Valued Constraint Satisfaction Problems Applied to Functional Harmony

Nele Verbiest, Chris Cornelis and Yvan Saeys
Ghent University, Ghent, Belgium
Email: {Nele.Verbiest, Chris.Cornelis, Yvan.Saeys}@UGent.be

Abstract— *Harmonization with four voices is a musical problem which is subject to hard constraints, which absolutely need to be fulfilled, as well as to soft constraints, which preferably hold, but are not mandatory. In this paper, we model this problem as a valued constraint satisfaction problem (VCSP): costs are assigned to possible solutions based on the constraints they violate. We design an algorithm that finds a minimal-cost solution, thus solving the harmonization problem, and we present initial results obtained by this algorithm.*

Keywords— functional harmony, valued constraint satisfaction problems, soft constraints

1 Introduction

Ever since people have had computers at their use, they have attempted to make them acquire human skills like rational thought and creativity. While the prospect of computers actually creating art might still seem out of reach, in musical composition some successful efforts have already been made; in [1], Truchet and Codognet give an overview of musical problems that were solved using computers.

In this paper, we discuss the use of Valued Constraint Satisfaction Problems (VCSPs) in functional harmony with four voices (soprano, alto, tenor, bass). Solving harmonization problems is a basic skill that every composer has to acquire before attempting more serious work. These problems range from very simple to advanced, with many gradations in between. Their solution is governed by a set of strict rules that have to be verified, but also by a variety of "softer" guidelines with varying, subjective importance, imposed to make the result more interesting, more varied, ... Valued constraints can be constructed and enforced sensibly to model them.

The structure of this paper is as follows: in Section 2, we discuss related work, and in particular solvers that have already been proposed for this problem, and position our own contribution. In Section 3, we briefly explain what functional harmony is about, and which constraints govern harmonization problems. In Section 4, we give the necessary background on VCSPs, while in Section 5 we translate the musical composition problem and its constraints to this framework. In Section 6 we describe the algorithm designed for solving our VCSP. In Section 7 we present some results obtained by this algorithm, and in the last section we discuss its practical use and suggest future research directions.

2 Related work

Programs to compose music have been designed since the 50s. We are specifically interested in harmonization problems with four voices, a problem that can be stated as a constraint satisfaction problem [2]. To solve this problem, two main classes of techniques have been described in the literature: backtracking and genetic algorithms.

Backtracking algorithms tackle the problem in the following way. The harmonization of a piece of music proceeds in a left to right fashion, analyzing every subsequent note. For each note, one seeks a possible chord for that note, and if there is no solution for this note, the algorithm goes back to a preceding note to try other possibilities. The standard work in this class is due to Ebcioğlu [3]: he introduced an expert system for harmonizing chorales with four voices, based on a set with about 350 rules. Apart from this, there are also backtracking algorithms where possible chords for a note are ranked by musical suitability. At every step, one tries the best chord. If this chord does not lead to a solution, the chord is removed from the ranking and a solution with the second best chord is sought [4]. Existing backtracking solvers look for one solution and then stop.

Genetic Algorithms (GAs) use an evaluation function to judge the musical quality of a set of solutions, and proceed in an iterative way to optimize these solutions [5, 6], using evolutionary reproduction operators like mutation and crossover.

Both approaches have their advantages and disadvantages. Backtracking algorithms are very efficient in finding a single solution, but little can be said about its quality: any solution found by the algorithm is guaranteed to meet the imposed hard constraints, but soft constraints are generally not considered. Ranking possible chords for a note deals only partially with this problem; this tactic will find a good solution, but it is not guaranteed to be the best one. By virtue of the evaluation function, genetic algorithms are better at catering for soft constraints, but they also suffer from the fact that their solutions are often suboptimal.

The solver we describe in this paper combines the use of soft constraints (which can be seen as the analogon of the evaluation function in GAs) with the potential to find an optimal solution in the search space.

