LEBANESE AMERICAN UNIVERSITY

AN AUTOMATED APPROACH TO CHORAL PART WRITING

By

HADI FAKHREDDINE

A thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

School of Arts and Sciences
June 2012

Lebanese American University
School of Arts & Sciences

Thesis Proposal Form

Name of Student: Hadi Fakhreddine I.D.#: 200104602

Department: Computer Science and Mathematics

On 27/03/2012, has presented a Thesis proposal entitled:

An Automated Approach to Choral Part Writing

in the presence of the Committee members and Thesis Advisor:

Faisal Abu Khram 27,3,2012
(Name, signature, and date of the Thesis Advisor)

Ramzi Haraty 23/4/2012
(Name, signature, and date of Committee Member)

Samir Habre 24/4/2012
(Name, signature, and date of Committee Member)

Comments/Remarks/Conditions to Proposal:

None.

Date: 24/4/2012 Acknowledged by [Signature]
(Dean of School of Arts & Sciences)

cc: Department Chair
Thesis Advisor
Student
Dean of Graduate Studies/School Dean

Thesis Defense Result Form

Name of student Hadi Fakhreddine I.D: 200104602
Program / Department: Computer Science & Mathematics
Date of thesis defense: June 5, 2012
Thesis title: An Automated Approach to Choral Part Writing.

Result of Thesis defense:

- ☒ Thesis was successfully defended. Passing grade is granted
☐ Thesis is approved pending corrections. Passing grade to be granted upon review and approval by thesis Advisor
☐ Thesis is not approved. Grade NP is recorded

Committee Members:

Advisor: Faisal Abu Khzam
(Name and Signature)
Committee Member: Samir Habre
(Name and Signature)
Committee Member: Ramzi A. Harab
(Name and Signature)

Advisor's report on completion of corrections (if any):

Changes Approved by Thesis Advisor: F. Abu Khzam Signature: _____

Date: _____ Acknowledge by _____
(Dean, School of)

Cc: Registrar, Dean, Chair, Advisor, Student

LEBANESE AMERICAN UNIVERSITY

School of Arts and Sciences - Beirut Campus

Thesis Approval Form

Student Name: Hadi Fakhreddine I.D. #: 200104602

Thesis Title An Automated Approach to Choral

Part Writing.

Program : Master of Computer Science

Department : Computer Science & Mathematics

School : School of Arts and Sciences

Approved by :

Thesis Advisor: Faisal Abu Khzam Signature :

Member : Samir Hake Signature :

Member : Ramy Haraty Signature :

Date : June 5, 2012

THESIS COPYRIGHT RELEASE FORM

LEBANESE AMERICAN UNIVERSITY NON-EXCLUSIVE DISTRIBUTION LICENSE

By signing and submitting this license, you (the author(s) or copyright owner) grants to Lebanese American University (LAU) the non-exclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic format and in any medium, including but not limited to audio or video. You agree that LAU may, without changing the content, translate the submission to any medium or format for the purpose of preservation. You also agree that LAU may keep more than one copy of this submission for purposes of security, backup and preservation. You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone's copyright. If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant LAU the rights required by this license, and that such third-party owned material is clearly identified and acknowledged within the text or content of the submission. IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN LAU, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT. LAU will clearly identify your name(s) as the author(s) or owner(s) of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

Name: Hadi Fakhreddine

Signature: [Redacted] Date: 29/05/2022

PLAGIARISM POLICY COMPLIANCE STATEMENT

I certify that I have read and understood LAU's Plagiarism Policy. I understand that failure to comply with this Policy can lead to academic and disciplinary actions against me.

This work is substantially my own, and to the extent that any part of this work is not my own I have indicated that by acknowledging its sources.

Name: Hadi Fakhreddine

Signature:

Date: 29/05/2012

To my parents

AN AUTOMATED APPROACH TO CHORAL PART WRITING

HADI FAKHREDDINE

Abstract

We present an automated model for distributing choral parts in the style of conventional four part writing, given a melody and the chord progressions. Our system is scalable and allows arrangements for more than four voices, if required. We use enumeration and support it with the constraints imposed by the nature of the problem. Our implementation can successfully transform most music pieces written for solo voices and instruments - old and modern - into singable a capella choral arrangements.

Keywords: Choral, Part writing, CSP, Dynamic Programming, A Capella, Harmony, Arrangement, SATB, Four Voices, Music

TABLE OF CONTENTS

1.	INTRODUCTION.....	1
2.	HARMONY AND ITS RULES.....	4
2.1	TERMS and definitions	4
2.1.1	Pitch	4
2.1.2	Staff.....	4
2.1.3	Pitch notation	4
2.1.4	Clef.....	5
2.1.5	Grand staff	6
2.1.6	Ledger lines.....	7
2.1.7	Accidentals.....	7
2.1.8	Intervals.....	8
2.1.9	Enharmonic equivalents.....	9
2.1.10	The Tritone.....	9
2.1.11	Duration	10
2.1.12	Rhythm.....	11
2.1.13	Scale	12
2.1.14	Harmony	13
2.1.15	Chord.....	13
2.1.16	Triad	13
2.1.17	Chord inversions	14
2.1.18	Seventh Chords	15
2.1.19	Other chords.....	15
2.2	Harmony and choral part writing	1
2.2.1	About four-voice part writing	1
2.2.2	Stylistic practices	1
2.2.2.1	Root position.....	2
2.2.2.2	First inversion	3
2.2.2.3	The viiø6 triad.....	4

2.2.3 The blacklist imposed by stylistic rules	4
2.2.4 Less severe blacklist	5
3. CONSTRAINT SATISFACTION PROBLEMS.....	7
3.1 Definition	7
3.2 CSP Example: map coloring.....	1
3.3 Applications	4
4. CHORAL PART WRITING AS A CSP.....	5
4.1 Problem variations	5
4.1.1 Problem variation: only the melody is imposed	5
4.1.2 Problem variation: Only unfigured bass is imposed	6
4.1.3 Problem variation: only figured bass is imposed	6
4.1.4 Problem variation: two-voice problems	7
4.2 Modeling choral part writing as a CSP	8
5. ALGORITHM.....	11
5.1 Overview	11
5.2 Input	12
5.3 Quality score: the algorithm's fitness function	13
5.4 Enumeration: listing all possible solutions	15
5.5 Code overview: Enumeration and constraints	19
5.6 Results.....	22
6. OPTIMIZATION WITH DYNAMIC PROGRAMMING.....	26
6.1 Overview	26
6.2 Code overview: Enumeration, constraints and dynamic programming..	28
6.3 Results.....	30
7. SUMMARY	33
8. REFERENCES.....	34

