489	0.447
18	0.7	0.697	0.478	0.403
19	0.7	0.697	0.517	0.389
0	0.8	0.802	0.703	0.539
1	0.8	0.802	0.694	0.539
2	0.8	0.802	0.719	0.586
3	0.8	0.802	0.706	0.511
4	0.8	0.802	0.694	0.561
5	0.8	0.802	0.683	0.578
6	0.8	0.802	0.703	0.558
7	0.8	0.802	0.683	0.600

(continued from previous page)

			Accuracy	
Sample	Goal Coverage	Coverage	Note to Note Alignment	Melodic Shape Alignment
8	0.8	0.802	0.667	0.572
9	0.8	0.802	0.689	0.569
10	0.8	0.802	0.711	0.572
11	0.8	0.802	0.689	0.556
12	0.8	0.802	0.678	0.514
13	0.8	0.802	0.675	0.578
14	0.8	0.802	0.717	0.575
15	0.8	0.802	0.694	0.536
16	0.8	0.802	0.686	0.542
17	0.8	0.802	0.703	0.558
18	0.8	0.802	0.686	0.519
19	0.8	0.802	0.686	0.561
0	0.9	0.898	0.850	0.719
1	0.9	0.898	0.856	0.742
2	0.9	0.898	0.864	0.750
3	0.9	0.898	0.808	0.758
4	0.9	0.898	0.822	0.742
5	0.9	0.898	0.836	0.708
6	0.9	0.898	0.844	0.739
7	0.9	0.898	0.811	0.742
8	0.9	0.898	0.814	0.756
9	0.9	0.898	0.836	0.772
10	0.9	0.898	0.833	0.753
11	0.9	0.898	0.833	0.708
12	0.9	0.898	0.811	0.739
13	0.9	0.898	0.819	0.733
14	0.9	0.898	0.844	0.775
15	0.9	0.898	0.850	0.747
16	0.9	0.898	0.836	0.744
17	0.9	0.898	0.839	0.750
18	0.9	0.898	0.839	0.764
19	0.9	0.898	0.839	0.742

Table D.3.: Raw results of the random note selection on MTC-ANN. For each (goal) coverage value, the resulting coverage value is given (due to rounding effects). For each coverage value, the result is averaged over 20 samples.

D.2.2. Irish Folk Tunes Dataset

			Accuracy	
Sample	Goal Coverage	Coverage	Note to Note Alignment	Melodic Shape Alignment
0	0.1	0.099	0.053	0.039
1	0.1	0.099	0.070	0.076
2	0.1	0.099	0.076	0.087
3	0.1	0.099	0.062	0.059
4	0.1	0.099	0.034	0.081
5	0.1	0.099	0.065	0.062
6	0.1	0.099	0.076	0.067
7	0.1	0.099	0.096	0.048
8	0.1	0.099	0.065	0.073
9	0.1	0.099	0.073	0.098
10	0.1	0.099	0.076	0.070
11	0.1	0.099	0.067	0.079
12	0.1	0.099	0.042	0.065
13	0.1	0.099	0.073	0.101
14	0.1	0.099	0.090	0.059

(continued from previous page)

			Accuracy	
Sample	**Goal Coverage**	**Coverage**	**Note to Note Alignment**	**Melodic Shape Alignment**
15	0.1	0.099	0.053	0.090
16	0.1	0.099	0.062	0.081
17	0.1	0.099	0.076	0.056
18	0.1	0.099	0.070	0.067
19	0.1	0.099	0.062	0.073
0	0.2	0.200	0.079	0.062
1	0.2	0.200	0.000	0.090
2	0.2	0.200	0.098	0.104
3	0.2	0.200	0.135	0.146
4	0.2	0.200	0.126	0.093
5	0.2	0.200	0.135	0.121
6	0.2	0.200	0.157	0.098
7	0.2	0.200	0.112	0.098
8	0.2	0.200	0.110	0.126
9	0.2	0.200	0.143	0.090
10	0.2	0.200	0.110	0.112
11	0.2	0.200	0.126	0.126
12	0.2	0.200	0.143	0.087
13	0.2	0.200	0.090	0.076
14	0.2	0.200	0.101	0.084
15	0.2	0.200	0.115	0.098
16	0.2	0.200	0.118	0.104
17	0.2	0.200	0.090	0.081
18	0.2	0.200	0.115	0.056
19	0.2	0.200	0.096	0.081
0	0.3	0.299	0.205	0.152
1	0.3	0.299	0.180	0.084
2	0.3	0.299	0.191	0.197
3	0.3	0.299	0.174	0.115
4	0.3	0.299	0.174	0.118
5	0.3	0.299	0.194	0.135
6	0.3	0.299	0.197	0.154
7	0.3	0.299	0.183	0.166
8	0.3	0.299	0.191	0.183
9	0.3	0.299	0.194	0.146
10	0.3	0.299	0.236	0.163
11	0.3	0.299	0.191	0.143
12	0.3	0.299	0.185	0.121
13	0.3	0.299	0.219	0.132
14	0.3	0.299	0.228	0.112
15	0.3	0.299	0.171	0.146
16	0.3	0.299	0.233	0.124
17	0.3	0.299	0.160	0.110
18	0.3	0.299	0.152	0.126
19	0.3	0.299	0.146	0.107
0	0.4	0.399	0.306	0.197
1	0.4	0.399	0.267	0.157
2	0.4	0.399	0.351	0.146
3	0.4	0.399	0.281	0.194
4	0.4	0.399	0.298	0.169
5	0.4	0.399	0.309	0.166
6	0.4	0.399	0.340	0.188
7	0.4	0.399	0.320	0.205
8	0.4	0.399	0.306	0.194
9	0.4	0.399	0.303	0.135
10	0.4	0.399	0.275	0.199
11	0.4	0.399	0.306	0.143
12	0.4	0.399	0.340	0.140
13	0.4	0.399	0.329	0.205
14	0.4	0.399	0.292	0.180

(continued from previous page)

			Accuracy	
Sample	**Goal Coverage**	**Coverage**	**Note to Note Alignment**	**Melodic Shape Alignment**
15	0.4	0.399	0.306	0.194
16	0.4	0.399	0.303	0.166
17	0.4	0.399	0.295	0.194
18	0.4	0.399	0.306	0.163
19	0.4	0.399	0.312	0.169
0	0.5	0.498	0.449	0.213
1	0.5	0.498	0.461	0.191
2	0.5	0.498	0.365	0.191
3	0.5	0.498	0.455	0.205
4	0.5	0.498	0.396	0.174
5	0.5	0.498	0.452	0.208
6	0.5	0.498	0.393	0.250
7	0.5	0.498	0.435	0.213
8	0.5	0.498	0.475	0.278
9	0.5	0.498	0.447	0.202
10	0.5	0.498	0.424	0.188
11	0.5	0.498	0.430	0.191
12	0.5	0.498	0.413	0.236
13	0.5	0.498	0.424	0.250
14	0.5	0.498	0.469	0.171
15	0.5	0.498	0.472	0.191
16	0.5	0.498	0.469	0.236
17	0.5	0.498	0.458	0.258
18	0.5	0.498	0.433	0.264
19	0.5	0.498	0.469	0.275
0	0.6	0.600	0.604	0.287
1	0.6	0.600	0.584	0.326
2	0.6	0.600	0.629	0.346
3	0.6	0.600	0.562	0.287
4	0.6	0.600	0.607	0.312
5	0.6	0.600	0.604	0.264
6	0.6	0.600	0.590	0.301
7	0.6	0.600	0.601	0.315
8	0.6	0.600	0.612	0.315
9	0.6	0.600	0.629	0.331
10	0.6	0.600	0.618	0.272
11	0.6	0.600	0.601	0.309
12	0.6	0.600	0.618	0.289
13	0.6	0.600	0.567	0.267
14	0.6	0.600	0.596	0.317
15	0.6	0.600	0.598	0.343
16	0.6	0.600	0.576	0.301
17	0.6	0.600	0.598	0.278
18	0.6	0.600	0.610	0.334
19	0.6	0.600	0.584	0.272
0	0.7	0.700	0.758	0.399
1	0.7	0.700	0.747	0.461
2	0.7	0.700	0.767	0.475
3	0.7	0.700	0.713	0.444
4	0.7	0.700	0.728	0.455
5	0.7	0.700	0.761	0.475
6	0.7	0.700	0.753	0.478
7	0.7	0.700	0.761	0.430
8	0.7	0.700	0.761	0.469
9	0.7	0.700	0.778	0.404
10	0.7	0.700	0.728	0.413
11	0.7	0.700	0.761	0.458
12	0.7	0.700	0.733	0.441
13	0.7	0.700	0.708	0.427
14	0.7	0.700	0.758	0.463

(continued from previous page)

			Accuracy	
Sample	Goal Coverage	Coverage	Note to Note Alignment	Melodic Shape Alignment
15	0.7	0.700	0.711	0.416
16	0.7	0.700	0.725	0.441
17	0.7	0.700	0.756	0.463
18	0.7	0.700	0.744	0.520
19	0.7	0.700	0.756	0.483
0	0.8	0.800	0.843	0.626
1	0.8	0.800	0.848	0.640
2	0.8	0.800	0.871	0.649
3	0.8	0.800	0.851	0.635
4	0.8	0.800	0.868	0.635
5	0.8	0.800	0.865	0.629
6	0.8	0.800	0.871	0.587
7	0.8	0.800	0.846	0.610
8	0.8	0.800	0.846	0.565
9	0.8	0.800	0.840	0.618
10	0.8	0.800	0.857	0.629
11	0.8	0.800	0.840	0.573
12	0.8	0.800	0.874	0.638
13	0.8	0.800	0.876	0.621
14	0.8	0.800	0.829	0.615
15	0.8	0.800	0.868	0.640
16	0.8	0.800	0.848	0.601
17	0.8	0.800	0.865	0.576
18	0.8	0.800	0.865	0.626
19	0.8	0.800	0.848	0.649

Table D.4.: Raw results of the random note selection on the Irish Folk Tunes Dataset. For each (goal) coverage value, the resulting coverage value is given (due to rounding effects). For each coverage value, the result is averaged over 20 samples.

D.3. Random Pattern Selection

D.3.1. MTC-ANN

	Accuracy	
Coverage	Note to Note Alignment	Melodic Shape Alignment
0.239	0.514	0.553
0.416	0.722	0.750
0.113	0.458	0.492
0.643	0.694	0.711
0.147	0.442	0.497
0.831	0.867	0.814
0.694	0.817	0.756
0.228	0.478	0.453
0.219	0.433	0.508
0.543	0.686	0.706
0.388	0.542	0.592
0.093	0.233	0.167
0.840	0.861	0.806
0.122	0.247	0.225
0.641	0.722	0.706
0.867	0.847	0.819
0.096	0.472	0.483
0.764	0.819	0.781

(continued from previous page)

	Accuracy	
Coverage	Note to Note Alignment	Melodic Shape Alignment
0.430	0.617	0.625
0.350	0.472	0.542
0.012	0.064	0.064
0.000	0.000	0.000
0.528	0.736	0.678
0.023	0.131	0.092
0.124	0.297	0.186
0.798	0.847	0.800
0.093	0.222	0.192
0.185	0.406	0.431
0.015	0.094	0.086
0.314	0.589	0.594
0.086	0.253	0.211
0.268	0.575	0.617
0.008	0.064	0.061
0.135	0.253	0.203
0.217	0.425	0.408
0.181	0.378	0.319
0.463	0.667	0.650
0.000	0.006	0.000
0.007	0.058	0.044
0.541	0.686	0.678
0.510	0.686	0.697
0.330	0.461	0.517
0.373	0.536	0.503
0.183	0.431	0.447
0.136	0.544	0.561
0.728	0.783	0.733
0.305	0.531	0.531
0.205	0.475	0.422
0.030	0.117	0.131
0.322	0.589	0.606
0.170	0.342	0.344
0.395	0.608	0.619
0.428	0.589	0.600
0.440	0.678	0.706
0.147	0.325	0.367
0.523	0.644	0.639
0.544	0.625	0.633
0.067	0.197	0.214
0.263	0.681	0.689
0.730	0.747	0.722
0.374	0.561	0.608
0.002	0.019	0.011
0.169	0.489	0.481
0.402	0.581	0.519
0.126	0.483	0.458
0.747	0.778	0.742
0.542	0.572	0.600
0.511	0.736	0.711
0.774	0.828	0.758
0.695	0.772	0.747
0.182	0.497	0.575
0.004	0.047	0.031
0.365	0.600	0.683
0.654	0.769	0.758
0.091	0.483	0.503
0.075	0.308	0.275
0.138	0.233	0.158
0.009	0.042	0.047

(continued from previous page)

	Accuracy	
Coverage	**Note to Note Alignment**	**Melodic Shape Alignment**
0.751	0.758	0.728
0.467	0.631	0.642
0.506	0.592	0.603
0.435	0.611	0.647
0.113	0.381	0.411
0.066	0.175	0.189
0.082	0.211	0.219
0.137	0.514	0.572
0.625	0.772	0.778
0.744	0.783	0.733
0.719	0.847	0.750
0.000	0.000	0.000
0.157	0.275	0.258
0.209	0.408	0.400
0.124	0.467	0.475
0.116	0.214	0.194
0.056	0.156	0.125
0.191	0.461	0.464
0.550	0.731	0.728
0.470	0.628	0.628
0.590	0.722	0.739
0.071	0.328	0.419
0.808	0.867	0.833
0.017	0.114	0.083
0.142	0.325	0.331
0.523	0.706	0.719
0.096	0.389	0.367
0.584	0.692	0.694
0.390	0.597	0.608
0.814	0.756	0.739
0.000	0.000	0.000
0.572	0.697	0.711
0.050	0.167	0.169
0.143	0.386	0.342
0.554	0.708	0.706
0.079	0.281	0.258
0.547	0.706	0.697
0.098	0.383	0.333
0.292	0.547	0.578
0.263	0.494	0.428
0.489	0.617	0.633
0.377	0.656	0.656
0.836	0.839	0.836
0.310	0.472	0.508
0.830	0.842	0.797
0.126	0.386	0.381
0.108	0.239	0.206
0.231	0.569	0.619
0.653	0.758	0.756
0.743	0.769	0.711
0.055	0.164	0.150
0.022	0.097	0.106
0.318	0.486	0.542
0.441	0.661	0.681
0.746	0.772	0.700
0.732	0.811	0.822
0.311	0.569	0.600
0.241	0.342	0.331
0.677	0.747	0.731
0.674	0.697	0.686

(continued from previous page)

	Accuracy	
Coverage	**Note to Note Alignment**	**Melodic Shape Alignment**
0.411	0.608	0.664
0.478	0.656	0.667
0.632	0.725	0.697
0.687	0.767	0.728
0.753	0.822	0.786
0.774	0.856	0.803
0.205	0.539	0.489
0.079	0.200	0.197
0.687	0.744	0.731
0.534	0.656	0.644
0.411	0.561	0.597
0.371	0.658	0.636
0.646	0.689	0.653
0.581	0.772	0.781
0.656	0.653	0.664
0.658	0.706	0.736
0.857	0.858	0.839
0.149	0.306	0.281
0.878	0.850	0.822
0.259	0.508	0.494
0.184	0.422	0.444
0.467	0.608	0.622
0.533	0.661	0.694
0.210	0.461	0.439
0.376	0.681	0.647
0.522	0.633	0.636
0.356	0.539	0.542
0.603	0.717	0.669
0.408	0.536	0.614
0.231	0.475	0.497
0.205	0.392	0.344
0.555	0.669	0.672
0.600	0.714	0.719
0.065	0.231	0.208
0.782	0.792	0.781
0.075	0.192	0.194
0.009	0.064	0.047
0.166	0.331	0.228
0.220	0.508	0.569
0.273	0.564	0.464
0.507	0.644	0.675
0.555	0.647	0.644
0.454	0.597	0.647
0.647	0.728	0.675
0.037	0.161	0.128
0.016	0.094	0.103
0.047	0.172	0.142
0.599	0.675	0.675
0.016	0.094	0.103
0.393	0.625	0.617
0.455	0.719	0.758
0.845	0.847	0.806
0.320	0.503	0.528
0.761	0.750	0.728
0.378	0.678	0.622
0.285	0.492	0.542
0.030	0.125	0.133
0.137	0.286	0.322
0.273	0.558	0.553
0.690	0.764	0.775

(continued from previous page)

	Accuracy	
Coverage	Note to Note Alignment	Melodic Shape Alignment
0.331	0.497	0.486
0.003	0.028	0.025

Table D.5.: Raw results of the random pattern selection on MTC-ANN. On these results, the coverage values are binned into intervals of 0.05 to get an average accuracy per bin value.

D.3.2. Irish Folk Tunes Dataset

	Accuracy	
Coverage	Note to Note Alignment	Melodic Shape Alignment
0.239	0.711	0.711
0.291	0.565	0.522
0.550	0.750	0.629
0.332	0.728	0.719
0.239	0.587	0.584
0.083	0.270	0.287
0.667	0.857	0.787
0.091	0.264	0.250
0.192	0.553	0.534
0.403	0.680	0.640
0.128	0.402	0.379
0.114	0.525	0.553
0.020	0.132	0.126
0.173	0.548	0.601
0.094	0.301	0.351
0.041	0.267	0.267
0.163	0.329	0.301
0.465	0.795	0.764
0.597	0.857	0.848
0.737	0.888	0.860
0.541	0.840	0.761
0.078	0.166	0.143
0.129	0.458	0.511
0.153	0.346	0.239
0.103	0.371	0.390
0.378	0.607	0.579
0.540	0.789	0.770
0.184	0.514	0.553
0.774	0.876	0.801
0.018	0.087	0.101
0.122	0.537	0.573
0.800	0.904	0.843
0.386	0.736	0.638
0.051	0.149	0.107
0.576	0.829	0.792
0.648	0.846	0.767
0.405	0.730	0.691
0.013	0.051	0.034
0.422	0.699	0.646
0.126	0.340	0.323
0.290	0.590	0.390
0.149	0.410	0.390
0.663	0.851	0.750
0.767	0.879	0.812
0.262	0.640	0.649
0.300	0.587	0.303

(continued from previous page)

	Accuracy	
Coverage	**Note to Note Alignment**	**Melodic Shape Alignment**
0.682	0.840	0.708
0.112	0.506	0.584
0.199	0.514	0.615
0.181	0.472	0.506
0.365	0.739	0.756
0.125	0.334	0.295
0.245	0.663	0.593
0.508	0.789	0.663
0.156	0.433	0.441
0.255	0.539	0.551
0.327	0.593	0.520
0.089	0.264	0.228
0.000	0.003	0.003
0.358	0.711	0.654
0.298	0.643	0.640
0.135	0.362	0.275
0.458	0.756	0.638
0.026	0.110	0.062
0.441	0.784	0.702
0.189	0.514	0.528
0.092	0.317	0.337
0.685	0.846	0.753
0.127	0.525	0.601
0.453	0.688	0.570
0.510	0.722	0.596
0.290	0.525	0.360
0.311	0.674	0.646
0.564	0.764	0.688
0.476	0.772	0.713
0.585	0.857	0.750
0.656	0.871	0.708
0.470	0.787	0.666
0.597	0.848	0.823
0.351	0.596	0.480
0.143	0.424	0.427
0.166	0.413	0.303
0.063	0.612	0.626
0.396	0.680	0.708
0.080	0.371	0.458
0.604	0.868	0.764
0.433	0.733	0.677
0.696	0.874	0.798
0.519	0.761	0.604
0.031	0.154	0.183
0.163	0.649	0.694
0.310	0.764	0.795
0.606	0.820	0.767
0.766	0.896	0.846
0.215	0.500	0.427
0.731	0.868	0.806
0.550	0.789	0.626
0.118	0.354	0.399
0.068	0.410	0.452
0.022	0.076	0.076
0.280	0.671	0.685
0.002	0.011	0.008
0.689	0.860	0.775
0.169	0.598	0.534
0.632	0.831	0.829
0.005	0.006	0.008

(continued from previous page)

	Accuracy	
Coverage	Note to Note Alignment	Melodic Shape Alignment
0.088	0.309	0.340
0.015	0.051	0.039
0.216	0.494	0.508
0.061	0.447	0.461
0.207	0.528	0.494
0.221	0.635	0.663
0.259	0.739	0.756
0.516	0.638	0.565
0.131	0.421	0.388
0.345	0.590	0.579
0.583	0.815	0.716
0.718	0.879	0.820
0.040	0.169	0.163
0.484	0.756	0.643
0.185	0.573	0.604
0.053	0.357	0.382
0.118	0.567	0.570
0.175	0.581	0.553
0.752	0.860	0.756
0.091	0.419	0.421
0.002	0.003	0.006
0.151	0.489	0.416
0.496	0.764	0.739
0.213	0.565	0.576
0.360	0.772	0.716
0.256	0.559	0.508
0.306	0.649	0.646
0.019	0.067	0.053
0.546	0.775	0.711
0.323	0.553	0.376
0.496	0.671	0.593
0.160	0.402	0.430
0.344	0.638	0.666
0.060	0.334	0.301
0.556	0.817	0.815
0.292	0.607	0.556
0.480	0.756	0.680
0.141	0.478	0.514
0.060	0.292	0.306
0.592	0.846	0.803
0.544	0.772	0.739
0.440	0.812	0.750
0.054	0.185	0.174
0.562	0.798	0.669
0.609	0.831	0.725
0.007	0.014	0.014
0.272	0.579	0.528
0.510	0.750	0.683
0.158	0.503	0.416
0.225	0.534	0.492
0.009	0.025	0.022
0.648	0.843	0.744
0.008	0.042	0.042
0.254	0.556	0.520
0.047	0.306	0.331
0.397	0.756	0.705
0.019	0.093	0.070
0.287	0.556	0.539
0.019	0.110	0.090
0.117	0.618	0.677

(continued from previous page)

	Accuracy	
Coverage	**Note to Note Alignment**	**Melodic Shape Alignment**
0.577	0.815	0.713
0.413	0.770	0.758
0.037	0.107	0.065
0.533	0.775	0.725
0.239	0.520	0.492
0.228	0.539	0.562
0.486	0.772	0.646
0.132	0.621	0.646
0.211	0.486	0.506
0.735	0.885	0.767
0.434	0.713	0.612
0.675	0.860	0.809
0.103	0.739	0.744
0.732	0.851	0.728
0.463	0.694	0.640
0.142	0.548	0.646
0.496	0.753	0.713
0.090	0.281	0.278
0.708	0.888	0.848
0.401	0.649	0.483
0.190	0.433	0.379
0.032	0.371	0.348
0.195	0.427	0.430
0.426	0.798	0.775
0.057	0.180	0.166
0.533	0.817	0.801
0.001	0.000	0.003
0.061	0.351	0.362
0.243	0.643	0.663
0.625	0.831	0.806
0.060	0.278	0.326
0.674	0.871	0.817
0.348	0.666	0.654
0.479	0.713	0.539

Table D.6.: Raw results of the random pattern selection on the Irish Folk Tunes Dataset. On these results, the coverage values are binned into intervals of 0.05 to get an average accuracy per bin value.

D.4. Song Pattern Discovery on MTC-ANN

D.4.1. SIATEC

Parameters		GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
Min. patt. size	Min. compactness	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.1	3	0.992	0.925	0.992	0.861	0.951	0.908	0.951	0.853
0.1	4	0.985	0.919	0.985	0.844	0.931	0.900	0.931	0.839
0.1	5	0.966	0.897	0.966	0.814	0.884	0.861	0.884	0.783
0.1	6	0.909	0.858	0.909	0.756	0.799	0.781	0.799	0.647
0.1	7	0.830	0.744	0.830	0.681	0.703	0.689	0.703	0.586
0.3	3	0.982	0.919	0.982	0.864	0.802	0.861	0.802	0.806
0.3	4	0.934	0.894	0.934	0.819	0.723	0.769	0.723	0.706
0.3	5	0.845	0.794	0.845	0.697	0.619	0.697	0.619	0.583
0.3	6	0.726	0.678	0.726	0.614	0.513	0.578	0.513	0.475
0.3	7	0.616	0.597	0.616	0.536	0.425	0.536	0.425	0.489
0.5	3	0.944	0.878	0.944	0.817	0.641	0.783	0.641	0.761
0.5	4	0.805	0.792	0.805	0.697	0.508	0.669	0.508	0.506
0.5	5	0.635	0.669	0.635	0.581	0.378	0.597	0.378	0.406
0.5	6	0.506	0.564	0.506	0.514	0.291	0.525	0.291	0.444
0.5	7	0.423	0.506	0.423	0.456	0.238	0.467	0.238	0.406
0.7	3	0.833	0.800	0.833	0.736	0.464	0.644	0.464	0.567
0.7	4	0.603	0.631	0.603	0.569	0.332	0.561	0.332	0.439
0.7	5	0.450	0.547	0.450	0.478	0.244	0.517	0.244	0.394
0.7	6	0.374	0.500	0.374	0.478	0.199	0.486	0.199	0.411
0.7	7	0.342	0.478	0.342	0.461	0.181	0.456	0.181	0.433
0.9	3	0.758	0.728	0.758	0.647	0.384	0.617	0.384	0.511
0.9	4	0.516	0.603	0.516	0.525	0.267	0.553	0.267	0.383
0.9	5	0.371	0.503	0.371	0.456	0.193	0.492	0.193	0.372
0.9	6	0.321	0.475	0.321	0.458	0.165	0.469	0.165	0.422
0.9	7	0.295	0.428	0.295	0.419	0.151	0.428	0.151	0.386
1.0	3	0.757	0.725	0.757	0.642	0.383	0.611	0.383	0.511
1.0	4	0.515	0.603	0.515	0.525	0.266	0.556	0.266	0.383
1.0	5	0.369	0.503	0.369	0.453	0.191	0.497	0.191	0.372
1.0	6	0.318	0.467	0.318	0.461	0.163	0.469	0.163	0.422
1.0	7	0.291	0.425	0.291	0.417	0.148	0.425	0.148	0.389

Table D.7.: Raw results of SIATEC using Song Pattern Discovery on MTC-ANN.

D.4.2. COSIACTTEC

Parameters		GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
Min. patt. size	Min. compactness	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.1	3	0.759	0.692	0.759	0.564	0.223	0.236	0.223	0.275
0.1	4	0.706	0.625	0.706	0.542	0.194	0.219	0.194	0.231
0.1	5	0.646	0.586	0.646	0.517	0.167	0.208	0.167	0.236
0.1	6	0.601	0.564	0.601	0.433	0.143	0.194	0.143	0.117
0.1	7	0.563	0.553	0.563	0.478	0.130	0.231	0.130	0.136
0.3	3	0.751	0.661	0.751	0.583	0.236	0.236	0.236	0.269
0.3	4	0.672	0.606	0.672	0.528	0.212	0.261	0.212	0.208
0.3	5	0.597	0.556	0.597	0.492	0.185	0.239	0.185	0.211
0.3	6	0.541	0.544	0.541	0.436	0.164	0.272	0.164	0.206
0.3	7	0.491	0.525	0.491	0.467	0.144	0.306	0.144	0.164
0.5	3	0.720	0.672	0.720	0.611	0.244	0.314	0.244	0.294
0.5	4	0.599	0.561	0.599	0.497	0.217	0.311	0.217	0.256
0.5	5	0.506	0.561	0.506	0.444	0.192	0.314	0.192	0.225
0.5	6	0.438	0.514	0.438	0.428	0.170	0.317	0.170	0.242
0.5	7	0.385	0.486	0.385	0.433	0.148	0.353	0.148	0.250

(COSIACTTEC, continued from previous page)

Parameters		GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
Min. patt. size	Min. compactness	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.7	3	0.666	0.625	0.666	0.578	0.243	0.361	0.243	0.339
0.7	4	0.530	0.569	0.530	0.497	0.217	0.339	0.217	0.300
0.7	5	0.417	0.547	0.417	0.481	0.183	0.472	0.183	0.336
0.7	6	0.365	0.500	0.365	0.481	0.168	0.433	0.168	0.383
0.7	7	0.336	0.481	0.336	0.456	0.156	0.408	0.156	0.369
0.9	3	0.634	0.628	0.634	0.536	0.236	0.406	0.236	0.342
0.9	4	0.476	0.589	0.476	0.508	0.208	0.433	0.208	0.353
0.9	5	0.360	0.514	0.360	0.456	0.170	0.481	0.170	0.378
0.9	6	0.317	0.469	0.317	0.453	0.154	0.436	0.154	0.392
0.9	7	0.292	0.428	0.292	0.408	0.141	0.414	0.141	0.356
1.0	3	0.634	0.628	0.634	0.539	0.237	0.408	0.237	0.342
1.0	4	0.475	0.589	0.475	0.514	0.208	0.444	0.208	0.353
1.0	5	0.359	0.514	0.359	0.458	0.171	0.483	0.171	0.383
1.0	6	0.317	0.469	0.317	0.453	0.155	0.439	0.155	0.394
1.0	7	0.290	0.428	0.290	0.408	0.143	0.422	0.143	0.358

Table D.8.: Raw results of COSIACTTEC using Song Pattern Discovery on MTC-ANN.

