Musiplectics: Computational Assessment of the Complexity of Music Scores

Ethan G. Holder

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Masters of Science
in
Computer Science

Eli Tilevich, Chair
Amy Gillick
R. Ben Knapp

April 23, 2015
Blacksburg, Virginia

Keywords: Music Scores; Music Complexity Assessment; Novel Computing Domains; MusicXML

Musiplectics: Computational Assessment of the Complexity of Music Scores

Ethan G. Holder

(ABSTRACT)

In the Western classical tradition, musicians play music from notated sheet music, called a score. When playing music from a score, a musician translates its visual symbols into sequences of instrument-specific physical motions. Hence, a music score's overall complexity represents a sum of the cognitive and mechanical acuity required for its performance. For a given instrument, different notes, intervals, articulations, dynamics, key signatures, and tempo represent dissimilar levels of difficulty, which vary depending on the performer's proficiency. Individual musicians embrace this tenet, but may disagree about the degrees of difficulty.

This paper introduces *musiplectics*[1], a systematic and objective approach to computational assessment of the complexity of a music score for any instrument. Musiplectics defines computing paradigms for automatically and accurately calculating the complexity of playing a music score on a given instrument. The core concept codifies a two-phase process. First, music experts rank the relative difficulty of individual musical components (e.g., notes, intervals, dynamics, etc.) for different playing proficiencies and instruments. Second, a computing engine automatically applies this ranking to music scores and calculates their respective complexity. As a proof of concept of musiplectics, we present an automated, Web-based application called Musical Complexity Scoring (MCS) for music educators and performers. Musiplectics can engender the creation of practical computing tools for objective and expeditious assessment of a music score's suitability for the abilities of intended performers.

This thesis is based on research submitted for publication at ONWARD'15.

That this work received support from a Virginia Tech ICAT SEED grant.

[1]musiplectics = music + plectics, Greek for the study of complexity

Dedication

To the memories of Robert Brooks Nance, Katherine Gayle Nance, and Durward Alexander Holder; I sorely miss and deeply love you all.

Acknowledgments

This thesis would not have been possible without the generous help of many people. First and foremost, I would like to thank my research advisor, Dr. Eli Tilevich, for taking a chance on me as a student, guiding me through the publication process, and supporting me throughout my studies. I owe a great deal of thanks to Dr. Cliff Shaffer, a dedicated professor and reference, who has encouraged me to get involved in research in only my first semester as an undergraduate. I must also thank the numerous professionals that advocated the research process as both an undergraduate and graduate student, including Mr. Mack McGhee, Mrs. Andrea Bailey, and Mr. Adrian Rodriguez.

I generously thank my fellow researchers, Eeshan Shah and Mohammed Davoodi, without whom much of this research would not have been possible. I must express significant thanks to all of the other various professors and advisors who have worked with me over my years at Virginia Tech. Among all of these, I would be remiss if I did not expressly thank Mrs. Terry Arthur. Your guidance from the very first time I met you, before I even began classes, all the way through until my end as an undergraduate has been outstanding in every way.

Last but not least, I must thank all of my family and friends who pushed me to achieve as much as I have. I especially thank my mother, Laurel Holder, and father, Mark Holder, for instilling a love of learning in me from a very young age and for constantly encouraging me even when I had stumbled along my way. I owe it all to you, and I can never thank you enough.

We would like to express our gratitude to the members of the Interdisciplinary Research Honor Society at Virginia Tech for their feedback during the early stages of this project. This project is supported in part by funding from a Virginia Tech ICAT SEED grant.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

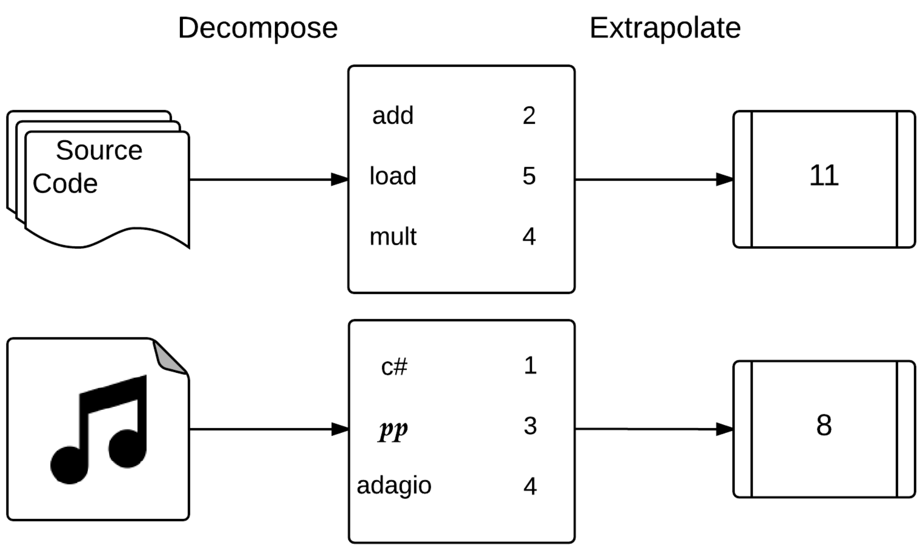
Which piano concerto is more difficult: Rachmaninoffs Second or Third? A newly appointed band director wonders if this new orchestral score is appropriate for a high school band, given that the clarinet and bassoon sections are quite advanced, while the flute and oboe sections are more novice. Music educators working on pedagogical guidelines for K-12 students are trying to decide whether a given piece belongs in the N or N+1 curricular level. A publisher wonders which audience to target when marketing new works, while the publisher's customers face great uncertainty when determining whether unfamiliar music matches their playing ability. Performers, band directors, educators, and publishers encounter these non-trivial questions throughout their professional careers.

Unfortunately, determining the relative complexity of music is a non-trivial cognitive task. Additionally, methods in the current state of the art depend solely on individual opinions, a process influenced by personal biases and lacking common criteria. In other words, the only way to answer these questions in a viable way is to carefully analyze music scores by hand, a tedious, error-prone, and time-consuming process. The stakeholders at hand would rather spend their precious time on more creative pursuits.

Can computing help decode these persistent and challenging questions? Is it possible to provide such technology in a ubiquitous and user-friendly way, accessible to any interested musician? To answer these questions, this paper presents musiplectics, a new computational paradigm, that systematically evaluates the relative difficulty of music scores, thus benefiting educators and performers. Two insights provide a foundation behind musiplectics. First, certain notes and other musical components, including intervals, dynamics, and articulations, are harder to play than the others. Second, automated computer processing can transform a prohibitively tedious, error-prone, and subjective process into a practical and pragmatic solution if exposed via an intuitive user interface. Hence, musiplectics fuses commonly accepted music tenets and novel computing paradigms, to objectively answer the questions above.

Musiplectics draws its inspiration from computational thinking Wing [2006]. The problem of estimating the expected performance efficiency of a given program has been studied in great detail. We have computational approaches that can predict the amount of computational resources that will be required to execute a program. By tallying the costs of individual instructions, one can estimate the overall cost of executing a program on a given platform. Analogously, individual musical components also have agreed upon costs, defined in terms of the difficulty they present to performers. By decomposing a music score into its individual musical components, one can use their unit costs to compute the total complexity of executing a score on a given instrument, as shown in Figure 1.1.

Figure 1.1: The process of decomposing code and music to extrapolate a conclusion.



Although one can draw various analogies between music and computing, the intricacy of determining the complexity of music scores is most similar to estimating program performance on different computing devices. While the same compiled program can be executed on any computing device of a given architecture, the device's processing power ultimately determines the efficiency of the resulting execution, which can vary widely. The same applies to the complexity experienced by musicians with dissimilar levels of proficiency when performing the same piece on a given instrument. Although all musicians read a piece of sheet music and understand it similarly, the complexity of playing that piece is determined by a performer's proficiency. Musiplectics aspires to blaze a trail toward objectively assessing these complexities by creating a practical computational framework that can capture the subtle nuances of musical complexity.

The solutions presented herein are instrument agnostic. Nevertheless, to realize the concept of musiplectics, our reference implementation of the computational framework targets the

B♭ Clarinet. This exclusive focus on clarinet is reflective of our own music performance expertise, rather than of any limitations of the presented concepts.