3 Functional harmony with four voices

We briefly describe the essentials of functional harmony, and refer the interested reader to [7, 8] for more details. The problem we consider is a simplification of the general harmonization problem: given a melody for a soprano, we aim to find corresponding melodies for alto, tenor and bass. We suppose that only quarter notes are used in all the melodies and we work in C major. In tonal music, one speaks about grades, which —for a particular tuning— coincide with a set of chords. The most important grades that are always considered are I, IV and V. Other grades that often appear are VI and sometimes II. In C major tuning, the first note is c which results in grade I being the chord built on c; the fourth note is f,

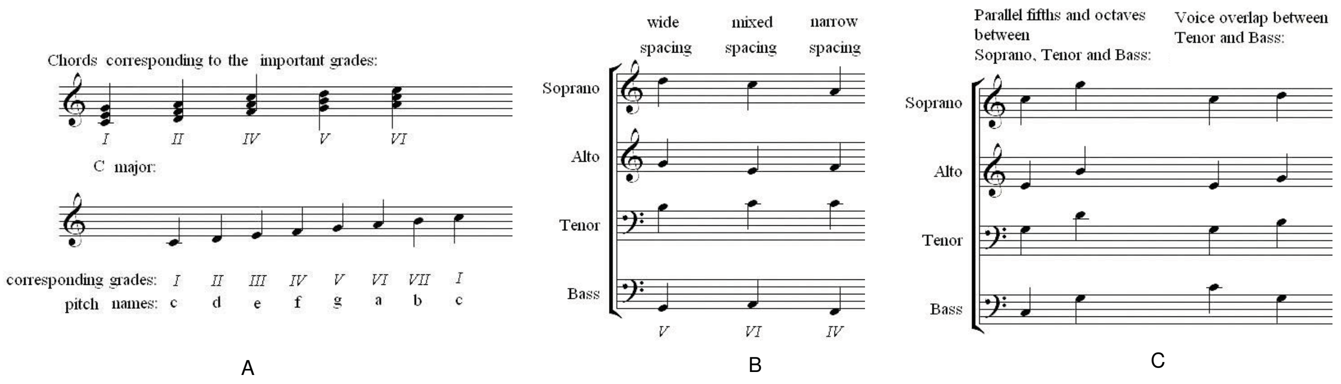
Chords corresponding to the important grades:
I II IV V VI
C major:
corresponding grades: I II III IV V VI VII I
pitch names: c d e f g a b c
A
wide spacing
mixed spacing
narrow spacing
Soprano
Alto
Tenor
Bass
V VI IV
B
Parallel fifths and octaves between Soprano, Tenor and Bass:
Voice overlap between Tenor and Bass:
Soprano
Alto
Tenor
Bass
C

Figure 1: Essentials of functional harmony, illustrating the grades and chords of C Major (A), the concepts of narrow, mixed and wide spacing (B), and some examples of constructs that are not allowed (C).

so the chord corresponding to grade IV is the chord built on f. Fig. 1A illustrates the grades and chords of C major.
Now consider a note in the soprano. This note is in one or more of these chords. In harmonizing the soprano, one must choose a chord, and attribute the corresponding notes in the other voices to a note of this particular chord. The bass must always be the ground note (the lowest note) of the chord. For example, if IV is chosen, then the bass must be f. The soprano, alto and tenor have to be different from each other, so each note of the chord appears once, except for the ground note which appears twice.
We now introduce the concepts *wide* and *narrow* spacing. For every note in the soprano, we still have some freedom for the notes in alto, tenor and bass. We now restrict the possibilities a bit more. For each chord we choose, we have to decide whether we write it in wide or in narrow spacing. Narrow spacing means that between soprano and alto, and between alto and tenor there is no other note of the chord. Wide spacing means that there is always one note of the chord between soprano and alto, and between alto and tenor. This is illustrated in Fig. 1B. If the previous chord is written in narrow spacing, then the next chord must be narrow spaced as well. If the previous chord is wide spaced, then so must be the next chord.
The VIth grade chord is written like the other chords, except for the case when it is preceded by a Vth grade. In this case, the second note of the chord is written twice. This means that it has neither wide nor narrow spacing; we refer to this situation as *mixed* spacing. When we encounter a sequence V-VI it is possible to go from wide to narrow spacing or vice versa using mixed spacing on the VIth grade. Fig. 1B shows a good V-VI-IV sequence where we switch spacing.
Now we know how to write the chords we must consider the problem of choosing the right grade. To this end, a number of rules have been defined in the domain of functional harmony:

- The first and the last grade of a music piece must be I. The other grades depend on the previous grades.
- Some transitions between grades are not allowed, others are considered very good, while still others are not considered wrong but should be avoided as much as possible.
- Each of the different voices has its own range of notes that it can sing. The soprano can sing the highest notes, the bass sings the lower notes, and care has to be taken that these ranges are respected.
- It is forbidden to write parallel fifths or parallel octaves, to write the same grade before and after a bar or to overlap voices.