D.4.3. MotivesExtractor

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
2	0.7	3	0	0.524	0.575	0.524	0.500	0.316	0.467	0.316	0.453
3	0.7	3	0	0.512	0.539	0.512	0.400	0.340	0.469	0.340	0.406
4	0.7	3	0	0.440	0.467	0.440	0.361	0.308	0.425	0.308	0.331
2	0.7	3	2	0.520	0.572	0.520	0.497	0.310	0.450	0.310	0.447
3	0.7	3	2	0.508	0.533	0.508	0.392	0.331	0.428	0.331	0.389
4	0.7	3	2	0.437	0.464	0.437	0.361	0.301	0.431	0.301	0.319
2	0.7	3	4	0.513	0.572	0.513	0.500	0.292	0.433	0.292	0.436
3	0.7	3	4	0.503	0.514	0.503	0.383	0.310	0.389	0.310	0.378
4	0.7	3	4	0.432	0.464	0.432	0.364	0.284	0.372	0.284	0.303
2	0.7	3	6	0.499	0.561	0.499	0.475	0.266	0.411	0.266	0.411
3	0.7	3	6	0.492	0.500	0.492	0.378	0.281	0.403	0.281	0.375
4	0.7	3	6	0.424	0.469	0.424	0.358	0.266	0.378	0.266	0.306
2	0.7	3	8	0.489	0.547	0.489	0.469	0.242	0.375	0.242	0.408
3	0.7	3	8	0.482	0.483	0.482	0.381	0.256	0.383	0.256	0.383
4	0.7	3	8	0.416	0.464	0.416	0.356	0.238	0.397	0.238	0.300
2	0.7	4	0	0.615	0.669	0.615	0.603	0.374	0.575	0.374	0.531
3	0.7	4	0	0.562	0.611	0.562	0.547	0.384	0.544	0.384	0.481
4	0.7	4	0	0.491	0.514	0.491	0.431	0.363	0.483	0.363	0.378
2	0.7	4	2	0.613	0.664	0.613	0.600	0.370	0.578	0.370	0.522
3	0.7	4	2	0.560	0.611	0.560	0.558	0.379	0.547	0.379	0.475
4	0.7	4	2	0.490	0.511	0.490	0.428	0.357	0.469	0.357	0.375
2	0.7	4	4	0.609	0.661	0.609	0.597	0.362	0.547	0.362	0.511
3	0.7	4	4	0.557	0.608	0.557	0.550	0.370	0.531	0.370	0.489
4	0.7	4	4	0.488	0.514	0.488	0.428	0.349	0.456	0.349	0.369
2	0.7	4	6	0.594	0.658	0.594	0.578	0.330	0.519	0.330	0.486
3	0.7	4	6	0.548	0.608	0.548	0.547	0.340	0.500	0.340	0.467
4	0.7	4	6	0.483	0.500	0.483	0.425	0.323	0.442	0.323	0.369
2	0.7	4	8	0.579	0.656	0.579	0.569	0.300	0.500	0.300	0.489
3	0.7	4	8	0.542	0.611	0.542	0.536	0.311	0.481	0.311	0.467
4	0.7	4	8	0.475	0.483	0.475	0.411	0.293	0.422	0.293	0.372
2	0.7	5	0	0.638	0.708	0.638	0.625	0.399	0.619	0.399	0.542
3	0.7	5	0	0.605	0.647	0.605	0.569	0.428	0.558	0.428	0.531
4	0.7	5	0	0.536	0.617	0.536	0.511	0.415	0.528	0.415	0.472
2	0.7	5	2	0.637	0.711	0.637	0.622	0.395	0.614	0.395	0.536

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
3	0.7	5	2	0.603	0.639	0.603	0.575	0.424	0.533	0.424	0.528
4	0.7	5	2	0.534	0.614	0.534	0.517	0.410	0.533	0.410	0.475
2	0.7	5	4	0.637	0.711	0.637	0.625	0.392	0.608	0.392	0.533
3	0.7	5	4	0.603	0.633	0.603	0.578	0.415	0.528	0.415	0.528
4	0.7	5	4	0.533	0.606	0.533	0.517	0.404	0.536	0.404	0.478
2	0.7	5	6	0.630	0.703	0.630	0.614	0.371	0.592	0.371	0.542
3	0.7	5	6	0.597	0.628	0.597	0.575	0.399	0.514	0.399	0.533
4	0.7	5	6	0.530	0.611	0.530	0.517	0.387	0.542	0.387	0.464
2	0.7	5	8	0.616	0.697	0.616	0.606	0.341	0.553	0.341	0.531
3	0.7	5	8	0.589	0.619	0.589	0.569	0.370	0.486	0.370	0.517
4	0.7	5	8	0.526	0.606	0.526	0.511	0.364	0.528	0.364	0.456
2	0.7	6	0	0.661	0.775	0.661	0.675	0.429	0.669	0.429	0.619
3	0.7	6	0	0.624	0.711	0.624	0.614	0.452	0.617	0.452	0.550
4	0.7	6	0	0.590	0.678	0.590	0.542	0.474	0.617	0.474	0.494
2	0.7	6	2	0.660	0.772	0.660	0.672	0.428	0.661	0.428	0.611
3	0.7	6	2	0.623	0.706	0.623	0.614	0.450	0.619	0.450	0.547
4	0.7	6	2	0.590	0.678	0.590	0.544	0.471	0.617	0.471	0.489
2	0.7	6	4	0.659	0.769	0.659	0.669	0.424	0.664	0.424	0.608
3	0.7	6	4	0.622	0.708	0.622	0.619	0.444	0.603	0.444	0.544
4	0.7	6	4	0.589	0.678	0.589	0.547	0.466	0.611	0.466	0.486
2	0.7	6	6	0.658	0.767	0.658	0.669	0.414	0.656	0.414	0.597
3	0.7	6	6	0.621	0.706	0.621	0.617	0.437	0.589	0.437	0.544
4	0.7	6	6	0.587	0.678	0.587	0.544	0.456	0.597	0.456	0.481
2	0.7	6	8	0.650	0.767	0.650	0.667	0.391	0.644	0.391	0.594
3	0.7	6	8	0.617	0.703	0.617	0.603	0.415	0.583	0.415	0.542
4	0.7	6	8	0.584	0.672	0.584	0.542	0.433	0.572	0.433	0.472
2	0.7	7	0	0.714	0.764	0.714	0.683	0.492	0.706	0.492	0.628
3	0.7	7	0	0.655	0.761	0.655	0.622	0.490	0.694	0.490	0.589
4	0.7	7	0	0.630	0.692	0.630	0.611	0.521	0.644	0.521	0.569
2	0.7	7	2	0.714	0.764	0.714	0.683	0.491	0.703	0.491	0.628
3	0.7	7	2	0.655	0.761	0.655	0.622	0.488	0.686	0.488	0.589
4	0.7	7	2	0.630	0.697	0.630	0.611	0.519	0.644	0.519	0.567
2	0.7	7	4	0.714	0.764	0.714	0.683	0.488	0.700	0.488	0.614
3	0.7	7	4	0.654	0.758	0.654	0.622	0.484	0.681	0.484	0.592
4	0.7	7	4	0.629	0.697	0.629	0.617	0.516	0.656	0.516	0.572
2	0.7	7	6	0.713	0.767	0.713	0.683	0.485	0.700	0.485	0.614
3	0.7	7	6	0.653	0.756	0.653	0.628	0.478	0.681	0.478	0.600
4	0.7	7	6	0.629	0.697	0.629	0.614	0.512	0.650	0.512	0.567
2	0.7	7	8	0.711	0.769	0.711	0.681	0.470	0.678	0.470	0.614
3	0.7	7	8	0.652	0.758	0.652	0.622	0.464	0.672	0.464	0.592
4	0.7	7	8	0.628	0.703	0.628	0.611	0.499	0.656	0.499	0.572
2	0.5	3	0	0.532	0.572	0.532	0.503	0.320	0.475	0.320	0.447
3	0.5	3	0	0.912	0.861	0.912	0.761	0.782	0.811	0.782	0.697
4	0.5	3	0	0.975	0.908	0.975	0.844	0.930	0.892	0.930	0.828
2	0.5	3	2	0.528	0.578	0.528	0.500	0.313	0.461	0.313	0.439
3	0.5	3	2	0.910	0.858	0.910	0.761	0.771	0.814	0.771	0.703
4	0.5	3	2	0.975	0.908	0.975	0.842	0.927	0.897	0.927	0.803
2	0.5	3	4	0.521	0.581	0.521	0.500	0.295	0.433	0.295	0.422
3	0.5	3	4	0.904	0.861	0.904	0.753	0.731	0.764	0.731	0.664
4	0.5	3	4	0.972	0.908	0.972	0.844	0.904	0.894	0.904	0.758
2	0.5	3	6	0.507	0.564	0.507	0.478	0.268	0.419	0.268	0.403
3	0.5	3	6	0.895	0.856	0.895	0.731	0.656	0.692	0.656	0.556
4	0.5	3	6	0.970	0.908	0.970	0.850	0.860	0.875	0.860	0.731
2	0.5	3	8	0.496	0.550	0.496	0.472	0.244	0.383	0.244	0.406
3	0.5	3	8	0.884	0.856	0.884	0.722	0.583	0.669	0.583	0.514
4	0.5	3	8	0.967	0.908	0.967	0.850	0.817	0.844	0.817	0.697
2	0.5	4	0	0.616	0.672	0.616	0.603	0.375	0.575	0.375	0.533
3	0.5	4	0	0.823	0.789	0.823	0.694	0.664	0.717	0.664	0.644
4	0.5	4	0	0.943	0.881	0.943	0.800	0.887	0.861	0.887	0.761
2	0.5	4	2	0.614	0.667	0.614	0.600	0.371	0.578	0.371	0.525

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
3	0.5	4	2	0.822	0.792	0.822	0.692	0.657	0.719	0.657	0.644
4	0.5	4	2	0.942	0.881	0.942	0.800	0.884	0.858	0.884	0.750
2	0.5	4	4	0.610	0.664	0.610	0.597	0.362	0.547	0.362	0.514
3	0.5	4	4	0.819	0.794	0.819	0.689	0.642	0.703	0.642	0.617
4	0.5	4	4	0.940	0.881	0.940	0.803	0.874	0.847	0.874	0.747
2	0.5	4	6	0.595	0.661	0.595	0.578	0.330	0.519	0.330	0.489
3	0.5	4	6	0.813	0.794	0.813	0.683	0.598	0.658	0.598	0.583
4	0.5	4	6	0.938	0.881	0.938	0.794	0.850	0.822	0.850	0.739
2	0.5	4	8	0.580	0.658	0.580	0.569	0.300	0.503	0.300	0.492
3	0.5	4	8	0.804	0.781	0.804	0.658	0.542	0.614	0.542	0.528
4	0.5	4	8	0.935	0.878	0.935	0.792	0.813	0.808	0.813	0.708
2	0.5	5	0	0.640	0.711	0.640	0.628	0.400	0.622	0.400	0.542
3	0.5	5	0	0.777	0.753	0.777	0.669	0.616	0.697	0.616	0.600
4	0.5	5	0	0.910	0.881	0.910	0.783	0.858	0.833	0.858	0.750
2	0.5	5	2	0.639	0.714	0.639	0.622	0.397	0.617	0.397	0.533
3	0.5	5	2	0.777	0.756	0.777	0.669	0.612	0.686	0.612	0.594
4	0.5	5	2	0.910	0.878	0.910	0.783	0.856	0.833	0.856	0.756
2	0.5	5	4	0.639	0.714	0.639	0.625	0.393	0.611	0.393	0.531
3	0.5	5	4	0.776	0.753	0.776	0.667	0.602	0.683	0.602	0.592
4	0.5	5	4	0.909	0.878	0.909	0.781	0.851	0.833	0.851	0.747
2	0.5	5	6	0.632	0.706	0.632	0.614	0.372	0.594	0.372	0.544
3	0.5	5	6	0.771	0.750	0.771	0.658	0.578	0.664	0.578	0.575
4	0.5	5	6	0.909	0.872	0.909	0.778	0.837	0.833	0.837	0.744
2	0.5	5	8	0.618	0.700	0.618	0.603	0.342	0.556	0.342	0.531
3	0.5	5	8	0.762	0.739	0.762	0.653	0.532	0.611	0.532	0.533
4	0.5	5	8	0.907	0.872	0.907	0.772	0.810	0.808	0.810	0.708
2	0.5	6	0	0.662	0.778	0.662	0.678	0.430	0.672	0.430	0.619
3	0.5	6	0	0.754	0.756	0.754	0.656	0.595	0.669	0.595	0.600
4	0.5	6	0	0.889	0.861	0.889	0.736	0.833	0.822	0.833	0.742
2	0.5	6	2	0.662	0.775	0.662	0.675	0.429	0.669	0.429	0.611
3	0.5	6	2	0.754	0.756	0.754	0.653	0.593	0.669	0.593	0.597
4	0.5	6	2	0.889	0.861	0.889	0.736	0.832	0.825	0.832	0.739
2	0.5	6	4	0.661	0.772	0.661	0.672	0.425	0.667	0.425	0.608
3	0.5	6	4	0.752	0.758	0.752	0.653	0.584	0.675	0.584	0.597
4	0.5	6	4	0.889	0.864	0.889	0.736	0.828	0.819	0.828	0.731
2	0.5	6	6	0.659	0.769	0.659	0.672	0.415	0.664	0.415	0.597
3	0.5	6	6	0.751	0.756	0.751	0.656	0.576	0.656	0.576	0.589
4	0.5	6	6	0.888	0.861	0.888	0.736	0.820	0.817	0.820	0.733
2	0.5	6	8	0.651	0.769	0.651	0.669	0.392	0.647	0.392	0.594
3	0.5	6	8	0.748	0.753	0.748	0.644	0.548	0.656	0.548	0.572
4	0.5	6	8	0.887	0.856	0.887	0.733	0.799	0.800	0.799	0.725
2	0.5	7	0	0.715	0.769	0.715	0.692	0.493	0.706	0.493	0.628
3	0.5	7	0	0.759	0.811	0.759	0.667	0.603	0.731	0.603	0.608
4	0.5	7	0	0.883	0.861	0.883	0.778	0.834	0.836	0.834	0.767
2	0.5	7	2	0.715	0.769	0.715	0.689	0.492	0.703	0.492	0.628
3	0.5	7	2	0.758	0.811	0.758	0.667	0.601	0.731	0.601	0.606
4	0.5	7	2	0.883	0.861	0.883	0.778	0.833	0.836	0.833	0.772
2	0.5	7	4	0.715	0.769	0.715	0.692	0.489	0.697	0.489	0.617
3	0.5	7	4	0.758	0.808	0.758	0.664	0.596	0.725	0.596	0.614
4	0.5	7	4	0.883	0.861	0.883	0.775	0.831	0.833	0.831	0.764
2	0.5	7	6	0.714	0.772	0.714	0.689	0.485	0.697	0.485	0.619
3	0.5	7	6	0.756	0.803	0.756	0.667	0.588	0.711	0.588	0.617
4	0.5	7	6	0.883	0.861	0.883	0.775	0.828	0.831	0.828	0.772
2	0.5	7	8	0.712	0.772	0.712	0.689	0.471	0.678	0.471	0.614
3	0.5	7	8	0.755	0.803	0.755	0.661	0.575	0.692	0.575	0.606
4	0.5	7	8	0.883	0.861	0.883	0.778	0.818	0.825	0.818	0.767
2	0.3	3	0	0.996	0.911	0.996	0.858	0.943	0.914	0.943	0.856
3	0.3	3	0	0.996	0.911	0.996	0.867	0.994	0.911	0.994	0.864
4	0.3	3	0	0.996	0.911	0.996	0.858	0.994	0.911	0.994	0.864
2	0.3	3	2	0.996	0.911	0.996	0.861	0.940	0.908	0.940	0.853

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
3	0.3	3	2	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.3	3	2	0.996	0.911	0.996	0.867	0.994	0.911	0.994	0.864
2	0.3	3	4	0.995	0.911	0.995	0.861	0.911	0.894	0.911	0.844
3	0.3	3	4	0.996	0.911	0.996	0.867	0.994	0.911	0.994	0.864
4	0.3	3	4	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.861
2	0.3	3	6	0.995	0.911	0.995	0.867	0.848	0.861	0.848	0.808
3	0.3	3	6	0.996	0.911	0.996	0.858	0.994	0.911	0.994	0.864
4	0.3	3	6	0.996	0.911	0.996	0.858	0.994	0.911	0.994	0.867
2	0.3	3	8	0.995	0.911	0.995	0.861	0.761	0.811	0.761	0.775
3	0.3	3	8	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.3	3	8	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.3	4	0	0.993	0.914	0.993	0.856	0.906	0.911	0.906	0.850
3	0.3	4	0	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.3	4	0	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.3	4	2	0.993	0.914	0.993	0.856	0.903	0.908	0.903	0.850
3	0.3	4	2	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.3	4	2	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.861
2	0.3	4	4	0.993	0.914	0.993	0.853	0.890	0.908	0.890	0.836
3	0.3	4	4	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.861
4	0.3	4	4	0.996	0.911	0.996	0.858	0.994	0.911	0.994	0.861
2	0.3	4	6	0.992	0.914	0.992	0.856	0.840	0.853	0.840	0.819
3	0.3	4	6	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.3	4	6	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.3	4	8	0.991	0.917	0.991	0.847	0.771	0.836	0.771	0.781
3	0.3	4	8	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.867
4	0.3	4	8	0.996	0.911	0.996	0.858	0.994	0.911	0.994	0.864
2	0.3	5	0	0.987	0.917	0.987	0.847	0.884	0.875	0.884	0.822
3	0.3	5	0	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.3	5	0	0.996	0.911	0.996	0.867	0.994	0.911	0.994	0.864
2	0.3	5	2	0.987	0.917	0.987	0.847	0.880	0.883	0.880	0.814
3	0.3	5	2	0.996	0.911	0.996	0.864	0.994	0.911	0.994	0.864
4	0.3	5	2	0.996	0.911	0.996	0.858	0.994	0.911	0.994	0.864
2	0.3	5	4	0.987	0.917	0.987	0.847	0.870	0.892	0.870	0.814
3	0.3	5	4	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.3	5	4	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.3	5	6	0.986	0.917	0.986	0.847	0.830	0.867	0.830	0.803
3	0.3	5	6	0.996	0.911	0.996	0.867	0.994	0.911	0.994	0.864
4	0.3	5	6	0.996	0.911	0.996	0.858	0.994	0.911	0.994	0.864
2	0.3	5	8	0.985	0.919	0.985	0.858	0.781	0.872	0.781	0.778
3	0.3	5	8	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.3	5	8	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.867
2	0.3	6	0	0.977	0.914	0.977	0.842	0.854	0.847	0.854	0.811
3	0.3	6	0	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.3	6	0	0.996	0.911	0.996	0.858	0.994	0.911	0.994	0.864
2	0.3	6	2	0.977	0.914	0.977	0.842	0.852	0.850	0.852	0.811
3	0.3	6	2	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.3	6	2	0.996	0.911	0.996	0.858	0.994	0.911	0.994	0.864
2	0.3	6	4	0.977	0.914	0.977	0.842	0.845	0.847	0.845	0.797
3	0.3	6	4	0.996	0.911	0.996	0.867	0.994	0.911	0.994	0.864
4	0.3	6	4	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.3	6	6	0.976	0.914	0.976	0.842	0.827	0.836	0.827	0.797
3	0.3	6	6	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.867
4	0.3	6	6	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.3	6	8	0.976	0.914	0.976	0.842	0.788	0.825	0.788	0.775
3	0.3	6	8	0.996	0.911	0.996	0.858	0.994	0.911	0.994	0.864
4	0.3	6	8	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.3	7	0	0.962	0.903	0.962	0.844	0.830	0.856	0.830	0.786
3	0.3	7	0	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.3	7	0	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.861
2	0.3	7	2	0.962	0.903	0.962	0.842	0.829	0.856	0.829	0.789

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
3	0.3	7	2	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.3	7	2	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.3	7	4	0.962	0.903	0.962	0.844	0.826	0.847	0.826	0.789
3	0.3	7	4	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.861
4	0.3	7	4	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.3	7	6	0.961	0.897	0.961	0.844	0.819	0.847	0.819	0.789
3	0.3	7	6	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.3	7	6	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.3	7	8	0.960	0.897	0.960	0.850	0.792	0.833	0.792	0.783
3	0.3	7	8	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.867
4	0.3	7	8	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.1	3	0	0.996	0.911	0.996	0.861	0.996	0.911	0.996	0.861
3	0.1	3	0	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.1	3	0	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.867
2	0.1	3	2	0.996	0.911	0.996	0.861	0.996	0.911	0.996	0.861
3	0.1	3	2	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.1	3	2	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.867
2	0.1	3	4	0.996	0.911	0.996	0.861	0.994	0.914	0.994	0.861
3	0.1	3	4	0.996	0.911	0.996	0.864	0.994	0.911	0.994	0.861
4	0.1	3	4	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.1	3	6	0.996	0.911	0.996	0.861	0.990	0.914	0.990	0.853
3	0.1	3	6	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.1	3	6	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.861
2	0.1	3	8	0.996	0.911	0.996	0.861	0.986	0.908	0.986	0.861
3	0.1	3	8	0.996	0.911	0.996	0.864	0.994	0.911	0.994	0.864
4	0.1	3	8	0.996	0.911	0.996	0.858	0.994	0.911	0.994	0.867
2	0.1	4	0	0.996	0.911	0.996	0.861	0.996	0.911	0.996	0.858
3	0.1	4	0	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.861
4	0.1	4	0	0.996	0.911	0.996	0.867	0.994	0.911	0.994	0.864
2	0.1	4	2	0.996	0.911	0.996	0.861	0.995	0.914	0.995	0.858
3	0.1	4	2	0.996	0.911	0.996	0.864	0.994	0.911	0.994	0.864
4	0.1	4	2	0.996	0.911	0.996	0.867	0.994	0.911	0.994	0.864
2	0.1	4	4	0.996	0.911	0.996	0.861	0.994	0.914	0.994	0.861
3	0.1	4	4	0.996	0.911	0.996	0.858	0.994	0.911	0.994	0.864
4	0.1	4	4	0.996	0.911	0.996	0.867	0.994	0.911	0.994	0.864
2	0.1	4	6	0.996	0.911	0.996	0.861	0.994	0.914	0.994	0.861
3	0.1	4	6	0.996	0.911	0.996	0.858	0.994	0.911	0.994	0.867
4	0.1	4	6	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.1	4	8	0.996	0.911	0.996	0.861	0.992	0.911	0.992	0.861
3	0.1	4	8	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.1	4	8	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.1	5	0	0.996	0.911	0.996	0.861	0.995	0.911	0.995	0.861
3	0.1	5	0	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.1	5	0	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.861
2	0.1	5	2	0.996	0.911	0.996	0.861	0.995	0.911	0.995	0.861
3	0.1	5	2	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.1	5	2	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.1	5	4	0.996	0.911	0.996	0.861	0.994	0.914	0.994	0.858
3	0.1	5	4	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.1	5	4	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.1	5	6	0.996	0.911	0.996	0.858	0.992	0.914	0.992	0.864
3	0.1	5	6	0.996	0.911	0.996	0.864	0.994	0.911	0.994	0.864
4	0.1	5	6	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.1	5	8	0.996	0.911	0.996	0.861	0.990	0.911	0.990	0.853
3	0.1	5	8	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.869
4	0.1	5	8	0.996	0.911	0.996	0.858	0.994	0.911	0.994	0.864
2	0.1	6	0	0.996	0.911	0.996	0.861	0.995	0.911	0.995	0.861
3	0.1	6	0	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.1	6	0	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.1	6	2	0.996	0.911	0.996	0.861	0.995	0.911	0.995	0.867

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
3	0.1	6	2	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.1	6	2	0.996	0.911	0.996	0.858	0.994	0.911	0.994	0.864
2	0.1	6	4	0.996	0.911	0.996	0.861	0.993	0.911	0.993	0.858
3	0.1	6	4	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.861
4	0.1	6	4	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.1	6	6	0.996	0.911	0.996	0.861	0.992	0.911	0.992	0.861
3	0.1	6	6	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.1	6	6	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.1	6	8	0.996	0.911	0.996	0.867	0.991	0.911	0.991	0.858
3	0.1	6	8	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.1	6	8	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
2	0.1	7	0	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
3	0.1	7	0	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.861
4	0.1	7	0	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.867
2	0.1	7	2	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
3	0.1	7	2	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.1	7	2	0.996	0.911	0.996	0.858	0.994	0.911	0.994	0.864
2	0.1	7	4	0.996	0.911	0.996	0.858	0.993	0.911	0.993	0.864
3	0.1	7	4	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.1	7	4	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.861
2	0.1	7	6	0.996	0.911	0.996	0.867	0.993	0.911	0.993	0.864
3	0.1	7	6	0.996	0.911	0.996	0.867	0.994	0.911	0.994	0.864
4	0.1	7	6	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.867
2	0.1	7	8	0.996	0.911	0.996	0.861	0.991	0.911	0.991	0.856
3	0.1	7	8	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864
4	0.1	7	8	0.996	0.911	0.996	0.861	0.994	0.911	0.994	0.864

Table D.9.: Raw results of MotivesExtractor using Song Pattern Discovery on MTC-ANN. The parameters are the path tracing tolerance (ρ), the scoring threshold (θ), the minimum pattern length (ν) and the tolerance parameter for the pattern clustering (Θ).

D.4.4. SIARCT-CFP

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
b	a	m	c	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
0.5	3	0.7	0.3	0.992	0.911	0.992	0.856	0.851	0.847	0.851	0.806
0.5	3	0.9	0.3	0.985	0.925	0.985	0.853	0.864	0.850	0.864	0.819
0.5	3	1.0	0.3	0.983	0.919	0.983	0.856	0.864	0.850	0.864	0.819
0.5	3	0.7	0.5	0.992	0.911	0.992	0.856	0.851	0.847	0.851	0.806
0.5	3	0.9	0.5	0.985	0.925	0.985	0.853	0.864	0.850	0.864	0.822
0.5	3	1.0	0.5	0.983	0.919	0.983	0.856	0.864	0.850	0.864	0.819
0.5	3	0.7	0.7	0.992	0.911	0.992	0.856	0.851	0.847	0.851	0.806
0.5	3	0.9	0.7	0.985	0.925	0.985	0.853	0.864	0.850	0.864	0.819
0.5	3	1.0	0.7	0.983	0.919	0.983	0.856	0.864	0.850	0.864	0.819
0.5	3	0.7	0.9	0.992	0.911	0.992	0.853	0.851	0.847	0.851	0.806
0.5	3	0.9	0.9	0.985	0.925	0.985	0.853	0.864	0.850	0.864	0.819
0.5	3	1.0	0.9	0.983	0.919	0.983	0.853	0.864	0.850	0.864	0.819
0.7	3	0.7	0.3	0.987	0.906	0.987	0.853	0.773	0.819	0.773	0.800
0.7	3	0.9	0.3	0.979	0.919	0.979	0.850	0.794	0.825	0.794	0.803
0.7	3	1.0	0.3	0.976	0.917	0.976	0.839	0.794	0.825	0.794	0.803
0.7	3	0.7	0.5	0.987	0.906	0.987	0.856	0.773	0.819	0.773	0.803
0.7	3	0.9	0.5	0.979	0.919	0.979	0.850	0.794	0.825	0.794	0.800
0.7	3	1.0	0.5	0.976	0.917	0.976	0.839	0.794	0.825	0.794	0.803
0.7	3	0.7	0.7	0.987	0.906	0.987	0.856	0.773	0.819	0.773	0.803
0.7	3	0.9	0.7	0.979	0.919	0.979	0.850	0.794	0.825	0.794	0.803
0.7	3	1.0	0.7	0.976	0.917	0.976	0.839	0.794	0.825	0.794	0.803