1.1 Research Contributions

By presenting our work, this paper makes the following contributions:

1. Musiplectics, a new application of computing that encompasses automated assessment of the complexity of music scores.
2. A computational model for music complexity; it assigns base levels of difficulty to an instrument's notes and intervals, with dynamics, articulations, key signatures, and note duration serving as multipliers that increase or reduce the base difficulty.
3. Musical Complexity Scoring or MCS, a concrete realization of the musiplectics-based model above, made publicly available online for teaching and experimentation.
4. A preliminary evaluation of our model's realization, which compares MCS to commonly accepted educational level guidelines for repertoire pieces.
5. An evaluation of music Optical Character Recognition (OCR) software as a means of converting sheet music into a computer-readable MusicXML format.

1.2 Paper Roadmap

The remainder of this paper is structured as follows. First, Chapter 2 presents background information on music to provide a base line for readers. Then, Chapter 3 highlights related work in this field. Next, Chapter 4 explains our basic model for determining music complexity. Afterwards, Chapter 5 details the proof of concept's design and details. This leads into the evaluation chapter 6 which shows how we have tested this system and what preliminary results we have uncovered. Chapter 7 interprets the results and discusses how our system compares to the state of the practice. Afterwards, Chapter 8 gives directions for future work. Finally, Chapter 9 presents our conclusions for this project.

Chapter 2

Background

This research is concerned with evaluating musical complexity. Music is a very specialized domain with its own unique set of concepts, terms, and conventions. Hence, for the benefit of the reader not familiar with Western music, we next present a brief refresher of the standard elements of music. The reader well-versed in this art can safely skip this Section. Although, we have striven to adhere to the standard definitions of the common musical elements, this Section distills music theory down to the core concepts for the sake of brevity.

The fundamental building block of music is the note. Notes in music are similar to words in spoken language or tokens in code. The characteristics expressed by a sound require multiple types of visual symbols to express a note. In written music, the most important characteristics of notes are pitch, duration, dynamics, and articulation. In sheet music, notes are depicted as ovals, both with and without a line (stem) and flag attached to them.

Notes are placed into measures. A measure is a uniform unit of time that breaks up a music piece into smaller segments. These smaller segments endow the piece with a repeating rhythmic emphasis, analogous to meter in poetry.

Pitch is the most salient characteristic of any note. It is the frequency of the sound wave the note produces, or how high or low a note sounds to the listener. Pitches are not continuous, but rather fall on specific discrete steps or half steps within the range of audible frequencies.

An interval is the difference in pitch between two notes. Each interval's name encodes the distance between pitches with different levels of gradation. For example, a minor third, a perfect fourth, a major sixth, etc.

The pitch of a note can be determined from the note's vertical placement on the staff (the five horizontal lines and four spaces in between), the clef (the range of notes possible to represent on the staff), and the key signature (the pitches to be altered up or down a half step). Accidentals can also similarly alter the pitch of a note up or down by half steps; but accidentals are not applied to the entire piece, only one specific measure at a time. Both accidentals and key signatures are represented by symbols called sharps or $\sharp$, which raise the pitch a half step, flats or $\flat$, which lower the pitch a half step, and naturals or $\natural$, which undo the effect of the previously applied pitch modification. Web designers can draw an analogy from key signatures and accidentals in music to global and inline cascading style sheets (CSS) in web page rendering.

Another central characteristic of a note is its duration. Duration is simply the length of time a note is sustained. Although written music does not explicitly specify duration, it is inferred from the note's value (a fraction expressed by the note's stem and flags), the time signature (how many beats are in a measure and what fraction of a whole note gets a beat), and the tempo (how many beats are played in a given time frame).

For example, the value of a quarter note is $1/4^{th}$ that of a whole note. In three four time, the first number says there are three beats in a measure, and the second number says that a quarter note receives one beat in a measure. If the tempo is 120 bpm (beats per minute), then this example measure would last exactly 1.5 seconds (3 beats in a measure / (120 beats per minute / 60 seconds/minute)).

Dynamics refers to the volume of notes. Dynamics are specified with different Italian words, such as piano (quiet), forte (loud), and mezzo (medium), expressed by letter **p**, **f**, and **m**, respectively. Composer combine these letters to express a wide variety of dynamic levels, typically ranging from **ppp** (pianississimo, meaning very, very quiet) to **fff** (fortississimo, meaning very, very loud). There are also markings for gradually changing dynamics to louder (crescendo) or softer (decrescendo or alternatively diminuendo) that look like elongated greater than or less than symbols below the staff.

Articulation is how a note is played and linked to subsequent notes. The simplest analogy for articulations are how different letters or sounds are formed and connected in spoken language. For example, speaking "ta" and "la" have the same "a" sound once held out, but their initial articulations are different because the "t" and "l" sounds are uttered differently. There are many different articulations, but the main ones utilized in this work are as follows:

- Accent or $>$, which means to play the note louder than those around it.
- Staccato or $\cdot$, which means to play the note shorter than its full value, cutting it off early.
- Tenuto or $-$, which means to play the note slightly longer than its full value, in a connected manner to the following note.
- Marcato (a strong accent) or $\wedge$, which means to play the note much louder than those around it, more than an accent.
- Slur or $\smile$, which means to separate only the first note, connecting all the following notes together.

Although musical notation possesses additional advanced characteristics, including timbre, further articulations, and elaborate music markings, we did not find these advanced symbols as common enough to be useful to consider in a general complexity model. When writing music, composers combine the concepts above to express the desired artistic impression the composition is intended to make on the listener. Hence, it is hard to distinguish which notational concept is more important than the other for artistic expression. Nevertheless, we have found the subset presented above as absolutely essential to realizing the ideas behind musiplectics.

Chapter 3

Related Work

Multiple prior research efforts study various facets of music. Although relatively few works concretely focus on analyzing music complexity 3.1, tangential studies into scanning and searching music 3.2 as well as classifying music 3.3 represent related research efforts. Each has tangible ties to this work's objectives, albeit some at a more abstract level than others.

3.1 Complexity Analysis

The most relevant works to musiplectics are those that also seek to analyze the complexity of music in some way. A representative example is presented in reference Chiu and Chen [2012]. This work seeks to automatically generate or predict the difficulty of a piano music piece. The authors apply regression theory over a set of features that cause difficulties for that specific instrument. Musiplectics leverages similar musical concepts to build up an aggregate measure of difficulty or complexity. However, our approach has a wider range of applicability, both in terms of musical instruments and playing proficiency, as well as portability, by embracing uniform types of complexity parameters. In other words complexity parameters in musiplectics encompass the cognitive and mechanical complexities for a given instrument. So all instruments have the same types of complexity parameters, which may take upon vastly different values.

Similar to Chiu and Chen [2012], Heijink and Meulenbroek [2002] studies the complexity of playing guitar. Their work extracts features that determine difficulty when playing guitar. However, their focus lies more on the mechanical difficulties specifically associated with that instrument rather than broad ranging complexities of playing music on any instrument. Musiplectics also takes into account the playing proficiency of the player at hand.

Another related approach to ours is Liou et al. [2010]. The authors analyze the complexity of the rhythmic components of various pieces of music. They utilize the L-system to breakdown the rhythm of a piece into a tree structure and then apply tree analysis algorithms to generate a score. Although musiplectics avoids complex algorithms for examining the rhythmic structure of a piece, it considers a full array of the elements of music scores rather than rhythm alone.

State organizations, such as Virginia Band and Orchestra Directors Association [2015] in Virginia and NYSSMA New York State School Music Association [2015] in New York as well as others, similarly analyze music scores by hand, an activity which we hope to automate. These organizations govern K-12 schools in their respective states. They also list music pieces and their respective difficulty grades for district, regional, or state competitions for each K-12 grade level. Other organizations, such as the Royal Conservatory Music Development Program Royal Conservatory Music Development Program [2014], offer similar pieces and respective grades as part of their level requirements and assessment regulations. Unlike state organizations, Royal Conservatory publishes their pieces and scores to the public, a provision that enables us to leverage them in evaluating our work.

