

A PROBABILISTIC MODEL OF HIERARCHICAL MUSIC ANALYSIS

A Dissertation Presented

by

PHILLIP B. KIRLIN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of
DOCTOR OF PHILOSOPHY

February 2014

School of Computer Science

© 2014 Phillip B. Kirlin

Committee will be listed as:

David Jensen, Chair

Neil Immerman, Member

Edwina Rissland, Member

Gary Karpinski, Member

Department Chair will be listed as:

Lori A. Clarke, Chair

*This dissertation is dedicated to the memory of Paul Utgoff:
scholar, musician, mentor, and friend.*

ACKNOWLEDGMENTS

A few pages of 8.5 by 11 paper are hardly sufficient to list all the people who have helped me along the way to completing this dissertation. Nevertheless, I will try.

First and foremost, I would like to thank my advisor, David Jensen. David has been a wonderful advisor, and I feel privileged to have worked under his tutelage. Though his extensive knowledge of artificial intelligence and machine learning have proved indispensable, it is his knowledge of research methods that has shaped me the most. The fact that he was able to guide a research project involving a significant amount of music theory — without possessing such specialized knowledge himself — speaks volumes to his ability to clarify the basic research questions that underlie a computational study. More importantly, he has taught me to do the same. Thank you, David, for your never-ending patience; for your lessons in all things investigational, experimental, pedagogical, and presentational; and for never letting me give up.

The other members of my committee — Neil Immerman, Edwina Rissland, and Gary Karpinski — have been excellent resources as well and I would not have been able to finish this work without them.

I owe a great debt of gratitude to my original advisor, Paul Utgoff, who died too soon. Paul gave me my start in artificial intelligence research and the freedom to explore the ideas I wanted to explore. Paul was a dedicated researcher, teacher, and mentor; throughout his cancer treatments, he always made time for his students. I will always remember his warm smile, his gentle personality, and late nights playing poker at his home.

I cannot thank enough all of the graduate students at UMass with whom I toiled through the years. Thank you to those in the Machine Learning Laboratory where my trek began: David Stracuzzi, Gary Holness, Steve Murtagh, and Ben Teixeira; and thank you to those in the Knowledge Discovery Laboratory where I finished: David Arbour, Elisabeth Baseman, Andrew Fast, Lisa Friedland, Dan Garant, Amanda Gentzel, Michael Hay, Marc Maier, Katerina Marazopoulou, Hüseyin Oktay, Matt Rattigan, and Brian Taylor. Thank you as well to the research and technical staff: Dan Corkill, Matt Cornell, and Cindy Loiselle.

I greatly appreciate the efforts made by the three music theorists who evaluated the analyses produced by the algorithms described in this work. While I cannot thank them by name, they know who they are.

My life has been greatly influenced by a number of phenomenal teachers. Thank you to Gerry Berry and Jeff Leaf for giving me my start in computer science and technology, to Richard Layton for introducing me to music theory, and to Laura Edelbrock, for reminding me of the importance of music in my life.

Finally, I would like to thank my parents, George and Sheila, for putting up with thirty-one years of me, especially because the last ten were probably much more difficult than the first twenty-one. Their confidence in me never wavered, even when I thought I had hit bottom, and completing this dissertation would not have been possible without their constant encouragement and unconditional love.

ABSTRACT

Degrees will be listed as:

B.S., UNIVERSITY OF MARYLAND

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor David Jensen

Schenkerian music theory supposes that Western tonal compositions can be viewed as hierarchies of musical objects. The process of Schenkerian analysis reveals this hierarchy by identifying connections between notes or chords of a composition that illustrate both the small- and large-scale construction of the music. We present a new probabilistic model of this variety of music analysis, details of how the parameters of the model can be learned from a corpus, an algorithm for deriving the most probable analysis for a given piece of music, and both quantitative and human-based evaluations of the algorithm's performance. In addition, we describe the creation of the corpus, the first publicly available data set to contain both musical excerpts and corresponding computer-readable Schenkerian analyses. Combining this corpus with the probabilistic model gives us the first completely data-driven computational approach to hierarchical music analysis.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	ii
CHAPTER	
INTRODUCTION	1
1. MOTIVATION	3
2. PRIOR WORK	9
3. THE MOP REPRESENTATION	14
4. THE CORPUS	26
5. A JOINT PROBABILITY MODEL FOR MUSICAL STRUCTURES	33
6. ALGORITHMS FOR MUSIC ANALYSIS	41
7. EVALUATION	49
8. SUMMARY AND FUTURE WORK	69
APPENDICES	
A. MUSICAL EXCERPTS AND MOPS	71
B. MAXIMUM ACCURACY AS A FUNCTION OF RANK	167
BIBLIOGRAPHY	174

INTRODUCTION

An adage often repeated among music theorists is that “theory follows practice” (Godt, 1984). This statement implies that music theory is *reactive*, rather than *proactive*: the goal of music theory is to explain the music that composers write, in an attempt to make generalizations about compositional practices.

Schenkerian analysis is a widely-used theory of music which posits that compositions are structured as hierarchies of musical events, such as notes or intervals, with the surface level music at the lowest level of the hierarchy and an abstract structure representing the entire composition at the highest level. This type of analysis is used to reveal *deep structure* in the music and illustrate the relationships between various notes or chords at multiple levels of the hierarchy.

For more than forty years, researchers have attempted to construct computational systems that perform automated or semi-automated analysis of musical structure (Winograd, 1968; Kessler, 1975; Frankel et al., 1978; Meehan, 1980; Lerdahl and Jackendoff, 1983). Unfortunately, early computational models that used traditional symbolic artificial intelligence methods often lead to initially-promising systems that fell short in the long run; such systems could not replicate human-level performance in analyzing music. More recent models, such as those that take advantage of new representational techniques (Mavromatis and Brown, 2004; Marsden, 2010) or machine learning algorithms (Gilbert and Conklin, 2007) are promising but still rely on a hand-created set of rules. In contrast, the work presented here represents the first purely data-driven approach to modeling hierarchical music analysis, in that all of the “rules” of analysis, including how, when, and where to apply them, are learned algorithmically from a corpus of musical excerpts and corresponding music analyses.

Our approach begins with representing a hierarchical music analysis as a maximal outerplanar graph, or MOP (Chapter 3). Yust (2006) first proposed the MOP data structure as an elegant method for storing multiple levels of linear voice-leading connections between the notes of a composition. We illustrate how this representation reduces the size of the search space of possible analyses, and also leads to efficient algorithms for selecting analyses at random and iterating through all possible analyses for a given piece of music.

The corpus constructed for this research contains 41 musical excerpts written by eight different composers during the Baroque, Classical, and Romantic periods of European art music. What makes this corpus unique is that it also contains the corresponding Schenkerian analyses of the excerpts in a computer-interpretable format; no publicly available corpora of such analyses has been created before. We use this corpus to demonstrate statistically significant regularities in the way that people perform Schenkerian analysis (Chapter 4).

We next augment the MOP representation with a probabilistic interpretation to introduce a method for determining whether one MOP analysis is more likely than another, and we verify that this new model preserves potential rankings of analyses even under a probabilistic independence assumption (Chapter 5). We use this model to derive an algorithm that efficiently determines the most probable MOP analysis for a given piece of music (Chapter 6). Finally, we evaluate the model by examining the performance of the analysis algorithm using standard comparison metrics, and also by asking three experienced music theorists to compare algorithmically-produced analyses against the ground-truth analyses in our corpus (Chapter 7).

The key contributions of the work presented here are: (1) a new probabilistic model of hierarchical music analysis, along with evidence for its utility in ranking analyses appropriately, and an algorithm it admits for finding the most likely analysis of a given composition; (2) a corpus, the first of its kind to contain not only musical excerpts, but computer-readable Schenkerian analyses that can be used for supervised machine learning; and (3) a study comparing human- and algorithmically-produced analyses that quantitatively estimates how much further computational models of Schenkerian analysis need to progress to rival human performance.

CHAPTER 1

MOTIVATION

Music analysis is largely concerned with the study of musical structures: identifying them, relating them to each other, and examining how they work together to form larger structures. Analysts apply various techniques to discover how the building blocks of music, such as notes, chords, phrases, or larger components, function in relation to each other and the whole composition (Bent and Pople, 2013).

People are interested in music analysis for the same reason people are interested in analyzing literature, film, or other creative works: because we are fascinated by how a single work — be it a book, painting, or musical composition — can be composed of individual pieces; that is, words, brush strokes, or notes; and yet be larger than the sum of those pieces. In analyzing music, we want to dive inside a work of music and deconstruct it, examine it, explore every nook and cranny of the notes until we can discover what makes it sound the way it does. We want to know why this certain combination of notes, and not some other combination, made the most sense at the time to the composer. Music analysis is closely related to music theory, “the study of the structure of music” (Palisca and Bent, 2013). Both of these topics are a standard part of the undergraduate music curriculum because a knowledge of theory helps “students to develop critical thinking skills unique to the study of music” (Kang, 2006).

1.1 Types of music analysis

The basic structures of music that analysts and theorists study are “melody, rhythm, counterpoint, harmony, and form, but these elements are difficult to distinguish from each other and to separate from their contexts” (Palisca and Bent, 2013). In other words, the varying facets of a musical composition are difficult to study in isolation, especially at a more than basic level of understanding. Nevertheless, we will try to give an overview of the characteristics of these facets.

The two primary axes one observes in a musical score are the horizontal and the vertical; that is, the way notes relate to each other over a period of time (the horizontal axis), and the way notes relate to each other in pitch (the vertical axis) (Merritt, 2000). These two elements are commonly referred to as *harmony*, “the combining of notes simultaneously, to produce chords, and successively, to produce chord progressions” (Dahlhaus et al., 2013), and *voice-leading* or *counterpoint*, the combining of notes sequentially to create a melodic line

or *voice*, and the techniques of combining multiple voices in a harmonious fashion (Drabkin, 2013). Usually, harmony and voice-leading are the first two aspects of music that one begins studying in a first college-level course in music theory, and both involve finding and identifying relationships among groups of notes. These topics evolved organically over time as composers adopted new techniques. As we have already noted, “theory follows practice,” and harmony and voice-leading are no exceptions.

Harmony in music arises when notes either sound together temporally or the illusion of such a sonority is obtained through other compositional techniques. Analyzing the harmonic content of a piece involves explaining how these combinations of notes fall into certain established patterns (or identifying the lack of any matching pattern) and how these patterns work together to drive the music forward. Music theory novices often first see harmony in the context of *chord labeling*, where students are taught to write Roman numerals in the musical score to label the harmonic function of the chords. The principles of harmony, however, run much deeper than just the surface of the music (which is all that chord labeling examines); modern views of harmony seek to identify the purpose or *function* of a harmony not by solely examining the notes of which the harmony is comprised, but rather relating it to surrounding harmonies.

At the same time students are taught to label chords, they are often taught the introductory principles of counterpoint and voice-leading, or how to create musical lines that not only sound pleasing to the ear by themselves, but also blend harmoniously when played simultaneously. It is because of the two-dimensional nature of music that harmony and voice-leading cannot be studied separately. Particular sequences or combinations of harmonies can arise because of appropriate use of voice-leading and contrapuntal techniques, whereas choosing a harmonic structure for a composition will often dictate using certain voice-leading idioms.

Rhythm is an aspect of music that is not given as much time in introductory classes as harmony and voice-leading, though it still contributes greatly to the overall sound of a musical work. Many of the fundamental issues in rhythm are glossed over in formal music analysis precisely because anyone somewhat familiar with a given musical genre can often identify the rhythmic structure of a composition just by listening; for instance, most people can clap along to the beat of a song that they hear. Analyzing rhythm involves studying the temporal patterns in a musical composition independently of the pitches of the notes being played. For instance, certain genres are associated with certain rhythmic structures, such as syncopation in ragtime and jazz.

1.2 Schenkerian analysis

Schenkerian analysis is one of the most comprehensive methods for music analysis that we have available today (Brown, 2005; Whittall, 2013). Developed over a period of forty years by the music theorist Heinrich Schenker (1868–1935), Schenkerian analysis has been described as “revolutionary” (Salzer, 1952) and “the *lingua franca* of tonal theory in the Anglo-American academy” (Rings, 2011); many of its “principles and ways of thinking . . . have become an integral part of the musical discourse” (Cadwallader and Gagné, 1998). Furthermore, a

study of all articles published in the *Journal of Music Theory* from its inception in 1957 to 2004 revealed that Schenkerian analysis is the most common analytical method discussed (Goldenberg, 2006).

Schenkerian analysis introduced the idea that musical compositions have a *deep structure*. Schenker posited that an implicit hierarchy of musical objects is embedded in a composition, an idea that has come to be known as *structural levels*. The hierarchy illustrates how surface-level musical objects can be related to a more abstract background musical structure that governs the entire work. Schenkerian analysis is the process of uncovering the specific hierarchy for a given composition and illustrating how each note functions in relation to notes above and below it in the hierarchy.

Crudely, the analysis procedure begins from a musical score and proceeds in a reductive manner, eliminating notes at each successive level of the hierarchy until a fundamental background structure is reached. At a given level, the notes that are preserved to the next highest-level are said to be more *structural* than the notes not retained. More structural notes play larger roles in the overall organization of a composition, though these notes are not necessarily more memorable aurally.

Viewed from the top down, the hierarchy consists of a collection of *prolongations*: “each subsequent level of the hierarchy expands, or prolongs, the content of the previous level” (Forte, 1959). The reductive process hinges on identifying these individual prolongations: situations where a group of notes is elaborating a more “fundamental” group of notes. An example of this idea is when a musician decorates a note with a trill: the score shows a single note, but the musician substitutes a sequence of notes that alternate between two pitches. A Schenkerian would say that the played sequence of notes prolongs the single written note in the score. Schenkerian analysis takes this concept to the extreme, hypothesizing that a composition is constructed from a nested collection of these prolongations.

It is critical to observe that the goal of this method of analysis is not the reductive process *per se*, but rather the identification and justification for which prolongations are identified in a composition. This is because the music frequently presents situations which could be analyzed in multiple ways, and the analyst must decide what sort of prolongation makes the most musical sense.

The analyses that Schenker provided to explain his method always reduced a composition to one of three specific musical patterns at the most abstract level of the musical hierarchy. Each pattern consisted of a simple descending melodic line, called the *Urfinie* or *fundamental line*, and an accompanying harmonic progression expressed through a *bass arpeggiation*, or *Bassbrechung*. Together, these components form the *Ursatz*, or *fundamental structure*. Furthermore, Schenker hypothesized that because of the way listeners perceive music centered around a given pitch (i.e., *tonal* music), *every* tonal composition should be reducible to one of the three possible fundamental structures shown in Figure 1.1. This idea has proved much more controversial than that of structural levels.

Possibly the most frustrating aspect of Schenkerian analysis is that Schenker himself did not explain specifically how his method works. The “rules” for the reductive analysis procedure are derived from how listeners perceive music (specifically, Western tonal music), but

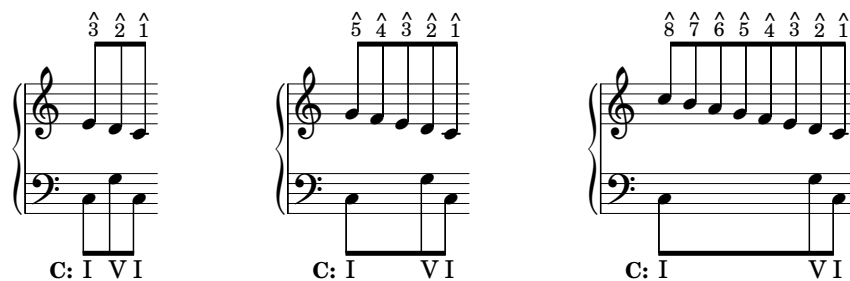


Figure 1.1: The three types of *Ursatz*.

Schenker did not explicitly state them. Instead, the process is illustrated through numerous examples of analyses completed by Schenker in his works *Der Tonwille* (1921) and *Der Freie Satz* (1935). Forte (1959) argues that “the important deficiencies in [Schenkerian analysis] arise from his failure to define with sufficient rigor the conditions under which particular structural events occur.”

While modern textbooks do try to give guidelines for how to execute an analysis, (e.g., Forte and Gilbert (1982a); Cadwallader and Gagné (1998); Pankhurst (2008)), they still often resort to illustrating the application of a prolongational technique by showing an analysis that uses it and then giving exercises for the student to practice applying it. Textbooks are almost useless for learning the analysis procedure without a teacher to lead a class through the book, assign exercises, and provide feedback to the students on their individual work.

1.3 Computational music

Introducing a computational aspect into musical endeavors is not new. In 18th century Europe, the *Musikalisches Würfelspiel* or “musical dice game” was a popular pastime in which people would use randomly-generated numbers to recombine pre-composed sections of music to generate new compositions. As electronic computers became commonplace in academia and government during the 20th century, people began to experiment with musical applications: two professors at the University of Illinois at Urbana-Champaign created a program that composed the *Iliac Suite* in 1956, the first piece of music written by computer.

The reasons why numerous people have chosen to study music through the lens of computer science are twofold. First, the problems are intellectually stimulating for their own sake, and both fields have paradigms that are readily adaptable to being combined with ideas from the other field. It is precisely this inherent adaptability that leads us to the second reason, namely that this interdisciplinary endeavor can lead to a wealth of new discoveries, knowledge, and useful applications in each of the parent fields of computer science and music.

Cook (2005) draws parallels between the recent research interests in studying music using computational methods and the interest in the 1980s of studying music from a psychological standpoint, stating that music naturally lends itself to scientific study because it is a “complex, culturally embedded activity that is open to quantitative analysis.” Music can be

quantified and digitized in various ways; the two main representations of music — the score and the aural performance — both can now be easily rendered in many varied computer-interpretable formats (Dannenberg, 1993). However, researchers acknowledge the “divide” that still exists in this interdisciplinary field, in that the music community has “not yet taken up the tools offered by mathematics and computation,” (Volk and Honingh, 2012). One argument for the lack of enthusiasm on the music side is the susceptibility of scientific researchers to create and study tools for their own sake, without relating the use of such tools and studies back to concrete musicological problems (Cook, 2005; Marsden, 2009). Additionally, some music scholars claim that “scientific methods may work to explain the physical world, [but] they cannot apply properly to music” (Brown and Dempster, 1989).

Historically, each domain of knowledge that computer science has approached has produced not only solutions to problems in that domain, but computational artifacts (such as algorithms or models) that are useful in other situations. For instance, speech recognition using hidden Markov models spurred future studies of such models, which can now be found in many other artificial intelligence domains. The sequence-alignment techniques refined by bioinformatics researchers have found other uses in the social sciences (Abbott and Tsay, 2000). Because music is a complex, multi-dimensional, human-created artifact, it is probable that research successes in computational musicology will be useful in other areas of computer science, for instance, in modeling similar complex phenomena or human creativity.

Computational methods also have great potential for advancing our knowledge and understanding of music. Traditionally, music research has proceeded with both limited representations of music (i.e., scores, which abstract away the nuances of individual performances), and small data sets (in many cases, single compositions for a study) (Cook, 2005). The raw processing capabilities of modern computers now permit us to use many more dimensions of music in studies (e.g., actual tempos of performances) and larger data sets. Additionally, approaching music from a scientific standpoint brings a certain empiricism to the domain which previously was difficult or impossible due to innate human biases (Volk et al., 2011; Marsden, 2009). Computational methods already have contributed numerous digital representations of music, formal models of musical phenomena, and ground-truth data sets, and will continue to do so.

If we restrict ourselves to discussing computational techniques as applied to music theory and analysis, we encounter a wealth of potential “real world” applications outside of the scholar’s ivory tower. Algorithms for and models of music analysis have applications in intelligent tutoring systems for teaching composition and analysis, notation or other music-creation software, and even algorithmic music composition. Music recommendation systems that use metrics for music similarity, such as Pandora or the iTunes Genius, could benefit from automated analysis procedures, as could systems for new music discovery such as last.fm. Models for music analysis could potentially have uses in predicting new hit songs or even in legal situations for discovering potential instances of musical plagiarism.

1.4 Evaluation of analyses

Historically, evaluation has been difficult for computational models of music analysis due to the lack of easily obtainable ground-truth data. This goes doubly for any model of Schenkerian analysis, because not only are there no computer-interpretable databases of Schenkerian analyses, but because Schenker declined to give any methodical description of his procedure, and therefore everyone does Schenkerian analysis slightly differently. As a result, the primary criteria for evaluating an analysis — produced by a human or computer — is the defensibility of the prolongations found and the resulting structural hierarchy produced, in that there should be musical evidence for why certain prolongations were identified and not others. Because the idea of “musical evidence” itself is maddeningly vague, there can be multiple possible “correct” analyses for a single composition when the music in question presents a conflicting situation. This happens frequently; experts do not always agree on the “correct” Schenkerian interpretation of a composition.

In a perfect world, in order to evaluate the quality of an algorithmic analysis system, one would have an exhaustive collection of correct analyses for each composition the system could ever analyze. This is, of course, infeasible. Therefore, for this study, we will adopt the convention of having a single ground-truth analysis for each input composition, and all algorithmically-produced analyses will be compared to the corresponding gold standard.

Naturally, some difficulties arise from this concession. A system that produces an output analysis that matches the ground-truth analysis is certainly good, but output that differs from the ground-truth is not necessarily bad. “Errors” can vary in magnitude: two analyses may differ in a surface-level prolongation that has no bearing on the background musical structure, or the analyses may identify vastly-different high-level abstractions of the same music. However, larger-magnitude differences between the ground-truth and the algorithmic output still do not necessarily mean the system-produced analysis is *wrong*; it could be a musically-defensible alternate way of analyzing the composition in question. While we are not trying to say that quantitative evaluation of Schenkerian analyses is impossible, it must be done carefully to avoid penalizing musically-plausible analyses that happen to differ from the analysis selected as the gold standard.

Evaluation is not the only issue in Schenkerian analysis that presents computational issues, however. In the next chapter, we will examine prior work in computational hierarchical analysis and see how others have tackled such issues.

CHAPTER 2

PRIOR WORK

2.1 Computational issues in Schenkerian analysis

Recall that the goal of Schenkerian analysis is to describe a musical composition as a series of increasingly-abstract hierarchical levels. Each level consists of *prolongations*: situations where an analyst has identified a set of notes S that is an elaboration of a more fundamental musical structure S' (usually a subset of S). A prolongation expresses the idea that the more abstract structure S' maintains musical control over the entire time span of S even though there may be additional notes during the time span that are not a part of S' .

In this chapter, we discuss the computational issues that arise in developing models and algorithms for Schenkerian analysis, along with previous lines of research and how they addressed these issues. Though researchers have been using computational methods to study Schenkerian analysis for over forty years, a number of challenges arise in nearly all studies, the primary one being the lack of a definitive, unambiguous set of rules for the analysis procedure. Lack of a unified ruleset leads to additional issues such as having multiple analyses possible for a single piece of music, determining whether the analysis procedure itself is done in a consistent manner among different people, and computational studies using *ad hoc* rules derived from guidelines in textbooks rather than learning such rules methodically from data.

There are additional challenges as well. First, while most models of Schenkerian analysis use a tree-based hierarchy, there are disagreements over which type of tree best represents a set of prolongations. Second, lack of an established representation wreaks havoc when it comes to evaluation metrics. In many studies, the analyses produced algorithmically are presented without any comparison to reference analyses simply because it is time consuming to turn human-produced analyses into machine-readable ones. Furthermore, representational choices sometimes make it difficult for a model of analysis to quantify how much better one candidate analysis is over another. Lastly, the time involved in producing a corpus of analyses in a machine-readable format has prevented any large-scale attempts at supervised learning for Schenkerian analysis; most previous automated learning attempts have been unsupervised. A supervised learning algorithm for Schenkerian analysis would require a corpus containing pieces of music and corresponding machine-readable analyses for each piece; an unsupervised algorithm would require only the music. Due to the time-intensive nature of encoding analyses for processing via computer, the lone supervised attempt at Schenkerian analysis used a corpus of only six pieces of music and analyses (Marsden, 2010).

Furthermore, the author conceded the evaluation of the computational model used in the study was not rigorous.

The “rules” of Schenkerian analysis

The primary challenge in computational Schenkerian analysis is the lack of a consistent set of rules for the procedure, leading to multiple musically-plausible analyses for a single piece of music. To most music theorists, however, this is not a problem. John Rahn argues that music theory is not about the search for *truth*, but rather the search for *explanation* and that the value in such theories of music is not derived from separating music into classes of “true” and “false” determined by a set of rules, but in the explanations that the theory offers as to how specific musical compositions are constructed (Rahn, 1980).

Experts sometimes disagree on what the “correct” Schenkerian analysis is for a composition. This issue arises precisely because Schenkerian analysis is concerned with explaining music rather than proving it has certain properties, coupled with the fact that Schenker himself did not offer any sort of algorithm for the analysis procedure. However, this is not a reason for despair, but rather an opportunity to refine the goals of computational Schenkerian analysis. Above, we mentioned how Schenkerian theory offers explanations (in the form of analyses) for how a composition works. Though multiple explanations are usually possible for any non-trivial piece of music, again, analysts endeavor to choose the “most musically satisfying description among the alternatives” (Rahn, 1980). Therefore, because any tonal musical composition can be analyzed from a Schenkerian standpoint (Brown, 2005), any useful computational model of Schenkerian analysis must have the ability to *compare* analyses to determine which one is a more musically satisfying interpretation of how the piece is constructed.

Modeling analysis

People often speak of “formalizing” Schenkerian analysis, but this term can mean different things to different researchers. To some, it means an attempt to formalize only the *representation* of an analysis (that is, devising appropriate computational abstractions for the input music, the prolongations available to act upon the music, and how they do so) without specifying any sort of *algorithm* for performing the analysis itself (e.g., by selecting a set of prolongations). Regardless of the presence or absence of an algorithm, a computational model of the prolongational hierarchy is necessary. Because Schenkerian analysis uses a rich symbolic language for expressing the musical intuitions of a listener or analyst (Yust, 2006), any attempt to formalize this language will usually restrict it in some way in order to make the resultant product more manageable. Therefore, we will occasionally refer to “hierarchical analysis” or “Schenkerian-like analysis” in order to distinguish the full, informal version of Schenkerian analysis and the formalized subset under study.

Schenkerian analysis operates in multiple dimensions simultaneously, most importantly along the melodic (temporal) and harmonic (pitch-based) axes. Full-fledged analyses take all the notes present in the score into account during the analysis process, due to the inextricably

linked nature of melody and harmony, but some studies choose to modify the input music in some fashion. This can simplify the prolongational model, and it usually reduces the size of the search space for any algorithms which use the model. A common simplification involves collapsing the polyphonic (multiple voice) musical into one of a few types of monophonic (single voice) input. Thus the prolongational model must only handle prolongations that occur in the “main melody” of the music and can represent the harmony of the composition as chords that occur simultaneously with the notes of the main melody (whether or not they are simultaneous in the original music).

An appropriate representation for the input music, therefore, goes hand-in-hand with an appropriate model for the prolongational hierarchy. Though full Schenkerian analysis includes other notations besides prolongations, most computational models prioritize effective methods for representing the prolongational hierarchy. Because this hierarchy is fundamentally recursive, most studies choose a recursively-structured model, most commonly a set of rules for prolongations that may be applied recursively, coupled with a tree-based structure to store the prolongations in the hierarchy.

In the rest of this chapter, we discuss these computational issues in the context of other researchers’ explorations of modeling Schenkerian analysis.

2.2 Previous approaches

The first piece of research in relating Schenkerian analysis to computing was the work of Michael Kassler, which began with his PhD dissertation in 1967, in which he developed a set of formal rules that govern the prolongations that operate from the middleground to the background levels in Schenkerian analysis. In other words, these rules operated on music that had already been reduced from the musical surface (what one sees in the score) to a two- or three-voice intermediary structure. Kassler went on to develop an algorithm that could derive Schenker-style hierarchical analyses from these middleground structures (Kassler, 1975, 1987). Kassler compared a Schenkerian analysis to a mathematical proof, in that both are attempts to show how a structure (musical or mathematical) can be derived from a finite set of axioms (the *Ursatz* in Schenker) according to rules of inference (prolongations in Schenker). Kassler handled the issue of multiple possible analyses of a single composition by orchestrating his rules such that only a single “minimal” music analysis (modulo the order of the rules being applied) would be possible for each middleground structure with which his program worked.

In a similar vein to Kassler, the team of Frankel, Rosenschein, and Smoliar created a set of rules for Schenkerian-style prolongations that operated from the musical foreground, rather than from the middleground. These rules were expressed as LISP functions, and similarly to Kassler, were initially produced to allow for “verification” of the “well-formedness” of musical compositions. The authors were initially optimistic about extending the system “in service of a computerized parsing (i.e., analysis) of a composition” (Frankel et al., 1976).

Later work, however, was not successful in doing so, and their last publication presented their system solely as an “aid” to the human analyst (Frankel et al., 1978; Smoliar, 1980).

The natural parallels between music and natural language, coupled with the recursive nature of Schenkerian analysis, led a number of researchers to explicitly study formalization of hierarchical analysis from the perspective of linguistics and formal grammars. The most well-known piece of work in this area is Lerdahl and Jackendoff’s *A Generative Theory of Tonal Music* (1983), in which the authors describe two different formal reductional systems — time-span reductions and prolongational reductions — along with general guidelines for conducting each type of reductive process. However, they acknowledged that their “theory cannot provide a computable procedure for determining musical analyses,” namely because while their reductional systems do include “preference rules” that come into play when encountering ambiguities during analysis, the rules are not specified with sufficient rigor (e.g., with numerical weights) to turn into an algorithm. Nevertheless, researchers have made attempts at replicating the analytical procedures in Lerdahl and Jackendoff’s work; the most successful being the endeavors of Hamanaka, Hirata, and Tojo (2005; 2006; 2007), which required user-supplied parameters to facilitate finding the “correct” analysis. Their later work (2009) focused on automating the parameter search, but results never produced analyses comparable to those done by humans.

Following in the footsteps of Lerdahl and Jackendoff, a number of projects appeared using formal grammars or similar techniques. Mavromatis and Brown (2004) explored using a context-free grammar for “parsing” music, and therefore producing a Schenkerian-style analysis via the parse tree. This initially promising work, however, again ran into difficulties later on because “the number of re-write rules required is preventatively large” (Marsden, 2010). However, the authors proposed a number of important guidelines for using grammars for Schenkerian analysis, one of the most important being that using *melodic intervals* (pairs of notes) rather than individual notes as terminal symbols in the grammar allows for a small amount of context hidden within the context-free grammar.

Other researchers also found great utility in using intervals rather than notes as grammatical atoms. Gilbert and Conklin (2007) used this technique to create the first *probabilistic* context-free grammar for hierarchical analysis, though their system was not explicitly Schenkerian because it did not attempt to reduce music to an *Ursatz*. Their system used a set of hand-created rules corresponding to Schenker-style prolongations and used unsupervised learning to train the system to give high probabilities to the compositions in their initial corpus: 1,350 melodic phrases chosen from Bach chorales. Analyses were computed using a “polynomial-time algorithm” (most likely the CYK algorithm, which runs in cubic time).

Marsden (2005b, 2007, 2010) also used a data structure built on prolongations of intervals rather than notes to model a hierarchical analysis. Like Gilbert and Conklin, Marsden’s model used a set of hand-specified rules corresponding to various types of prolongations commonly found in Schenkerian analysis. Marsden combined this model with a set of heuristics to create a chart-parser algorithm that runs in $O(n^3)$ time to find candidate analyses. Finding this space of possible analyses still prohibitively large, Marsden used a small corpus of

six themes and corresponding analyses from Mozart piano sonatas to derive a feature-based “goodness metric” using linear regression to score candidate analyses. After revising the chart parser to rank analyses based on this metric, he evaluated the algorithm on the six examples in the corpus. The results (accuracy levels for the top-ranked analysis varying from 79% to 98%), were biased, however, because the goodness metric used in the evaluation was trained on the entire corpus at once. With identical training and testing sets, it is unclear how well this model would generalize to new data. Furthermore, the corpus of analyses contained information about the notes present in each structural level in the musical hierarchy, but no information about how the notes in each level were related to notes in surrounding levels. That is, there was no explicit information about individual prolongations in the corpus. Without this additional information, such a corpus would be difficult to use to deduce the rules of Schenkerian analysis from the ground up.

In all of the approaches discussed above, the major stumbling block was the set of rules used for analysis: all of the projects used a set of rules *created by hand*. Furthermore, out of all the studies, only three algorithms were produced. Two of these algorithms were only made possible by sacrificing some accuracy for feasibility: Kassler’s worked from the middleground rather than the foreground, while Gilbert and Conklin’s was trained through unsupervised learning and could not produce analyses with an *Urfinie*; the true performance of Marsden’s algorithm is unclear.

In the remainder of this dissertation, we present the first probabilistic corpus-based exploration of modeling hierarchical music analysis. This approach uses a probabilistic context-free grammar, but is capable of reducing music to an *Urfinie* unlike Gilbert and Conklin. It uses a large corpus to allow for a non-biased evaluation, unlike Marsden, and works from the foreground, unlike Kassler.

CHAPTER 3

THE MOP REPRESENTATION

In Chapter 1, we discussed Schenkerian analysis and its central tenet: the idea that a tonal composition is structured as a series of hierarchical levels. During the analysis process, structural levels are uncovered by identifying *prolongations*, situations where a musical event (a note, chord, or harmony) remains in control of a musical passage even when the event is not physically sounding during the entire passage. In Chapter 2, we discussed the various methods researchers have used for storing the collection of prolongations found in an analysis, as well as techniques for modeling the algorithmic process of analysis itself. In this chapter, we will present the data structure that we use to model the prolongational hierarchy, along with some algorithms for manipulating the model.

3.1 Data structures for prolongations

Though the concept of prolongation is crucial to Schenker’s work, he neglected to give a precise meaning for how he was using the idea. In fact, not only did Schenker’s interpretation of prolongation change over time, modern theorists use the term inconsistently themselves. Nevertheless, modern meanings can be divided into two categories (Yust, 2006). First, the term can refer to a *static* prolongation, where “the musical events themselves are the subjects and objects of prolongation.” Second, some authors refer to a *dynamic* prolongation, where the “motion between tonal events [is] prolonged by motions to other tonal events.” The key word in the second definition is “motion,” in that the objects of prolongation in the dynamic sense are not notes, but the spaces between notes: melodic intervals. Interestingly, these two categories align nicely with the two groups of data structures discussed in Chapter 2: those that represent prolongations as hierarchies of notes (static), and those that use hierarchies of intervals (dynamic).

These two conceptualizations of prolongation can be made clearer with an example. Suppose a musical composition contains the five-note melodic sequence shown in Figure 3.1, a descending sequence from D down to G. Assume that an analyst interprets this passage as an outline of a G-major chord, and the analyst wishes to express the fact that the first, third, and fifth notes of the sequence (D, B, and G) are more structurally important in the music than the second and fourth notes (C and A). In this situation, the analyst would interpret the C and A as *passing tones*: notes that serve to transition smoothly between the preceding and following notes by filling in the space in between. From a Schenkerian aspect,

we would say that there are two *dynamic* prolongations at work here: the motion from D to B is prolonged by the motion from the D to the intermediate note C, and then from the C to the B. The motion from the B to the G is prolonged in a similar fashion.

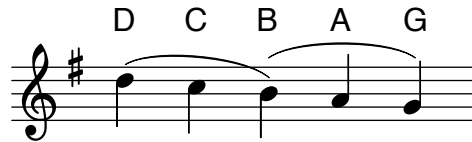


Figure 3.1: An arpeggiation of a G-major chord with passing tones. The slurs are a Schenkerian notation used to indicate the locations of prolongations.

However, there is another level of prolongation at work in the hierarchy here. Because the two critical notes that aurally determine a G chord are the G itself and the D a fifth above, a Schenkerian would say that the entire melodic span from D to G is being prolonged by the arpeggiation obtained by adding the B in the middle. More formally, the span from G to D is prolonged by the motion from D to B, and then from B to G. Therefore, the entire intervallic hierarchy can be represented by the tree structure shown in Figure 3.2. Note that the labels on the internal nodes are superfluous; they can be determined automatically from each internal node’s children.

Though the subjects and objects of dynamic prolongation are always melodic intervals (time spans from one note to another), it is not uncommon to shorten the rather verbose “motion”-centric language used to describe a prolongation. For instance, in the passing tone figure D–C–B mentioned above, we could rephrase the description of the underlying prolongation by saying the note C prolongs the motion from D to B. While this muddies the prolongational waters — it is the motion to and from the C that does the prolonging, not the note itself — the intent of the phrase is still clear.

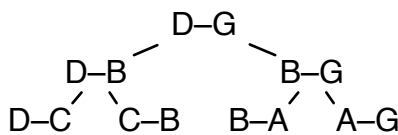


Figure 3.2: The prolongational hierarchy of a G-major chord with passing tones represented as a tree of melodic intervals.

Now let us examine the same five-note sequence using the *static* sense of prolongation, where individual notes are prolonged, rather than the spaces between them. Not surprisingly, such a hierarchy can be represented as a tree containing notes for nodes, rather than intervals. However, we immediately encounter a problem when trying to represent the passing tone sequence D–C–B. The note C needs to connect to both the D and the B in our tree because the C derives its musical function from both notes, yet if we restrict ourselves to binary trees,

we cannot represent this passing tone sequence elegantly: the C cannot connect to both D and B at the same level of the hierarchy, and in a passing tone structure like this one, neither the D nor the B is inherently more structural. Therefore, we have to make an arbitrary choice and elevate one of the notes to a higher level in the tree. We need to make a similar choice for other passing tone sequence B–A–G, which leads to another issue: the middle note B occurs only once in the music, yet it participates in two different prolongations. There is no elegant way to have the B be present on both sides of the prolongational tree, and again we are forced to make an arbitrary choice regarding which prolongation “owns” the B. Such choices destroy the inherent symmetry in the original musical passage, forcing us to draw a tree such as in Figure 3.3.

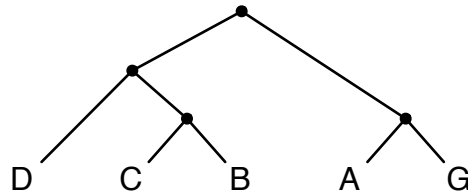


Figure 3.3: The prolongational hierarchy of a G-major chord with passing tones represented as a tree of notes. Notice how the tree cannot show the symmetry of the passing tone sequences.

We mentioned above how the internal labels on the interval tree are not necessary because they can be automatically determined from child nodes. This is easy to see because combining two adjacent melodic intervals yields another interval (by removing the middle note and considering the parent interval to be the entire time span from the first note to the last). It is not immediately clear how to transfer this idea to a tree of separate notes; combining two child notes does not yield a single parent note. Therefore, returning to the passing tone example, if we want to represent that the C is less structural than the D, we must label the internal nodes with additional information about structure, as in Figure 3.4.

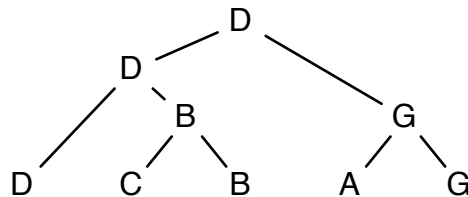


Figure 3.4: The prolongational hierarchy using internal labels.

Clearly, using the interval tree — and the dynamic interpretation of prolongation over static — leads to a cleaner representation of the prolongational hierarchy.

Interval trees can be more concisely represented using an alternate formulation. For an interval tree T , consider creating a graph G where the vertices in G are all the individual notes represented in T , and for every node x – y in T , we add the edge (x, y) to G . For Figure 3.2, this results in the structure shown in Figure 3.5, known as a *maximal outerplanar graph*, henceforth known as a *MOP*¹.

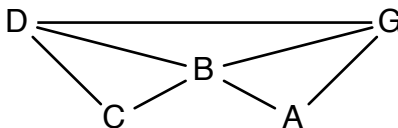


Figure 3.5: The prolongational hierarchy represented as a maximal outerplanar graph.

MOPs were first proposed as elegant structures for representing dynamic prolongations in a Schenkerian-style hierarchy by Yust (2006). A single MOP represents a hierarchy of intervals of a monophonic sequence of notes, though Yust proposed some extensions for polyphony. A MOP contains the same information present in an interval tree. For instance, the passing tone motion D–C–B mentioned frequently above is shown in a MOP by the presence of the triangle D–C–B in Figure 3.5.

Formally, a MOP is a complete triangulation of a polygon, where the vertices of the polygon are notes and the outer perimeter of the polygon consists of the melodic intervals between consecutive notes of the original music, except for the edge connecting the first note to the last, which we will refer to as the *root edge*, which is analogous to the root node of an interval tree. Each triangle in the polygon specifies a prolongation. By expressing the hierarchy in this fashion, each edge (x, y) carries the interpretation that notes x and y are “consecutive” at some level of abstraction of the music. Edges closer to the root edge express more abstract relationships than edges farther away.

Outerplanarity is a property of a graph that can be drawn such that all the vertices are on the perimeter of the graph. Such a condition is necessary for us to enforce the strict hierarchy among the prolongations. A *maximal* outerplanar graph cannot have any additional edges added to it without destroying the outerplanarity; such graphs are necessarily polygon triangulations, and under this interpretation, all prolongations must occur over triples of notes.

Using a MOP as a Schenkerian formalism for prolongations presents a number of representational issues to overcome. First is the issue of only permitting prolongations among triples of notes. Analysts sometimes identify prolongations occurring over larger groups of notes; a prolongation over four notes, for example, would appear as an open quadrilateral region in the MOP, waiting to be filled by an additional edge to turn the region into two triangles. Yust argues that analyses with “holes” such as these are incomplete, because they

¹Though perhaps a clearer abbreviation would be “MOPG,” we use the original abbreviation put forth by Yust (2006).

fail to completely specify how the notes of the music relate to each other. Therefore, we adopt the convention that analyses must be complete: MOPs must be completely triangulated.

Second, there is no way to represent a prolongation with only a single “parent” note in a MOP. Because MOPs inherently model prolongations as a way of moving *from* one musical event *to* another event, every prolongation must always have two parent notes and a single child note (these are the three notes of every triangle in the MOP). Music sometimes presents situations that an analyst would model with a one-parent prolongation, such as an incomplete neighbor tone. Yust interprets such prolongations as having a “missing” origin or goal note that has been elided with a nearby structural note, which substitutes in the MOP for the missing note. Yust uses dotted lines in his MOPs to illustrate this concept, though we omit them in the work described here as they do not directly figure into the discussion.

A third representational issue stems from trying to represent prolongations involving the first or last notes in the music. Prolongations necessarily take place over time, and in a MOP, every prolongation must involve exactly three notes, where we interpret the temporally middle note as prolonging the motion from the earliest note to the latest. Following this temporal logic, we can infer that the root edge of a MOP must therefore necessarily be between the first note of the music and the last, implying these are the two most structurally important notes of a composition. As this is not always true in compositions, Yust adds two pseudo-events to every MOP: an initiation event that is located temporally before the first note of the music, and a termination event, which is temporally positioned after the last note. The root edge of a MOP is fixed to always connect the initial event and the termination event. These extra events allow for any melodic interval — and therefore any pair of notes in the music — to be represented as the most structural event in the composition. For instance, in Figure 3.6, which shows the D–C–B–A–G pattern with initiation and termination events (labeled START and FINISH), the analyst has indicated that the G is the most structurally significant note in the passage, as this note prolongs the motion along the root edge.

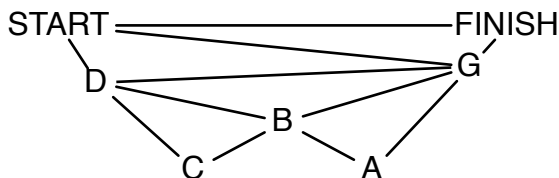


Figure 3.6: A MOP containing initiation and termination events.