LIST OF TABLES

Table 5.1 Penalty types and values at the score bar level.....	13
Table 5.2 Penalty types and values at the score bar couples level.....	14
Table 6.1 Dynamic programming score tracker	27

LIST OF FIGURES

Figure 2.1 An empty staff	4
Figure 2.2 Pitch notation on the piano keyboard	5
Figure 2.3 Staff with treble clef (clef de Sol in French notation)	5
Figure 2.4 Staff with bass clef (clef de Fa in French notation)	5
Figure 2.5 The Grand staff	6
Figure 2.6 The relationship between the grand staff, the standard 88-key piano keyboard and middle C	6
Figure 2.7 Ledger lines	7
Figure 2.8 Notation of accidentals	7
Figure 2.9 Intervals	8
Figure 2.10 Interval examples: perfect fourth and perfect fifth	8
Figure 2.11 Interval examples: perfect unison and perfect octave	8
Figure 2.12 Interval examples: Major intervals	9
Figure 2.13 Interval examples: Minor intervals	9
Figure 2.14 Enharmonic equivalents	9
Figure 2.15 The tritone	10
Figure 2.16 Notation of duration	10
Figure 2.17 The dot notation	11
Figure 2.18 Examples of time signatures	11
Figure 2.19 The notation of quarter notes per measure	11
Figure 2.20 Beat division	12
Figure 2.21 Extract from Haydn's 94 th Symphony in G Major (also known as the “Surprise” symphony)	13
Figure 2.22 Harmony: the vertical aspect of music	13

Figure 2.23 Chords.....	13
Figure 2.24 Triad	14
Figure 2.25 Chord inversion	14
Figure 2.26 An example of seventh chords	15
Figure 2.27 Other chords	15
Figure 2.28 Four-voice example notation.....	1
Figure 2.29 Keep common tone and move other voices by one step to nearest chord tone.....	2
Figure 2.30 Keeping common tone is not possible, move all three upper voices in similar motion to nearest chord tone.....	2
Figure 2.31 Preserving common tones when roots are in thirds.....	3
Figure 2.32 Upper voices moved in “contrary motion” to the bass.....	3
Figure 2.33 Double the third (bass note)	4
Figure 2.34 Vocal ranges	5
Figure 2.35 Rule violations of stylistic practices (1)	5
Figure 2.36 Rule violations of stylistic practices (2)	6
Figure 2.37 Voice crossing	6
Figure 3.1 Map coloring: overview map of Australia.....	1
Figure 3.2 Map coloring: example solution of Australia's map (1)	2
Figure 3.3 Map coloring: example of Australia's map (2)	3
Figure 3.4 Map coloring problem represented as a constraint graph.....	3
Figure 4.1 Problem variation: melody and chord progression imposed	6
Figure 4.2 Figured bass as originally given	7
Figure 4.3 Harmony deduced from the same figured bass in simple position.....	7
Figure 4.4 Beats divisions.....	8

Figure 4.5 Violation of C_a for rule “Bass 6-chord” at beat B_1	9
Figure 4.6 Violation of C_b for rule “parallel-5th” between beats B_1 and B_2	10
Figure 5.1 Enumeration of a score bar.....	16
Figure 5.2 Enumeration growth.....	17
Figure 5.3 Solution scoring and solution pruning.....	18
Figure 5.4 Results: initial solution of Bach's chorale N.69	22
Figure 5.5 Results: attempt to fix initial solution of Bach's chorale n.69	23
Figure 5.6 Results: best solutions (1).....	24
Figure 5.7 Results: best solutions (2).....	24
Figure 5.8 Results: best solutions (3).....	25
Figure 6.1 Results: Brahms - Twinkle, twinkle little star	30
Figure 6.2 Results: Beethoven - Ode to joy	30
Figure 6.3 Results: Frank Sinatra - My way	31
Figure 6.4 Results: Pitbull - Give me everything tonight	31
Figure 6.5 Results: Black eyed peas - gotta feeling	32
Figure 6.6 Results: Fairuz - El helwa di (arabic folk song)	32

1. INTRODUCTION

Since its early discovery and practices, music theory has been through a lot of formalization. Specifically “tonal music” – in contrast to “atonal” music, has been subjected to a good deal problem solving in computer models. We call music “tonal” when its aspect is based on the idea of tonality, where different pitches are related to each other through a key center.

In this thesis we focus on tonal music since this form has long been formulated and structured in a well-established mathematical approach, as opposed to other kinds of musical structures (such as the aforementioned atonal music). Western music is essentially based on tonal music. Baroque, Classical, Romantic music and today’s Jazz, Blues, Rock and Pop are all based on western music’s well established theories. The idea behind tonality classifies the sounds in a musical oeuvre into musical stacks of notes called scales. These scaled are hierarchically arranged within the musical piece. From an artistic point of view, arranging music consists in shaping notes around “tonal” centers. This organization makes tonal musical pieces possible and meaningful.

Arranging and composing music within the scope of this “tonality” has long been the subject of many treatises through the centuries. Johan Fux’s “Gradus ad parnassum” written in 1725 (Fux, 1965) is probably one of the most influential of them all until date. Fux was one of the first theorists to define specific rules for counterpoint. This form of composition consists of combining different musical lines together; these different lines, or melodies, interplay with each other in a meaningful fashion, producing a rich sound. Many influential composers, including Johan-Sebastian Bach and Franz-Joseph Haydn, have studied composition basing themselves on Fux’s theories. Tonality was later on formalized as theory by Jean-Philippe Rameau, a very famous and influential French

composer who developed his famous treatises in 1722 (Rameau, 1971). Rameau's approach involved reducing music theory or more accurately said - developed it - into mathematical structures. While previous treatises on harmony had been purely practical, Rameau added a philosophical dimension whereby he derived harmonic principle from natural causes, and the composer quickly rose to prominence in France as the "Isaac Newton of Music" (Thomas Christensen, 2002).

More than a thousand theories on harmony have been written. Almost all of today's recognized composers have written their own treatise. Most of these serve as a basis of rules that specify how notes should and should not be combined together. The notes are interrelated by the notion of chord. Rules support these chords and their sequences. Most of the harmonic rules will be described in the next chapter.