The difficulty grading schemes in both types of organizations are analogous to the complexity of the piece, except that in these organizations pieces are graded subjectively by a group of people rather than a uniform algorithm. Additionally, the grades are typically listed as integer values between 1 and 10, thus lacking a sufficient level of granularity to express nuanced differences between musical pieces.

3.2 Music Scan & Search

One area of research tangential to analyzing music complexity is scanning and searching for music. The overlap lies chiefly in the end use cases in both areas of research. Educators, performers, and other stakeholders, all find themselves needing to efficiently locate musical pieces that meet their respective requirements. Providing a complexity score is one means to improve the efficiency of searching for new music, since users can see complexity at a glance or even search by complexity to find a piece to prepare. Byrd [2001] gives many reasons for why this is necessary, but there is a whole body of research related to music information retrieval that deals with similar problems.

Another area of overlap between musiplectics and music scanning is in translation. From a high-level, reading in any form of music and writing out a related, different form is essentially translation. There are many research efforts to translate forms of music into other languages or versions. An especially interesting effort in this regard is Allali et al. [2009]. The authors work to convert polyphonic music (music with several simultaneous notes on one or several instruments, such as a band playing together in harmony or one person playing piano or guitar for instance) into equivalent monophonic music. Their end goal is to reduce a large, expressive format to a more simple one for comparing pieces during a lookup. The reduction in complexity of chords down to single notes represents an interesting approach that could be leveraged with musiplectics, so as to generate a potentially less complex version of a given piece.

3.3 Music Classification

Music classification presents another area of potential research overlap. Much research has previously dealt with using computers to understand music and thus classify into various genres. While the efforts of music classification are not the same as determining complexity, the approaches taken to classify certain pieces via machine learning and statistical methods are important, because they present means of automatically analyzing music. At some point musiplectics could potentially apply similar machine learning concepts to interpret what makes music complex (rather than decomposing music scores into individual elements and calculating their summary complexity) as well as leverage genre classification as another source of potential complexity.

Cuthbert, Friza, and Friedland Cuthbert et al. [2011] for example focused heavily on using machine learning to classify different types of music. The authors can extract multiple features from a variety of input types and apply their theorem to successfully classify the genre of several inputs.

Similarly, Cataltepe et al. [2007] proposes an approach to classifying music files in MIDI format specifically. The authors form an approximation of the Kolmogorov distance using the normalized compression distance between approximate string representations. They use this approximation as the main feature to classify numerous music files.

Chapter 4

Computational Model for Music Complexity

In Chapter 2 above, we discussed several fundamental music concepts. These concepts serve as the baseline elements that factor into the complexity score.

The most straightforward approach to calculating an overall score is to assign whole number weights to each element perceived to be especially important (i.e., notes, intervals, dynamics, articulations, key signatures, and note durations) and add all the weights up. This scheme, however viable, fails to adequately reflect the experience of playing music. At their core, notes and intervals present distinct difficulties on their own, whereas dynamics, articulations, key signatures, and note durations only modify those difficulties. For instance, a small interval may seem easy on its own, but with changing dynamics, with differing articulations, in a strange key, or at a high tempo, that interval could become much more difficult.

Hence, a more authentic approach to calculating an overall score is to only assign whole numbers weights to notes and intervals. These are still counted and summed up into a final score. However, dynamics, articulations, key signatures, and note durations become multipliers onto notes and intervals. Each dynamic, articulation, and key signature thus receives a multiplier weight that is a decimal (typically between 0 and 2). Those multipliers are applied to every occurrence of a note or interval.

Note duration is factored into the score as an average over all notes. The total amount of notes and associated duration in seconds is calculated at the end and applied as its own multiplier. The more notes in a given span of time, the higher the multiplier becomes.

This scheme captures the concept we envisioned that makes duration complex, except in the extreme case of playing excessively long notes. In cases of wind instruments, such as our main target of B♭ Clarinet, holding notes out for long durations may possess its own difficulty in providing adequate breath support, rather than the difficulties of changing finger positions and embouchure quickly.

Thus, our model adapts the note duration multiplier to be a multiplicative or fractional difference from one. If the average of notes per second in the piece is 1.5, then the multiplier remains 1.5. However, in the case of many long notes, the average of notes per second might be something closer to 0.5. In this case (when the average is less than 1), the fraction becomes its reciprocal, 2 in the example.

Users cannot directly change this parameter as it is built into the model. However, they can still influence the degree to which this parameter affects the overall score. To that end, the user can parameterize note durations with a multiplier value that increases or reduces the impact of the note duration parameter. For the cases when the user is content with the built-in setting, the model applies the default value of 1.

Chapter 5

Proof of Concept

In 5.1, we outline the basic software design of our proof of concept and its extensible architecture. Then, we describe the implementation choices we have made while realizing our design in 5.2.

5.1 Design Overview

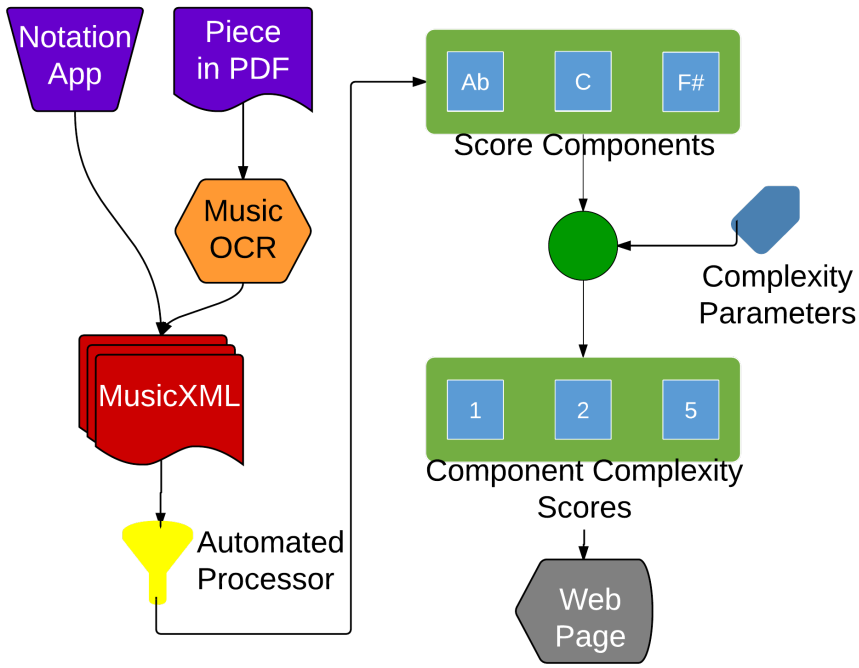
The complexity model presented above outlines the basic functionality of our approach. Nevertheless, the implementation's complete control flow involves several steps. The complete control flow can be seen in Figure 5.1.

From the top left, one can see the inputs to our a musiplectics system are Notation Apps and Pieces in PDF format. Many common notation applications, such as Finale Notepad Makemusic Inc. [2015a] and Sibelius Avid Technology Inc. [2015], support a universal format called MusicXML Makemusic Inc. [2015b] Good et al. [2001]. MusicXML files can be imported, edited, and output back to MusicXML or other proprietary formats. MusicXML is the underlying representation on which our reference implementation operates.

Alternatively, the system can be extended to work directly with music pieces in PDF format. It can accept PDF files and transform them into their MusicXML representation, by means of music OCR (Optical Character Recognition) software. Our reference implementation relies on free software from Audiveris Bitteur [2013] currently, but it could equivalently utilize other off-the-shelf OCR applications.

Once the MusicXML representation of the piece is obtained, the automated processor then takes it as an input. The processor computes the complexity score by going through a sequence of steps. First, it parses the piece to extract individual music elements to be analyzed. Then, it looks up the weights for each element as specified by the complexity parameterization.

Figure 5.1: The overall control flow from a user's perspective.



Recall that our system can be parameterized with different complexity weights to reflect dissimilar levels of per-instrument playing proficiency. Currently, these weights are specified in an xml file to facilitate tool integration. Because the weights are meant to be decided on by music experts, our design foresees the creation of visual editors to specify this information. These editors can easily save their inputs in some xml format. In our proof of concept, we experimented with 5 different sets of weights, which represent proficiency levels that range from that of a beginner up to a trained professional.