We can now provide a formal definition of a MOP as used for representing musical prolongations. Suppose we are given a monophonic sequence of notes n_1, n_2, \dots, n_L . Define a set of vertices $V = \{n_1, n_2, \dots, n_L, \text{START}, \text{FINISH}\}$. Consider a set of directed edges $E \subseteq V \times V$, with the requirements that (a) for all integers $1 \leq i < L$, the edge $(n_i, n_{i+1}) \in E$, and (b) the edge $(\text{START}, \text{FINISH}) \in E$. The graph $G = (V, E)$ is a MOP if and only if E contains additional edges in order to make G a maximal outerplanar graph.

If G is a MOP, then G has the following musical interpretation. For every temporally-ordered triple of vertices $(x, y, z) \in V^3$, if the edges (x, y) , (y, z) , and (x, z) are members of E , then we say that the melodic interval $x-z$ is prolonged by the sub-intervals $x-y$ and $y-z$, or slightly less formally, that the melodic interval $x-z$ is prolonged by the note y . Hierarchically, the parent interval $x-z$ has two child intervals, $x-y$ and $y-z$; or equivalently, the child note y has two parent notes, x and z .

3.2 MOPs and search space size

Later we propose a number of algorithms for automatic Schenkerian-style analysis, but in this section we discuss how our choice of MOPs for modeling Schenkerian-style analysis affects the size of the search space that these algorithms must explore to find the “best” analysis.

First, we calculate the size of the search space under the MOP model. Given a sequence of n notes, we want to compute the total number of MOPs possible that could be constructed from these n notes. Any MOP containing n notes must have one vertex for each note, plus two additional vertices for the initiation and termination events, for $n + 2$ total vertices. These $n + 2$ vertices will fall on the perimeter of a polygon that the resulting MOP will triangulate, and therefore the perimeter will be comprised of $n + 2$ edges. One of these edges is the root edge, leaving $n + 1$ other perimeter edges, each of which would correspond to a leaf node in an equivalent (binary) interval tree. The number of possible binary trees having $n + 1$ leaf nodes is the n th Catalan number (C_n), so the size of the search space with the MOP representation is

$$C_n = \frac{1}{n+1} \binom{2n}{n}.$$

Using Stirling’s approximation, we can rewrite this as

$$C_n \approx \frac{4^n}{n^{3/2}\sqrt{\pi}} = O(4^n).$$

Now, we will consider the size of the search space if we used a static prolongation model, such as a hierarchy of individual notes, rather than a dynamic prolongation model like MOPs. Recall that if we use static prolongations, we must construct a tree of notes, rather than melodic intervals. Again, let us assume we are given a sequence of n notes to analyze. We know by the same logic used above that there are C_{n-1} possible binary trees that could be constructed from these notes, but we are forgetting that we also must choose the labels for the $n - 1$ *internal* nodes of the tree — an extra step not necessary for interval trees. Each internal node may inherit the label of either child node, leading to a total of

$$2^{n-1}C_{n-1} = O(8^n)$$

possible note hierarchies.

Clearly, though both search spaces are exponential in size, the MOP model leads to an asymptotically smaller space.

3.3 Algorithms for MOPs

Later, we examine an algorithm that produces the most likely MOP analysis for a given piece of music. In order to judge the algorithm’s performance, it will be useful to have a baseline level of accuracy that could be obtained from a hypothetical algorithm that creates MOPs in a stochastic fashion. Therefore, we derive two algorithms that allow us to (a) select a MOP uniformly at random from all possible MOPs for a given note sequence, and (b) iterate through all possible MOPs for such a sequence of notes.

3.3.1 Creating a MOP uniformly at random

The first algorithm addresses the problem of creating a random-constructed MOP. More specifically, given a sequence of n notes, we would like to choose a MOP uniformly at random from the C_n possible MOPs that could be created from the notes, and then construct this MOP efficiently.

Because MOPs are equivalent to polygon triangulations, we phrase this algorithm in terms of finding a random polygon triangulation. A completely triangulated polygon contains two types of edges: edges on the perimeter of the polygon, which we will call *perimeter edges*, and edges not on the perimeter, which we will call *internal edges*. Clearly, every perimeter edge in a triangulation is part of exactly one triangle (internal edges participate in two triangles, one on each side of the edge). Therefore, an algorithm to construct a complete polygon triangulation can proceed by iterating through each perimeter edge in a polygon and if the edge in question is not on the boundary of a triangle, then we can add either one or two edges to the triangulation to *triangulate the edge* in question.

Let us use the following example. Say we have the polygon A–B–C–D–E, as appears in the top row of Figure 3.7, and we want to triangulate perimeter edge A–B. This can be done by selecting one of vertices C, D, or E, and adding edges to form the triangle connecting A, B, and the selected vertex, as shown in the middle row of the figure. From here, depending on which vertex we chose, we either have a complete triangulation (having chosen vertex D), or we need to continue by triangulating an additional perimeter edge (having chosen vertex C or E), which can be accomplished via another iteration of the same procedure we just described. The result is a completely triangulated polygon; the shaded pentagons in the figure illustrate the five possible outcomes of the algorithm.

The only caveat left in describing our algorithm is the procedure for choosing the third vertex when triangulating a perimeter edge. In our example, consider completing the triangle for perimeter edge A–B choosing between vertices C, D, and E. We would like to make a selection in a probabilistic manner such that each of the five complete triangulations has an equally likely chance of being produced. However, choosing uniformly at random among the three vertices (i.e., each with probability $1/3$) will not lead to a uniform probability over the five complete triangulations. This is evident because if we choose from the vertices uniformly, the probability of the algorithm creating the complete triangulation in the middle row of Figure 3.7 is $1/3$, and the remaining four triangulations have probabilities each of $(1/3)(1/2) = 1/6$, which is clearly not a uniform distribution.

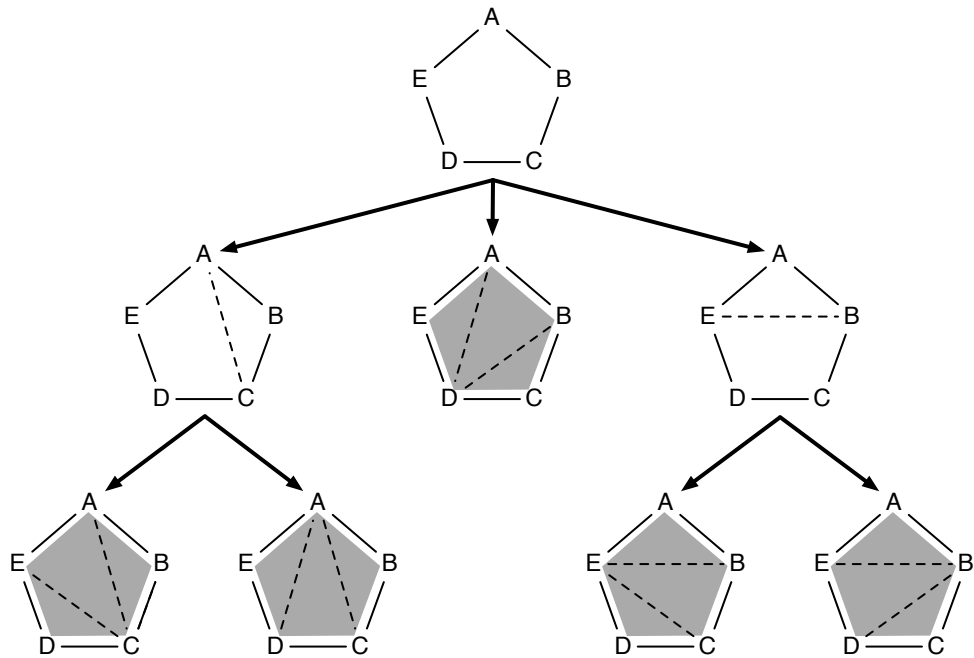


Figure 3.7: The decisions inherent in creation a MOP uniformly at random. The top row shows a completely untriangulated pentagon. The middle row shows the three possibilities for triangulating the edge A–B. The bottom row shows the possibilities for triangulating a remaining perimeter edge.

Instead, given a perimeter edge, we will weight the probability of choosing each possible vertex for a triangle non-uniformly using the following idea. Notice that whenever we triangulate a particular perimeter edge, the new triangle added divides the polygon into at most three subpolygons. At least one of these subpolygons will be a triangle, leaving at most two subpolygons remaining to be triangulated. We can calculate the number of further subtriangulations possible for each subpolygon using the Catalan numbers, and thereby calculate the total number of MOPs possible for each vertex. We can then use these numbers to weight the choice of vertex appropriately.

Suppose we label the vertices in our polygon v_1, v_2, \dots, v_n , and without loss of generality, consider completing the triangle for edge v_1-v_2 . The possible third vertices are $\{v_3, \dots, v_n\}$; suppose we choose v_i . The number of vertices in the two resulting subpolygons that may need further triangulation are $i-1$ and $n-i+2$, meaning the number of subtriangulations for each of the two subpolygons, are C_{i-3} and C_{n-i} respectively, where C_m is the m th Catalan number.

Define a probability mass function P as

$$P(v_i) = \frac{C_{i-3} \cdot C_{n-i}}{C_{n-2}}.$$

This pmf P gives rise to a probability distribution known as the Narayana distribution (Johnson et al., 2005), a shifted variant of the hypergeometric distribution. It can be shown that $\sum_{i=3}^n P(v_i) = 1$, demonstrating that this pmf leads to a valid probability distribution. We argue that using P to select vertices for our triangles leads to a uniform random distribution over MOPs.

Figure 3.7 illustrates how this works. To move from the top row of the figure to the middle row, we can choose from vertices C, D, or E to triangulate perimeter edge A–B. If we choose vertex D, our two subpolygons are triangles themselves (A–D–E and B–C–D), so $P(D) = (C_1 \cdot C_1)/C_3 = (1 \cdot 1)/5 = 1/5$. Appropriately, we learn $P(C) = P(E) = C_0 \cdot C_2/C_3 = (1 \cdot 2)/5 = 2/5$. Choosing vertex C or E requires us to triangulate another edge to move to the bottom row of the figure; the probabilities for each choice at this step are all $(C_1 \cdot C_1)/C_2 = 1/2$, so all four complete triangulations on the bottom row end up with total probabilities of $(2/5)(1/2) = 1/5$, which makes all five complete triangulations equiprobable.

Pseudocode for this algorithm is presented as Algorithm 1. The running time is linear in the number of vertices of the polygon, or equivalently, the number of notes of the music in question.

3.3.2 Enumerating all MOPs

Our next algorithm is a method for efficiently enumerating all MOPs possible for a fixed set of notes. Again, as in the previous section, we will phrase this algorithm in terms of polygon triangulations.

Let us assume we have a polygon with n vertices that we wish to triangulate. Consider the set of non-decreasing sequences of length $n - 2$ consisting of elements chosen

Algorithm 1 Create a MOP selected uniformly at random

```

1: procedure CREATE-RANDOM-MOP( $p$ )      ▷  $p$  is a polygon with vertices  $v_1, \dots, v_n$ 
2:   for each perimeter edge  $e = (v_x, v_y)$  in  $p$  do
3:     if  $e$  is not triangulated then
4:       Choose a vertex  $v_i$  according to the probability distribution defined by pmf  $P$ .
5:       Add edges  $(v_x, v_i)$  and  $(v_i, v_y)$  to  $p$ 
6:     end if
7:   end for
8: end procedure

```

from the set of integers $[0, n - 3]$. Furthermore, restrict this set to only those sequences $[x_0, x_1, \dots, x_j, \dots, x_{n-3}]$ such that for all j , $x_j \leq j$. As an example, the possible sequences that meet this criteria for $n = 5$ are $[0, 0, 0]$, $[0, 0, 1]$, $[0, 0, 2]$, $[0, 1, 1]$, and $[0, 1, 2]$.

Črepinšek and Mernik (2009) demonstrated that the total number of possible sequences that meet the criteria above for a given n is C_n , and also provided an algorithm for iterating through the sequences, imposing a total order upon them. We will provide an algorithm that provides a one-to-one mapping between a sequence, henceforth known as a *configuration*, and a polygon triangulation, therefore supplying a method for iterating over MOPs in a logical manner.

Our algorithm uses the fact that for a polygon with n vertices, a complete polygon triangulation is comprised of $n - 2$ triangles, and a configuration also has $n - 2$ elements. Each element in the configuration will become a triangle in the triangulation.

Assume a polygon p 's vertices are labeled clockwise from v_0 to v_{n-1} , and we wish to obtain the triangulation corresponding to a sequence $x = [x_0, \dots, x_{n-3}]$. We maintain a subpolygon p' that corresponds to the region of p that remains untriangulated; initially, $p' = p$. For each element x_i in x , examined *from right to left*, we interpret x_i as a vertex of p' , and find the two smallest integers j and k such that (a) v_j and v_k that are in p' , and (b) $x_i < j < k$. Graphically, this can be interpreted as inspecting the vertices of p' clockwise, starting from vertex v_{x_i} . We then add the edge (v_{x_i}, v_k) to our triangulation. This new edge will necessarily create the triangle (v_{x_i}, v_j, v_k) , so we remove the vertex v_j from p' to update the untriangulated region of our polygon.

Let us show an example of how this algorithm would work for a hexagon using the configuration $x = [0, 1, 2, 2]$. As is illustrated in Figure 3.8, initially, the untriangulated region consists of all the vertices $p' = \{v_0, v_1, v_2, v_3, v_4, v_5\}$. Examining the rightmost element of x , a 2, we locate the two lowest numbered vertices in p' greater than 2, which are $j = 3$ and $k = 4$. We add the edge (v_2, v_4) to our triangulation and remove v_3 from p' . The next element in x is another 2, so we repeat this procedure to obtain $x_j = 4$ and $k = 5$. We add the edge (v_4, v_5) to our triangulation and remove v_4 from p' . The next element in x is a 1, so $j = 2$ and $k = 5$. We add the edge (v_1, v_5) to our triangulation. The algorithm may terminate here because we do not need to examine the last element in x — notice that creating the second-to-last triangle also necessarily creates the last one.

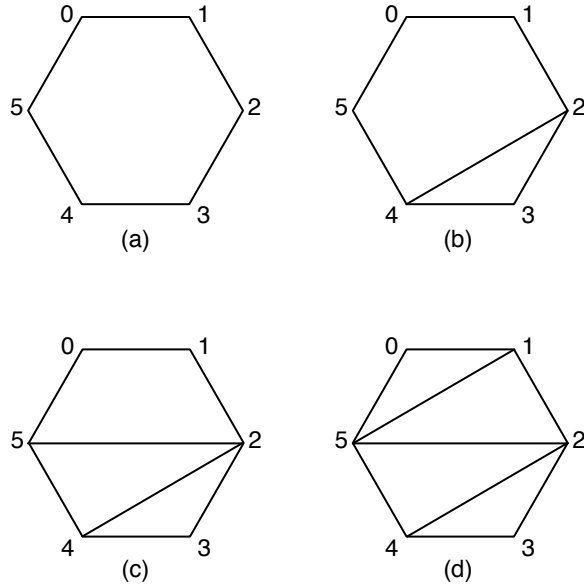


Figure 3.8: The four steps of creating the polygon triangulation of a hexagon corresponding to configuration $[0, 1, 2, 2]$. (a) Untriangulated polygon. (b) After step 1. (c) After step 2. (d) After step 3.

Pseudocode for the algorithm is provided as Algorithm 2. The vertices of the p' polygon can be maintained via a linked list, for $O(1)$ removal, with an additional array maintained for $O(1)$ indexing. Using this method sacrifices some space but turns the search for appropriate values for j and k into a constant-time operation. For a polygon with n vertices, the main loop of the algorithm will always create exactly $n - 3$ edges, so the running time is $O(n)$.

Algorithm 2 Create a MOP from a configuration sequence

```
1: procedure MOP-FROM-CONFIG( $p, x$ )      ▷  $p$  is a polygon with vertices  $v_0, \dots, v_{n-1}$ 
2:                                          ▷  $x$  is a valid configuration sequence
3:    $p' = \{v_0, \dots, v_{n-1}\}$ 
4:   for  $i \leftarrow n - 3$  downto 1 do
5:     find the smallest  $j > x_i$  in  $p'$ 
6:     find the smallest  $k > j$  in  $p'$ 
7:     add edge  $(v_{x_i}, v_k)$  to triangulation
8:      $p' \leftarrow p' - \{v_j\}$ 
9:   end for
10: end procedure
```

CHAPTER 4

THE CORPUS

In the remainder of this dissertation, we study a completely data-driven approach to modeling music analysis in a Schenkerian style. We use a supervised learning approach, and therefore require a corpus of data — in this case, musical compositions and their corresponding analyses — with which to derive information which will become part of an algorithm. In this chapter we explore the creation of this corpus and describe the results of an experiment that strongly suggests that finding an algorithm for hierarchical analysis is feasible.¹

4.1 Creation of a corpus

A supervised machine learning algorithm is designed to process a collection of (x, y) pairs in order to produce a function f that can map previously-unseen x -values to their corresponding y -values. In our situation, x -values are pieces of music and y -values are the corresponding hierarchical analyses. Therefore, under this paradigm, we require a corpus of musical compositions along with their Schenkerian analyses so that the resulting function f will be able to accept new pieces of music and output analyses for the pieces. However, creating such a corpus is a challenging task for a number of reasons.

First, although Schenkerian analysis is the primary technique for structural analysis of music, there are no central repositories of analyses available. Because analyses are usually produced for specific pedagogical or research purposes, analyses are usually found scattered throughout textbooks or music theory journals. Second, the very form of the analyses makes them difficult to store in printed format: a Schenkerian analysis is illustrated using the musical score itself and commonly requires multiple staves to show the hierarchy of levels. This requires substantial space on the printed page and is a deterrent to retaining large sets of analyses. Third, there is no established computer-interpretable format for Schenkerian analysis storage, and fourth, even if there were a format, it would take a great deal of effort to encode a number of analyses into processable computer files.

We solved these issues by scouring textbooks, journals, and notes from Schenkerian experts, devising a text-based representation of Schenkerian analysis, and manually encoding a large number of analyses in this representation. We selected excerpts from scores by Johann Sebastian Bach, George Frideric Handel, Joseph Haydn, Muzio Clementi, Wolfgang

¹This chapter draws heavily on the description and experimental results first published in Kirlin and Jensen (2011).

Amadeus Mozart, Ludwig van Beethoven, Franz Schubert, and Frédéric Chopin. All of the compositions were either for a solo keyboard instrument (or arranged for such an instrument) or for voice with keyboard accompaniment. All were in major keys, and we only used excerpts of the music that did not modulate. All the excerpts contained a single linear progression as the fundamental background structure — either an instance of the *Ursatz* or a rising linear progression. Some excerpts contained an *Ursatz* with an *interruption*: a Schenkerian construct that occurs when a musical phrase ends with an incomplete instance of the *Ursatz*, then repeats with a complete version; these excerpts were encoded as two separate examples in the corpus. These restrictions were put in place because we expected that machine learning algorithms would be able to better model a corpus with less variability among the pieces. In other words, we hypothesized that the underlying prolongations found in Schenkerian analyses done on (for instance) major-key compositions could be different than those found in minor-key pieces.

Analyses for the 41 excerpts chosen came from four places: Forte and Gilbert’s textbook *Introduction to Schenkerian Analysis* (1982a) and the corresponding instructor’s manual (1982b), Cadwallader and Gagne’s textbook *Analysis of Tonal Music* (1998), Pankhurst’s handbook *SchenkerGUIDE* (2008), and a professor of music theory who teaches a Schenkerian analysis class. These four sources are denoted by the labels F&G, C&G, SG, and Expert in the Table 4.1, which lists the excerpts in the corpus.

Table 4.1: The music excerpts in the corpus.

Excerpt ID	Composition	Analysis source
mozart1	Piano Sonata 11 in A major, K. 331, I, mm. 1–8	F&G
mozart2	Piano Sonata 13 in B-flat major, K. 333, III, mm. 1–8	F&G manual
mozart3	Piano Sonata 16 in C major, K. 545, III, mm. 1–8	F&G manual
mozart4	Six Variations on an Allegretto, K. Anh. 137, mm. 1–8	F&G manual
mozart5	Piano Sonata 7 in C major, K. 309, I, mm. 1–8	C&G
mozart6	Piano Sonata 13 in B-flat major, K. 333, I, mm. 1–4	F&G
mozart7	7 Variations in D major on “Willem van Nassau,” K. 25, mm. 1–6	SG
mozart8	Twelve Variations on “Ah vous dirai-je, Maman,” K. 265, Var. 1, mm. 23–32	SG, C&G
mozart9	12 Variations in E-flat major on “La belle Française,” K. 353, Theme, mm. 1–3	SG
mozart10	Minuet in F for Keyboard, K. 5, mm. 1–4	SG
mozart11	8 Minuets, K. 315, No. 1, Trio, mm. 1–8	SG
mozart12	12 Minuets, K. 103, No. 4, Trio, mm. 15–16	SG
mozart13	12 Minuets, K. 103, No. 3, Trio mm. 7–8,	SG
mozart14	Untitled from the London Sketchbook, K. 15a, No. 1, mm. 12–14	SG
mozart15	9 Variations in C major on “Lison dort,” K. 264, Theme, mm. 5–8	SG
mozart16	12 Minuets, K. 103, No. 12, Trio, mm. 13–16	SG
mozart17	12 Minuets, K. 103, No. 1, Trio, mm. 1–8	SG

Continued on next page

Table 4.1 — continued from previous page

Composer	Composition	Analysis source
mozart18	Piece in F for Keyboard, K. 33B, mm. 7–12	SG
schubert1	Impromptu in B-flat major, Op. 142, No. 3, mm. 1–8	F&G manual
schubert2	Impromptu in G-flat major, Op. 90, No. 3, mm. 1–8	F&G manual
schubert3	Impromptu in A-flat major, Op. 142, No. 2, mm. 1–8	C&G
schubert4	Wanderer’s Nachtlied, Op. 4, No. 3, mm. 1–3	SG
handell1	Trio Sonata in B-flat major, Gavotte, mm. 1–4	Expert
haydn1	Divertimento in B-flat major, Hob. 11/46, II, mm. 1–8	F&G
haydn2	Piano Sonata in C major, Hob. XVI/35, I, mm. 1–8	F&G
haydn3	Twelve Minuets, Hob. IX/11, Minuet No. 3, mm. 1–8	SG
haydn4	Piano Sonata in G major, Hob. XVI/39, I, mm. 1–2	SG
haydn5	Hob. XVII/3, Variation I, mm. 19–20	SG
haydn6	Hob. I/85, Trio, mm. 39–42	SG
haydn7	Hob. I/85, Menuetto, mm. 1–8	SG
bach1	Minuet in G major, BWV Anh. 114, mm. 1–16	Expert
bach2	Chorale 233, Werde munter, mein Gemute, mm. 1–4	Expert
bach3	Chorale 317 (BWV 156), Herr, wie du willst, so schicks mit mir, mm. 1–5	F&G manual
beethoven1	Seven Variations on a Theme by P. Winter, WoO 75, Variation 1, mm.1–8	C&G
beethoven2	Seven Variations on a Theme by P. Winter, WoO 75, Theme, mm. 1–8	C&G
beethoven3	Ninth Symphony, Ode to Joy theme from finale (8 measures)	SG
beethoven4	Piano Sonata in F minor, Op. 2, No. 1, Trio, mm. 1–4	SG
beethoven5	Seven Variations on God Save the King, Theme, mm. 1–6	SG
chopin1	Mazurka, Op. 17, No. 1, mm. 1–4	SG
chopin2	Grande Valse Brilliante, Op. 18, mm. 5–12	SG
clementi1	Sonatina for Piano, Op. 38, No. 1, mm. 1–2	SG

4.2 Encoding the corpus

With our selected musical excerpts and our corresponding analyses in hand, we needed to translate the musical information into machine-readable form. Musical data has many established encoding schemes; we used MusicXML, a format that preserves more information from the original score than say, MIDI.

To encode the analyses, we devised a text-based file format that could encode any sort of prolongation found in a Schenkerian analysis, as well as other Schenkerian phenomena, such as manifestations of the *Ursatz*. The format is easy for the human to input and easy for the computer to parse. Prolongations are represented using the syntax $X(Y)Z$, where X and Z are individual notes in the score and Y is a non-empty list of notes. Such a statement means that the notes in Y prolong the motion from note X to note Z . Additionally, we permit incomplete prolongations in the text file representation: one of X or Z may be omitted.

The text file description is more relaxed than the MOP representation to allow for easy human creation of analyses. Frequently, analyses found in textbooks or articles do not show every prolongation in the score, lest the analysis become visually cluttered. For instance,

it is common for an analyst to indicate a prolongation with multiple child notes, without identifying any further structural importance among the child notes. Such a prolongation, if translated into a MOP, would appear as a region larger than a triangle (e.g., a quadrilateral for a prolongation with two child notes). Sometimes, one can infer what the omitted prolongations should be from context, but in other cases the analyst’s intent is unclear.

We devised an algorithm to translate the text file analyses into MOPs. The algorithm largely translates prolongations in the text files to equivalent MOP prolongations, though extra steps are needed to handle incomplete prolongations in the text files. We permitted any prolongations with multiple child notes to be translated unchanged, meaning we allow MOPs to appear in the corpus not fully triangulated. Situations where we require fully-triangulated MOPs will be mentioned later, along with procedures for working around the missing prolongations.

Overall, the corpus contained 253 measures of music, 907 notes, and 792 prolongations. The lengths of individual excerpts ranged from 6 to 53 notes, which implies that the sizes of the individual search spaces ranged from 132 possible analyses (for an excerpt of 6 notes) to approximately 1.16×10^{29} possible analyses (for an excerpt of 53 notes).

4.3 The feasibility of analysis

We used this corpus to evaluate whether humans perform Schenkerian analysis in a consistent manner. We calculated the frequencies of prolongations found in our corpus in order to determine whether humans prefer locating certain types of prolongations over others. Finding such prolongations, or equivalently, a disinclination to have certain prolongations in an analysis, suggests that there are rules governing the process for analysis that could be extracted from the corpus.

Our first step was to calculate how often every type of prolongation appeared in the analyses in our corpus. Because each triangle in a MOP specifies the prolongation of an interval by two other intervals, we simply counted the frequencies of every type of triangle found in the MOP analyses. Triangles were defined by an ordered triple of the size of the parent interval and the sizes of the two child intervals. Intervals were denoted by size only, not quality or direction (e.g., an ascending major third was considered equivalent to a descending minor third), except in the case of unisons, where we distinguished between perfect and non-perfect unisons. Intervening octaves in intervals were removed (e.g., octaves were reduced to unisons), and furthermore, if any interval was larger than a fourth, it was inverted in the triple. These transformations equate prolongations that are identical under octave displacement. For example, the triples of notes (C, D, E), (E, D, C), (C, D \flat , E), (E, D \flat , C), (C, D \flat , E \flat), and (E \flat , D \flat , C) are all considered equivalent in our definition because each triple consists of a parent interval of a melodic third being elaborated by two melodic seconds.

Because the corpus of analyses contains polygons larger than triangles, extra care was required to derive appropriate triangle frequencies for these larger polygons. Our procedure was to enumerate all possible triangles that *could* appear in a triangulation of a larger

polygon, and, for each of these triangles, calculate the probability that the triangle would appear in a triangulation. If we assume a uniform distribution over all possible triangulations of the larger polygon, this is a straightforward calculation.

Assume we have a polygon with n vertices, numbered clockwise from 0 to $n - 1$, and we are interested in the probability that the triangle between vertices x , y , and z ($x < y < z$) appears in a complete triangulation of this polygon. This probability can be calculated from the number of complete triangulations of the polygon, which we know to be C_{n-2} , and the number of complete triangulations that contain the triangle in question, Δxyz .

To calculate this second quantity, we observe that any triangle drawn inside a polygon necessarily divides the interior of the polygon into four regions: the triangle itself, plus the three regions outside the triangle but inside the polygon (though it is possible for some of these regions to be degenerate line segments). Any complete triangulation of the polygon that contains Δxyz must necessarily completely triangulate the three remaining regions outside of the triangle, and we simply multiply the number of ways of triangulating each of those three regions to obtain the total number of complete triangulations that contain Δxyz .

The number of ways of triangulating each of the three regions is directly related to the size of each region, which we can calculate from the values of the vertices x , y , and z . The sizes (number of vertices in the polygons) of these regions are $y - x + 1$, $z - y + 1$, and $n + x - z + 1$, respectively. The number of ways to triangulate each region is the Catalan number for the size of each region minus two, and therefore the complete calculation for our desired probability is

$$P(\Delta xyz) = \frac{C_{y-x-1} \cdot C_{z-y-1} \cdot C_{n+x-z-1}}{C_{n-2}}.$$

After calculating frequencies for all the types of triangle in the corpus, we tested them to see which were statistically significant given the null hypothesis that the corpus analyses resemble randomly-performed analyses (where any triangulation of a MOP is as likely as any other) in their triangle frequencies. To calculate the expected frequencies under the null hypothesis, we took each analysis from the corpus, removed all internal edges to obtain a completely untriangulated polygon, and used the same probability calculation as above to compute the expected frequencies of every type of triangle possible. We compared these frequencies to the observed frequencies from the corpus analyses and ran individual binomial tests for each type of triangle to determine if the observed frequency differed significantly from the expected frequency.

There were 49 different types of triangle possible, considering the music in the corpus. Assuming we are interested in triangles whose difference in frequencies is statistically significant at the 5% level, using the Šidák correction indicates we need to look for triangles whose binomial tests resulted in p -values smaller than 0.001. There were eleven types of triangles that met this criteria, not counting triangles that used the START or FINISH vertices as endpoints. Tables 4.2 and 4.3 show the eleven types, though because intervals have had intervening octaves removed and are inverted if larger than a fourth, each type of triangle represents an entire class of prolongations.

Interval $L-R$	Interval $L-M$	Interval $M-R$	Observed frequency	Expected frequency	p -value
second	perfect unison	second	101	28	3.0×10^{-28}
third	second	second	216	109	7.2×10^{-23}
perfect unison	third	third	46	20	7.8×10^{-7}
second	third	second	77	50	1.9×10^{-4}

Table 4.2: The four triangle types whose differences in observed and expected frequency were statistically significant and appear *more* frequently than would be expected under the null hypothesis. The first three columns indicate the intervals from the left note to the right note, the left to the middle, and the middle to the right.

Interval $L-R$	Interval $L-M$	Interval $M-R$	Observed frequency	Expected frequency	p -value
fourth	second	third	4	38	2.3×10^{-12}
third	fourth	second	3	26	2.4×10^{-8}
third	second	fourth	1	21	2.7×10^{-8}
fourth	third	second	13	41	3.8×10^{-7}
fourth	second	fourth	3	22	1.1×10^{-6}
fourth	fourth	second	6	25	7.1×10^{-6}
second	second	perfect unison	9	27	6.2×10^{-5}

Table 4.3: The seven triangle types whose differences in observed and expected frequency were statistically significant and appear *less* frequently than would be expected under the null hypothesis.

We can provide musical explanations for a number of the different triangle types in Tables 4.2 and 4.3. The first row of the Table 4.2 indicates that the type of prolongation that had the most statistically significant differences between the observed and expected frequencies was the interval of a second being prolonged by a perfect unison and then another second. Musically, this often occurs in the ground-truth analyses when repeated notes are merged into a single note (a perfect unison means the notes of the interval are identical). The second row of the table has a much more musically interesting explanation. A third being elaborated by two seconds is exactly the passing tone pattern discussed earlier in this dissertation. This is one of the most fundamental types of prolongation, so it makes sense that such a pattern was observed almost twice as often as would be expected.

The third row of Table 4.2 indicates that prolonging a perfect unison with a leap of a third and then back to the original note also appears more frequently than would be expected. A musical interpretation of this prolongation is when a composer chooses to decorate a note of a chord by leaping to another chord tone a third away, then back to the starting pitch, a common prolongation that could be identified by an analyst during arpeggiations in the music. The last row of Table 4.2 corresponds to another kind of frequently-found prolongation: a second being prolonged by a skip of a third, and then a step (a second) in the other direction. (We can deduce the direction change from context — it is impossible to have the third and the second be in the same direction, because then the overall interval would be a fourth.) This type of prolongation is found when composers choose to decorate a stepwise pattern with a leap in the middle to add melodic interest.

The rows of Table 4.3 illustrate prolongations that are found less frequently than would be expected if the corpus analyses were chosen at random. Thus, these prolongations can be interpreted as those that analysts tend to avoid for musical reasons, or at least those that are less musically plausible when other prolongations are available.

The fact that we can identify these consistencies in human-produced analyses strongly suggests that the Schenkerian analysis procedure is not random, and that there are rules that we can uncover through methodical examination of the corpus. It also suggests that at least some of these rules are not specific to the analyst, as we could uncover these statistically significant differences in triangle frequencies even from a corpus containing analyses produced by different people.

CHAPTER 5

A JOINT PROBABILITY MODEL FOR MUSICAL STRUCTURES

In Chapter 3 we introduced the *MOP*, a data structure conceived for storing Schenkerian-style hierarchical analyses of a monophonic sequence of notes. In this chapter, we impose a mathematical model upon a MOP that will allow us to calculate the probability that a particular MOP analysis is the best one for a given piece of music.¹

5.1 A probabilistic interpretation of MOPs

Recall that, given a sequence of n notes, a MOP is constructed over a polygon whose vertices are the n notes plus two additional START and FINISH pseudo-events, for a total of $n + 2$ vertices. There are C_n (the n th Catalan number) possible ways to triangulate such a polygon, and every possible triangulation will contain n internal triangles. Each triangle has three endpoints, which we will denote by L , R , and C , corresponding to the left parent note, the right parent note, and the child note, respectively. The assignment of these labels to a triangle is unambiguous because MOPs are “oriented” by virtue of the temporal dimension: the left endpoint is always the earliest note of the three, while the right endpoint is always the latest. This corresponds exactly to our interpretation of a musical prolongation as described earlier: a prolongation always occurs among exactly three notes, where the middle (child) note prolongs the motion from the left note to the right.

We now define two probability distributions over triangles defined by their three endpoints: the *joint triangle distribution* $P(L, C, R)$, and the *conditional triangle distribution* $P(C \mid L, R)$. The joint distribution tells us how likely it is for a certain type of triangle to appear in an analysis, whereas the conditional distribution tells us how likely it is for given melodic interval (from L to R) to be prolonged by a given child note (C).

We are interested in these distributions because they can be used to build a probabilistic model for an entire analysis in MOP form. Assume we are given a sequence of notes $N =$

¹Preliminary results of some of the work described in this section are presented in Kirilin and Jensen (2011).

(n_1, n_2, \dots, n_L) . We may define the probability of a MOP analysis A as $P(A | N)$. Because a MOP analysis is defined by the set of triangles T_{all} comprising the MOP, we will define

$$P(A | N) := P(T_{\text{all}}) = P(T_1, T_2, \dots, T_m).$$

We would like to use the corpus of analyses described in Chapter 5 to train a mathematical model to estimate the probability above. However, the curse of dimensionality prevents us from directly using the joint probability distribution $P(T_1, T_2, \dots, T_m)$ as the basis for the model because doing so would require many more ground-truth analyses than we have in our corpus — and almost certainly more than anyone has available — to get good estimates of the joint probabilities for every combination of triangles. Instead, as an approximation, we will assume that the presence of a given type of triangle in a MOP is independent of the presence of all other triangles in the MOP. In other words,

$$P(A | N) := P(T_{\text{all}}) = P(T_1, T_2, \dots, T_m) = P(T_1) \cdot P(T_2) \cdots P(T_m).$$

From here, we can use either the joint triangle distribution or the conditional triangle distribution to define $P(T_i)$. The main issue distinguishing the distributions can be thought of as how they reward (or penalize) frequently-occurring (or uncommon) triangles found in analyses. The joint distribution is blind to the context in which a triangle occurs, in that the joint probability can be thought of as a straightforward representation of how common it is for a triangle to be found in an analysis. The conditional distribution, on the other hand, calculates a triangle’s probability by assuming — in a sense — that the left and right parent notes are fixed and only an appropriate child note needs to be selected. The conditional distribution also has some elegant mathematical properties that relate to the algorithms that we will describe in the next chapter.

As will become clear in later chapters, we are primarily concerned with using $P(A | N)$ to rank a set of candidate analyses, rather than using the probability $P(A | N)$ itself as a number in further calculations. That is, we are more interested in having numeric *comparisons* between $P(A_1 | N)$ and $P(A_2 | N)$ being accurate for all pairs of analyses A_1, A_2 as opposed to the probability estimates being close to the true probabilities. We can perform an experiment to answer two related questions regarding these relative comparisons. First, we want to know if making the independence assumption preserves the relative rank ordering for analyses; and second, we wish to learn if the joint or conditional triangle distribution performs better than the other at the task of ranking analyses.

A single experiment will answer both questions for us. Briefly, we begin by generating a number of different possible analyses for a single piece of music and ranking them by decreasing similarity to a randomly-chosen “best” analysis. We use this synthetic ranking to generate a synthetic corpus of MOPs by sampling the MOP analyses from the ranking proportionally to rank (higher-ranked MOPs are sampled more often). Next, we compute the triangle frequencies in the synthetic corpus using the same procedure described in the previous chapter, and use these frequencies to construct the $P(A | N)$ distributions for the joint and conditional triangle models using the independence assumption. We use these

distributions to re-rank all of the candidate analyses, and then compare this new ranking to our original synthetic ranking. If the independence assumption preserves rank orderings, then the two rankings should be similar. This procedure is illustrated in Figure 5.1.

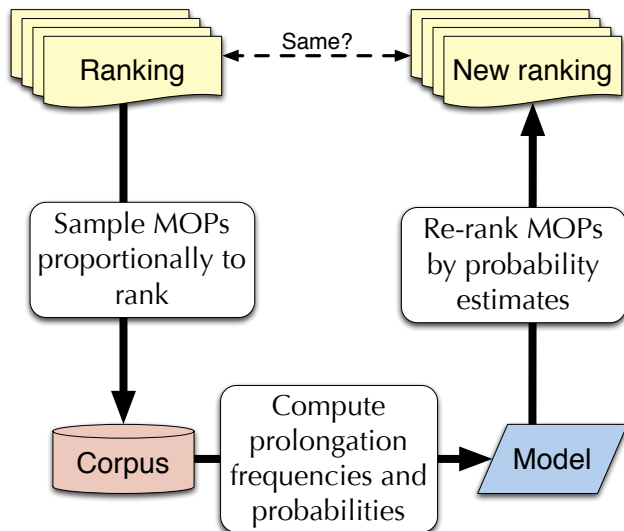


Figure 5.1: A visual depiction of the reranking procedure used to judge the appropriateness of the independence assumption.

The exact procedure is as follows. We assume that every note in a piece of music is distinguishable from every other note, something not feasible with smaller data sets, but done here with the knowledge that humans may use a note’s location within the piece as a feature of the note to guide the analysis procedure. Therefore, each piece of music is represented by a sequence of integers $N = (1, 2, \dots, n)$. We take a uniform sample of 1,000 MOPs (using Algorithm 1) from the space of possible MOPs over N ; the sampling is necessary as we have already illustrated how the number of MOPs grows exponentially in relation to N .

We randomly select one MOP to be the “best” analysis, and create an array A with the 1,000 MOPs sorted in decreasing order of similarity to the best MOP, where similarity is defined as the number of triangles in common between two MOPs. We place the best MOP at $A[0]$. We use a variation of the normal distribution to sample one million MOPs from A as follows: each sample is the MOP at position i in the array, where i is the absolute value of a normal random variable with $\mu = 0$ and varying σ , rounded down. Values of i that correspond to MOPs outside of array A are resampled. The one million sampled MOPs are placed into a multiset M and sorted by decreasing frequency into an array R , representing the ground-truth ranking of MOPs.

We then compute the frequency of each triangle in multiset M , calculate the probabilities for each triangle under the joint and conditional models, and use the independence assumption to compute a probability estimate for each MOP. We generate a new ranking R'

of the MOPs from their probability estimates, and compute three different rank correlation coefficients to test how well the re-ranking R' compares with the original ranking R . We use Spearman's ρ and Kendall's τ , which are standard correlation metrics, but we also use a coefficient developed by da Costa and Soares (2005) called r_W that more heavily weights higher items in the ranking, which Spearman and Kendall's coefficients do not.

We compute these ranking correlation coefficients for R versus R' using lengths of note sequences n between 10 and 50, and standard deviations σ for the normal distribution varying between 1 and 20. σ determines the number of analyses r ranked in R and R' by the formula $r \approx 4.66\sigma + 1.65$. In other words, when $\sigma = 1$, the random procedure only selects five or six analyses from the 1,000 available in A , but when $\sigma = 20$, approximately 95 are selected.

The three correlation coefficients always take on values between -1 and 1 , with 1 indicating the two rankings are identical and -1 meaning the two rankings are reverses of each other. Figure 5.2 shows heatmaps for the three coefficients using the joint and conditional triangle models. Under the joint model, the mean values of ρ , τ , and r_W are 0.9633, 0.8852, and 0.9722, respectively, while under the conditional model the means are 0.9482, 0.8298, 0.9525. These numbers indicate (a) that the independence assumption does seem to preserve relative probability judgements because the mean rank coefficients are all close to 1 and individual coefficients for the various combinations of n and σ never fall below 0.5, and (b) the joint model slightly outperforms the conditional model.

5.2 Estimating triangle frequencies

Given the high performance of ranking given the assumption of independence, we return to the issue of using the corpus of analyses to obtain estimates for the probability of an individual triangle being found in an analysis, under either the joint or conditional models. A straightforward way to estimate these quantities is to count their frequencies in the corpus and normalize them to obtain a probability distribution. For example, to estimate the probability of the triangle formed by endpoints L , C , and R under the joint triangle model, we would divide the number of times triangle $\triangle LCR$ appears in the corpus by the total number of triangles in the corpus (here, 907 triangles). Under the conditional triangle model, we would divide the number of times triangle $\triangle LCR$ appears by the number of times any triangle containing endpoints L and R (with any C) appears.

This approach yields better estimates with a large corpus and a small number of triangle categories. In Chapter 5, we showed how even when categorizing triangles by the melodic intervals between the endpoints — intervals that had had intervening octaves removed and some intervals inverted to diminish the number of possible categories — there were still 49 categories of triangle to consider. Furthermore, we wish to use more information from the music to guide the analysis process by introducing more features into the triangle descriptions, such as harmonic and metrical information. The curse of dimensionality again would require us to have a much larger corpus than we had access to.

In order to create a more accurate model that can handle more features with a smaller corpus, we use an ensemble learning method known as a *random forest* (Breiman, 2001).