(SIARCT-CFP, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
b	*a*	*m*	*c*	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
0.7	3	0.7	0.9	0.987	0.906	0.987	0.856	0.773	0.819	0.773	0.800
0.7	3	0.9	0.9	0.979	0.919	0.979	0.850	0.794	0.825	0.794	0.803
0.7	3	1.0	0.9	0.976	0.917	0.976	0.839	0.794	0.825	0.794	0.803
0.9	3	0.7	0.3	0.975	0.894	0.975	0.836	0.679	0.769	0.679	0.761
0.9	3	0.9	0.3	0.963	0.911	0.963	0.822	0.707	0.781	0.707	0.775
0.9	3	1.0	0.3	0.959	0.906	0.959	0.819	0.707	0.783	0.707	0.775
0.9	3	0.7	0.5	0.975	0.894	0.975	0.839	0.679	0.769	0.679	0.758
0.9	3	0.9	0.5	0.963	0.911	0.963	0.822	0.707	0.781	0.707	0.775
0.9	3	1.0	0.5	0.959	0.906	0.959	0.819	0.707	0.783	0.707	0.775
0.9	3	0.7	0.7	0.975	0.894	0.975	0.839	0.679	0.769	0.679	0.758
0.9	3	0.9	0.7	0.963	0.911	0.963	0.822	0.707	0.781	0.707	0.775
0.9	3	1.0	0.7	0.959	0.906	0.959	0.817	0.707	0.783	0.707	0.775
0.9	3	0.7	0.9	0.975	0.894	0.975	0.839	0.679	0.769	0.679	0.758
0.9	3	0.9	0.9	0.963	0.911	0.963	0.822	0.707	0.781	0.707	0.775
0.9	3	1.0	0.9	0.959	0.906	0.959	0.817	0.707	0.783	0.707	0.775
1.0	3	0.7	0.3	0.975	0.894	0.975	0.839	0.678	0.772	0.678	0.750
1.0	3	0.9	0.3	0.963	0.911	0.963	0.819	0.706	0.789	0.706	0.772
1.0	3	1.0	0.3	0.959	0.906	0.959	0.819	0.707	0.781	0.707	0.772
1.0	3	0.7	0.5	0.975	0.894	0.975	0.839	0.678	0.772	0.678	0.750
1.0	3	0.9	0.5	0.963	0.911	0.963	0.822	0.706	0.789	0.706	0.772
1.0	3	1.0	0.5	0.959	0.906	0.959	0.819	0.707	0.781	0.707	0.772
1.0	3	0.7	0.7	0.975	0.894	0.975	0.839	0.678	0.772	0.678	0.753
1.0	3	0.9	0.7	0.963	0.911	0.963	0.819	0.706	0.789	0.706	0.769
1.0	3	1.0	0.7	0.959	0.906	0.959	0.817	0.707	0.781	0.707	0.772
1.0	3	0.7	0.9	0.975	0.894	0.975	0.839	0.678	0.772	0.678	0.753
1.0	3	0.9	0.9	0.963	0.911	0.963	0.819	0.706	0.789	0.706	0.772
1.0	3	1.0	0.9	0.959	0.906	0.959	0.817	0.707	0.781	0.707	0.772
0.5	4	0.7	0.3	0.977	0.908	0.977	0.833	0.817	0.814	0.817	0.767
0.5	4	0.9	0.3	0.967	0.911	0.967	0.833	0.828	0.831	0.828	0.778
0.5	4	1.0	0.3	0.965	0.903	0.965	0.839	0.829	0.833	0.829	0.778
0.5	4	0.7	0.5	0.977	0.908	0.977	0.833	0.817	0.814	0.817	0.767
0.5	4	0.9	0.5	0.967	0.911	0.967	0.833	0.828	0.831	0.828	0.778
0.5	4	1.0	0.5	0.965	0.903	0.965	0.836	0.829	0.833	0.829	0.778
0.5	4	0.7	0.7	0.977	0.908	0.977	0.833	0.817	0.814	0.817	0.767
0.5	4	0.9	0.7	0.967	0.911	0.967	0.833	0.828	0.831	0.828	0.778
0.5	4	1.0	0.7	0.965	0.903	0.965	0.839	0.829	0.833	0.829	0.781
0.5	4	0.7	0.9	0.977	0.908	0.977	0.833	0.817	0.814	0.817	0.767
0.5	4	0.9	0.9	0.967	0.911	0.967	0.833	0.828	0.831	0.828	0.778
0.5	4	1.0	0.9	0.965	0.903	0.965	0.836	0.829	0.833	0.829	0.778
0.7	4	0.7	0.3	0.944	0.869	0.944	0.819	0.689	0.708	0.689	0.683
0.7	4	0.9	0.3	0.927	0.858	0.927	0.789	0.712	0.722	0.712	0.700
0.7	4	1.0	0.3	0.922	0.858	0.922	0.792	0.713	0.725	0.713	0.700
0.7	4	0.7	0.5	0.944	0.869	0.944	0.814	0.689	0.708	0.689	0.683
0.7	4	0.9	0.5	0.927	0.858	0.927	0.789	0.712	0.722	0.712	0.700
0.7	4	1.0	0.5	0.922	0.858	0.922	0.792	0.713	0.725	0.713	0.700
0.7	4	0.7	0.7	0.944	0.869	0.944	0.817	0.689	0.708	0.689	0.683
0.7	4	0.9	0.7	0.927	0.858	0.927	0.786	0.712	0.722	0.712	0.700
0.7	4	1.0	0.7	0.922	0.858	0.922	0.792	0.713	0.725	0.713	0.700
0.7	4	0.7	0.9	0.944	0.869	0.944	0.811	0.689	0.708	0.689	0.683
0.7	4	0.9	0.9	0.927	0.858	0.927	0.789	0.712	0.722	0.712	0.700
0.7	4	1.0	0.9	0.922	0.858	0.922	0.792	0.713	0.725	0.713	0.700
0.9	4	0.7	0.3	0.874	0.794	0.874	0.703	0.534	0.603	0.534	0.500
0.9	4	0.9	0.3	0.838	0.753	0.838	0.694	0.563	0.633	0.563	0.522
0.9	4	1.0	0.3	0.830	0.750	0.830	0.694	0.564	0.636	0.564	0.525
0.9	4	0.7	0.5	0.874	0.794	0.874	0.703	0.534	0.603	0.534	0.500
0.9	4	0.9	0.5	0.838	0.753	0.838	0.694	0.563	0.633	0.563	0.522
0.9	4	1.0	0.5	0.830	0.750	0.830	0.694	0.564	0.636	0.564	0.525
0.9	4	0.7	0.7	0.874	0.794	0.874	0.703	0.534	0.603	0.534	0.500
0.9	4	0.9	0.7	0.838	0.753	0.838	0.694	0.563	0.633	0.563	0.522
0.9	4	1.0	0.7	0.830	0.750	0.830	0.694	0.564	0.636	0.564	0.525

(SIARCT-CFP, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
b	a	m	c	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
0.9	4	0.7	0.9	0.874	0.794	0.874	0.703	0.534	0.603	0.534	0.500
0.9	4	0.9	0.9	0.838	0.753	0.838	0.694	0.563	0.633	0.563	0.522
0.9	4	1.0	0.9	0.830	0.750	0.830	0.694	0.564	0.636	0.564	0.525
1.0	4	0.7	0.3	0.874	0.794	0.874	0.703	0.532	0.611	0.532	0.500
1.0	4	0.9	0.3	0.838	0.753	0.838	0.694	0.562	0.639	0.562	0.522
1.0	4	1.0	0.3	0.830	0.750	0.830	0.694	0.563	0.642	0.563	0.525
1.0	4	0.7	0.5	0.874	0.794	0.874	0.703	0.532	0.611	0.532	0.500
1.0	4	0.9	0.5	0.838	0.753	0.838	0.694	0.562	0.639	0.562	0.522
1.0	4	1.0	0.5	0.830	0.750	0.830	0.694	0.563	0.642	0.563	0.525
1.0	4	0.7	0.7	0.874	0.794	0.874	0.703	0.532	0.611	0.532	0.500
1.0	4	0.9	0.7	0.838	0.753	0.838	0.694	0.562	0.639	0.562	0.522
1.0	4	1.0	0.7	0.830	0.750	0.830	0.694	0.563	0.642	0.563	0.525
1.0	4	0.7	0.9	0.874	0.794	0.874	0.703	0.532	0.611	0.532	0.500
1.0	4	0.9	0.9	0.838	0.753	0.838	0.694	0.562	0.639	0.562	0.522
1.0	4	1.0	0.9	0.830	0.750	0.830	0.694	0.563	0.642	0.563	0.525
0.5	5	0.7	0.3	0.925	0.861	0.925	0.758	0.746	0.756	0.746	0.681
0.5	5	0.9	0.3	0.918	0.844	0.918	0.764	0.752	0.761	0.752	0.700
0.5	5	1.0	0.3	0.915	0.842	0.915	0.756	0.752	0.761	0.752	0.700
0.5	5	0.7	0.5	0.925	0.861	0.925	0.758	0.746	0.756	0.746	0.681
0.5	5	0.9	0.5	0.918	0.844	0.918	0.764	0.752	0.761	0.752	0.700
0.5	5	1.0	0.5	0.915	0.842	0.915	0.756	0.752	0.761	0.752	0.700
0.5	5	0.7	0.7	0.925	0.861	0.925	0.758	0.746	0.756	0.746	0.681
0.5	5	0.9	0.7	0.918	0.844	0.918	0.764	0.752	0.761	0.752	0.703
0.5	5	1.0	0.7	0.915	0.842	0.915	0.756	0.752	0.761	0.752	0.700
0.5	5	0.7	0.9	0.925	0.861	0.925	0.761	0.746	0.756	0.746	0.681
0.5	5	0.9	0.9	0.918	0.844	0.918	0.767	0.752	0.761	0.752	0.697
0.5	5	1.0	0.9	0.915	0.842	0.915	0.756	0.752	0.761	0.752	0.700
0.7	5	0.7	0.3	0.837	0.767	0.837	0.717	0.572	0.639	0.572	0.553
0.7	5	0.9	0.3	0.817	0.758	0.817	0.686	0.584	0.633	0.584	0.561
0.7	5	1.0	0.3	0.814	0.758	0.814	0.692	0.584	0.631	0.584	0.561
0.7	5	0.7	0.5	0.837	0.767	0.837	0.719	0.572	0.639	0.572	0.553
0.7	5	0.9	0.5	0.817	0.758	0.817	0.686	0.584	0.633	0.584	0.561
0.7	5	1.0	0.5	0.814	0.758	0.814	0.692	0.584	0.631	0.584	0.561
0.7	5	0.7	0.7	0.837	0.767	0.837	0.714	0.572	0.639	0.572	0.553
0.7	5	0.9	0.7	0.817	0.758	0.817	0.686	0.584	0.633	0.584	0.561
0.7	5	1.0	0.7	0.814	0.758	0.814	0.692	0.584	0.631	0.584	0.561
0.7	5	0.7	0.9	0.837	0.767	0.837	0.717	0.572	0.639	0.572	0.553
0.7	5	0.9	0.9	0.817	0.758	0.817	0.683	0.584	0.633	0.584	0.561
0.7	5	1.0	0.9	0.814	0.758	0.814	0.692	0.584	0.631	0.584	0.561
0.9	5	0.7	0.3	0.637	0.625	0.637	0.528	0.357	0.492	0.357	0.411
0.9	5	0.9	0.3	0.604	0.603	0.604	0.514	0.366	0.517	0.366	0.419
0.9	5	1.0	0.3	0.599	0.594	0.599	0.517	0.366	0.511	0.366	0.422
0.9	5	0.7	0.5	0.637	0.625	0.637	0.528	0.357	0.492	0.357	0.411
0.9	5	0.9	0.5	0.604	0.603	0.604	0.511	0.366	0.517	0.366	0.419
0.9	5	1.0	0.5	0.599	0.594	0.599	0.517	0.366	0.511	0.366	0.422
0.9	5	0.7	0.7	0.637	0.625	0.637	0.528	0.357	0.492	0.357	0.411
0.9	5	0.9	0.7	0.604	0.603	0.604	0.511	0.366	0.517	0.366	0.419
0.9	5	1.0	0.7	0.599	0.594	0.599	0.517	0.366	0.511	0.366	0.422
0.9	5	0.7	0.9	0.637	0.625	0.637	0.528	0.357	0.492	0.357	0.411
0.9	5	0.9	0.9	0.604	0.603	0.604	0.511	0.366	0.517	0.366	0.419
0.9	5	1.0	0.9	0.599	0.594	0.599	0.517	0.366	0.511	0.366	0.422
1.0	5	0.7	0.3	0.637	0.625	0.637	0.528	0.355	0.494	0.355	0.400
1.0	5	0.9	0.3	0.603	0.608	0.603	0.517	0.364	0.519	0.364	0.408
1.0	5	1.0	0.3	0.598	0.600	0.598	0.514	0.365	0.519	0.365	0.408
1.0	5	0.7	0.5	0.637	0.625	0.637	0.528	0.355	0.494	0.355	0.400
1.0	5	0.9	0.5	0.603	0.608	0.603	0.517	0.364	0.519	0.364	0.408
1.0	5	1.0	0.5	0.598	0.600	0.598	0.514	0.365	0.519	0.365	0.408
1.0	5	0.7	0.7	0.637	0.625	0.637	0.528	0.355	0.494	0.355	0.400
1.0	5	0.9	0.7	0.603	0.608	0.603	0.514	0.364	0.519	0.364	0.408
1.0	5	1.0	0.7	0.598	0.600	0.598	0.514	0.365	0.519	0.365	0.408

(SIARCT-CFP, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
b	*a*	*m*	*c*	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
1.0	5	0.7	0.9	0.637	0.625	0.637	0.528	0.355	0.494	0.355	0.400
1.0	5	0.9	0.9	0.603	0.608	0.603	0.514	0.364	0.519	0.364	0.408
1.0	5	1.0	0.9	0.598	0.600	0.598	0.514	0.365	0.519	0.365	0.408
0.5	6	0.7	0.3	0.831	0.778	0.831	0.678	0.635	0.650	0.635	0.553
0.5	6	0.9	0.3	0.824	0.786	0.824	0.678	0.640	0.669	0.640	0.556
0.5	6	1.0	0.3	0.823	0.786	0.823	0.681	0.640	0.667	0.640	0.556
0.5	6	0.7	0.5	0.831	0.778	0.831	0.678	0.635	0.650	0.635	0.553
0.5	6	0.9	0.5	0.824	0.786	0.824	0.678	0.640	0.669	0.640	0.556
0.5	6	1.0	0.5	0.823	0.786	0.823	0.681	0.640	0.667	0.640	0.553
0.5	6	0.7	0.7	0.831	0.778	0.831	0.678	0.635	0.650	0.635	0.553
0.5	6	0.9	0.7	0.824	0.786	0.824	0.678	0.640	0.669	0.640	0.556
0.5	6	1.0	0.7	0.823	0.786	0.823	0.681	0.640	0.667	0.640	0.556
0.5	6	0.7	0.9	0.831	0.778	0.831	0.678	0.635	0.650	0.635	0.553
0.5	6	0.9	0.9	0.824	0.786	0.824	0.678	0.640	0.669	0.640	0.556
0.5	6	1.0	0.9	0.823	0.786	0.823	0.681	0.640	0.667	0.640	0.556
0.7	6	0.7	0.3	0.644	0.614	0.644	0.567	0.411	0.528	0.411	0.406
0.7	6	0.9	0.3	0.632	0.592	0.632	0.564	0.418	0.542	0.418	0.417
0.7	6	1.0	0.3	0.631	0.594	0.631	0.564	0.418	0.542	0.418	0.419
0.7	6	0.7	0.5	0.644	0.614	0.644	0.567	0.411	0.528	0.411	0.408
0.7	6	0.9	0.5	0.632	0.592	0.632	0.564	0.418	0.542	0.418	0.419
0.7	6	1.0	0.5	0.631	0.594	0.631	0.561	0.418	0.542	0.418	0.419
0.7	6	0.7	0.7	0.644	0.614	0.644	0.567	0.411	0.528	0.411	0.406
0.7	6	0.9	0.7	0.632	0.592	0.632	0.567	0.418	0.542	0.418	0.419
0.7	6	1.0	0.7	0.631	0.594	0.631	0.564	0.418	0.542	0.418	0.417
0.7	6	0.7	0.9	0.644	0.614	0.644	0.567	0.411	0.528	0.411	0.406
0.7	6	0.9	0.9	0.632	0.592	0.632	0.567	0.418	0.542	0.418	0.419
0.7	6	1.0	0.9	0.631	0.594	0.631	0.561	0.418	0.542	0.418	0.419
0.9	6	0.7	0.3	0.450	0.500	0.450	0.453	0.245	0.475	0.245	0.383
0.9	6	0.9	0.3	0.439	0.489	0.439	0.461	0.249	0.481	0.249	0.394
0.9	6	1.0	0.3	0.437	0.489	0.437	0.456	0.250	0.481	0.250	0.392
0.9	6	0.7	0.5	0.450	0.500	0.450	0.453	0.245	0.475	0.245	0.383
0.9	6	0.9	0.5	0.439	0.489	0.439	0.461	0.249	0.481	0.249	0.394
0.9	6	1.0	0.5	0.437	0.489	0.437	0.456	0.250	0.481	0.250	0.392
0.9	6	0.7	0.7	0.450	0.500	0.450	0.453	0.245	0.475	0.245	0.383
0.9	6	0.9	0.7	0.439	0.489	0.439	0.461	0.249	0.481	0.249	0.394
0.9	6	1.0	0.7	0.437	0.489	0.437	0.456	0.250	0.481	0.250	0.394
0.9	6	0.7	0.9	0.450	0.500	0.450	0.453	0.245	0.475	0.245	0.383
0.9	6	0.9	0.9	0.439	0.489	0.439	0.461	0.249	0.481	0.249	0.394
0.9	6	1.0	0.9	0.437	0.489	0.437	0.456	0.250	0.481	0.250	0.392
1.0	6	0.7	0.3	0.448	0.506	0.448	0.447	0.242	0.475	0.242	0.378
1.0	6	0.9	0.3	0.436	0.492	0.436	0.461	0.246	0.475	0.246	0.392
1.0	6	1.0	0.3	0.435	0.489	0.435	0.450	0.247	0.475	0.247	0.392
1.0	6	0.7	0.5	0.448	0.506	0.448	0.450	0.242	0.475	0.242	0.378
1.0	6	0.9	0.5	0.436	0.492	0.436	0.461	0.246	0.475	0.246	0.392
1.0	6	1.0	0.5	0.435	0.489	0.435	0.450	0.247	0.475	0.247	0.389
1.0	6	0.7	0.7	0.448	0.506	0.448	0.450	0.242	0.475	0.242	0.378
1.0	6	0.9	0.7	0.436	0.492	0.436	0.461	0.246	0.475	0.246	0.392
1.0	6	1.0	0.7	0.435	0.489	0.435	0.450	0.247	0.475	0.247	0.392
1.0	6	0.7	0.9	0.448	0.506	0.448	0.450	0.242	0.475	0.242	0.378
1.0	6	0.9	0.9	0.436	0.492	0.436	0.461	0.246	0.475	0.246	0.392
1.0	6	1.0	0.9	0.435	0.489	0.435	0.450	0.247	0.475	0.247	0.392
0.5	7	0.7	0.3	0.710	0.636	0.710	0.603	0.520	0.553	0.520	0.450
0.5	7	0.9	0.3	0.707	0.650	0.707	0.600	0.523	0.561	0.523	0.464
0.5	7	1.0	0.3	0.707	0.650	0.707	0.600	0.524	0.561	0.524	0.464
0.5	7	0.7	0.5	0.710	0.636	0.710	0.600	0.520	0.553	0.520	0.453
0.5	7	0.9	0.5	0.707	0.650	0.707	0.597	0.523	0.561	0.523	0.464
0.5	7	1.0	0.5	0.707	0.650	0.707	0.600	0.524	0.561	0.524	0.464
0.5	7	0.7	0.7	0.710	0.636	0.710	0.603	0.520	0.553	0.520	0.453
0.5	7	0.9	0.7	0.707	0.650	0.707	0.600	0.523	0.561	0.523	0.464
0.5	7	1.0	0.7	0.707	0.650	0.707	0.600	0.524	0.561	0.524	0.464

(SIARCT-CFP, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
b	a	m	c	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.5	7	0.7	0.9	0.710	0.636	0.710	0.600	0.520	0.553	0.520	0.450
0.5	7	0.9	0.9	0.707	0.650	0.707	0.600	0.523	0.561	0.523	0.464
0.5	7	1.0	0.9	0.707	0.650	0.707	0.603	0.524	0.561	0.524	0.461
0.7	7	0.7	0.3	0.522	0.533	0.522	0.489	0.321	0.450	0.321	0.431
0.7	7	0.9	0.3	0.516	0.539	0.516	0.494	0.326	0.453	0.326	0.433
0.7	7	1.0	0.3	0.515	0.542	0.515	0.494	0.326	0.453	0.326	0.433
0.7	7	0.7	0.5	0.522	0.533	0.522	0.489	0.321	0.450	0.321	0.431
0.7	7	0.9	0.5	0.516	0.539	0.516	0.494	0.326	0.453	0.326	0.431
0.7	7	1.0	0.5	0.515	0.542	0.515	0.494	0.326	0.453	0.326	0.431
0.7	7	0.7	0.7	0.522	0.533	0.522	0.489	0.321	0.450	0.321	0.431
0.7	7	0.9	0.7	0.516	0.539	0.516	0.494	0.326	0.453	0.326	0.431
0.7	7	1.0	0.7	0.515	0.542	0.515	0.494	0.326	0.453	0.326	0.431
0.7	7	0.7	0.9	0.522	0.533	0.522	0.489	0.321	0.450	0.321	0.431
0.7	7	0.9	0.9	0.516	0.539	0.516	0.494	0.326	0.453	0.326	0.431
0.7	7	1.0	0.9	0.515	0.542	0.515	0.494	0.326	0.453	0.326	0.433
0.9	7	0.7	0.3	0.352	0.447	0.352	0.425	0.187	0.414	0.187	0.397
0.9	7	0.9	0.3	0.347	0.447	0.347	0.425	0.188	0.414	0.188	0.403
0.9	7	1.0	0.3	0.346	0.447	0.346	0.428	0.188	0.414	0.188	0.397
0.9	7	0.7	0.5	0.352	0.447	0.352	0.425	0.187	0.414	0.187	0.397
0.9	7	0.9	0.5	0.347	0.447	0.347	0.425	0.188	0.414	0.188	0.403
0.9	7	1.0	0.5	0.346	0.447	0.346	0.428	0.188	0.414	0.188	0.397
0.9	7	0.7	0.7	0.352	0.447	0.352	0.425	0.187	0.414	0.187	0.397
0.9	7	0.9	0.7	0.347	0.447	0.347	0.425	0.188	0.414	0.188	0.403
0.9	7	1.0	0.7	0.346	0.447	0.346	0.428	0.188	0.414	0.188	0.397
0.9	7	0.7	0.9	0.352	0.447	0.352	0.425	0.187	0.414	0.187	0.397
0.9	7	0.9	0.9	0.347	0.447	0.347	0.425	0.188	0.414	0.188	0.403
0.9	7	1.0	0.9	0.346	0.447	0.346	0.428	0.188	0.414	0.188	0.397
1.0	7	0.7	0.3	0.347	0.447	0.347	0.425	0.182	0.403	0.182	0.392
1.0	7	0.9	0.3	0.341	0.450	0.341	0.425	0.183	0.400	0.183	0.394
1.0	7	1.0	0.3	0.341	0.450	0.341	0.425	0.184	0.400	0.184	0.394
1.0	7	0.7	0.5	0.347	0.447	0.347	0.425	0.182	0.403	0.182	0.392
1.0	7	0.9	0.5	0.341	0.450	0.341	0.425	0.183	0.400	0.183	0.394
1.0	7	1.0	0.5	0.341	0.450	0.341	0.425	0.184	0.400	0.184	0.394
1.0	7	0.7	0.7	0.347	0.447	0.347	0.425	0.182	0.403	0.182	0.392
1.0	7	0.9	0.7	0.341	0.450	0.341	0.425	0.183	0.400	0.183	0.394
1.0	7	1.0	0.7	0.341	0.450	0.341	0.425	0.184	0.400	0.184	0.394
1.0	7	0.7	0.9	0.347	0.447	0.347	0.425	0.182	0.403	0.182	0.392
1.0	7	0.9	0.9	0.341	0.450	0.341	0.425	0.183	0.400	0.183	0.394
1.0	7	1.0	0.9	0.341	0.450	0.341	0.425	0.184	0.400	0.184	0.394

Table D.10.: Raw results of SIARCT-CFP using Song Pattern Discovery on MTC-ANN. The parameters are: minimum compactness (b), minimum pattern size (a), inexactness threshold (m) and similarity threshold (c).

D.4.5. PatMinr

Parameters	GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
Min. patt. size	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
3	0.778	0.767	0.778	0.667	0.435	0.644	0.435	0.528
4	0.519	0.586	0.519	0.522	0.272	0.522	0.272	0.428
5	0.369	0.469	0.369	0.425	0.190	0.447	0.190	0.389
6	0.311	0.431	0.311	0.403	0.157	0.425	0.157	0.386
7	0.275	0.403	0.275	0.381	0.138	0.389	0.138	0.372

Table D.11.: Raw results of PatMinr using Song Pattern Discovery on MTC-ANN.

D.5. Class Pattern Discovery on MTC-ANN

D.5.1. SIACTTEC

Parameters		GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
Min. patt. size	Min. compactness	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.1	3	0.922	0.897	0.922	0.858	0.921	0.894	0.921	0.856
0.1	4	0.915	0.886	0.915	0.858	0.915	0.883	0.915	0.856
0.1	5	0.885	0.878	0.885	0.814	0.884	0.878	0.884	0.817
0.1	6	0.801	0.786	0.801	0.717	0.799	0.781	0.799	0.717
0.1	7	0.658	0.647	0.658	0.492	0.656	0.636	0.656	0.492
0.3	3	0.921	0.897	0.921	0.850	0.918	0.894	0.918	0.850
0.3	4	0.917	0.892	0.917	0.844	0.913	0.892	0.913	0.842
0.3	5	0.907	0.897	0.907	0.844	0.902	0.897	0.902	0.839
0.3	6	0.891	0.881	0.891	0.831	0.885	0.886	0.885	0.831
0.3	7	0.856	0.872	0.856	0.800	0.849	0.864	0.849	0.797
0.5	3	0.918	0.903	0.918	0.856	0.906	0.903	0.906	0.850
0.5	4	0.909	0.894	0.909	0.853	0.895	0.894	0.895	0.850
0.5	5	0.886	0.886	0.886	0.819	0.867	0.889	0.867	0.822
0.5	6	0.833	0.878	0.833	0.786	0.814	0.878	0.814	0.789
0.5	7	0.769	0.844	0.769	0.747	0.752	0.847	0.752	0.753
0.7	3	0.912	0.903	0.912	0.853	0.880	0.903	0.880	0.847
0.7	4	0.876	0.883	0.876	0.836	0.841	0.875	0.841	0.828
0.7	5	0.804	0.864	0.804	0.772	0.774	0.881	0.774	0.778
0.7	6	0.719	0.844	0.719	0.697	0.693	0.850	0.693	0.681
0.7	7	0.672	0.839	0.672	0.697	0.648	0.842	0.648	0.706
0.9	3	0.902	0.897	0.902	0.856	0.841	0.883	0.841	0.836
0.9	4	0.823	0.853	0.823	0.808	0.765	0.861	0.765	0.789
0.9	5	0.688	0.839	0.688	0.722	0.644	0.844	0.644	0.731
0.9	6	0.592	0.822	0.592	0.678	0.557	0.803	0.557	0.692
0.9	7	0.524	0.783	0.524	0.594	0.496	0.769	0.496	0.603
1.0	3	0.902	0.900	0.902	0.856	0.838	0.892	0.838	0.836
1.0	4	0.821	0.856	0.821	0.808	0.760	0.858	0.760	0.792
1.0	5	0.684	0.847	0.684	0.717	0.636	0.844	0.636	0.728
1.0	6	0.585	0.808	0.585	0.681	0.546	0.808	0.546	0.686
1.0	7	0.513	0.783	0.513	0.586	0.482	0.781	0.482	0.589

Table D.12.: Raw results of SIACTTEC using Class Pattern Discovery on MTC-ANN.

D.5.2. COSIACTTEC

Parameters		GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
Min. patt. size	Min. compactness	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.1	3	0.578	0.683	0.578	0.608	0.450	0.614	0.450	0.533
0.1	4	0.476	0.714	0.476	0.567	0.379	0.661	0.379	0.550
0.1	5	0.399	0.647	0.399	0.547	0.330	0.661	0.330	0.572
0.1	6	0.314	0.617	0.314	0.506	0.267	0.617	0.267	0.539
0.1	7	0.242	0.531	0.242	0.433	0.211	0.531	0.211	0.442
0.3	3	0.689	0.733	0.689	0.728	0.550	0.697	0.550	0.664
0.3	4	0.603	0.778	0.603	0.694	0.489	0.756	0.489	0.647
0.3	5	0.524	0.772	0.524	0.711	0.432	0.767	0.432	0.736
0.3	6	0.471	0.786	0.471	0.717	0.397	0.756	0.397	0.697
0.3	7	0.444	0.736	0.444	0.647	0.373	0.750	0.373	0.647
0.5	3	0.714	0.789	0.714	0.753	0.562	0.750	0.562	0.683
0.5	4	0.637	0.803	0.637	0.739	0.512	0.744	0.512	0.678
0.5	5	0.568	0.803	0.568	0.786	0.470	0.778	0.470	0.756
0.5	6	0.510	0.819	0.510	0.692	0.426	0.808	0.426	0.700
0.5	7	0.478	0.789	0.478	0.714	0.404	0.792	0.404	0.728

(COSIACTTEC, continued from previous page)

Parameters		GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
Min. patt. size	Min. compactness	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.7	3	0.718	0.803	0.718	0.758	0.557	0.706	0.557	0.700
0.7	4	0.619	0.800	0.619	0.744	0.492	0.753	0.492	0.686
0.7	5	0.555	0.794	0.555	0.719	0.457	0.775	0.457	0.689
0.7	6	0.497	0.772	0.497	0.683	0.418	0.764	0.418	0.692
0.7	7	0.467	0.786	0.467	0.736	0.395	0.783	0.395	0.697
0.9	3	0.710	0.786	0.710	0.758	0.539	0.753	0.539	0.683
0.9	4	0.606	0.767	0.606	0.689	0.474	0.761	0.474	0.667
0.9	5	0.517	0.742	0.517	0.667	0.418	0.733	0.418	0.625
0.9	6	0.442	0.708	0.442	0.622	0.366	0.711	0.366	0.600
0.9	7	0.391	0.708	0.391	0.683	0.326	0.719	0.326	0.669
1.0	3	0.704	0.803	0.704	0.739	0.537	0.742	0.537	0.636
1.0	4	0.606	0.775	0.606	0.681	0.476	0.761	0.476	0.656
1.0	5	0.516	0.767	0.516	0.617	0.420	0.736	0.420	0.586
1.0	6	0.444	0.733	0.444	0.656	0.369	0.717	0.369	0.628
1.0	7	0.389	0.694	0.389	0.664	0.324	0.714	0.324	0.661

Table D.13.: Raw results of COSIACTTEC using Class Pattern Discovery on MTC-ANN.