Using the specified weights, the system calculates the complexity by tallying up the difficulty values of each individual type of element, computed separately during a single pass over the structure. The final piece of functionality presents the calculated complexity in a user friendly fashion. To that end the system uses a web-based interface with specialized javascript libraries.

One key design feature is ease of extendability at each level. Any application that generates valid MusicXML files can feasibly provide input to our system. Furthermore, the overall process can be extended if other applications can generate the piece of music in PDF form or some intermediate that can be represented in PDF or MusicXML.

The system itself can be extended to operate only on specific selections of pieces or on batches of multiple pieces if necessary (thus showing the complexity of an entire performance). Finally, the parameters that determine complexity can easily be changed on the fly. This provision enables individual performers or groups to set their own complexity levels to find a score even more relevant to their playing style.

5.2 Implementation Details

The reference implementation spans several different code bases, most notably the automated processor backend and the web UI frontend. Both are publicly available on GitHub Holder [2015] along with instructions as to how to set them up and links to the current deployed version.[1]

The backend code is written in Java with some limited PHP to facilitate server communication. We've utilized the Apache Xerces 2.11.0 library for parsing xml files Apache Xerces [2013] and the JSON Simple 1.1.1 library JSON Simple [2012] for constructing JSON to pass along to the frontend. The backend utilizes the visitor pattern heavily so as to ease the handling of the intricacies and redundancies of processing MusicXML.

The frontend code is written in Javascript, HTML, and CSS. To minimize the amount of hand-written code, our implementation makes use of numerous libraries, including jQuery 2.1.0 The jQuery Foundation [2015], Bootstrap 3.3.4 Otto and Thornton [2015], Datafs 1.10.5 SpryMedia Ltd. [2015], D3 3.5.5 Bostock [2013], and VexFlow 1.2.27 Cheppudira [2010]. These libraries greatly facilitate various standard facets of system implementation, making it possible for us to focus on the novel, research-related issues of musiplectics.

[1] The current version is deployed at `http://mickey.cs.vt.edu/`

Chapter 6

Evaluation

The goal of our evaluation is to demonstrate that the reference implementation of musiplectics can be a useful tool for music educators. To that end, we ran our system with different parameterizations on a set of music scores, used in educational settings. Music educators have also ranked these same scores by hand, producing a baseline for comparison. As expected, even though the general trends reported by our tool corresponds to that provided by music educators, we have also observed some outliers. Some pieces turned out to have been much more complex on the relative scale than the pieces immediately preceding and following them in the rankings. These discrepancies can be explained either by immaturities of our implementation or by inaccuracies of ranking of music scores by hand. It is easy to overlook some really complex parts in the middle of a score when trying to assess its suitability for a given playing proficiency. While an automated tool analyzes scores in their entirety, producing the results based on an exhaustive analysis of each and every element.

We first specify the complexity parameters used in the experiments in 6.1. Then, we unveil the strategies one can follow to obtain the settings that represent a consensus among musicians in 6.2. Next, we describe the test pieces selected in 6.3. Finally, we discuss the accuracy of the OCR program tested for suitability in musiplectics in 6.4.

6.1 Clarinet Complexity Parameterization

Chapter 4 above explains the rationale behind complexity parameters. In particular, it differentiates between parameters, expressed as whole number weights and those which are decimal number multipliers. In that presentation, we did not link the parameters to any particular instrument. By contrast, in this Section, we discuss their specific application for a specific instrument, the B♭ Clarinet.

We decided to focus specifically on the clarinet, because it exhibits many forms of complexity

Table 6.1: Note ranges and weights for beginner B♭ Clarinet.

Note Range	Weight
G3-G♯4	1
A4	2
B♭4-C5	5
$\geq$ C♯5	10

Table 6.2: Intervals, note ranges, and weights for beginner B♭ Clarinet.

Interval	Low Range	High Range	Weight
Unison	Anywhere	Anywhere	1
Second	G3-G♯4	G3-G♯4	2
Third	G3-G♯4	G3-G♯4	3
Fourth	G3-G♯4	G3-G♯4	4
Fifth	G3-G♯4	G3-G♯4	5
Any	G3-G♯4	A4-C5	8
Any	A4-C5	$\geq$ C♯5	10
Sixth	Anywhere	Anywhere	9
Seventh	Anywhere	Anywhere	9
Octave	Anywhere	Anywhere	9
$>$ Octave	Anywhere	Anywhere	10

and relates to many similar woodwind and brass instruments. Additionally, our own musical expertise favors this instrument above all others, making it possible for us to define our own complexity parameters with confidence and not requiring external confirmation.

The initial complexity settings for B♭ Clarinet were for beginners. Based on these values, complexity settings for other levels of B♭ Clarinet were later adapted, but those levels largely changed only the associated weights. Because we utilized only the beginner settings in our tests, they are the only ones discussed below.

Note complexities were broken up into the ranges and assigned weights, as shown in Table 6.1. Intervals were similarly broken up further and assigned weights, as shown in Table 6.2.

Dynamics and articulations were specified for those specific types mentioned previously in Chapter 2. Their weights can be found in Tables 6.3 and 6.4, respectively. Unlike dynamics and articulations, all possible major key signatures are specified with weights and shown in Table 6.5. Finally, the note duration modifier is kept at 1.0, so the note duration modifier works exactly as specified in 4.

Table 6.3: Dynamics and weights for beginner B♭ Clarinet.

Dynamic	Abbreviation	Weight
mezzo forte	mf	1.0
mezzo piano	mp	1.0
forte	f	1.1
fortissimo	ff	1.2
piano	p	1.3
pianissimo	pp	1.5

Table 6.4: Articulations and weights for beginner B♭ Clarinet.

Articulation	Weight
Slur	0.5
Normal/None	1.0
Accent	1.1
Staccato	1.2
Tenuto	1.2
Marcato (Strong Accent)	1.4

Table 6.5: The major key signatures and weights for beginner B♭ Clarinet.

Key Signature	Sharps/Flats	Weight
C	None	1.0
G	F♯	1.1
D	F♯, C♯	1.1
A	F♯, C♯, G♯	1.2
E	F♯, C♯, G♯, D♯	1.3
B	F♯, C♯, G♯, D♯, A♯	1.4
F♯	F♯, C♯, G♯, D♯, A♯, E♯	1.5
C♯	F♯, C♯, G♯, D♯, A♯, E♯, B♯	1.6
F	B♭	1.1
B♭	B♭, E♭	1.1
E♭	B♭, E♭, A♭	1.2
A♭	B♭, E♭, A♭, D♭	1.3
D♭	B♭, E♭, A♭, D♭, G♭	1.4
G♭	B♭, E♭, A♭, D♭, G♭, C♭	1.5
C♭	B♭, E♭, A♭, D♭, G♭, C♭, F♭	1.6

6.2 External Survey

Although the complexity parameterization presented in 6.1 may be accurate, one would not be able to validate them empirically, as they reflect one's subjective personal experiences and beliefs. However, musiplectics this subjectivity, enabling individual musicians to specify the parameterizations that reflect their own individual understanding of their own or their students' proficiency.

As a logical consequence of the previous observation, it would be equally impossible to empirically validate the "correctness" of the computed complexity score of an analyzed piece. However, if stakeholders in a score can generally agree on its relative complexity, the resulting consensus can serve as a viable form of validation.

Based on this assumption, experts, musicians, and educators seem to have a vested stake in the results of these complexity scores. Therefore, we've begun to survey those related to B♭ Clarinet in an effort to ascertain their opinions. In the survey we ask simple questions about the complexity parameters already established, both how the parameters are implemented and the weights assigned to each. Once a statistically significant consensus has been reached or some threshold of responses have been given, the results of the survey will become the new complexity parameters. At the time of writing, neither condition has been met so our own parameters are in use, but it is important to note that we are striving to find an amicable means of determining these parameters.

6.3 Graded Music Pieces for Comparison

Based on the 2014 syllabus for B♭ Clarinet available from Royal Conservatory Royal Conservatory Music Development Program [2014], we selected 2-4 pieces for each grade, 1-10. The pieces were chosen based on availability, so as to minimize the amount of companion book or subscription purchases required. In whole 32 pieces are used for the main comparison: 10 from Standard of Excellence Pearson [1993], 7 from Clarinet Solos Arnold [1939], 4 from Concert and Contest Collection Voxman [1992], and 11 publicly available on IMSLP Project Petrucci LLC [2015]. Each of these is listed with its author and grade in Table 6.6. These pieces are translated into MusicXML using the OCR process and subsequently passed through MCS to obtain a complexity score for each.