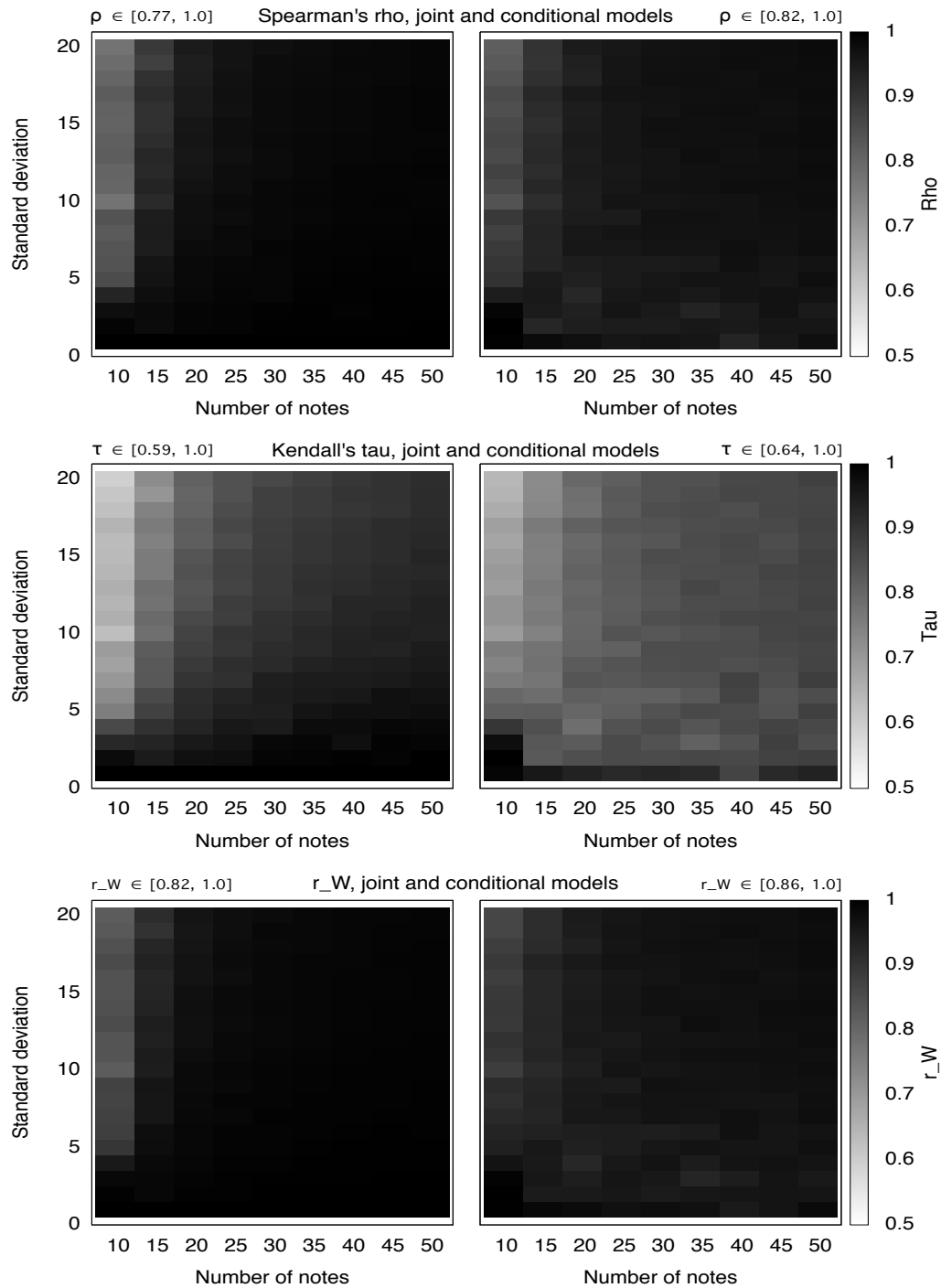


Figure 5.2: Heatmaps of the ranking correlation coefficients. Darker areas (closer to 1) indicate combinations of number of notes (n) and standard deviation (σ) where the experiment produced a new ranking (using the independence assumption) similar to the original ranking. Note that the color scale begins at 0.5, even though the coefficients' theoretical minimum is -1 .

Traditionally, a random forest is a classification method based on creating a collection of decision trees at training time. Each decision tree is only trained on a subset of the features available. The output of the random forest is normally the mode of the output of each individual tree, but we counted the frequencies of the outputs of all the trees and normalized them into a probability distribution instead (Provost and Domingos, 2003).

It is straightforward to use random forests for obtaining estimates for the probabilities comprising the conditional triangle distribution $P(C | L, R)$: we use features of the left and right endpoints to predict features of the middle point. Recall that in general, supervised learning techniques use a training set consisting of (x, y) pairs to learn a generalized function f that can predict a value for y for a previously-unseen value for x . As an example, in a simplistic model where we consider pitch as the only feature, our training set would consist of triangles taken from the corpus of analyses, transformed so that the left and right pitches for each triangle become the x -value, and the center pitch becomes the y -value. The random forest algorithm would process this data set and output a collection of decision trees where each tree could be treated as a function that would accept two pitches as input (L and R) and output a third pitch (C) as a prediction of the most likely middle pitch for the triangle. It is a simple mathematical calculation to then process the output of each tree for a given L and R to obtain a probability distribution over values of C .

Considering a single feature per triangle endpoint, however, will not give us the robust classifier that we desire: we want to incorporate more features, such as harmonic and metrical information, as we suspect these factors play a role in the Schenkerian analysis process. However, asking each individual decision tree in a random forest to predict multiple features in the output leads to another curse of dimensionality situation. Therefore, we will factor the conditional model using the rules of conditional probability. Assuming the features of note C — the note the random forest is trying to learn a probability distribution over — are denoted C_1 through C_n , we can rewrite $P(C | L, R)$ as

$$\begin{aligned}
 P(C | L, R) &= P(C_1, C_2, \dots, C_n | L, R) \\
 &= P(C_1 | L, R) \cdot P(C_2, C_3, \dots, C_n | C_1, L, R) \\
 &= P(C_1 | L, R) \cdot P(C_2, | C_1, L, R) \cdot P(C_3, C_4, \dots, C_n | C_1, C_2, L, R) \\
 &= P(C_1 | L, R) \cdot P(C_2, | C_1, L, R) \cdots P(C_n | C_1, \dots, C_{n-1}, L, R). \quad (5.1)
 \end{aligned}$$

This formulation allows us to model each feature of the note using its own separate random forest. We chose six features to use for the middle note C :

- C_6 : The scale degree (1–7) of the note.
- C_5 : The harmony present in the music at the time of the note, represented as a Roman numeral I through VII.
- C_4 : The category of harmony present in the music at the time of the note, represented as a selection from the set *tonic* (any I chord), *dominant* (any V or VII chord), *pre-dominant* (II, II⁶, or IV), *applied dominant*, or *VI chord*. (Our data set did not have any III chords.)

- C_3 : Whether the note is a chord tone in the harmony present at the time.
- C_2 : The metrical strength of the note's position as compared to the metrical strength of note L .
- C_1 : The metrical strength of the note's position as compared to the metrical strength of note R .

Note that the features are labeled C_6 through C_1 ; this ordering is used to factor the model as described in Equation 5.1. This ordering of the features is used because the features convey more specific musical information as one moves from from C_1 to C_6 , and therefore it makes sense to allow the more specific features extra training information from the less specific features.

We also used a variety of features for the left and right notes, L and R . These were:

- The scale degree (1–7) of the notes L and R (two features).
- The melodic interval from L to R , with intervening octaves removed.
- The melodic interval from L to R , with intervening octaves removed and intervals larger than a fourth inverted.
- The direction of the melodic interval from L to R ; i.e., *up* or *down*.
- The harmony present in the music at the time of L or R , represented as a Roman numeral I through VII (two features).
- The category of harmony present in the music at the time of L or R , represented as a selection from the set *tonic*, *dominant*, *predominant*, *applied dominant*, or *six* (two features).
- Whether L or R was a chord tone in the harmony present at the time (two features).
- A number indicating the beat strength of the metrical position of L or R . The downbeat of a measure is 0. For duple meters, the halfway point of the measure is 1; for triple meters, the one-third and two-thirds points are 1. This pattern continues with strength levels of 2, 3, and so on (two features).
- Whether L and R are consecutive notes in the music.
- Whether L and R are in the same measure in the music.
- Whether L and R are in consecutive measures in the music.

When L or R correspond to either the START or FINISH vertices, most of these features do not make sense because they can only be calculated for actual notes. In those situations, a special value (e.g., *invalid*) is used as the value of the feature.

Random forests can be customized by controlling the number of trees in each forest, how many features are used per tree, and each tree's maximum depth. Through trial and error, we settled on forests containing 1,000 trees with a maximum depth of four. We used Breiman's original idea of choosing a random selection of $m = \text{int}(\log_2 M + 1)$ features to construct each individual tree in the forest, where M is the total number of features available to us. In our case, $M = 16$, so $m = 5$.

In the next chapter, we illustrate how to use this model to construct a class of algorithms that can analyze music in a hierarchical manner.

CHAPTER 6

ALGORITHMS FOR MUSIC ANALYSIS

In the previous chapters we explored the MOP representation of music analysis and a probabilistic model for determining the likelihood that a given MOP corresponds to a correct hierarchical music analysis. In this chapter, we present a variety of algorithms for efficiently searching the space of possible MOP analyses to determine both the most probable analysis and a probability ranking over all analyses.

6.1 A probabilistic grammar approach

Some previous studies in algorithms for automatic music analysis reference the concept of “parsing” music, mirroring the similar concept in natural language processing of parsing sentences (Bod, 2001; Marsden, 2005a; Gilbert and Conklin, 2007; Marsden and Wiggins, 2008). Parsing algorithms usually are designed to work with a particular class of languages in the Chomsky hierarchy: a hierarchy of language classes where each class is associated with a particular kind of *formal grammar*.

A formal grammar consists of a set of symbols, divided into *terminal symbols* (or just terminals), and *non-terminal symbols* (or just non-terminals), a set of *production rules*, and a *start symbol*. Each production rule allows a string of symbols that fit a certain pattern to be replaced by a different string of symbols. A string s is derivable in a formal grammar if one may begin with the start symbol, follow some sequence of production rules, and arrive at the given string s . The set of all strings derivable within the confines of a formal grammar constitute that grammar’s *language*.

A commonly-used class of formal grammars for music parsing systems are the *context-free grammars*, or *CFGs*. CFGs are popular for parsing music because they permit a wide variety of hierarchical musical constructs to be easily represented within the production rules of the grammar. Every production rule in a CFG must be of the form $A \rightarrow b$, where A is a single non-terminal and b is any string of terminals and non-terminals. Rules following this pattern are called context-free is because no context (additional symbols) are permitted around non-terminal A in the rule, and therefore surrounding symbols in the string cannot govern which rules are appropriate to be used at certain times and which ones are not.

A *parse tree* for a context-free grammar illustrates how a sentence conforms to the rules of the grammar. Production rules of the form $A \rightarrow b$ are illustrated by non-leaf nodes labeled with the non-terminal A which branch into each component of the string b . Figure 6.1 shows how a parse tree for how the sentence “John hit the ball” is understood.

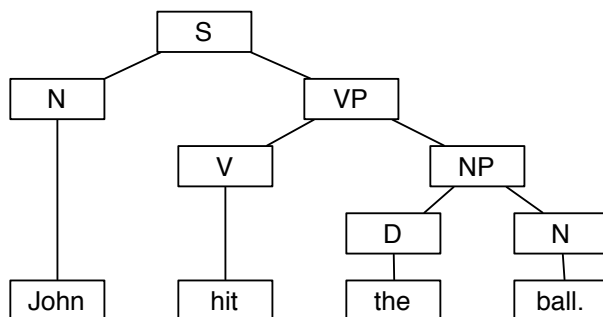


Figure 6.1: A parse tree for the phrase “John hit the ball.” The non-leaf nodes each are labeled with a non-terminal. The labels on the immediate children of non-leaf nodes describe the production rule used at the parent node. For instance, the root node S was expanded using the production rule $S \rightarrow N-VP$. (A sentence is composed of a noun followed by a verb phrase.)

The production rules of some context-free grammars may permit a sentence to be parsed in more than one way; that is, there may be more than one parse tree that can be found for a single sentence that obeys the production rules of the grammar. A CFG by itself, however, has no intrinsic method for discriminating between multiple valid parses of the same string, i.e., determining which one is a “better” parse.

A common strategy that is used when one needs to determine the best parse among a set of candidate parses for a single sentence is to use a *probabilistic context-free grammar*, or *PCFG*. A PCFG is a CFG where each production rule is associated with a probability. The probability for a rule $A \rightarrow b$ can be interpreted as the probability that non-terminal A will be expanded into the string b when A is encountered during a parse; that is, $P(A \rightarrow b | A)$. This is a probability distribution, therefore $\sum_b P(A \rightarrow b | A) = 1$.

Given a string S and a parse tree T , the probability that T is an appropriate parse of S is defined to be the product of the probabilities attached to the n rules used in T :

$$P(T | S) = \prod_{i=1}^n P(A_i \rightarrow b_i | A_i).$$

Therefore, the most probable parse tree for S is the one among all the parse trees which yield string S that also maximizes the expression above.

We can use the PCFG formalism to find the most probable MOP for a given piece of music by relying on two relationships between MOPs and formal grammars. First, the one-to-one correspondence between a MOP and a binary interval tree allows us to re-interpret the latter as a parse tree. Second, our independence assumption from Chapter 5 that allows us to state that the probability of a MOP analysis is equal to the product of the probabilities of the individual triangles within the MOP is the same assumption that is used in deriving the expression above: that the probability of using each individual production rule does not depend on using (or not using) any other production rules.

We will illustrate the correspondence between MOPs and parse trees with an example from earlier in this dissertation. Consider the melodic passage from Chapter 3, presented here as Figure 6.2: a descending five-note melodic line from D down to G, interpreted as outlining a G-major chord. This passage, when structured as a binary tree of intervals, would appear as in Figure 6.3. Each non-leaf node in this tree can be interpreted as a production rule at work, similarly to how we interpret Figure 6.1. For instance, the root node with its left and right children corresponds to the production rule $D-G \rightarrow (D-B)(B-G)$. The probability attached to this production rule corresponds to the probability learned for the triangle $D-B-G$ (see Figure 6.4) through the techniques described in the previous chapter. Therefore, the probability that Figure 6.3 is the correct musical parsing of the phrase $D-C-B-A-G$ is the product of the three probabilities corresponding to the three prolongations in the tree.

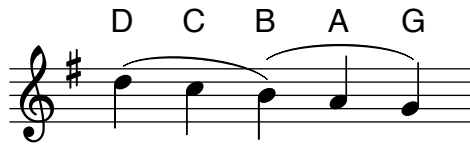


Figure 6.2: An arpeggiation of a G-major chord with passing tones.

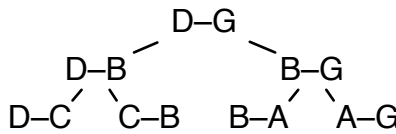


Figure 6.3: The prolongational hierarchy of a G-major chord. Each use of a production rule corresponds to a non-leaf node and its two children.

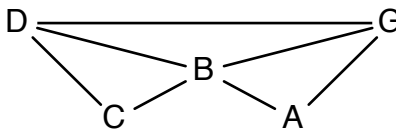


Figure 6.4: The prolongational hierarchy represented as a MOP. Each use of a production rule corresponds to a triangle.

6.2 The ParseMOP algorithm

We have created an algorithm that can accept a monophonic string of notes as input and produce the most probable MOP analysis according to the formula above. The algorithm is based on the CYK algorithm used to parse CFGs, adapted to both (1) take probabilities into account, and (2) permit *ranking* the parses efficiently rather than just finding the single most probable parse (Jurafsky and Martin, 2009; Jiménez and Marzal, 2000). The traditional CYK algorithm is based around parse trees, but because we use the MOP representation, our version of the algorithm, called PARSEMOP, is structured as a search for an optimal polygon triangulation, which the reader will recall is an equivalent way of interpreting a MOP analysis.

PARSEMOP uses dynamic programming to optimally triangulate successively larger sections of the MOP. The algorithm maintains two counters, i and j , which correspond to notes of the input music, as well as vertices on the perimeter of the MOP polygon. PARSEMOP attempts to find the most probable way to triangulate the sub-polygon formed by the vertices $i, i + 1, \dots, j$, assuming that the all the notes between i and j must be produced by a sequence of production rules beginning from a non-terminal N . This is done by searching through all possible notes k between $i + 1$ and $j - 1$, and all production rules of the form $N \rightarrow AB$ where A and B are either terminals (i.e., notes), or other non-terminals. There is one production rule for every possible combination of parent interval N and child intervals A and B as determined by the input music. These rules are not explicitly stored in memory; instead, probabilities are computed on the fly by using the appropriate random forest ensemble classifiers as described in Chapter 5. The parse with the highest probability is recorded in a table T indexed by i , j , and N . Figure 6.5 shows the dynamic programming formulation used by PARSEMOP.

$$T[i][j][N] = \begin{cases} 1 & \text{if } j - i = 1 \text{ and } N \rightarrow ij \text{ is a valid production rule} \\ 0 & \text{if } j - i = 1 \text{ and } N \rightarrow ij \text{ is not a valid production rule} \\ \max_{\substack{\forall k \in [i+1, j-1] \\ \forall N \rightarrow AB}} (T[i, k, A] \cdot P(N \rightarrow AB | N) \cdot T[k, j, B]) & \text{if } j - i \geq 2. \end{cases}$$

Figure 6.5: The dynamic programming formulation used by PARSEMOP.

An example

Suppose PARSEMOP is given the sequence of notes B–C–B–A–G to analyze, as shown in Figure 6.6. We make two simplifying assumptions for this example. First, we will not use the START and FINISH pseudo-events because disregarding them limits the number of possibilities for prolongations we will need to consider; and second, we will not be concerned

with finding an appropriate *Urlinie* for this example. PARSEMOP identifies the *Urlinie* of a musical excerpt by using additional production rules (described later in this chapter), and dispensing with this allows us to simplify table T to use only two dimensions, rather than three.



Figure 6.6: An example five-note sequence used as an example for PARSEMOP.

The probability distributions needed for this example are shown in Table 6.1. The table does not show all of the individual probabilities involved in each distribution, but rather only the ones necessary for this example.

$P(C \mid B, G)$	=	0.1
$P(B \mid B, G)$	=	0.2
$P(A \mid B, G)$	=	0.5
$P(B \mid C, G)$	=	0.2
$P(A \mid C, G)$	=	0.2
$P(B \mid B, A)$	=	0.15
$P(C \mid B, A)$	=	0.1
$P(C \mid B, B)$	=	0.5
$P(B \mid C, A)$	=	0.5

Table 6.1: The probability distributions used in the PARSEMOP example.

PARSEMOP begins by creating five vertices, v_0 through v_4 , corresponding to the five notes in the B–C–B–A–G pattern. Table T, which begins empty, is gradually filled by PARSEMOP in order to calculate the most probable MOP over all the vertices v_0, \dots, v_4 ; this is done by finding the most probable MOPs for increasingly-larger sub-ranges of vertices. PARSEMOP begins by examining pairs of adjacent vertices, then finds the most probable MOP over all sub-ranges of three vertices, then four vertices, then five.

First, PARSEMOP assigns a probability of 1 to $T[0, 1]$, $T[1, 2]$, $T[2, 3]$, and $T[3, 4]$, corresponding to the musical intervals between vertices v_0-v_1 , v_1-v_2 , v_2-v_3 , and v_3-v_4 . The dynamic programming formulation (see Figure 6.5) uses a probability of 1 for all intervals between consecutive notes in the music: such intervals are necessarily at the lowest level of the musical hierarchy and cannot have any prolongations below them.

Second, PARSEMOP examines all spans of three consecutive vertices. For v_0-v_2 (the notes B–C–B at the beginning of the music), there is only one possible middle note, namely v_1 , so because we know that $P(C \mid B, B) = 0.5$, the dynamic programming formulation tells

PARSEMOP to store 0.5 in $T[0,2]$. Similarly, PARSEMOP assigns 0.5 to both $T[1,3]$ and $T[2,4]$ as well because $P(B | C, A) = P(A | B, G) = 0.5$.

Third, PARSEMOP considers all spans of four consecutive vertices. For v_0-v_3 , the algorithm must choose either v_1 (C) or v_2 (B) as the best middle note. Selecting v_1 as the middle note results in a probability of $1 \cdot 0.1 \cdot 0.5 = 0.05$, whereas selecting v_2 as the middle note results in a probability of $0.5 \cdot 0.15 \cdot 1 = 0.075$. Because $0.075 > 0.05$, PARSEMOP chooses v_2 as the preferable middle note for the interval v_0-v_3 , and assigns 0.075 to $T[0,3]$. When examining the range v_1-v_4 , both middle note possibilities v_2 and v_3 result in the same probability of 0.1, so PARSEMOP will select the earlier note, v_2 (A).

The last step is for PARSEMOP to consider all spans of five consecutive vertices, of which there is only one, the entire musical span v_0-v_4 . There are three possible middle notes, and the maximum probability is obtained by choosing v_2 (B), for a probability of $0.2 \cdot 0.5 \cdot 0.5 = 0.05$.

The completed table T for this example is shown in Figure 6.7. Entries in the table show, for each span of vertices v_i-v_j , the maximum probability p obtainable from choosing a middle note to prolong that span of vertices, and the subscript m of the middle note itself that leads to that probability. Figure 6.8 shows the most probable MOP for our musical sequence B-C-B-A-G, along with the other four possible MOPs that have lower overall probabilities.

		j			
		1	2	3	4
i	0	$p = 1$	$p = 0.5$ $m = 1$	$p = 0.075$ $m = 2$	$p = 0.05$ $m = 2$
	1		$p = 1$	$p = 0.5$ $m = 2$	$p = 0.1$ $m = 2$
	2			$p = 1$	$p = 0.5$ $m = 3$
	3				$p = 1$

Figure 6.7: The complete table of partial MOP probabilities computed by PARSEMOP during the example.

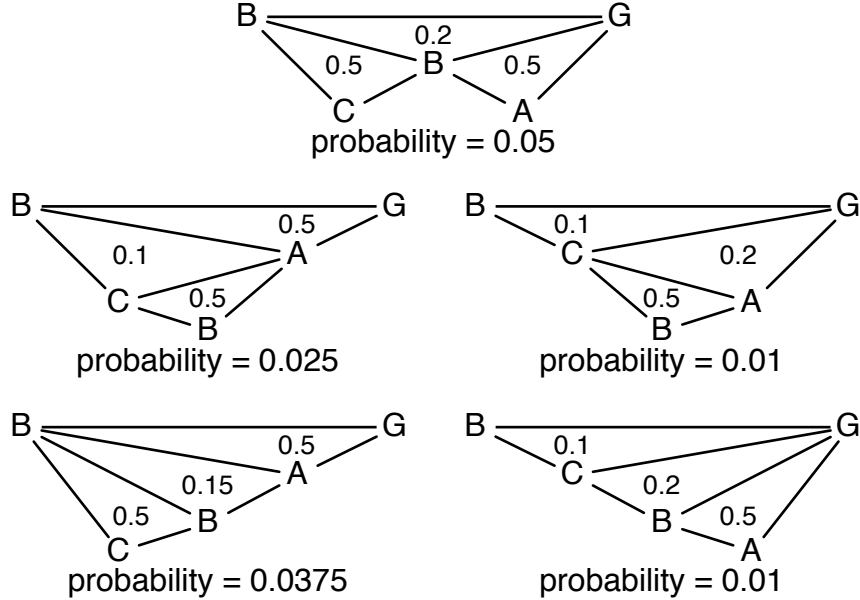


Figure 6.8: The five MOPs possible for the musical sequence B–C–B–A–G and their corresponding probabilities.

The full implementation of PARSEMOP runs in $O(n^3)$ time, where n is the number of input notes, as the algorithm requires three nested loops: one each for i , j , and k . There is a fourth nested loop to examine each production rule N as well, but as the number of production rules does not depend on the input size, we do not include it in our complexity calculation.

We examined three variations of the PARSEMOP algorithm that use slightly different sets of production rules: the differences all relate to how the “upper levels” of the musical hierarchy — the *Urlinie* and any initial ascents or arpeggiations — are parsed.

The first variation, known as PARSEMOP-A, uses the same set of production rules at all levels of the musical hierarchy. This set of rules allows any musical interval to be expanded into any other pair of musical intervals without restriction. Because of the context-free nature of the grammar and the mechanism of the parsing algorithm, PARSEMOP-A cannot force the background structure of the most probable parse to conform to an *Urlinie* pattern or to any specific contour — the parse cannot discern when it is at the upper levels of the musical hierarchy and therefore cannot enforce any sort of specific structure at the upper levels of the hierarchy. Therefore, parses produced by PARSEMOP-A often fail to find an *Urlinie* even when there is one in the ground-truth interpretation of a note sequence.

In contrast, the PARSEMOP-B algorithm accepts not only a sequence of notes as input, but also information specifying exactly which notes of the input sequence should be part of the background structure (*Urlinie* and any initial ascent or arpeggiation). A set of production rules enforcing the specific background structure desired is used for each individual piece of

music. Thus PARSEMOP-B will always find the correct background structure of a piece, but is only useful in situations where this information is known beforehand.

The PARSEMOP-C algorithm is a compromise between the A and B algorithms to better reflect the real-world scenario of being able to identify the contour of the correct background musical structure for a piece ahead of time but not which specific notes of the piece will become part of that structure. While the input to PARSEMOP-B is a sequence of notes, some of which are specifically identified as belonging to the background structure, PARSEMOP-C accepts the same note sequence but along with only the *names* of the notes, in order, that belong to the background structure. For example, given the sequence of notes E-F-D-C-E-D-B-C with the correct background structure underlined, PARSEMOP-B must be informed ahead of time that the first, sixth, and eighth notes of the input must appear at the uppermost levels of the musical hierarchy. PARSEMOP-C, on the other hand, is provided only with the information that the background must contain the notes E-D-C in that order, but will not know *which* E, D, and C are the “correct” notes.

The traditional probabilistic CYK algorithm is designed only to find the most probable parse of a sentence. As we would like to produce a list of musically-plausible parses ranked by descending probability, we need an additional algorithm. Jiménez and Marzal (2000) showed that it is feasible to obtain this ranked list of parses by augmenting the CYK algorithm with some additional data structures, and also provided an efficient procedure, NEXTTREE, to iterate through the list without storing all of the possible parses at once. In the next section, we discuss the qualitative results of applying the PARSEMOP and NEXTTREE algorithms to our corpus of data, along with evaluation metrics and quantitative data illustrating the performance of the three PARSEMOP variations.

CHAPTER 7

EVALUATION

In this chapter, we evaluate the quality of our probabilistic model of music analysis from Chapter 5 by studying — both qualitatively and quantitatively — the analyses that the PARSEMOP algorithm from Chapter 6 produces for the music in our corpus from Chapter 4. We also evaluate the utility of providing PARSEMOP with prior information about the structure of the *Uralinie*. Specifically, we show the results of four experiments, which (a) quantitatively compare analyses produced by PARSEMOP to corresponding analyses from textbooks, (b) show the locations within analyses produced by PARSEMOP where the algorithm is more likely to make a mistake, (c) illustrate how the accuracy of the analyses produced by PARSEMOP changes as one moves through a list of analyses ranked by probability, and (d) use experienced music theorists to judge the analyses produced by PARSEMOP.

Due to the difficulty of finding additional musical excerpts with corresponding analyses to use as a testing data set, coupled with the small size of the corpus (41 musical excerpts), we used a leave-out-one cross-validation approach for training and testing in these experiments. Specifically, for each excerpt in the corpus, we generated a training set consisting of the music from the other 40 excerpts, trained the probabilistic model on these data, and used each PARSEMOP variant to produce a list of the top 500 analyses for the original excerpt that was left out. The music notation for each excerpt, each excerpt’s textbook analysis, and all of the PARSEMOP-produced MOPs, are provided in Appendix A.

As we mentioned in Chapters 1 and 2, even experts in Schenkerian analysis sometimes disagree on the “correct” analysis for a composition. It is therefore possible to have more than one musically-plausible hierarchical analysis of an excerpt; occasionally these various analyses will differ radically in their hierarchical structures. However, due to limited data, our experiments rely on using a single textbook analysis as ground truth. This sometimes leads to PARSEMOP finding alternative correct analyses that are not counted as such in the purely quantitative experiments. One example of this phenomenon is revealed in our last experiment which uses humans to judge the PARSEMOP analyses.

Through all of these experiments, we demonstrate that

- analyses produced by PARSEMOP by using the probabilistic MOP model and varying amounts of prior knowledge about the *Uralinie* achieve average accuracy levels between 64% and 90% (using one accuracy metric), whereas an analysis generated uniformly at random would only achieve an accuracy level of 22%,

- the locations of errors in the most-probable PARSEMOP analyses are correlated with the musical texture of the excerpt and the PARSEMOP variant used, and
- on average, according to human music theorists, PARSEMOP-produced analyses are only between half a letter and a full letter grade worse than expert-produced analyses, on an A–F scale.

7.1 Evaluation metrics

We use two similar evaluation metrics for determining the accuracy of a PARSEMOP analysis. In natural language processing systems that produce parse trees as output, the traditional methods for computing the accuracy of a candidate parse tree is to compare it against a reference parse tree. Specifically, one counts the number of internal nodes of the candidate parse tree that are identical (in both content and location) with internal nodes in the reference parse tree. For comparing a candidate MOP against a reference MOP, the equivalent procedure is to count the number of triangles in a PARSEMOP-produced MOP analysis for which there is an identical triangle in the corresponding textbook MOP analysis. Therefore, we will define the *triangle accuracy* of a candidate MOP as the percentage of triangles in the candidate MOP that match with a triangle in the equivalent textbook MOP:

$$\text{TriangleAccuracy}(M) = \frac{\# \text{ of correct triangles in candidate MOP } M}{\# \text{ of total triangles in textbook MOP for } M}$$

In addition, we can compute this accuracy metric based on corresponding *edges* between the candidate and reference MOPs rather than entire triangles. The necessity for an additional metric is made clear by the situation illustrated in Figure 7.1, where two MOPs share an internal edge, indicating a melodic connection between two non-adjacent notes. However, if one of the MOPs is denoted the candidate MOP and the other the reference MOP, the triangle-based accuracy metric is zero because the two MOPs have no triangles in common, an overly harsh judgement that neglects the internal edge that the two MOPs have in common (edge 2–5). Therefore, we will also report the fraction of internal edges that match between the candidate and reference MOPs.

Calculating triangle or edge accuracy from a candidate MOP and a reference MOP is straightforward except for the possibility of untriangulated subregions inside the reference MOPs (candidate MOPs produced by PARSEMOP are always fully-triangulated). To handle this situation, we will modify our definitions for “correct” triangles and edges as follows. A “correct” triangle in a candidate MOP is a triangle that either (1) matches exactly with a triangle in the reference MOP, or (2) could fit in an untriangulated area of the reference MOP. In other words, triangles are correct if they appear in the reference MOP or could hypothetically appear, if the reference MOP were completely triangulated. Correct edges are defined analogously.

While it may seem that compensating for untriangulated regions in this fashion could distort accuracy statistics, we take this into account by providing a randomized reference

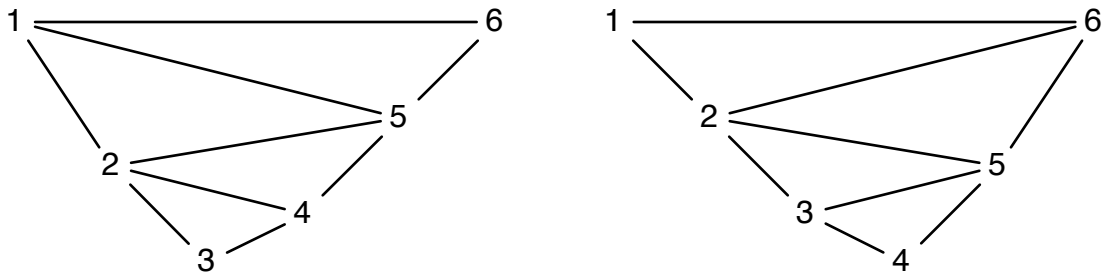


Figure 7.1: These two MOPs share no common triangles, yet have an interior edge in common. The edge 2–5 in both MOPs indicates that the motion from note 2 to note 5 is important melodically, and even though both MOPs contain this edge, a triangle-based accuracy metric will report an accuracy of zero.

triangulation that is scored in the same manner, providing a baseline level of accuracy for comparison. Furthermore, discounting triangles from each excerpt’s *Urtle* (whose special handling for evaluation is described in the next section), untriangulated regions account for only 183 of the 907 triangles in the corpus (about 20%).

7.2 ParseMOP accuracy

Figures 7.2 and 7.3 show the triangle- and edge-based accuracy metrics calculated for each of the three PARSEMOP variants (A, B, and C). Three points are plotted for each excerpt of music, showing the accuracy level as a number between 0 and 1 — the fraction of triangles or edges deemed correct — for each PARSEMOP variant. The gray horizontal bars for each excerpt show a baseline level of accuracy that could be obtained by a hypothetical triangulation algorithm that operated randomly. That is, the baseline accuracy is the average accuracy for all possible triangulations of the excerpt (this quantity can be calculated analytically, without resorting to sampling, by using the mathematical ideas from Chapter 3.) In the figure, filled-in shapes are used for each point where the difference between the PARSEMOP accuracy and the baseline accuracy is statistically significant at the 0.05 level. p -values were calculated using individual binomial tests under the null hypothesis that the PARSEMOP accuracies did not differ significantly from the baseline accuracy.

Recall that the three PARSEMOP variants use different levels of *a priori* information to do their work: PARSEMOP-A uses no extra information, PARSEMOP-B is provided the exact positions of the notes of the background structure, and PARSEMOP-C is given only the pitches and relative ordering of the background structure’s notes, but not the notes’ exact positions. Therefore, because PARSEMOP-C is a “compromise” between the no-extra-information-given PARSEMOP-A and the big-head-start-given PARSEMOP-B, one would expect to see this reflected in the graphs by having the accuracy level for PARSEMOP-C on a given excerpt of music be between the levels obtained from PARSEMOP-A and PARSEMOP-B, and indeed, this is the case for a large number of pieces. In the graphs, this situation

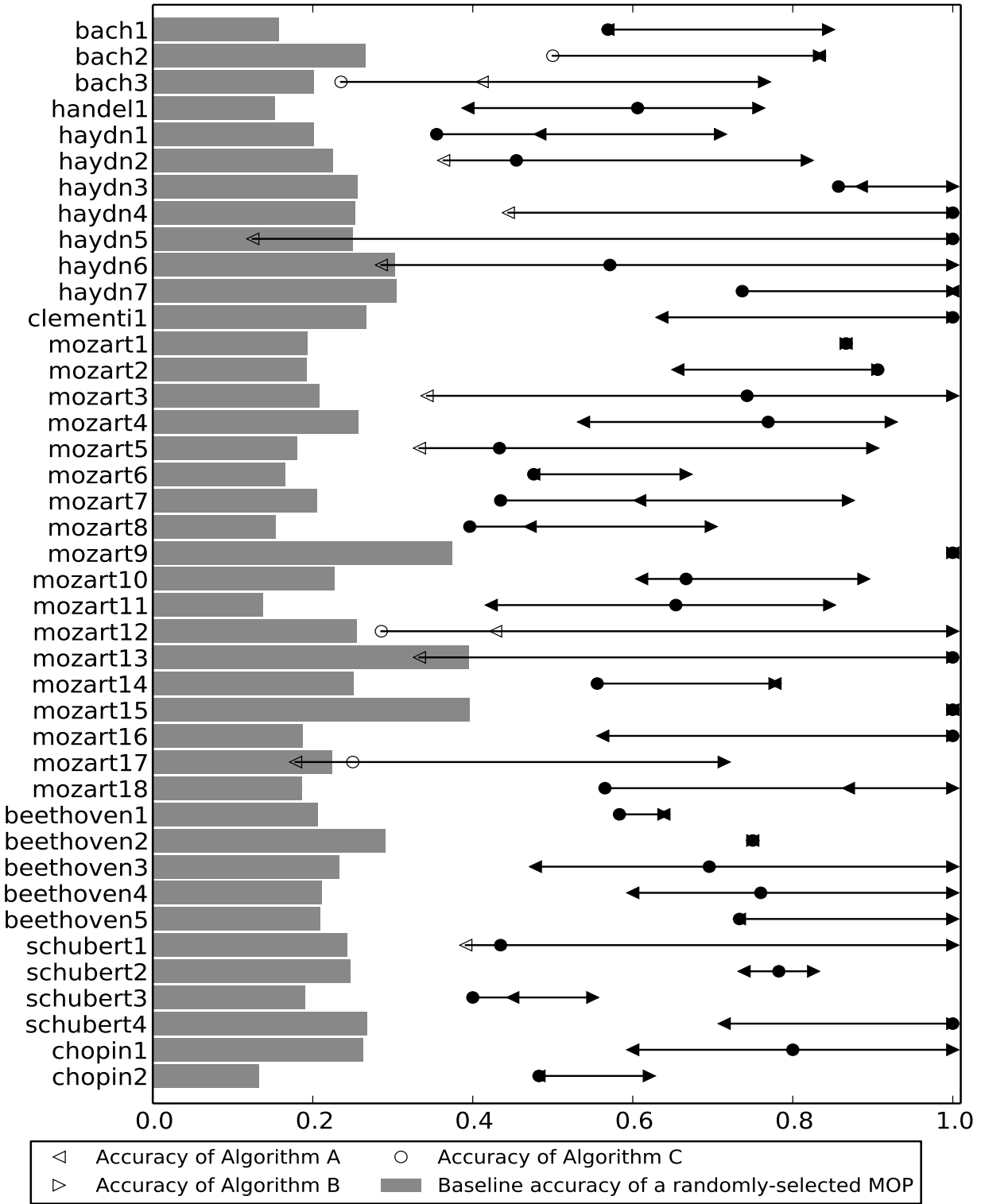


Figure 7.2: Triangle accuracy for the three PARSEMOP variants.

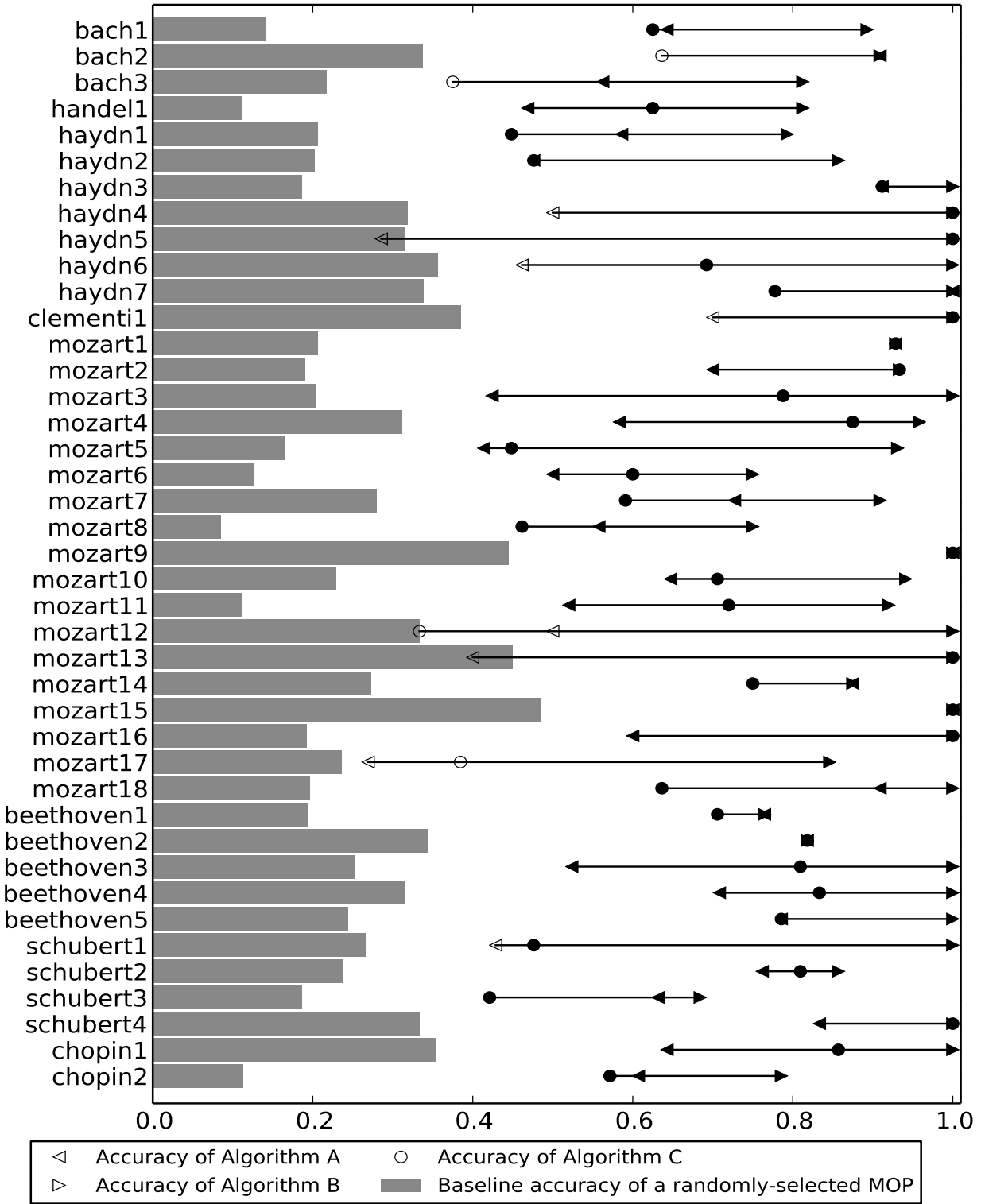


Figure 7.3: Edge accuracy for the three PARSEMOP variants.

is visually depicted by the points shown as circles (PARSEMOP-C) being located between the points shown as left- and right-pointing triangles (PARSEMOP-A and PARSEMOP-B). However, there are some situations where PARSEMOP-C performs worse than PARSEMOP-A. One reason for this is the evaluation metrics for PARSEMOP-C (and PARSEMOP-B) use a ground-truth MOP that has the upper-level melodic structure (the *Urlinie*) completely triangulated. This triangulation, however, is produced algorithmically because the textbook analyses do not contain any information on how the *Urlinie* should be triangulated (this information is typically not illustrated at all in Schenkerian notation). Furthermore, the algorithmic triangulation does not necessarily correspond to the “correct” musical interpretation of the hierarchy of the notes within the *Urlinie*, however, the algorithm does ensure that musically-similar background structures will have similar triangulations in the corpus, thereby assisting the machine learning algorithm.

More specifically, there are certain pairs of *Urlinie* patterns that are used to analyze consecutive musical phrases that function together. These patterns are used when the first phrase is analyzed as a descent from the first note of the *Urlinie* but ends one note *before* the end of the *Urlinie*, creating a situation called an *interruption*: the *Urlinie* has seemingly terminated early. However, the second phrase is analyzed by “restarting” the *Urlinie* and having it continue to its natural conclusion. Consider the musical phrase in Figure 7.4; this can be interpreted as an *Urlinie* descending from the fifth note of a musical scale down to the first. Consider a musical phrase that descends from D down to A, then restarts at D and descends again all the way to G. The initial pseudo-ending on A *interrupts* the natural descent of the melody, creating tension that is only resolved after the whole phrase repeats and completes the descent to G.

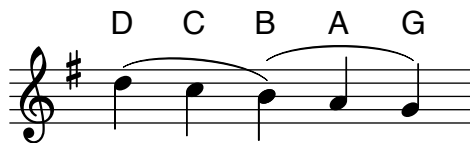


Figure 7.4: An arpeggiation of a G-major chord, now interpreted as an *Urlinie*. An interruption occurs when a musical phrase descends from D to A, then restarts and re-descends from D to G.

From a hierarchical standpoint, an interruption presents a representational challenge because the second-to-last note of the *Urlinie* — the A in Figure 7.4 — contains a great deal of melodic tension because its function is to move from the B to the G, where the tension is resolved. As the first phrase (the interrupted phrase) is usually melodically similar to the second phrase (the completed phrase), we would want the MOPs representing the two phrases to be similar because we would want the machine learning algorithm to identify the structural similarities between the two phrases. However, this is impossible with the missing G in the first phrase. Therefore, in order to maintain similar hierarchical representations of the first and second melodic descents, we chose to triangulate the *Urlinie* in a MOP

according to the order of the notes rather than the true hierarchical structure. This allows for the four-note descent D–C–B–A of the first phrase to be represented exactly the same way as the first four notes of the complete five-note pattern of the second phrase.

Because the PARSEMOP-B and -C algorithms are given *a priori* information about the *Urlinie* of the music in question, these algorithms will always produce an analysis with an *Urlinie* that matches the textbook *Urlinie* exactly (PARSEMOP-B) or at least in melodic contour (PARSEMOP-C). Therefore, it is straightforward to compare such an analysis against the textbook analysis, as both will have an *Urlinie* triangulated according to the same algorithmic pattern. PARSEMOP-A, however, is not given information about the *Urlinie*, and therefore will only find one purely by chance. As a result, it is unfair to compute the accuracy of a PARSEMOP-A analysis by comparing it against a textbook MOP with an algorithmically-triangulated upper region, because any accuracy judgements for triangles or edges in that region would have no musical interpretation. Therefore, when computing accuracies for PARSEMOP-A analyses, we chose to compare against a textbook MOP with an *untriangulated* *Urlinie* region. The effect of this is any time a PARSEMOP-A analysis places a note of the *Urlinie* at a sufficiently high enough level of its hierarchy, it will likely be rewarded with either a correct edge or triangle determination. While this may be a less than perfect solution, it is decidedly better than arbitrarily rewarding or penalizing the PARSEMOP-A analyses as would happen if compared against textbook MOPs with algorithmically-triangulated *Urlinie* regions. Furthermore, the random baseline accuracy is computed in this same fashion, allowing for comparisons.