D.5.3. MotivesExtractor

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
2	0.7	3	0	0.114	0.322	0.114	0.297	0.112	0.322	0.112	0.292
3	0.7	3	0	0.330	0.514	0.330	0.369	0.323	0.500	0.323	0.364
4	0.7	3	0	0.166	0.350	0.166	0.222	0.162	0.336	0.162	0.219
2	0.7	3	2	0.111	0.319	0.111	0.289	0.107	0.317	0.107	0.283
3	0.7	3	2	0.327	0.503	0.327	0.333	0.310	0.475	0.310	0.325
4	0.7	3	2	0.166	0.336	0.166	0.211	0.156	0.336	0.156	0.217
2	0.7	3	4	0.111	0.319	0.111	0.286	0.106	0.311	0.106	0.283
3	0.7	3	4	0.323	0.511	0.323	0.317	0.297	0.486	0.297	0.322
4	0.7	3	4	0.164	0.342	0.164	0.228	0.150	0.333	0.150	0.231
2	0.7	3	6	0.110	0.314	0.110	0.286	0.103	0.306	0.103	0.283
3	0.7	3	6	0.318	0.489	0.318	0.308	0.285	0.461	0.285	0.289
4	0.7	3	6	0.162	0.339	0.162	0.228	0.145	0.325	0.145	0.242
2	0.7	3	8	0.108	0.317	0.108	0.289	0.097	0.311	0.097	0.281
3	0.7	3	8	0.314	0.492	0.314	0.303	0.272	0.461	0.272	0.283
4	0.7	3	8	0.163	0.336	0.163	0.231	0.145	0.317	0.145	0.231
2	0.7	4	0	0.211	0.389	0.211	0.361	0.208	0.389	0.208	0.361
3	0.7	4	0	0.170	0.314	0.170	0.258	0.167	0.311	0.167	0.258
4	0.7	4	0	0.089	0.194	0.089	0.153	0.088	0.192	0.088	0.153
2	0.7	4	2	0.209	0.392	0.209	0.361	0.204	0.397	0.204	0.358
3	0.7	4	2	0.173	0.317	0.173	0.231	0.162	0.314	0.162	0.225
4	0.7	4	2	0.091	0.203	0.091	0.150	0.082	0.197	0.082	0.144
2	0.7	4	4	0.209	0.392	0.209	0.361	0.203	0.403	0.203	0.364
3	0.7	4	4	0.173	0.317	0.173	0.228	0.159	0.314	0.159	0.228
4	0.7	4	4	0.092	0.197	0.092	0.150	0.081	0.203	0.081	0.153
2	0.7	4	6	0.204	0.386	0.204	0.358	0.195	0.394	0.195	0.353
3	0.7	4	6	0.171	0.317	0.171	0.219	0.155	0.311	0.155	0.217
4	0.7	4	6	0.090	0.200	0.090	0.150	0.078	0.206	0.078	0.156
2	0.7	4	8	0.198	0.372	0.198	0.361	0.186	0.375	0.186	0.356
3	0.7	4	8	0.169	0.311	0.169	0.225	0.151	0.308	0.151	0.217
4	0.7	4	8	0.091	0.200	0.091	0.147	0.078	0.208	0.078	0.156
2	0.7	5	0	0.221	0.414	0.221	0.378	0.219	0.414	0.219	0.375
3	0.7	5	0	0.143	0.286	0.143	0.228	0.142	0.286	0.142	0.225
4	0.7	5	0	0.087	0.189	0.087	0.167	0.086	0.192	0.086	0.164
2	0.7	5	2	0.219	0.403	0.219	0.378	0.215	0.408	0.215	0.375

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
3	0.7	5	2	0.145	0.278	0.145	0.222	0.139	0.286	0.139	0.222
4	0.7	5	2	0.089	0.189	0.089	0.169	0.083	0.186	0.083	0.164
2	0.7	5	4	0.217	0.403	0.217	0.367	0.211	0.406	0.211	0.367
3	0.7	5	4	0.145	0.275	0.145	0.219	0.137	0.278	0.137	0.217
4	0.7	5	4	0.089	0.189	0.089	0.175	0.082	0.181	0.082	0.169
2	0.7	5	6	0.211	0.383	0.211	0.358	0.204	0.389	0.204	0.358
3	0.7	5	6	0.142	0.275	0.142	0.217	0.132	0.275	0.132	0.217
4	0.7	5	6	0.088	0.189	0.088	0.175	0.081	0.183	0.081	0.169
2	0.7	5	8	0.210	0.372	0.210	0.364	0.201	0.378	0.201	0.358
3	0.7	5	8	0.141	0.275	0.141	0.214	0.131	0.278	0.131	0.214
4	0.7	5	8	0.087	0.189	0.087	0.178	0.079	0.181	0.079	0.172
2	0.7	6	0	0.170	0.353	0.170	0.344	0.170	0.353	0.170	0.344
3	0.7	6	0	0.226	0.389	0.226	0.358	0.225	0.389	0.225	0.358
4	0.7	6	0	0.105	0.211	0.105	0.178	0.105	0.208	0.105	0.178
2	0.7	6	2	0.168	0.353	0.168	0.347	0.166	0.353	0.166	0.344
3	0.7	6	2	0.226	0.389	0.226	0.347	0.221	0.386	0.221	0.353
4	0.7	6	2	0.105	0.211	0.105	0.169	0.102	0.203	0.102	0.164
2	0.7	6	4	0.165	0.350	0.165	0.344	0.162	0.350	0.162	0.342
3	0.7	6	4	0.226	0.400	0.226	0.350	0.216	0.397	0.216	0.350
4	0.7	6	4	0.105	0.208	0.105	0.169	0.101	0.203	0.101	0.164
2	0.7	6	6	0.166	0.353	0.166	0.339	0.161	0.356	0.161	0.339
3	0.7	6	6	0.227	0.397	0.227	0.353	0.215	0.386	0.215	0.356
4	0.7	6	6	0.104	0.206	0.104	0.169	0.099	0.197	0.099	0.167
2	0.7	6	8	0.164	0.336	0.164	0.333	0.158	0.344	0.158	0.331
3	0.7	6	8	0.226	0.394	0.226	0.358	0.212	0.392	0.212	0.353
4	0.7	6	8	0.101	0.206	0.101	0.169	0.096	0.194	0.096	0.164
2	0.7	7	0	0.148	0.311	0.148	0.292	0.148	0.311	0.148	0.292
3	0.7	7	0	0.196	0.367	0.196	0.328	0.195	0.364	0.195	0.328
4	0.7	7	0	0.084	0.172	0.084	0.169	0.084	0.172	0.084	0.169
2	0.7	7	2	0.145	0.306	0.145	0.289	0.142	0.306	0.142	0.292
3	0.7	7	2	0.194	0.358	0.194	0.336	0.191	0.350	0.191	0.331
4	0.7	7	2	0.084	0.172	0.084	0.169	0.082	0.178	0.082	0.169
2	0.7	7	4	0.143	0.303	0.143	0.294	0.140	0.306	0.140	0.297
3	0.7	7	4	0.192	0.361	0.192	0.333	0.187	0.353	0.187	0.331
4	0.7	7	4	0.084	0.169	0.084	0.172	0.081	0.178	0.081	0.172
2	0.7	7	6	0.142	0.297	0.142	0.292	0.139	0.303	0.139	0.294
3	0.7	7	6	0.190	0.358	0.190	0.328	0.185	0.350	0.185	0.325
4	0.7	7	6	0.084	0.169	0.084	0.172	0.080	0.175	0.080	0.169
2	0.7	7	8	0.141	0.294	0.141	0.289	0.136	0.306	0.136	0.294
3	0.7	7	8	0.187	0.358	0.187	0.331	0.178	0.347	0.178	0.325
4	0.7	7	8	0.084	0.169	0.084	0.172	0.080	0.175	0.080	0.167
2	0.5	3	0	0.114	0.322	0.114	0.297	0.112	0.322	0.112	0.292
3	0.5	3	0	0.912	0.883	0.912	0.847	0.911	0.883	0.911	0.847
4	0.5	3	0	0.422	0.417	0.422	0.361	0.412	0.422	0.412	0.353
2	0.5	3	2	0.111	0.319	0.111	0.289	0.107	0.317	0.107	0.283
3	0.5	3	2	0.911	0.889	0.911	0.847	0.908	0.883	0.908	0.847
4	0.5	3	2	0.426	0.422	0.426	0.381	0.371	0.414	0.371	0.339
2	0.5	3	4	0.111	0.319	0.111	0.286	0.106	0.311	0.106	0.283
3	0.5	3	4	0.908	0.886	0.908	0.844	0.896	0.864	0.896	0.839
4	0.5	3	4	0.428	0.425	0.428	0.400	0.344	0.419	0.344	0.339
2	0.5	3	6	0.110	0.314	0.110	0.286	0.103	0.306	0.103	0.283
3	0.5	3	6	0.904	0.878	0.904	0.836	0.883	0.869	0.883	0.822
4	0.5	3	6	0.427	0.425	0.427	0.397	0.324	0.389	0.324	0.331
2	0.5	3	8	0.108	0.317	0.108	0.289	0.097	0.311	0.097	0.281
3	0.5	3	8	0.904	0.881	0.904	0.844	0.864	0.850	0.864	0.794
4	0.5	3	8	0.427	0.425	0.427	0.397	0.298	0.383	0.298	0.325
2	0.5	4	0	0.211	0.389	0.211	0.361	0.208	0.389	0.208	0.361
3	0.5	4	0	0.891	0.872	0.891	0.825	0.891	0.872	0.891	0.828
4	0.5	4	0	0.410	0.439	0.410	0.386	0.403	0.428	0.403	0.386
2	0.5	4	2	0.209	0.392	0.209	0.361	0.204	0.397	0.204	0.358

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
3	0.5	4	2	0.892	0.872	0.892	0.831	0.888	0.869	0.888	0.819
4	0.5	4	2	0.412	0.444	0.412	0.389	0.364	0.408	0.364	0.353
2	0.5	4	4	0.209	0.392	0.209	0.364	0.203	0.403	0.203	0.367
3	0.5	4	4	0.887	0.881	0.887	0.831	0.875	0.864	0.875	0.831
4	0.5	4	4	0.411	0.444	0.411	0.389	0.341	0.389	0.341	0.339
2	0.5	4	6	0.204	0.386	0.204	0.358	0.195	0.394	0.195	0.353
3	0.5	4	6	0.884	0.869	0.884	0.819	0.865	0.864	0.865	0.814
4	0.5	4	6	0.410	0.442	0.410	0.392	0.326	0.378	0.326	0.286
2	0.5	4	8	0.198	0.372	0.198	0.361	0.186	0.375	0.186	0.356
3	0.5	4	8	0.881	0.861	0.881	0.819	0.848	0.844	0.848	0.797
4	0.5	4	8	0.407	0.442	0.407	0.389	0.312	0.383	0.312	0.264
2	0.5	5	0	0.221	0.414	0.221	0.378	0.219	0.414	0.219	0.375
3	0.5	5	0	0.858	0.850	0.858	0.803	0.858	0.850	0.858	0.797
4	0.5	5	0	0.386	0.419	0.386	0.369	0.380	0.417	0.380	0.361
2	0.5	5	2	0.219	0.403	0.219	0.378	0.215	0.408	0.215	0.375
3	0.5	5	2	0.858	0.875	0.858	0.817	0.852	0.864	0.852	0.806
4	0.5	5	2	0.388	0.422	0.388	0.369	0.357	0.411	0.357	0.347
2	0.5	5	4	0.217	0.403	0.217	0.367	0.211	0.406	0.211	0.367
3	0.5	5	4	0.855	0.864	0.855	0.811	0.841	0.853	0.841	0.797
4	0.5	5	4	0.386	0.422	0.386	0.361	0.341	0.417	0.341	0.336
2	0.5	5	6	0.211	0.383	0.211	0.358	0.204	0.389	0.204	0.358
3	0.5	5	6	0.850	0.861	0.850	0.817	0.829	0.867	0.829	0.781
4	0.5	5	6	0.382	0.419	0.382	0.367	0.332	0.414	0.332	0.297
2	0.5	5	8	0.210	0.372	0.210	0.364	0.201	0.378	0.201	0.358
3	0.5	5	8	0.848	0.853	0.848	0.797	0.818	0.839	0.818	0.769
4	0.5	5	8	0.381	0.422	0.381	0.369	0.320	0.417	0.320	0.269
2	0.5	6	0	0.170	0.353	0.170	0.344	0.170	0.353	0.170	0.344
3	0.5	6	0	0.817	0.825	0.817	0.764	0.817	0.828	0.817	0.764
4	0.5	6	0	0.362	0.394	0.362	0.353	0.357	0.389	0.357	0.347
2	0.5	6	2	0.168	0.353	0.168	0.347	0.166	0.353	0.166	0.344
3	0.5	6	2	0.822	0.858	0.822	0.775	0.816	0.858	0.816	0.778
4	0.5	6	2	0.363	0.386	0.363	0.350	0.344	0.383	0.344	0.322
2	0.5	6	4	0.165	0.350	0.165	0.342	0.162	0.350	0.162	0.342
3	0.5	6	4	0.813	0.839	0.813	0.756	0.802	0.842	0.802	0.756
4	0.5	6	4	0.361	0.386	0.361	0.350	0.331	0.364	0.331	0.325
2	0.5	6	6	0.166	0.353	0.166	0.342	0.161	0.356	0.161	0.339
3	0.5	6	6	0.807	0.842	0.807	0.747	0.794	0.833	0.794	0.747
4	0.5	6	6	0.355	0.389	0.355	0.350	0.319	0.375	0.319	0.314
2	0.5	6	8	0.164	0.336	0.164	0.333	0.158	0.344	0.158	0.331
3	0.5	6	8	0.805	0.836	0.805	0.731	0.784	0.822	0.784	0.744
4	0.5	6	8	0.356	0.381	0.356	0.347	0.315	0.361	0.315	0.314
2	0.5	7	0	0.148	0.311	0.148	0.292	0.148	0.311	0.148	0.292
3	0.5	7	0	0.760	0.817	0.760	0.681	0.759	0.814	0.759	0.681
4	0.5	7	0	0.303	0.328	0.303	0.292	0.302	0.333	0.302	0.289
2	0.5	7	2	0.145	0.306	0.145	0.289	0.142	0.306	0.142	0.292
3	0.5	7	2	0.760	0.819	0.760	0.681	0.756	0.814	0.756	0.683
4	0.5	7	2	0.310	0.336	0.310	0.311	0.299	0.344	0.299	0.292
2	0.5	7	4	0.143	0.303	0.143	0.294	0.140	0.306	0.140	0.297
3	0.5	7	4	0.759	0.817	0.759	0.678	0.751	0.808	0.751	0.675
4	0.5	7	4	0.309	0.333	0.309	0.308	0.293	0.336	0.293	0.289
2	0.5	7	6	0.142	0.297	0.142	0.292	0.139	0.303	0.139	0.294
3	0.5	7	6	0.752	0.814	0.752	0.672	0.739	0.803	0.739	0.672
4	0.5	7	6	0.307	0.328	0.307	0.308	0.285	0.333	0.285	0.297
2	0.5	7	8	0.141	0.294	0.141	0.289	0.136	0.306	0.136	0.294
3	0.5	7	8	0.747	0.819	0.747	0.642	0.731	0.794	0.731	0.644
4	0.5	7	8	0.303	0.322	0.303	0.292	0.275	0.325	0.275	0.261
2	0.3	3	0	0.919	0.897	0.919	0.853	0.919	0.897	0.919	0.853
3	0.3	3	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	3	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	3	2	0.918	0.897	0.918	0.853	0.918	0.897	0.918	0.853

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
3	0.3	3	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	3	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	3	4	0.917	0.897	0.917	0.853	0.915	0.889	0.915	0.847
3	0.3	3	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	3	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	3	6	0.916	0.897	0.916	0.842	0.905	0.889	0.905	0.842
3	0.3	3	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	3	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	3	8	0.916	0.900	0.916	0.844	0.899	0.886	0.899	0.828
3	0.3	3	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	3	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	4	0	0.919	0.894	0.919	0.853	0.919	0.894	0.919	0.853
3	0.3	4	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	4	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	4	2	0.919	0.894	0.919	0.856	0.918	0.894	0.918	0.856
3	0.3	4	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	4	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	4	4	0.916	0.900	0.916	0.850	0.912	0.892	0.912	0.847
3	0.3	4	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	4	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	4	6	0.916	0.900	0.916	0.856	0.907	0.892	0.907	0.847
3	0.3	4	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	4	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	4	8	0.915	0.900	0.915	0.844	0.899	0.883	0.899	0.847
3	0.3	4	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	4	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	5	0	0.918	0.897	0.918	0.853	0.918	0.897	0.918	0.853
3	0.3	5	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	5	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	5	2	0.917	0.903	0.917	0.853	0.916	0.903	0.916	0.850
3	0.3	5	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	5	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	5	4	0.914	0.897	0.914	0.847	0.910	0.892	0.910	0.842
3	0.3	5	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	5	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	5	6	0.912	0.892	0.912	0.833	0.900	0.878	0.900	0.828
3	0.3	5	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	5	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	5	8	0.909	0.886	0.909	0.831	0.893	0.864	0.893	0.814
3	0.3	5	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	5	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	6	0	0.914	0.892	0.914	0.847	0.914	0.892	0.914	0.850
3	0.3	6	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	6	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	6	2	0.913	0.894	0.913	0.850	0.912	0.892	0.912	0.850
3	0.3	6	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	6	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	6	4	0.908	0.886	0.908	0.847	0.904	0.883	0.904	0.839
3	0.3	6	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	6	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	6	6	0.906	0.889	0.906	0.853	0.897	0.878	0.897	0.831
3	0.3	6	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	6	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	6	8	0.902	0.881	0.902	0.839	0.884	0.881	0.884	0.817
3	0.3	6	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	6	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	7	0	0.907	0.889	0.907	0.833	0.907	0.886	0.907	0.836
3	0.3	7	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	7	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	7	2	0.907	0.889	0.907	0.825	0.905	0.889	0.905	0.822

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
3	0.3	7	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	7	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	7	4	0.902	0.892	0.902	0.831	0.897	0.889	0.897	0.822
3	0.3	7	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	7	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	7	6	0.899	0.881	0.899	0.822	0.889	0.867	0.889	0.822
3	0.3	7	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	7	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	7	8	0.897	0.883	0.897	0.822	0.880	0.875	0.880	0.819
3	0.3	7	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	7	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	3	0	0.011	0.017	0.011	0.017	0.011	0.017	0.011	0.017
3	0.1	3	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	3	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	3	2	0.011	0.017	0.011	0.017	0.011	0.017	0.011	0.017
3	0.1	3	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	3	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	3	4	0.011	0.017	0.011	0.017	0.011	0.017	0.011	0.017
3	0.1	3	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	3	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	3	6	0.011	0.017	0.011	0.017	0.011	0.017	0.011	0.017
3	0.1	3	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	3	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	3	8	0.011	0.017	0.011	0.017	0.011	0.017	0.011	0.017
3	0.1	3	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	3	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	4	0	0.014	0.022	0.014	0.022	0.014	0.022	0.014	0.022
3	0.1	4	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	4	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	4	2	0.014	0.022	0.014	0.022	0.014	0.022	0.014	0.022
3	0.1	4	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	4	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	4	4	0.014	0.022	0.014	0.022	0.014	0.022	0.014	0.022
3	0.1	4	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	4	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	4	6	0.014	0.022	0.014	0.022	0.014	0.022	0.014	0.022
3	0.1	4	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	4	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	4	8	0.014	0.022	0.014	0.022	0.014	0.022	0.014	0.022
3	0.1	4	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	4	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	5	0	0.015	0.028	0.015	0.028	0.015	0.028	0.015	0.028
3	0.1	5	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	5	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	5	2	0.015	0.028	0.015	0.028	0.015	0.028	0.015	0.028
3	0.1	5	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	5	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	5	4	0.015	0.028	0.015	0.028	0.015	0.028	0.015	0.028
3	0.1	5	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	5	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	5	6	0.015	0.028	0.015	0.028	0.015	0.028	0.015	0.028
3	0.1	5	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	5	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	5	8	0.014	0.028	0.014	0.028	0.014	0.028	0.014	0.028
3	0.1	5	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	5	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	6	0	0.019	0.028	0.019	0.025	0.019	0.028	0.019	0.025
3	0.1	6	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	6	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	6	2	0.019	0.028	0.019	0.025	0.019	0.028	0.019	0.025

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
3	0.1	6	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	6	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	6	4	0.019	0.028	0.019	0.025	0.019	0.028	0.019	0.025
3	0.1	6	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	6	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	6	6	0.019	0.025	0.019	0.022	0.019	0.025	0.019	0.022
3	0.1	6	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	6	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	6	8	0.019	0.025	0.019	0.022	0.019	0.025	0.019	0.022
3	0.1	6	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	6	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	7	0	0.021	0.028	0.021	0.022	0.021	0.028	0.021	0.022
3	0.1	7	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	7	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	7	2	0.021	0.028	0.021	0.022	0.021	0.028	0.021	0.022
3	0.1	7	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	7	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	7	4	0.021	0.028	0.021	0.022	0.021	0.028	0.021	0.022
3	0.1	7	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	7	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	7	6	0.021	0.028	0.021	0.022	0.021	0.028	0.021	0.022
3	0.1	7	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	7	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	7	8	0.020	0.031	0.020	0.022	0.020	0.031	0.020	0.022
3	0.1	7	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	7	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table D.14.: Raw results of MotivesExtractor using Class Pattern Discovery on MTC-ANN. The parameters are the path tracing tolerance (ρ), the scoring threshold (θ), the minimum pattern length (ν) and the tolerance parameter for the pattern clustering (Θ).

D.5.4. MGDP

Parameters					GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
$p(P\|\ominus)$	$I(P)$	n	Min. patt.	Enforce	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.9	10	3	5	true	0.501	0.808	0.501	0.625	0.249	0.683	0.249	0.517
0.9	10	3	10	true	0.590	0.819	0.590	0.647	0.302	0.689	0.302	0.494
0.9	20	3	5	true	0.489	0.803	0.489	0.625	0.242	0.686	0.242	0.536
0.9	20	3	10	true	0.589	0.817	0.589	0.639	0.301	0.683	0.301	0.489
0.9	26	3	5	true	0.487	0.806	0.487	0.619	0.242	0.686	0.242	0.542
0.9	26	3	10	true	0.589	0.817	0.589	0.639	0.301	0.683	0.301	0.489
0.9	40	3	5	true	0.487	0.806	0.487	0.619	0.242	0.686	0.242	0.542
0.9	40	3	10	true	0.589	0.817	0.589	0.639	0.301	0.683	0.301	0.489
0.7	10	3	5	true	0.563	0.836	0.563	0.717	0.414	0.819	0.414	0.692
0.7	10	3	10	true	0.635	0.850	0.635	0.747	0.465	0.822	0.465	0.722
0.7	20	3	5	true	0.451	0.831	0.451	0.703	0.326	0.797	0.326	0.664
0.7	20	3	10	true	0.533	0.833	0.533	0.736	0.385	0.803	0.385	0.689
0.7	26	3	5	true	0.427	0.800	0.427	0.681	0.309	0.781	0.309	0.644
0.7	26	3	10	true	0.512	0.825	0.512	0.700	0.370	0.794	0.370	0.664
0.7	40	3	5	true	0.412	0.808	0.412	0.675	0.300	0.783	0.300	0.650
0.7	40	3	10	true	0.501	0.825	0.501	0.694	0.365	0.806	0.365	0.672
0.5	10	3	5	true	0.700	0.894	0.700	0.661	0.591	0.900	0.591	0.678
0.5	10	3	10	true	0.707	0.897	0.707	0.708	0.598	0.906	0.598	0.697
0.5	20	3	5	true	0.555	0.822	0.555	0.633	0.468	0.819	0.468	0.642
0.5	20	3	10	true	0.584	0.842	0.584	0.644	0.494	0.850	0.494	0.656
0.5	26	3	5	true	0.516	0.825	0.516	0.633	0.434	0.819	0.434	0.633

(MGDP, continued from previous page)

Parameters					GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
$p(P\|\ominus)$	$I(P)$	n	**Min. patt.**	**Enforce**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
0.5	26	3	10	true	0.555	0.839	0.555	0.636	0.464	0.853	0.464	0.633
0.5	40	3	5	true	0.470	0.803	0.470	0.636	0.397	0.803	0.397	0.628
0.5	40	3	10	true	0.512	0.825	0.512	0.625	0.430	0.839	0.430	0.617
0.9	10	4	5	true	0.330	0.675	0.330	0.467	0.174	0.644	0.174	0.442
0.9	10	4	10	true	0.400	0.694	0.400	0.497	0.210	0.633	0.210	0.439
0.9	20	4	5	true	0.318	0.675	0.318	0.444	0.168	0.639	0.168	0.436
0.9	20	4	10	true	0.399	0.692	0.399	0.458	0.209	0.633	0.209	0.431
0.9	26	4	5	true	0.316	0.675	0.316	0.478	0.167	0.642	0.167	0.444
0.9	26	4	10	true	0.399	0.692	0.399	0.458	0.209	0.633	0.209	0.431
0.9	40	4	5	true	0.316	0.675	0.316	0.478	0.167	0.642	0.167	0.444
0.9	40	4	10	true	0.399	0.692	0.399	0.458	0.209	0.633	0.209	0.431
0.7	10	4	5	true	0.545	0.836	0.545	0.622	0.402	0.836	0.402	0.622
0.7	10	4	10	true	0.632	0.864	0.632	0.647	0.461	0.836	0.461	0.700
0.7	20	4	5	true	0.433	0.825	0.433	0.617	0.315	0.808	0.315	0.597
0.7	20	4	10	true	0.531	0.836	0.531	0.675	0.380	0.803	0.380	0.689
0.7	26	4	5	true	0.410	0.789	0.410	0.614	0.298	0.778	0.298	0.597
0.7	26	4	10	true	0.510	0.825	0.510	0.661	0.366	0.806	0.366	0.683
0.7	40	4	5	true	0.395	0.792	0.395	0.622	0.290	0.781	0.290	0.589
0.7	40	4	10	true	0.501	0.828	0.501	0.658	0.362	0.811	0.362	0.692
0.5	10	4	5	true	0.690	0.878	0.690	0.664	0.584	0.886	0.584	0.683
0.5	10	4	10	true	0.705	0.875	0.705	0.728	0.596	0.889	0.596	0.708
0.5	20	4	5	true	0.555	0.822	0.555	0.628	0.469	0.825	0.469	0.631
0.5	20	4	10	true	0.592	0.839	0.592	0.647	0.499	0.858	0.499	0.633
0.5	26	4	5	true	0.516	0.831	0.516	0.628	0.434	0.825	0.434	0.631
0.5	26	4	10	true	0.562	0.842	0.562	0.644	0.469	0.853	0.469	0.625
0.5	40	4	5	true	0.469	0.808	0.469	0.644	0.396	0.806	0.396	0.631
0.5	40	4	10	true	0.519	0.833	0.519	0.644	0.434	0.839	0.434	0.628
0.9	10	5	5	true	0.230	0.503	0.230	0.353	0.131	0.486	0.131	0.336
0.9	10	5	10	true	0.267	0.494	0.267	0.356	0.152	0.467	0.152	0.325
0.9	20	5	5	true	0.219	0.500	0.219	0.358	0.125	0.483	0.125	0.364
0.9	20	5	10	true	0.266	0.492	0.266	0.356	0.151	0.467	0.151	0.325
0.9	26	5	5	true	0.218	0.503	0.218	0.364	0.124	0.481	0.124	0.367
0.9	26	5	10	true	0.266	0.492	0.266	0.356	0.151	0.467	0.151	0.325
0.9	40	5	5	true	0.218	0.503	0.218	0.364	0.124	0.481	0.124	0.367
0.9	40	5	10	true	0.266	0.492	0.266	0.356	0.151	0.467	0.151	0.325
0.7	10	5	5	true	0.504	0.769	0.504	0.550	0.371	0.803	0.371	0.533
0.7	10	5	10	true	0.534	0.781	0.534	0.536	0.390	0.808	0.390	0.514
0.7	20	5	5	true	0.407	0.767	0.407	0.581	0.293	0.764	0.293	0.542
0.7	20	5	10	true	0.441	0.775	0.441	0.567	0.317	0.772	0.317	0.531
0.7	26	5	5	true	0.382	0.756	0.382	0.550	0.276	0.742	0.276	0.514
0.7	26	5	10	true	0.418	0.772	0.418	0.544	0.301	0.758	0.301	0.506
0.7	40	5	5	true	0.368	0.753	0.368	0.575	0.268	0.747	0.268	0.558
0.7	40	5	10	true	0.410	0.772	0.410	0.539	0.298	0.761	0.298	0.519
0.5	10	5	5	true	0.669	0.864	0.669	0.672	0.564	0.864	0.564	0.675
0.5	10	5	10	true	0.687	0.872	0.687	0.703	0.579	0.875	0.579	0.692
0.5	20	5	5	true	0.551	0.792	0.551	0.608	0.463	0.803	0.463	0.586
0.5	20	5	10	true	0.584	0.822	0.584	0.675	0.489	0.822	0.489	0.644
0.5	26	5	5	true	0.523	0.811	0.523	0.606	0.433	0.822	0.433	0.589
0.5	26	5	10	true	0.557	0.842	0.557	0.681	0.461	0.839	0.461	0.650
0.5	40	5	5	true	0.480	0.792	0.480	0.628	0.398	0.794	0.398	0.625
0.5	40	5	10	true	0.517	0.814	0.517	0.650	0.428	0.817	0.428	0.611
0.9	10	6	5	true	0.162	0.406	0.162	0.297	0.097	0.381	0.097	0.278
0.9	10	6	10	true	0.175	0.400	0.175	0.303	0.105	0.383	0.105	0.281
0.9	20	6	5	true	0.152	0.406	0.152	0.300	0.092	0.383	0.092	0.286
0.9	20	6	10	true	0.174	0.397	0.174	0.311	0.104	0.381	0.104	0.281
0.9	26	6	5	true	0.151	0.406	0.151	0.292	0.091	0.381	0.091	0.289
0.9	26	6	10	true	0.174	0.397	0.174	0.311	0.104	0.381	0.104	0.281
0.9	40	6	5	true	0.151	0.406	0.151	0.292	0.091	0.381	0.091	0.289
0.9	40	6	10	true	0.174	0.397	0.174	0.311	0.104	0.381	0.104	0.281
0.7	10	6	5	true	0.404	0.664	0.404	0.531	0.292	0.644	0.292	0.494

(MGDP, continued from previous page)