Figure 6.1 displays the complexity score of each piece by associated grade. The pieces are in the same order as previously listed in Table 6.6, but are displayed here by grade for readability. Please note that pieces in all grades do have a complexity score, but those in grades 1-3 are less than 1000 and are not discernible in the graphic. Similarly, Figure 6.2 shows the average complexity of pieces by grade from Royal Conservatory. Again, scores for

Table 6.6: The works from Royal Conservatory chosen for comparison along with their grade and composer (or book reference if no composer information was available).

Gr.	Title	Composer
1	Bingo	S.o.E.
1	Eerie Canal Capers	S.o.E.
1	Go for Excellence no. 61	S.o.E.
2	Alouette	S.o.E.
2	Grandfather's Whiskers	S.o.E.
2	Ming Court	S.o.E.
3	Just Fine	S.o.E.
3	Variations on a Theme	Mozart
3	Loch Lomond	S.o.E.
3	Theme from Symphony 9	Beethoven
4	Minuet in G	Beethoven
4	Gavotte	Gossec
4	Song without Words	Tschaikowsky
5	Humoresque	Dvorak
5	The Dancing Doll	Poldini
5	Hymn to the Sun	Korsakoff
6	Serenade	Drdla
6	Promenade	Delmas
6	Scherzo	Koepke
6	Nocturne	Bassi
7	Sonata Mvmt. 2	Hindemith
7	Scene and Air	Bergsen
8	Canzonetta	Pierné
8	Concerto Opus 36 Mvmt. 1	Krommer
8	Sonata Mvmt. 1	Saint-Saëns
9	Sonata Mvmts. 3 and 4	Hindemith
9	Sonata Mvmts. 2, 3, and 4	Saint-Saëns
9	Solo de Concours	Rabaud
10	Concerto no. 3 Mvmts. 1 and 2	Crussell
10	Concerto no. 3 Mvmts. 2 and 3	Crussell
10	Solo de Concours	Messager
10	Sonata no. 2 Mvmt. 1	Stanford

pieces in grades 1-3 are present, but are barely visible due to scale.[1]

[1] For more information, the full set of data can be found on our GitHub Holder [2015] under Documents/-Data/.

Figure 6.1: The complexity score of pieces by grade from Royal Conservatory. Everything to the right of a grade including that number represents a piece with that grade.

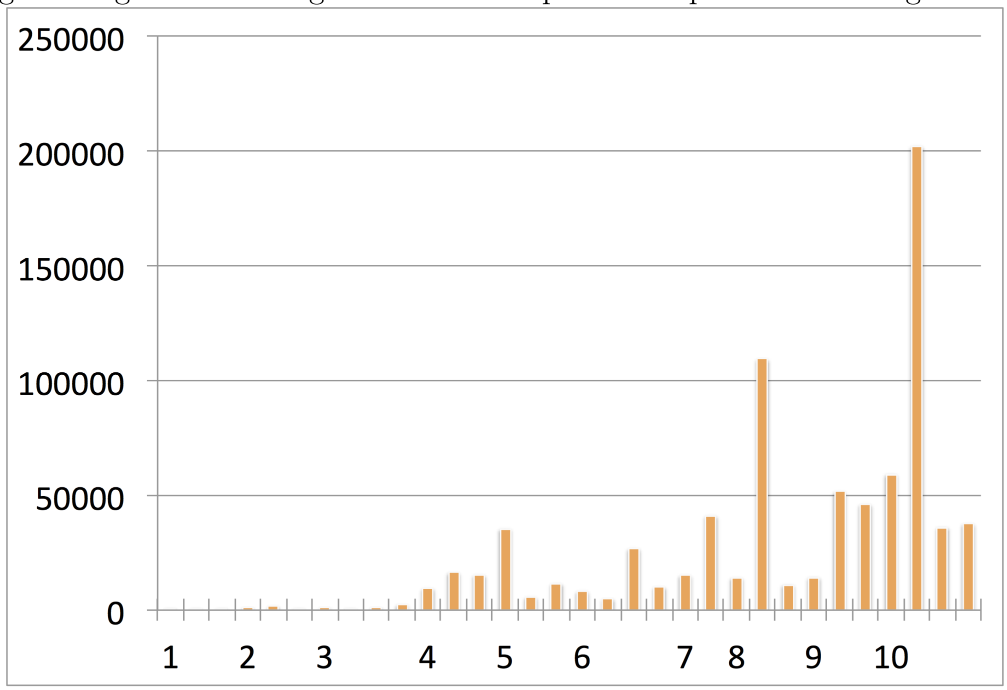
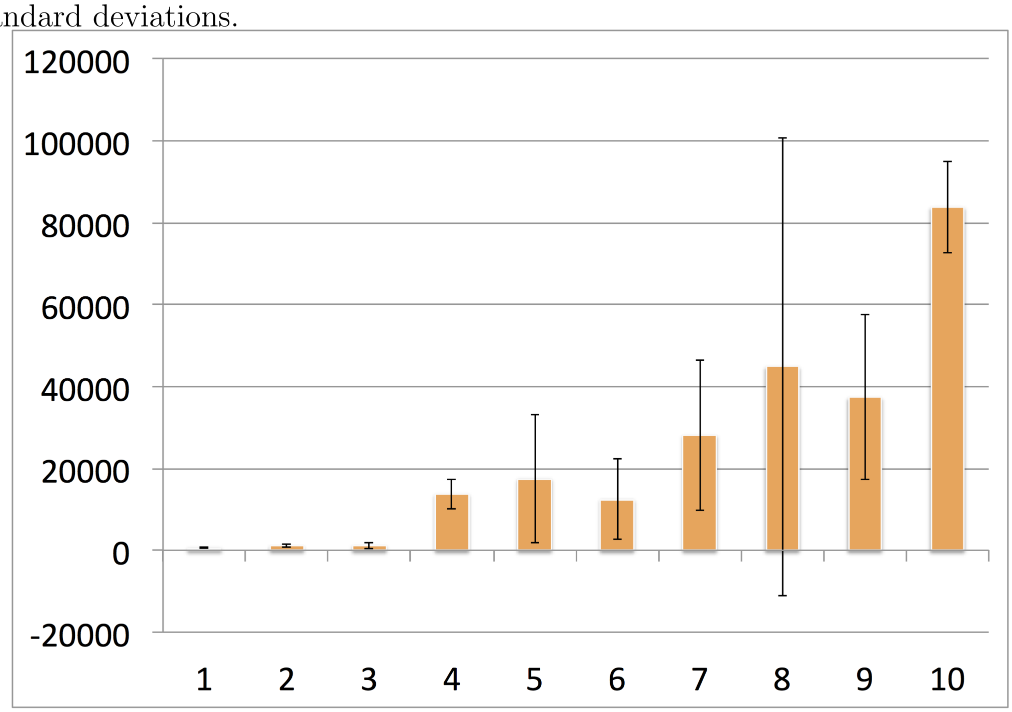


Figure 6.2: The average complexity score of pieces by grade from Royal Conservatory along with standard deviations.



6.4 Optical Character Recognition (OCR)

The ability to use music scores in PDF format would greatly enhance the usability of our reference implementation. However, this ability hinges on the accuracy of the music OCR software that can translate PDF into MusicXML. Hence, we empirically evaluated the accuracy of a widely used music OCR application to assess its suitability for our system.

In our experiments, we used freely available music OCR software from Audiveris Bitteur [2013]. We evaluated the reliability of this process as follows: find pre-matched PDF files with the correct MusicXML, convert the PDF files into a new MusicXML, and finally compare the correct and converted MusicXML to each other.

The pre-matched PDF and MusicXML files we used are all from MuseScore MuseScore BVBA [2015] and are listed in Table 6.7. They were selected randomly from the list of single part pieces for clarinet. The MusicXML files came in .mxl (compressed MusicXML) format from MuseScore, and they were uniformly imported into Finale Notepad 2012 Makemusic Inc. [2015a] to then export the uncompressed format for comparison. For this test we use only Audiveris for conversion.