Discussion

There is a great deal of variance in the accuracy of the PARSEMOP algorithms: variance in how well each algorithm performs individually on different pieces, and also in how each of the three algorithm variants performs on the same piece.

A common cause of poor accuracy is repeated pitches in the input music. Often these repeated pitches are instances of the same musical idea being decorated with intervening notes. While the PARSEMOP algorithms will often correctly identify that the decorative notes are subservient to the surrounding notes, confusion arises when PARSEMOP must relate the repeated pitches to each other; sometimes there is no clear choice regarding which pitch should appear at a more abstract level of the hierarchy.

An example of this confusion occurs in the “mozart17” excerpt, shown in Figure 7.5, along with the corresponding textbook and PARSEMOP-C analyses. Notice that the pitches of the *Urlinie* (the beamed notes in the algorithmic output) are the same in both analyses, and differ only in which E is identified as the “correct” one. The confusion arises here from the arpeggiations of the dominant (V) and tonic (I) chords in the music: there are many repeated pitches with no immediately obvious hierarchical structure among them, including four instances of E in the first four measures. A human analyst would likely collapse the arpeggiations into block chords (as in the bass) and identify the multiple implied voices, but PARSEMOP cannot do this; instead, the algorithm must choose a single E to be part of the

main melody. The G that begins the excerpt and reoccurs throughout the music also causes problems for PARSEMOP as it is uncommon to have the *Urlinie* (in this case the pitches E–D–C) appear below a harmonizing voice; usually the *Urlinie* is the top-most voice in the music.

Figure 7.5: Excerpt of Mozart, KV 103, No. 1, Trio (mozart17), along with the textbook and PARSEMOP-C analyses of the first four measures.

7.3 Error locations

In addition to studying what kinds of errors PARSEMOP makes, it is worthwhile to identify where the errors are being made. In other words, we would like to know if PARSEMOP is making more mistakes at the surface level of the music or at the more abstract levels. We can quantify the notion of “level” by numbering the interior edges of a candidate MOP produced by PARSEMOP with increasing integers starting from zero, with zero corresponding to the most abstract interior edge. Perimeter edges are not assigned a level because all such edges represent either connections between adjacent notes in the input music, or the connection between the START vertex and the FINISH vertex; in both cases, these edges are always present in any MOP and therefore cannot be in error. These integers correspond to the standard idea of vertex depth in a tree, and therefore can be regarded as *edge depths*. We will normalize the edge depths of a MOP by dividing each integer assigned to an edge by the maximum edge depth within the MOP, giving an error location always between 0 and 1, with 0 corresponding to the most abstract edges, and 1 corresponding to those closest to the musical surface.

Figure 7.6 shows the probability for the three PARSEMOP variants to include an incorrect edge at different normalized error depths. Unsurprisingly, the probability of making an error at the most abstract level corresponds exactly to how much extra information the PARSEMOP variant is given about the contour of the main melody of the musical excerpt in

question: PARSEMOP-B has the lowest probability for an error at depths between 0.0 and 0.2, while PARSEMOP-A has the largest. Unsurprisingly, this then skews the error locations for PARSEMOP-B to the lower depths between 0.6 and 1.0, while variants A and C maintain much flatter contours. We suspect the preponderance of errors at the middle depths for variants A and C occurs for a few reasons. First, fewer errors likely occur at the less abstract musical levels (closer to 1.0) because the surface-level music is simply easier to analyze. This is the level where it is easiest to hear the elaborations present in the music as they take place over shorter time spans and often involve only a single harmony. Such elaborations are studied by undergraduates in their first year of music theory and are usually uncomplicated in their presentations.

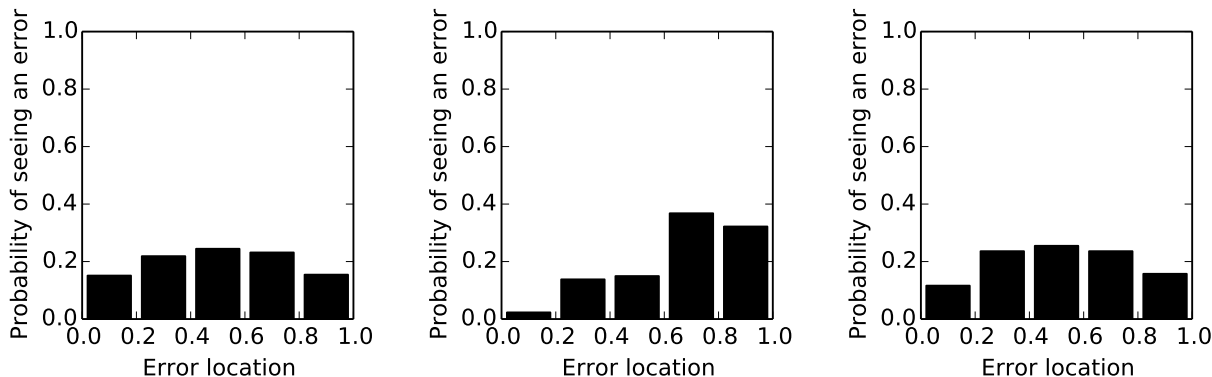


Figure 7.6: Histogram showing the locations of errors for the three PARSEMOP variants over all excerpts in the corpus.

On the other hand, the smaller probability of making a mistake at the highest levels of abstraction for PARSEMOP-A and -C we speculate to be for different reasons. Recall that the evaluation procedure for PARSEMOP-A uses an untriangulated upper region for the most abstract main melody notes, which naturally leads to fewer errors as there are fewer edges with which to find conflicts. However, the PARSEMOP-C algorithm is given the basic melodic contour ahead of time, and this information also reduces high-level errors. PARSEMOP-A and -C likely have the most probability density in the middle areas of the graph because these areas are the most difficult to analyze: one cannot use the desired upper-level melodic contour as a goal, nor does one have the benefit of making the simpler analytic decisions present at the surface-level.

It is instructive to examine the depths of the edges in error on a per-excerpt basis. Table 7.1 shows, for each excerpt and PARSEMOP variant used, where the errors occur in the top-ranked analysis produced by the algorithm on the excerpt. The small histograms in the table can be interpreted similarly to the probability density diagrams in Figure 7.6: the x -axis ranges from 0 to 1 and describes the edge depth where the edge errors are occurring, with 0 meaning the most abstract levels of the hierarchy and 1 meaning the least abstract.

The height of each bar describes how many errors are occurring at each level. The heights of the bars can be compared across the three PARSEMOP variants within each musical excerpt, but not across excerpts.
































































The table also gives the percentage that the accuracy increased when each analysis produced by PARSEMOP is compared to the average accuracy of a randomly-selected analysis (“% Improve”), normalized so a score of 0 corresponds to an accuracy level that is no better than random, and 1 corresponds to a perfect analysis (note that improvement scores of 1 correspond to blank histograms because there are no errors to show). Thus, a score of 0.5 indicates that the PARSEMOP improved 50% of the way from a randomly-selected analysis to a perfect analysis.

Table 7.1: Edge accuracy improvement over random and error locations for each excerpt.

Excerpt	PARSEMOP-A		PARSEMOP-B		PARSEMOP-C	
	% Improve	Errors	% Improve	Errors	% Improve	Errors
bach1	0.58		0.88		0.56	
bach2	0.86		0.86		0.45	
bach3	0.44		0.76		0.20	
handel1	0.40		0.79		0.58	
haydn1	0.48		0.74		0.31	
haydn2	0.34		0.82		0.34	
haydn3	0.89		1.00		0.89	
haydn4	0.27		1.00		1.00	
haydn5	-0.04		1.00		1.00	
haydn6	0.16		1.00		0.52	
haydn7	1.00		1.00		0.66	
clementi1	0.51		1.00		1.00	
mozart1	0.91		0.91		0.91	
mozart2	0.63		0.92		0.92	



















Continued on next page

Table 7.1 — continued from previous page

Excerpt	PARSEMOP-A		PARSEMOP-B		PARSEMOP-C	
	% Improve	Errors	% Improve	Errors	% Improve	Errors
mozart3	0.28		1.00		0.73	
mozart4	0.40		0.94		0.82	
mozart5	0.30		0.92		0.34	
mozart6	0.43		0.71		0.54	
mozart7	0.62		0.87		0.43	
mozart8	0.52		0.73		0.41	
mozart9	1.00		1.00		1.00	
mozart10	0.54		0.92		0.62	
mozart11	0.46		0.91		0.68	
mozart12	0.25		1.00		0.00	
mozart13	-0.09		1.00		1.00	
mozart14	0.83		0.83		0.66	
mozart15	1.00		1.00		1.00	
mozart16	0.50		1.00		1.00	
mozart17	0.04		0.80		0.19	
mozart18	0.89		1.00		0.55	
beethoven1	0.71		0.71		0.63	
beethoven2	0.72		0.72		0.72	
beethoven3	0.36		1.00		0.74	
beethoven4	0.57		1.00		0.76	
beethoven5	0.72		1.00		0.72	

Continued on next page

Table 7.1 — continued from previous page

Excerpt	PARSEMOP-A		PARSEMOP-B		PARSEMOP-C	
	% Improve	Errors	% Improve	Errors	% Improve	Errors
schubert1	0.22		1.00		0.29	
schubert2	0.69		0.81		0.75	
schubert3	0.55		0.61		0.29	
schubert4	0.75		1.00		1.00	
chopin1	0.45		1.00		0.78	
chopin2	0.56		0.76		0.52	

7.4 Maximum accuracy as a function of rank

So far we only have examined the accuracy of the top-ranked analysis produced by the PARSEMOP algorithms for each musical excerpt. However, it is instructive to examine the accuracies of the lower-ranked analyses as well, in order to investigate how accuracy relates to the ranking of the analyses. In particular, we are interested in studying the maximum accuracy obtained among the analyses at ranks 1 through n , where n is allowed to vary between 1 and 500. We would hope that analyses that are judged as being accurate are not buried far down in the rankings, especially when the top-ranked analysis is not perfectly accurate.

Figure 7.7 illustrates how the maximum accuracy changes for each music excerpt as one moves through the ranked list (exact numbers are provided in Appendix B). A few oddities in the graphs are worth mentioning. A single musical excerpt appears to present problems for PARSEMOP-A; this corresponds to the single line on the PARSEMOP-A graphs that is clearly isolated from the other lines on the lower part of the graphs. This is “mozart17,” which was mentioned earlier in this chapter as being particularly difficult to analyze correctly due to multiple repeated notes and that the *Urlinie* is hidden in an inner voice. PARSEMOP-A repeatedly tries to analyze the main voice of this excerpt to include the upper note G, which is just an auxiliary note.

On the other hand, “mozart8” is the most difficult for PARSEMOP-C to analyze: even when examining the top 500 analyses, the maximum accuracy in that set was only 49% by the triangle metric. This excerpt, shown in Figure 7.8, is complicated to analyze due to its highly figured texture and that the primary notes of the *Urlinie* are often located on metrically weak beats, a rather uncommon condition that PARSEMOP, through its learning capabilities,

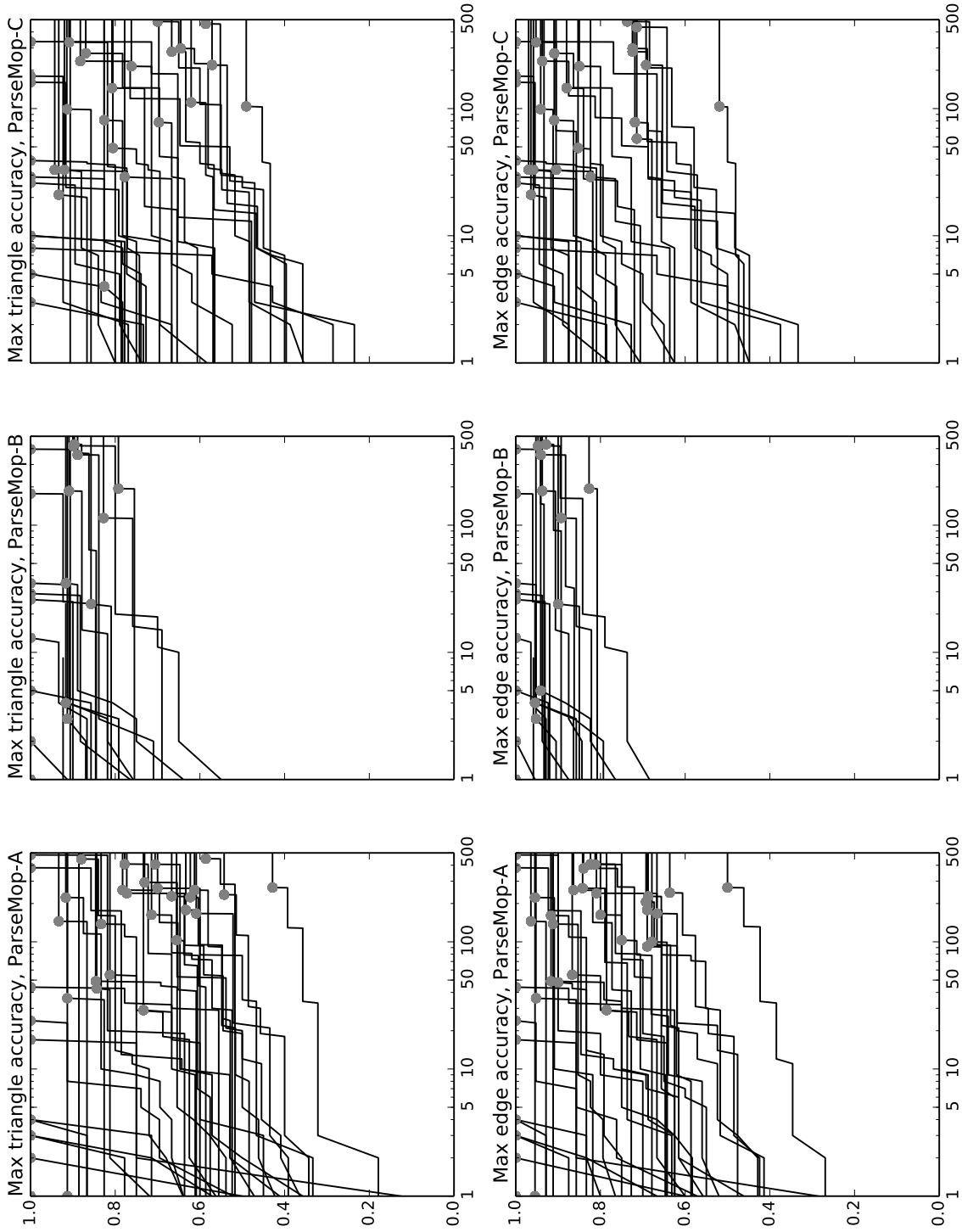


Figure 7.7: Maximum accuracy as a function of rank. Each graph shows, for a particular PARSEMOPvariant, how the maximum accuracy for each piece of music changes as one examines the ranked analyses.

tries to avoid. Figure 7.9 illustrates the correct analysis of this excerpt, while Figure 7.10 shows the top-ranked PARSEMOP-C analysis, clearly locating an incorrect *Urlinie*, though analyzing many individual sections of the music correctly.



Figure 7.8: Excerpt of Mozart, K. 265, Variations on Twinkle, Twinkle Little Star (mozart8).

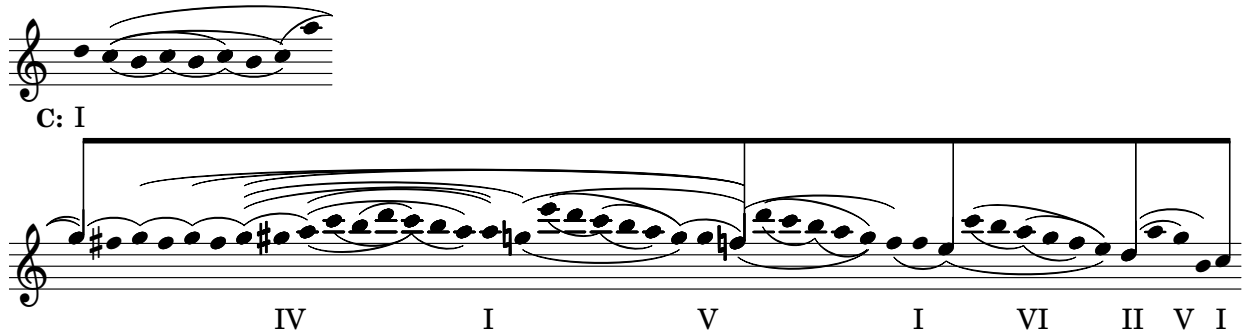


Figure 7.9: Textbook analysis of Mozart, K. 265 (mozart8).

7.5 Human-based evaluation

While it is useful to examine mathematical accuracy metrics, there is no substitute for having human evaluations of the PARSEMOP analyses. In particular, having experienced music theorists evaluate the analyses is indispensable because humans can make both qualitative and quantitative judgements that the accuracy metrics cannot.

For instance, it is certainly the case that some errors in the PARSEMOP analyses are more serious than others. However, it is unclear how to invent a weighting scheme to determine

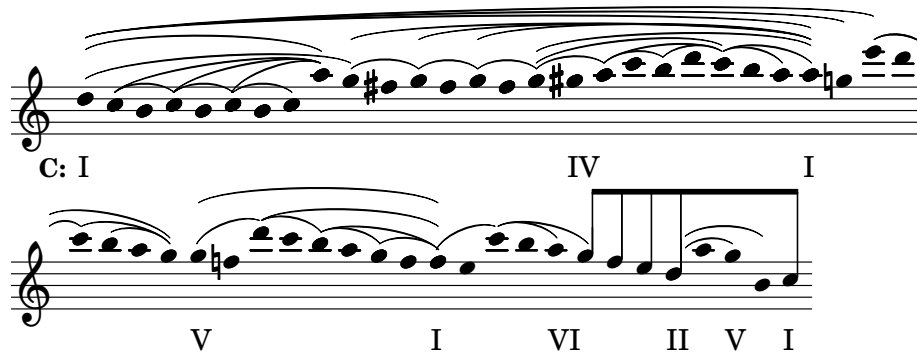


Figure 7.10: PARSEMOP-C analysis of Mozart, K. 265 (mozart8).

which edges or triangles are more important to an analysis than others. In particular, while hierarchical level is certainly a contributing factor to the importance of any particular edge or triangle, mistakes at both the surface and more abstract levels can have large effects on how a person evaluates an analysis. In other words, the triangle and edge accuracy calculations will always operate on a prolongation-by-prolongation basis, but a person may opt to make a more holistic judgement of an analysis.

Another reason for seeking human evaluations is that accuracy metrics based on comparison against a gold standard will necessarily suffer when there are multiple gold standards. Specifically, as we mentioned in the beginning of this chapter, it is possible to have more than one musically-plausible hierarchical analysis of an excerpt. Because a textbook analysis can only present one perspective, we designed an experiment relying on human judgements that removes the idea of comparison against a gold standard entirely.

We recruited three expert music theorists to assist with this experiment. All three are tenured faculty members in music departments at colleges or universities: one at a large public state university, one at a large private university, and one at a small liberal arts college.

In the experiment, we asked the experts to grade pairs of analyses. The order of the musical excerpts in the corpus was randomized and for each excerpt, the graders were provided with the music notation of the excerpt itself, along with the corresponding PARSEMOP-C analysis and the textbook analysis. The graders were not given any information on the sources of the analyses; in particular, they did not know that the two analyses within each pair came from very different places. Furthermore, the order in which the two analyses of each pair were presented on the page was randomized so that sometimes the PARSEMOP analysis was presented first and sometimes second. Both analyses were displayed using a pseudo-Schenkerian notation scheme that uses slurs to illustrate elaborations and beams to show the notes of the main melody and other hierarchically-important notes; this notation can be seen throughout the graphics in this chapter. It is important to note that the textbook analyses were also presented using this notation style; this was done because the output of the algorithm which translates MOPs to pseudo-Schenkerian notation does not yet rival

the true Schenkerian analytic notation used by humans, and so reproducing the textbook analyses verbatim would be too revealing to the graders.

The graders were instructed to evaluate each analysis the way they would evaluate “a student homework submission;” the exact text is provided in Figure 7.11. Each grader was asked to assign a letter grade to each analysis from the set A, A-, B+, B, B-, C+, C, C-, D+, D, D-, F, according to a grading scheme of their own choosing. The goal of this experiment was to determine how much the PARSEMOP-C analyses differ in quality from their textbook counterparts. Note that Figure 7.11 refers to the graders being provided with “ten” excerpts of music; each grader was given the 41 excerpts in batches with approximately ten analyses per batch.

Below are ten excerpts of music from the common practice period, along with musical analyses of those excerpts. The analyses are supposed to evoke the style of Schenkerian analysis, though they do not claim to emulate the process exactly, and therefore do not show all of the traditional Schenkerian notation.

Each page contains the music of the excerpt itself, along with two separate Schenkerian-style analyses of the excerpt. (Note that some analyses are on a single system, while others extend over two systems.)

For each analysis, please evaluate the quality of the music analysis as you would do if you were evaluating a student homework submission. Please assign each analysis a grade chosen from the set: A+, A, A-, B+, B, B-, C+, C, C-, D+, D, D-, F. You have complete control over how you assign grades to analyses.

Figure 7.11: The directions provided to the human graders for judging pairs of analyses.

Figure 7.12 illustrates how the graders judged the PARSEMOP-C analyses and the textbook analyses. For each musical excerpt, the judgements of the three graders are shown as polygons: filled-in polygons for the textbook grades, and hollow polygons for the PARSEMOP grades. A solid line connects the three grades for each excerpt and analysis type.

The data show many interesting situations worthy of further investigation. For instance, there are times when the three graders give good grades to both the PARSEMOP and textbook analyses, yet the accuracy metrics from earlier in this chapter judge the PARSEMOP analysis as being highly inaccurate. This can be interpreted as PARSEMOP locating a musically-plausible “alternate” analysis.

Consider, for instance, the “mozart5” excerpt presented in Figure 7.13. The textbook and PARSEMOP-C analyses of this excerpt from Mozart’s Piano Sonata #7 are given in Figures 7.14 and 7.15. Both analyses feature a clear descent from G down to C — both

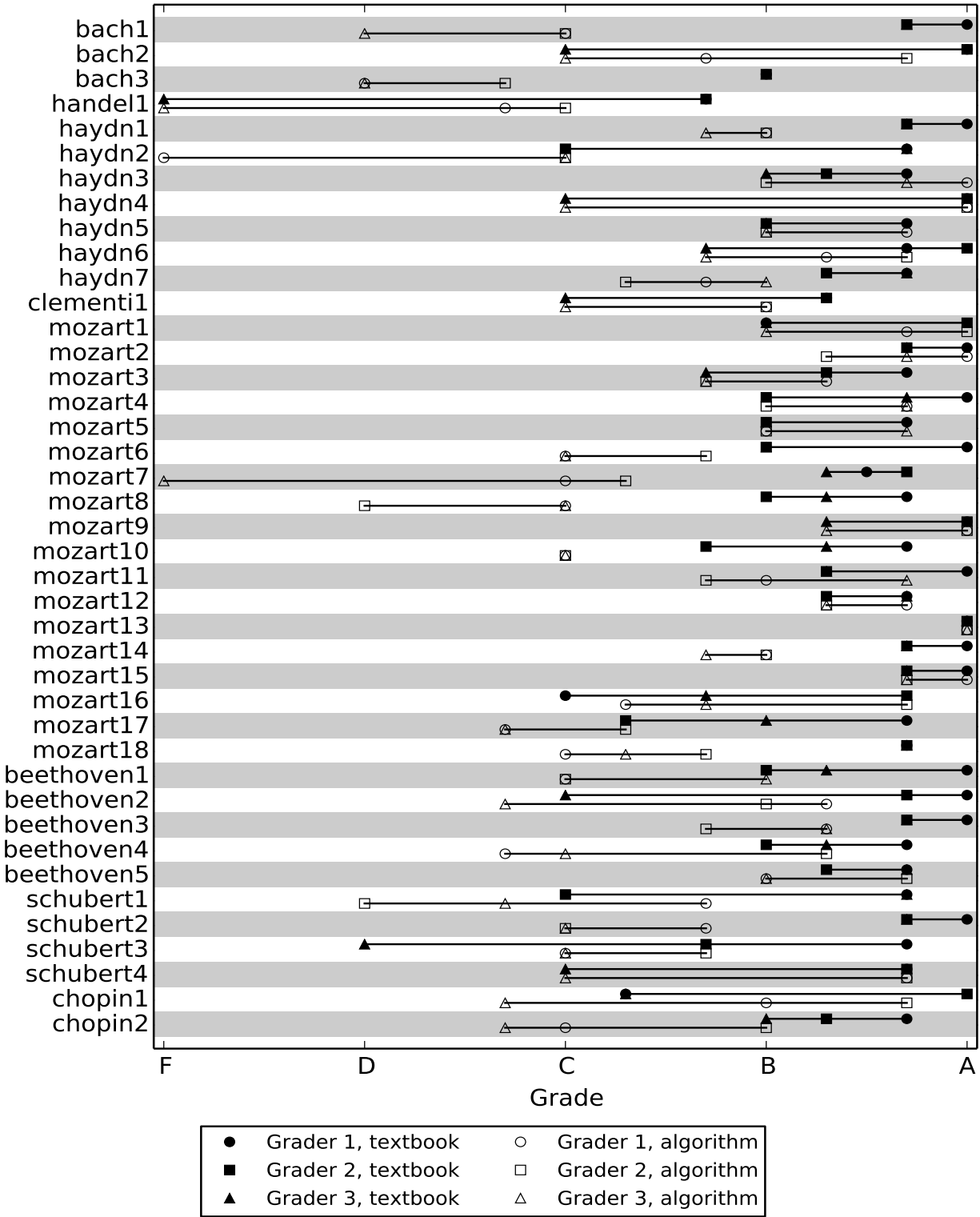


Figure 7.12: Grades assigned by human judges to the textbook analyses and algorithmically-produced analyses from PARSEMOP-C.

identify the G in measure 3 as the primary tone of the *Urlinie* — but after that, their paths diverge.



Figure 7.13: Excerpt of Mozart, Piano Sonata #7 in C major, K309, I (mozart5)

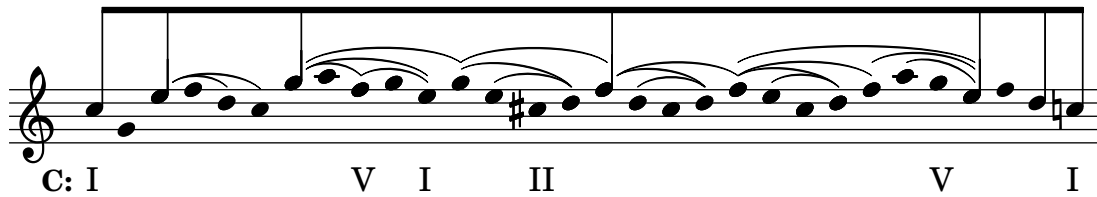


Figure 7.14: Textbook analysis of mozart5.

The textbook analysis chooses to analyze the entire timespan of measures 3–5 as an offshoot of the primary tone G, identifying the melodic descent G–F–E within those measures as a descent into an inner voice that then rises back to the G in measure 5. All of measure 6 is analyzed as being derived from the note F in that measure, which is plausible as that measure is expressing dominant harmony.

In contrast, PARSEMOP-C identifies the G–F–E pattern in measures 3–5 as the first three notes of the *Urlinie*. This results in the notes in measure 6 being analyzed as derived from the D in that measure, rather than the F, which is seen as an additional voice above the main melody at that point. We submit that either analysis is plausible in this situation; there are no major clues in the music that one of the interpretations is preferable.

We can also compare the three graders pairwise to discover patterns in their grading styles. Figure 7.16 shows six contingency tables illustrating how common it is for a pair

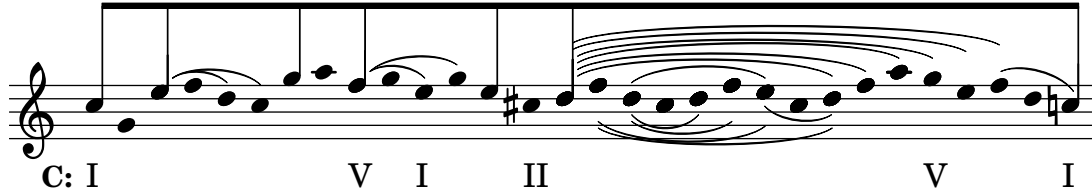


Figure 7.15: PARSEMOP-C analysis of mozart5.

of graders to assign various combinations of grades to the same excerpt. It is clear from the figure that the textbook analyses are usually given better grades than the PARSEMOP analyses: the general clustering is in the upper left corner for the textbook graphs but is more spread out for the PARSEMOP graphs. Apparently, not only are there differences in the way experts analyze music, but there are differences in the way experts *evaluate* analyses of music.

If one converts the A–F grades to a standard 4.0 grading scale (A = 4, B = 3, C = 2, D = 1, F = 0, with a plus adding an additional 0.3 points and a minus subtracting 0.3 points), the average grades assigned to the textbook analyses by the three graders are 3.67, 3.35, and 3.03, respectively, or roughly between a B and just shy of an A-. The corresponding average PARSEMOP grades are 2.78, 2.87, and 2.42, or somewhere between a C+ and a B-. The average differences (the amount the textbook analyses are preferred over the PARSEMOP analyses) are 0.89, 0.48, and 0.61, meaning that the graders preferred the textbook analyses by somewhere between half a letter grade and a full letter grade.

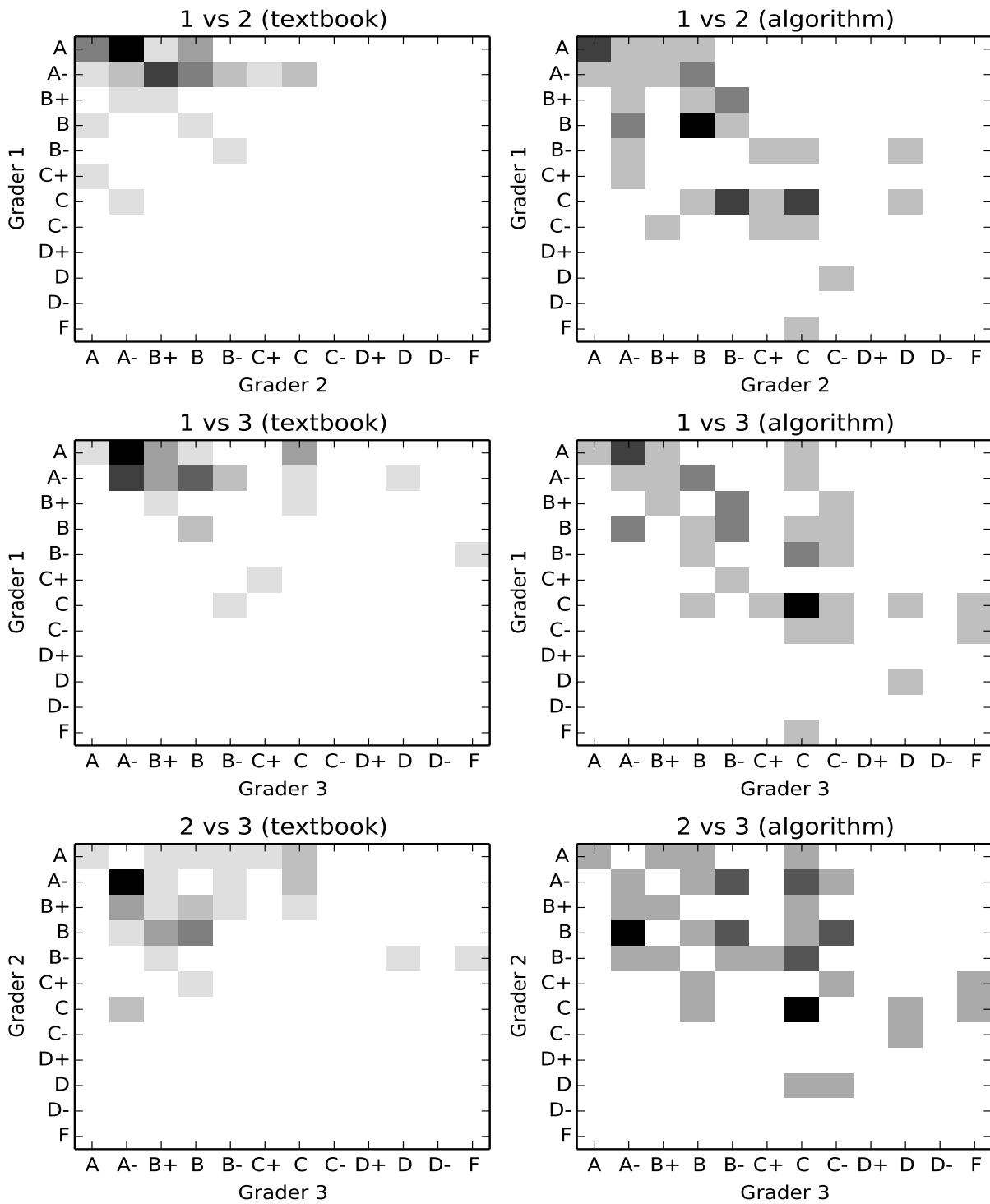


Figure 7.16: Contingency tables for the human evaluations.

CHAPTER 8

SUMMARY AND FUTURE WORK

In this dissertation, we have demonstrated that there are consistent patterns in the way that people perform Schenkerian analysis, and that it is both feasible and practical to (a) model the analysis process probabilistically, and (b) analyze new music compositions using an algorithm derived from the model. In this chapter, we will summarize the major contributions of this work and discuss the possibilities for future research.

We began with the task of modeling the process of Schenkerian analysis and turned to maximal outerplanar graphs, or MOPs. We illustrated how this representation can concisely represent the prolongational structure of a hierarchical music analysis. Next, we augmented the purely representational MOP structure with a probabilistic interpretation of its construction and illustrated how making a strong assumption about independences among the probabilities allowed us to develop a computationally-efficient inference algorithm to locate the most probable MOP. Combining this algorithm with the first computer-interpretable corpus of Schenkerian analyses, we produced three variants of an algorithm for “parsing” excerpts of music to identify the most probable analysis using various amounts of *a priori* information.

In analyzing these algorithms, we learned that overall, the amount of *a priori* information given to the parsing algorithm has a large effect on the errors in the resulting MOP analyses, and also affects where these errors occur within the analyses. We also learned that the texture of a musical excerpt can greatly effect parsing accuracy; namely repeated notes, arpeggiations, and syncopation (important melody notes occurring on weak beats) often lead to mistakes in the analysis. We used three experienced music theorists to grade the algorithmic output of the parsing algorithms, and we concluded that while each person has their own idiosyncrasies in their grading methods, the algorithmically-produced analyses averaged between a C+ and a B-, while the textbook analyses averaged between a B and an A-.

Our majors contributions are the following:

- A novel probabilistic model used to represent hierarchical music analyses,
- Evidence that making a strong independence assumption about the probabilistic model does not alter the model’s usefulness in ranking candidate analyses,
- Efficient algorithms for (a) choosing a MOP uniformly at random from all possible MOPs for a given monophonic note sequence, and (b) iterating through all possible MOPs for such a sequence,

- The first corpus of computer-interpretable Schenkerian analyses,
- An algorithm for determining the most probable MOP analysis for a given piece of music, and
- A study using three human judges to determine the quantifiable differences between human-produced and algorithmically-generated analyses.

This computational model has extensive potential for further research and application. We suspect the accuracy of the analyses produced by PARSEMOP could be raised by using a larger corpus of Schenkerian analyses for training. Though it should not be overlooked that the corpus created for this project is the first of its kind, it is still small when compared to other corpora for machine learning, only containing 41 music excerpts and analyses. Because of the cost of encoding analyses by hand, it is worth investigating if semisupervised or unsupervised learning techniques can be used instead of a purely supervised approach. This technique has been used before — specifically, using an expectation-maximization algorithm with unlabeled corpus — to train a grammar for understanding the structure of melodies (Gilbert and Conklin, 2007).

A natural next step for improving the probabilistic model itself is to include support for multiple voices. At the moment, the model assumes all of the notes of the input music constitute a single monophonic voice; a more sophisticated model capable of correctly interpreting polyphonic music — music with multiple notes sounding at once — would be extremely desirable. Many of the errors in the current PARSEMOP results are due to the inability of the model to handle multiple simultaneous voices implied through compositional techniques such as arpeggiation, and therefore a fully polyphonic model is necessary to bring the model to the next level of musical understanding.

One fascinating application of this model is in the field of algorithmic composition, specifically, the idea of using the hierarchical structure of a composition to create a musical variation of the original input music. For a highly-constrained type of composition, such as a Bach chorale or a Scott Joplin ragtime-style piece, it may be possible to combine the compositional rules with this probabilistic model to algorithmically compose a variation on a piece of music in a new style.

Other clear uses of this model include systems that need to calculate metrics for music similarity, such as applications for music recommendation or new music discovery. There are also potential uses in intelligent tutoring systems for teaching music composition or Schenkerian analysis itself, or in music notation software.

APPENDIX A

MUSICAL EXCERPTS AND MOPS

This appendix contains, for each musical excerpt in the corpus,

- the score of the excerpt in standard music notation,
- the textbook analysis of the excerpt, algorithmically converted into pseudo-Schenkerian notation,
- the top-ranked MOP produced by PARSEMOP-C, obtained under leave-one-out cross-validation, algorithmically converted into pseudo-Schenkerian notation, and
- the top-ranked MOPs produced by PARSEMOP-A, -B, and -C under leave-one-out cross-validation.

The musical scores and the pseudo-Schenkerian diagrams are the same ones provided to the three graders who were tasked with evaluating the PARSEMOP output. The diagrams were not altered except to adjust the positioning of slurs in cases where many slurs overlapped.

Excerpt identifier: bach1

MOP(s) contained in this excerpt: bach1a, bach1b

Score:

Textbook analysis:

PARSEMOP-C analysis:

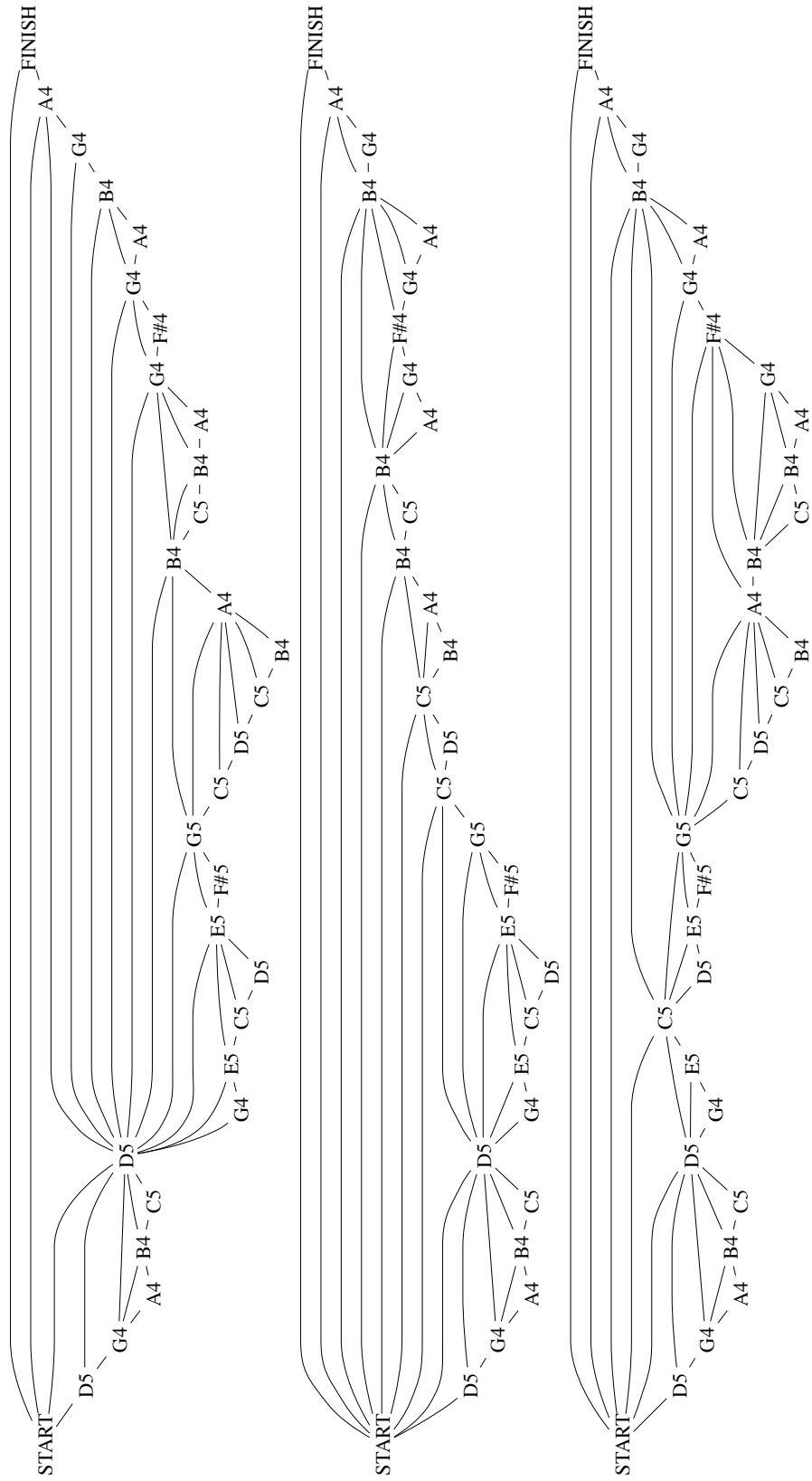


Figure A.1: MOPs produced by PARSEMOP-A, -B, and -C for bach1a

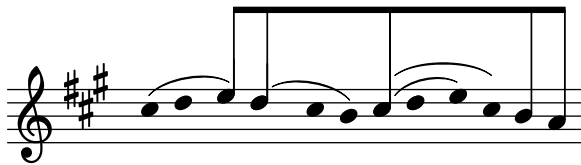
Excerpt identifier: bach2

MOP(s) contained in this excerpt: bach2a

Score:

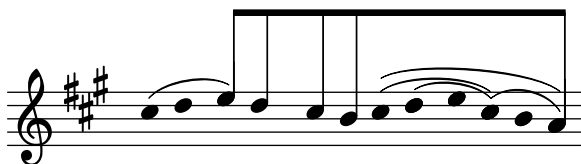


Textbook analysis:



A: I IVI VIII VI IVVI VI

PARSEMOP-C analysis:



A: I IVI VIII VI IVVI VI

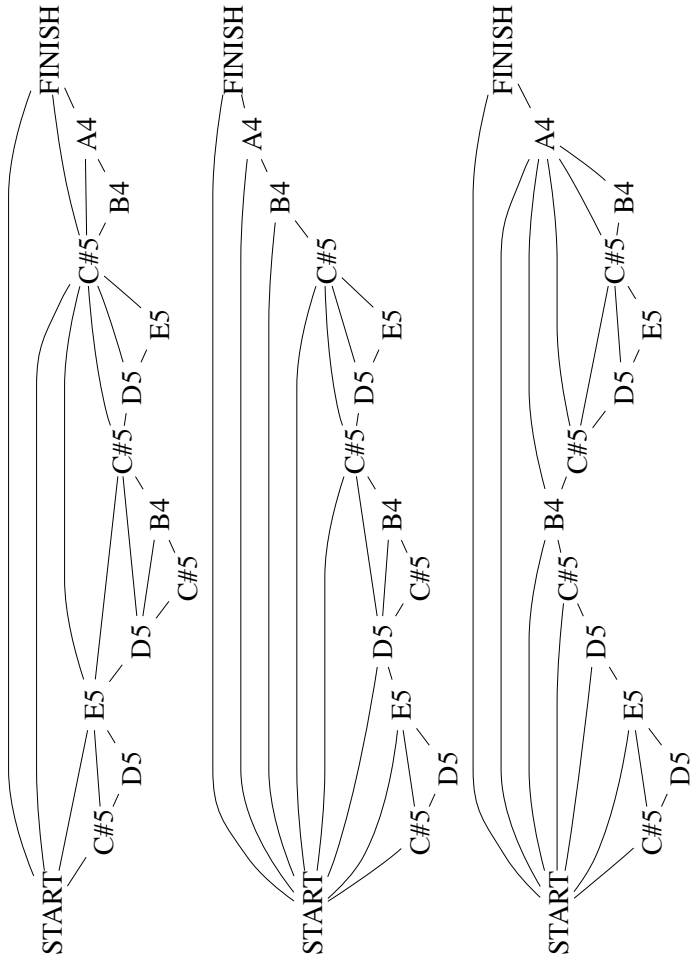
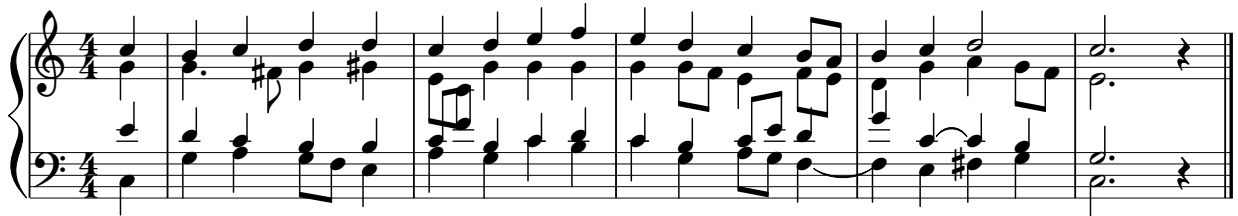


Figure A.3: MOPs produced by PARSEMOP-A, -B, and -C for bach2a

Excerpt identifier: bach3

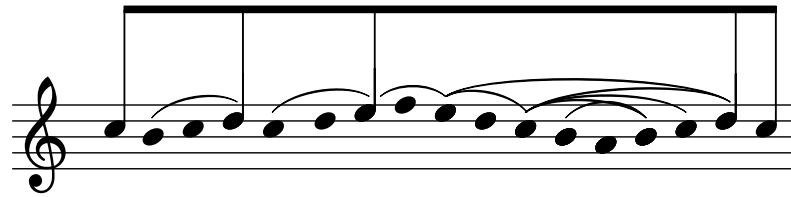
MOP(s) contained in this excerpt: bach3a

Score:



A musical score for a piano excerpt in 4/4 time. The score consists of two staves: a treble staff and a bass staff. The treble staff begins with a treble clef and a 4/4 time signature. The music features a series of chords and melodic lines, including a prominent sequence of eighth notes in the right hand. The bass staff provides a harmonic accompaniment with chords and a steady bass line.