Some of these forbidden constructs are shown in Fig. 1C.

4 Valued constraint satisfaction problems

A constraint satisfaction problem (CSP) involves assigning values to variables that are subject to some constraints [9]. If there are constraints that are preferred but not essential, we can consider the problem of assigning values to the variables in such a way that the more important constraints are fulfilled and the less important ones are fulfilled to the extent possible. The latter yields a valued CSP (VCSP), which is characterized by a set of hard constraints (that must be fulfilled), and a set of so-called *valued constraints*. If the valued constraints are treated as mandatory, the problem is often unsolvable. If the valued constraints are ignored, we get solutions of bad quality. Our interest is therefore in the solution that best respects the set of constraints. To express what the best solution is, we assign costs to valued constraints. Every time a valued constraint is violated, we count the cost of this violation. In the end, we are interested in the solution with the lowest cost. We recall a formal definition of VCSP's.

Definition 1. *A Valued Constraint Satisfaction Problem (VCSP) is a quadruple (X, D, C, V), where*
$X = \{X_1, ..., X_n\}$ *is a set of variables,*
$D = \{D_1, ..., D_n\}$ *is a set of finite domains: D_i is the set of possible values for X_i,*
$C = \{C_1, ..., C_r\}$ *is a set of hard constraints and*
$V = \{V_1, ..., V_s\}$ *is a set of valued constraints.*

A hard constraint C_i on the ordered set of variables $var(C_i)$ is the relation of the allowed combinations of values for the variables in $var(C_i)$; C_i is a mapping from the cartesian product of domains of variables in $var(C_i)$ to $\{0, 1\}$.
A valued constraint V_i on the ordered set of variables $var(V_i)$ specifies the cost of the combination of the values for the variables in $var(V_i)$; V_i is a mapping from the cartesian product of domains of variables in $var(V_i)$ to $[0, 1]$.
For example, if V_i pertains to the variables X_1, X_2 (i.e.

ISBN: 978-989-95079-6-8

$Var(V_i) = (X_1, X_2)$) and we assign values x_1 and x_2 to X_1 and X_2 respectively, the cost of this combination is given by $V_i(x_1, x_2)$. If C_i pertains to the variables X_1, X_2, X_3 (i.e. $Var(C_i) = (X_1, X_2, X_3)$) and we assign values x_1, x_2 and x_3 to these variables, the combination (x_1, x_2, x_3) is allowed if $C_i(x_1, x_2, x_3) = 1$ and forbidden if $C_i(x_1, x_2, x_3) = 0$.

5 Modeling the problem

We can now translate the musical problem of harmonizing a soprano to a purely mathematical problem. Suppose we have a soprano that lasts n times, with every time divided in 4 counts. All the times contain four quarter notes, except for the last time which contains one whole note. This makes a total of $l := (n-1)*4+1$ notes. We call l the length of the problem. For every note, we want to store the notes for the bass, tenor, alto and soprano and we want to store the grade we use. This can be done using a $5 \times l$ matrix X.
The grades and notes are represented by a positive integer, and the central c is represented by the number 24. We assign integers to the other notes as follows: if the note is higher/lower than the central c, we count the number of half tones between the central c and this note and add/subtract it to 24. In this way, we fix the domain and variables of our VCSP. The variables are the entries $x_{i,j}, i = 1, ..., 5, j = 1, ..., l$ of the matrix X and the domain is the set of natural numbers $\mathbb{N}$.