Table 6.7: The works with matched PDF and MusicXML files from MuseScore chosen for testing OCR reliability along with their composer or arranger.

Title	Comp./Arr.
Dancing Clarinet	KRM
Mi Razon De Ser	Banda Central
Rudy	Jerry Goldsmith
The Hobbit: The Desolation of Smaug	Howard Shore
The Rose	Dan White

Figure 6.3: The comparison command for MusicXML files.

```
sdiff -B -b -s Original.xml OCR.xml | wc
```

We make use of a file differencing tool with counts of words, lines, and characters as shown in Figure 6.3, as well as a comparison of complexity scores to highlight the potential effects of the OCR process. We show the difference measurements for select pieces in Figure 6.4. We also present the average across pieces for each difference in the same figure. Figure 6.5 shows the difference expressed as the percentage of change from the values of the original MusicXML file. It also presents the average across pieces in the same figure.

Figure 6.4: The difference between matched and OCR generated MusicXML files in words, lines, and characters via sdiff as well as the positive difference in complexity score.

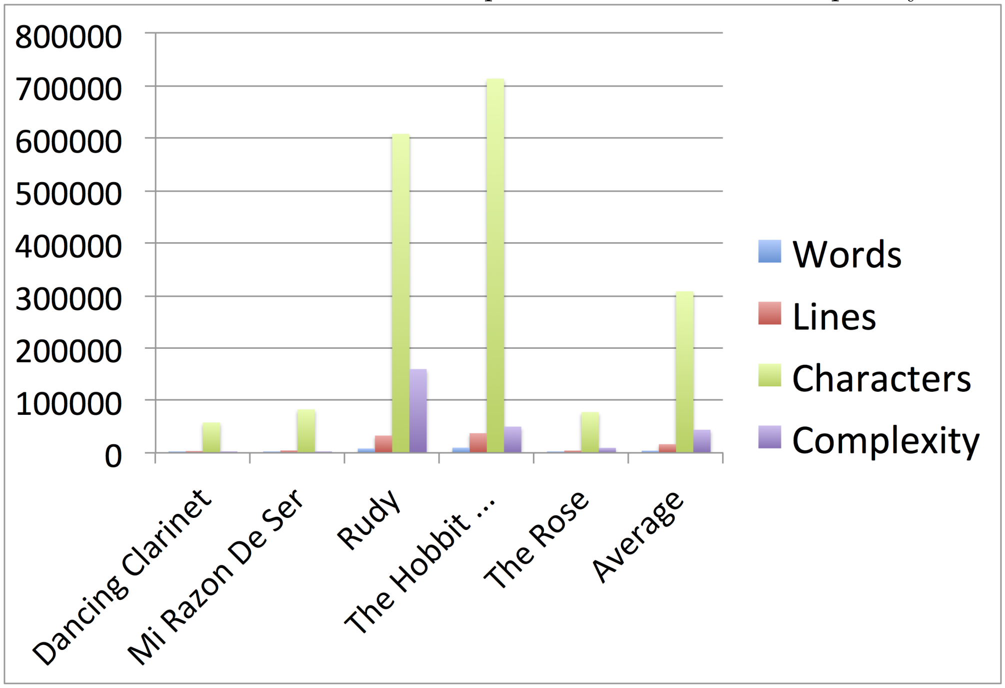
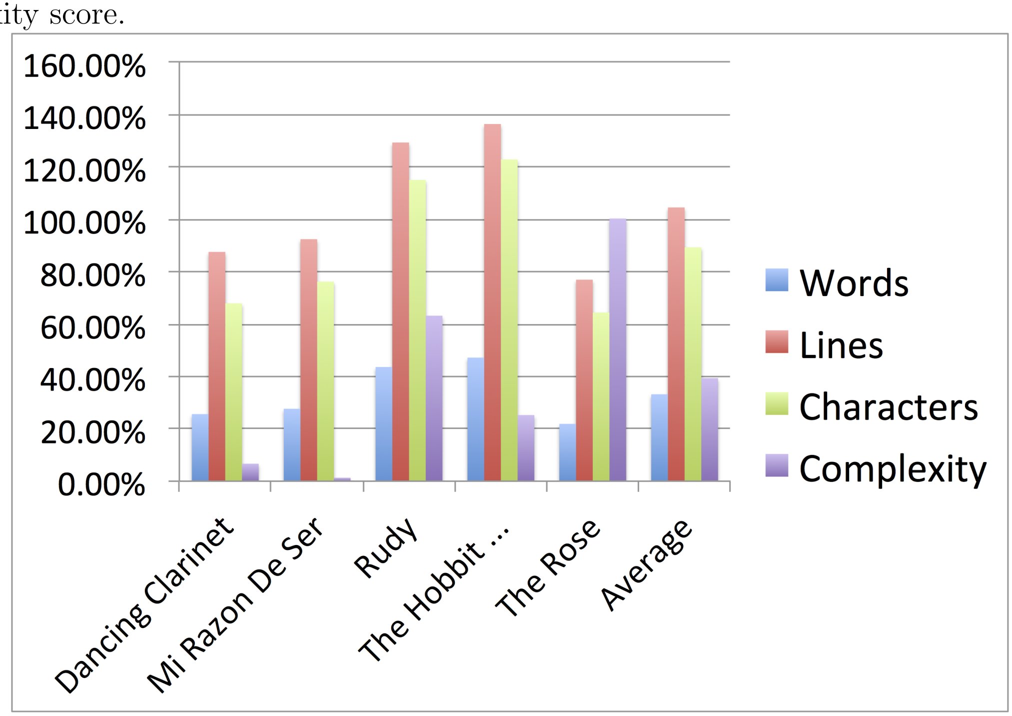


Figure 6.5: The percentage difference between matched and OCR generated MusicXML files in words, lines, and characters via sdiff as well as the positive percentage difference in complexity score.



Chapter 7

Discussion

We discuss our findings as follows: Section 7.1 covers our results from assessing previously graded pieces from Royal Conservatory; Section 7.2 talks about our results from comparing matched and OCR generated MusicXML files; and Section 7.3 discusses our findings with regards to the website's usability.

7.1 Manually Graded Pieces and Their Calculated Complexity Scores

Figures 6.1 presents the results of applying our reference implementation on 32 scores, which have been manually ranked by music educators as belonging to levels between 1 and 10. One would naturally expect the lowest and highest complexity scores to come from grade 1 and grade 10 pieces, respectively. Although the lowest score piece was a higher grade than expected, the second lowest score was indeed for a grade 1 piece, "Bingo", and the highest score was for a grade 10 piece, "Concerto no. 3," 2nd and 3rd Movements by Crussell. This outcome shows in one respect, that our results generally correlate with the expectation. Other discrepancies and outliers are to be expected given the subjective nature of grading pieces.

The graph reveals that even though the automatically calculated complexity scores follow the overall trend set by their manual rankings, there is a lot of noise in the calculated scores. This noise reflects the discrepancies between the manual (baseline) and automated (evaluated) rankings. Some outliers are worth examining in detail. In particular, the lowest computed complexity score of 514.19 was for "Variations on a Theme" by Mozart, a grade 3 piece, while the highest one of 202214.85 was for "Concerto no. 3," 2nd and 3rd Movements by Crussell, a grade 10 piece. Musicians would argue that music by Mozart is known to be deceptively simple. Hence, the low mechanical difficulty calculated by our tool may not truly represent how human experts see this piece by Mozart, which is a well-understood exception in classical music. The Concerto looks deceptively hard due to the presence of a free-form cadenza that uses the very unusual 17/4 time signature and numerous consecutive tuplets, including both common 16th and 32nd note groups as well as unusual 6-tuplets. It is possible that performers would find this piece much less daunting once they make sense of these rhythms.

To further illustrate our correlation with the expectation, Figure 6.2 shows the average complexity score of pieces by grade along with the standard deviation. Here, the complexity scores seem to more closely match what one would expect. The average complexity scores roughly increase and are within one standard deviation of consistently increasing. There are still outliers present, including grade 6 pieces having an average complexity less than grade 5 or 4 pieces, and grade 8 pieces having larger average complexity than those of grade 9. However, all of this evidence is simply a testament to the highly subjective nature of complexity assessment. It is likely that experts presented with our automated results would consider revising their ranking recommendations.

