Harmonisation of Bach chorales

KBS project report

Martin Elmer Jørgensen, årskort 951174,
Søren Tjagvad Madsen, årskort 961701.
This report contains 31 pages.

Daimi, University of Aarhus
August 2002

Contents

1 Introduction

This project is an attempt to produce intelligent machine harmonisation of chorales. The goal is to harmonise melodies in the style of J.S.Bach; we wish to find tendencies and perhaps rules that Bach may have followed consciously or unconsciously.

We have chosen a GA approach to construct the new harmonisations. We discuss the use of rule-based evaluation and proceed to make experiments that are based on neural net fitness evaluation after training on a set of Bach chorales.

1.1 Harmonisation

Chorales are originally one voiced melodies from the German reformed church's singing tradition, started by Luther.

In the beginning it was very different kinds of melodies, which were used in church, but today the word chorale means a homophonic, often 4 part vocal movement. It is a harmonisation of the melody, with a new harmony for each new melody note.

In the music, there a lot of conventions have evolved about what sounds good and less good (for instance Bach's conventions), and these conventions can be collected as a set of "rules" for the music. The rules apply in different places and circumstances in the music, e.g. there are rules for voice leading and harmony.

Chorale harmonisation in general as in [1] is a generalisation of many years of development in western art music – a development which still changes, but is well studied by composers and music analysers. It can therefore be seen as a craft, and indeed as an introduction to the major-minor tonal music which has been developed in the western music tradition.

The chorales of Bach are much more complex than the simple style found in [1], but also Bach's techniques can be made rather concrete. Bach composed around 30 and reharmonised around 400 melodies, so we have quite a large material to search through.

Consequently, you could learn to make a well working harmonisation af a melody, by adding three notes to the soprano voice, and with a couple of extra tricks, you could make it sound like Bach.

So the task is to make that valid harmonisation, which meets the condidions on voiceleading and harmony. The hard part is to do both at the same time.

1.1.1 Complexity of the chorales

A chorale can be split into phrases of the melody. Each phrase is a musically coherent unit. On the phrase ends there is often a fermate [1] A phrase can start in one key and modulate to another (related) key. However, chorales almost always begin and end in the same key.

[1] An interpretation mark, for prolonging of the chord (unfortunately not written in the midi-files).

Bach uses a fixed harmonisation rythm, so each 1/4-beat has a new harmonisation (repetitions do occur). Each voice can eventually shift to another tone on these beats, and you hear the harmonic progression as a regularly shifting stream of chords.

Bach chorales are however more complex than described. The voices are often ornamented with "turning notes" and "going through" notes with smaller rhythmic values. These are mostly just ornamentations, but also affect the underlying harmonic progression. This affects our ability to make analysis on the chords directly on the 1/4 beats, since sometimes the real or intended harmony for that beat is displaced by 1/8. We try not to care about this in the beginning, and just hold on to harmonising the quaver beats. For an example of this problem read section 4 on preprosessing.

You can split the harmonisation ideals in two, those concerning voice leading, and those concerning harmony.

The voice leading rules ensure that the melodies are singable for the human voice. Moreover, they emphasize sounds we like, but also forbid certain interrelative movements of the voices, which sound bad.

For example we aim for countermovement in the outer voices, and we don't like the sound of a parallel fifth. Other rules are about dissonance treatment: If a dissonant tone is on the way, it should sound on a stressed beat, and be resolved downwards on the next unstressed beat. Furthermore the dissonant tone should be prepared in the chord before (in the same voice). This very old way of dissonance treatment (which Bach could have learned from Palestrina) consequently involves three consecutive chords.

Concerning harmony, there is also a strong tradition. We like to hear subdominant-dominant-tonic cadences – a strong harmonic connection, which is all over in our western music tradition. On the other hand it is most unlikely to find dominant-subdominant-tonic cadences in the music of Bach (but indispensible in blues).

The tonic is an expression for the triad[2] you can make from the key root. The dominant is the triad on the fifth scale step and the subdominant is the triad on the fourth. Major-minor tonal music consists roughly speaking of an alternation between triads on different scale steps.