It is good to note that all treatises on harmony do not take into account the same rules. In fact, stylistic practices have evolved along with music and history. To be more accurate, music came to existence first, and then its rules followed. However, some basic rules haven't changed during this evolution of "tonal" music; these rules have always been included in all the written treatises. An example of one of these rules is the parallel fifth restriction, which is discussed later on. Another note worth mentioning is that these rules do not intend to form a well-defined sequence of steps when composing or arranging music. They are in most cases a form of specifications of allowable and forbidden practices – that are sometimes purposely broken. Truly, some composers defend the fact that the freedom of composition arises from violations of some of the constraints imposed by these rules (Pachet & Roy, 2001). In a way or the other, this is what separates one composer from the other. Finally, abiding by the rules in a composition does not guaranty that the result will be "musically interesting" to the ear;

the final result can only be considered musically – and theoretically – correct. Art simply does not arise from a series of regulations; it needs to be humanly created. Important

2. HARMONY AND ITS RULES

Before delving into the theory behind music, harmony in particular, it is worthy to list and define some of the common terms we will encounter in the following section.

2.1 TERMS AND DEFINITIONS

2.1.1 Pitch

The term pitch describes the highness or lowness (the frequency) of a tone. In music notation, pitches are represented by symbols positioned on a staff and identified with letter names.

2.1.2 Staff

The staff consists of five equally spaced horizontal lines. The figure below illustrates an empty staff.

Figure 2.1 An empty staff

2.1.3 Pitch notation

The various pitches are referred to by the first seven letters of the alphabet: A to G, as shown on the piano keyboard in Figure 1.2. This notation varies from culture to culture. For example in the French music notation, these pitches are referred to La Si Do Ré Mi Fa Sol respectively.

Figure 2.2 Pitch notation on the piano keyboard

2.1.4 Clef

At the beginning of a music line, we place the clef symbol. The clef defines the pitch names of the lines and spaces of the staff.

The treble clef or G clef (clef de Sol in French notation) is an ornate letter G. The curved line terminates at the second line of the staff, thus designating the letter name of a note on that line as G.

Figure 2.3 Staff with treble clef (clef de Sol in French notation)

The bass clef is called the F clef because it was derived from the letter F. The dots are placed above and below the fourth line of the staff, designating that line as F.

Figure 2.4 Staff with bass clef (clef de Fa in French notation)

2.1.5 Grand staff

Together, the treble and bass staves make up a grand staff. Figure 1.5 shows the point at which both clefs converge. The two Cs are the same pitch: middle C.

Figure 2.5 The Grand staff

The grand staff is associated most often with keyboard music. The below figure shows the relationship between the grand staff, the standard 88-key piano keyboard and middle C.

Figure 2.6 The relationship between the grand staff, the standard 88-key piano keyboard and middle C

2.1.6 Ledger lines

Pitches that go beyond the limits of the staff are written by adding ledger lines above or below the staff. Ledger lines, which parallel the staff, accommodate only one note. Some examples are illustrated below.

Figure 2.7 Ledger lines

2.1.7 Accidentals

Accidentals are symbols that are placed to the left of the note heads to indicate the raising or lowering of a pitch.

- The flat symbol (b) lowers the pitch value by a half step downwards.
- The sharp symbol (#) raises the pitch value by a half step upwards.
- The natural symbol (n) cancels of the previous and reduces back the pitch to its natural state.

Figure 2.8 Notation of accidentals

2.1.8 Intervals

An interval is the relationship between two tones. In Western music, the half step is the smallest interval used. It is the interval between any two adjacent keys—black or white—on the keyboard.

Figure 2.9 Intervals

Figure 2.10 Interval examples: perfect fourth and perfect fifth

Figure 2.11 Interval examples: perfect unison and perfect octave

Figure 2.12 Interval examples: Major intervals

Figure 2.13 Interval examples: Minor intervals

2.1.9 Enharmonic equivalents

Enharmonic equivalents are tones that have the same pitch but different letter names.

Figure 2.14 Enharmonic equivalents

2.1.10 The Tritone

A tritone contains three whole steps as illustrated in the figure below. Tritones are naturally difficult for singers.

Figure 2.15 The tritone

2.1.11 Duration

The notation of duration is illustrated in the following chart.

Name	Note	Rest	Equivalents
Breve (Double Whole Note)			Two Whole Notes
Whole Note			Two Half Notes
Half Note			Two Quarter Notes
Quarter Note			Two Eighth Notes
Eighth Note			Two Sixteenth Notes
Sixteenth Note			Two Thirty-second Notes
Thirty-second Note			Two Sixty-fourth Notes
Sixty-fourth Note			Two One Hundred Twenty-eighth Notes

Figure 2.16 Notation of duration

A dot next to the note allows the lengthening the value of the note by half again its value. A second dot lengthens the dotted note value by half the length of the first dot.

Figure 2.17 The dot notation

2.1.12 Rhythm

Rhythm is a general term used to describe the motion of music in time. The fundamental unit of rhythm is the pulse or beat. Even persons untrained in music generally sense the pulse and may respond by tapping a foot or clapping.

Meter and the time signature

Meter can be defined as regular beats. This recurring pattern is identified at the beginning of a composition by a meter signature (time signature).

Figure 2.18 Examples of time signatures

In the figure above, the upper digit indicates the number of basic note values per measure. It indicates the number of pulses per measure. The lower digit indicates a basic note value: the upper number signifies a half note and the lower number refers to a quarter note, 8 to an eighth note, and so forth. The following figure illustrates how this notation translates into musical terms.

Figure 2.19 The notation of quarter notes per measure

Following is an illustration of simple meters showing the division of the beat.

Figure 2.20 Beat division

2.1.13 Scale

A scale is a collection of pitches. The pitches are represented in the scale in an ascending and descending order. The scale is a convenient way of displaying the notes used in a musical piece. The next figure shows a melody consisting of 24 notes but only seven different letter names.

Figure 2.21 Extract from Haydn's 94th Symphony in G Major (also known as the "Surprise" symphony)

2.1.14 Harmony

Harmony is said to be the vertical aspect of music: it is the audible result of tones sounding simultaneously.

Figure 2.22 Harmony: the vertical aspect of music

2.1.15 Chord

A chord is stack of notes comprised of at least three different tones sounding together.

Figure 2.23 Chords

2.1.16 Triad

A triad is any three-tone chord.

Figure 2.24 Triad

When arranging for four voices – or more – the notes of a triad can only be assigned to three singers. The remaining singers would be assigned enharmonic tones of the triad. The composer can choose to either assign perfect octaves or perfect unisons to the remaining voices.

2.1.17 Chord inversions

When the root of the chord is not the lowest-sounding pitch, we say the chord is “inverted”.

Figure 2.25 Chord inversion

Triads in first inversion are used for a number of purposes, including to smooth bass lines and to provide melodic motion in repeated chords. Triads in root position establish stability in the chorale and are considered anchor positions, but if all chorales or hymns were composed only of root positions, bass lines would be disorganized and fragmented. One of the reasons first inversions are employed is to provide smooth bass lines with a musical balance of steps and skips (Benward & Saker, 2008).