Parameters					GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
$p(P\|\ominus)$	$I(P)$	n	**Min. patt.**	**Enforce**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
0.7	10	6	10	true	0.428	0.667	0.428	0.525	0.307	0.628	0.307	0.478
0.7	20	6	5	true	0.322	0.658	0.322	0.556	0.227	0.639	0.227	0.494
0.7	20	6	10	true	0.351	0.661	0.351	0.539	0.247	0.619	0.247	0.475
0.7	26	6	5	true	0.299	0.647	0.299	0.561	0.211	0.631	0.211	0.508
0.7	26	6	10	true	0.329	0.658	0.329	0.536	0.232	0.614	0.232	0.481
0.7	40	6	5	true	0.286	0.644	0.286	0.550	0.202	0.631	0.202	0.514
0.7	40	6	10	true	0.322	0.658	0.322	0.536	0.229	0.619	0.229	0.489
0.5	10	6	5	true	0.613	0.822	0.613	0.644	0.520	0.817	0.520	0.586
0.5	10	6	10	true	0.640	0.814	0.640	0.606	0.539	0.806	0.539	0.575
0.5	20	6	5	true	0.520	0.772	0.520	0.589	0.436	0.772	0.436	0.544
0.5	20	6	10	true	0.565	0.781	0.565	0.594	0.466	0.783	0.466	0.544
0.5	26	6	5	true	0.492	0.775	0.492	0.600	0.409	0.772	0.409	0.575
0.5	26	6	10	true	0.538	0.794	0.538	0.589	0.440	0.781	0.440	0.575
0.5	40	6	5	true	0.455	0.756	0.455	0.619	0.378	0.761	0.378	0.606
0.5	40	6	10	true	0.507	0.767	0.507	0.600	0.416	0.775	0.416	0.592
0.9	10	7	5	true	0.095	0.272	0.095	0.208	0.060	0.256	0.060	0.222
0.9	10	7	10	true	0.100	0.275	0.100	0.219	0.064	0.267	0.064	0.219
0.9	20	7	5	true	0.090	0.269	0.090	0.214	0.058	0.256	0.058	0.225
0.9	20	7	10	true	0.100	0.275	0.100	0.219	0.063	0.269	0.063	0.219
0.9	26	7	5	true	0.088	0.269	0.088	0.211	0.057	0.253	0.057	0.225
0.9	26	7	10	true	0.100	0.275	0.100	0.219	0.063	0.269	0.063	0.219
0.9	40	7	5	true	0.088	0.269	0.088	0.211	0.057	0.253	0.057	0.225
0.9	40	7	10	true	0.100	0.275	0.100	0.219	0.063	0.269	0.063	0.219
0.7	10	7	5	true	0.295	0.531	0.295	0.397	0.222	0.519	0.222	0.433
0.7	10	7	10	true	0.295	0.531	0.295	0.397	0.222	0.519	0.222	0.433
0.7	20	7	5	true	0.228	0.497	0.228	0.428	0.170	0.500	0.170	0.431
0.7	20	7	10	true	0.249	0.517	0.249	0.439	0.183	0.506	0.183	0.442
0.7	26	7	5	true	0.217	0.483	0.217	0.425	0.161	0.500	0.161	0.436
0.7	26	7	10	true	0.240	0.503	0.240	0.428	0.176	0.508	0.176	0.442
0.7	40	7	5	true	0.209	0.489	0.209	0.428	0.155	0.503	0.155	0.436
0.7	40	7	10	true	0.234	0.506	0.234	0.428	0.171	0.508	0.171	0.447
0.5	10	7	5	true	0.542	0.781	0.542	0.583	0.457	0.764	0.457	0.567
0.5	10	7	10	true	0.559	0.792	0.559	0.619	0.467	0.783	0.467	0.600
0.5	20	7	5	true	0.479	0.728	0.479	0.569	0.401	0.736	0.401	0.547
0.5	20	7	10	true	0.495	0.744	0.495	0.583	0.410	0.747	0.410	0.550
0.5	26	7	5	true	0.454	0.733	0.454	0.556	0.376	0.744	0.376	0.544
0.5	26	7	10	true	0.472	0.744	0.472	0.569	0.387	0.747	0.387	0.553
0.5	40	7	5	true	0.420	0.736	0.420	0.589	0.349	0.736	0.349	0.567
0.5	40	7	10	true	0.447	0.728	0.447	0.586	0.366	0.728	0.366	0.556
0.9	10	3	0	false	0.095	0.253	0.095	0.236	0.065	0.236	0.065	0.222
0.9	20	3	0	false	0.040	0.175	0.040	0.167	0.030	0.175	0.030	0.167
0.9	26	3	0	false	0.025	0.128	0.025	0.119	0.017	0.125	0.017	0.117
0.9	40	3	0	false	0.012	0.069	0.012	0.069	0.009	0.069	0.009	0.069
0.7	10	3	0	false	0.377	0.581	0.377	0.506	0.293	0.583	0.293	0.492
0.7	20	3	0	false	0.248	0.531	0.248	0.467	0.191	0.528	0.191	0.458
0.7	26	3	0	false	0.218	0.478	0.218	0.442	0.169	0.472	0.169	0.428
0.7	40	3	0	false	0.191	0.444	0.191	0.403	0.148	0.444	0.148	0.408
0.5	10	3	0	false	0.683	0.878	0.683	0.650	0.579	0.878	0.579	0.656
0.5	20	3	0	false	0.521	0.756	0.521	0.625	0.444	0.753	0.444	0.617
0.5	26	3	0	false	0.475	0.744	0.475	0.572	0.401	0.739	0.401	0.567
0.5	40	3	0	false	0.422	0.711	0.422	0.561	0.358	0.717	0.358	0.561
0.9	10	4	0	false	0.092	0.222	0.092	0.203	0.064	0.206	0.064	0.189
0.9	20	4	0	false	0.040	0.175	0.040	0.167	0.030	0.175	0.030	0.167
0.9	26	4	0	false	0.025	0.128	0.025	0.119	0.017	0.125	0.017	0.117
0.9	40	4	0	false	0.012	0.069	0.012	0.069	0.009	0.069	0.009	0.069
0.7	10	4	0	false	0.371	0.572	0.371	0.492	0.288	0.575	0.288	0.483
0.7	20	4	0	false	0.245	0.531	0.245	0.469	0.189	0.531	0.189	0.461
0.7	26	4	0	false	0.216	0.478	0.216	0.442	0.167	0.472	0.167	0.428
0.7	40	4	0	false	0.190	0.444	0.190	0.403	0.147	0.444	0.147	0.408
0.5	10	4	0	false	0.673	0.864	0.673	0.661	0.572	0.861	0.572	0.678

(MGDP, continued from previous page)

Parameters					GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
$p(P\|\ominus)$	$I(P)$	n	Min. patt.	Enforce	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.5	20	4	0	false	0.521	0.761	0.521	0.614	0.444	0.761	0.444	0.606
0.5	26	4	0	false	0.474	0.753	0.474	0.561	0.401	0.744	0.401	0.567
0.5	40	4	0	false	0.421	0.722	0.421	0.567	0.358	0.719	0.358	0.569
0.9	10	5	0	false	0.089	0.222	0.089	0.203	0.063	0.206	0.063	0.192
0.9	20	5	0	false	0.040	0.172	0.040	0.164	0.029	0.172	0.029	0.164
0.9	26	5	0	false	0.025	0.125	0.025	0.117	0.016	0.122	0.016	0.114
0.9	40	5	0	false	0.012	0.069	0.012	0.069	0.009	0.069	0.009	0.069
0.7	10	5	0	false	0.343	0.531	0.343	0.469	0.266	0.550	0.266	0.467
0.7	20	5	0	false	0.240	0.519	0.240	0.467	0.184	0.519	0.184	0.461
0.7	26	5	0	false	0.208	0.450	0.208	0.411	0.161	0.444	0.161	0.403
0.7	40	5	0	false	0.183	0.414	0.183	0.372	0.142	0.414	0.142	0.381
0.5	10	5	0	false	0.644	0.861	0.644	0.656	0.546	0.853	0.546	0.658
0.5	20	5	0	false	0.498	0.728	0.498	0.561	0.424	0.733	0.424	0.558
0.5	26	5	0	false	0.466	0.750	0.466	0.550	0.394	0.750	0.394	0.558
0.5	40	5	0	false	0.418	0.719	0.418	0.539	0.355	0.717	0.355	0.544
0.9	10	6	0	false	0.074	0.219	0.074	0.192	0.052	0.197	0.052	0.189
0.9	20	6	0	false	0.036	0.147	0.036	0.139	0.027	0.147	0.027	0.142
0.9	26	6	0	false	0.021	0.100	0.021	0.092	0.014	0.097	0.014	0.089
0.9	40	6	0	false	0.012	0.069	0.012	0.069	0.009	0.069	0.009	0.069
0.7	10	6	0	false	0.314	0.483	0.314	0.419	0.242	0.472	0.242	0.431
0.7	20	6	0	false	0.230	0.469	0.230	0.408	0.177	0.461	0.177	0.403
0.7	26	6	0	false	0.201	0.433	0.201	0.394	0.155	0.425	0.155	0.394
0.7	40	6	0	false	0.180	0.414	0.180	0.361	0.138	0.408	0.138	0.369
0.5	10	6	0	false	0.583	0.794	0.583	0.633	0.499	0.792	0.499	0.589
0.5	20	6	0	false	0.459	0.667	0.459	0.544	0.392	0.672	0.392	0.544
0.5	26	6	0	false	0.428	0.669	0.428	0.544	0.362	0.672	0.362	0.556
0.5	40	6	0	false	0.386	0.636	0.386	0.542	0.328	0.633	0.328	0.531
0.9	10	7	0	false	0.050	0.136	0.050	0.122	0.036	0.131	0.036	0.117
0.9	20	7	0	false	0.019	0.069	0.019	0.069	0.013	0.069	0.013	0.069
0.9	26	7	0	false	0.018	0.069	0.018	0.069	0.012	0.069	0.012	0.069
0.9	40	7	0	false	0.012	0.069	0.012	0.069	0.009	0.069	0.009	0.069
0.7	10	7	0	false	0.272	0.453	0.272	0.381	0.207	0.436	0.207	0.386
0.7	20	7	0	false	0.205	0.419	0.205	0.375	0.155	0.419	0.155	0.369
0.7	26	7	0	false	0.184	0.408	0.184	0.383	0.139	0.417	0.139	0.386
0.7	40	7	0	false	0.166	0.408	0.166	0.378	0.122	0.400	0.122	0.378
0.5	10	7	0	false	0.509	0.719	0.509	0.558	0.431	0.700	0.431	0.533
0.5	20	7	0	false	0.419	0.575	0.419	0.486	0.359	0.572	0.359	0.475
0.5	26	7	0	false	0.393	0.578	0.393	0.500	0.332	0.581	0.332	0.506
0.5	40	7	0	false	0.358	0.581	0.358	0.494	0.305	0.567	0.305	0.497

Table D.15.: Raw results of MGDP using Class Pattern Discovery on MTC-ANN. The parameters are minimum support $(p(P|\ominus))$, minimum distinctiveness $(I(P))$, minimum pattern size (n), minimum number of patterns and enforce minimum number of patterns.

D.5.5. PatMinr

Parameters	GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
Min. patt. size	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
3	0.741	0.686	0.741	0.567	0.713	0.689	0.713	0.556
4	0.698	0.681	0.698	0.547	0.669	0.653	0.669	0.531
5	0.615	0.592	0.615	0.472	0.587	0.567	0.587	0.481
6	0.524	0.533	0.524	0.422	0.501	0.536	0.501	0.425
7	0.454	0.489	0.454	0.414	0.435	0.494	0.435	0.408

Table D.16.: Raw results of PatMinr using Class Pattern Discovery on MTC-ANN.

D.6. Song Pattern Discovery on the Irish Folk Tunes Dataset

D.6.1. SIACTTEC

Parameters		GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
Min. patt. size	Min. compactness	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.1	3	0.999	0.958	0.999	0.921	0.979	0.955	0.979	0.924
0.1	4	0.998	0.955	0.998	0.921	0.977	0.958	0.977	0.924
0.1	5	0.995	0.955	0.995	0.919	0.972	0.947	0.972	0.916
0.1	6	0.992	0.955	0.992	0.924	0.965	0.941	0.965	0.919
0.1	7	0.988	0.955	0.988	0.919	0.957	0.941	0.957	0.910
0.3	3	0.997	0.958	0.997	0.921	0.864	0.933	0.864	0.921
0.3	4	0.991	0.955	0.991	0.919	0.847	0.933	0.847	0.913
0.3	5	0.980	0.947	0.980	0.907	0.820	0.924	0.820	0.874
0.3	6	0.963	0.933	0.963	0.899	0.788	0.910	0.788	0.888
0.3	7	0.941	0.933	0.941	0.882	0.752	0.888	0.752	0.868
0.5	3	0.992	0.952	0.992	0.921	0.728	0.924	0.728	0.910
0.5	4	0.972	0.944	0.972	0.913	0.695	0.916	0.695	0.902
0.5	5	0.941	0.938	0.941	0.882	0.649	0.904	0.649	0.843
0.5	6	0.892	0.933	0.892	0.879	0.593	0.907	0.593	0.846
0.5	7	0.836	0.916	0.836	0.848	0.541	0.888	0.541	0.817
0.7	3	0.973	0.941	0.973	0.904	0.576	0.902	0.576	0.876
0.7	4	0.916	0.930	0.916	0.904	0.535	0.904	0.535	0.876
0.7	5	0.854	0.913	0.854	0.862	0.490	0.885	0.490	0.837
0.7	6	0.775	0.890	0.775	0.823	0.439	0.862	0.439	0.772
0.7	7	0.717	0.874	0.717	0.806	0.405	0.860	0.405	0.758
0.9	3	0.953	0.935	0.953	0.893	0.491	0.896	0.491	0.854
0.9	4	0.865	0.916	0.865	0.871	0.449	0.882	0.449	0.840
0.9	5	0.767	0.899	0.767	0.801	0.401	0.868	0.401	0.761
0.9	6	0.680	0.854	0.680	0.772	0.356	0.846	0.356	0.744
0.9	7	0.624	0.834	0.624	0.770	0.326	0.806	0.326	0.733
1.0	3	0.953	0.935	0.953	0.893	0.486	0.896	0.486	0.854
1.0	4	0.863	0.916	0.863	0.871	0.444	0.888	0.444	0.840
1.0	5	0.763	0.899	0.763	0.801	0.395	0.868	0.395	0.753
1.0	6	0.672	0.857	0.672	0.770	0.350	0.851	0.350	0.739
1.0	7	0.614	0.829	0.614	0.772	0.319	0.815	0.319	0.744

Table D.17.: Raw results of SIACTTEC using Song Pattern Discovery on the Irish Folk Tunes Dataset.

D.6.2. COSIACTTEC

Parameters		GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
Min. patt. size	Min. compactness	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.1	3	0.863	0.904	0.863	0.820	0.197	0.385	0.197	0.351
0.1	4	0.826	0.899	0.826	0.753	0.184	0.354	0.184	0.340
0.1	5	0.793	0.899	0.793	0.711	0.174	0.337	0.174	0.295
0.1	6	0.772	0.865	0.772	0.733	0.162	0.337	0.162	0.258
0.1	7	0.756	0.840	0.756	0.699	0.159	0.346	0.159	0.317
0.3	3	0.845	0.907	0.845	0.787	0.211	0.447	0.211	0.399
0.3	4	0.795	0.876	0.795	0.744	0.206	0.452	0.206	0.427
0.3	5	0.756	0.854	0.756	0.691	0.202	0.455	0.202	0.374
0.3	6	0.728	0.840	0.728	0.705	0.197	0.463	0.197	0.385
0.3	7	0.702	0.857	0.702	0.725	0.189	0.469	0.189	0.343
0.5	3	0.829	0.874	0.829	0.772	0.219	0.584	0.219	0.469
0.5	4	0.767	0.885	0.767	0.758	0.224	0.553	0.224	0.528
0.5	5	0.726	0.865	0.726	0.716	0.236	0.607	0.236	0.517
0.5	6	0.680	0.857	0.680	0.694	0.232	0.624	0.232	0.503
0.5	7	0.645	0.831	0.645	0.705	0.236	0.660	0.236	0.570

(COSIACTTEC, continued from previous page)

Parameters		GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
Min. patt. size	**Min. compactness**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
0.7	3	0.811	0.879	0.811	0.770	0.227	0.649	0.227	0.559
0.7	4	0.740	0.882	0.740	0.747	0.236	0.683	0.236	0.576
0.7	5	0.687	0.857	0.687	0.694	0.242	0.742	0.242	0.615
0.7	6	0.650	0.857	0.650	0.730	0.250	0.784	0.250	0.635
0.7	7	0.604	0.851	0.604	0.730	0.248	0.775	0.248	0.671
0.9	3	0.808	0.902	0.808	0.815	0.228	0.691	0.228	0.610
0.9	4	0.723	0.848	0.723	0.758	0.239	0.761	0.239	0.601
0.9	5	0.662	0.848	0.662	0.725	0.251	0.764	0.251	0.621
0.9	6	0.609	0.843	0.609	0.730	0.248	0.806	0.248	0.669
0.9	7	0.551	0.831	0.551	0.725	0.235	0.781	0.235	0.674
1.0	3	0.808	0.899	0.808	0.809	0.228	0.697	0.228	0.596
1.0	4	0.724	0.854	0.724	0.761	0.240	0.761	0.240	0.596
1.0	5	0.660	0.846	0.660	0.733	0.249	0.781	0.249	0.629
1.0	6	0.604	0.840	0.604	0.728	0.245	0.817	0.245	0.677
1.0	7	0.547	0.826	0.547	0.713	0.232	0.778	0.232	0.666

Table D.18.: Raw results of COSIACTTEC using Song Pattern Discovery on the Irish Folk Tunes Dataset.

D.6.3. MotivesExtractor

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
2	0.7	3	0	0.360	0.669	0.360	0.621	0.198	0.640	0.198	0.590
3	0.7	3	0	0.389	0.677	0.389	0.615	0.236	0.601	0.236	0.565
4	0.7	3	0	0.360	0.621	0.360	0.517	0.243	0.537	0.243	0.475
2	0.7	3	2	0.359	0.677	0.359	0.626	0.195	0.629	0.195	0.590
3	0.7	3	2	0.388	0.680	0.388	0.615	0.233	0.598	0.233	0.573
4	0.7	3	2	0.358	0.621	0.358	0.508	0.238	0.545	0.238	0.469
2	0.7	3	4	0.349	0.688	0.349	0.626	0.182	0.643	0.182	0.581
3	0.7	3	4	0.379	0.671	0.379	0.615	0.217	0.590	0.217	0.562
4	0.7	3	4	0.351	0.615	0.351	0.506	0.221	0.528	0.221	0.463
2	0.7	3	6	0.340	0.680	0.340	0.610	0.166	0.635	0.166	0.576
3	0.7	3	6	0.370	0.657	0.370	0.610	0.198	0.576	0.198	0.559
4	0.7	3	6	0.345	0.629	0.345	0.508	0.204	0.508	0.204	0.469
2	0.7	3	8	0.334	0.677	0.334	0.601	0.154	0.626	0.154	0.587
3	0.7	3	8	0.363	0.677	0.363	0.601	0.180	0.598	0.180	0.562
4	0.7	3	8	0.340	0.624	0.340	0.506	0.188	0.514	0.188	0.475
2	0.7	4	0	0.407	0.702	0.407	0.621	0.225	0.660	0.225	0.579
3	0.7	4	0	0.433	0.680	0.433	0.626	0.277	0.652	0.277	0.590
4	0.7	4	0	0.418	0.688	0.418	0.581	0.301	0.612	0.301	0.548
2	0.7	4	2	0.406	0.702	0.406	0.626	0.223	0.674	0.223	0.579
3	0.7	4	2	0.431	0.683	0.431	0.638	0.272	0.654	0.272	0.593
4	0.7	4	2	0.416	0.688	0.416	0.579	0.296	0.610	0.296	0.542
2	0.7	4	4	0.404	0.702	0.404	0.626	0.219	0.654	0.219	0.567
3	0.7	4	4	0.428	0.685	0.428	0.635	0.264	0.663	0.264	0.593
4	0.7	4	4	0.414	0.685	0.414	0.579	0.287	0.607	0.287	0.545
2	0.7	4	6	0.393	0.694	0.393	0.621	0.202	0.652	0.202	0.567
3	0.7	4	6	0.420	0.674	0.420	0.640	0.247	0.660	0.247	0.596
4	0.7	4	6	0.408	0.677	0.408	0.562	0.268	0.612	0.268	0.522
2	0.7	4	8	0.385	0.694	0.385	0.635	0.190	0.652	0.190	0.553
3	0.7	4	8	0.413	0.680	0.413	0.635	0.228	0.638	0.228	0.593
4	0.7	4	8	0.404	0.683	0.404	0.567	0.247	0.584	0.247	0.537
2	0.7	5	0	0.439	0.758	0.439	0.688	0.239	0.697	0.239	0.646
3	0.7	5	0	0.482	0.711	0.482	0.590	0.305	0.671	0.305	0.567
4	0.7	5	0	0.466	0.733	0.466	0.601	0.351	0.680	0.351	0.565

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
2	0.7	5	2	0.438	0.758	0.438	0.685	0.238	0.705	0.238	0.643
3	0.7	5	2	0.481	0.713	0.481	0.587	0.302	0.663	0.302	0.559
4	0.7	5	2	0.464	0.728	0.464	0.601	0.346	0.666	0.346	0.562
2	0.7	5	4	0.437	0.756	0.437	0.685	0.235	0.702	0.235	0.638
3	0.7	5	4	0.479	0.713	0.479	0.593	0.296	0.657	0.296	0.562
4	0.7	5	4	0.463	0.728	0.463	0.598	0.341	0.654	0.341	0.556
2	0.7	5	6	0.432	0.758	0.432	0.683	0.227	0.691	0.227	0.649
3	0.7	5	6	0.474	0.713	0.474	0.590	0.284	0.652	0.284	0.553
4	0.7	5	6	0.460	0.722	0.460	0.598	0.329	0.649	0.329	0.553
2	0.7	5	8	0.426	0.756	0.426	0.683	0.214	0.699	0.214	0.638
3	0.7	5	8	0.467	0.716	0.467	0.593	0.266	0.643	0.266	0.542
4	0.7	5	8	0.456	0.713	0.456	0.596	0.308	0.621	0.308	0.551
2	0.7	6	0	0.478	0.775	0.478	0.688	0.265	0.722	0.265	0.669
3	0.7	6	0	0.498	0.733	0.498	0.671	0.325	0.702	0.325	0.649
4	0.7	6	0	0.496	0.770	0.496	0.596	0.389	0.733	0.389	0.581
2	0.7	6	2	0.478	0.775	0.478	0.691	0.264	0.725	0.264	0.666
3	0.7	6	2	0.497	0.733	0.497	0.669	0.323	0.702	0.323	0.643
4	0.7	6	2	0.496	0.770	0.496	0.598	0.387	0.725	0.387	0.579
2	0.7	6	4	0.476	0.772	0.476	0.685	0.261	0.725	0.261	0.660
3	0.7	6	4	0.496	0.733	0.496	0.663	0.319	0.699	0.319	0.643
4	0.7	6	4	0.495	0.767	0.495	0.604	0.383	0.725	0.383	0.565
2	0.7	6	6	0.474	0.772	0.474	0.688	0.258	0.722	0.258	0.660
3	0.7	6	6	0.494	0.733	0.494	0.663	0.313	0.699	0.313	0.632
4	0.7	6	6	0.493	0.761	0.493	0.601	0.376	0.719	0.376	0.567
2	0.7	6	8	0.467	0.770	0.467	0.680	0.245	0.713	0.245	0.643
3	0.7	6	8	0.488	0.744	0.488	0.657	0.298	0.694	0.298	0.624
4	0.7	6	8	0.489	0.764	0.489	0.604	0.357	0.711	0.357	0.545
2	0.7	7	0	0.516	0.803	0.516	0.742	0.291	0.772	0.291	0.728
3	0.7	7	0	0.556	0.801	0.556	0.716	0.380	0.728	0.380	0.674
4	0.7	7	0	0.546	0.815	0.546	0.660	0.440	0.787	0.440	0.652
2	0.7	7	2	0.516	0.798	0.516	0.742	0.290	0.772	0.290	0.728
3	0.7	7	2	0.555	0.803	0.555	0.716	0.379	0.728	0.379	0.674
4	0.7	7	2	0.545	0.820	0.545	0.660	0.438	0.784	0.438	0.635
2	0.7	7	4	0.515	0.798	0.515	0.742	0.288	0.770	0.288	0.722
3	0.7	7	4	0.555	0.803	0.555	0.716	0.374	0.725	0.374	0.674
4	0.7	7	4	0.544	0.815	0.544	0.660	0.433	0.784	0.433	0.638
2	0.7	7	6	0.514	0.798	0.514	0.742	0.285	0.770	0.285	0.719
3	0.7	7	6	0.554	0.803	0.554	0.716	0.370	0.730	0.370	0.674
4	0.7	7	6	0.543	0.815	0.543	0.657	0.430	0.781	0.430	0.635
2	0.7	7	8	0.509	0.795	0.509	0.736	0.275	0.764	0.275	0.719
3	0.7	7	8	0.550	0.801	0.550	0.713	0.359	0.713	0.359	0.666
4	0.7	7	8	0.540	0.815	0.540	0.663	0.418	0.772	0.418	0.626
2	0.5	3	0	0.411	0.719	0.411	0.649	0.224	0.660	0.224	0.610
3	0.5	3	0	0.791	0.907	0.791	0.851	0.629	0.829	0.629	0.761
4	0.5	3	0	0.895	0.935	0.895	0.893	0.828	0.910	0.828	0.896
2	0.5	3	2	0.409	0.713	0.409	0.657	0.220	0.646	0.220	0.607
3	0.5	3	2	0.789	0.910	0.789	0.848	0.620	0.826	0.620	0.725
4	0.5	3	2	0.894	0.938	0.894	0.893	0.823	0.904	0.823	0.882
2	0.5	3	4	0.398	0.716	0.398	0.640	0.205	0.638	0.205	0.593
3	0.5	3	4	0.781	0.902	0.781	0.834	0.581	0.801	0.581	0.694
4	0.5	3	4	0.889	0.933	0.889	0.890	0.793	0.888	0.793	0.831
2	0.5	3	6	0.387	0.711	0.387	0.635	0.187	0.632	0.187	0.590
3	0.5	3	6	0.769	0.896	0.769	0.829	0.526	0.772	0.526	0.669
4	0.5	3	6	0.885	0.927	0.885	0.882	0.753	0.871	0.753	0.812
2	0.5	3	8	0.380	0.716	0.380	0.638	0.173	0.632	0.173	0.593
3	0.5	3	8	0.761	0.890	0.761	0.812	0.464	0.784	0.464	0.610
4	0.5	3	8	0.880	0.927	0.880	0.888	0.703	0.871	0.703	0.798
2	0.5	4	0	0.422	0.708	0.422	0.649	0.232	0.683	0.232	0.593
3	0.5	4	0	0.690	0.815	0.690	0.756	0.523	0.781	0.523	0.694
4	0.5	4	0	0.814	0.907	0.814	0.831	0.739	0.857	0.739	0.820