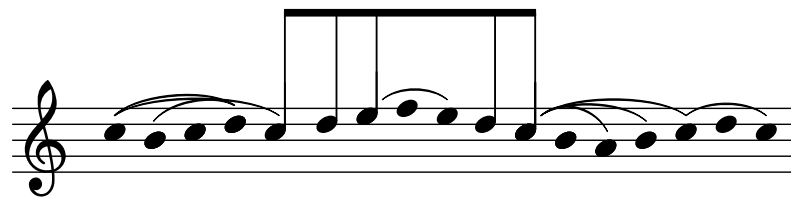
Textbook analysis:



A textbook analysis of the excerpt, showing a single melodic line on a treble staff. The line consists of eighth notes, with a bracketed section indicating a specific intervallic pattern. The notes are: C4, D4, E4, F4, G4, A4, B4, C5, B4, A4, G4, F4, E4, D4, C4.

C: I V V I V I VI VI

PARSEMOP-C analysis:



A PARSEMOP-C analysis of the excerpt, showing a single melodic line on a treble staff. The line consists of eighth notes, with a bracketed section indicating a specific intervallic pattern. The notes are: C4, D4, E4, F4, G4, A4, B4, C5, B4, A4, G4, F4, E4, D4, C4.

C: I V V I V I VI VI

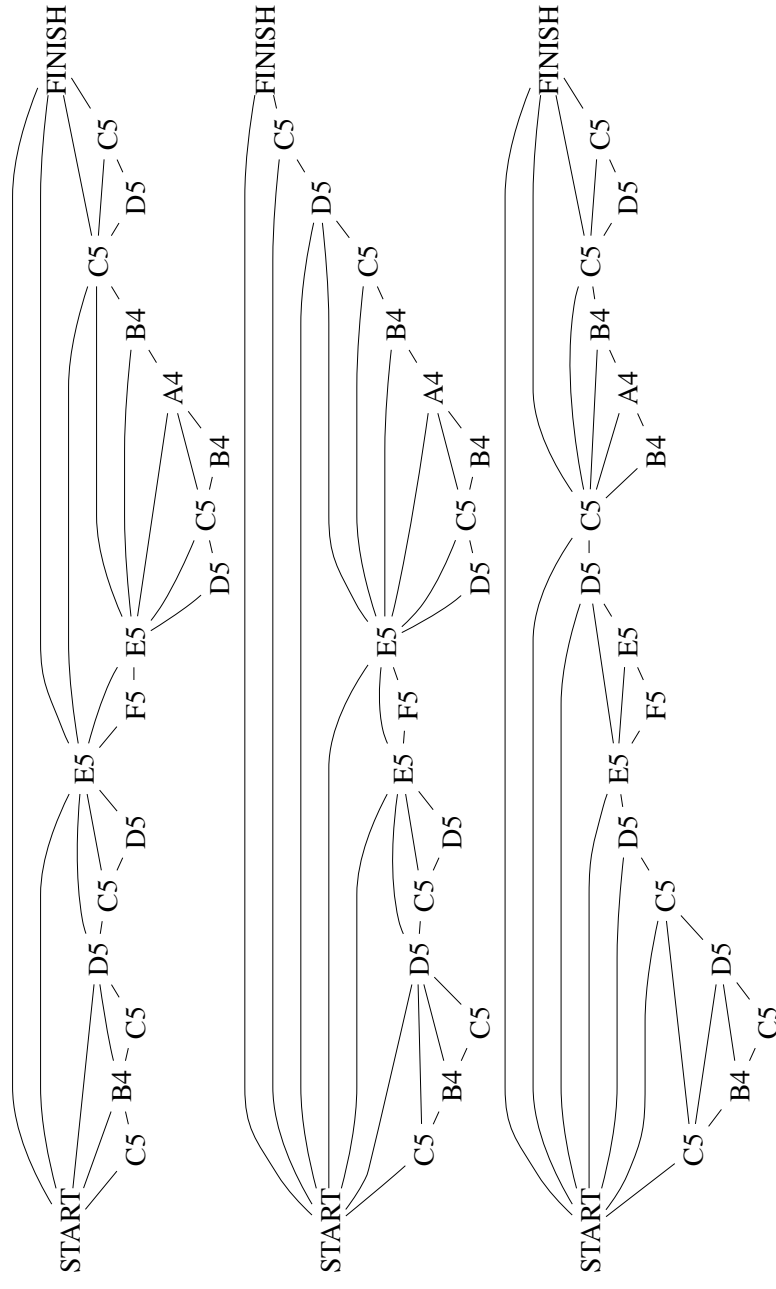


Figure A.4: MOPs produced by PARSEMOP-A, -B, and -C for bach3a

Excerpt identifier: handell1

MOP(s) contained in this excerpt: handella

Score:

Score for three staves in 4/4 time, key of B-flat major. The top staff is a treble clef, the middle is a treble clef, and the bottom is a bass clef. The music consists of eighth and quarter notes with various rests and ties.

Textbook analysis:

Textbook analysis of the first staff, showing a treble clef with a B-flat key signature and a 4/4 time signature. The melody is annotated with Roman numerals and slurs.

B^b: I IVI III IV IIV VIIIVI IIVVIVI V I IIVI

PARSEMOP-C analysis:

PARSEMOP-C analysis of the first two staves, showing treble clefs with B-flat key signatures and 4/4 time signatures. The analysis includes Roman numerals and slurs.

B^b: I IVI III IV IIV VIIIVI

IIVVIVI V I IIVI

Excerpt identifier: haydn1

MOP(s) contained in this excerpt: haydn1a, haydn1b

Score:

Andante

Textbook analysis:

B^b: I IVI V VIIII V

B^b: I IVI V VIIII V I

PARSEMOP-C analysis:

B^b: I IVI V VIIII V

B^b: I IVI V VIIII V I

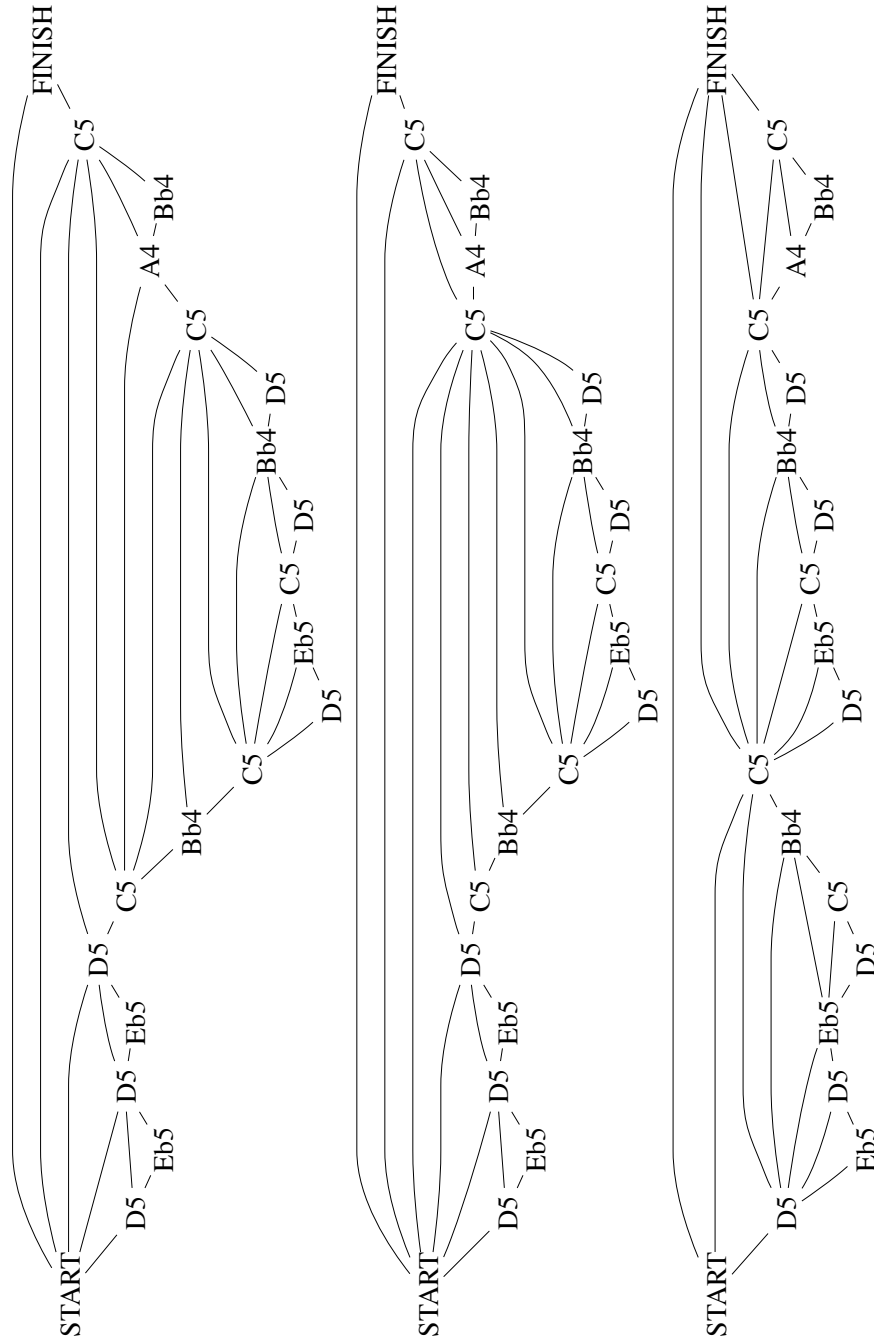


Figure A.6: MOPs produced by PARSEMOP-A, -B, and -C for haydn1a

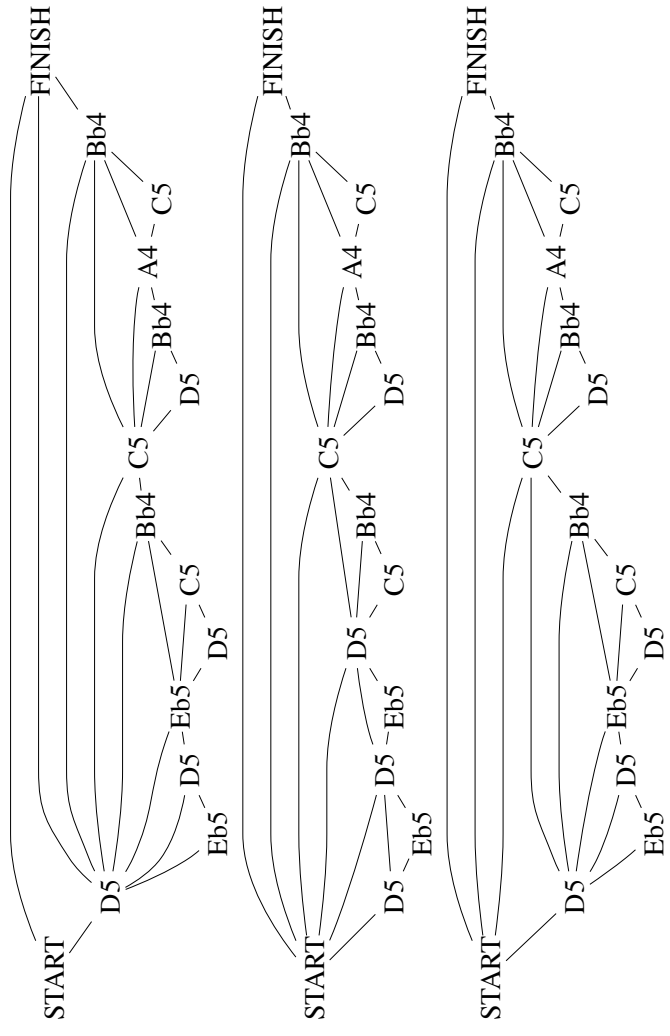


Figure A.7: MOPs produced by PARSEMOP-A, -B, and -C for haydn1b

Excerpt identifier: haydn2

MOP(s) contained in this excerpt: haydn2a

Score:

Allegro con brio

The first system of the musical score consists of two staves. The upper staff is in treble clef and the lower staff is in bass clef. The time signature is 2/2. The music begins with a half note G4 in the treble and a whole rest in the bass. The melody in the treble staff moves stepwise: G4, A4, B4, C5, B4, A4, G4. The bass staff provides harmonic support with chords: G2-B2-E3 (first two measures), G2-B2-E3 (third measure), and G2-B2-E3 (fourth measure).

The second system of the musical score consists of two staves. The upper staff is in treble clef and the lower staff is in bass clef. The time signature is 2/2. The melody in the treble staff continues: F4, E4, D4, C4, B3, A3, G3. The bass staff provides harmonic support with chords: G2-B2-E3 (first two measures), G2-B2-E3 (third measure), and G2-B2-E3 (fourth measure). A triplet of eighth notes (G3, A3, B3) is marked in the final measure of the bass staff.

Textbook analysis:

The textbook analysis shows the melody from the first system on a single staff. The notes are G4, A4, B4, C5, B4, A4, G4. The notes are grouped into four measures by vertical lines. Below the staff, Roman numerals are placed: C: I under the first measure, V under the second, I under the third, and II VI under the fourth.

PARSEMOP-C analysis:

The PARSEMOP-C analysis shows the melody from the first system on a single staff. The notes are G4, A4, B4, C5, B4, A4, G4. The notes are grouped into four measures by vertical lines. Below the staff, Roman numerals are placed: C: I under the first measure, V under the second, I under the third, and II VI under the fourth.

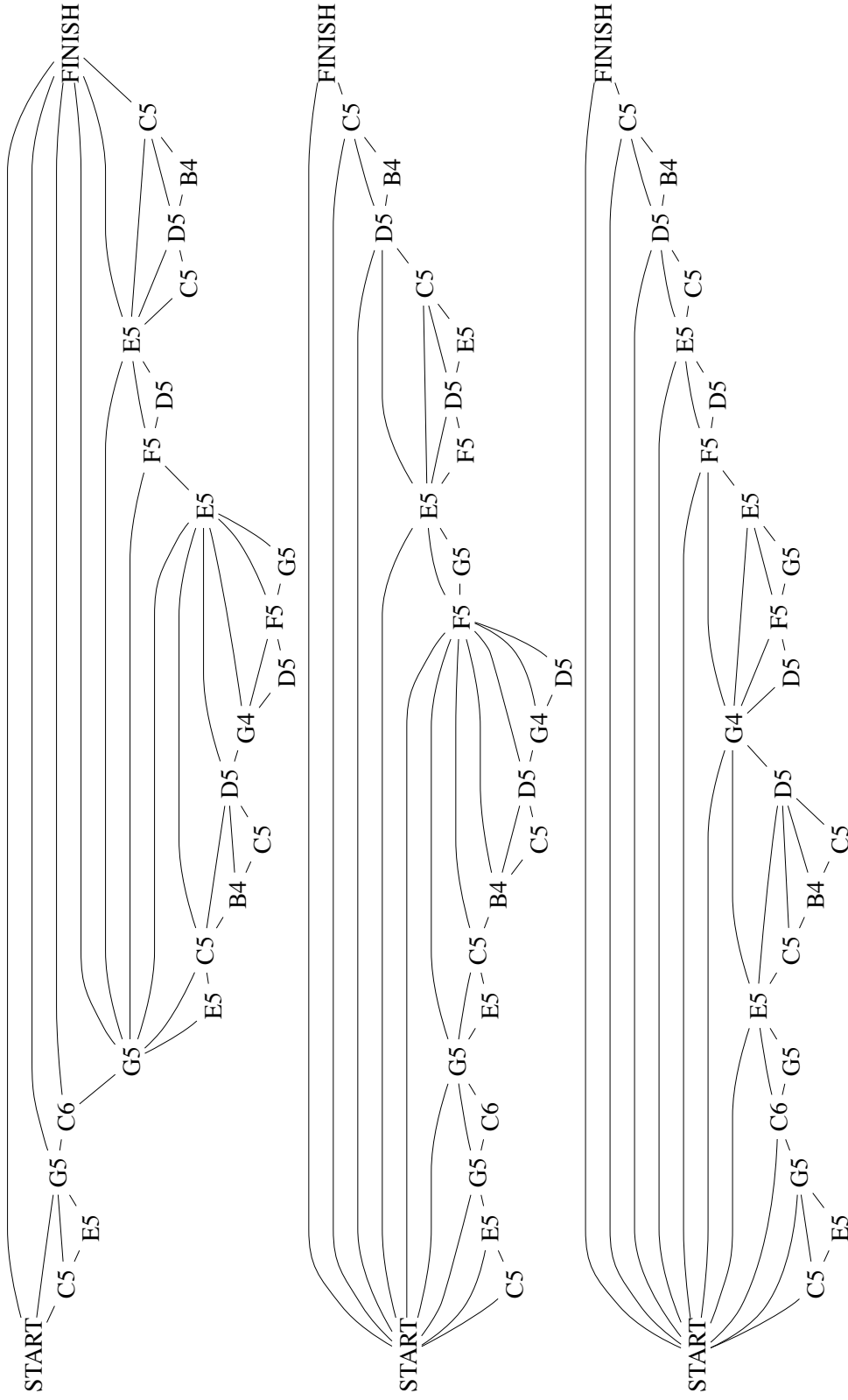


Figure A.8: MOPs produced by PARSEMOP-A, -B, and -C for haydn2a

Excerpt identifier: haydn3

MOP(s) contained in this excerpt: haydn3a

Score:

The first system of the musical score is in G major and 3/4 time. The right hand features a melodic line with a triplet of eighth notes in the first measure, followed by quarter notes. The left hand provides a steady accompaniment of eighth-note chords. A dynamic marking of *f* is present in the second measure.

The second system continues the piece. The right hand has a triplet of eighth notes in the first measure, followed by quarter notes. The left hand continues with eighth-note chords. The system concludes with a double bar line.

Textbook analysis:

The textbook analysis shows the first system with a thick black box above the staff. Below the staff, Roman numerals indicate the chord progression: G: V I V VII II V I.

PARSEMOP-C analysis:

The PARSEMOP-C analysis shows the first system with a thick black box above the staff. Below the staff, Roman numerals indicate the chord progression: G: V I V VII II V I.

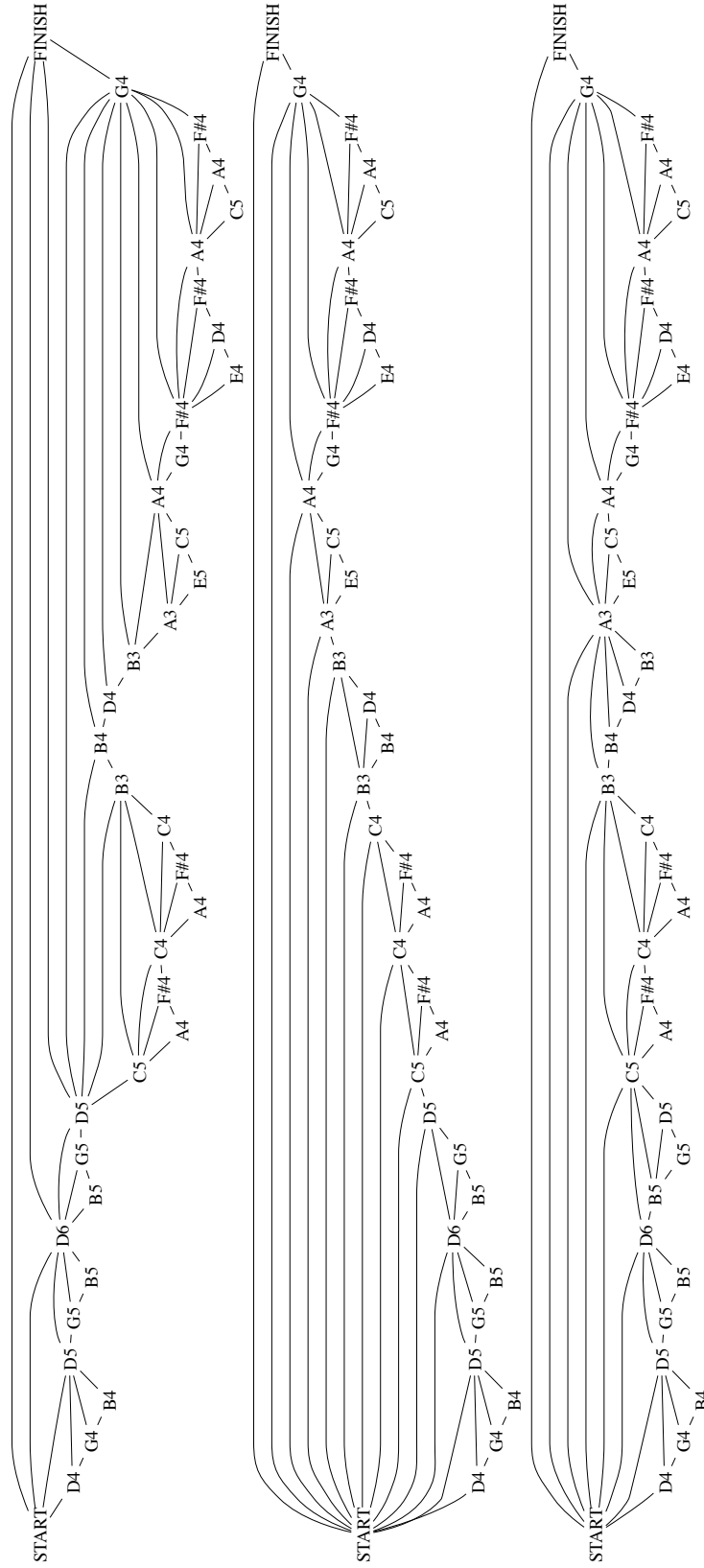


Figure A.9: MOPs produced by PARSEMOP-A, -B, and -C for haydn3a

Excerpt identifier: haydn4

MOP(s) contained in this excerpt: haydn4a

Score:

A musical score in G major, 2/4 time, showing three measures. The first measure contains a quarter note G4, a quarter note A4, and a quarter note B4. The second measure contains a quarter note C5, a quarter note B4, a quarter note A4, and a quarter note G4. The third measure contains a quarter note F#4, a quarter rest, and a quarter note G4. The bass line consists of a quarter rest in the first measure, a quarter note G3 in the second, and a quarter note G3 in the third.

Textbook analysis:

A musical staff showing the first measure of the excerpt. The notes G4, A4, and B4 are grouped with a slur. Below the staff, the Roman numeral analysis is given as G: I V I.

PARSEMOP-C analysis:

A musical staff showing the first measure of the excerpt. The notes G4, A4, and B4 are grouped with a slur. Below the staff, the Roman numeral analysis is given as G: I V I.

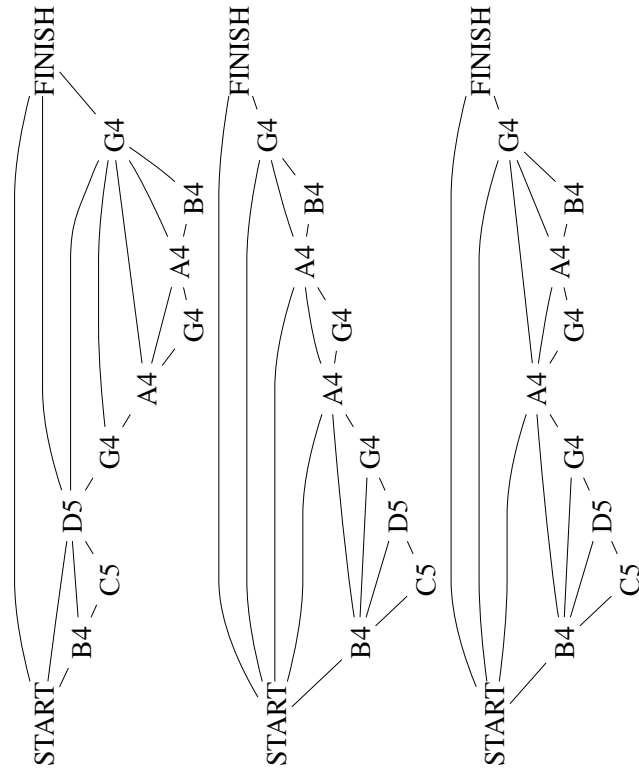


Figure A.10: MOPs produced by PARSEMOP-A, -B, and -C for haydn4a

Excerpt identifier: haydn5

MOP(s) contained in this excerpt: haydn5a

Score:

Textbook analysis:

E^b: I II V I

PARSEMOP-C analysis:

E^b: I II V I

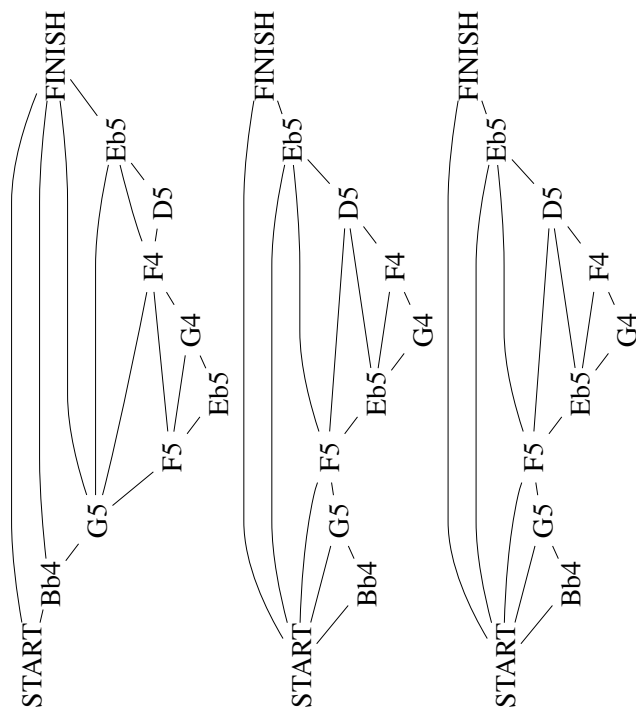


Figure A.11: MOPs produced by PARSEMOP-A, -B, and -C for haydn5a

Excerpt identifier: haydn6

MOP(s) contained in this excerpt: haydn6a

Score:

Musical score for piano in 3/4 time, key of B-flat major. The score consists of five measures. The right hand plays a melodic line starting with a quarter note G4, followed by eighth notes A4-B4, C5-B4-A4, and a quarter note G4. The left hand provides harmonic support with chords: a whole note B-flat major chord in the first measure, and quarter notes G3, F3, and E3 in the second, third, and fourth measures, respectively. The piece ends with a quarter note G4 in the fifth measure.

Textbook analysis:

Textbook analysis of the score, showing the melodic line with a box above it and Roman numerals B^b: I, V, I below it.

PARSEMOP-C analysis:

PARSEMOP-C analysis of the score, showing the melodic line with a box above it and Roman numerals B^b: I, V, I below it.

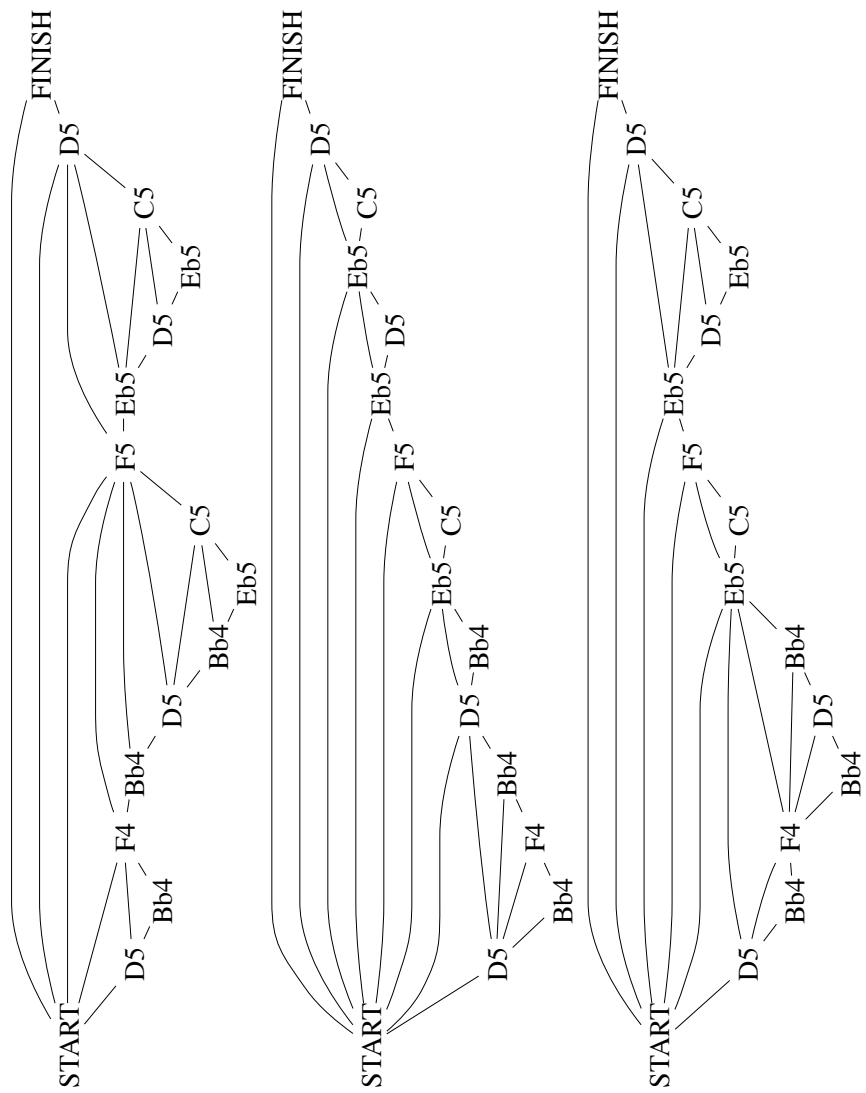


Figure A.12: MOPs produced by PARSEMOP-A, -B, and -C for haydn6a

Excerpt identifier: haydn7

MOP(s) contained in this excerpt: haydn7a

Score:

Textbook analysis:

B^b: I VI VI V VI V VII V I

PARSEMOP-C analysis:

B^b: I VI VI V VI V VII V I

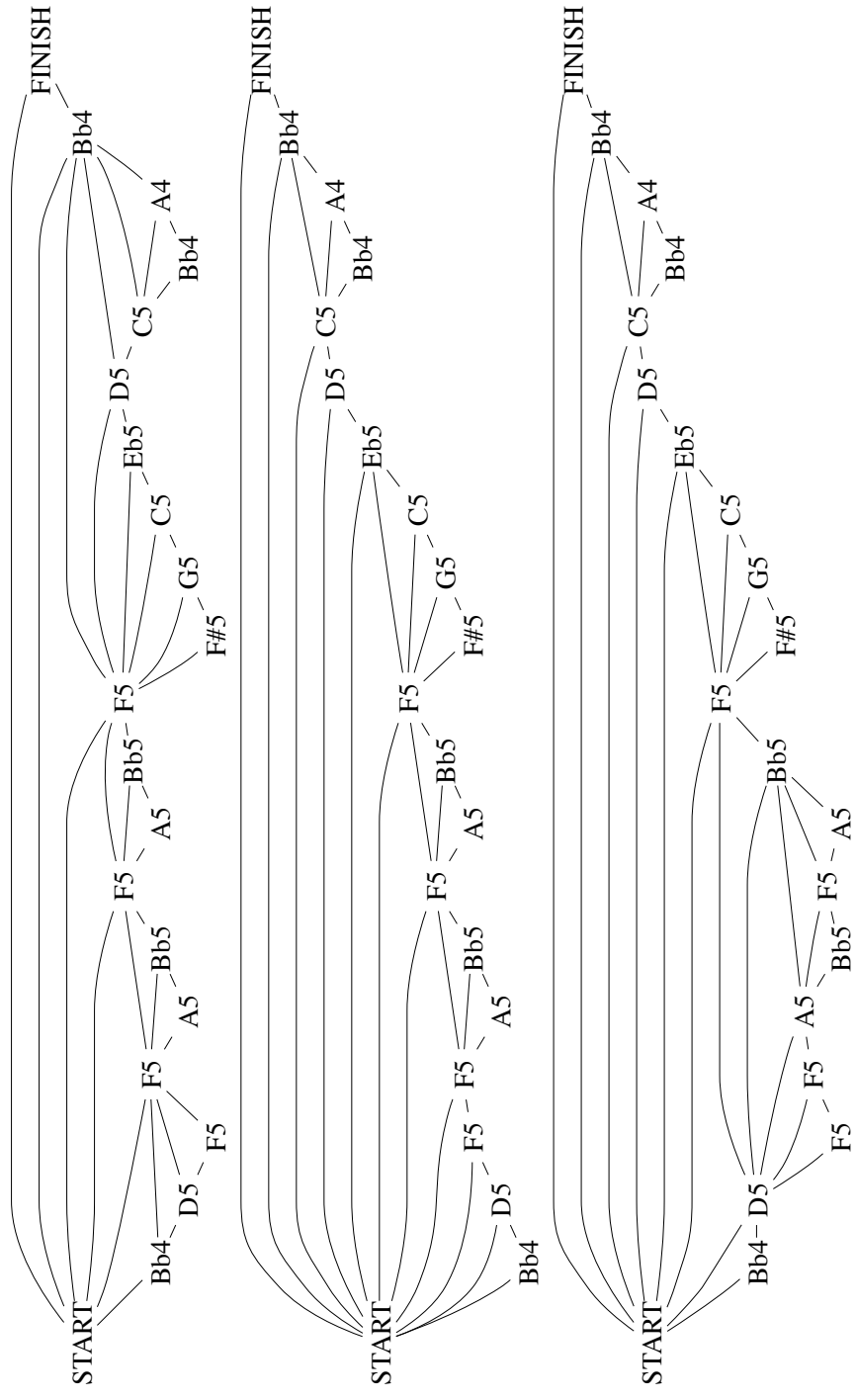


Figure A.13: MOPs produced by PARSEMOP-A, -B, and -C for haydn7a

Excerpt identifier: clementil

MOP(s) contained in this excerpt: clementila

Score:

A musical score for piano in D major, 4/4 time. The right hand (treble clef) starts with a quarter rest, followed by a quarter note D4, a quarter note E4, a quarter note F#4, and a quarter note G4. This is followed by a half note chord of D4 and F#4, then a quarter note G4, a quarter note A4, and a quarter note B4. The piece concludes with a quarter note G4, a quarter note F#4, and a quarter note E4. The left hand (bass clef) starts with a quarter rest, followed by a quarter note D3, a quarter note E3, a quarter note F#3, and a quarter note G3. This is followed by a half note chord of D3 and F#3, then a quarter note G3, a quarter note A3, and a quarter note B3. The piece concludes with a quarter note G3, a quarter note F#3, and a quarter note E3.

Textbook analysis:

A textbook analysis of the melodic line from the score. It shows a treble clef with a key signature of two sharps (D major). The notes are D4, E4, F#4, G4, A4, B4, G4, F#4, E4. A slur covers the notes from E4 to B4. Vertical lines connect the notes to the chord analysis below.

D: I V I

PARSEMOP-C analysis:

A PARSEMOP-C analysis of the melodic line from the score. It shows a treble clef with a key signature of two sharps (D major). The notes are D4, E4, F#4, G4, A4, B4, G4, F#4, E4. A slur covers the notes from E4 to B4. Vertical lines connect the notes to the chord analysis below.

D: I V I

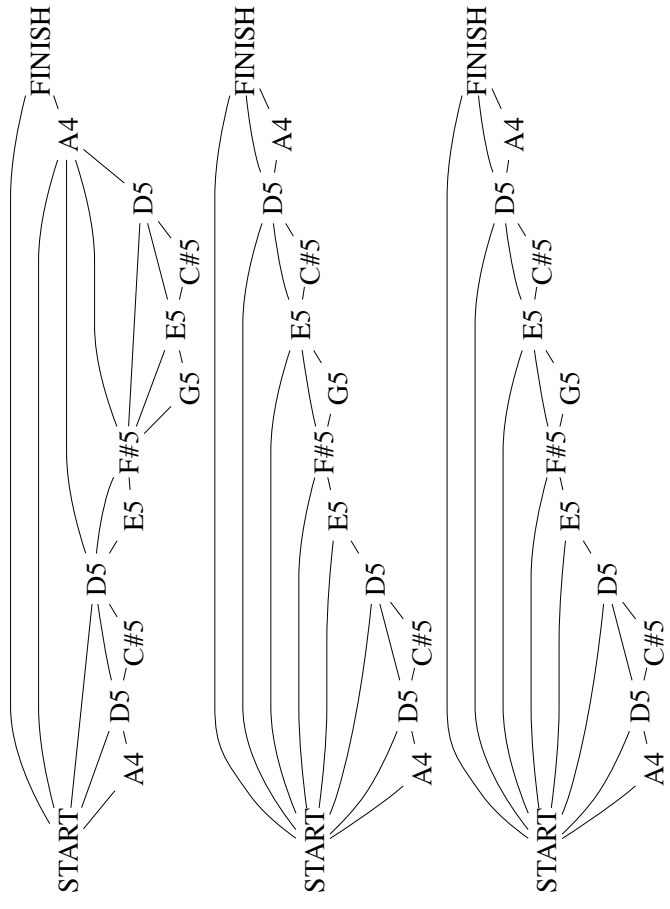


Figure A.14: MOPs produced by PARSEMOP-A, -B, and -C for clement1a

Excerpt identifier: mozart1

MOP(s) contained in this excerpt: mozart1a, mozart1b

Score:

Adagio

p

f *p*

Textbook analysis:

A: I V VIV I II V

A: I V VIV I II V I

PARSEMOP-C analysis:

A: I V VIV I II V

A: I V VIV I II V I

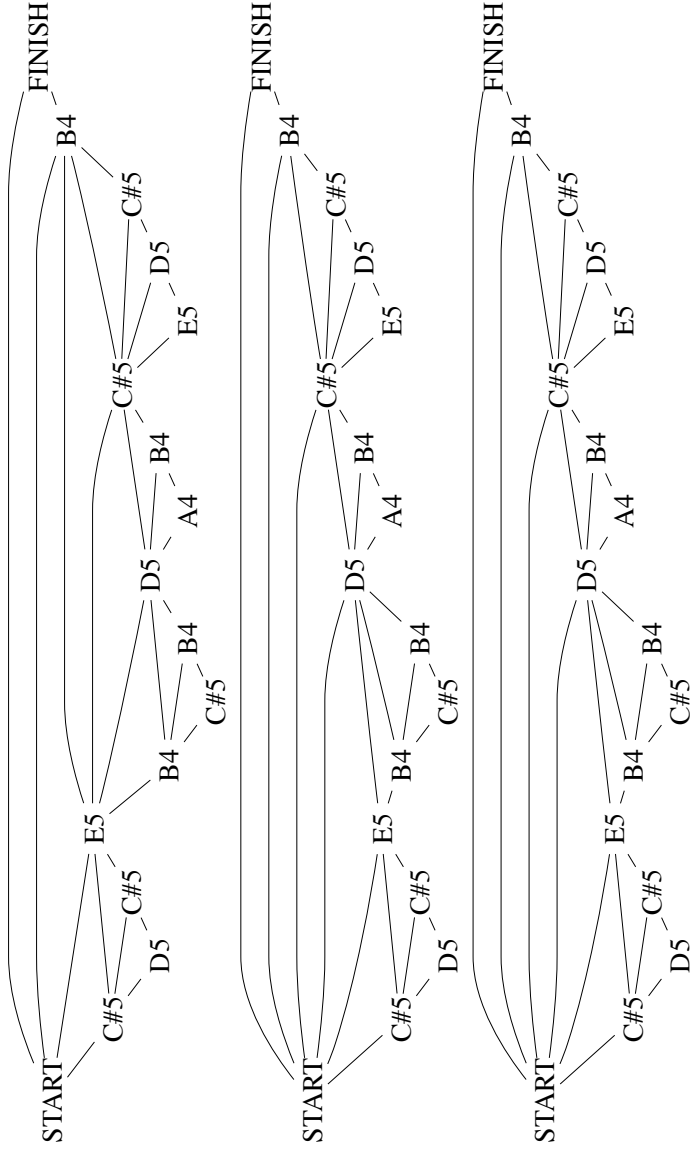


Figure A.15: MOPs produced by PARSEMOP-A, -B, and -C for mozart1a

Excerpt identifier: mozart2

MOP(s) contained in this excerpt: mozart2a, mozart2b

Score:

Textbook analysis:

B^b: I II VI V I II V

B^b: I II VI II V I

PARSEMOP-C analysis:

B^b: I II VI V I II V

B^b: I II VI II V I

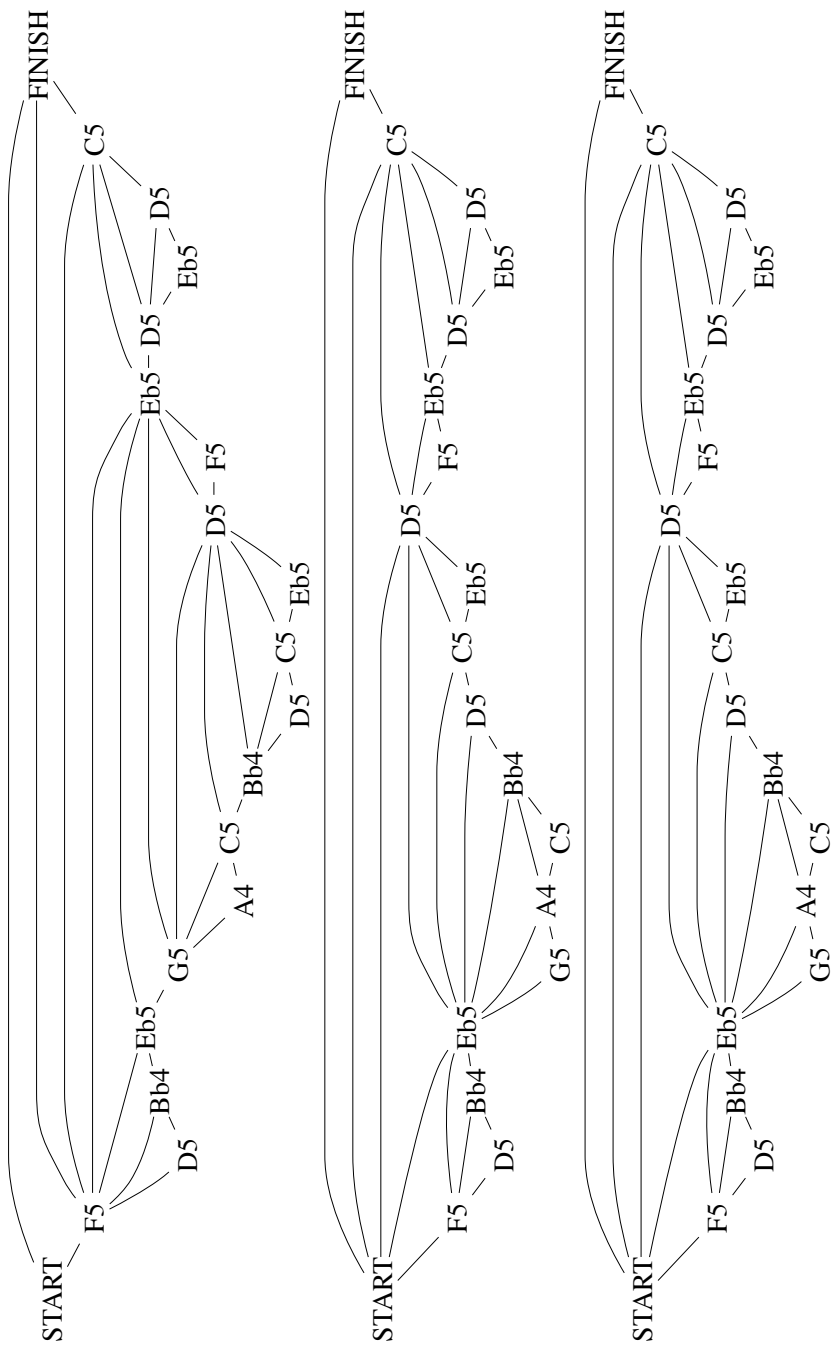


Figure A.17: MOPs produced by PARSEMOP-A, -B, and -C for mozart2a

Excerpt identifier: mozart3

MOP(s) contained in this excerpt: mozart3a, mozart3b

Score:

Rondo

The score consists of two systems of piano accompaniment. The first system has five measures, and the second system has four measures. The music is in 2/4 time and features a mix of chords and melodic lines in both the treble and bass staves.