By adding tones, which don't belong in the scale you can make triads that don't belong to the key. It can be used as a coloration, but is also an opportunity for moving to other keys. This is called a modulation. Modulations often end in related keys (keys whose scales have many scaletones in common). The order ef the chords is of great importance, since there is this logical connection between them.

Besides the soprano, which sings the melody, the bass is also an important voice, since the note of the triad in the bass is of great importance.

Chords can consist of 3 or 4 different notes, and it is possible to double notes from time to time. Rules for doubling exists. Root doubling of a tonic is very common, but you are never allowed to double the third in an dominant.

[2]A stack of three notes in third distance

The melodies are by far the most in major and minor keys, and since there are 12 possible beginning tones, we are dealing with chorales in 24 keys. There are in the litterature different rules for harmonising in major and minor, but the starting note (key) is not important. A triad has different meaning in different keys, so it is easier to use the expressions introduced above to describe the relations of the chords. The point is, that the musical events are not dependent on pitch. So to have all these informations could reduce the problem by a factor 12.

Unfortunately this is not a simple task to do. It can depend on many delicate factors in the music. [2] describes an algorithmic approach to study this problem.

2 Existing work on automatic harmonisation

Harmonisation has been studied from several angles.

[3] discusses how much expressive power is necessary to represent simple harmonic progressions and concludes that Markov models are sufficient to learn harmonic progressions, although other aspects of music require more intricate abstractions.

2.1 GAs in harmonisation

[4] and [5] are two articles about the same project, where a GA is used to harmonise melodies. The individual chromosomes are harmonisations, and fitness is evaluated according to basic rules - the sort of rules you would learn in an introductory course on chorale harmonisation. The idea is that a large part of music can actually be described by a relatively small number of rules.

Chromosomes are initialised with randomly generated chords with the correct note in the soprano voice. It is worth pointing out here that the chromosomes are initialised from the start, then, with real chords, i.e. major or minor triads, which are the only chords allowed.

Harmonisations are reproduced in several ways. There are the usual GA crossover and mutation operators (called *splice* and *perturb* respectively), but also musically meaningful reproduction operators are used: *Rechord* changes one chord to another; *PhraseStart* forces the chromosome to begin with a tonic on a stressed beat, and *PhraseEnd* forces the chromosome to end on a chord with the root note in the bass.

The harmonisations output by the GA have been rated by a music teacher as scoring between 30% and 50% according to the criteria used for 1st year undergraduate students' harmony. It turns out that no matter how long the evolution process is being run, the harmonisations produced are never flawless.

The authors conclude that the search space is very rough. For example, changing a single note in a harmonisation can break a lot of rules at once and deteriorate the quality of the harmonisation substantially.

Musically speaking, a non-specialised crossover operation is almost certainly doomed to failure. Two harmonisations may each be so well formed that making even a very minor change will ruin both. In other words, two good, partial harmonisations will only rarely combine to anything meaningful.

To sum up: ([5], p.6)

> . . . the problem is due to a multimodal fitness landscape, characterised by many local optima in basins of attraction which are separated by very high barriers [. . .]. Before the GA can move from one basin of attraction to another, multiple factors leading to a fitness penalty need to be changed. Such a simultaneous change is very unlikely to occur.

2.2 Expert systems for chorale harmonisation

In [6], a harmonisation GA is compared to a rule-based, or expert, harmonisation system. The expert system contains a knowledge base with the ideals of chorale harmonisation, and an inference engine. This is implemented as a constraint programming problem, using Prolog.

A number of comparative tests show that the expert system is clearly better. The larger contexts, such as harmonic progression, are best handled by the rule-based system, but also voice leading is more succesful.

Chorale harmonisation seems to be a problem well suited to algorithmic or constraint programming solutions, which is not surprising, since these techniques are closer to the way the craft of harmonisation is done.

2.3 Neural networks in chorale harmonisation

[7] and [8] present the use of "sequential neural nets for real-time harmonization", which means that the only information used for harmonising at time t is whatever happened up to time t and thus uses no global information on the continuation of the melody line. The sequential net includes e.g. a sub-net that interprets metric organisation.