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
2	0.5	4	2	0.420	0.708	0.420	0.646	0.229	0.694	0.229	0.593
3	0.5	4	2	0.689	0.812	0.689	0.756	0.517	0.772	0.517	0.652
4	0.5	4	2	0.813	0.907	0.813	0.829	0.735	0.857	0.735	0.806
2	0.5	4	4	0.418	0.708	0.418	0.643	0.225	0.677	0.225	0.587
3	0.5	4	4	0.686	0.809	0.686	0.756	0.503	0.784	0.503	0.640
4	0.5	4	4	0.811	0.907	0.811	0.829	0.726	0.848	0.726	0.806
2	0.5	4	6	0.407	0.702	0.407	0.640	0.209	0.674	0.209	0.587
3	0.5	4	6	0.678	0.806	0.678	0.747	0.472	0.772	0.472	0.638
4	0.5	4	6	0.809	0.904	0.809	0.829	0.701	0.837	0.701	0.775
2	0.5	4	8	0.400	0.699	0.400	0.654	0.196	0.671	0.196	0.579
3	0.5	4	8	0.671	0.806	0.671	0.742	0.431	0.764	0.431	0.601
4	0.5	4	8	0.806	0.902	0.806	0.829	0.670	0.834	0.670	0.775
2	0.5	5	0	0.448	0.758	0.448	0.697	0.244	0.694	0.244	0.649
3	0.5	5	0	0.653	0.798	0.653	0.742	0.463	0.784	0.463	0.688
4	0.5	5	0	0.787	0.879	0.787	0.803	0.720	0.831	0.720	0.767
2	0.5	5	2	0.447	0.758	0.447	0.694	0.243	0.708	0.243	0.646
3	0.5	5	2	0.652	0.801	0.652	0.736	0.459	0.792	0.459	0.691
4	0.5	5	2	0.787	0.879	0.787	0.803	0.718	0.834	0.718	0.764
2	0.5	5	4	0.446	0.756	0.446	0.694	0.240	0.705	0.240	0.640
3	0.5	5	4	0.650	0.801	0.650	0.736	0.451	0.787	0.451	0.685
4	0.5	5	4	0.786	0.876	0.786	0.803	0.713	0.831	0.713	0.761
2	0.5	5	6	0.442	0.758	0.442	0.691	0.232	0.694	0.232	0.663
3	0.5	5	6	0.644	0.803	0.644	0.733	0.431	0.767	0.431	0.666
4	0.5	5	6	0.784	0.874	0.784	0.803	0.701	0.848	0.701	0.761
2	0.5	5	8	0.435	0.761	0.435	0.691	0.219	0.699	0.219	0.649
3	0.5	5	8	0.637	0.798	0.637	0.730	0.404	0.736	0.404	0.652
4	0.5	5	8	0.781	0.865	0.781	0.798	0.675	0.826	0.675	0.742
2	0.5	6	0	0.482	0.781	0.482	0.688	0.268	0.730	0.268	0.669
3	0.5	6	0	0.608	0.801	0.608	0.750	0.429	0.767	0.429	0.722
4	0.5	6	0	0.770	0.848	0.770	0.781	0.709	0.840	0.709	0.758
2	0.5	6	2	0.482	0.781	0.482	0.691	0.267	0.733	0.267	0.663
3	0.5	6	2	0.607	0.801	0.607	0.747	0.426	0.764	0.426	0.711
4	0.5	6	2	0.770	0.846	0.770	0.781	0.708	0.829	0.708	0.753
2	0.5	6	4	0.480	0.778	0.480	0.685	0.264	0.733	0.264	0.660
3	0.5	6	4	0.606	0.801	0.606	0.742	0.422	0.756	0.422	0.708
4	0.5	6	4	0.769	0.843	0.769	0.781	0.704	0.829	0.704	0.744
2	0.5	6	6	0.478	0.778	0.478	0.688	0.261	0.728	0.261	0.663
3	0.5	6	6	0.604	0.795	0.604	0.739	0.415	0.756	0.415	0.702
4	0.5	6	6	0.768	0.843	0.768	0.781	0.698	0.815	0.698	0.742
2	0.5	6	8	0.471	0.778	0.471	0.680	0.248	0.722	0.248	0.649
3	0.5	6	8	0.598	0.803	0.598	0.736	0.392	0.744	0.392	0.697
4	0.5	6	8	0.766	0.846	0.766	0.781	0.676	0.812	0.676	0.750
2	0.5	7	0	0.518	0.803	0.518	0.744	0.292	0.772	0.292	0.730
3	0.5	7	0	0.624	0.851	0.624	0.781	0.446	0.806	0.446	0.733
4	0.5	7	0	0.780	0.876	0.780	0.803	0.720	0.862	0.720	0.775
2	0.5	7	2	0.518	0.798	0.518	0.744	0.291	0.772	0.291	0.730
3	0.5	7	2	0.624	0.851	0.624	0.781	0.444	0.806	0.444	0.730
4	0.5	7	2	0.780	0.876	0.780	0.803	0.719	0.865	0.719	0.775
2	0.5	7	4	0.516	0.798	0.516	0.744	0.289	0.770	0.289	0.725
3	0.5	7	4	0.623	0.851	0.623	0.781	0.440	0.798	0.440	0.733
4	0.5	7	4	0.779	0.879	0.779	0.803	0.715	0.862	0.715	0.770
2	0.5	7	6	0.516	0.798	0.516	0.744	0.285	0.770	0.285	0.722
3	0.5	7	6	0.622	0.848	0.622	0.781	0.436	0.792	0.436	0.736
4	0.5	7	6	0.778	0.876	0.778	0.806	0.710	0.865	0.710	0.772
2	0.5	7	8	0.511	0.795	0.511	0.739	0.276	0.764	0.276	0.722
3	0.5	7	8	0.618	0.846	0.618	0.778	0.422	0.781	0.422	0.725
4	0.5	7	8	0.776	0.876	0.776	0.806	0.699	0.851	0.699	0.770
2	0.3	3	0	0.982	0.955	0.982	0.913	0.934	0.944	0.934	0.910
3	0.3	3	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	3	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
2	0.3	3	2	0.982	0.955	0.982	0.913	0.932	0.947	0.932	0.910
3	0.3	3	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	3	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	3	4	0.982	0.955	0.982	0.913	0.910	0.941	0.910	0.907
3	0.3	3	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	3	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	3	6	0.982	0.955	0.982	0.913	0.874	0.930	0.874	0.904
3	0.3	3	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	3	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	3	8	0.982	0.955	0.982	0.910	0.808	0.924	0.808	0.871
3	0.3	3	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	3	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	4	0	0.982	0.955	0.982	0.913	0.901	0.935	0.901	0.893
3	0.3	4	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	4	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	4	2	0.982	0.955	0.982	0.913	0.899	0.935	0.899	0.893
3	0.3	4	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	4	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	4	4	0.982	0.955	0.982	0.913	0.886	0.938	0.886	0.893
3	0.3	4	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	4	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	4	6	0.981	0.955	0.981	0.913	0.852	0.933	0.852	0.888
3	0.3	4	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	4	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	4	8	0.981	0.955	0.981	0.913	0.799	0.919	0.799	0.871
3	0.3	4	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	4	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	5	0	0.980	0.955	0.980	0.913	0.867	0.935	0.867	0.902
3	0.3	5	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	5	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	5	2	0.980	0.955	0.980	0.913	0.866	0.933	0.866	0.902
3	0.3	5	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	5	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	5	4	0.979	0.955	0.979	0.913	0.857	0.927	0.857	0.896
3	0.3	5	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	5	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	5	6	0.979	0.955	0.979	0.913	0.836	0.927	0.836	0.890
3	0.3	5	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	5	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	5	8	0.978	0.955	0.978	0.910	0.797	0.935	0.797	0.882
3	0.3	5	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	5	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	6	0	0.973	0.952	0.973	0.916	0.834	0.935	0.834	0.902
3	0.3	6	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	6	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	6	2	0.973	0.955	0.973	0.916	0.832	0.935	0.832	0.902
3	0.3	6	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	6	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	6	4	0.973	0.955	0.973	0.916	0.827	0.933	0.827	0.902
3	0.3	6	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	6	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	6	6	0.972	0.952	0.972	0.916	0.815	0.927	0.815	0.899
3	0.3	6	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	6	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	6	8	0.970	0.952	0.970	0.913	0.784	0.924	0.784	0.888
3	0.3	6	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	6	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	7	0	0.963	0.952	0.963	0.916	0.792	0.927	0.792	0.885
3	0.3	7	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	7	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
2	0.3	7	2	0.963	0.952	0.963	0.916	0.791	0.924	0.791	0.885
3	0.3	7	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	7	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	7	4	0.963	0.952	0.963	0.916	0.787	0.933	0.787	0.882
3	0.3	7	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	7	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	7	6	0.962	0.952	0.962	0.916	0.780	0.930	0.780	0.876
3	0.3	7	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	7	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.3	7	8	0.960	0.952	0.960	0.916	0.757	0.924	0.757	0.871
3	0.3	7	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.3	7	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	3	0	0.982	0.955	0.982	0.913	0.981	0.952	0.981	0.913
3	0.1	3	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	3	0	0.982	0.955	0.982	0.913	0.977	0.949	0.977	0.913
2	0.1	3	2	0.982	0.955	0.982	0.913	0.980	0.952	0.980	0.913
3	0.1	3	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	3	2	0.982	0.955	0.982	0.913	0.977	0.949	0.977	0.913
2	0.1	3	4	0.982	0.955	0.982	0.913	0.980	0.952	0.980	0.913
3	0.1	3	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	3	4	0.982	0.955	0.982	0.913	0.977	0.949	0.977	0.913
2	0.1	3	6	0.982	0.955	0.982	0.913	0.978	0.952	0.978	0.913
3	0.1	3	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	3	6	0.982	0.955	0.982	0.913	0.977	0.949	0.977	0.913
2	0.1	3	8	0.982	0.955	0.982	0.913	0.976	0.952	0.976	0.913
3	0.1	3	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	3	8	0.982	0.955	0.982	0.913	0.977	0.949	0.977	0.913
2	0.1	4	0	0.982	0.955	0.982	0.913	0.980	0.952	0.980	0.913
3	0.1	4	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	4	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	4	2	0.982	0.955	0.982	0.913	0.979	0.952	0.979	0.913
3	0.1	4	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	4	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	4	4	0.982	0.955	0.982	0.913	0.979	0.952	0.979	0.913
3	0.1	4	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	4	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	4	6	0.982	0.955	0.982	0.913	0.978	0.952	0.978	0.913
3	0.1	4	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	4	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	4	8	0.982	0.955	0.982	0.913	0.978	0.952	0.978	0.913
3	0.1	4	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	4	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	5	0	0.982	0.955	0.982	0.913	0.979	0.949	0.979	0.913
3	0.1	5	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	5	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	5	2	0.982	0.955	0.982	0.913	0.979	0.949	0.979	0.913
3	0.1	5	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	5	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	5	4	0.982	0.955	0.982	0.913	0.979	0.949	0.979	0.913
3	0.1	5	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	5	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	5	6	0.982	0.955	0.982	0.913	0.977	0.949	0.977	0.913
3	0.1	5	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	5	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	5	8	0.982	0.955	0.982	0.913	0.975	0.949	0.975	0.913
3	0.1	5	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	5	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	6	0	0.982	0.955	0.982	0.913	0.978	0.944	0.978	0.913
3	0.1	6	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	6	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
2	0.1	6	2	0.982	0.955	0.982	0.913	0.978	0.944	0.978	0.913
3	0.1	6	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	6	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	6	4	0.982	0.955	0.982	0.913	0.977	0.944	0.977	0.913
3	0.1	6	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	6	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	6	6	0.982	0.955	0.982	0.913	0.976	0.944	0.976	0.913
3	0.1	6	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	6	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	6	8	0.982	0.955	0.982	0.913	0.975	0.944	0.975	0.913
3	0.1	6	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	6	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	7	0	0.982	0.955	0.982	0.913	0.976	0.944	0.976	0.913
3	0.1	7	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	7	0	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	7	2	0.982	0.955	0.982	0.913	0.976	0.944	0.976	0.913
3	0.1	7	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	7	2	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	7	4	0.982	0.955	0.982	0.913	0.976	0.947	0.976	0.913
3	0.1	7	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	7	4	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	7	6	0.982	0.955	0.982	0.913	0.976	0.947	0.976	0.913
3	0.1	7	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	7	6	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
2	0.1	7	8	0.982	0.955	0.982	0.913	0.974	0.947	0.974	0.913
3	0.1	7	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913
4	0.1	7	8	0.982	0.955	0.982	0.913	0.977	0.947	0.977	0.913

Table D.19.: Raw results of MotivesExtractor using Song Pattern Discovery on the Irish Folk Tunes Dataset. The parameters are the path tracing tolerance (ρ), the scoring threshold (θ), the minimum pattern length (ν) and the tolerance parameter for the pattern clustering (Θ).

D.6.4. SIARCT-CFP

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
b	a	m	c	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
0.5	3	0.7	0.3	1.000	0.958	1.000	0.921	0.906	0.933	0.906	0.913
0.5	3	0.9	0.3	0.998	0.955	0.998	0.919	0.919	0.944	0.919	0.916
0.5	3	1.0	0.3	0.995	0.952	0.995	0.921	0.919	0.944	0.919	0.916
0.5	3	0.7	0.5	1.000	0.958	1.000	0.921	0.906	0.933	0.906	0.913
0.5	3	0.9	0.5	0.998	0.955	0.998	0.919	0.919	0.944	0.919	0.916
0.5	3	1.0	0.5	0.995	0.952	0.995	0.921	0.919	0.944	0.919	0.916
0.5	3	0.7	0.7	1.000	0.958	1.000	0.921	0.906	0.933	0.906	0.913
0.5	3	0.9	0.7	0.998	0.955	0.998	0.919	0.919	0.944	0.919	0.916
0.5	3	1.0	0.7	0.995	0.952	0.995	0.921	0.919	0.944	0.919	0.916
0.5	3	0.7	0.9	1.000	0.958	1.000	0.921	0.906	0.933	0.906	0.913
0.5	3	0.9	0.9	0.998	0.955	0.998	0.919	0.919	0.944	0.919	0.916
0.5	3	1.0	0.9	0.995	0.952	0.995	0.921	0.919	0.944	0.919	0.916
0.7	3	0.7	0.3	0.999	0.955	0.999	0.919	0.841	0.941	0.841	0.904
0.7	3	0.9	0.3	0.998	0.952	0.998	0.916	0.865	0.935	0.865	0.913
0.7	3	1.0	0.3	0.995	0.949	0.995	0.916	0.865	0.935	0.865	0.913
0.7	3	0.7	0.5	0.999	0.955	0.999	0.919	0.841	0.941	0.841	0.907
0.7	3	0.9	0.5	0.998	0.952	0.998	0.916	0.865	0.935	0.865	0.913
0.7	3	1.0	0.5	0.995	0.949	0.995	0.916	0.865	0.935	0.865	0.913
0.7	3	0.7	0.7	0.999	0.955	0.999	0.919	0.841	0.941	0.841	0.907

(SIARCT-CFP, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
b	a	m	c	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
0.7	3	0.9	0.7	0.998	0.952	0.998	0.916	0.865	0.935	0.865	0.910
0.7	3	1.0	0.7	0.995	0.949	0.995	0.916	0.865	0.935	0.865	0.913
0.7	3	0.7	0.9	0.999	0.955	0.999	0.919	0.841	0.941	0.841	0.907
0.7	3	0.9	0.9	0.998	0.952	0.998	0.916	0.865	0.935	0.865	0.910
0.7	3	1.0	0.9	0.995	0.949	0.995	0.916	0.865	0.935	0.865	0.913
0.9	3	0.7	0.3	0.998	0.958	0.998	0.919	0.757	0.919	0.757	0.882
0.9	3	0.9	0.3	0.996	0.958	0.996	0.916	0.792	0.927	0.792	0.899
0.9	3	1.0	0.3	0.993	0.958	0.993	0.919	0.794	0.924	0.794	0.896
0.9	3	0.7	0.5	0.998	0.958	0.998	0.919	0.757	0.919	0.757	0.882
0.9	3	0.9	0.5	0.996	0.958	0.996	0.916	0.792	0.927	0.792	0.899
0.9	3	1.0	0.5	0.993	0.958	0.993	0.919	0.794	0.924	0.794	0.896
0.9	3	0.7	0.7	0.998	0.958	0.998	0.919	0.757	0.919	0.757	0.882
0.9	3	0.9	0.7	0.996	0.958	0.996	0.916	0.792	0.927	0.792	0.899
0.9	3	1.0	0.7	0.993	0.958	0.993	0.919	0.794	0.924	0.794	0.896
0.9	3	0.7	0.9	0.998	0.958	0.998	0.919	0.757	0.919	0.757	0.882
0.9	3	0.9	0.9	0.996	0.958	0.996	0.916	0.792	0.927	0.792	0.899
0.9	3	1.0	0.9	0.993	0.958	0.993	0.919	0.794	0.924	0.794	0.896
1.0	3	0.7	0.3	0.998	0.958	0.998	0.919	0.755	0.919	0.755	0.882
1.0	3	0.9	0.3	0.996	0.955	0.996	0.916	0.791	0.933	0.791	0.899
1.0	3	1.0	0.3	0.992	0.955	0.992	0.919	0.793	0.933	0.793	0.896
1.0	3	0.7	0.5	0.998	0.958	0.998	0.919	0.755	0.919	0.755	0.882
1.0	3	0.9	0.5	0.996	0.955	0.996	0.916	0.791	0.933	0.791	0.899
1.0	3	1.0	0.5	0.992	0.955	0.992	0.919	0.793	0.933	0.793	0.896
1.0	3	0.7	0.7	0.998	0.958	0.998	0.919	0.755	0.919	0.755	0.882
1.0	3	0.9	0.7	0.996	0.955	0.996	0.916	0.791	0.933	0.791	0.899
1.0	3	1.0	0.7	0.992	0.955	0.992	0.919	0.793	0.933	0.793	0.896
1.0	3	0.7	0.9	0.998	0.958	0.998	0.919	0.755	0.919	0.755	0.882
1.0	3	0.9	0.9	0.996	0.955	0.996	0.916	0.791	0.933	0.791	0.899
1.0	3	1.0	0.9	0.992	0.955	0.992	0.919	0.793	0.933	0.793	0.896
0.5	4	0.7	0.3	0.999	0.955	0.999	0.921	0.899	0.927	0.899	0.907
0.5	4	0.9	0.3	0.996	0.955	0.996	0.919	0.912	0.935	0.912	0.913
0.5	4	1.0	0.3	0.993	0.952	0.993	0.919	0.912	0.935	0.912	0.913
0.5	4	0.7	0.5	0.999	0.955	0.999	0.921	0.899	0.927	0.899	0.907
0.5	4	0.9	0.5	0.996	0.955	0.996	0.919	0.912	0.935	0.912	0.913
0.5	4	1.0	0.5	0.993	0.952	0.993	0.919	0.912	0.935	0.912	0.913
0.5	4	0.7	0.7	0.999	0.955	0.999	0.921	0.899	0.927	0.899	0.907
0.5	4	0.9	0.7	0.996	0.955	0.996	0.919	0.912	0.935	0.912	0.913
0.5	4	1.0	0.7	0.993	0.952	0.993	0.919	0.912	0.935	0.912	0.913
0.5	4	0.7	0.9	0.999	0.955	0.999	0.921	0.899	0.927	0.899	0.907
0.5	4	0.9	0.9	0.996	0.955	0.996	0.919	0.912	0.935	0.912	0.913
0.5	4	1.0	0.9	0.993	0.952	0.993	0.919	0.912	0.935	0.912	0.913
0.7	4	0.7	0.3	0.996	0.952	0.996	0.913	0.813	0.933	0.813	0.904
0.7	4	0.9	0.3	0.992	0.952	0.992	0.907	0.842	0.933	0.842	0.904
0.7	4	1.0	0.3	0.988	0.949	0.988	0.907	0.842	0.933	0.842	0.904
0.7	4	0.7	0.5	0.996	0.952	0.996	0.913	0.813	0.933	0.813	0.904
0.7	4	0.9	0.5	0.992	0.952	0.992	0.907	0.842	0.933	0.842	0.904
0.7	4	1.0	0.5	0.988	0.949	0.988	0.907	0.842	0.933	0.842	0.904
0.7	4	0.7	0.7	0.996	0.952	0.996	0.913	0.813	0.933	0.813	0.904
0.7	4	0.9	0.7	0.992	0.952	0.992	0.907	0.842	0.933	0.842	0.904
0.7	4	1.0	0.7	0.988	0.949	0.988	0.907	0.842	0.933	0.842	0.904
0.7	4	0.7	0.9	0.996	0.952	0.996	0.913	0.813	0.933	0.813	0.904
0.7	4	0.9	0.9	0.992	0.952	0.992	0.907	0.842	0.933	0.842	0.902
0.7	4	1.0	0.9	0.988	0.949	0.988	0.907	0.842	0.933	0.842	0.904
0.9	4	0.7	0.3	0.993	0.952	0.993	0.913	0.697	0.902	0.697	0.876
0.9	4	0.9	0.3	0.985	0.952	0.985	0.904	0.744	0.916	0.744	0.879
0.9	4	1.0	0.3	0.979	0.952	0.979	0.910	0.747	0.916	0.747	0.879
0.9	4	0.7	0.5	0.993	0.952	0.993	0.913	0.697	0.902	0.697	0.876
0.9	4	0.9	0.5	0.985	0.952	0.985	0.904	0.744	0.916	0.744	0.879
0.9	4	1.0	0.5	0.979	0.952	0.979	0.910	0.747	0.916	0.747	0.879
0.9	4	0.7	0.7	0.993	0.952	0.993	0.913	0.697	0.902	0.697	0.876

(SIARCT-CFP, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
b	*a*	*m*	*c*	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
0.9	4	0.9	0.7	0.985	0.952	0.985	0.904	0.744	0.916	0.744	0.879
0.9	4	1.0	0.7	0.979	0.952	0.979	0.910	0.747	0.916	0.747	0.879
0.9	4	0.7	0.9	0.993	0.952	0.993	0.913	0.697	0.902	0.697	0.876
0.9	4	0.9	0.9	0.985	0.952	0.985	0.904	0.744	0.916	0.744	0.879
0.9	4	1.0	0.9	0.979	0.952	0.979	0.910	0.747	0.916	0.747	0.879
1.0	4	0.7	0.3	0.993	0.952	0.993	0.913	0.691	0.899	0.691	0.876
1.0	4	0.9	0.3	0.985	0.949	0.985	0.904	0.742	0.919	0.742	0.879
1.0	4	1.0	0.3	0.979	0.952	0.979	0.910	0.745	0.919	0.745	0.879
1.0	4	0.7	0.5	0.993	0.952	0.993	0.913	0.691	0.899	0.691	0.876
1.0	4	0.9	0.5	0.985	0.949	0.985	0.904	0.742	0.919	0.742	0.879
1.0	4	1.0	0.5	0.979	0.952	0.979	0.910	0.745	0.919	0.745	0.879
1.0	4	0.7	0.7	0.993	0.952	0.993	0.913	0.691	0.899	0.691	0.876
1.0	4	0.9	0.7	0.985	0.949	0.985	0.904	0.742	0.919	0.742	0.879
1.0	4	1.0	0.7	0.979	0.952	0.979	0.910	0.745	0.919	0.745	0.879
1.0	4	0.7	0.9	0.993	0.952	0.993	0.913	0.691	0.899	0.691	0.876
1.0	4	0.9	0.9	0.985	0.949	0.985	0.904	0.742	0.919	0.742	0.879
1.0	4	1.0	0.9	0.979	0.952	0.979	0.910	0.745	0.919	0.745	0.879
0.5	5	0.7	0.3	0.996	0.955	0.996	0.916	0.882	0.924	0.882	0.896
0.5	5	0.9	0.3	0.990	0.958	0.990	0.916	0.893	0.930	0.893	0.902
0.5	5	1.0	0.3	0.987	0.952	0.987	0.916	0.893	0.930	0.893	0.902
0.5	5	0.7	0.5	0.996	0.955	0.996	0.916	0.882	0.924	0.882	0.896
0.5	5	0.9	0.5	0.990	0.958	0.990	0.916	0.893	0.930	0.893	0.902
0.5	5	1.0	0.5	0.987	0.952	0.987	0.916	0.893	0.930	0.893	0.902
0.5	5	0.7	0.7	0.996	0.955	0.996	0.916	0.882	0.924	0.882	0.896
0.5	5	0.9	0.7	0.990	0.958	0.990	0.916	0.893	0.930	0.893	0.902
0.5	5	1.0	0.7	0.987	0.952	0.987	0.916	0.893	0.930	0.893	0.902
0.5	5	0.7	0.9	0.996	0.955	0.996	0.916	0.882	0.924	0.882	0.896
0.5	5	0.9	0.9	0.990	0.958	0.990	0.916	0.893	0.930	0.893	0.902
0.5	5	1.0	0.9	0.987	0.952	0.987	0.916	0.893	0.930	0.893	0.902
0.7	5	0.7	0.3	0.990	0.947	0.990	0.907	0.777	0.916	0.777	0.857
0.7	5	0.9	0.3	0.979	0.952	0.979	0.904	0.801	0.924	0.801	0.871
0.7	5	1.0	0.3	0.974	0.949	0.974	0.902	0.802	0.924	0.802	0.871
0.7	5	0.7	0.5	0.990	0.947	0.990	0.907	0.777	0.916	0.777	0.857
0.7	5	0.9	0.5	0.979	0.952	0.979	0.904	0.801	0.924	0.801	0.871
0.7	5	1.0	0.5	0.974	0.949	0.974	0.902	0.802	0.924	0.802	0.871
0.7	5	0.7	0.7	0.990	0.947	0.990	0.907	0.777	0.916	0.777	0.857
0.7	5	0.9	0.7	0.979	0.952	0.979	0.904	0.801	0.924	0.801	0.871
0.7	5	1.0	0.7	0.974	0.949	0.974	0.902	0.802	0.924	0.802	0.871
0.7	5	0.7	0.9	0.990	0.947	0.990	0.904	0.777	0.916	0.777	0.857
0.7	5	0.9	0.9	0.979	0.952	0.979	0.904	0.801	0.924	0.801	0.871
0.7	5	1.0	0.9	0.974	0.949	0.974	0.902	0.802	0.924	0.802	0.871
0.9	5	0.7	0.3	0.966	0.935	0.966	0.899	0.614	0.902	0.614	0.837
0.9	5	0.9	0.3	0.943	0.938	0.943	0.888	0.650	0.907	0.650	0.854
0.9	5	1.0	0.3	0.934	0.935	0.934	0.888	0.654	0.904	0.654	0.854
0.9	5	0.7	0.5	0.966	0.935	0.966	0.899	0.614	0.902	0.614	0.837
0.9	5	0.9	0.5	0.943	0.938	0.943	0.888	0.650	0.907	0.650	0.854
0.9	5	1.0	0.5	0.934	0.935	0.934	0.888	0.654	0.904	0.654	0.854
0.9	5	0.7	0.7	0.966	0.935	0.966	0.899	0.614	0.902	0.614	0.837
0.9	5	0.9	0.7	0.943	0.938	0.943	0.888	0.650	0.907	0.650	0.854
0.9	5	1.0	0.7	0.934	0.935	0.934	0.888	0.654	0.904	0.654	0.854
0.9	5	0.7	0.9	0.966	0.935	0.966	0.899	0.614	0.902	0.614	0.837
0.9	5	0.9	0.9	0.943	0.938	0.943	0.888	0.650	0.907	0.650	0.854
0.9	5	1.0	0.9	0.934	0.935	0.934	0.888	0.654	0.904	0.654	0.854
1.0	5	0.7	0.3	0.965	0.933	0.965	0.902	0.606	0.902	0.606	0.834
1.0	5	0.9	0.3	0.942	0.938	0.942	0.888	0.646	0.902	0.646	0.851
1.0	5	1.0	0.3	0.933	0.935	0.933	0.888	0.649	0.899	0.649	0.851
1.0	5	0.7	0.5	0.965	0.933	0.965	0.902	0.606	0.902	0.606	0.834
1.0	5	0.9	0.5	0.942	0.938	0.942	0.888	0.646	0.902	0.646	0.851
1.0	5	1.0	0.5	0.933	0.935	0.933	0.888	0.649	0.899	0.649	0.851
1.0	5	0.7	0.7	0.965	0.933	0.965	0.902	0.606	0.902	0.606	0.834

(SIARCT-CFP, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
b	*a*	*m*	*c*	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
1.0	5	0.9	0.7	0.942	0.938	0.942	0.888	0.646	0.902	0.646	0.851
1.0	5	1.0	0.7	0.933	0.935	0.933	0.888	0.649	0.899	0.649	0.851
1.0	5	0.7	0.9	0.965	0.933	0.965	0.902	0.606	0.902	0.606	0.834
1.0	5	0.9	0.9	0.942	0.938	0.942	0.888	0.646	0.902	0.646	0.851
1.0	5	1.0	0.9	0.933	0.935	0.933	0.888	0.649	0.899	0.649	0.851
0.5	6	0.7	0.3	0.988	0.949	0.988	0.916	0.853	0.924	0.853	0.893
0.5	6	0.9	0.3	0.980	0.941	0.980	0.910	0.865	0.924	0.865	0.896
0.5	6	1.0	0.3	0.978	0.944	0.978	0.904	0.865	0.924	0.865	0.896
0.5	6	0.7	0.5	0.988	0.949	0.988	0.916	0.853	0.924	0.853	0.893
0.5	6	0.9	0.5	0.980	0.941	0.980	0.910	0.865	0.924	0.865	0.896
0.5	6	1.0	0.5	0.978	0.944	0.978	0.904	0.865	0.924	0.865	0.896
0.5	6	0.7	0.7	0.988	0.949	0.988	0.916	0.853	0.924	0.853	0.893
0.5	6	0.9	0.7	0.980	0.941	0.980	0.910	0.865	0.924	0.865	0.896
0.5	6	1.0	0.7	0.978	0.944	0.978	0.904	0.865	0.924	0.865	0.896
0.5	6	0.7	0.9	0.988	0.949	0.988	0.916	0.853	0.924	0.853	0.893
0.5	6	0.9	0.9	0.980	0.941	0.980	0.910	0.865	0.924	0.865	0.896
0.5	6	1.0	0.9	0.978	0.944	0.978	0.904	0.865	0.924	0.865	0.896
0.7	6	0.7	0.3	0.966	0.938	0.966	0.893	0.709	0.904	0.709	0.865
0.7	6	0.9	0.3	0.950	0.938	0.950	0.885	0.734	0.896	0.734	0.868
0.7	6	1.0	0.3	0.945	0.944	0.945	0.882	0.735	0.893	0.735	0.871
0.7	6	0.7	0.5	0.966	0.938	0.966	0.893	0.709	0.904	0.709	0.865
0.7	6	0.9	0.5	0.950	0.938	0.950	0.885	0.734	0.896	0.734	0.868
0.7	6	1.0	0.5	0.945	0.944	0.945	0.882	0.735	0.893	0.735	0.871
0.7	6	0.7	0.7	0.966	0.938	0.966	0.893	0.709	0.904	0.709	0.865
0.7	6	0.9	0.7	0.950	0.938	0.950	0.885	0.734	0.896	0.734	0.868
0.7	6	1.0	0.7	0.945	0.944	0.945	0.882	0.735	0.893	0.735	0.871
0.7	6	0.7	0.9	0.966	0.938	0.966	0.893	0.709	0.904	0.709	0.865
0.7	6	0.9	0.9	0.950	0.938	0.950	0.885	0.734	0.896	0.734	0.868
0.7	6	1.0	0.9	0.945	0.944	0.945	0.882	0.735	0.893	0.735	0.871
0.9	6	0.7	0.3	0.891	0.913	0.891	0.857	0.508	0.851	0.508	0.756
0.9	6	0.9	0.3	0.854	0.910	0.854	0.837	0.539	0.857	0.539	0.753
0.9	6	1.0	0.3	0.846	0.910	0.846	0.834	0.543	0.848	0.543	0.756
0.9	6	0.7	0.5	0.891	0.913	0.891	0.857	0.508	0.851	0.508	0.756
0.9	6	0.9	0.5	0.854	0.910	0.854	0.837	0.539	0.857	0.539	0.753
0.9	6	1.0	0.5	0.846	0.910	0.846	0.837	0.543	0.848	0.543	0.756
0.9	6	0.7	0.7	0.891	0.913	0.891	0.857	0.508	0.851	0.508	0.756
0.9	6	0.9	0.7	0.854	0.910	0.854	0.837	0.539	0.857	0.539	0.753
0.9	6	1.0	0.7	0.846	0.910	0.846	0.837	0.543	0.848	0.543	0.756
0.9	6	0.7	0.9	0.891	0.913	0.891	0.857	0.508	0.851	0.508	0.756
0.9	6	0.9	0.9	0.854	0.910	0.854	0.837	0.539	0.857	0.539	0.753
0.9	6	1.0	0.9	0.846	0.910	0.846	0.837	0.543	0.848	0.543	0.756
1.0	6	0.7	0.3	0.888	0.910	0.888	0.854	0.496	0.848	0.496	0.750
1.0	6	0.9	0.3	0.851	0.904	0.851	0.837	0.532	0.851	0.532	0.750
1.0	6	1.0	0.3	0.842	0.913	0.842	0.834	0.535	0.846	0.535	0.758
1.0	6	0.7	0.5	0.888	0.910	0.888	0.854	0.496	0.848	0.496	0.750
1.0	6	0.9	0.5	0.851	0.904	0.851	0.837	0.532	0.851	0.532	0.750
1.0	6	1.0	0.5	0.842	0.913	0.842	0.834	0.535	0.846	0.535	0.758
1.0	6	0.7	0.7	0.888	0.910	0.888	0.854	0.496	0.848	0.496	0.750
1.0	6	0.9	0.7	0.851	0.904	0.851	0.837	0.532	0.851	0.532	0.750
1.0	6	1.0	0.7	0.842	0.913	0.842	0.834	0.535	0.846	0.535	0.758
1.0	6	0.7	0.9	0.888	0.910	0.888	0.854	0.496	0.848	0.496	0.750
1.0	6	0.9	0.9	0.851	0.904	0.851	0.837	0.532	0.851	0.532	0.750
1.0	6	1.0	0.9	0.842	0.913	0.842	0.834	0.535	0.846	0.535	0.758
0.5	7	0.7	0.3	0.974	0.944	0.974	0.904	0.816	0.913	0.816	0.879
0.5	7	0.9	0.3	0.966	0.935	0.966	0.899	0.828	0.910	0.828	0.879
0.5	7	1.0	0.3	0.965	0.938	0.965	0.893	0.828	0.910	0.828	0.879
0.5	7	0.7	0.5	0.974	0.944	0.974	0.904	0.816	0.913	0.816	0.876
0.5	7	0.9	0.5	0.966	0.935	0.966	0.899	0.828	0.910	0.828	0.879
0.5	7	1.0	0.5	0.965	0.938	0.965	0.893	0.828	0.910	0.828	0.879
0.5	7	0.7	0.7	0.974	0.944	0.974	0.904	0.816	0.913	0.816	0.879