5.1 *Hard constraints*

We will first translate the hard constraints to mathematical formulas. We only use grades I, II, IV, V and VI, which entails:

$$x_{5,j} \in \{1, 2, 4, 5, 6\}\ \forall j = 1, ..., l.$$

The first and last grade must be I, so

$$x_{5,1} = x_{5,l} = 1.$$

Every voice has a specific range of notes that it can sing, which can be expressed as:

$$\text{alto} : 19 \leq x_{2,j} \leq 38\ \forall j = 1, ..., l$$

$$\text{tenor} : 12 \leq x_{2,j} \leq 28\ \forall j = 1, ..., l$$

$$\text{bass} : 5 \leq x_{2,j} \leq 26\ \forall j = 1, ..., l$$

The grades before and after a bar must be different:

$$\forall j = 1, ..., l :$$

$$\text{if } j \bmod 4 \equiv\ 1 \text{ and } j \neq 1$$

$$\text{then } x_{5,j} \neq x_{5,j-1}$$

No parallel fifths or octaves are allowed:

$$\forall i, k, j : i, k = 1, 2, 3, 4, j = 1, ...l-1 :$$

$$\text{if } (x_{i,j} - x_{k,j} \bmod 12 \equiv 0) \text{ and } k \neq i$$

$$\text{then } (x_{i,j+1} - x_{k,j+1} \bmod 12 \not\equiv 0)$$

and

$$\text{if } (x_{i,j} - x_{k,j} \bmod 12 \equiv 7) \text{ and } k \neq i$$

$$\text{then}(x_{i,j+1} - x_{k,j+1} \bmod 12 \not\equiv\ 7).$$

It is forbidden to write voice overlap (strictly between soprano, alto and tenor, not strictly between tenor and bass):

$$\forall j = 1, ..., l :$$

$$x_{1,j} > x_{2,j} > x_{3,j} \geq x_{4,j},$$

$$\forall j = 1, ..., l-1 :$$

$$x_{2,j+1} < x_{1,j}$$

$$x_{3,j+1} < x_{2,j}$$

$$x_{4,j+1} \leq x_{3,j}.$$

The possibilities for the notes in the bass, tenor and alto and the forming of the chords (with wide and narrow spacing) are also hard constraints. We will not explicitly state these constraints here as mathematical formulas due to space constraints.

5.2 *Valued constraints*

We are now ready to introduce some valued constraints. There are five aspects of functional harmony we model using valued constraints. In the following subsections we propose techniques for calculating the cost of certain combinations. The results are five costs in $[0, 1]$ that correspond to the transition of one grade to another (c_1), the frequency of grades (c_2), the range of notes a voice can sing (c_3), the distance between two succeeding notes (c_4) and the use of contrary motion between bass and soprano (c_5). To allow a composer to determine the relative importance of each type of valued constraints, we attach weights $\{w_1, w_2, ..., w_5\}$ and define the total cost as $\sum_{i=1}^{5} w_i c_i$ where all weights sum to one. Assigning these weights depends on personal taste. Some composers might think it is important to have a lot of variation in the use of grades, while others might prefer to write music that is in the right range for the different voices. The weights we propose are as follows: $w_1 = 0.3, w_2 = 0.2, w_3 = 0.1, w_4 = 0.2, w_5 = 0.2$.

5.2.1 *Grade transition constraints*

As stated before, we only work with grades I, II, IV, V and VI. The most important grades are I, IV and V. V can be seen as a chord that builds up tension, I creates rest and IV is an advancing chord (to V). In music, it is important to have enough tension but there are points of rest needed as well.
II is a grade that can be used as a replacement grade for IV. It is also an advancing grade to V. VI is a grade that can replace I or IV, depending on the context. Generally speaking, when followed by V or preceded by I, VI is a replacement grade for IV, while when preceded by V or followed by IV, VI is a replacement grade for I. (This is also valid if we replace IV by II.) So depending on the context, VI can bring rest in music or can advance to tension. II and VI are used to bring more variation in music. For example, instead of writing I-IV-V-I (a very good sequence) all the time, we can also write I-II-V-VI.
To express the aptitude of grade sequences, we introduce a valued constraint G. It is a mapping from the cartesian product of the possible grades to the interval $[0, 1]$ that represents the cost of the sequence:
$G : \{1, 2, 4, 5, 6\} \times \{1, 2, 4, 5, 6\} \to [0, 1]$.
To determine the cost of a sequence, we only look at two consecutive grades. This is sufficient since every bad sequence is

the result of one or more separate bad successions in the sequence.