2.1.18 Seventh Chords

Seventh chords are formed when we add a third above the fifth in a triad.

Figure 2.26 An example of seventh chords

The seventh chord imposes a minimum of four voices. However in some situations, the composer can omit the fifth of the chord.

2.1.19 Other chords

We can continue adding thirds to triads. This results in even more complex chords: 9th chords, 11th chords, and even 13th chords.

Figure 2.27 Other chords

To illustrate the complexity of what these chords can impose on the arranger, the thirteenth chord illustrated in the above figure requires seven different singers. Of course in practice, these chords are rarely used.

2.2 HARMONY AND CHORAL PART WRITING

2.2.1 About four-voice part writing

In four-voice textures, the interaction of harmony and melody and their equal importance become clear. The four individual melodic lines come together, generating a chord, while maintaining smooth melodic connections from pitch to pitch (Benward & Saker, 2008).

The four voices (soprano, alto, tenor and bass) are drawn on the grand staff. The G-clef staff shows the pitches of the two higher voices: soprano and alto. The F-clef staff shows the pitches of the two lower voices: tenor and bass.

Figure 2.28 Four-voice example notation

In the previous figure, notice the melodic contour of the bass voice. It consists mostly of leaps because the bass voice sings the root factor of each of the chords in the phrase. In four-voice textures the bass is usually a harmonic voice that is controlled more by the chords than by melodic considerations. In contrast, the soprano, alto, and tenor voices move mostly in conjunct motion (i.e.: no big leaps).

2.2.2 Stylistic practices

Stylistic practices are common methods for part writing a particular progression. In our model, we tend to adhere as closely as possible to this practice in order to achieve stylistic results in our generated solution(s).

In this section, we list the most common of these practices.

2.2.2.1 Root position

Root position indicates that a chord is its “original” form, i.e.: has no inversion.

When both chords are in root position, and the two roots lie a perfect 5th or 4th apart, keep the “common tone” and move the remaining upper voices by steps. If it is not possible to keep the “common tone” (this mainly occurs when the soprano voice descends scale degrees “2̂” to “1̂”), move all upper voices.

This practice is illustrated in the following two figures.

Figure 2.29 Keep common tone and move other voices by one step to nearest chord tone

Figure 2.30 Keeping common tone is not possible, move all three upper voices in similar motion to nearest chord tone

When both roots are a major or minor third apart, keep both “common tones” and move the remaining upper voice by steps.

Figure 2.31 Preserving common tones when roots are in thirds

When both roots are in minor or major seconds apart, move the upper voices in “contrary motion” to the bass.

Figure 2.32 Upper voices moved in “contrary motion” to the bass

2.2.2.2 First inversion

When chords are in first inversion, double any triad factor that facilitates smooth voice leading. Favored notes are the soprano and bass. Never double the leading tone.

2.2.2.3 The viiø6 triad

The viiø6 triad, or the leading-tone triad, is nearly always found in first inversion and progresses most often to the tonic.

On a leading-tone triad, double the bass note or fifth factor.

Figure 2.33 Double the third (bass note)

2.2.3 The blacklist imposed by stylistic rules

The previous stylistic rules can occasionally be bent. However, when arranging for choirs, some rules are stricter and cannot be bent. Following is a list of four strictly forbidden practices.

1. Avoid parallel perfect octaves (P8ths), parallel perfect fifths (P5ths), and parallel unisons (P1s).
2. Never double the leading tone of the scale.
3. Keep voices within vocal range of the assigned voice.
4. Avoid augmented seconds.

Figure 2.34 Vocal ranges

The theoretical vocal range of each voice is illustrated in the previous figure.

The next figure shows some violations of these rules.

Figure 2.35 Rule violations of stylistic practices (1)

2.2.4 Less severe blacklist

Following is a list of rules that can only be broken in situations where no alternatives exist.

1. Avoid cross overs. Keep voices arranged from highest to lowest: soprano, alto, tenor then bass.
2. Spacing between adjacent voices should not exceed an octave in the three upper voices.
3. Do not overlap two adjacent voices more than a whole step.

4. Unequal fifths, P5ths to d5ths or vice versa, are found in chorale harmonization and may be used sparingly.
5. The leading tone should progress upward to the tonic when it is in an outer voice (soprano or bass).

The next figure shows some violations of these rules.

Figure 2.36 Rule violations of stylistic practices (2)

Figure 2.37 Voice crossing

3. CONSTRAINT SATISFACTION PROBLEMS

In this section we explore briefly the theory behind constraint satisfaction problems and their applications.

In short, we define constraint satisfaction as an efficient way to solve a wide variety of problems (Russell & Norvig, 2010). In order to do that, we formulate the problem as a set of variables, each of which has a value. We then list a set of constraints on these variables. The problem is considered solved when each of these variables satisfies all the constraints applied to them. This type of problems is called Constraint Satisfaction Problem, or CSP for short.

The main advantage behind CSPs is that it uses general-purpose heuristics; in contrast, other heuristics deal with the problem in a problem-specific fashion.

3.1 DEFINITION

Formally, CSPs are defined as the triple: $\langle X, D, C \rangle$ where,

- X is the finite set of variables defined in the problem: $X_1, X_2, \dots, X_n$
- D is the non-empty domain of possible values for each of these variables:
 $D_{X_1}, D_{X_2}, \dots, D_{X_n}$
- C is the finite set of constraints applied to the variables: $C_1, C_2, \dots, C_n$

Each constraint C_i limits the values a variable X_n can take. For instance, a constraint C_1 can limit the variable X_1 's value to a positive non-zero integer: $C_1: X_1 > 0$.

A solution to a CSP is considered *complete* if and only if all the constraints in the domain of the variables are satisfied.

3.2 CSP EXAMPLE: MAP COLORING

The map coloring problem is a straight-forward example of a typical constraint satisfaction problem. The following figure shows an overview map of Australia.

Figure 3.1 Map coloring: overview map of Australia

The problem imposes the coloring of each of the regions either blue, red or green in such a way that no neighboring states have the same color.

From a CSP aspect, we define the variables of the problem to be the regions:

$$X = \{WA, NT, SA, Q, NSW, V, T\}$$

The domain D of the variables is the set of allowable colors:

$$D = \{\text{blue, green, red}\}$$

The constraints require the neighboring regions to have distinct colors. One representation of these constraints is:

$$C = \{WA \neq NT, WA \neq SA, NT \neq SA, NT \neq Q, SA \neq NSW, SA \neq V, V \neq T\}$$

Taking a closer look at the constraint $(WA \neq NT)$, we intuitively deduce that the enumeration of colors that the couple (SA, WA) can have is:

$\text{ENUM}(\{\text{SA}, \text{WA}\}) = \{(\text{green}, \text{red}), (\text{blue}, \text{red}), (\text{red}, \text{green}), (\text{blue}, \text{green}), (\text{red}, \text{blue}), (\text{green}, \text{blue})\}$

This, rather long, enumeration opens the possibility for many possible solutions to the problem. An example solution is the set of region-color couples:

$\text{SOL}_1 = \{(\text{WA}, \text{red}), (\text{NT}, \text{green}), (\text{SA}, \text{blue}), (\text{Q}, \text{red}), (\text{NSW}, \text{green}), (\text{V}, \text{red}), (\text{T}, \text{green})\}$

This example solution is illustrated below.