7.2 Matched and OCR Generated MusicXML

Another possible explanation for the discrepancies highlighted between graded pieces could be the inaccuracies of our toolchain, specifically music OCR. During our tests, we largely utilized Audiveris Bitteur [2013] for its simplicity and speed in generating MusicXML as well as its plethora of options for input image formats. However, some files were simply too poor of image quality for it to initially accept. With some manual effort to change image formats and attempts to improve the resolution of scanned images, Audiveris was finally able to catch the remainder of cases.

The process of converting PDF's to MusicXML is admittedly an imperfect means of generating accurate MusicXML representations. As Figure 6.4 shows, there were large differences between our test bed of matched and OCR generated MusicXML files. The difference in characters is the most alarming, however that could be explained by the limited features OCR is able to analyze with respect to the entire set encoded into the matched MusicXML. Nevertheless, the end result of the file differences leads to the associated differences in complexity scores.

Figure 6.5 underscores this difference by showing it as a percentage of the original measure of words, lines, characters, and complexity. While the piece "Dancing Clarinet" has only a 6.68% difference from the original complexity score, the piece "The Rose" has over 100% difference from the original complexity score. Both of these show over 20% difference in words and over 60% difference in lines and characters. Interestingly, the pieces "Rudy" and "The Hobbit: The Desolation of Smaug" both show over 110% difference in lines and characters, yet the change in underlying MusicXML only causes about 63% and 25% difference in complexity for these pieces, respectively.

Therefore, it would seem that the process of using OCR to generate MusicXML from PDF files of music scores is simply not yet mature enough to handle the demands of complex music scores. This conclusion is strengthened by both outside research Byrd and Schindele [2006] and our experience with acquiring such software. Our original intention was to deploy such OCR software at the beginning of our control flow from a web UI to allow users to easily upload PDF files without performing a manual conversion on their own. However, of the 5 different OCR packages we experimented with and nearly purchased, most did not even offer an option for batch execution. Those that did offer this option could not operate in this mode consistently or with a measure of reliability.

While this process is obviously imperfect, its speed and automation allow us much more flexibility in generating MusicXML even outside of batch mode. This process is still absolutely necessary for comparing complexity scores of well-known works, since so many have not previously been rewritten into MusicXML. At this point there is no clear substitute for music OCR, but it is our hope that future efforts will strive to improve both the accuracy and reliability of this process so more research can be performed with sheet music.

7.3 Website Usability

We do not provide any empirical data so far on the performance of our deployed implementation, but it is publicly available for general use. In our informal discussions with potential users, we uncovered many points of design that we discuss here.

First, the website must provide access to the matched PDF file as well as the complexity score from the MusicXML representation. The PDF file is made available so that there is no confusion about exactly what the piece of music is that is generating the score.

Second, we found that many users especially wanted to see what the most complex measure is in the piece as a means of determining whether the complexity score was due to one very difficult spot or a collection of many less difficult areas. Thus, we determine the complexity score for each individual measure in the pieces available. We not only show the measure number and associated score however, we also utilize VexFlow Cheppudira [2010] to graphically represent this measure to further accommodate users.

Finally, the website also features the ability to run on music pieces with multiple parts or instruments. The relevant data for each part or instrument is extracted and even graphed against one another through D3 Bostock [2013]. This comparison is not necessarily correct, given that each part or instrument is assigned a complexity score based off of parameters for B♭ Clarinet. Yet, it is still a valuable design point to show the general applicability of this approach to works that are not only for instruments besides B♭ Clarinet, but also for entire ensemble or orchestral pieces.

Chapter 8

Future Work

The following Subsections address planned future work in a number of different directions, including expanding complexity parameters in 8.1, mapping parts and instruments in 8.2, understanding complexity scores in 8.3, integrating scores with sources in 8.4, including more input formats in 8.5, and measuring complexity from physiology in 8.6.

8.1 Expanding Instrument Complexity Parameters

As mentioned in 6.2, the current complexity parameters lack validation. One way to improve their accuracy is thus to gather a consensus from those with a stake in this complexity measurement, such as experts, performers, and educators. We have already begun the process of surveying these people for their opinions on complexity parameters for B♭ Clarinet. However, a notable direction for future work is to expand this survey and indeed the viability of the overall complexity score out to other instruments.

Our reference implementation in its current state can run effectively for any instrument, and any complexity parameters can be utilized. In this way it is currently agnostic to what instruments are being played in the piece. We do not attempt here to generate complexity parameters for other instruments (besides B♭ Clarinet) both for brevity and for accuracy. If the approach of surveying stakeholders is practical and viable, then we must expand to utilize it further. If surveying will not work, then we must find another approach to validate complexity parameters.

The alternative to this arranged validation is to allow users to supply their own parameters each time without any set standard. Although this workaround would seem necessary to get customized complexity scores for a given playing level of multiple instruments, it requires potentially specifying all the parameters for an entire orchestra. It is unclear at this point how necessary this feature is compared to a more streamlined process for using the program.

8.2 Mapping Separate Parts and Instruments

A related tangential point of future work to gathering more instrument complexity parameters is to separate out different parts and instruments so that each can have its own complexity parameters applied. As mentioned before, our reference implementation does not currently differentiate one part from another. Each part in a musical piece has the complexity parameters applied to it equally (as if each part in the piece was for the same instrument and playing level). It is simple enough to separate out these parts, but the problem becomes matching them to standard complexity parameters.

Within an orchestral or similar piece, the different parts can be named a variety of ways by referencing instruments, players, sections, etc. These can be specific or vague, such as "1st Chair B♭ Clarinet" and "High Brass", respectively. There is no widely accepted, practical standard for how these are specified.

However, we can still attempt to perform this matching. One trivial approach would be to simply keep track of all possible part names our reference implementation ever encounters and periodically update a table that matches the part name in the piece to a set of complexity parameters. A more elegant approach could be to apply some natural language processing techniques to attempt to automatically match the two or, at worst, provide a small subset of alternatives that a user could choose from when running the tool. Yet another alternative could simply be to allow the user to choose exactly which complexity parameters to use for each part at every run. Each of these has its drawbacks in efficiency, usability, and expressiveness. Nonetheless, this problem looms as we move towards more complexity parameters and remains an open area of research that we plan to address.

8.3 Understanding Complexity Scores

One potential issue users face is that to understand a complexity score requires referencing other complexity scores. For instance, the score of 1000 for some piece (or part equivalently) X cannot be meaningfully interpreted without knowing what that piece is, such as a Mozart symphony, or knowing other scores of pieces, such as a Beethoven symphony scoring only 500 so piece X is twice as complex.

One possible solution for this problem is to track the names of pieces (and parts) along with their complexity score in a database. Then, upon scoring some piece, the reference implementation can also output the closest scores of well-known pieces, thus providing a reference point. While keeping track of these scores in a database may also help speed up computation by not repeatedly calculating the score for the same piece over and over, this introduces much more overhead if users are allowed to input any complexity parameters.

Another solution to this problem would be to scale the score down to some range of numbers, such as 0 to 100. Scaling would mean that no piece could have a complexity score greater than 100 or less than 0. While this may not directly solve the problem of understanding the complexity score, it does bound the possible scores and thus provide its own reference point.

This approach may be more useful for competition rankings so that the complexity score can be easily factored into the score for a performance. However, there is no simple way to scale all complexity scores down without knowing what would receive the highest possible score. We are nevertheless investigating this currently to see how we could at least limit scores to some arbitrarily high value and scale based on that.

8.4 Integrating Complexity Scores with Sources

To make musiplectics more accessible, one direction for future work lies in integrating the complexity score into various music applications. For instance, we envision notation applications, such as Finale Notepad Makemusic Inc. [2015a] or Sibelius Avid Technology Inc. [2015], displaying the complexity score of a piece as it is being written so composers can readily see exactly how complex their piece is numerically. Similarly, we would like to partner with music sharing websites, such as International Music Score Library Project Project Petrucci LLC [2015], that allow users to search, view, and download pieces of music. We envision the complexity score of a piece being available before downloading, or more importantly purchasing, the piece so as to give users some reassurance of what they are getting. This could also lead to users being able to search pieces by their complexity score (if they were pre-computed and stored somewhere) should the user need to find a piece to match his or her playing level.