[9] presents HARMONET, a neural net for harmonising chorales in the style of J.S.Bach, which is evaluated by "an audience of music professionals" as performing "on the level of an improvising organist". The problem is divided into subtasks:

2.3.1 Harmonic structure

The Bach chorales are abstracted to a series of quarterbeat harmonies. The network is trained using a sliding time window technique where it is shown, at each time step (or quarter beat position) t:

- The soprano or melody voice at times $t - 1$, t and $t + 1$
- The harmonies at times $t - 3$ up to t
- t's position relative to the beginning or end of a musical phrase

- Whether or not t is a stressed quarter beat

Nets with different window sizes are used in parallel and vote for which harmonic function should be chosen at each new time t.

[9] stresses the importance of the choice of pitch encoding. “A note s is represented as the set of harmonic functions that contain s”. Notes that are harmonically close are also close in this representation space, whereas notes that are neighbours by pitch are distributed into separate parts of the space.

2.3.2 Chord skeleton

Chosing the harmonic function gives the bass voice, so now the alto and tenor voices should be filled in. All possible positions that are consistent with the soprano voice and the chosen harmony. These possible chords are then evaluated according to standard harmonisation rules.

2.3.3 Ornamentations

At last, another net is trained to output the set of eighth notes by which a given chord may be augmented. Again, the input is a window including some of the surrounding context.

2.4 Summary and discussion

We have found no acceptably succesful examples of GA chorale harmonisation. The problem seems to be that it is very difficult to combine different parts of a solution. Even including musical knowledge in the reproduction operators, the GA didn't solve the problem satisfactorily.

How much domain knowledge should be encoded? David B. Fogel (see [10]) takes the extreme view that the machine should be allowed to learn by itself what is good and what is not; it should not be led away from the question by what we think it should know.

The GA approach has a weakness when it comes to harmonisation in a greater context. It has no means of controlling the harmonic progression. For example, it is impossible to know if the harmonisation will end in the same key as it started in, which is very desirable.

On the other hand, even though it is possible to find good results in chorale harmonisation using an algorithmic or rule-based approach, this is still a deterministic method and plain hard work to encode the knowledge or the algorithms to be used. In our view, it would be more interesting to have a system that may learn a way of harmonising from a set of examples.

A neural net could perhaps fulfill this wish, being trained on the Bach chorales and then in turn used to harmonise melodies on its own. The HARMONET project shows that this is possible. The hope is to find a balance where the net is able to produce good sequences of harmonies in the style of the training material but also to generalise to other good combinations of harmonies.

3 Our thoughts on automatic harmonisation

This section presents some initial thoughts on possible approaches to the problem.

3.1 GA approach with hardcoded Bach-weighted rules fitness evaluation.

Our first idea was to construct harmonisations using a genetic algorithm. Given a melody, the GA would search for a solution to fit it by constructing random harmonisations, evaluating them and recombining the better ones. As pointed out by [5], one GA problem is that harmonisation search spaces may have unrelated basins of local optima that are hard to escape.

Since there exists a standard set of harmonisation rules, it was natural to think of a rule-based fitness evaluation. But even though many of these rules are derived in part from the harmonisation practices of Bach, he also breaks some of the rules from time to time. If we had a complete set of harmonisation rules, we could learn from our Bach chorales which rules he is most prone to break and weight them accordingly.

As a small example, running through all the midi files and checking harmonic movements from one chord to another by way of two simple hand coded rules[3], we found that Bach breaks both rules from time to time.

```
Total number of chord pairs=20340
        Rule 0 broken 556 times
        Rule 1 broken 116 times
        Contrary outer movements: 7365
        Chord repetitions: 2214
```

Rule 0 checks for parallel movement of two voices that are positioned with an interval of 0, 7 or 12 semitones between them. Rule 1 similarly checks for hidden parallels in the outer voices. Following common harmonisation practice, contrary outer movement is an embellishment to be strived for.

Each voice follows a melody, or a “path in note space”, jumping up this many semi-tones and down that many semi-tones successively. We have counted the intervals jumped up and down by the soprano, alto, tenor and bass voices in the chords extracted from the 328 Bach chorales used.