The assignment of the costs is an intuitive and subjective task. However, we try to bring some structure in it. The model sequence is I-IV-V-I. Therefore, we assign cost zero to every subsequence of length two that appears in it. All sequences that can be obtained from those subsequences by substituting a grade by its replacement grade (we call it *derivative sequences*) also have no cost.

Going from advancing to tension to rest is not very interesting. Therefore, we assign cost 1 to the sequence IV-I and all derivative sequences. Going from rest to tension without advancing should be avoided but is not particularly bad, so we assign cost 0.5 to the sequence I-V. Using the same sequence two times is not always interesting. We assign cost 0.3 to sequences I-I, V-V and derivatives, and cost 0.4 to sequences IV-IV and derivatives. An overview of the assignments is given in Table 1.

5.2.2 *Grade frequency constraints*

As stated before, in music it is important to maintain sufficient variation. This is not covered by the previous constraints: writing I-IV-V-I the whole time would give zero cost, but it would be boring to listen to. To express this we introduce additional valued constraints.

The algorithm designed in Section 6.1 has three options: it can generate a solution with only grades I, IV and V, only grades I, IV, V and VI, or using all grades I,II, IV, V and VI. The number of times a grade should ideally be used depends on this option, so we introduce three different valued constraints. First consider the option where only grades I, IV and V can be used. The best sequence here is I-IV-V-I. We want some variation so we sometimes write I-I, I-IV-I or I-V-I. Say that ideally, every three times we write I-IV-V, we should write I-I, I-IV-I or I-V-I once. The corresponding grade occurence percentages are given in Table 2.

We now define the valued constraint $H_1 : \{1, 4, 5\}^l \to [0, 1]$ as follows: for the whole piece of music, count how many times grades I, IV or V are used and calculate the percentages. Now for every grade, add the absolute difference between the ideal and real percentage, and divide the result by three to obtain a value in $[0, 1]$.

Since grade VI is a replacement grade for I, in the option where I, IV, V and VI can be used, we consider the same percentages as in the first option for IV and V, but we split the ideal percentage for grade I. Analogously, as grade II has the same function as grade IV, we split the ideal percentage for grade IV in the third option. The percentages are presented in Table 2. The valued constraints H_2 and H_3 are defined as before.

Again, these ideal percentages are constructed subjectively and intuitively. Different composers would propose different percentages, but they should be similar. Moreover, as for the previous valued constraints, the exact percentages are not very important, more significant are the gradations.

5.2.3 *Voice range constraints*

The problem we consider is to write a music piece for four voices. Of course, these voices cannot sing all existing notes. Theoretically, a bass can sing notes presented by integers from 5 to 26, yet there exist very good basses that can not reach the very low note presented by 5 or the very high note presented by 26. It is not wrong to write these extreme notes but it should be avoided, because even if the bass can sing these notes, they will not sound as good as the notes in the middle of their register. To express this mathematically, we introduce new valued constraints R_a, R_b and R_t as follows: for every note in the reach of a voice, R_b, R_t and R_a express how bad this note is for bass, tenor or alto respectively. Fig. 2 presents the values of R_a, R_b and R_t. The total cost is given by $(R_a + R_b + R_t)/3 \in [0, 1]$.

5.2.4 *Distance constraints*

In tonal music, it is important to have fluent melodies. Modeling this requires counterpoint techniques, which we do not consider here. Instead, we model the soundness of the melodies by considering the distances between the notes. Furthermore, it is hard to sing large intervals correctly. Therefore, we introduce a valued constraint D that limits the distance between two notes. We assign cost 1 to all intervals larger than an octave (12 half tones). Intervals smaller than an octave get cost $c_4 = d/12$, proportional to the distance d between the notes. An exception is made for the octave, which gets the same cost as a fifth because it sounds equally well and is equally easy to sing. The costs are presented in Table 3.