Figure 3.2 Map coloring: example solution of Australia's map (1)

Another very similar solution is:

$\text{SOL}_2 = \{(\text{WA}, \text{red}), (\text{NT}, \text{green}), (\text{SA}, \text{blue}), (\text{Q}, \text{red}), (\text{NSW}, \text{green}), (\text{V}, \text{red}), (\text{T}, \text{blue})\}$

Figure 3.3 Map coloring: example of Australia's map (2)

The only difference between SOL_1 and SOL_2 is the last couple.

The problem can be visualized as the constraint graph illustrated next, whereby regions are represented as vertices and the edges connect the neighboring regions.

Figure 3.4 Map coloring problem represented as a constraint graph

The leading advantage in formulating this problem as a CSP and solving it is that a CSP solver can quickly eliminate large color swatches of the search space. For instance, once the chosen region-color couple is (SA, blue), the CSP solver immediately concludes

that none of the five neighboring variables can take on the value blue. Since the neighboring regions of SA are five (WA, NT, Q, NSW and V) and the colors' domain consists of three colors (red, blue and green), a standard search procedure would consider $3^5 = 243$ assignments. With the advantage of the CSP solver's conclusion mentioned earlier, we reduce the number of assignments to $2^5 = 32$, a reduction of 87%. In some way, instead of searching the entire space on every iterative assignment, the CSP solver is drawing its attention only to the variables that matter.

As a result of this behavior, many problems that tend to yield a large search space (such as our choral part writing problem) can be resolved using CSPs with a more optimal performance.

3.3 APPLICATIONS

The most successful applications of CSPs in real-life situations include, but are not limited to, solving Sudoku problems, the Eight Queens puzzle and the map coloring problem we saw earlier. Because of the constraints naturally imposed in these types of problems, CSPs give a great approach to their resolution and present themselves as effective in most situations. One very important point to note is the following: the more constraints a problem has, the better the results. This idea obviously emerges from the fact that the more constraints there are in a given problem, the smaller the search space becomes, and therefore the faster and the more efficient the algorithm can perform.

In this thesis, we elevate the scope of application of CSPs to the artistic domain of music. As mentioned earlier, this domain presents itself with theories that impose an elaborate number of constraints in the cycle of the arranger's process in choral part writing. The next section expands on how this aspect can be formulated as a CSP.

4. CHORAL PART WRITING AS A CSP

In this chapter we describe our approach in modeling the choral part writing problem as a constraint satisfaction problem.

A combinatorial aspect of the problem arises when we are imposed to produce harmony with a melody imposed. In this case, the harmonization problem naturally becomes a constraint satisfaction problem. Its variables are the notes sought to produce the harmony. The vocal range of the corresponding voice (soprano, alto, tenor and bass) would be the domain. More constraints are added while trying to abide by the rules of harmony.

Before diving into the model itself, it is important to note that in such a problem we require a minimum of given input. In traditional exercises in composition, there are many variations for formulating the choral part writing problem. The input given differs from one variation to another.

4.1 PROBLEM VARIATIONS

4.1.1 Problem variation: only the melody is imposed

In this first variation, the melody line (soprano line) is imposed, and the arranger is asked to fill in the 3 other lower voices. Given that a chord progression is imposed, the arranger would first attempt to figure out the bass line, remembering that the bass is always singing the root of each chord. Filling in the remaining middle parts becomes a mechanical task (or so). The next challenge that arises is the correct assignment of voices of these middle parts. Of course, the difficulty of the problem increases in parallel with the number of voices required by the chord.

In this paper, we focus on this variation of the problem.

Figure 4.1 Problem variation: melody and chord progression imposed

4.1.2 Problem variation: Only unfigured bass is imposed

In this variation, only the bass line is imposed and no chord progressions are indicated. The minimal amount of information given leaves the arranger with an huge amount of possibilities for the harmony's arrangement. "The problem of unfigured bass" has been long debated in music theory, and studies have shown that it is in practice largely under-specified" (Rothgeb, 1968).

4.1.3 Problem variation: only figured bass is imposed

In this variation, the bass line is imposed as well as the chords' progressions. By having a bass line and chord structure, this reduces drastically the number of possibilities for filling the notes of missing vocal parts. It is good to note that this kind of problem is used to train harpsichord players, where they are asked to "figure out and play chords that satisfy the bass voice, the melody figures and another sung melody, all in real-time" (Pachet & Roy, 2001).

Figure 4.2 Figured bass as originally given

Figure 4.3 Harmony deduced from the same figured bass in simple position

4.1.4 Problem variation: two-voice problems

In this variation, a melody is imposed and the arranger needs to find the bass only. This requires a more relaxed set of rule. There is no need to elaborate on this variation in our focus.

4.2 MODELING CHORAL PART WRITING AS A CSP

In our model, at each given beat, we consider the following four variables:

X_s , X_a , X_t and X_b , where:

- X_s is the pitch sang by the soprano
- X_a is the pitch sang by the alto
- X_t is the pitch sang by the tenor
- X_b is the pitch sang by the bass

Each of these variables has for domain the vocal range of the respective voice:

$[Low_s, High_s]$ is the domain of variable X_s where:

- Low_s is the lowest singable pitch by the soprano voice
- $High_s$ is the highest singable pitch by the soprano voice

Similarly, the domains of the remaining three variables are $[Low_a, High_a]$, $[Low_t, High_t]$ and $[Low_b, High_b]$ respectively.

The next figure separates harmony divisions in terms of beats.

Figure 4.4 Beats divisions

Of course in practice, this division is not realistic; but for simplicity, we will base the next formulations on this abstraction.

The stylistic practices clearly impose constraints on two levels: the level of a single beat, where all pitches of the beat need to satisfy the rules among each other, and the level of the beat succession where the collection of pitches of the beat need to satisfy the rules in relation to the collection of the previous beat.