Textbook analysis:

The textbook analysis shows two systems of melodic lines. The first system has five measures with Roman numerals: C: I VIIIV I IV II V. The second system has four measures with Roman numerals: C: I VIIIV I II V I.

PARSEMOP-C analysis:

The PARSEMOP-C analysis shows two systems of melodic lines. The first system has five measures with Roman numerals: C: I VIIIV I IV II V. The second system has four measures with Roman numerals: C: I VIIIV I II V I.

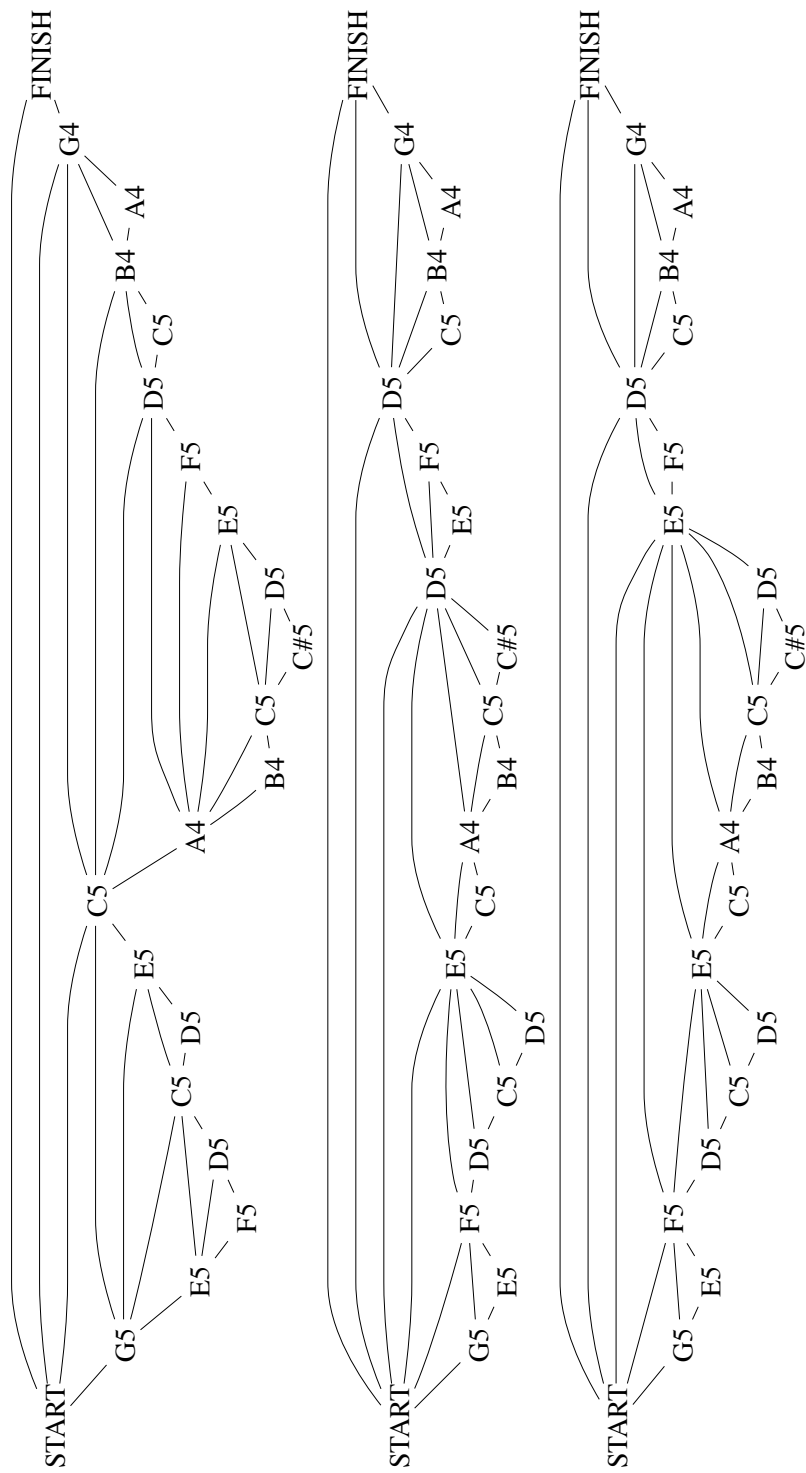


Figure A.19: MOPs produced by PARSEMOP-A, -B, and -C for mozart3a

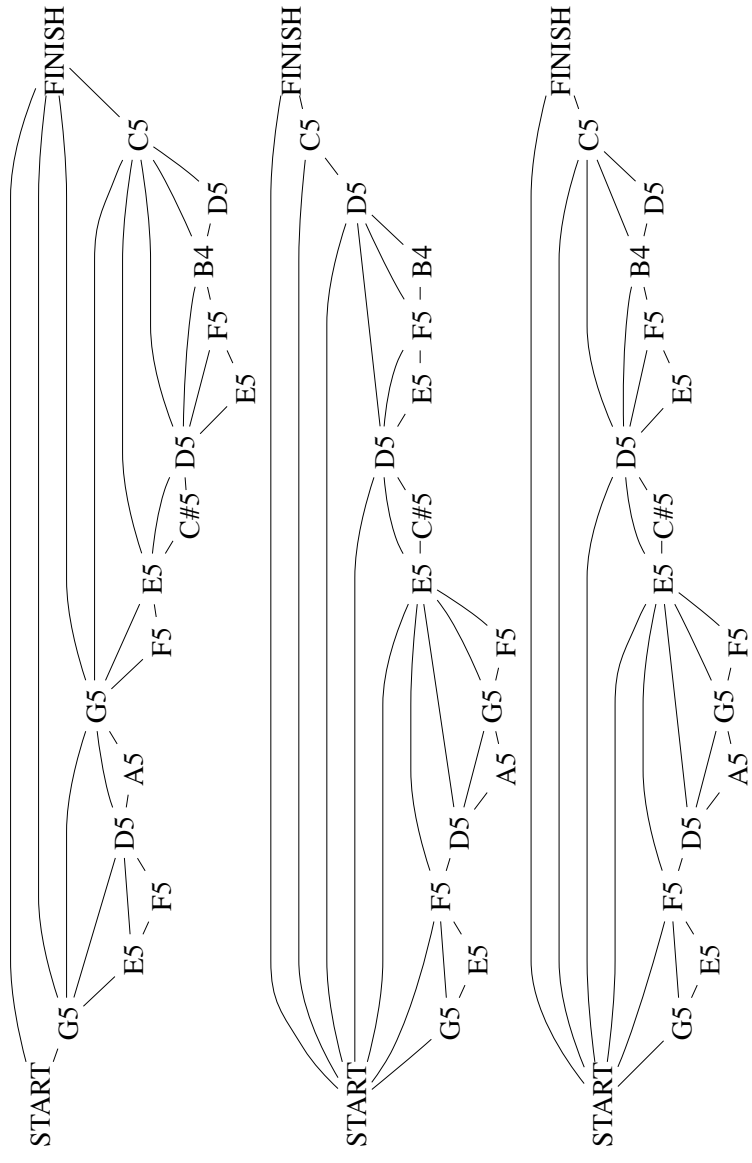


Figure A.20: MOPs produced by PARSEMOP-A, -B, and -C for mozart3b

Excerpt identifier: mozart4

MOP(s) contained in this excerpt: mozart4a, mozart4b

Score:

Allegretto

The score is for a piano accompaniment in A major, 2/2 time, marked 'Allegretto'. It consists of two systems of four measures each. The first system shows the first four measures, and the second system shows the next four measures. The music features a steady bass line with chords and a treble line with chords and a melodic line in the final measure of each system.

Textbook analysis:

A: I II VI VI V

A: I II V VIIIV I

The textbook analysis shows the chord progressions for the first system (measures 1-4) and the second system (measures 5-8). The first system has chords I, II, VI, VI, and V. The second system has chords I, II, V, VIIIV, and I.

PARSEMOP-C analysis:

A: I II VI VI V

A: I II V VIIIV I

The PARSEMOP-C analysis shows the chord progressions for the first system (measures 1-4) and the second system (measures 5-8). The first system has chords I, II, VI, VI, and V. The second system has chords I, II, V, VIIIV, and I.

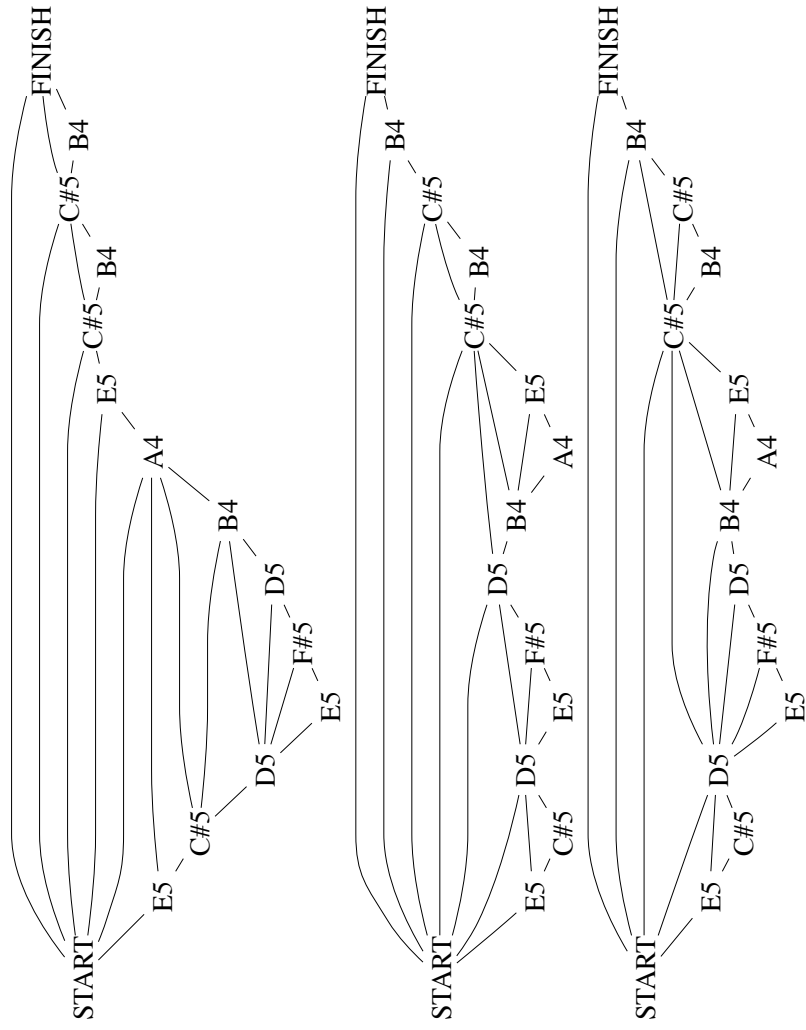


Figure A.21: MOPs produced by PARSEMOP-A, -B, and -C for mozart4a

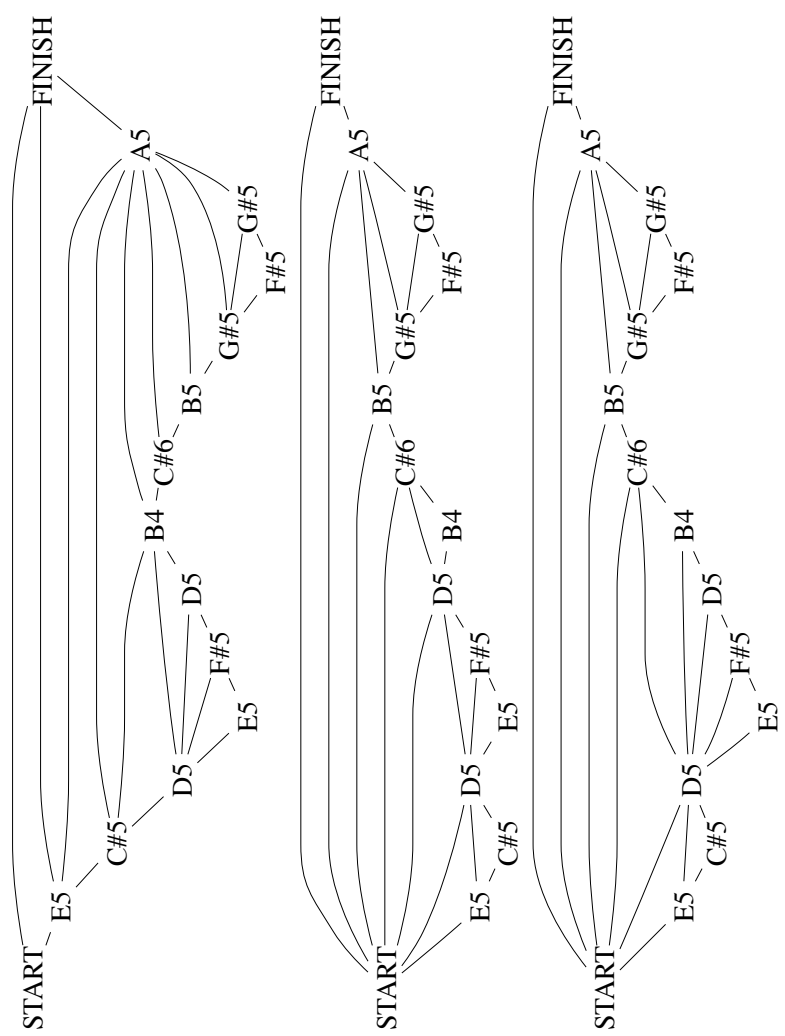


Figure A.22: MOPs produced by PARSEMOP-A, -B, and -C for mozart4b

Excerpt identifier: mozart5

MOP(s) contained in this excerpt: mozart5a

Score:

The score is in 4/4 time. The first system shows the right hand with a treble clef and the left hand with a bass clef. The second system shows the right hand with a treble clef and the left hand with a bass clef, featuring a piano accompaniment with chords and a melodic line.

Textbook analysis:

C: I V I II V I

PARSEMOP-C analysis:

C: I V I II V I

Excerpt identifier: mozart7

MOP(s) contained in this excerpt: mozart7a

Score:

Textbook analysis:

D: V I V I V I II V I

PARSEMOP-C analysis:

D: V I V I V I II V I

Excerpt identifier: mozart8

MOP(s) contained in this excerpt: mozart8a

Score:

The first system of the score consists of two staves. The upper staff is in treble clef with a 2/4 time signature, featuring a melodic line with eighth-note patterns and a key signature of one sharp (F#). The lower staff is in bass clef, providing a simple harmonic accompaniment with quarter notes and rests.

The second system continues the piece. The upper staff shows the melodic line with some slurs and a final quarter rest. The lower staff continues the accompaniment, ending with a quarter note and a final sharp sign.

Textbook analysis:

A single staff showing the melodic line from the score with slurs and ties, highlighting the phrasing and intervallic structure.

C: I

A staff showing the harmonic structure of the excerpt. Roman numerals are placed below the staff to indicate the chords: IV, I, V, I, VI, II, V, I. The staff includes slurs and ties to show the melodic flow across the chords.

PARSEMOP-C analysis:

A staff showing the melodic line with slurs and ties, similar to the textbook analysis but with different phrasing, indicating a different MOP analysis.

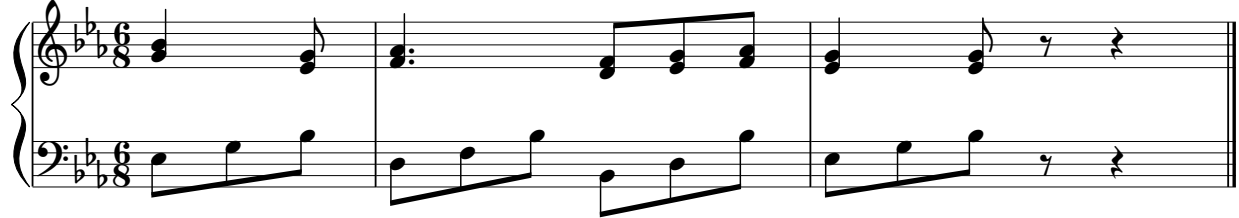
C: I

A staff showing the harmonic structure for the PARSEMOP-C analysis. Roman numerals are placed below the staff: V, I, VI, II, V, I. The staff includes slurs and ties to show the melodic flow.

Excerpt identifier: mozart9

MOP(s) contained in this excerpt: mozart9a

Score:



Textbook analysis:



E^b: I V I

PARSEMOP-C analysis:



E^b: I V I

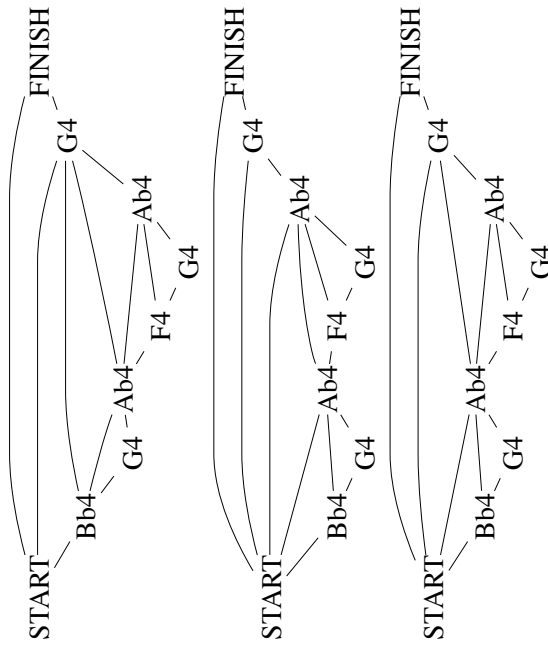


Figure A.27: MOPs produced by PARSEMOP-A, -B, and -C for mozart9a

Excerpt identifier: mozart10

MOP(s) contained in this excerpt: mozart10a

Score:

A musical score for a piano excerpt in 3/4 time, key of B-flat major. The score consists of two staves: a treble clef staff and a bass clef staff. The treble staff begins with a triplet of eighth notes (F4, G4, A4), followed by a quarter note (Bb4), a quarter note (C5), and a quarter note (Bb4). The bass staff begins with a quarter note (F3), a quarter note (Bb2), and a quarter note (C3). The piece concludes with a quarter rest in the treble and a quarter note (F3) in the bass.

Textbook analysis:

A textbook analysis of the musical score, showing the chord progressions for the first eight measures. The analysis is presented on a single treble clef staff with a key signature of one flat. The chords are labeled as follows: F: I, VII, I, V, I, IV, V, I.

PARSEMOP-C analysis:

A PARSEMOP-C analysis of the musical score, showing the chord progressions for the first eight measures. The analysis is presented on a single treble clef staff with a key signature of one flat. The chords are labeled as follows: F: I, VII, I, V, I, IV, V, I.

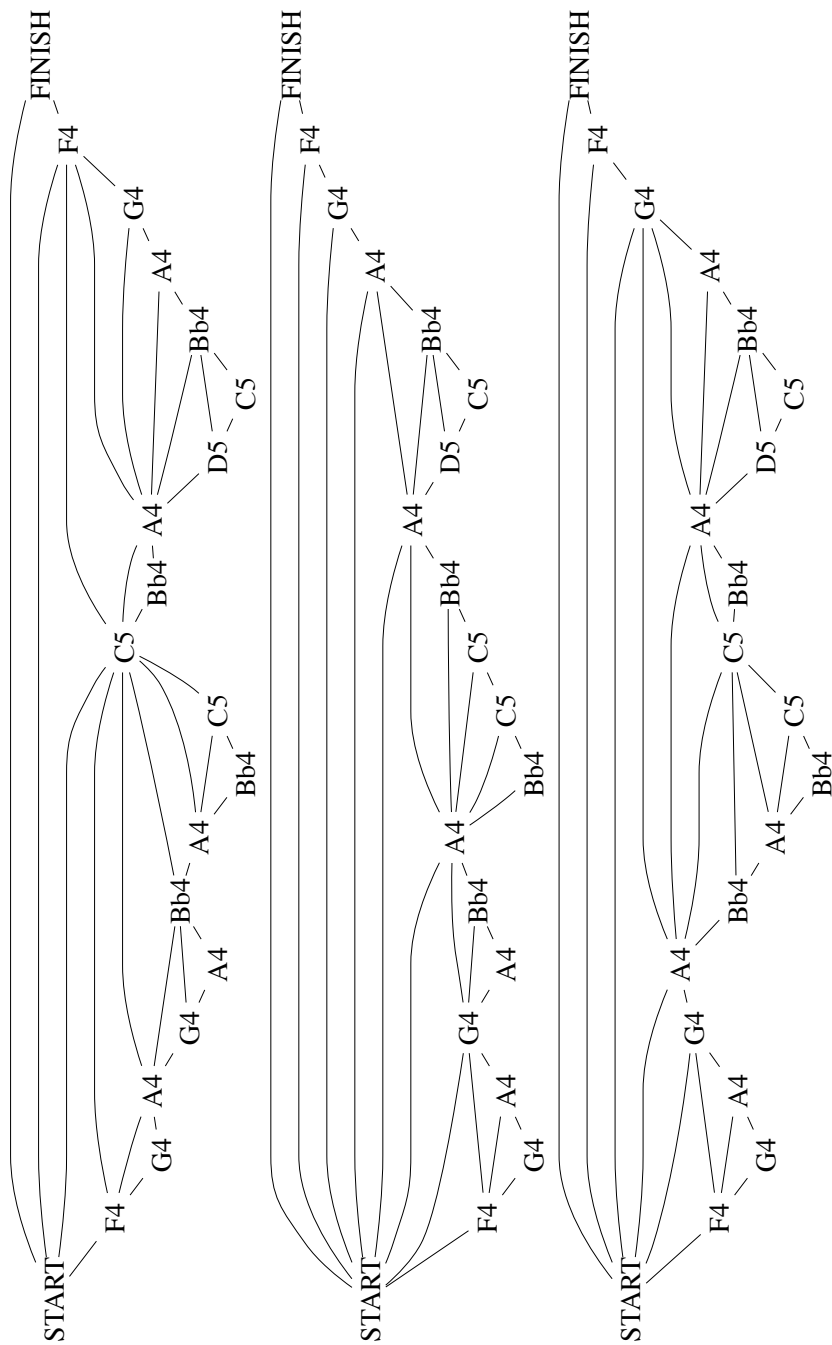


Figure A.28: MOPs produced by PARSEMOP-A, -B, and -C for mozart10a

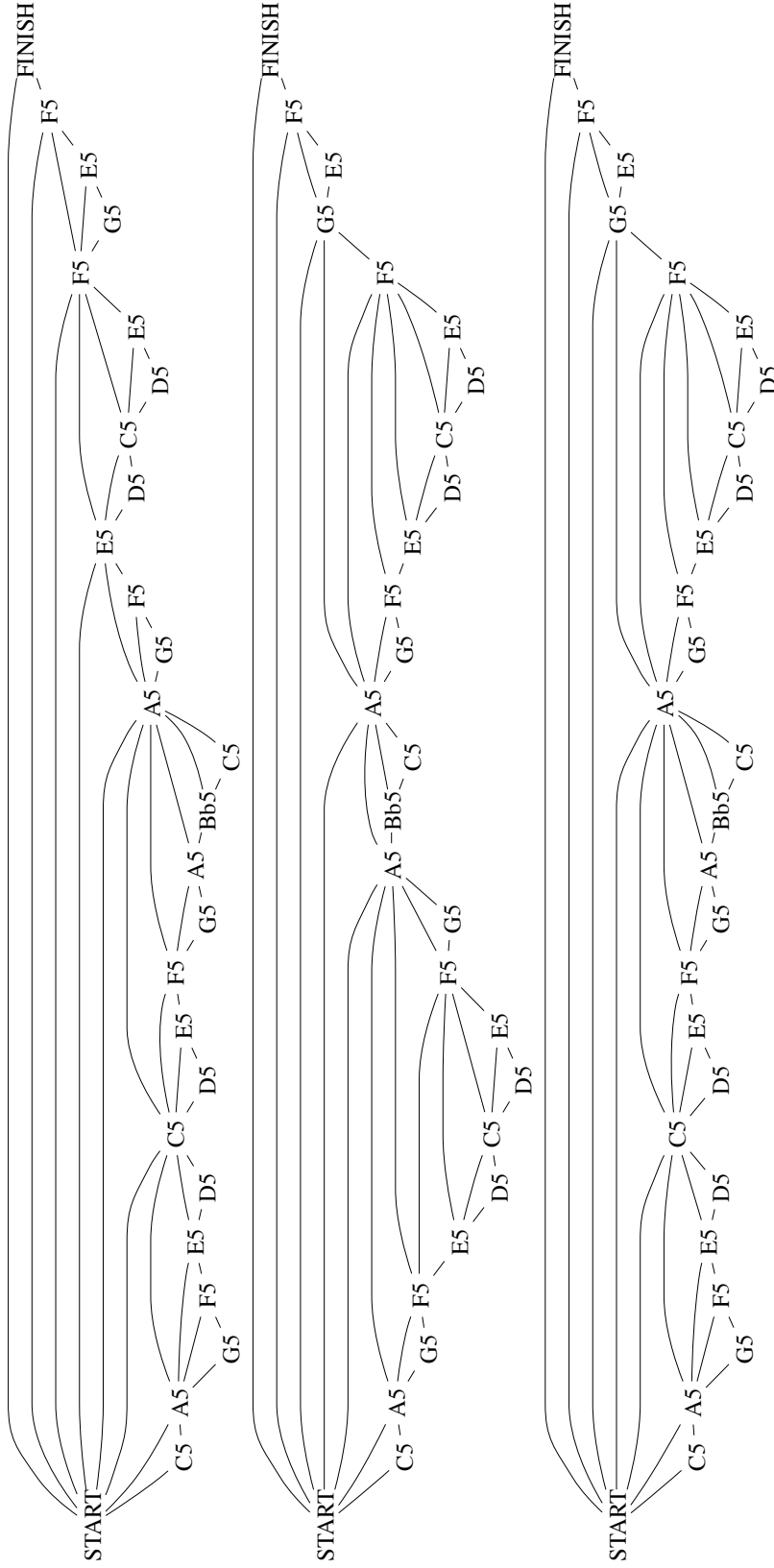


Figure A.29: MOPs produced by PARSEMOP-A, -B, and -C for mozart11a

Excerpt identifier: mozart12

MOP(s) contained in this excerpt: mozart12a

Score:

Textbook analysis:

F: I II V I

PARSEMOP-C analysis:

F: I II V I

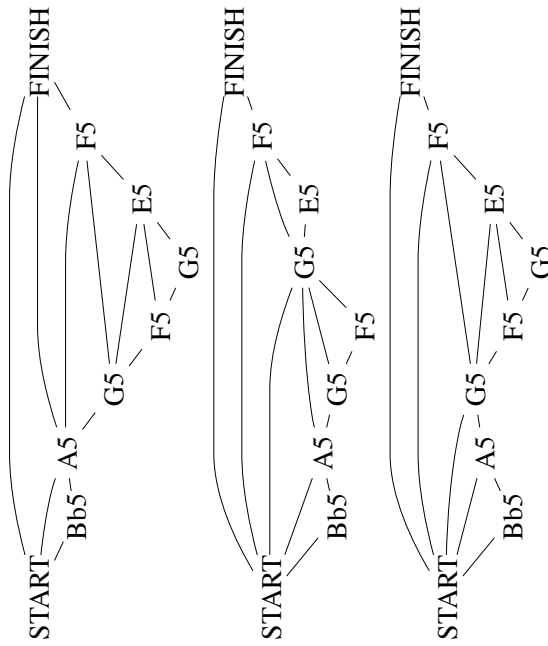
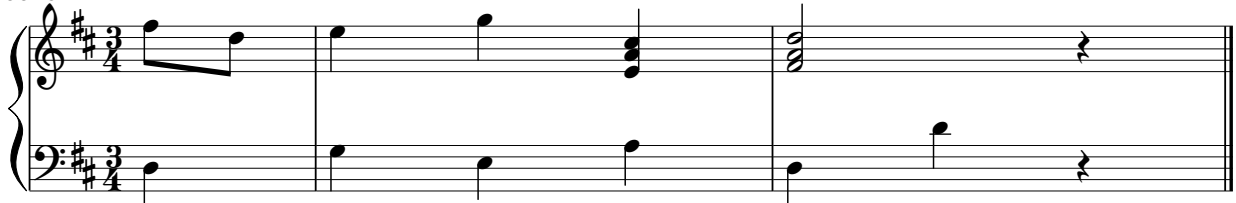


Figure A.30: MOPs produced by PARSEMOP-A, -B, and -C for mozart12a

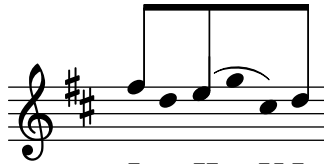
Excerpt identifier: mozart13

MOP(s) contained in this excerpt: mozart13a

Score:

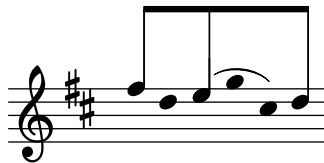


Textbook analysis:



D: I II VI

PARSEMOP-C analysis:



D: I II VI

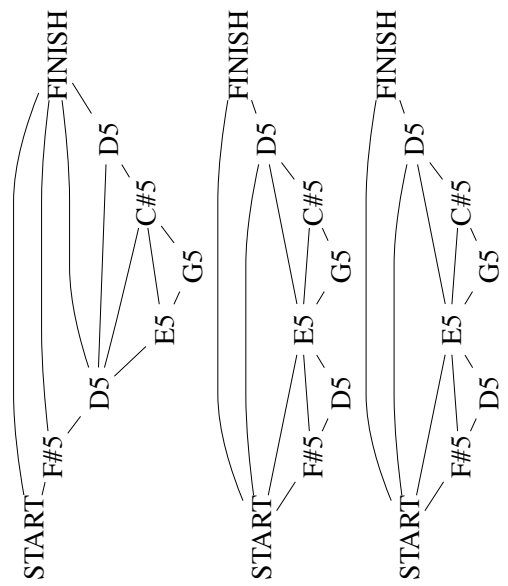


Figure A.31: MOPs produced by PARSEMOP-A, -B, and -C for mozart13a

Excerpt identifier: mozart14

MOP(s) contained in this excerpt: mozart14a

Score:

A musical score for a piano excerpt in 3/8 time, F major. The score consists of two staves. The right hand plays a melody with a triplet of eighth notes in the second measure. The left hand plays a simple accompaniment. The piece ends with a fermata over the final note.

Textbook analysis:

A single staff of music with a treble clef and a key signature of one flat. The notes are grouped with a slur and a fermata. Roman numerals F: I, IV, V, I are written below the notes.

PARSEMOP-C analysis:

A single staff of music with a treble clef and a key signature of one flat. The notes are grouped with a slur and a fermata. Roman numerals F: I, IV, V, I are written below the notes.

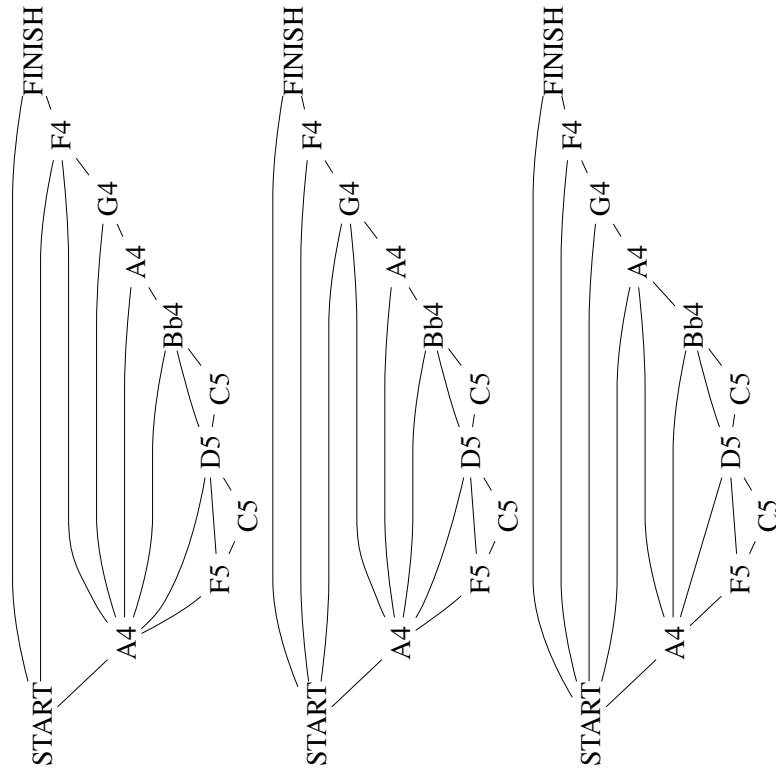
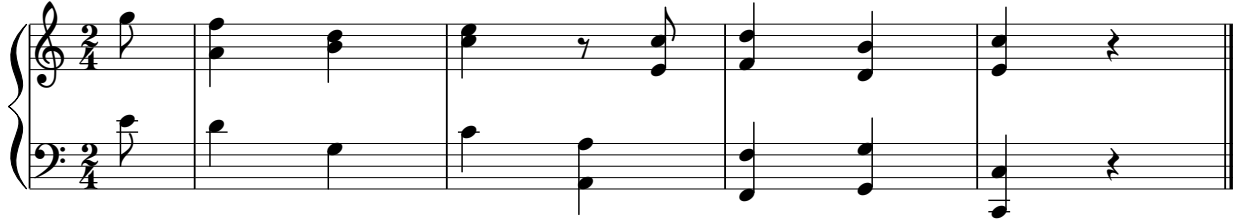


Figure A.32: MOPs produced by PARSEMOP-A, -B, and -C for mozart14a

Excerpt identifier: mozart15

MOP(s) contained in this excerpt: mozart15a

Score:

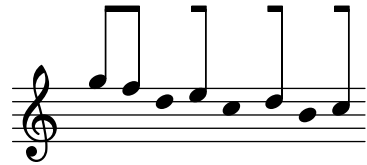


Textbook analysis:



C: I II V I VII I V I

PARSEMOP-C analysis:



C: I II V I VII I V I

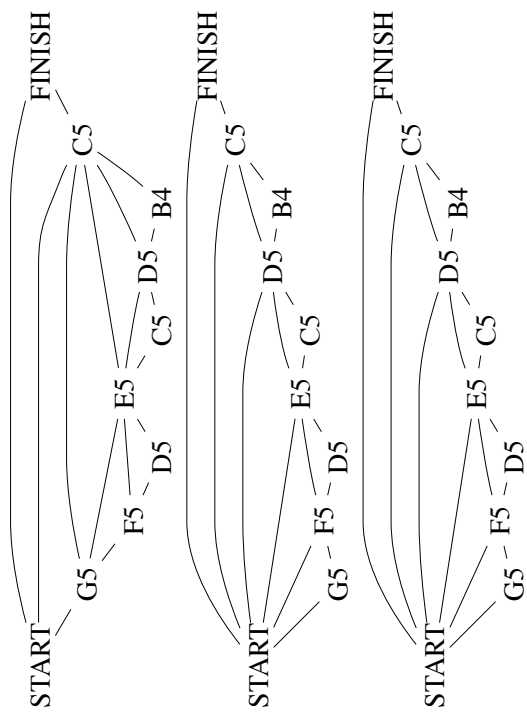


Figure A.33: MOPs produced by PARSEMOP-A, -B, and -C for mozart15a

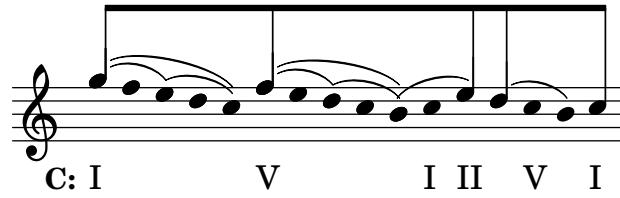
Excerpt identifier: mozart16

MOP(s) contained in this excerpt: mozart16a

Score:

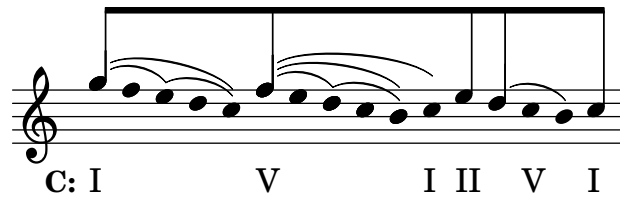


Textbook analysis:



C: I V I II V I

PARSEMOP-C analysis:



C: I V I II V I

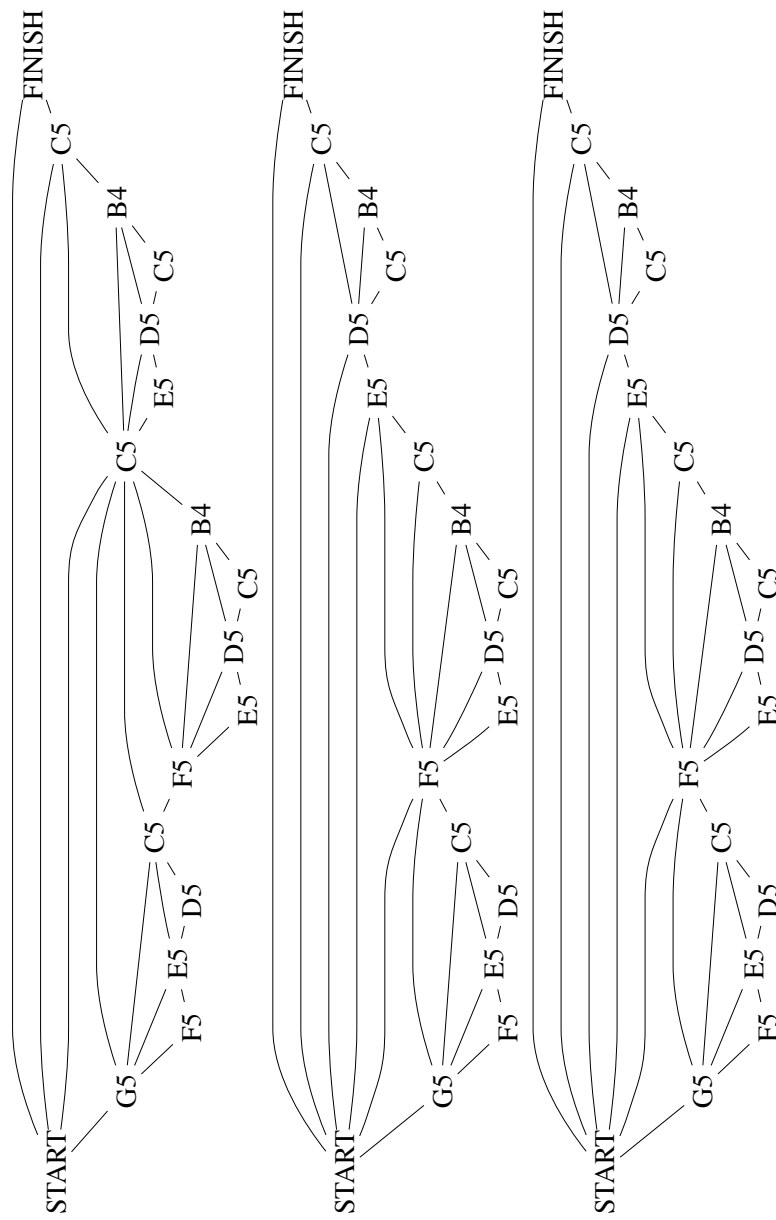


Figure A.34: MOPs produced by PARSEMOP-A, -B, and -C for mozzart16a

Excerpt identifier: mozart17

MOP(s) contained in this excerpt: mozart17a, mozart17b

Score:

Textbook analysis:

C: I V I V

C: VI V I II V I

PARSEMOP-C analysis:

C: I V I V

C: VI V I II V I

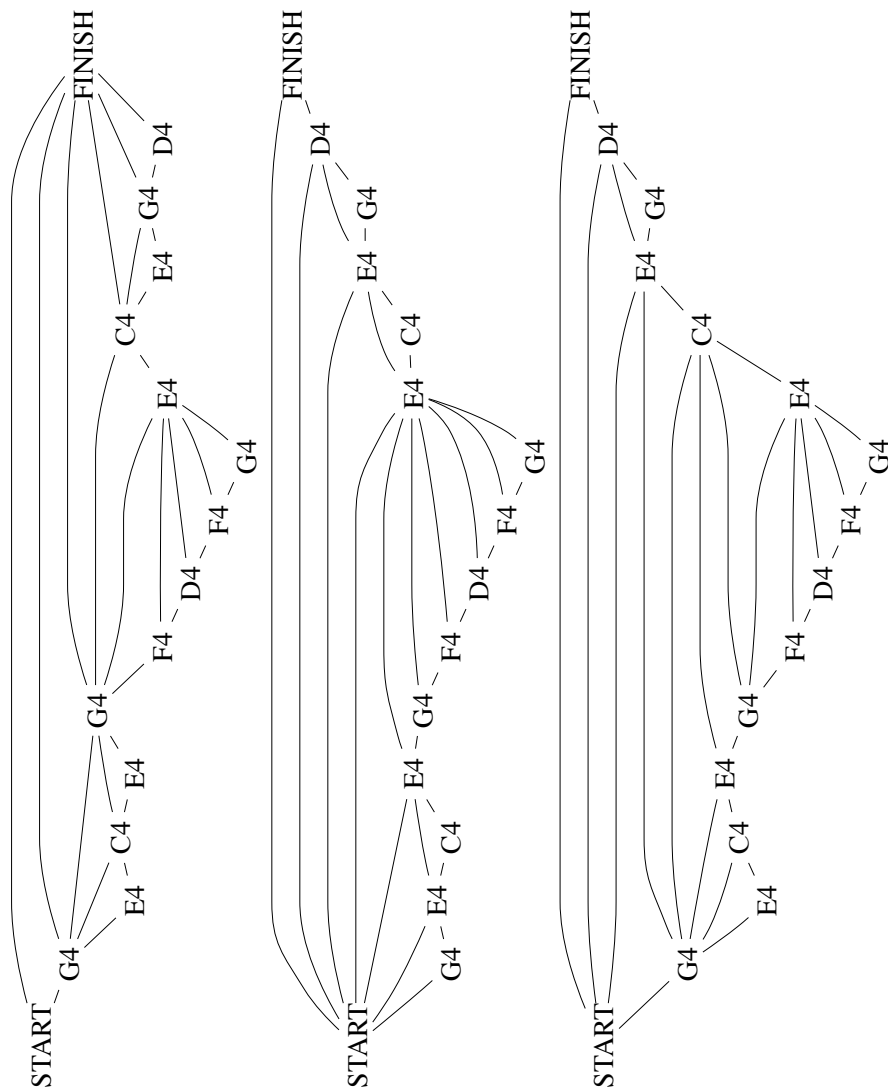


Figure A.35: MOPs produced by PARSEMOP-A, -B, and -C for mozart17a

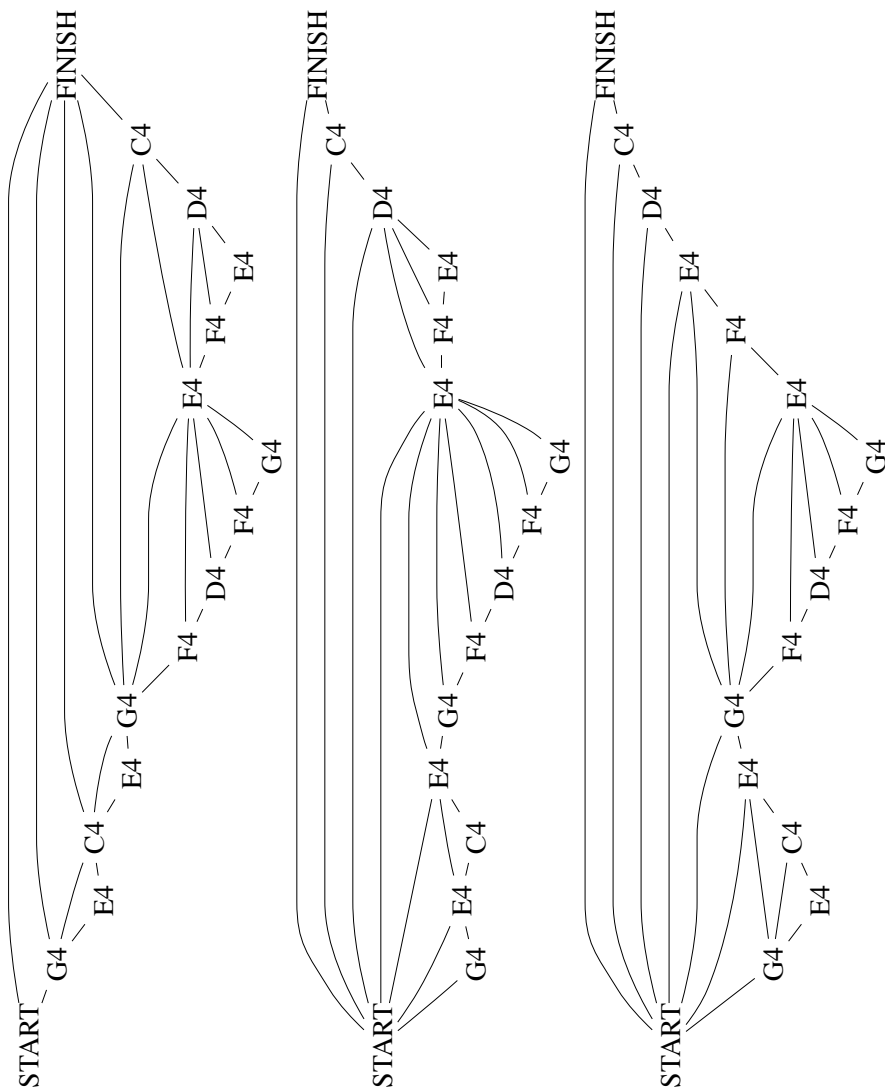


Figure A.36: MOPs produced by PARSEMOP-A, -B, and -C for mozart17b

Excerpt identifier: mozart18

MOP(s) contained in this excerpt: mozart18a

Score:

Textbook analysis:

F: I II I VI III V I II V I

PARSEMOP-C analysis:

F: I II I VI III V I II V I

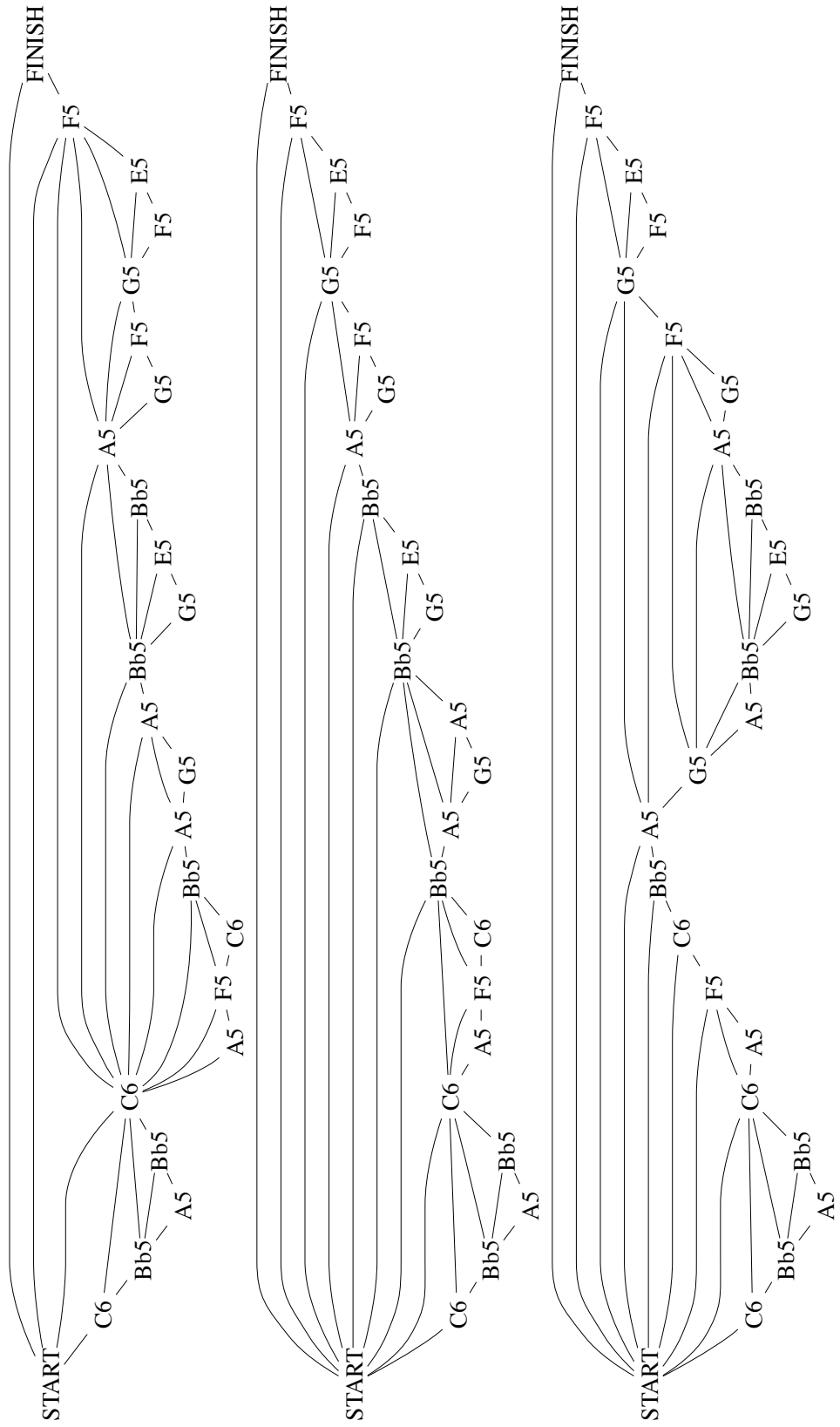


Figure A.37: MOPs produced by PARSEMOP-A, -B, and -C for mozart18a

Excerpt identifier: beethoven1

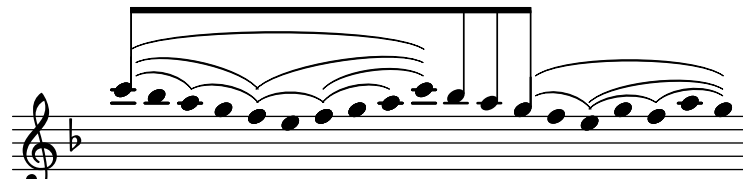
MOP(s) contained in this excerpt: beethoven1a, beethoven1b

Score:

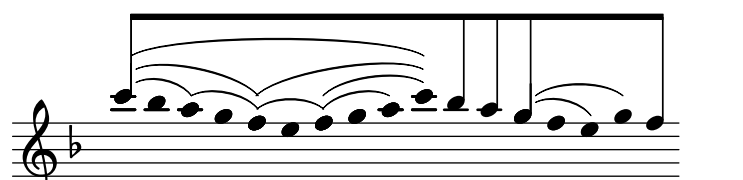
Allegro



Textbook analysis:

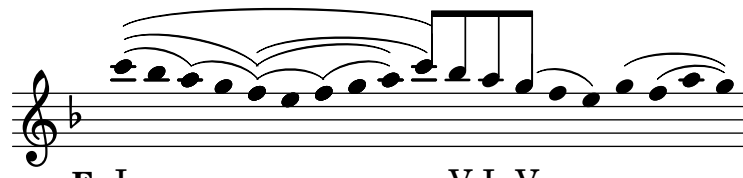


F: I VI V

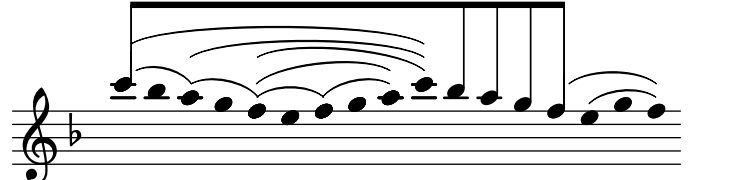


F: I VI II VI

PARSEMOP-C analysis:



F: I VI V



F: I VI II VI

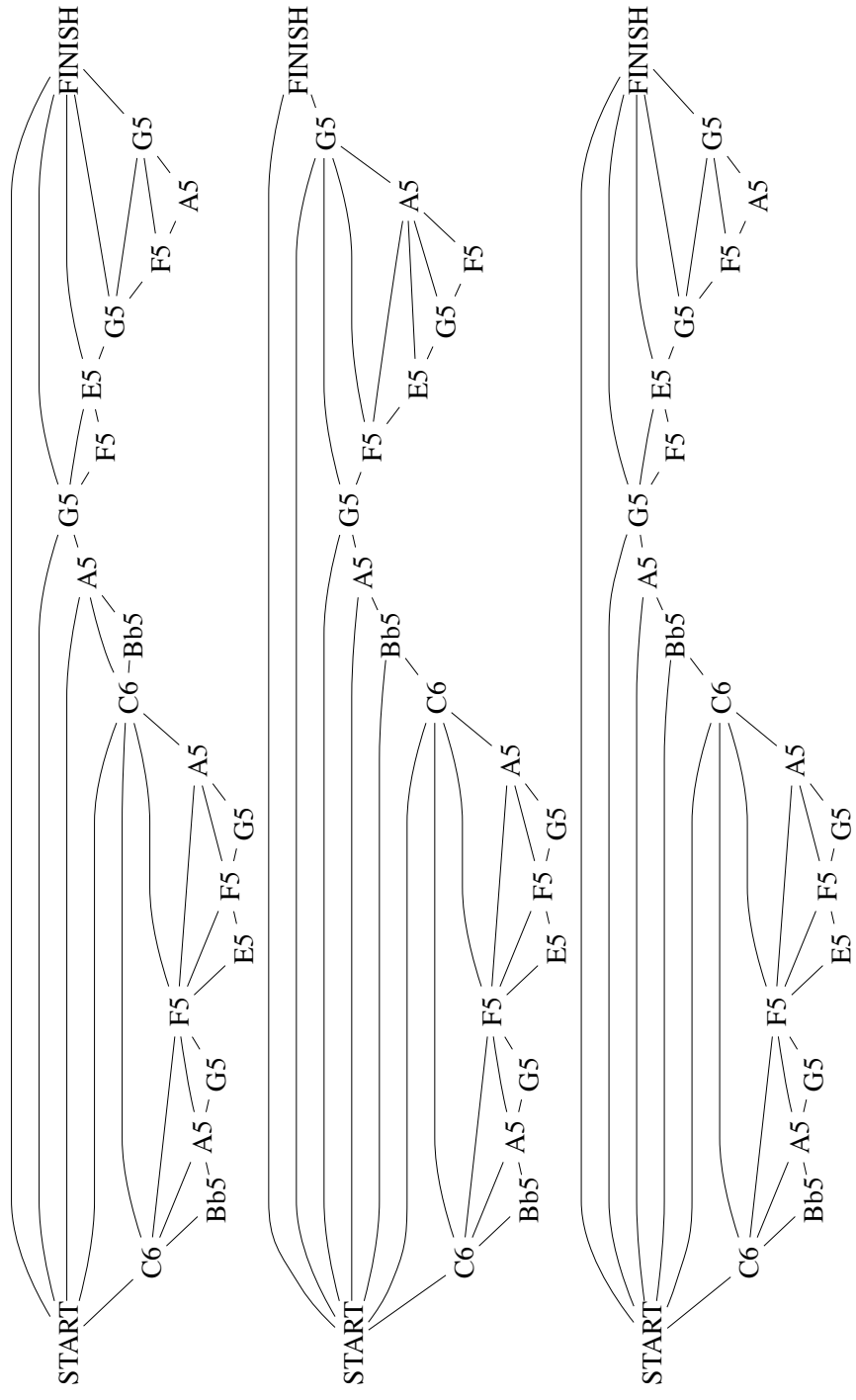


Figure A.38: MOPs produced by PARSEMOP-A, -B, and -C for beethoven1a

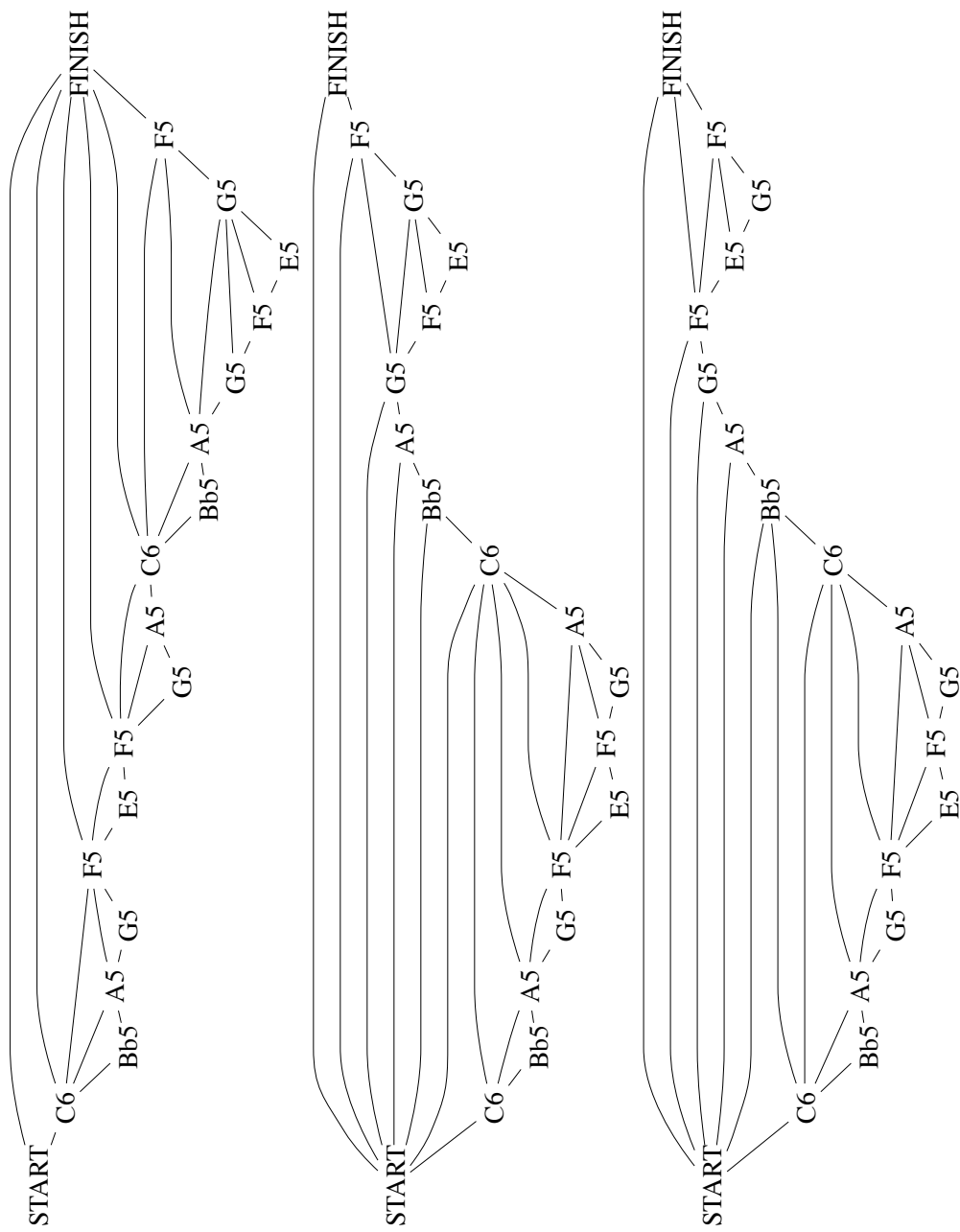


Figure A.39: MOPs produced by PARSEMOP-A, -B, and -C for beethoven1b

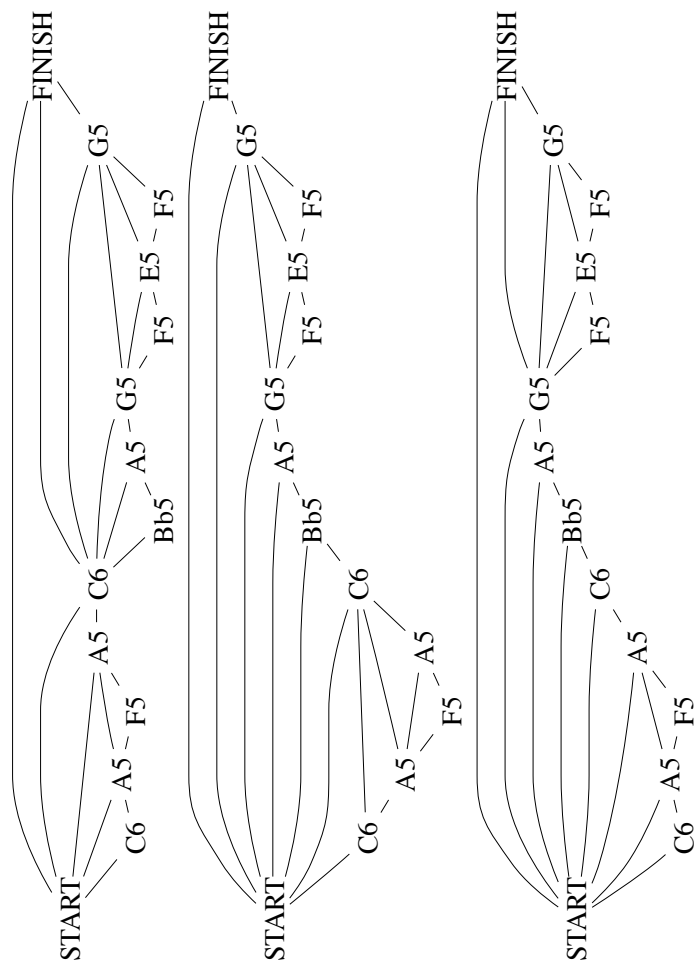


Figure A.40: MOPs produced by PARSEMOP-A, -B, and -C for beethoven2a

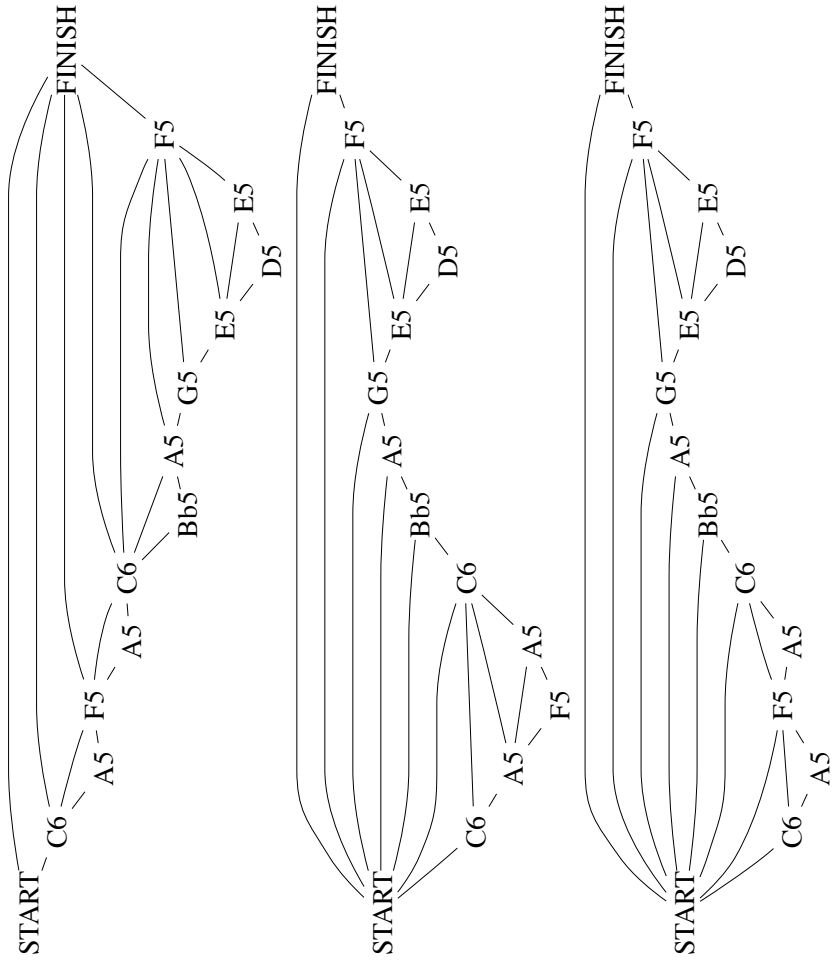


Figure A.41: MOPs produced by PARSEMOP-A, -B, and -C for beethoven2b

Excerpt identifier: beethoven3

MOP(s) contained in this excerpt: beethoven3a, beethoven3b

Score:

A musical score for piano in D major, 4/4 time. The score consists of two staves: a treble clef staff and a bass clef staff. The treble staff contains a melodic line with eighth and quarter notes, and a few dotted notes. The bass staff contains a simple accompaniment of quarter and eighth notes. The key signature has two sharps (F# and C#), and the time signature is 4/4.

Textbook analysis:

A musical staff showing the first measure of the excerpt. The notes are grouped with a slur. Below the staff, the chord analysis is given as "D: I" under the first note and "V" under the last note of the measure.

A musical staff showing the second measure of the excerpt. The notes are grouped with a slur. Below the staff, the chord analysis is given as "D: I" under the first note, "V" under the penultimate note, and "I" under the final note of the measure.

PARSEMOP-C analysis:

A musical staff showing the first measure of the excerpt. The notes are grouped with a slur. Below the staff, the chord analysis is given as "D: I" under the first note and "V" under the last note of the measure.

A musical staff showing the second measure of the excerpt. The notes are grouped with a slur. Below the staff, the chord analysis is given as "D: I" under the first note, "V" under the penultimate note, and "I" under the final note of the measure.

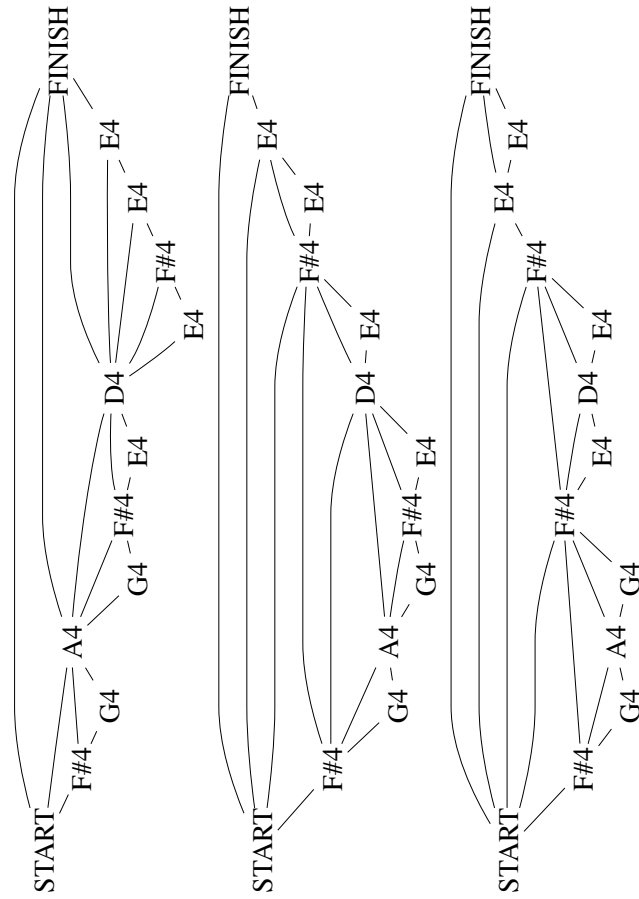


Figure A.42: MOPs produced by PARSEMOP-A, -B, and -C for beethoven3a

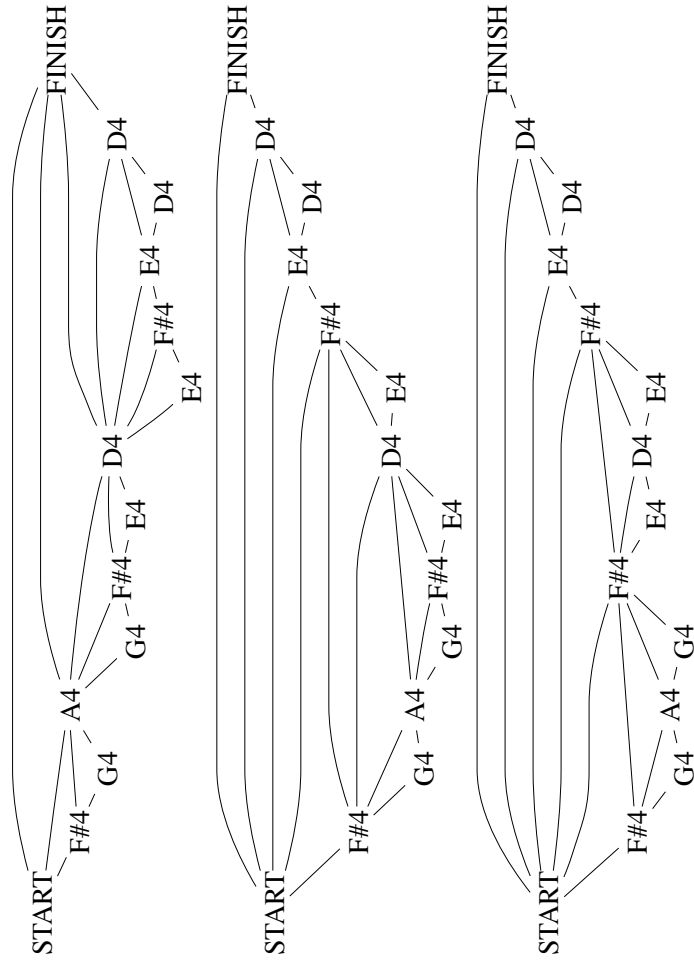


Figure A.43: MOPs produced by PARSEMOP-A, -B, and -C for beethoven3b

Excerpt identifier: beethoven4

MOP(s) contained in this excerpt: beethoven4a

Score:

A musical score for piano in 3/4 time, featuring a melody in the right hand and a bass line in the left hand. The key signature has one flat (B-flat). The melody consists of a series of eighth notes, with a chromatic alteration in the third measure. The bass line provides harmonic support with a steady eighth-note pattern.

Textbook analysis:

A textbook analysis of the musical score, showing chord progressions labeled F: V I, II, V, I. The analysis is presented on a single staff with a treble clef and a key signature of one flat. The chords are indicated by Roman numerals: V, I, II, V, I. A bracket above the staff groups the first four measures, and another bracket groups the last four measures.

PARSEMOP-C analysis:

A PARSEMOP-C analysis of the musical score, showing chord progressions labeled F: V I, II, V, I. The analysis is presented on a single staff with a treble clef and a key signature of one flat. The chords are indicated by Roman numerals: V, I, II, V, I. A bracket above the staff groups the first four measures, and another bracket groups the last four measures.

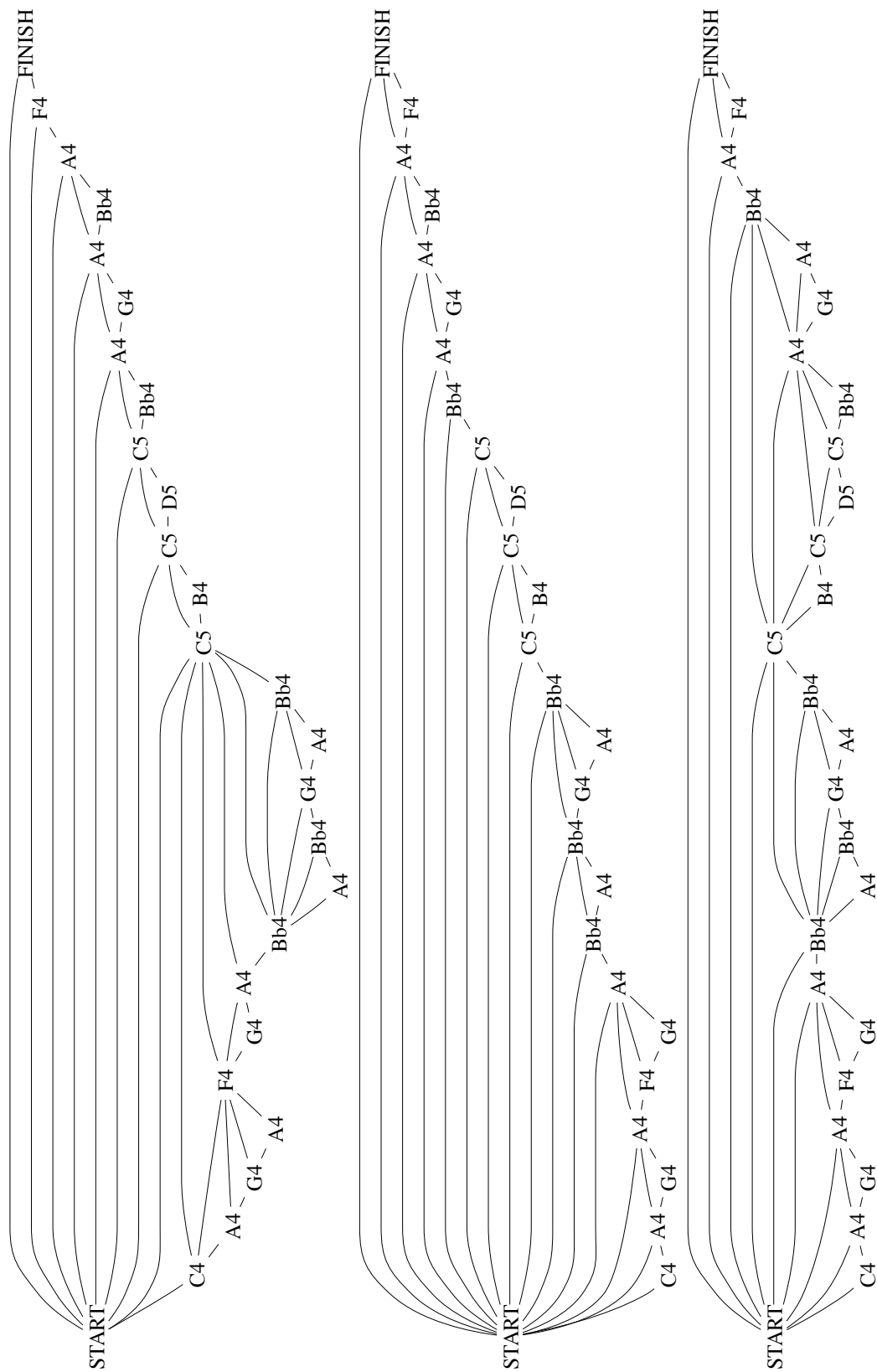


Figure A.44: MOPs produced by PARSEMOP-A, -B, and -C for beethoven4a

Excerpt identifier: beethoven5

MOP(s) contained in this excerpt: beethoven5a

Score:

A musical score for piano in 3/4 time, consisting of six measures. The treble staff contains chords and melodic fragments, while the bass staff contains a steady eighth-note accompaniment. The key signature is one sharp (F#).

Textbook analysis:

A single staff of music showing the melodic line from the excerpt. The notes are connected by slurs, and a large bracket above the staff spans the entire six-measure phrase.

C: I II V I IV VI II V I

PARSEMOP-C analysis:

A single staff of music showing the melodic line from the excerpt, identical to the textbook analysis. The notes are connected by slurs, and a large bracket above the staff spans the entire six-measure phrase.

C: I II V I IV VI II V I

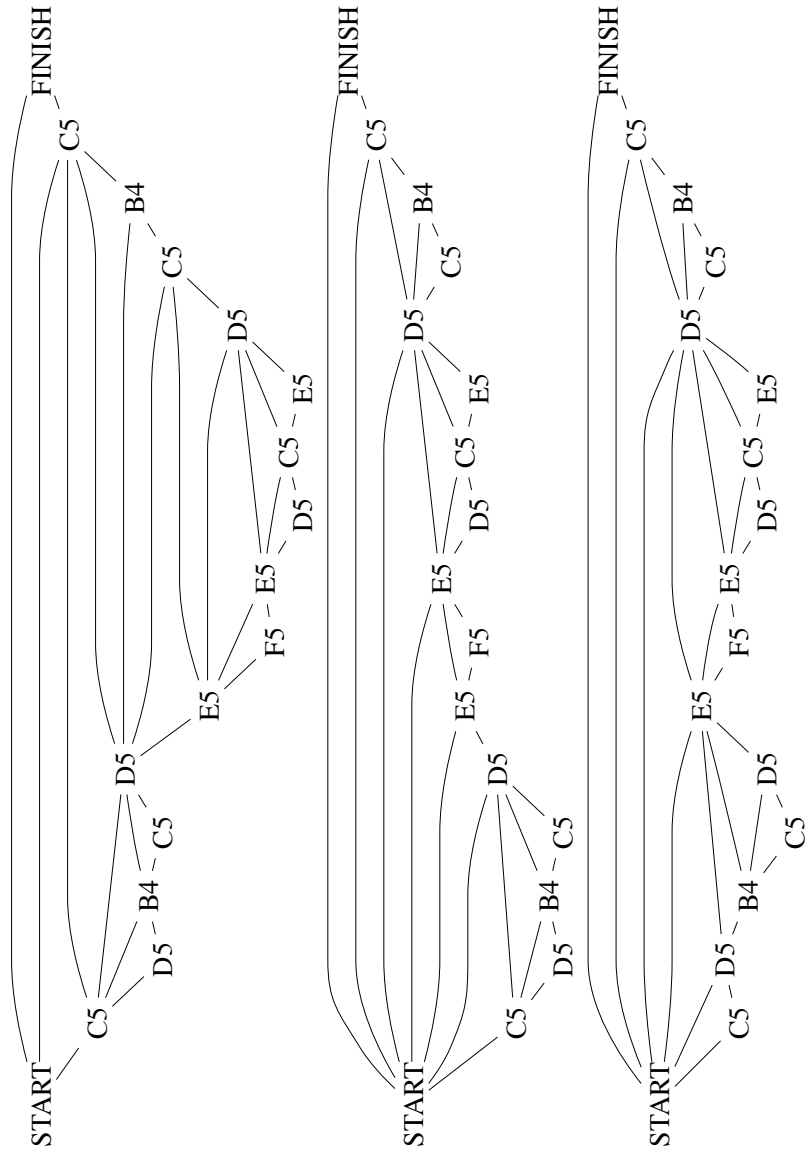


Figure A.45: MOPs produced by PARSEMOP-A, -B, and -C for beethoven5a

Excerpt identifier: schubert1

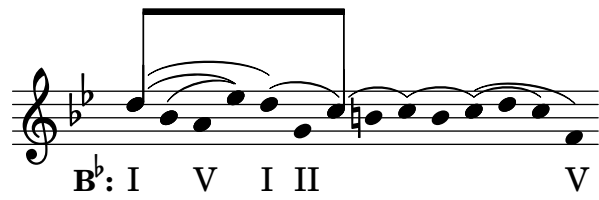
MOP(s) contained in this excerpt: schubert1a, schubert1b

Score:

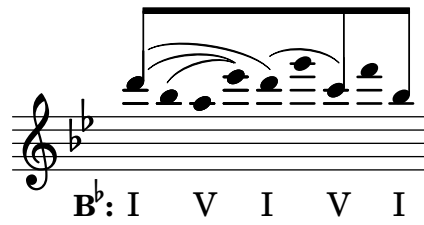
Andante



Textbook analysis:



B^b: I V I II V




B^b: I V I V I

PARSEMOP-C analysis:



B^b: I V I II V



B^b: I V I V I

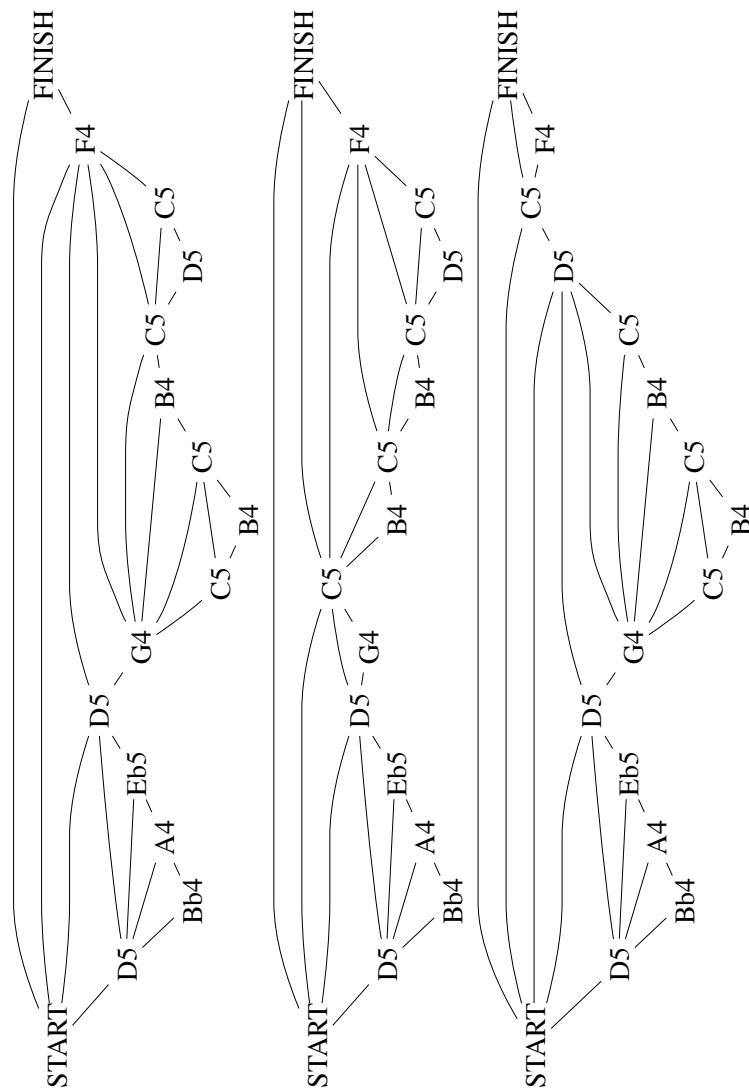


Figure A.46: MOPs produced by PARSEMOP-A, -B, and -C for schubert1a

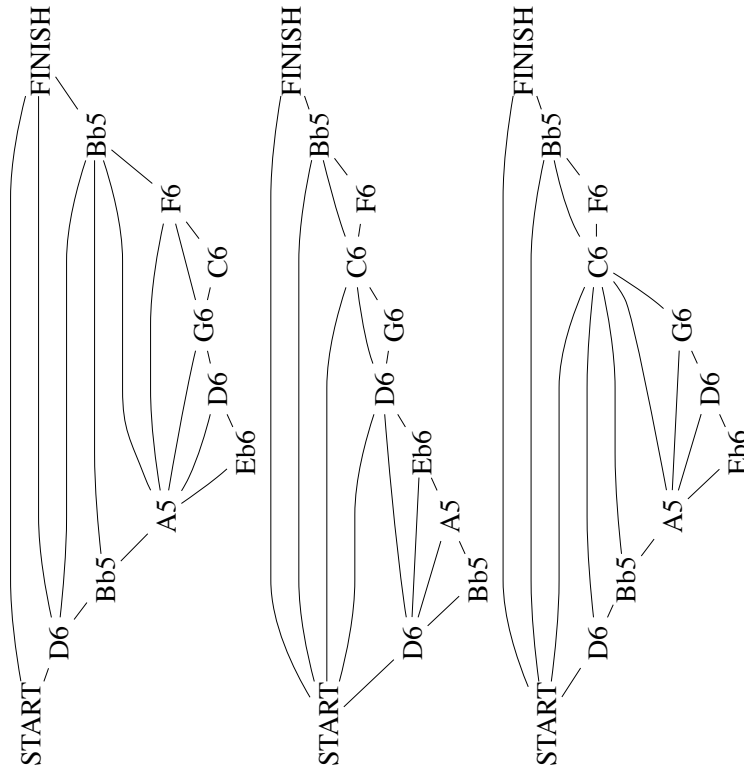


Figure A.47: MOPs produced by PARSEMOP-A, -B, and -C for schubert1b

Excerpt identifier: schubert2

MOP(s) contained in this excerpt: schubert2a, schubert2b

Score:

Allegretto

The score is written for piano and consists of five systems. Each system contains two staves: a treble staff and a bass staff. The key signature is G-flat major (three flats: B-flat, E-flat, A-flat), and the time signature is 2/4. The tempo is marked 'Allegretto'. The right hand in the treble staff plays a rhythmic pattern of eighth notes, often with a fermata over a specific note. The left hand in the bass staff provides harmonic accompaniment with chords and moving lines. The piece ends with a double bar line.

Textbook analysis:

The textbook analysis shows a treble staff with a melodic line and a bass staff with a harmonic line. A box highlights the first two measures of the treble staff. Below the bass staff, the chord progression is identified as G-flat: I VI II V I V.