5.2.5 *Contrary motion constraints*

Contrary motion is the general movement of two melodic lines in opposite directions. That is, when one of the lines moves up, the other line moves down. In tonal music, contrary motion is important to maintain independence of melodic movement. To express this as a valued constraint we assign a cost

Table 1: The mapping G that assigns a cost to every allowed succession of two grades

GRADE1-GRADE2	COST=G(GRADE1,GRADE2)
II-V, V-I, I-II, IV-V	0
VI-V, VI-II, I-IV, V-VI	0
VI-IV, V-I, I-VI	0
I-I, V-V, VI-I, VI-VI	0.3
IV-II, IV-IV, II-II	0.4
I-V	0.5
IV-I, II-I, II-VI	1.0

Table 2: valued constraint H compares the count of grades to the ideal percentages.

GRADE	IDEAL PERCENTAGES		
	OPTION 1: I, IV AND V	OPTION 2: I, IV, V AND VI	OPTION 3: I, II, IV, V, VI
I	0.394	0.263	0.263
II	-	-	0.075
IV	0.303	0.303	0.228
V	0.303	0.303	0.303
VI	-	0.131	0.131

ISBN: 978-989-95079-6-8

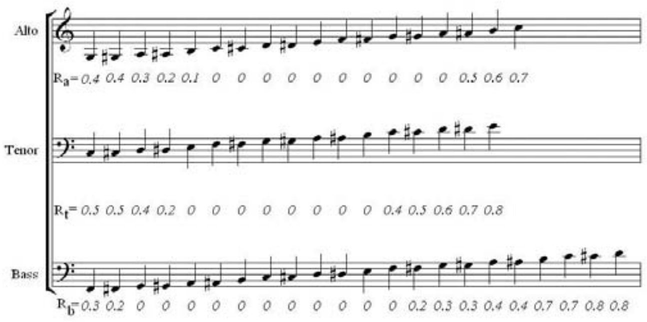
Alto
Tenor
Bass
R_a= 0.4 0.4 0.3 0.2 0.1 0 0 0 0 0 0 0 0 0 0 0.5 0.6 0.7
R_t= 0.5 0.5 0.4 0.2 0 0 0 0 0 0 0 0 0.4 0.5 0.6 0.7 0.8
R_b= 0.3 0.2 0 0 0 0 0 0 0 0 0 0 0 0.2 0.3 0.3 0.4 0.4 0.7 0.7 0.8 0.8

Figure 2: Costs that are incrurred when notes at the extreme ends of a voice's scale are used.

$c_5 = 1$ if there is no contrary motion between bass and soprano.

6 Solving the four voices harmonization VCSP

Now we translated the musical problem to a purely mathematical problem, we can design an algorithm that solves it. There exist good solvers for VCSPs [9] but we choose to design and implement a separate algorithm for the harmonization problem. The algorithm follows the thinking of a composer that solves the exercise: from left to right, try out all the possibilities and return when a solution gets stuck.

6.1 A backtracking algorithm

Algorithm 1 shows the outline of the main part of the algorithm. First note that it returns a single minimal-cost solution; if there are several such solutions, our algorithm returns the first one encountered. At each stage, max_cost and total_cost represent the cost of the optimal solution so far, and of the currently investigated solution, respectively. They are initialized in lines 1 and 2.

From Section 2, we know that the grade on the first note must be I. Either narrow or wide spacing can be used for the first chord; the former option is considered in lines 3–5, the second one in line 6–8.

The central part of the algorithm is in the recursive procedure Calculate_Solution. For every note i except the last one, all possible solutions are generated (line 2), that is, all permissible combinations of grades for note i, and corresponding notes for bass, tenor and alto. Assuming no hard constraints are violated (line 3), the cost of a solution w.r.t. the valued constraints is evaluated (line 4); all costs can be calculated directly, except from c_2 which is calculated considering the problem from the beginning until position i. c_2 is only calculated when $i > 4$. If the total cost for the current solution is smaller than the new costs added up to the total cost, one has to try the next possibility, because from here we will never find a solution that is cheaper than the solution we already found (line 5-6). Otherwise, one adds the costs except from c_2 up to the total cost and goes further looking for a solution (lines 8-9). c_2 is not added up to the total cost since this cost is calculated every step and it counts for the whole piece.

When we have reached the final note, we change the current solution and the maximum cost and we return in the recursion to find a cheaper solution (lines 16-17).