In summary, at any given beat B_n :

- All four pitches at beat B_n must satisfy the set of constraints C_a
- All four pitches at beat B_n must satisfy the set of constraints C_b in relation to the pitches in beat B_{n-1}

where we divide the stylistic practices into two separate constraint domains:

- C_a is the set of constraints affecting pitch values in a single beat
- C_b is the set of constraints affecting pitch values of beat B_n in relation to pitch values of beat B_{n-1}

To help illustrate better, we show examples of violations at both constraint levels next.

Figure 4.5 Violation of C_a for rule “Bass 6-chord” at beat B_1

In the previous figure, the stylistic practice “bass 6-chord” is violated at the first beat. The set of constraints C_a does not permit an octave difference between any two given voices.

Figure 4.6 Violation of C_b for rule “parallel-5th” between beats B_1 and B_2

In the previous figure, beats B_1 and B_2 both satisfy the constraints in the set C_a , however they violate one of the constraints in the set C_b : no parallel fifths can occur between two successive beats.

As we can notice, constraints can be applied at two different levels: at the level of a given beat and at the level of a succession of beats. The latter gives a great advantage to a CSP problem: this violation of one or more constraints at two successive beats leads to a huge drop of the enumerated search space, therefore optimizing the CSP solver’s performance.

5. ALGORITHM

The proposed algorithm is based on enumerations of voice-note couples supported by the natural constraints imposed by the problem.

5.1 OVERVIEW

In our approach to the choral part writing problem we assume the melody line and chord progressions are given. By requiring the chord progression to be given, we keep the composer's artistic thoughts visible throughout the final result – the arranged score. Following the stylistic practices, we also assume that the melody line is the highest part on every chord change and every beat.

Given the melody and chord on every beat, we can instantly deduce the bass line on every chord: the bass always sings the root of the chord. With that input, we can assume that both border lines are given: melody and bass; we only need to draw our attention to the middle lines: alto and tenor. In a triad chord, we can deduce 3 notes. If the melody is a non-chord tone, then we are immediately restricted to the 3 notes of the chord and the melody note in the case where we are arranging for 4 voices. If the melody is a chord tone, we are forced to have one of the 3 notes doubled by a second voice. This is the natural process of a choral arranger. Conclusively, our problem is reduced to this question: in the inner voices, which note is sung by the alto and which is sung by the tenor?

5.2 INPUT

Our input is given in the form of a semi-colon delimited file, whereby each line is translated into a beat, or as we call it: a score bar. The input line consists of the melody note, followed by the chord and length of the note. In the following example, the note given is D with length 4, and the chord is D Major (DM):

D;DM;4

Optionally we can add text for every note, such as in the example of Brahms' famous "twinkle, twinkle little star":

D;DM;4;Twinkle

Our algorithm first reads this line and immediately assigns the note D on the 4th octave to the soprano. Since the chord is D Major, the allowable notes are: D, F#, and A. The bass takes the root D an octave lower than the melody, the 3rd octave. The remaining notes to be assigned are F# and A. By a quick peak at the ranges of these remaining notes, the algorithm deduces that both are either best fit in the alto's range or the tenor's. It assigns each to a voice. This forms a basic "guess" for the question "which voice sings which note?" in the middle parts on every score bar.

After reading the entire input file we would have deduced a simple, non-feasible, solution that most probably violates a record number of constraints imposed by the stylistic practices. Up to this point, we would have solved this exercise using the skills of a bad theory student and obtained the most obvious result on every score bar regardless of the advanced rules and relations between the bars.

5.3 QUALITY SCORE: THE ALGORITHM'S FITNESS FUNCTION

How do we determine that this basic solution is good or bad? The algorithm provides a way to qualify a solution's score on 3 levels:

- The level of a single note: by qualifying how good this note fits in its assigned voice's range
- The level of the score bar: by qualifying how well the score bar and its note-to-note relations abide by the stylistic practices
- The level of the set of score bars, or solution: by qualifying how well the relations between the score bars abide by the stylistic practices

The quality's score comes in the form of penalties: for every stylistic practice violation, the algorithm penalizes with a value. For example, a penalty at the note's level drops the solution's quality by -60 if the note is outside or close to the border of the voice's range. A penalty of -600 is given at the score bar's level if a voice cross-over is detected. Notice that this number is far greater than the previous, alerting that this violation is more critical to the overall solution's health. A penalty of -600 is also given at the level of two successive score bars in the occurrence of parallel 5ths.

We list next the most used penalty types and their values at the score bar and solution levels in our algorithm. At the note's level, the only penalty given is -60 for every semitone outside the range of the assigned voice (i.e.: if the note is a single semitone above the maximum vocal range we penalize with a value of -60; if it is two semitones above the maximum vocal range, we penalize with a value of $2 \times -60 = -120$ and so forth).

Table 5.1 Penalty types and values at the score bar level

Penalty type	Penalty value
--------------	---------------

VocalRangeTop	-60
VocalRangeBottom	-60
CrossOver	-600
MissingVoiceInScoreBar	-200
MissingNoteInChord	-200
DoubledVoices	-5
VocalDistance	-5

Table 5.2 Penalty types and values at the score bar couples level

Parallel4th	-600
Parallel5th	-600
Parallel8th	-600
ParallelUnison	-600
AltoStepWiseLeap	-50
TenorStepWiseLeap	-50
OctaveSkipLeap	-400
AltoLeapOver9	-400
AltoLeap3or4	-10
AltoLeap5or7or8	-400
AltoLeap6	-800
TenorLeapOver9	-400
TenorLeap3or4	-10
TenorLeap5or7or8	-100

TenorLeap6	-500
------------	------

This quality score measurement serves as the algorithm’s fitness function and allows us to qualify each solution’s overall suitability. We also define a threshold for a solution’s minimum allowable quality score which proves to be of great importance in our enumeration described later. The threshold varies from one input to the other and is left for the user’s experimentation in which threshold results in better solutions and optimal performance of the algorithm.

5.4 ENUMERATION: LISTING ALL POSSIBLE SOLUTIONS

Our enumeration consists of shuffling the inner voices (alto and tenor) and combining them with all possible notes lower than the soprano and higher than the bass.

In the example of “twinkle, twinkle little star”, the choice of assigning the F# to the tenor and A to the alto is one possible solution. The other is to invert the voices: the F# is assigned to the alto and the A is assigned to the tenor. This is another possible solution. As long as the note is in the range of the voice, we assume it is a possible choice before proceeding to the remaining restrictions of the stylistic practices. This gives the algorithm a chance for finding an optimal solution in its next stages.

We perform this shuffling of voice-note couples for every score bar. If the total number of voice-note couples on every bar were 2, then our initial basic solution would branch out to a total of 2^n possible solutions, n being the number of score bars.

Figure 5.1 Enumeration of a score bar

In practice, a typical pop sheet music consists 100 score bars on average. By following this shuffling scheme, our algorithm would attempt to generate a total of $2^{100} = 1,267,650,600,228,229,401,496,703,205,376$ solutions! An enumeration explosion.