G^b: I VI IV II V I

PARSEMOP-C analysis:

G^b: I VI II V I V

G^b: I VI IV II V I

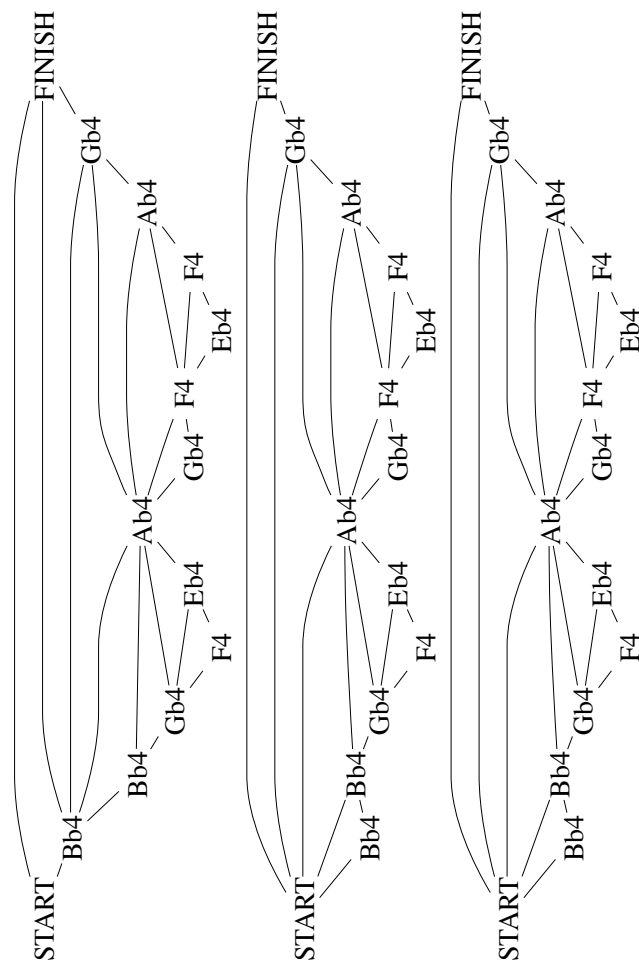


Figure A.49: MOPs produced by PARSEMOP-A, -B, and -C for schubert2b

Excerpt identifier: schubert3

MOP(s) contained in this excerpt: schubert3a

Score:

Textbook analysis:

A^b: VI V I VI VI V I V I

PARSEMOP-C analysis:

A^b: VI V I VI VI V I V I

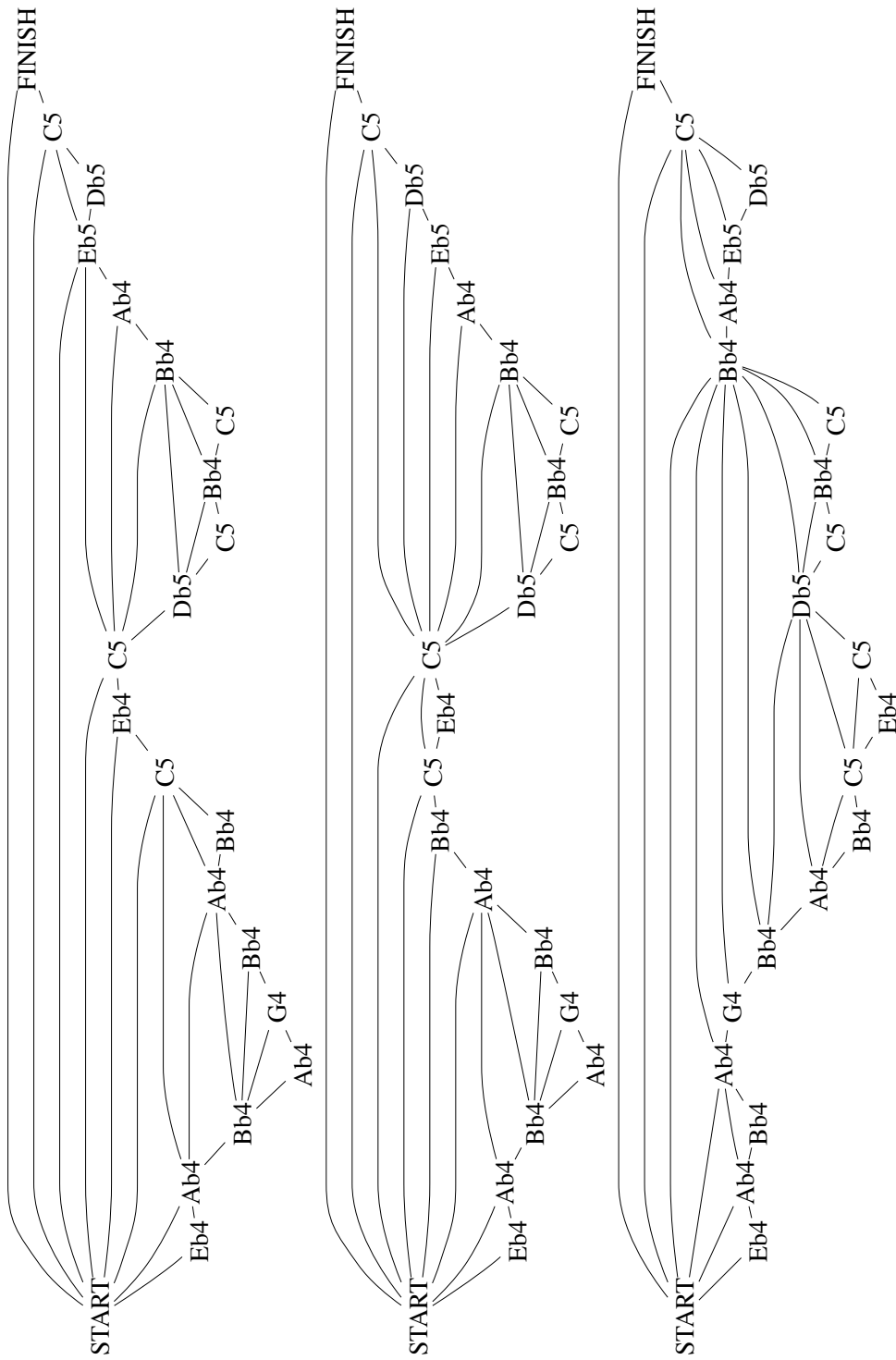


Figure A.50: MOPs produced by PARSEMOP-A, -B, and -C for schubert3a

Excerpt identifier: schubert4

MOP(s) contained in this excerpt: schubert4a

Score:

A musical score for Schubert's 'Schubert4a' in G-flat major, 4/4 time. The score consists of three staves: a single treble clef staff for the melody and a grand staff (treble and bass clefs) for the piano accompaniment. The melody is a simple eighth-note sequence: G4, A4, B4, C5, B4, A4, G4. The piano accompaniment features a steady eighth-note bass line and chords in the right hand.

Textbook analysis:

A textbook analysis of the melody from 'Schubert4a'. It shows the melody on a treble clef staff with a key signature of two flats and a 4/4 time signature. The melody is G4, A4, B4, C5, B4, A4, G4. Below the staff, the chord progression is indicated as G^b: I VI II V I.

PARSEMOP-C analysis:

A PARSEMOP-C analysis of the melody from 'Schubert4a'. It shows the melody on a treble clef staff with a key signature of two flats and a 4/4 time signature. The melody is G4, A4, B4, C5, B4, A4, G4. Below the staff, the chord progression is indicated as G^b: I VI II V I.

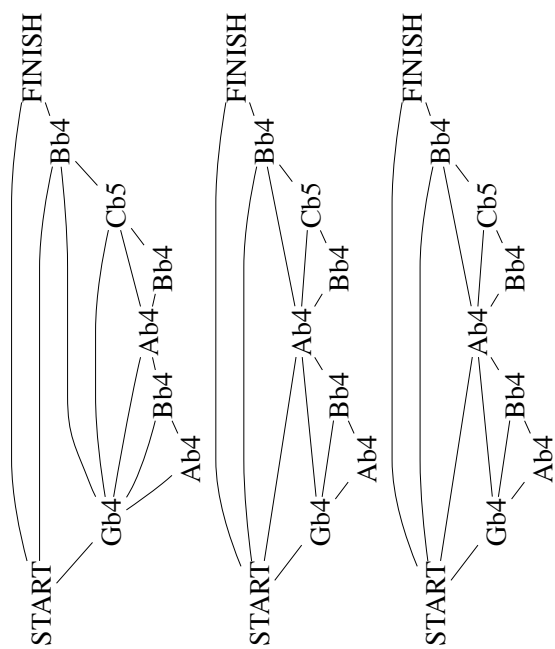


Figure A.51: MOPs produced by PARSEMOP-A, -B, and -C for schubert4a

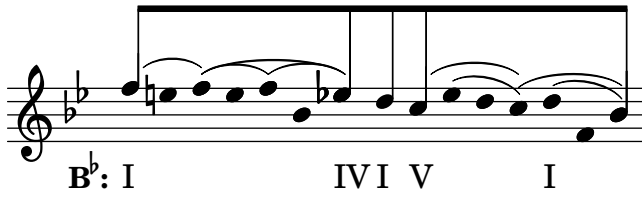
Excerpt identifier: chopin1

MOP(s) contained in this excerpt: chopin1a

Score:

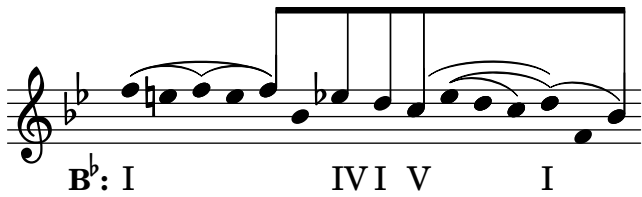


Textbook analysis:



B^b: I **IVI V** **I**

PARSEMOP-C analysis:



B^b: I **IVI V** **I**

Excerpt identifier: chopin2

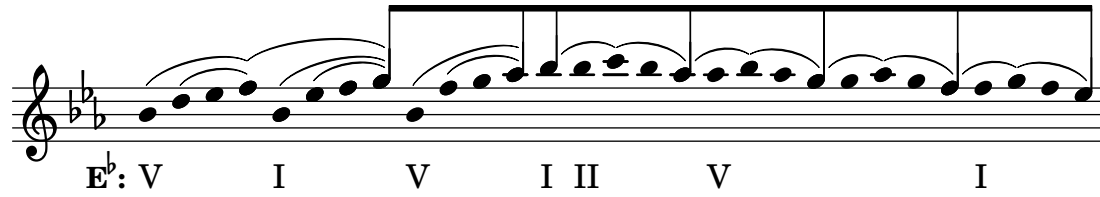
MOP(s) contained in this excerpt: chopin2a

Score:

Vivace

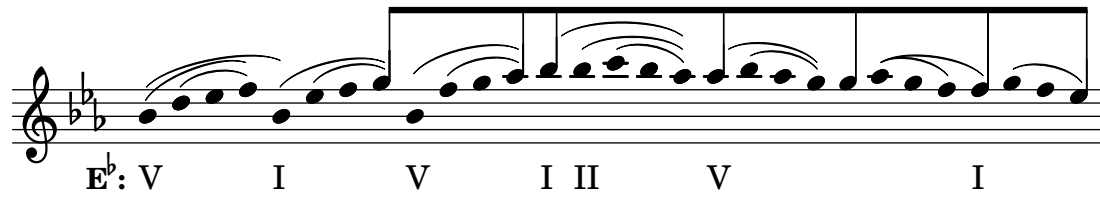


Textbook analysis:



E^b: V I V I II V I

PARSEMOP-C analysis:



E^b: V I V I II V I

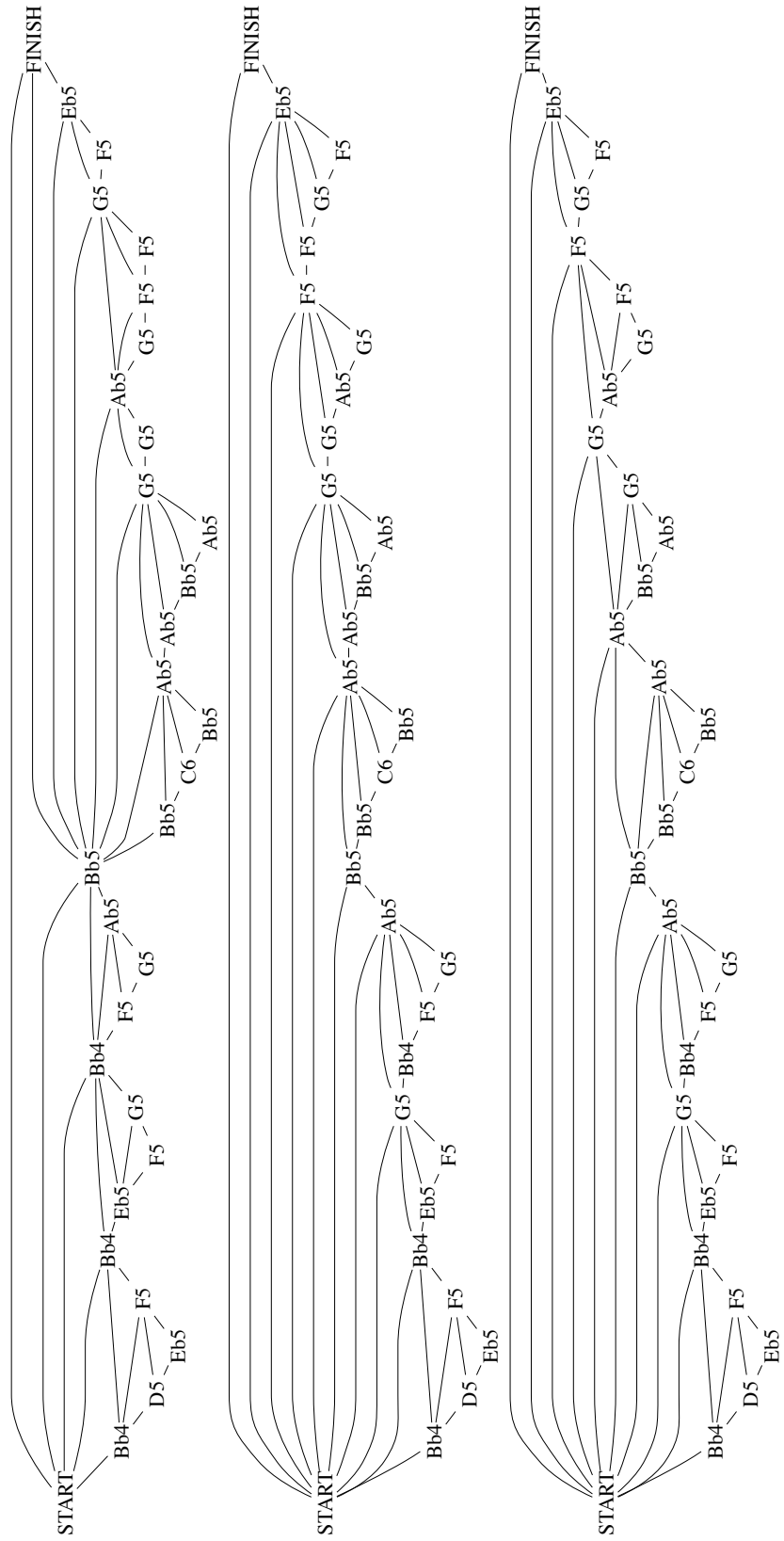


Figure A.53: MOPs produced by PARSEMOP-A, -B, and -C for chopin2a

APPENDIX B

MAXIMUM ACCURACY AS A FUNCTION OF RANK

Tables B.1 through B.6 show the maximum accuracy of the top n ranked PARSEMOP analyses that were produced through leave-one-out cross-validation. The six tables account for all combinations of the the three PARSEMOP variants and the two accuracy metrics: triangle- and edge-based.

Excerpt	Accuracy rank 1	Max acc. ranks 1–5	Max acc. ranks 1–20	Max acc. ranks 1–50	Max acc. ranks 1–100	Max acc. ranks 1–500
bach1	0.57	0.59	0.59	0.60	0.64	0.66
bach2	0.83	1.00	1.00	1.00	1.00	1.00
bach3	0.41	0.53	0.65	0.65	0.65	0.71
handel1	0.39	0.58	0.61	0.61	0.64	0.67
haydn1	0.48	0.48	0.52	0.55	0.58	0.61
haydn2	0.36	0.45	0.50	0.55	0.64	0.77
haydn3	0.89	0.89	0.89	0.89	0.89	0.89
haydn4	0.44	0.56	0.56	1.00	1.00	1.00
haydn5	0.12	0.75	1.00	1.00	1.00	1.00
haydn6	0.29	0.57	0.79	0.79	0.79	1.00
haydn7	1.00	1.00	1.00	1.00	1.00	1.00
clementi1	0.64	0.64	0.64	0.82	0.82	1.00
mozart1	0.87	0.87	0.87	0.87	0.87	0.93
mozart2	0.66	0.81	0.81	0.84	0.84	0.84
mozart3	0.34	0.40	0.46	0.49	0.49	0.54
mozart4	0.54	0.62	0.65	0.85	0.85	0.85
mozart5	0.33	0.47	0.47	0.50	0.60	0.63
mozart6	0.48	0.52	0.52	0.57	0.67	0.71
mozart7	0.61	0.61	0.61	0.65	0.65	0.78
mozart8	0.47	0.53	0.53	0.55	0.60	0.62
mozart9	1.00	1.00	1.00	1.00	1.00	1.00
mozart10	0.61	0.61	0.67	0.67	0.72	0.78
mozart11	0.42	0.46	0.50	0.54	0.65	0.73
mozart12	0.43	1.00	1.00	1.00	1.00	1.00
mozart13	0.33	1.00	1.00	1.00	1.00	1.00
mozart14	0.78	1.00	1.00	1.00	1.00	1.00
mozart15	1.00	1.00	1.00	1.00	1.00	1.00
mozart16	0.56	0.62	0.69	0.75	0.81	0.81
mozart17	0.18	0.32	0.32	0.36	0.36	0.43
mozart18	0.87	0.91	0.91	0.91	0.91	0.91
beethoven1	0.64	0.67	0.75	0.75	0.75	0.83
beethoven2	0.75	0.75	0.83	0.83	0.83	0.92
beethoven3	0.48	0.70	0.91	1.00	1.00	1.00
beethoven4	0.60	0.72	0.80	0.80	0.80	0.88
beethoven5	0.73	1.00	1.00	1.00	1.00	1.00
schubert1	0.39	0.43	0.43	0.52	0.52	0.61
schubert2	0.74	0.83	0.83	0.91	0.91	0.91
schubert3	0.45	0.60	0.60	0.60	0.60	0.70
schubert4	0.71	1.00	1.00	1.00	1.00	1.00
chopin1	0.60	0.60	0.67	0.73	0.73	0.73
chopin2	0.48	0.52	0.52	0.52	0.52	0.59

Table B.1: Maximum triangle accuracy obtained using PARSEMOP-A for each musical excerpt at various rank levels.

Excerpt	Accuracy rank 1	Max acc. ranks 1–5	Max acc. ranks 1–20	Max acc. ranks 1–50	Max acc. ranks 1–100	Max acc. ranks 1–500
bach1	0.64	0.68	0.68	0.68	0.71	0.75
bach2	0.91	1.00	1.00	1.00	1.00	1.00
bach3	0.56	0.69	0.75	0.75	0.75	0.81
handel1	0.47	0.62	0.66	0.66	0.66	0.69
haydn1	0.59	0.59	0.62	0.66	0.69	0.69
haydn2	0.48	0.52	0.57	0.62	0.67	0.81
haydn3	0.91	0.91	0.91	0.91	0.91	0.91
haydn4	0.50	0.62	0.62	1.00	1.00	1.00
haydn5	0.29	0.86	1.00	1.00	1.00	1.00
haydn6	0.46	0.62	0.85	0.85	0.85	1.00
haydn7	1.00	1.00	1.00	1.00	1.00	1.00
clementi1	0.70	0.70	0.80	0.90	0.90	1.00
mozart1	0.93	0.93	0.93	0.93	0.93	0.96
mozart2	0.70	0.83	0.83	0.90	0.90	0.90
mozart3	0.42	0.48	0.55	0.58	0.58	0.64
mozart4	0.58	0.67	0.75	0.92	0.92	0.92
mozart5	0.41	0.52	0.52	0.55	0.66	0.69
mozart6	0.50	0.55	0.70	0.70	0.75	0.80
mozart7	0.73	0.73	0.73	0.77	0.77	0.86
mozart8	0.56	0.62	0.62	0.63	0.67	0.69
mozart9	1.00	1.00	1.00	1.00	1.00	1.00
mozart10	0.65	0.65	0.71	0.71	0.76	0.82
mozart11	0.52	0.56	0.64	0.68	0.72	0.84
mozart12	0.50	1.00	1.00	1.00	1.00	1.00
mozart13	0.40	1.00	1.00	1.00	1.00	1.00
mozart14	0.88	1.00	1.00	1.00	1.00	1.00
mozart15	1.00	1.00	1.00	1.00	1.00	1.00
mozart16	0.60	0.67	0.73	0.80	0.87	0.87
mozart17	0.27	0.35	0.38	0.42	0.42	0.50
mozart18	0.91	0.95	0.95	0.95	0.95	0.95
beethoven1	0.76	0.79	0.85	0.85	0.85	0.91
beethoven2	0.82	0.82	0.91	0.91	0.91	0.95
beethoven3	0.52	0.76	0.95	1.00	1.00	1.00
beethoven4	0.71	0.79	0.83	0.83	0.83	0.92
beethoven5	0.79	1.00	1.00	1.00	1.00	1.00
schubert1	0.43	0.48	0.52	0.62	0.62	0.67
schubert2	0.76	0.86	0.86	0.95	0.95	0.95
schubert3	0.63	0.74	0.74	0.74	0.79	0.84
schubert4	0.83	1.00	1.00	1.00	1.00	1.00
chopin1	0.64	0.64	0.71	0.79	0.79	0.79
chopin2	0.61	0.64	0.64	0.64	0.64	0.68

Table B.2: Maximum edge accuracy obtained using PARSEMOP-A for each musical excerpt at various rank levels.

Excerpt	Accuracy rank 1	Max acc. ranks 1–5	Max acc. ranks 1–20	Max acc. ranks 1–50	Max acc. ranks 1–100	Max acc. ranks 1–500
bach1	0.84	0.84	0.84	0.84	0.86	0.90
bach2	0.83	1.00	1.00	1.00	1.00	1.00
bach3	0.76	0.88	0.88	1.00	1.00	1.00
handel1	0.76	0.82	0.88	0.88	0.88	0.91
haydn1	0.71	0.84	0.84	0.84	0.84	0.89
haydn2	0.82	0.91	0.91	1.00	1.00	1.00
haydn3	1.00	1.00	1.00	1.00	1.00	1.00
haydn4	1.00	1.00	1.00	1.00	1.00	1.00
haydn5	1.00	1.00	1.00	1.00	1.00	1.00
haydn6	1.00	1.00	1.00	1.00	1.00	1.00
haydn7	1.00	1.00	1.00	1.00	1.00	1.00
clementi1	1.00	1.00	1.00	1.00	1.00	1.00
mozart1	0.87	0.93	1.00	1.00	1.00	1.00
mozart2	0.91	0.91	0.91	0.91	0.91	1.00
mozart3	1.00	1.00	1.00	1.00	1.00	1.00
mozart4	0.92	0.92	0.92	0.92	0.92	0.92
mozart5	0.90	0.90	0.90	1.00	1.00	1.00
mozart6	0.67	0.81	0.81	0.86	0.86	0.86
mozart7	0.87	1.00	1.00	1.00	1.00	1.00
mozart8	0.70	0.75	0.75	0.75	0.75	0.79
mozart9	1.00	1.00	1.00	1.00	1.00	1.00
mozart10	0.89	1.00	1.00	1.00	1.00	1.00
mozart11	0.85	0.85	0.85	0.92	0.92	1.00
mozart12	1.00	1.00	1.00	1.00	1.00	1.00
mozart13	1.00	1.00	1.00	1.00	1.00	1.00
mozart14	0.78	1.00	1.00	1.00	1.00	1.00
mozart15	1.00	1.00	1.00	1.00	1.00	1.00
mozart16	1.00	1.00	1.00	1.00	1.00	1.00
mozart17	0.71	0.86	1.00	1.00	1.00	1.00
mozart18	1.00	1.00	1.00	1.00	1.00	1.00
beethoven1	0.64	0.81	0.89	0.92	0.92	0.92
beethoven2	0.75	0.92	0.92	0.92	0.92	0.92
beethoven3	1.00	1.00	1.00	1.00	1.00	1.00
beethoven4	1.00	1.00	1.00	1.00	1.00	1.00
beethoven5	1.00	1.00	1.00	1.00	1.00	1.00
schubert1	1.00	1.00	1.00	1.00	1.00	1.00
schubert2	0.83	0.91	0.91	0.91	0.91	0.91
schubert3	0.55	0.65	0.70	0.80	0.80	0.90
schubert4	1.00	1.00	1.00	1.00	1.00	1.00
chopin1	1.00	1.00	1.00	1.00	1.00	1.00
chopin2	0.62	0.69	0.76	0.76	0.76	0.83

Table B.3: Maximum triangle accuracy obtained using PARSEMOP-B for each musical excerpt at various rank levels.

Excerpt	Accuracy rank 1	Max acc. ranks 1–5	Max acc. ranks 1–20	Max acc. ranks 1–50	Max acc. ranks 1–100	Max acc. ranks 1–500
bach1	0.89	0.89	0.89	0.89	0.91	0.93
bach2	0.91	1.00	1.00	1.00	1.00	1.00
bach3	0.81	0.94	0.94	1.00	1.00	1.00
handel1	0.81	0.88	0.91	0.91	0.91	0.94
haydn1	0.79	0.86	0.86	0.88	0.88	0.94
haydn2	0.86	0.95	0.95	1.00	1.00	1.00
haydn3	1.00	1.00	1.00	1.00	1.00	1.00
haydn4	1.00	1.00	1.00	1.00	1.00	1.00
haydn5	1.00	1.00	1.00	1.00	1.00	1.00
haydn6	1.00	1.00	1.00	1.00	1.00	1.00
haydn7	1.00	1.00	1.00	1.00	1.00	1.00
clementi1	1.00	1.00	1.00	1.00	1.00	1.00
mozart1	0.93	0.96	1.00	1.00	1.00	1.00
mozart2	0.93	0.93	0.93	0.93	0.93	1.00
mozart3	1.00	1.00	1.00	1.00	1.00	1.00
mozart4	0.96	0.96	0.96	0.96	0.96	0.96
mozart5	0.93	0.93	0.93	1.00	1.00	1.00
mozart6	0.75	0.85	0.85	0.90	0.90	0.90
mozart7	0.91	1.00	1.00	1.00	1.00	1.00
mozart8	0.75	0.81	0.81	0.81	0.81	0.83
mozart9	1.00	1.00	1.00	1.00	1.00	1.00
mozart10	0.94	1.00	1.00	1.00	1.00	1.00
mozart11	0.92	0.92	0.92	0.96	0.96	1.00
mozart12	1.00	1.00	1.00	1.00	1.00	1.00
mozart13	1.00	1.00	1.00	1.00	1.00	1.00
mozart14	0.88	1.00	1.00	1.00	1.00	1.00
mozart15	1.00	1.00	1.00	1.00	1.00	1.00
mozart16	1.00	1.00	1.00	1.00	1.00	1.00
mozart17	0.85	0.92	1.00	1.00	1.00	1.00
mozart18	1.00	1.00	1.00	1.00	1.00	1.00
beethoven1	0.76	0.88	0.94	0.94	0.94	0.94
beethoven2	0.82	0.95	0.95	0.95	0.95	0.95
beethoven3	1.00	1.00	1.00	1.00	1.00	1.00
beethoven4	1.00	1.00	1.00	1.00	1.00	1.00
beethoven5	1.00	1.00	1.00	1.00	1.00	1.00
schubert1	1.00	1.00	1.00	1.00	1.00	1.00
schubert2	0.86	0.95	0.95	0.95	0.95	0.95
schubert3	0.68	0.74	0.79	0.84	0.84	0.95
schubert4	1.00	1.00	1.00	1.00	1.00	1.00
chopin1	1.00	1.00	1.00	1.00	1.00	1.00
chopin2	0.79	0.82	0.86	0.86	0.86	0.89

Table B.4: Maximum edge accuracy obtained using PARSEMOP-B for each musical excerpt at various rank levels.

Excerpt	Accuracy rank 1	Max acc. ranks 1–5	Max acc. ranks 1–20	Max acc. ranks 1–50	Max acc. ranks 1–100	Max acc. ranks 1–500
bach1	0.57	0.57	0.57	0.59	0.59	0.62
bach2	0.50	0.83	1.00	1.00	1.00	1.00
bach3	0.24	0.47	0.47	0.53	0.65	0.88
handel1	0.61	0.61	0.64	0.67	0.70	0.70
haydn1	0.35	0.48	0.48	0.58	0.58	0.65
haydn2	0.45	0.73	0.77	0.82	0.82	0.91
haydn3	0.86	0.86	0.86	0.86	0.91	0.91
haydn4	1.00	1.00	1.00	1.00	1.00	1.00
haydn5	1.00	1.00	1.00	1.00	1.00	1.00
haydn6	0.57	0.79	1.00	1.00	1.00	1.00
haydn7	0.74	0.79	0.89	1.00	1.00	1.00
clementi1	1.00	1.00	1.00	1.00	1.00	1.00
mozart1	0.87	0.87	0.87	0.93	0.93	0.93
mozart2	0.91	0.91	0.91	0.91	0.91	0.91
mozart3	0.74	0.74	0.77	0.94	0.94	0.94
mozart4	0.77	0.92	0.92	0.92	0.92	1.00
mozart5	0.43	0.43	0.57	0.60	0.63	0.67
mozart6	0.48	0.62	0.71	0.71	0.71	0.76
mozart7	0.43	0.70	0.78	0.78	0.83	0.83
mozart8	0.40	0.40	0.43	0.45	0.45	0.49
mozart9	1.00	1.00	1.00	1.00	1.00	1.00
mozart10	0.67	0.67	0.67	0.78	0.78	0.78
mozart11	0.65	0.65	0.73	0.73	0.73	0.81
mozart12	0.29	0.43	1.00	1.00	1.00	1.00
mozart13	1.00	1.00	1.00	1.00	1.00	1.00
mozart14	0.56	0.78	1.00	1.00	1.00	1.00
mozart15	1.00	1.00	1.00	1.00	1.00	1.00
mozart16	1.00	1.00	1.00	1.00	1.00	1.00
mozart17	0.25	0.36	0.54	0.54	0.54	0.57
mozart18	0.57	0.57	0.65	0.70	0.70	0.87
beethoven1	0.58	0.69	0.69	0.81	0.81	0.81
beethoven2	0.75	0.75	0.79	0.92	0.92	1.00
beethoven3	0.70	0.74	0.83	1.00	1.00	1.00
beethoven4	0.76	0.84	0.88	0.92	0.92	0.92
beethoven5	0.73	1.00	1.00	1.00	1.00	1.00
schubert1	0.43	0.48	0.65	0.65	0.65	1.00
schubert2	0.78	0.83	0.83	0.83	0.83	0.83
schubert3	0.40	0.40	0.45	0.55	0.55	0.70
schubert4	1.00	1.00	1.00	1.00	1.00	1.00
chopin1	0.80	0.80	0.80	1.00	1.00	1.00
chopin2	0.48	0.48	0.52	0.55	0.55	0.59

Table B.5: Maximum triangle accuracy obtained using PARSEMOP-C for each musical excerpt at various rank levels.

Excerpt	Accuracy rank 1	Max acc. ranks 1–5	Max acc. ranks 1–20	Max acc. ranks 1–50	Max acc. ranks 1–100	Max acc. ranks 1–500
bach1	0.62	0.68	0.68	0.68	0.71	0.71
bach2	0.64	0.91	1.00	1.00	1.00	1.00
bach3	0.38	0.56	0.56	0.69	0.81	0.94
handel1	0.62	0.62	0.66	0.69	0.72	0.72
haydn1	0.45	0.59	0.59	0.66	0.69	0.72
haydn2	0.48	0.76	0.81	0.86	0.90	0.95
haydn3	0.91	0.91	0.91	0.91	0.94	0.94
haydn4	1.00	1.00	1.00	1.00	1.00	1.00
haydn5	1.00	1.00	1.00	1.00	1.00	1.00
haydn6	0.69	0.85	1.00	1.00	1.00	1.00
haydn7	0.78	0.89	0.94	1.00	1.00	1.00
clementi1	1.00	1.00	1.00	1.00	1.00	1.00
mozart1	0.93	0.93	0.93	0.96	0.96	0.96
mozart2	0.93	0.93	0.93	0.93	0.93	0.93
mozart3	0.79	0.79	0.82	0.97	0.97	0.97
mozart4	0.88	0.96	0.96	0.96	0.96	1.00
mozart5	0.45	0.45	0.59	0.62	0.69	0.72
mozart6	0.60	0.70	0.80	0.80	0.80	0.85
mozart7	0.59	0.82	0.86	0.86	0.91	0.91
mozart8	0.46	0.46	0.48	0.50	0.50	0.52
mozart9	1.00	1.00	1.00	1.00	1.00	1.00
mozart10	0.71	0.71	0.76	0.82	0.82	0.82
mozart11	0.72	0.72	0.76	0.84	0.84	0.88
mozart12	0.33	0.50	1.00	1.00	1.00	1.00
mozart13	1.00	1.00	1.00	1.00	1.00	1.00
mozart14	0.75	0.88	1.00	1.00	1.00	1.00
mozart15	1.00	1.00	1.00	1.00	1.00	1.00
mozart16	1.00	1.00	1.00	1.00	1.00	1.00
mozart17	0.38	0.50	0.65	0.65	0.65	0.69
mozart18	0.64	0.64	0.73	0.77	0.77	0.91
beethoven1	0.71	0.76	0.79	0.85	0.85	0.85
beethoven2	0.82	0.82	0.86	0.95	0.95	1.00
beethoven3	0.81	0.86	0.86	1.00	1.00	1.00
beethoven4	0.83	0.88	0.92	0.96	0.96	0.96
beethoven5	0.79	1.00	1.00	1.00	1.00	1.00
schubert1	0.48	0.52	0.67	0.67	0.67	1.00
schubert2	0.81	0.86	0.86	0.90	0.90	0.90
schubert3	0.42	0.47	0.53	0.58	0.63	0.74
schubert4	1.00	1.00	1.00	1.00	1.00	1.00
chopin1	0.86	0.86	0.86	1.00	1.00	1.00
chopin2	0.57	0.57	0.57	0.64	0.64	0.71

Table B.6: Maximum edge accuracy obtained using PARSEMOP-C for each musical excerpt at various rank levels.

BIBLIOGRAPHY

- A. Abbott and A. Tsay. Sequence analysis and optimal matching methods in sociology: Review and prospect. *Sociological Methods & Research*, 29(1):3–33, Aug. 2000. doi: 10.1177/0049124100029001001.
- I. D. Bent and A. Pople. Analysis. In *Grove Music Online. Oxford Music Online*. Oxford University Press, May 2013. URL <http://www.oxfordmusiconline.com/subscriber/article/grove/music/41862pg1>. Web.
- R. Bod. Probabilistic grammars for music. In *Proceedings of the Belgian-Dutch Conference on Artificial Intelligence*, 2001.
- L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- M. Brown. *Explaining Tonality*. University of Rochester Press, 2005.
- M. Brown and D. J. Dempster. The scientific image of music theory. *Journal of Music Theory*, 33(1):65–106, 1989.
- A. Cadwallader and D. Gagné. *Analysis of Tonal Music: A Schenkerian Approach*. Oxford University Press, Oxford, 1998.
- N. Cook. Towards the compleat musicologist? Invited talk, Sixth International Conference on Music Information Retrieval, Sept. 2005.
- M. Črepinšek and L. Mernik. An efficient representation for solving Catalan number related problems. *International Journal of Pure and Applied Mathematics*, 56(4):598–604, 2009.
- J. P. da Costa and C. Soares. A weighted rank measure of correlation. *Australian & New Zealand Journal of Statistics*, 47:515–529, 2005.
- C. Dahlhaus, J. Anderson, C. Wilson, R. Cohn, and B. Hyer. Harmony. In *Grove Music Online. Oxford Music Online*. Oxford University Press, May 2013. URL <http://www.oxfordmusiconline.com/subscriber/article/grove/music/50818>. Web.
- R. B. Dannenberg. A brief survey of music representation issues, techniques, and systems. *Computer Music Journal*, 17(3):20–30, 1993.

- W. Drabkin. Part-writing. In *Grove Music Online. Oxford Music Online*. Oxford University Press, May 2013. URL <http://www.oxfordmusiconline.com/subscriber/article/grove/music/20989>. Web.
- A. Forte. Schenker's conception of musical structure. *Journal of Music Theory*, 3(1):1–30, Apr. 1959.
- A. Forte and S. E. Gilbert. *Introduction to Schenkerian Analysis*. W. W. Norton and Company, New York, 1982a.
- A. Forte and S. E. Gilbert. *Instructor's Manual for "Introduction to Schenkerian Analysis"*. W. W. Norton and Company, New York, 1982b.
- R. E. Frankel, S. J. Rosenschein, and S. W. Smoliar. A LISP-based system for the study of Schenkerian analysis. *Computers and the Humanities*, 10(1):21–32, 1976.
- R. E. Frankel, S. J. Rosenschein, and S. W. Smoliar. Schenker's theory of tonal music—its explication through computational processes. *International Journal of Man-Machine Studies*, 10(2):121–138, 1978. doi: 10.1016/S0020-7373(78)80008-X.
- É. Gilbert and D. Conklin. A probabilistic context-free grammar for melodic reduction. In *Proceedings of the International Workshop on Artificial Intelligence and Music, 20th International Joint Conference on Artificial Intelligence*, pages 83–94, Hyderabad, India, 2007.
- I. Godt. New voices and old theory. *The Journal of Musicology*, 3(3):312–319, 1984.
- Y. Goldenberg. “Journal of Music Theory” over the years: Content analysis of the articles and related aspects. *Journal of Music Theory*, 50(1):25–63, 2006.
- M. Hamanaka and S. Tojo. Interactive GTTM analyzer. In *Proceedings of the 10th International Society for Music Information Retrieval Conference*, pages 291–296, 2009.
- M. Hamanaka, K. Hirata, and S. Tojo. ATTA: Automatic time-span tree analyzer based on extended GTTM. In *Proceedings of the Sixth International Conference on Music Information Retrieval*, pages 358–365, 2005.
- M. Hamanaka, K. Hirata, and S. Tojo. Implementing “A Generative Theory of Tonal Music”. *Journal of New Music Research*, 35(4):249–277, 2006.
- M. Hamanaka, K. Hirata, and S. Tojo. ATTA: Implementing GTTM on a computer. In *Proceedings of the Eighth International Conference on Music Information Retrieval*, pages 285–286, 2007.

- V. M. Jiménez and A. Marzal. Computation of the n best parse trees for weighted and stochastic context-free grammars. In F. J. Ferri, J. M. Iñesta, A. Amin, and P. Pudil, editors, *Advances in Pattern Recognition*, volume 1876 of *Lecture Notes in Computer Science*, pages 183–192. Springer-Verlag, 2000.
- N. L. Johnson, A. W. Kemp, and S. Kotz. *Univariate Discrete Distributions*. John Wiley & Sons, Inc., Hoboken, NJ, 2005.
- D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall, second edition, 2009.
- Y. Y. Kang. Defending music theory in a multicultural curriculum. *College Music Symposium*, 46:45–63, 2006.
- M. Kassler. Proving musical theorems I: The middleground of Heinrich Schenker’s theory of tonality. Technical Report 103, Basser Department of Computer Science, School of Physics, The University of Sydney, Sydney, Australia, Aug. 1975.
- M. Kassler. APL applied in music theory. *APL Quote Quad*, 18(2):209–214, 1987. ISSN 0163-6006. doi: <http://doi.acm.org/10.1145/377719.55654>.
- P. B. Kirlin and D. D. Jensen. Probabilistic modeling of hierarchical music analysis. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, 2011.
- F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, Cambridge, MA, 1983.
- A. Marsden. Generative structural representation of tonal music. *Journal of New Music Research*, 34(4):409–428, Dec. 2005a. doi: 10.1080/09298210600578295.
- A. Marsden. Towards Schenkerian analysis by computer: A reductional matrix. In *Proceedings of the International Computer Music Conference*, pages 247–250, 2005b.
- A. Marsden. Automatic derivation of musical structure: A tool for research on Schenkerian analysis. In *Proceedings of the Eighth International Conference on Music Information Retrieval*, pages 55–58, 2007.
- A. Marsden. “What was the question?”: Music analysis and the computer. In T. Crawford and L. Gibson, editors, *Modern Methods for Musicology*, pages 137–147. Ashgate, Farnham, England, 2009.
- A. Marsden. Schenkerian analysis by computer: A proof of concept. *Journal of New Music Research*, 39(3):269–289, 2010. doi: 10.1080/09298215.2010.503898.

- A. Marsden and G. A. Wiggins. Schenkerian reduction as search. In *Proceedings of the Fourth Conference on Interdisciplinary Musicology*, Thessaloniki, Greece, July 2008.
- P. Mavromatis and M. Brown. Parsing context-free grammars for music: A computational model of Schenkerian analysis. In *Proceedings of the 8th International Conference on Music Perception & Cognition*, pages 414–415, 2004.
- J. R. Meehan. An artificial intelligence approach to tonal music theory. *Computer Music Journal*, 4(2):60–64, 1980.
- A. T. Merritt. Undergraduate training in music theory. *College Music Symposium*, 40:91–100, 2000. ISSN 00695696. Originally published in *College Music Symposium*, Volume 5, 1965.
- C. V. Palisca and I. D. Bent. Theory, theorists. In *Grove Music Online. Oxford Music Online*. Oxford University Press, May 2013. URL <http://www.oxfordmusiconline.com/subscriber/article/grove/music/44944>. Web.
- T. Pankhurst. *SchenkerGUIDE: A Brief Handbook and Website for Schenkerian Analysis*. Routledge, New York, 2008.
- F. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(3):199–215, Sept. 2003. ISSN 0885-6125. doi: 10.1023/A:1024099825458.
- J. Rahn. On some computational models of music theory. *Computer Music Journal*, 4(2):66–72, 1980.
- S. Rings. *Tonality and Transformation*. Oxford University Press, New York, 2011.
- F. Salzer. *Structural Hearing: Tonal Coherence in Music*. Charles Boni, New York, 1952.
- H. Schenker. *Der Tonwille: Pamphlets in Witness of the Immutable Laws of Music*. Oxford University Press, New York, 1921. Edited by William Drabkin, translated by Ian Bent.
- H. Schenker. *Der Freie Satz*. Universal Edition, Vienna, 1935. Published in English as *Free Composition*, translated and edited by E. Oster, Longman, 1979.
- S. W. Smoliar. A computer aid for Schenkerian analysis. *Computer Music Journal*, 2(4):41–59, 1980.
- A. Volk and A. Honingh. Mathematical and computational approaches to music: Challenges in an interdisciplinary enterprise. *Journal of Mathematics and Music*, 6(2):73–81, July 2012. doi: 10.1080/17459737.2012.704154.
- A. Volk, F. Wiering, and P. van Kranenburg. Unfolding the potential of computational musicology. *Proceedings of the International Conference on Informatics and Semiotics in Organisations*, pages 137–144, 2011.

- A. Whittall. Analysis. In *The Oxford Companion to Music*. *Oxford Music Online*. Oxford University Press, May 2013. URL <http://www.oxfordmusiconline.com/subscriber/article/opr/t114/e257>. Web.
- T. Winograd. Linguistics and the computer analysis of tonal harmony. *Journal of Music Theory*, 12(1):2–49, 1968.
- J. Yust. *Formal Models of Prolongation*. PhD thesis, University of Washington, 2006.