This can lead to a solution or to failure. In both cases we return to try the next posibilities. If there are no posibilities left, the algorithm does not go further in this branch of the recursion.

Table 3: valued constraint D is a measure for the soundness of an interval.

INTERVAL (IN HALF TONES)	COST c_4	INTERVAL (IN HALF TONES)	COST c_4
> 12	1.000	6	0.500
12	0.583	5	0.417
11	0.917	4	0.333
10	0.833	3	0.250
9	0.750	2	0.167
8	0.667	1	0.083
7	0.583	0	0.000

Algorithm 1: Algorithm that solves the problem of harmonizing a melody for a soprano, l is the length of the problem.

```
1 max_cost=Double.max;
2 total_cost=0;
3 Fill in the first chord, using narrow spacing;
4 total_cost=cost of this first chord (only $w_3c_3$);
5 Calculate_Solution(2,$l$);
6 Fill in the first chord, using wide spacing;
7 total_cost=cost of this first chord (only $w_3c_3$);
8 Calculate_Solution(2,$l$);
```

Procedure `Calculate_Solution(`**note position** *i*, **total length** *l*`)`

```
1  if ($i < l$) then
2      foreach possibility for note $i$ do
3          if no hard constraints violated then
4              calculate costs ;
5              if total_cost+$\sum_{i=1}^{5} w_i c_i >$ max_cost then
6                  go to next possibility;
7              else
8                  total_cost=total_cost+$\sum_{i=1}^{5} w_i c_i - w_2 c_2$;
9                  Calculate_Solution(i+1);
10             end
11         else
12             go to next possibility;
13         end
14     end
15 else
16     max_cost=total_cost+$w_2 c_2$;
17     save solution;
18 end
```

7 Results

In this section we present some test results. We tested the program for several problems but we will show only one representative example here.

We illustrate the suitability of our approach by showing four

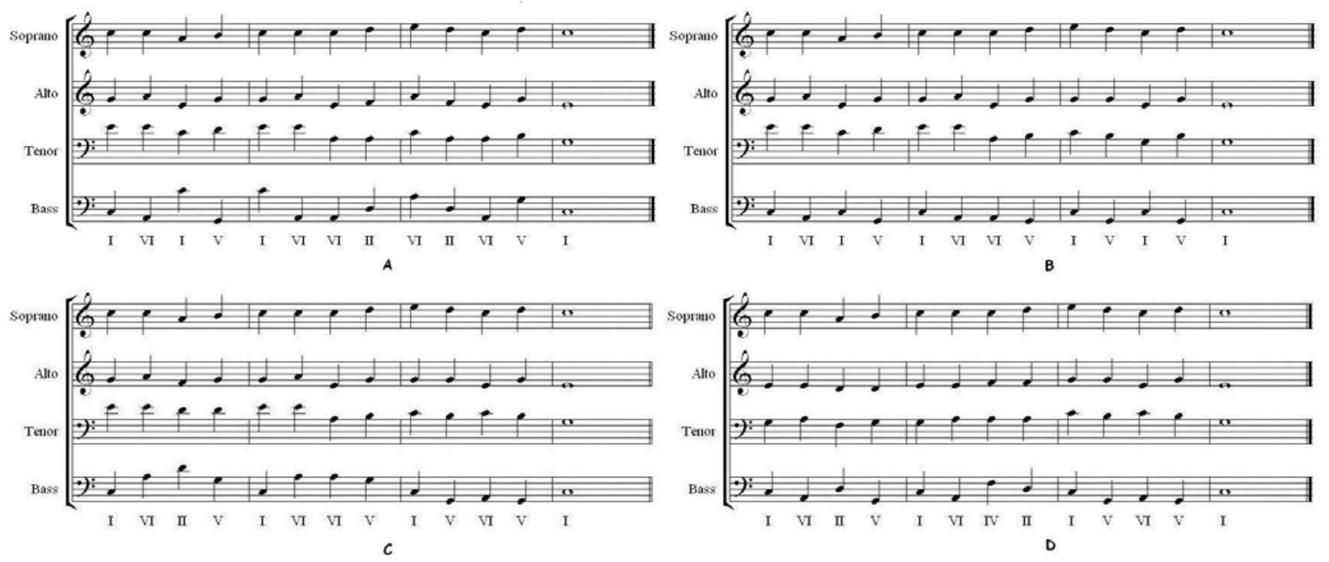
Soprano
Alto
Tenor
Bass
I VI I V I VI VI II VI II VI V I
A
I VI I V I VI VI V I V I V I
B
I VI II V I VI VI V I V VI V I
C
I VI II V I VI IV II I V VI V I
D