Figure 1 shows tendencies in the voice making. The alto and tenor have quite similar graps, corresponding to the similar tasks they fulfill in the harmonisation. They are allowed to jump up to a fifth (7 half tones), (at least by [1] p. 19) but

[3] Rules forbidding parallels and hidden parallels

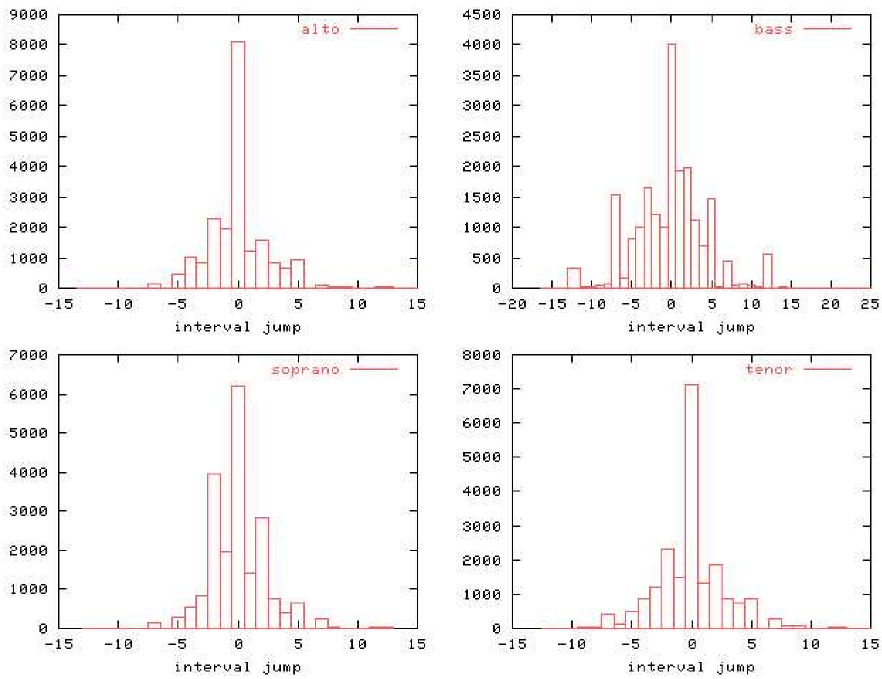

Figure 1: Voice interval jumps over 328 midi files with ChordFinder resolution=1.0 (quarter notes)

should strive to always go the shortest way (smallest jump) to the next note. The concentration of repeating and semi- and whole tone jumps shows this.

The bass is more "spread out". It is allowed to jump up to a fifth (7 semitones), but also a sixth and (8-9 semitones) an octave (12 semitones). Again the figure reflect this. About the soprano, one could say that stepwise and repeating is widely used in German singing traditon. (Remember, that a diatonic scale is made entirely from semitone and wholetone steps, so $-2, -1, 1$ and 2 all count as stepwise).

Please notice that the 6 semitones jump has almost completely been avoided. This is the tritone interval – the most dissonant interval from this period.

The greatest problem with the rule-based approach seems to be the lack of coded rules - we need to encode a knowledge base of harmonisation rules, which is a time-consuming process. Many of the more advanced rules require harmonic function analysis of the music in order to be applied. The basic analysis is not hard to construct, but a thorough and correct analysis is worth a study in itself.

We could have tried to learn a knowledge base of rules from the Bach chorales. Instead of learning logic rules, we decided to try a neural network.

3.2 EA approach with neural net fitness evaluation

Taking a point from the problems with GA harmonisation (see Section 2.1), we have narrowed our use of evolutionary methods down to an EA used as a search heuristic for individual chords. Evolving chords individually and sequentially from beginning to end of the melody avoids some of the recombination problems with GAs described above. Chords evolved this way are evaluated in conjunction with preceding chords by a neural network trained on the Bach chorales.