Figure 5.2 Enumeration growth

As a remedy and precaution, we enforce a threshold limit per score bar quality: anything below that limit is ignored and the solution is automatically ignored. This pruning proves itself key in our approach.

Figure 5.3 Solution scoring and solution pruning

One of our test results shows that the number of enumerated solutions for a 91 score bars sheet music resulted in 107 solutions instead of 2^{91} . Another enumerative explosion problem arises when the number of possible notes is larger than 2. This can occur on 7th, 9th and 13th chords, or any chord that requires more than 3 notes. When a chord requires 5 notes, we are left with 3 unassigned notes (having given one for the soprano and one for the bass). The enumeration instantly jumps to factors of 3 instead of 2; 2^n becomes 3^n . Luckily our threshold and strong constraints restrict the enumeration to a reasonable amount of solutions.

5.5 CODE OVERVIEW: ENUMERATION AND CONSTRAINTS

```
void Main()

{
    List<ScoreBar> scoreBars = readScoreBarsFromFile();
    List<Solution> solutions = generateSolutions(scoreBars);
    ExportSolutionsAsPDF(solutions);
}

List<Solution> generateSolutions(List<ScoreBar> scoreBars)
{
    Solution init_sol = deduceInitSol(scoreBars);

    List<Solution> possibleSolutions = enumerateSolutions(init_sol, 0);
    possibleSolutions = sortByQualityScore(possibleSolutions);

    return possibleSolutions;
}

Solution deduceInitSol(scoreBars)
{
    For each scoreBar in scoreBars
    {
        Note melody = scoreBar.getMelody();
        Note bass = scoreBar.getBass();
        List<Note> innerVoices = scoreBar.AssignInnerVoices();
    }

    Solution solution = buildSolution(melody, bass, innerVoices);

    Return solution;
}
```

```

List<Solution> enumerateSolutions(Solution sol, int startIndex)
{
    List<Solution> solutions;
    List<ScoreBar> shuffled = shuffleInnerVoices(sol, startIndex);

    For each scoreBar in shuffled
    {
        Solution solution = buildSolution(shuffled);

        If (getQualityScore(solution) >= SolutionPenaltyLimit)
        {
            // Add this solution, then all its branched out solutions
            Solutions.Add (solution);
            Solutions.AddMany(enumerateSolutions(solution, scoreBar.Index + 1));
        }
    }

    Return solutions;
}

```

The enumeration takes place in *enumerateSolutions()*. It is done for each of the read score bars in the initial solution. On every iteration through the score bars, the function recursively branches out into possible solutions on the next score bar.


```

int getQualityScore(Solution solution)
{
    int qualityScore = 0; // Neutral score

    For i = 0 solution.ScoreBars.Count - 1
    {
        For each note in solution.ScoreBars[i].Notes
        {
            qualityScore += getNoteScore(note);
        }

        qualityScore += getScoreBarQualityScore(solution.ScoreBars[i]);
        qualityScore += getCouplesScore(solution.ScoreBars[i], solution.ScoreBars[i+1]);
    }

    Return qualityScore;
}

```

The function *getQualityScore()* is our fitness function and bases itself on the values described in the previous tables. Notice that the solution's overall quality score is dependent on every note's quality, every score bar's quality and every couple of score successive score bars' quality. Each quality is measured with the respective functions: *getNoteScore()*, *getScoreBarQualityScore()* and *getCouplesScore()*.

5.6 RESULTS

We performed various tests on music scores of various lengths, all of which are shown in the next section. Here we show the results of a simple Bach chorale.

In the next figure, we see the result deduced from our initial solution.

Figure 5.4 Results: initial solution of Bach's chorale N.69

The score of -5170 is achieved by simply attempting to deduce an initial solution. A relatively low score for only 7 score bars. For comparison purposes, we next show the result of an attempt to fix some of the errors with a few utility functions that attempt to fix some major stylistic practices violations such as parallel 5ths, cross overs, etc...

Figure 5.5 Results: attempt to fix initial solution of Bach's chorale n.69

A reduction of some negligible 70 score points does not get us close to a feasible result. The next figure shows a series of top choices we achieved after our enumeration and selection of the leading solutions.

Figure 5.6 Results: best solutions (1)

Figure 5.7 Results: best solutions (2)

Figure 5.8 Results: best solutions (3)

The leading result with a score of -630 shows an example of a feasible solution which violates a minimal number of constraints. A great improvement from the mediocre -5170 indeed.

6. OPTIMIZATION WITH DYNAMIC PROGRAMMING

6.1 OVERVIEW

Having enumerated the possible solutions and listed them in ascending order of fitness, the human intervention can take place where the arranger decides which of the top proposed solutions fits best. The enumerative method proposes best choices based on the algorithm's quality scoring.

On top of our enumeration-based approach, we additionally provide a strong optimal method for deducing the optimal solution for the choral part writing problem by using Dynamic Programming, or DP for short. This second approach results in a single solution, judged to be the fittest.

By using DP we start by selecting the fittest combination of voice-note couples on the first score bar – by leaning on the bar-based quality scoring – then move on to selecting the next score bar, making sure it complies best with the first. This technique is applied by iteration throughout the entire list of score bars read from the input. The end result is a single solution designed to have all successive score bars made compatible as much as possible.

Throughout our iteration among the score bars we keep track, using a table, of the best scores obtained in each variation. We then re-visit this table and select the score bar with the top registered quality.

Table 6.1 Dynamic programming score tracker

	Score Bar Index								
Variation Index	0	1	2	3	4	5	6	...	n
	0	-60	-50	-72	-60	-80	-800	...	-950
	1	-78	-41	-23	-82	-42	-10	...	-10
	...	-82	-96	-960	-950	-50	-35	...	-610
	k	-800	-10	0	-65	-982	-15	...	-800

The table above shows the quality scores for all variations at a given score bar's index. Each column represents a score bar given by the input, ranging from 0 to n . A row in the column is a variation obtained by shuffling the inner voices. The value stored in each score bar's variation is the quality score deduced for that particular variation. At a given score bar index i , the variation j has a solution quality $Q[i][j]$. For instance, the first score bar's initial variation has a score of $Q[0][0] = -60$. Its second variation has a quality score of $Q[0][1] = -78$. The algorithm initially populates the table with the quality scores deduced for each score bar's variation based on the score bar level constraints. Next, it updates each of these values by deducing the quality score of every two consecutive score bar couples. When selecting the optimal solution, we visit each score bar's variations in the table and pick the one with the best score as the optimal score bar. Our end solution would then be comprised of a set of fittest score bars and, most importantly, fittest consecutive score bar couples. The performance of this selection is in the order of $O(kn)$, k being the maximum number of variations available per score bar and n being the total number of score bars given by the input.