Figure 3: A representative harmonization exercise, showing four solutions: (A), (B), (C) show solutions with weights as proposed in Section 5. (A) has cost 3.14, (B) has cost 1.12 and (C) is the optimal solution with cost 0.65. (D) shows the optimal solution with weights $w_1 = 0.3, w_2 = 0.2, w_3 = 0.1, w_4 = 0.3$ and $w_5 = 0.1$. It has cost 0.47.

solutions with different costs. In Fig. 3A we show the solution with the highest cost (3.14). In this solution there are unacceptable distances in the bass. Furthermore, the melodies in the alto and tenor are not fluent, there is almost no contrary motion between soprano and bass and the grade sequence is bad.

Fig. 3B shows a solution with cost 1.12. This solution has a better bass line. The grade sequence is slightly better since we now have tension (grade V) in the second bar. This solution sometimes lacks contrary movement and there is no second grade used.

The optimal solution is shown in Fig. 3C. The grade sequence and grade frequencies are very well but there are many large distances between the bass notes. This can be explained by the weights we assigned to the costs: contrary motion is considered equally important as the distances between the notes. If we raise the weights corresponding to the distances between notes and we lower the weight corresponding to the use of contrary movement we obtain the result in Fig. 3D. Except from the fact that there is no tension in the second bar this is a good solution.

8 Practical Use And Further Work

The algorithm we presented is designed to solve harmonizing problems. It can be used as a didactic tool: teachers can use it to test quickly if problems have solutions, if these solutions are interesting enough and if they are not too difficult to solve. For instance, if a problem has only one solution where only grades I and IV are used, this problem cannot be solved well. Furthermore, students can use this program to compare their solutions to an optimal solution.

The model we presented can be extended in various ways. First of all, we can overcome the simplification of the harmonization problem by allowing modulations, chord inversions, non-chord tones, rhythm and ornaments, seventh chords and so on. This requires an extension of the domains and the introduction of new hard constraints. Furthermore, we can also introduce more soft constraints. For instance, some counterpoint rules can be expressed as soft constraints.

As illustrated in the previous section, the assignment of the weights is important. Fine-tuning of these parameters is needed and should be done more profoundly.

Acknowledgment

Chris Cornelis and Yvan Saeys thank the Research–Foundation Flanders for funding their research.

References

[1] C. Truchet and P. Codognet. Musical constraint satisfaction problems solved with adaptive search. *Soft Comput.*, 8(9):633–640, 2004.

[2] François Pachet and Pierre Roy. Musical harmonization with constraints: A survey. *Constraints*, 6(1):7–19, 2001.

[3] Kemal Ebcioğlu. An expert system for harmonizing chorales in the style of J. S. Bach. pages 294–333, 1992.

[4] Stéphanie Vanhove. Backtrackalgoritmen voor regelgebaseerde vierstemmige harmonisatie. Master's thesis, Ghent University, 2007.

[5] Somnuk Phon-amnuaisuk, Andrew Tuson, and Geraint Wiggins. *Evolving Musical Harmonisation*. 1999.

[6] Somnuk Phon-amnuaisuk and Geraint A. Wiggins. The four-part harmonisation problem: a comparison between genetic algorithms and a rule-based system. In *Proceedings of the AISB99 Symposium on Musical Creativity*, pages 28–34. AISB, 1999.

[7] Allan Forte. *Tonal Harmony in Concept and Practice*. 1962.

[8] Imogen Holst. *An ABC of music: a short practical guide to the basic essentials of rudiments, harmony, and form. With a foreword by Benjamin Britten*. 1963.

[9] Pedro Meseguer, Noureddine Bouhmala, Taoufik Bouzoubaa, Morten Irgens, and Martí Sánchez. Current approaches for solving over-constrained problems. *Constraints*, 8:9–39, 2003.