(SIARCT-CFP, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
b	a	m	c	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.5	7	0.9	0.7	0.966	0.935	0.966	0.899	0.828	0.910	0.828	0.879
0.5	7	1.0	0.7	0.965	0.938	0.965	0.893	0.828	0.910	0.828	0.879
0.5	7	0.7	0.9	0.974	0.944	0.974	0.904	0.816	0.913	0.816	0.876
0.5	7	0.9	0.9	0.966	0.935	0.966	0.899	0.828	0.910	0.828	0.879
0.5	7	1.0	0.9	0.965	0.938	0.965	0.893	0.828	0.910	0.828	0.879
0.7	7	0.7	0.3	0.930	0.933	0.930	0.882	0.645	0.899	0.645	0.809
0.7	7	0.9	0.3	0.912	0.933	0.912	0.876	0.668	0.893	0.668	0.817
0.7	7	1.0	0.3	0.908	0.930	0.908	0.876	0.670	0.890	0.670	0.820
0.7	7	0.7	0.5	0.930	0.933	0.930	0.882	0.645	0.899	0.645	0.809
0.7	7	0.9	0.5	0.912	0.933	0.912	0.876	0.668	0.893	0.668	0.817
0.7	7	1.0	0.5	0.908	0.930	0.908	0.876	0.670	0.890	0.670	0.820
0.7	7	0.7	0.7	0.930	0.933	0.930	0.882	0.645	0.899	0.645	0.809
0.7	7	0.9	0.7	0.912	0.933	0.912	0.876	0.668	0.893	0.668	0.817
0.7	7	1.0	0.7	0.908	0.930	0.908	0.876	0.670	0.890	0.670	0.820
0.7	7	0.7	0.9	0.930	0.933	0.930	0.882	0.645	0.899	0.645	0.809
0.7	7	0.9	0.9	0.912	0.933	0.912	0.876	0.668	0.893	0.668	0.817
0.7	7	1.0	0.9	0.908	0.930	0.908	0.876	0.670	0.890	0.670	0.820
0.9	7	0.7	0.3	0.791	0.893	0.791	0.812	0.421	0.809	0.421	0.733
0.9	7	0.9	0.3	0.752	0.885	0.752	0.787	0.444	0.829	0.444	0.747
0.9	7	1.0	0.3	0.747	0.882	0.747	0.787	0.448	0.831	0.448	0.750
0.9	7	0.7	0.5	0.791	0.893	0.791	0.815	0.421	0.809	0.421	0.733
0.9	7	0.9	0.5	0.752	0.885	0.752	0.784	0.444	0.829	0.444	0.747
0.9	7	1.0	0.5	0.747	0.882	0.747	0.784	0.448	0.831	0.448	0.750
0.9	7	0.7	0.7	0.791	0.893	0.791	0.812	0.421	0.809	0.421	0.733
0.9	7	0.9	0.7	0.752	0.885	0.752	0.784	0.444	0.829	0.444	0.747
0.9	7	1.0	0.7	0.747	0.882	0.747	0.787	0.448	0.831	0.448	0.750
0.9	7	0.7	0.9	0.791	0.893	0.791	0.815	0.421	0.809	0.421	0.733
0.9	7	0.9	0.9	0.752	0.885	0.752	0.784	0.444	0.829	0.444	0.747
0.9	7	1.0	0.9	0.747	0.882	0.747	0.784	0.448	0.831	0.448	0.750
1.0	7	0.7	0.3	0.785	0.885	0.785	0.803	0.403	0.809	0.403	0.728
1.0	7	0.9	0.3	0.745	0.874	0.745	0.789	0.431	0.823	0.431	0.753
1.0	7	1.0	0.3	0.739	0.871	0.739	0.778	0.435	0.815	0.435	0.750
1.0	7	0.7	0.5	0.785	0.885	0.785	0.803	0.403	0.809	0.403	0.728
1.0	7	0.9	0.5	0.745	0.874	0.745	0.789	0.431	0.823	0.431	0.753
1.0	7	1.0	0.5	0.739	0.871	0.739	0.781	0.435	0.815	0.435	0.750
1.0	7	0.7	0.7	0.785	0.885	0.785	0.803	0.403	0.809	0.403	0.728
1.0	7	0.9	0.7	0.745	0.874	0.745	0.789	0.431	0.823	0.431	0.753
1.0	7	1.0	0.7	0.739	0.871	0.739	0.781	0.435	0.815	0.435	0.750
1.0	7	0.7	0.9	0.785	0.885	0.785	0.803	0.403	0.809	0.403	0.728
1.0	7	0.9	0.9	0.745	0.874	0.745	0.789	0.431	0.823	0.431	0.753
1.0	7	1.0	0.9	0.739	0.871	0.739	0.781	0.435	0.815	0.435	0.750

Table D.20.: Raw results of SIARCT-CFP using Song Pattern Discovery on the Irish Folk Tunes Dataset. The parameters are: minimum compactness (b), minimum pattern size (a), inexactness threshold (m) and similarity threshold (c).

D.6.5. PatMinr

Parameters	GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
Min. patt. size	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
3	0.947	0.924	0.947	0.899	0.544	0.893	0.544	0.871
4	0.849	0.910	0.849	0.834	0.472	0.876	0.472	0.801
5	0.737	0.876	0.737	0.795	0.397	0.831	0.397	0.753
6	0.657	0.840	0.657	0.767	0.350	0.812	0.350	0.697
7	0.592	0.815	0.592	0.733	0.313	0.792	0.313	0.697

Table D.21.: Raw results of PatMinr using Song Pattern Discovery on the Irish Folk Tunes Dataset.

D.7. Class Pattern Discovery on the Irish Folk Tunes Dataset

D.7.1. SIACTTEC

Parameters		GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
Min. patt. size	Min. compactness	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.1	3	0.997	0.958	0.997	0.921	0.997	0.955	0.997	0.921
0.1	4	0.996	0.958	0.996	0.921	0.995	0.955	0.995	0.921
0.1	5	0.995	0.958	0.995	0.919	0.994	0.955	0.994	0.919
0.1	6	0.992	0.955	0.992	0.916	0.991	0.952	0.991	0.916
0.1	7	0.984	0.947	0.984	0.899	0.984	0.944	0.984	0.899
0.3	3	0.996	0.958	0.996	0.919	0.993	0.955	0.993	0.919
0.3	4	0.994	0.955	0.994	0.916	0.991	0.952	0.991	0.916
0.3	5	0.991	0.955	0.991	0.910	0.987	0.952	0.987	0.907
0.3	6	0.985	0.949	0.985	0.907	0.980	0.947	0.980	0.907
0.3	7	0.975	0.949	0.975	0.902	0.969	0.947	0.969	0.899
0.5	3	0.995	0.958	0.995	0.921	0.978	0.952	0.978	0.919
0.5	4	0.991	0.955	0.991	0.916	0.972	0.955	0.972	0.913
0.5	5	0.980	0.949	0.980	0.896	0.960	0.952	0.960	0.893
0.5	6	0.964	0.944	0.964	0.885	0.943	0.941	0.943	0.885
0.5	7	0.941	0.938	0.941	0.888	0.918	0.938	0.918	0.888
0.7	3	0.992	0.958	0.992	0.910	0.942	0.949	0.942	0.907
0.7	4	0.974	0.952	0.974	0.902	0.926	0.938	0.926	0.899
0.7	5	0.949	0.949	0.949	0.876	0.902	0.941	0.902	0.885
0.7	6	0.912	0.947	0.912	0.896	0.868	0.941	0.868	0.902
0.7	7	0.883	0.941	0.883	0.890	0.842	0.944	0.842	0.902
0.9	3	0.987	0.952	0.987	0.907	0.882	0.947	0.882	0.899
0.9	4	0.950	0.949	0.950	0.899	0.851	0.947	0.851	0.890
0.9	5	0.889	0.949	0.889	0.896	0.801	0.941	0.801	0.896
0.9	6	0.822	0.944	0.822	0.874	0.749	0.938	0.749	0.871
0.9	7	0.779	0.955	0.779	0.896	0.714	0.944	0.714	0.882
1.0	3	0.986	0.952	0.986	0.907	0.864	0.947	0.864	0.893
1.0	4	0.949	0.952	0.949	0.899	0.832	0.938	0.832	0.879
1.0	5	0.884	0.941	0.884	0.879	0.779	0.927	0.779	0.888
1.0	6	0.814	0.941	0.814	0.874	0.725	0.938	0.725	0.868
1.0	7	0.768	0.955	0.768	0.885	0.687	0.941	0.687	0.885

Table D.22.: Raw results of SIACTTEC using Class Pattern Discovery on the Irish Folk Tunes Dataset.

D.7.2. COSIACTTEC

Parameters		GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
Min. patt. size	Min. compactness	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.1	3	0.766	0.882	0.766	0.761	0.506	0.854	0.506	0.691
0.1	4	0.708	0.885	0.708	0.770	0.478	0.831	0.478	0.705
0.1	5	0.637	0.879	0.637	0.801	0.444	0.854	0.444	0.730
0.1	6	0.573	0.865	0.573	0.728	0.403	0.840	0.403	0.691
0.1	7	0.524	0.840	0.524	0.744	0.378	0.834	0.378	0.750
0.3	3	0.828	0.921	0.828	0.834	0.569	0.890	0.569	0.809
0.3	4	0.767	0.927	0.767	0.756	0.540	0.893	0.540	0.744
0.3	5	0.710	0.921	0.710	0.820	0.508	0.910	0.508	0.761
0.3	6	0.648	0.907	0.648	0.834	0.468	0.916	0.468	0.809
0.3	7	0.625	0.916	0.625	0.846	0.462	0.921	0.462	0.831
0.5	3	0.849	0.924	0.849	0.829	0.582	0.890	0.582	0.784
0.5	4	0.771	0.921	0.771	0.812	0.544	0.899	0.544	0.775
0.5	5	0.726	0.933	0.726	0.848	0.521	0.910	0.521	0.829
0.5	6	0.685	0.927	0.685	0.862	0.499	0.916	0.499	0.831
0.5	7	0.651	0.941	0.651	0.862	0.475	0.930	0.475	0.854

(COSIACTTEC, continued from previous page)

Parameters		GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
Min. patt. size	Min. compactness	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.7	3	0.828	0.919	0.828	0.868	0.536	0.899	0.536	0.840
0.7	4	0.751	0.935	0.751	0.846	0.499	0.913	0.499	0.820
0.7	5	0.700	0.941	0.700	0.857	0.481	0.919	0.481	0.851
0.7	6	0.661	0.947	0.661	0.865	0.463	0.927	0.463	0.854
0.7	7	0.623	0.941	0.623	0.885	0.448	0.924	0.448	0.879
0.9	3	0.829	0.941	0.829	0.885	0.510	0.893	0.510	0.820
0.9	4	0.755	0.916	0.755	0.879	0.481	0.904	0.481	0.798
0.9	5	0.679	0.935	0.679	0.862	0.444	0.919	0.444	0.848
0.9	6	0.633	0.933	0.633	0.871	0.431	0.927	0.431	0.860
0.9	7	0.594	0.927	0.594	0.854	0.417	0.907	0.417	0.843
1.0	3	0.830	0.949	0.830	0.876	0.512	0.916	0.512	0.879
1.0	4	0.750	0.930	0.750	0.854	0.476	0.921	0.476	0.809
1.0	5	0.674	0.933	0.674	0.837	0.439	0.927	0.439	0.831
1.0	6	0.627	0.938	0.627	0.868	0.424	0.930	0.424	0.857
1.0	7	0.587	0.930	0.587	0.862	0.409	0.919	0.409	0.848

Table D.23.: Raw results of COSIACTTEC using Class Pattern Discovery on the Irish Folk Tunes Dataset.

D.7.3. MotivesExtractor

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
2	0.7	3	0	0.135	0.472	0.135	0.427	0.131	0.472	0.131	0.430
3	0.7	3	0	0.190	0.567	0.190	0.469	0.184	0.567	0.184	0.461
4	0.7	3	0	0.055	0.225	0.055	0.225	0.053	0.225	0.053	0.222
2	0.7	3	2	0.135	0.469	0.135	0.419	0.127	0.475	0.127	0.427
3	0.7	3	2	0.189	0.559	0.189	0.458	0.175	0.553	0.175	0.449
4	0.7	3	2	0.056	0.222	0.056	0.216	0.050	0.233	0.050	0.216
2	0.7	3	4	0.134	0.461	0.134	0.416	0.121	0.463	0.121	0.404
3	0.7	3	4	0.187	0.556	0.187	0.458	0.168	0.548	0.168	0.449
4	0.7	3	4	0.056	0.222	0.056	0.216	0.048	0.236	0.048	0.216
2	0.7	3	6	0.132	0.452	0.132	0.407	0.118	0.455	0.118	0.402
3	0.7	3	6	0.184	0.551	0.184	0.438	0.159	0.534	0.159	0.452
4	0.7	3	6	0.056	0.222	0.056	0.216	0.048	0.236	0.048	0.211
2	0.7	3	8	0.131	0.449	0.131	0.416	0.113	0.461	0.113	0.399
3	0.7	3	8	0.180	0.545	0.180	0.438	0.151	0.534	0.151	0.438
4	0.7	3	8	0.056	0.225	0.056	0.213	0.047	0.236	0.047	0.205
2	0.7	4	0	0.091	0.362	0.091	0.343	0.089	0.362	0.089	0.343
3	0.7	4	0	0.112	0.427	0.112	0.374	0.109	0.430	0.109	0.371
4	0.7	4	0	0.055	0.225	0.055	0.188	0.053	0.222	0.053	0.185
2	0.7	4	2	0.090	0.365	0.090	0.340	0.086	0.371	0.086	0.337
3	0.7	4	2	0.112	0.424	0.112	0.376	0.105	0.424	0.105	0.368
4	0.7	4	2	0.054	0.205	0.054	0.191	0.049	0.199	0.049	0.183
2	0.7	4	4	0.090	0.365	0.090	0.340	0.085	0.371	0.085	0.337
3	0.7	4	4	0.110	0.413	0.110	0.362	0.102	0.416	0.102	0.346
4	0.7	4	4	0.054	0.213	0.054	0.194	0.049	0.202	0.049	0.183
2	0.7	4	6	0.090	0.365	0.090	0.337	0.084	0.368	0.084	0.326
3	0.7	4	6	0.110	0.416	0.110	0.365	0.100	0.416	0.100	0.343
4	0.7	4	6	0.054	0.213	0.054	0.194	0.047	0.205	0.047	0.185
2	0.7	4	8	0.089	0.365	0.089	0.334	0.080	0.362	0.080	0.320
3	0.7	4	8	0.109	0.402	0.109	0.371	0.096	0.416	0.096	0.348
4	0.7	4	8	0.053	0.211	0.053	0.194	0.046	0.202	0.046	0.185
2	0.7	5	0	0.142	0.435	0.142	0.396	0.141	0.435	0.141	0.396
3	0.7	5	0	0.145	0.449	0.145	0.385	0.143	0.449	0.143	0.385
4	0.7	5	0	0.071	0.256	0.071	0.225	0.070	0.258	0.070	0.228

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
2	0.7	5	2	0.141	0.430	0.141	0.390	0.137	0.430	0.137	0.390
3	0.7	5	2	0.144	0.447	0.144	0.382	0.137	0.444	0.137	0.385
4	0.7	5	2	0.071	0.247	0.071	0.225	0.066	0.250	0.066	0.222
2	0.7	5	4	0.142	0.419	0.142	0.390	0.135	0.424	0.135	0.388
3	0.7	5	4	0.144	0.449	0.144	0.379	0.134	0.444	0.134	0.376
4	0.7	5	4	0.072	0.247	0.072	0.222	0.066	0.250	0.066	0.222
2	0.7	5	6	0.141	0.424	0.141	0.376	0.131	0.433	0.131	0.374
3	0.7	5	6	0.144	0.449	0.144	0.371	0.132	0.438	0.132	0.376
4	0.7	5	6	0.072	0.244	0.072	0.216	0.065	0.247	0.065	0.219
2	0.7	5	8	0.139	0.424	0.139	0.374	0.125	0.430	0.125	0.376
3	0.7	5	8	0.143	0.441	0.143	0.371	0.128	0.438	0.128	0.385
4	0.7	5	8	0.072	0.247	0.072	0.208	0.063	0.250	0.063	0.213
2	0.7	6	0	0.165	0.466	0.165	0.427	0.163	0.466	0.163	0.427
3	0.7	6	0	0.146	0.430	0.146	0.393	0.145	0.430	0.145	0.399
4	0.7	6	0	0.058	0.216	0.058	0.208	0.057	0.219	0.057	0.208
2	0.7	6	2	0.165	0.469	0.165	0.427	0.160	0.466	0.160	0.427
3	0.7	6	2	0.146	0.427	0.146	0.390	0.140	0.424	0.140	0.399
4	0.7	6	2	0.058	0.211	0.058	0.205	0.053	0.211	0.053	0.205
2	0.7	6	4	0.165	0.461	0.165	0.424	0.155	0.458	0.155	0.419
3	0.7	6	4	0.147	0.430	0.147	0.385	0.138	0.430	0.138	0.393
4	0.7	6	4	0.058	0.208	0.058	0.208	0.052	0.211	0.052	0.211
2	0.7	6	6	0.165	0.458	0.165	0.421	0.154	0.458	0.154	0.416
3	0.7	6	6	0.147	0.430	0.147	0.388	0.136	0.430	0.136	0.393
4	0.7	6	6	0.058	0.211	0.058	0.208	0.051	0.208	0.051	0.208
2	0.7	6	8	0.162	0.455	0.162	0.413	0.148	0.447	0.148	0.416
3	0.7	6	8	0.146	0.430	0.146	0.390	0.133	0.427	0.133	0.390
4	0.7	6	8	0.057	0.211	0.057	0.208	0.049	0.216	0.049	0.202
2	0.7	7	0	0.161	0.458	0.161	0.430	0.160	0.458	0.160	0.430
3	0.7	7	0	0.166	0.472	0.166	0.404	0.165	0.472	0.165	0.404
4	0.7	7	0	0.115	0.346	0.115	0.303	0.114	0.348	0.114	0.301
2	0.7	7	2	0.162	0.458	0.162	0.427	0.157	0.458	0.157	0.435
3	0.7	7	2	0.167	0.475	0.167	0.399	0.161	0.469	0.161	0.399
4	0.7	7	2	0.118	0.346	0.118	0.303	0.112	0.357	0.112	0.303
2	0.7	7	4	0.162	0.458	0.162	0.427	0.156	0.458	0.156	0.433
3	0.7	7	4	0.168	0.480	0.168	0.396	0.157	0.466	0.157	0.399
4	0.7	7	4	0.118	0.346	0.118	0.309	0.111	0.357	0.111	0.303
2	0.7	7	6	0.160	0.463	0.160	0.424	0.151	0.461	0.151	0.433
3	0.7	7	6	0.169	0.475	0.169	0.402	0.158	0.469	0.158	0.404
4	0.7	7	6	0.117	0.346	0.117	0.306	0.110	0.351	0.110	0.303
2	0.7	7	8	0.159	0.458	0.159	0.421	0.148	0.461	0.148	0.419
3	0.7	7	8	0.168	0.472	0.168	0.399	0.156	0.463	0.156	0.396
4	0.7	7	8	0.117	0.346	0.117	0.309	0.109	0.357	0.109	0.303
2	0.5	3	0	0.135	0.472	0.135	0.427	0.131	0.472	0.131	0.430
3	0.5	3	0	0.959	0.949	0.959	0.890	0.957	0.952	0.957	0.890
4	0.5	3	0	0.858	0.834	0.858	0.792	0.851	0.834	0.851	0.792
2	0.5	3	2	0.135	0.469	0.135	0.419	0.127	0.475	0.127	0.427
3	0.5	3	2	0.959	0.947	0.959	0.890	0.947	0.949	0.947	0.890
4	0.5	3	2	0.859	0.834	0.859	0.789	0.831	0.834	0.831	0.781
2	0.5	3	4	0.134	0.461	0.134	0.416	0.121	0.463	0.121	0.404
3	0.5	3	4	0.958	0.947	0.958	0.885	0.937	0.947	0.937	0.882
4	0.5	3	4	0.858	0.831	0.858	0.787	0.816	0.826	0.816	0.770
2	0.5	3	6	0.132	0.452	0.132	0.407	0.118	0.455	0.118	0.402
3	0.5	3	6	0.955	0.944	0.955	0.885	0.921	0.947	0.921	0.874
4	0.5	3	6	0.857	0.834	0.857	0.787	0.798	0.817	0.798	0.758
2	0.5	3	8	0.131	0.449	0.131	0.416	0.113	0.461	0.113	0.399
3	0.5	3	8	0.953	0.952	0.953	0.882	0.894	0.938	0.894	0.868
4	0.5	3	8	0.854	0.831	0.854	0.787	0.775	0.829	0.775	0.728
2	0.5	4	0	0.091	0.362	0.091	0.343	0.089	0.362	0.089	0.343
3	0.5	4	0	0.903	0.935	0.903	0.874	0.898	0.938	0.898	0.874
4	0.5	4	0	0.847	0.831	0.847	0.787	0.840	0.829	0.840	0.787

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
2	0.5	4	2	0.090	0.365	0.090	0.340	0.086	0.371	0.086	0.337
3	0.5	4	2	0.904	0.944	0.904	0.876	0.887	0.949	0.887	0.871
4	0.5	4	2	0.848	0.831	0.848	0.787	0.819	0.829	0.819	0.784
2	0.5	4	4	0.090	0.365	0.090	0.340	0.085	0.371	0.085	0.337
3	0.5	4	4	0.905	0.938	0.905	0.874	0.878	0.949	0.878	0.876
4	0.5	4	4	0.846	0.829	0.846	0.792	0.798	0.823	0.798	0.761
2	0.5	4	6	0.090	0.365	0.090	0.337	0.084	0.368	0.084	0.326
3	0.5	4	6	0.898	0.935	0.898	0.871	0.856	0.933	0.856	0.851
4	0.5	4	6	0.844	0.829	0.844	0.784	0.785	0.831	0.785	0.744
2	0.5	4	8	0.089	0.365	0.089	0.334	0.080	0.362	0.080	0.320
3	0.5	4	8	0.899	0.933	0.899	0.854	0.842	0.941	0.842	0.848
4	0.5	4	8	0.842	0.826	0.842	0.784	0.765	0.820	0.765	0.691
2	0.5	5	0	0.142	0.435	0.142	0.396	0.141	0.435	0.141	0.396
3	0.5	5	0	0.824	0.913	0.824	0.829	0.821	0.913	0.821	0.831
4	0.5	5	0	0.811	0.803	0.811	0.753	0.807	0.803	0.807	0.747
2	0.5	5	2	0.141	0.430	0.141	0.390	0.137	0.430	0.137	0.390
3	0.5	5	2	0.830	0.916	0.830	0.834	0.808	0.916	0.808	0.820
4	0.5	5	2	0.811	0.803	0.811	0.750	0.792	0.809	0.792	0.742
2	0.5	5	4	0.142	0.419	0.142	0.390	0.135	0.424	0.135	0.388
3	0.5	5	4	0.829	0.919	0.829	0.831	0.800	0.913	0.800	0.829
4	0.5	5	4	0.807	0.801	0.807	0.753	0.772	0.809	0.772	0.722
2	0.5	5	6	0.141	0.424	0.141	0.376	0.131	0.433	0.131	0.374
3	0.5	5	6	0.828	0.919	0.828	0.823	0.787	0.913	0.787	0.792
4	0.5	5	6	0.805	0.806	0.805	0.744	0.759	0.809	0.759	0.728
2	0.5	5	8	0.139	0.424	0.139	0.374	0.125	0.430	0.125	0.376
3	0.5	5	8	0.827	0.921	0.827	0.837	0.772	0.924	0.772	0.803
4	0.5	5	8	0.805	0.806	0.805	0.753	0.749	0.809	0.749	0.711
2	0.5	6	0	0.165	0.466	0.165	0.427	0.163	0.466	0.163	0.427
3	0.5	6	0	0.747	0.907	0.747	0.787	0.745	0.907	0.745	0.787
4	0.5	6	0	0.775	0.795	0.775	0.739	0.770	0.795	0.770	0.733
2	0.5	6	2	0.165	0.469	0.165	0.427	0.160	0.466	0.160	0.427
3	0.5	6	2	0.751	0.907	0.751	0.784	0.734	0.910	0.734	0.778
4	0.5	6	2	0.776	0.795	0.776	0.739	0.747	0.798	0.747	0.719
2	0.5	6	4	0.165	0.461	0.165	0.424	0.155	0.458	0.155	0.419
3	0.5	6	4	0.754	0.913	0.754	0.798	0.727	0.913	0.727	0.789
4	0.5	6	4	0.774	0.795	0.774	0.744	0.730	0.789	0.730	0.708
2	0.5	6	6	0.165	0.458	0.165	0.421	0.154	0.458	0.154	0.416
3	0.5	6	6	0.751	0.913	0.751	0.784	0.716	0.919	0.716	0.781
4	0.5	6	6	0.772	0.792	0.772	0.744	0.717	0.787	0.717	0.702
2	0.5	6	8	0.162	0.455	0.162	0.413	0.148	0.447	0.148	0.416
3	0.5	6	8	0.749	0.896	0.749	0.781	0.702	0.902	0.702	0.775
4	0.5	6	8	0.773	0.792	0.773	0.744	0.708	0.787	0.708	0.674
2	0.5	7	0	0.161	0.458	0.161	0.430	0.160	0.458	0.160	0.430
3	0.5	7	0	0.666	0.876	0.666	0.801	0.664	0.876	0.664	0.798
4	0.5	7	0	0.748	0.789	0.748	0.739	0.744	0.792	0.744	0.739
2	0.5	7	2	0.162	0.458	0.162	0.427	0.157	0.458	0.157	0.435
3	0.5	7	2	0.667	0.857	0.667	0.798	0.656	0.865	0.656	0.792
4	0.5	7	2	0.750	0.789	0.750	0.739	0.727	0.795	0.727	0.725
2	0.5	7	4	0.162	0.458	0.162	0.427	0.156	0.458	0.156	0.433
3	0.5	7	4	0.667	0.860	0.667	0.787	0.647	0.868	0.647	0.778
4	0.5	7	4	0.747	0.784	0.747	0.733	0.712	0.775	0.712	0.719
2	0.5	7	6	0.160	0.463	0.160	0.424	0.151	0.461	0.151	0.433
3	0.5	7	6	0.668	0.857	0.668	0.775	0.646	0.862	0.646	0.764
4	0.5	7	6	0.748	0.789	0.748	0.733	0.702	0.798	0.702	0.719
2	0.5	7	8	0.159	0.458	0.159	0.421	0.148	0.461	0.148	0.419
3	0.5	7	8	0.669	0.862	0.669	0.772	0.634	0.865	0.634	0.758
4	0.5	7	8	0.743	0.792	0.743	0.733	0.692	0.787	0.692	0.711
2	0.3	3	0	0.978	0.952	0.978	0.910	0.978	0.952	0.978	0.910
3	0.3	3	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	3	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**	**Cov.**	**Acc.**
2	0.3	3	2	0.978	0.952	0.978	0.904	0.977	0.952	0.977	0.904
3	0.3	3	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	3	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	3	4	0.978	0.952	0.978	0.904	0.974	0.952	0.974	0.896
3	0.3	3	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	3	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	3	6	0.976	0.949	0.976	0.904	0.966	0.949	0.966	0.896
3	0.3	3	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	3	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	3	8	0.977	0.949	0.977	0.904	0.962	0.952	0.962	0.890
3	0.3	3	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	3	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	4	0	0.977	0.952	0.977	0.910	0.977	0.952	0.977	0.910
3	0.3	4	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	4	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	4	2	0.977	0.952	0.977	0.902	0.975	0.952	0.975	0.902
3	0.3	4	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	4	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	4	4	0.975	0.952	0.975	0.904	0.971	0.952	0.971	0.904
3	0.3	4	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	4	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	4	6	0.975	0.952	0.975	0.899	0.965	0.947	0.965	0.885
3	0.3	4	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	4	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	4	8	0.973	0.952	0.973	0.899	0.955	0.958	0.955	0.882
3	0.3	4	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	4	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	5	0	0.977	0.952	0.977	0.910	0.977	0.952	0.977	0.910
3	0.3	5	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	5	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	5	2	0.975	0.955	0.975	0.899	0.973	0.955	0.973	0.899
3	0.3	5	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	5	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	5	4	0.974	0.955	0.974	0.902	0.968	0.952	0.968	0.899
3	0.3	5	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	5	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	5	6	0.973	0.952	0.973	0.899	0.962	0.949	0.962	0.893
3	0.3	5	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	5	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	5	8	0.972	0.952	0.972	0.893	0.952	0.952	0.952	0.890
3	0.3	5	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	5	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	6	0	0.972	0.949	0.972	0.896	0.972	0.949	0.972	0.896
3	0.3	6	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	6	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	6	2	0.971	0.952	0.971	0.890	0.969	0.944	0.969	0.888
3	0.3	6	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	6	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	6	4	0.969	0.952	0.969	0.890	0.964	0.949	0.964	0.888
3	0.3	6	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	6	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	6	6	0.967	0.949	0.967	0.893	0.958	0.949	0.958	0.890
3	0.3	6	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	6	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	6	8	0.963	0.952	0.963	0.893	0.946	0.952	0.946	0.882
3	0.3	6	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	6	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	7	0	0.962	0.947	0.962	0.899	0.961	0.947	0.961	0.899
3	0.3	7	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	7	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

(MotivesExtractor, continued from previous page)

Parameters				GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
ρ	θ	ν	Θ	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
2	0.3	7	2	0.962	0.952	0.962	0.896	0.959	0.952	0.959	0.896
3	0.3	7	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	7	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	7	4	0.959	0.952	0.959	0.899	0.952	0.955	0.952	0.896
3	0.3	7	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	7	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	7	6	0.958	0.952	0.958	0.896	0.944	0.958	0.944	0.893
3	0.3	7	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	7	6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.3	7	8	0.956	0.955	0.956	0.890	0.940	0.958	0.940	0.888
3	0.3	7	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.3	7	8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	3	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
3	0.1	3	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	3	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	3	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
3	0.1	3	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.1	3	2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.1	3	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
3	0.1	3	4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table D.24.: Raw results of MotivesExtractor using Class Pattern Discovery on the Irish Folk Tunes Dataset. The parameters are the path tracing tolerance (ρ), the scoring threshold (θ), the minimum pattern length (ν) and the tolerance parameter for the pattern clustering (Θ).