8.5 Including More Input Formats

Yet another direction for future work is the expansion of the formats that can be input in general. At the moment MusicXML files are of course supported, and PDF files can be manually translated to MusicXML via OCR. As mentioned above, this process is not yet mature enough to be run automatically, but OCR in general can operate on many other formats, such as PNG, TIFF, and BMP images, so it would be trivial to expand to allow those inputs.

Beyond what OCR can handle, even more inputs can be translated into MusicXML. Software, such as NotationSoft Notation Software Germany [2014], can translate event-driven MIDI files into MusicXML. This type of translation could bring a wealth more of input since a large amount of music literature is stored in this fashion. In fact, efforts such as Choudhury et al. [2000] are already taking place to digitize a large amount of publicly available music into MIDI. An extension to incorporate translated MIDI files would greatly expand the applicability of musiplectics.

8.6 Measuring Complexity From Physiological Signals

Some prior work has focused on measuring physiological characteristics, especially in the field of human-computer interaction. Often these measurements have been used as indicators of emotional state Knapp et al. [2011]. In relation to music, these measurements have been used both to gauge an audience's reaction to a piece of music as well as a means for people to play their own music Knapp and Lusted [1990] Tanaka and Knapp [2002]. We would like to leverage these types of works to incorporate physiological measurements and biofeedback as a means of forming or validating complexity scores. The knowledge of a performer's relative playing proficiency combined with their basic physiological traits while playing a piece could form a model for extrapolating the cognitive load or mental complexity being endured. This method can become a reliable means of parameterizing our system for individual players.

Chapter 9

Conclusions

This paper presented musiplectics, a new computational paradigm, that systematically evaluates the relative difficulty of music scores, thus benefiting educators and performers. Our hope is that musiplectics can improve the landscape of assessment of music scores. The work presented here unveils our first steps towards an objective and automatic approach to computing the complexity of music pieces. The contributions of this paper include our model for computing complexity scores and its concrete realization in our reference implementation. The automatically computed complexity scores of many well-known pieces and their respective manual grades demonstrate the promise of musiplectics to alleviate the burden of music complexity rankings, freeing musicians for more creative pursuits. In addition, future work directions present many exciting opportunities to apply computing to solve important problem in music arts.

Bibliography

J. Allali, P. Ferraro, P. Hanna, C. Iliopoulos, and M. Robine. Toward a general framework for polyphonic comparison. *Fundam. Inf.*, 97(3):331–346, Aug. 2009. ISSN 0169-2968. URL `http://dl.acm.org/citation.cfm?id=2365196.2365200`.

Apache Xerces. The apache xerces project. `http://xerces.apache.org/index.html`, 1 2013.

J. Arnold. *Clarinet Solos*. Amsco Music Pub. Co, New York, 1939. ISBN 978-0-8256-2028-7.

Avid Technology Inc. Sibelius. `http://www.avid.com/US/products/sibelius`, 3 2015.

H. Bitteur. Audiveris. `https://audiveris.kenai.com/`, 1 2013.

M. Bostock. D3 data-driven documents. `http://d3js.org/`, 2013.

D. Byrd. Music-notation searching and digital libraries. In *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '01, pages 239–246, New York, NY, USA, 2001. ACM. ISBN 1-58113-345-6. doi: 10.1145/379437.379662. URL `http://doi.acm.org/10.1145/379437.379662`.

D. Byrd and M. Schindele. Prospects for improving omr with multiple recognizers. In *ISMIR*, pages 41–46, 2006.

Z. Cataltepe, Y. Yaslan, and A. Sonmez. Music genre classification using midi and audio features. *EURASIP J. Appl. Signal Process.*, 2007(1):150–150, Jan. 2007. ISSN 1110-8657. doi: 10.1155/2007/36409. URL `http://dx.doi.org/10.1155/2007/36409`.

M. M. Cheppudira. vexflow. `https://github.com/0xfe/vexflow`, 2010.

S.-C. Chiu and M.-S. Chen. A study on difficulty level recognition of piano sheet music. In *Proceedings of the 2012 IEEE International Symposium on Multimedia*, ISM '12, pages 17–23, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-0-7695-4875-3. doi: 10.1109/ISM.2012.11. URL `http://dx.doi.org/10.1109/ISM.2012.11`.

G. S. Choudhury, M. Droetboom, T. DiLauro, I. Fujinaga, and B. Harrington. Optical music recognition system within a large-scale digitization project. In *ISMIR*, 2000.

M. S. Cuthbert, C. Ariza, and L. Friedland. Feature extraction and machine learning on symbolic music using the music21 toolkit. In *ISMIR*, pages 387–392, 2011.

M. Good et al. MusicXML: An internet-friendly format for sheet music. In *XML Conference and Expo*, pages 03–04. Citeseer, 2001.

H. Heijink and R. Meulenbroek. On the complexity of classical guitar playing: Functional adaptations to task constraints. *Journal of motor behavior*, 34(4):339–351, 2002. URL `www.summon.com`.

E. Holder. Musicscoring. `https://github.com/xwsxethan/MusicScoring`, 2 2015.

JSON Simple. json-simple. `https://code.google.com/p/json-simple/`, 2 2012.

R. Knapp, J. Kim, and E. Andr. Physiological signals and their use in augmenting emotion recognition for humanmachine interaction. In R. Cowie, C. Pelachaud, and P. Petta, editors, *Emotion-Oriented Systems*, Cognitive Technologies, pages 133–159. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-15183-5. doi: 10.1007/978-3-642-15184-2_9. URL `http://dx.doi.org/10.1007/978-3-642-15184-2_9`.

R. B. Knapp and H. S. Lusted. A bioelectric controller for computer music applications. *Computer Music Journal*, 14(1):pp. 42–47, 1990. ISSN 01489267. URL `http://www.jstor.org/stable/3680115`.

C.-Y. Liou, T.-H. Wu, and C.-Y. Lee. Modeling complexity in musical rhythm. *Complex.*, 15(4):19–30, Mar. 2010. ISSN 1076-2787. doi: 10.1002/cplx.v15:4. URL `http://dx.doi.org/10.1002/cplx.v15:4`.

Makemusic Inc. Finale notepad. `http://www.finalemusic.com/products/finale-notepad/`, 3 2015a.

Makemusic Inc. MusicXML. `http://www.musicxml.com/`, 3 2015b.

MuseScore BVBA. Musescore share your sheet music. `https://musescore.com/`, 2015.

Notation Software Germany. Notation composer. `http://www.notation.com/NotationComposer.php`, 2014.

NYSSMA New York State School Music Association. Nyssma new york state school music association. `http://www.nyssma.org/`, 2015.

Otto and Thornton. Bootstrap. `http://getbootstrap.com/`, 2015.

B. Pearson. *Standard of excellence : comprehensive band method, B clarinet.* Neil A. Kjos Music Co, San Diego, Calif, 1993. ISBN 978-0849759291.

Project Petrucci LLC. International music score library project. `http://imslp.org/`, 2015.

Royal Conservatory Music Development Program. Clarinet syllabus. `https://www.musicdevelopmentprogram.org/sites/default/files/files/S42_Clarinet%20Syl_MDP_2014_online_SECURED.pdf`, 2014.

SpryMedia Ltd. Datatables table plug-in for jQuery. `https://www.datatables.net/`, 2015.

A. Tanaka and R. B. Knapp. Multimodal interaction in music using the electromyogram and relative position sensing. In *Proceedings of the 2002 Conference on New Interfaces for Musical Expression*, NIME '02, pages 1–6, Singapore, Singapore, 2002. National University of Singapore. ISBN 1-87465365-8. URL `http://dl.acm.org/citation.cfm?id=1085171.1085183`.

The jQuery Foundation. jQuery. `https://jquery.com/`, 2015.

Virginia Band and Orchestra Directors Association. Virginia band and orchestra directors association. `http://www.vboda.org/`, 2015.

H. Voxman. *Concert and Contest Collection for Bb Clarinet: Solo Part*. Rubank Publications, Chicago, Illinois, 1992. ISBN 978-1423445449.

J. M. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.

Appendix A

External Survey