6.2 CODE OVERVIEW: ENUMERATION, CONSTRAINTS AND DYNAMIC PROGRAMMING

```
void Main()

{
    List<ScoreBar> scoreBars = readScoreBarsFromFile();
    Solution solution = generateFittestSolution(scoreBars);
    ExportSolutionAsPDF(solution);
}

// Stores best variations index per score bar
int[] BestSolutions;

Solution generateFittestSolution (List<ScoreBar> scoreBars)
{
    Solution init_sol = deduceInitSol(scoreBars);

    // Enumerate each voice-note couple variation on each score bar and store in matrix
    ScoreBar[][] Variations = enumerateScoreBarVariations(scoreBars);

    // Select fittest variation on first score bar
    int bestInitialVariationIndex = getFittestVariation(scoreBars[0]);

    BestSolutions[0] = bestInitialVariationIndex;

    // Start coupling with the next score bar
    couple(Variations, 0, bestInitialVariationIndex);

    // Build solution containing the fittest successive score bar couples
    Solution bestSol = new Solution();

    // Take each variation's best index from BestSolutions
    for (int i=0; i < BestSolutions.Count; i++)
        bestSol.ScoreBars.Add(Variations[i][BestSolutions[i]]);

    return bestSol;
}
```



```

void couple(ScoreBar[][] Variations, int barIndex, int VariationIndex)
{
    // Base case
    if (barIndex >= Variations.Count - 1)
        return;

    // Initialize best score as worse value
    int bestScore = int.MinValue;
    int bestVariationIndex = 0;

    // Test each variation of the next score bar and select the fittest
    for (int i=0; i < Variations[barIndex + 1].Count; i++)
    {
        // Create solution for testing fitness
        Solution solutionTest = new Solution();
        solutionTest.ScoreBars.Add(Variations[barIndex][variationIndex]);
        solutionTest.ScoreBars.Add(Variations[barIndex + 1][i]);

        if (solutionTest.getQualityScore() > bestScore)
        {
            bestScore = solutionTest.getQualityScore();
            bestVariationIndex = i;
        }
    }

    BestSolutions[barIndex + 1] = bestVariationIndex;

    // Recursive call to next score bar
    couple(Variations, barIndex + 1, bestVariationIndex);
}

```

6.3 RESULTS

The results of the DP approach are remarkable. We show next different results of music scores with different lengths. We even had success with an Arabic folk song as shown below.

Twinkle, twinkle little star

The image displays a musical score for the song "Twinkle, twinkle little star". It consists of two staves. The top staff is in treble clef with a key signature of one sharp (F#) and a common time signature (C). The bottom staff is in bass clef with a key signature of one sharp (F#) and a common time signature (C). The notes are color-coded: red for the melody, green for the bass line, and blue for the accompaniment. The lyrics are written below the top staff: "Twin- kle twin- kle lit- tle star how I won- der what you are". The score is labeled with "1" on the left and "-500" on the right.

Figure 6.1 Results: Brahms - Twinkle, twinkle little star

Beethoven - Ode to joy

The image displays a musical score for the song "Beethoven - Ode to joy". It consists of two staves. The top staff is in treble clef with a key signature of one sharp (F#) and a common time signature (C). The bottom staff is in bass clef with a key signature of one sharp (F#) and a common time signature (C). The notes are color-coded: red for the melody, green for the bass line, and blue for the accompaniment. The lyrics are written below the top staff: "Freu- de scho- ner Got- ter- fun- ken Toch- ter aus El- y- si- um". The score is labeled with "1" on the left and "-440" on the right.

Figure 6.2 Results: Beethoven - Ode to joy

Frank Sinatra - My way

1 Yes there were times I'm sure you knew when I bit off more than I could chew but through it

-4920

6 all when there was doubt I ate it up and spit it out I faced it all and I stood tall and did it my way

Figure 6.3 Results: Frank Sinatra - My way

Pitbull - Give me everything tonight

1 To- -night I will love love you to- -night give me e- -very- thing to- -ni-

-1100

5 -i- -ight for all we know we might not get to- -mo- -rrow let's do it to- -night

Figure 6.4 Results: Pitbull - Give me everything tonight

Black eyed peas - I gotta feeling

1 I got a fee- ling that to- night's gon- -na be a good night that to- -night's go- -nna be

-2900

6 a good night that to- -night's go- -na be a good good ni- -i- -ight I got a fee- ling

Figure 6.5 Results: Black eyed peas - gotta feeling

Fairuz - El helwa di

1 El hel- l- wa di a- a- mit te- e- gin be- l ba- d- ri- i- a wed- di-

-4070

6 ik bi yen- dah kou- kou- kou- kou bel fa- g- ri- i- yya yal- la bi- na a- la

Figure 6.6 Results: Fairuz - El helwa di (arabic folk song)

7. SUMMARY

The choral part writing problem is naturally a constraint satisfaction problem and can be solved using standard CSP techniques. In our approach we enumerate the many solutions available from the given input, and prune the unfit subsets by applying the constraints given in the part writing stylistic practices. This approach yields into a smaller set of suitable solutions and gives the user an opportunity to select a solution he/she prefers best.

On top of our proposed enumeration, we provide a variation that outputs a single, best fit, solution using dynamic programming. This variation gives the user an optimal solution in polynomial time.

Experimentation proves that with constraint satisfaction problems better and faster results can be achieved with more constraints imposed on the variables. Indeed, the more constraints we have the more we eliminate and prune from our solution subsets. On top of the twenty two constraints we imposed in our approach, more practical results can be achieved by implementing even more constraints naturally imposed by the choral part writing problem.

8. REFERENCES

- Benward, B., & Saker, M. (2008). *Music in theory and practice (8th ed.)*. New York: McGraw-Hill.
- Fux, J. J. (1965). *The study of counterpoint from Johann Joseph Fux's Gradus ad Parnassum*. New York: Norton & Company.
- Pachet, F., & Roy, P. (2001). Musical harmonization with constraints: A survey. *Constraints Journal*, 6(1):7-19.
- Rameau, J. (1971). *Treatise on Harmony*. New York: Dover.
- Rothgeb, J. (1968). *Harmonizing the unfigured bass: A Computational Study*. Cambridge: UML.
- Russell, S. & Norvig, P. (2010). *Artificial Intelligence: A modern approach (3rd ed.)*. New Jersey: Pearson Education Inc.
- Christensen, T. (2002). *The Cambridge History of Western Music Theory*. Cambridge: Cambridge University Press.