D.7.4. MGDP

Parameters					GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
$p(P\|\ominus)$	$I(P)$	n	Min. patt.	Enforce	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.9	10	3	5	true	0.595	0.888	0.595	0.817	0.342	0.854	0.342	0.688
0.9	10	3	10	true	0.688	0.907	0.688	0.854	0.385	0.871	0.385	0.736
0.9	20	3	5	true	0.510	0.882	0.510	0.795	0.268	0.812	0.268	0.685
0.9	20	3	10	true	0.613	0.904	0.613	0.837	0.315	0.846	0.315	0.728
0.9	30	3	5	true	0.471	0.885	0.471	0.815	0.240	0.831	0.240	0.722
0.9	30	3	10	true	0.584	0.893	0.584	0.840	0.290	0.854	0.290	0.764
0.9	40	3	5	true	0.456	0.876	0.456	0.817	0.230	0.820	0.230	0.719
0.9	40	3	10	true	0.569	0.885	0.569	0.843	0.281	0.846	0.281	0.761
0.7	10	3	5	true	0.741	0.919	0.741	0.798	0.560	0.904	0.560	0.761
0.7	10	3	10	true	0.776	0.927	0.776	0.843	0.583	0.930	0.583	0.803
0.7	20	3	5	true	0.621	0.888	0.621	0.730	0.451	0.879	0.451	0.719
0.7	20	3	10	true	0.656	0.899	0.656	0.781	0.474	0.904	0.474	0.761
0.7	30	3	5	true	0.559	0.874	0.559	0.764	0.404	0.862	0.404	0.764
0.7	30	3	10	true	0.604	0.896	0.604	0.775	0.434	0.902	0.434	0.758
0.7	40	3	5	true	0.521	0.860	0.521	0.767	0.374	0.857	0.374	0.756
0.7	40	3	10	true	0.568	0.893	0.568	0.772	0.405	0.885	0.405	0.770
0.9	10	4	5	true	0.597	0.904	0.597	0.770	0.346	0.837	0.346	0.736
0.9	10	4	10	true	0.639	0.899	0.639	0.784	0.368	0.851	0.368	0.736
0.9	20	4	5	true	0.512	0.899	0.512	0.764	0.272	0.820	0.272	0.742
0.9	20	4	10	true	0.563	0.893	0.563	0.781	0.299	0.837	0.299	0.744
0.9	30	4	5	true	0.475	0.885	0.475	0.806	0.246	0.831	0.246	0.787
0.9	30	4	10	true	0.537	0.885	0.537	0.820	0.276	0.851	0.276	0.787
0.9	40	4	5	true	0.459	0.882	0.459	0.806	0.236	0.829	0.236	0.781
0.9	40	4	10	true	0.520	0.882	0.520	0.823	0.266	0.848	0.266	0.778
0.7	10	4	5	true	0.728	0.924	0.728	0.831	0.559	0.910	0.559	0.781
0.7	10	4	10	true	0.756	0.927	0.756	0.846	0.578	0.927	0.578	0.795

(MGDP, continued from previous page)

Parameters					GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
$p(P\|\ominus)$	$I(P)$	n	Min. patt.	Enforce	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.7	20	4	5	true	0.614	0.896	0.614	0.758	0.452	0.876	0.452	0.736
0.7	20	4	10	true	0.642	0.899	0.642	0.775	0.471	0.893	0.471	0.764
0.7	30	4	5	true	0.550	0.876	0.550	0.787	0.404	0.871	0.404	0.761
0.7	30	4	10	true	0.593	0.888	0.593	0.764	0.433	0.899	0.433	0.761
0.7	40	4	5	true	0.510	0.862	0.510	0.787	0.374	0.854	0.374	0.761
0.7	40	4	10	true	0.556	0.890	0.556	0.775	0.405	0.888	0.405	0.778
0.9	10	5	5	true	0.523	0.843	0.523	0.699	0.325	0.798	0.325	0.612
0.9	10	5	10	true	0.606	0.829	0.606	0.719	0.369	0.817	0.369	0.640
0.9	20	5	5	true	0.452	0.843	0.452	0.654	0.258	0.789	0.258	0.607
0.9	20	5	10	true	0.540	0.834	0.540	0.677	0.303	0.812	0.303	0.632
0.9	30	5	5	true	0.417	0.840	0.417	0.677	0.231	0.784	0.231	0.638
0.9	30	5	10	true	0.515	0.829	0.515	0.697	0.280	0.820	0.280	0.654
0.9	40	5	5	true	0.402	0.840	0.402	0.674	0.222	0.778	0.222	0.646
0.9	40	5	10	true	0.500	0.823	0.500	0.697	0.271	0.815	0.271	0.660
0.7	10	5	5	true	0.692	0.913	0.692	0.809	0.540	0.916	0.540	0.772
0.7	10	5	10	true	0.722	0.933	0.722	0.823	0.559	0.919	0.559	0.781
0.7	20	5	5	true	0.580	0.885	0.580	0.761	0.435	0.874	0.435	0.742
0.7	20	5	10	true	0.616	0.907	0.616	0.775	0.458	0.879	0.458	0.770
0.7	30	5	5	true	0.525	0.862	0.525	0.787	0.391	0.871	0.391	0.756
0.7	30	5	10	true	0.571	0.890	0.571	0.761	0.421	0.874	0.421	0.753
0.7	40	5	5	true	0.488	0.848	0.488	0.792	0.362	0.857	0.362	0.764
0.7	40	5	10	true	0.536	0.888	0.536	0.778	0.393	0.868	0.393	0.772
0.9	10	6	5	true	0.474	0.747	0.474	0.610	0.304	0.691	0.304	0.604
0.9	10	6	10	true	0.555	0.736	0.555	0.688	0.343	0.708	0.343	0.635
0.9	20	6	5	true	0.414	0.744	0.414	0.632	0.246	0.688	0.246	0.626
0.9	20	6	10	true	0.496	0.739	0.496	0.671	0.286	0.699	0.286	0.629
0.9	30	6	5	true	0.384	0.742	0.384	0.652	0.221	0.677	0.221	0.640
0.9	30	6	10	true	0.472	0.730	0.472	0.685	0.264	0.702	0.264	0.654
0.9	40	6	5	true	0.371	0.739	0.371	0.654	0.213	0.674	0.213	0.638
0.9	40	6	10	true	0.459	0.730	0.459	0.688	0.255	0.699	0.255	0.654
0.7	10	6	5	true	0.659	0.902	0.659	0.708	0.517	0.919	0.517	0.666
0.7	10	6	10	true	0.693	0.913	0.693	0.761	0.537	0.919	0.537	0.671
0.7	20	6	5	true	0.556	0.871	0.556	0.683	0.420	0.882	0.420	0.674
0.7	20	6	10	true	0.590	0.885	0.590	0.725	0.441	0.879	0.441	0.711
0.7	30	6	5	true	0.504	0.854	0.504	0.761	0.376	0.876	0.376	0.719
0.7	30	6	10	true	0.548	0.885	0.548	0.753	0.404	0.879	0.404	0.719
0.7	40	6	5	true	0.467	0.834	0.467	0.761	0.348	0.871	0.348	0.716
0.7	40	6	10	true	0.515	0.868	0.515	0.753	0.381	0.879	0.381	0.725
0.9	10	7	5	true	0.471	0.728	0.471	0.643	0.299	0.699	0.299	0.587
0.9	10	7	10	true	0.538	0.733	0.538	0.680	0.333	0.716	0.333	0.610
0.9	20	7	5	true	0.414	0.728	0.414	0.629	0.242	0.691	0.242	0.584
0.9	20	7	10	true	0.482	0.744	0.482	0.663	0.276	0.716	0.276	0.610
0.9	30	7	5	true	0.381	0.728	0.381	0.635	0.214	0.685	0.214	0.598
0.9	30	7	10	true	0.456	0.736	0.456	0.674	0.252	0.722	0.252	0.624
0.9	40	7	5	true	0.372	0.725	0.372	0.629	0.208	0.683	0.208	0.598
0.9	40	7	10	true	0.447	0.730	0.447	0.671	0.246	0.719	0.246	0.621
0.7	10	7	5	true	0.620	0.837	0.620	0.708	0.488	0.857	0.488	0.691
0.7	10	7	10	true	0.645	0.848	0.645	0.674	0.503	0.860	0.503	0.691
0.7	20	7	5	true	0.521	0.823	0.521	0.685	0.396	0.826	0.396	0.685
0.7	20	7	10	true	0.547	0.843	0.547	0.674	0.412	0.829	0.412	0.708
0.7	30	7	5	true	0.473	0.809	0.473	0.691	0.356	0.817	0.356	0.677
0.7	30	7	10	true	0.517	0.840	0.517	0.688	0.384	0.829	0.384	0.713
0.7	40	7	5	true	0.447	0.789	0.447	0.728	0.333	0.806	0.333	0.694
0.7	40	7	10	true	0.492	0.831	0.492	0.699	0.362	0.829	0.362	0.730
0.9	10	3	0	false	0.387	0.643	0.387	0.565	0.256	0.638	0.256	0.570
0.9	20	3	0	false	0.262	0.506	0.262	0.461	0.165	0.506	0.165	0.455
0.9	30	3	0	false	0.199	0.472	0.199	0.447	0.125	0.472	0.125	0.435
0.9	40	3	0	false	0.160	0.404	0.160	0.396	0.105	0.404	0.105	0.390
0.7	10	3	0	false	0.717	0.924	0.717	0.784	0.546	0.910	0.546	0.761
0.7	20	3	0	false	0.542	0.728	0.542	0.626	0.409	0.728	0.409	0.612

(MGDP, continued from previous page)

Parameters					GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
$p(P\|\ominus)$	$I(P)$	n	Min. patt.	Enforce	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
0.7	30	3	0	false	0.472	0.674	0.472	0.612	0.356	0.677	0.356	0.604
0.7	40	3	0	false	0.428	0.671	0.428	0.601	0.323	0.660	0.323	0.598
0.9	10	4	0	false	0.377	0.618	0.377	0.548	0.251	0.607	0.251	0.537
0.9	20	4	0	false	0.256	0.478	0.256	0.435	0.162	0.483	0.162	0.433
0.9	30	4	0	false	0.195	0.447	0.195	0.421	0.124	0.444	0.124	0.410
0.9	40	4	0	false	0.152	0.374	0.152	0.365	0.102	0.374	0.102	0.360
0.7	10	4	0	false	0.699	0.919	0.699	0.739	0.538	0.896	0.538	0.722
0.7	20	4	0	false	0.537	0.725	0.537	0.626	0.406	0.722	0.406	0.607
0.7	30	4	0	false	0.463	0.669	0.463	0.610	0.352	0.674	0.352	0.596
0.7	40	4	0	false	0.418	0.666	0.418	0.596	0.319	0.660	0.319	0.587
0.9	10	5	0	false	0.351	0.559	0.351	0.506	0.241	0.545	0.241	0.480
0.9	20	5	0	false	0.250	0.452	0.250	0.427	0.161	0.461	0.161	0.421
0.9	30	5	0	false	0.192	0.424	0.192	0.399	0.123	0.419	0.123	0.388
0.9	40	5	0	false	0.153	0.374	0.153	0.365	0.103	0.371	0.103	0.360
0.7	10	5	0	false	0.668	0.907	0.668	0.770	0.522	0.899	0.522	0.730
0.7	20	5	0	false	0.516	0.722	0.516	0.626	0.396	0.713	0.396	0.604
0.7	30	5	0	false	0.448	0.663	0.448	0.593	0.344	0.671	0.344	0.570
0.7	40	5	0	false	0.404	0.657	0.404	0.590	0.311	0.646	0.311	0.576
0.9	10	6	0	false	0.315	0.556	0.315	0.469	0.224	0.545	0.224	0.447
0.9	20	6	0	false	0.221	0.419	0.221	0.396	0.149	0.416	0.149	0.388
0.9	30	6	0	false	0.173	0.376	0.173	0.368	0.114	0.382	0.114	0.360
0.9	40	6	0	false	0.149	0.376	0.149	0.365	0.100	0.374	0.100	0.360
0.7	10	6	0	false	0.627	0.899	0.627	0.716	0.498	0.890	0.498	0.669
0.7	20	6	0	false	0.494	0.694	0.494	0.596	0.386	0.691	0.386	0.596
0.7	30	6	0	false	0.433	0.640	0.433	0.573	0.336	0.640	0.336	0.567
0.7	40	6	0	false	0.390	0.635	0.390	0.553	0.305	0.635	0.305	0.545
0.9	10	7	0	false	0.282	0.517	0.282	0.463	0.206	0.508	0.206	0.430
0.9	20	7	0	false	0.197	0.354	0.197	0.320	0.134	0.348	0.134	0.317
0.9	30	7	0	false	0.154	0.343	0.154	0.334	0.102	0.340	0.102	0.326
0.9	40	7	0	false	0.138	0.343	0.138	0.331	0.092	0.340	0.092	0.326
0.7	10	7	0	false	0.576	0.772	0.576	0.657	0.463	0.775	0.463	0.649
0.7	20	7	0	false	0.460	0.626	0.460	0.579	0.361	0.621	0.361	0.587
0.7	30	7	0	false	0.399	0.607	0.399	0.559	0.314	0.615	0.314	0.562
0.7	40	7	0	false	0.364	0.593	0.364	0.553	0.287	0.593	0.287	0.542

Table D.25.: Raw results of MGDP using Class Pattern Discovery on the Irish Folk Tunes Dataset. The parameters are minimum support ($p(P|\ominus)$), minimum distinctiveness ($I(P)$), minimum pattern size (n), minimum number of patterns and enforce minimum number of patterns.

D.7.5. PatMinr

Parameters	GR-NNA		GR-MSA		FPOR-NNA		FPOR-MSA	
Min. patt. size	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.	Cov.	Acc.
3	0.990	0.952	0.990	0.921	0.914	0.958	0.914	0.919
4	0.961	0.958	0.961	0.919	0.883	0.949	0.883	0.924
5	0.894	0.963	0.894	0.904	0.824	0.955	0.824	0.904
6	0.820	0.947	0.820	0.902	0.762	0.938	0.762	0.882
7	0.757	0.938	0.757	0.865	0.709	0.938	0.709	0.840

Table D.26.: Raw results of PatMinr using Class Pattern Discovery on the Irish Folk Tunes Dataset.

Bibliography

Arzt, A., Böck, S., and Widmer, G. (2012). “Fast Identification of Piece and Score Position via Symbolic Fingerprinting.” In: *Proceedings of the 13th International Society for Music Information Retrieval Conference*. Porto, Portugal.

Ayres, J., Flannick, J., Gehrke, J., and Yiu, T. (2002). “Sequential Pattern mining using a bitmap representation.” In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '02*. New York, New York, USA: ACM Press, p. 429.

Barlow, H. and Morgenstern, S. (1948). *A dictionary of musical themes*. New York: Crown Publishers.

Bayard, S. (1950). “Prolegomena to a study of the principal melodic families of British-American folk song.” In: *Journal of American Folklore* 63.247, pp. 1–44.

Bergeron, M. and Conklin, D. (2011). “Subsumption of vertical viewpoint patterns.” In: *Mathematics and Computation in Music*. Ed. by C. Agon, M. Andreatta, G. Assayag, E. Amiot, J. Bresson, and J. Mandereau. Vol. 6726. Springer Berlin Heidelberg, pp. 1–12.

Cambouropoulos, E. (1998). “Towards a General Computational Theory of Musical Structure.” Doctoral dissertation. The University of Edinburgh, p. 184.

Collins, T. (2011). “Improved methods for pattern discovery in music, with applications in automated stylistic composition.” Doctoral dissertation. The Open University, p. 421.

Collins, T. (2014a). *Discovery of Repeated Themes & Sections*. URL: http://www.music-ir.org/mirex/wiki/2014:Discovery_of_Repeated_Themes_&_Sections (visited on 05/19/2014).

Collins, T. (2014b). “Inter-Opus Analyses of Beethoven’s Piano Sonatas.” In: *Eighth European Music Analysis Conference*. Leuven, Belgium, p. 69.

Collins, T., Arzt, A., Flossmann, S., and Widmer, G. (2013). “SIARCT-CFP: Improving precision and the discovery of inexact musical patterns in point-set representations.” In: *Proceedings of the 14th International Society for Music Information Retrieval Conference*. Curitiba, Brazil, pp. 549–554.

Collins, T., Laney, R., Willis, A., and Garthwaite, P. (2011). “Modeling Pattern Importance in Chopin’s Mazurkas.” en. In: *Music Perception: An Interdisciplinary Journal* 28.4, pp. 387–414.

Collins, T., Thurlow, J., Laney, R., Willis, A., and Garthwaite, P. (2010). “A comparative evaluation of algorithms for discovering translational patterns in Baroque keyboard works.” In: *Proceedings of the 11th International Society for Music Information Retrieval Conference*. Miami, Florida, pp. 3–8.

Conklin, D. and Witten, I. H. (1995). “Multiple viewpoint systems for music prediction.” In: *Journal of New Music Research* 24.1, pp. 51–73.

Conklin, D. (2009). “Melody classification using patterns.” In: *Proceedings of the Second International Workshop on Machine Learning and Music*. Bled, Slovenia, pp. 37–41.

Conklin, D. (2010). “Discovery of distinctive patterns in music.” In: *Intelligent Data Analysis* 14.5, pp. 547–554.

Conklin, D. and Anagnostopoulou, C. (2011). “Comparative Pattern Analysis of Cretan Folk Songs.” In: *Journal of New Music Research* 40.2, pp. 119–125.

Cover, T. and Hart, P. (1967). “Nearest neighbor pattern classification.” English. In: *IEEE Transactions on Information Theory* 13.1, pp. 21–27.

Cunningham, S. J. (2002). “User studies: a first step in designing an MIR testbed.” In: *MIR/MDL Evaluation Project White Paper Collection* 2, pp. 17–19.

Cuthbert, M. S. and Ariza, C. (2010). “music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data.” In: *Proceedings of the 11th International Society for Music Information Retrieval Conference*. Utrecht, The Netherlands, pp. 637–642.

De Boor, C. (1972). “On calculating with B-splines.” In: *Journal of Approximation Theory* 6.1, pp. 50–62.

Downie, J. S. (2004). “The Scientific Evaluation of Music Information Retrieval Systems: Foundations and Future.” In: *Computer Music Journal* 28.2, pp. 12–23.

Downie, J. S., Byrd, D., and Crawford, T. (2009). “Ten years of ISMIR: Reflections on challenges and opportunities.” In: *Proceedings of the 10th International Conference on Music Information Retrieval*. Kobe, Japan, pp. 13–18.

Good, M. (2001). “MusicXML: An internet-friendly format for sheet music.” In: *XML Conference and Expo*.

Gotoh, O. (1982). “An improved algorithm for matching biological sequences.” In: *Journal of Molecular Biology* 162.3, pp. 705–708.

Hillewaere, R., Manderick, B., Conklin, D., and Ehu, U. P. V. (2012). “String Methods for Folk Tune Genre Classification.” In: *Proceedings of the 13th International Society for Music Information Retrieval Conference*. Ismir. Curitiba, Brazil, pp. 217–222.

Huron, D. (1997). “Humdrum and Kern: Selective feature encoding.” In: *Beyond MIDI: The Handbook of Musical Codes*. Ed. by E. Selfridge-Field. Cambridge, Massachusetts: MIT Press.

Janssen, B., De Haas, W. B., Volk, A., and Van Kranenburg, P. (2013). “Discovering repeated patterns in music: state of knowledge, challenges perspectives.” In: *Proceedings of the 10th International Symposium on Computer Music Multidisciplinary Research*. Marseille, France, pp. 225–240.

Jones, G. T. (1974). *Music theory*. New York: HarperPerennial.

Karydis, I., Nanopoulos, A., and Manolopoulos, Y. (2006). “Symbolic musical genre classification based on repeating patterns.” In: *Proceedings of the 1st ACM workshop on Audio and music computing multimedia - AMCMM '06*, p. 53.

Lartillot, O. and Toiviainen, P. (2007). “Motivic matching strategies for automated pattern extraction.” In: *Musicae Scientiae* 11.1 Suppl, pp. 281–314.

Lartillot, O. (2014a). “In-depth motivic analysis based on multiparametric closed pattern and cyclic sequence mining.” In: *Proceedings of the 15th International Society for Music Information Retrieval Conference*. Taipei, Taiwan.

Lartillot, O. (2014b). “Submission to MIREX Discovery of Repeated Themes and Sections.” In: *10th Annual Music Information Retrieval eXchange (MIREX'14)*. Tapei, Taiwan.

Lerdahl, F. and Jackendorff, R. (1985). *A Generative Theory of Tonal Music*. Cambridge, Massachusetts: MIT Press, p. 368.

Li, M., Chen, X., Li, X., Ma, B., and Vitanyi, P. (2004). “The Similarity Metric.” English. In: *IEEE Transactions on Information Theory* 50.12, pp. 3250–3264.

Lin, C., Liu, N., Wu, Y., and Chen, A. (2004). "Music classification using significant repeating patterns." In: *Database Systems for Advanced Applications* 2973, pp. 506–518.

Livingstone, S. R., Palmer, C., and Schubert, E. (2012). "Emotional response to musical repetition." In: *Emotion (Washington, D.C.)* 12.3, pp. 552–67.

Lloyd, A. L. (1967). *Folk Song in England*. London: Unwin Brothers.

Manning, C. D., Raghavan, P., and Schütze, H. (2009). *An Introduction to Information Retrieval*. Cambridge, England: Cambridge University Press.

Meredith, D. (2006). "Point-set algorithms for pattern discovery and pattern matching in music." In: *Proceedings of the Dagstuhl Seminar on Content-Based Retrieval*. Dagstuhl, Germany.

Meredith, D. (2013). "COSIATEC and SIATECCompress: Pattern discovery by geometric compression." In: *Music Information Retrieval Evaluation Exchange*. Curitiba, Brazil.

Meredith, D. (2014). "Using point-set compression to classify folk songs." In: *Proceedings of the Fourth International Workshop on Folk Music Analysis*. Istanbul, Turkey, pp. 29–35.

Meredith, D. (2015). "Music Analysis and Point-Set Compression." In: *Journal of New Music Research* 44.3. Forthcoming.

Meredith, D., Lemström, K., and Wiggins, G. A. (2002). "Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music." In: *Journal of New Music Research* 31.4, pp. 321–345.

Mirka, D. (2009). *Metric Manipulations in Haydn and Mozart*. New York, New York, USA: Oxford University Press.

Müller, M. and Clausen, M. (2007). "Transposition-Invariant Self-Similarity Matrices." In: *Proceedings of the 8th International Society for Music Information Retrieval Conference*. Vienna, Austria, pp. 47–50.

Needleman, S. B. and Wunsch, C. D. (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins." In: *Journal of Molecular Biology* 48.3, pp. 443–453.

Nieto, O. and Farbood, M. (2014a). "Identifying Polyphonic Patterns From Audio Recordings Using Music Segmentation Techniques." In: *Proceedings of the 15th International Society for Music Information Retrieval Conference*. Taipei, Taiwan.

Nieto, O. and Farbood, M. (2014b). "Submission to MIREX discovery of repeated themes and sections." In: *10th Annual Music Information Retrieval eXchange (MIREX'14)*. Taipei, Taiwan.

Pohlmann, K. C. (2010). *Principles of Digital Audio*. 6th ed. Berkeley CA, USA: McGraw-Hill Professional, pp. 19–44.

Rodríguez-López, M. E. and Volk, A. (2015). "Location Constraints for Repetition-Based Segmentation of Melodies." In: *Mathematics and Computation in Music*. Ed. by T. Collins, D. Meredith, and A. Volk. Vol. 9110. Springer International Publishing, pp. 73–84.

Schenker, H. (1954). *Harmony*. Ed. by O. Jonas. Chicago: University of Chicago Press.

Selfridge-Field, E. (1997). *Beyond MIDI: the handbook of musical codes*. Cambridge, Massachusetts: MIT Press.

Uitdenbogerd, A. and Zobel, J. (1999). "Melodic matching techniques for large music databases." In: *Proceedings of the seventh ACM international conference on Multimedia*, pp. 57–66.

Urbano, J. (2013). "MIREX 2013 Symbolic Melodic Similarity: A Geometric Model supported with Hybrid Sequence Alignment." In: *Music Information Retrieval Evaluation eXchange*. Music Information Retrieval Evaluation eXchange.

Van Kranenburg, P. (2010). "A Computational Approach to Content-Based Retrieval of Folk Song Melodies." Doctoral dissertation, p. 172.

Van Kranenburg, P., Bruin, M. de, Grijp, L. P., and Wiering, F. (2014). *The Meertens Tune Collections*. Tech. rep. 1. Amsterdam: Meertens Institute.

Van Kranenburg, P., Volk, A., and Wiering, F. (2013). "A Comparison between Global and Local Features for Computational Classification of Folk Song Melodies." In: *Journal of New Music Research* 42.1, pp. 1–18.

Van Kranenburg, P., Volk, A., Wiering, F., and Veltkamp, R. C. (2009). "Musical Models for Folk-Song Melody Alignment." In: *Proceedings of the 10th International Society for Music Information Retrieval Conference*. Kobe, Japan, pp. 507–512.

Volk, A. and Van Kranenburg, P. (2012). "Melodic similarity among folk songs: An annotation study on similarity-based categorization in music." In: *Musicae Scientiae* 16.3, pp. 317–339.

Walshaw, C. (2011). *The abc music standard 2.1*. URL: http://abcnotation.com/wiki/abc:standard:v2.1 (visited on 06/19/2015).

Wilcoxon, F. (1946). "Individual comparisons of grouped data by ranking methods." In: *Journal of economic entomology* 39.6, p. 269.

Xia, G., Huang, T., Ma, Y., Dannenberg, R., and Faloutsos, C. (2014). "MidiFind: Similarity Search and Popularity Mining in Large MIDI Databases." In: *Sound, Music, and Motion* 8905, pp. 259–276.

Young, R. W. (1939). "Terminology for Logarithmic Frequency Units." In: *The Journal of the Acoustical Society of America* 11.1, p. 134.