In [11], a comparison is made of neural net critics and handcoded rule-based critics for the genetic evolution of "musicians", i.e. small programs that produce a melody in response to a call melody. The conclusion is that the neural net is not sufficient in itself and benefits noticeably by making a joint evaluation of the programs together with the rule critic. Still, we could hope that a network recognising chord progressions may do a better job than a network evaluating the output of "musician" programs in a GA.

3.3 Neural network prediction of chords

Another possibility is to produce the harmonisations in the style of HARMONET, where the neural nets output harmonies on the basis of preceding chords an additional musical information.

4 Data and preprocessing

We chose to learn from a set of 440 midi files with J.S.Bach chorales that are available at `ftp://jsbchorales.net/sets/jsb403.zip`. This section describes the preprocessing done to create our current data set from these midi files.

4.1 Weeding

Unzipping the archive with the midi files, there are two files that exist in two copies. We included only the first copy of each. A first run through the remaining 438 midi files revealed that some of them had fewer or more than four voices (see Table 1); e.g. some files contained several copies of the soprano voice. We chose to weed out these and concentrate on the 348 files that had exactly four voices.

Another set of files were removed from the data set because they contained overlapping phrases, allowing possibly more than 4 simultaneous voices.

For the purpose of neural net training, we decided to transform the midi files into progressions of 4-voice chords. Here we found that some files contained chords with only 3 voices (i.e. one voice had a rest on such occasions). We call this an incomplete rest, since not all voices are resting at the same time. These files were weeded out because it is not clear what role an incomplete chord plays in a 4-voice harmonic progression. The files were a sort of extended version of chorales (rythmic variations) – too complex for our first experiments.

Number n of voices	**Number of n-voice files**
0	0
1	1
2	0
3	3
4	348
5	40
6	6
7	11
8	29
9	0
10	0

Table 1: Count of voices in files

After weeding, we were left with a set of 328 4-voice midi files. A complete list of the accepted files and of the files that were weeded out can be found in Appendix A.

4.2 jMusic

Instead of using Java's own midi library, we found a sound library called jMusic which is able to handle midi files in a standard music score notation. This means that once a midi file is loaded, the notes played within it may be handled abstractly as notes, half notes, quarter notes etc. disregarding performance related irregularities such as expressive timing and loudness. Certainly expressive performance of notated scores is as musically relevant as the composition of the score, but the Bach chorales are interesting composition works obeying a number of rules which may be studied purposefully without considering any performed interpretation of the works.

The jMusic java sound library can be found at:
`http://jmusic.ci.qut.edu.au/`

4.3 Extracting a data set

A four part chorale can be seen as a harmonic progression, i.e. a sequence of quarter note length four-voice harmonies. This of course is a simplification, since the four voices also play longer and shorter notes as ornamentations and rhythmic alterations to the fundamental quarter note progression. Moreover, the voices may have rests such that not all harmonies contain all four voices.

We have chosen to simplify the chorales by extracting only the harmonies, or chords, happening on quarter note beats. Thus ornamentations and rhythmic alterations are ignored completely. The resulting sequence of chords is a sequence of snapshots of what notes were sounding on the quarter note beats.

Figure 2: The 3 first measures of BWV 87, original and preprosessed with resolution quarter note.

The issue of rhythm in the individual voices, and their rhythmic interplay, could be studied separately or perhaps in some relation to our study of harmonic progressions. The ChordFinder java class that performs the chord extraction can be set to find chords with another resolution, e.g. extracting a sequence of all chords sounding on eigths og sixteenths, which is a higher resolution than the fundamental quarter note chord progression. Thus the (short-note, or high-resolution) ornamentations are also captured in the extracted sequence. But on the other hand, longer notes will appear several times in successive chords as the same note, since they will be sounding throughout several snapshots.

5 Neural net setup

We have used the SNNS 4.2 package for simulating neural networks. SNNS is available for download at

`http://www-ra.informatik.uni-tuebingen.de/SNNS`

5.1 Network layout and training

We have used simple, fully connected feed-forward networks with one input layer, one hidden layer and one output layer. Brief experiments with two hidden layers showed no serious improvement over the networks with only one hidden layer. The number of units in the input and output layers have varied according to the data representation as described in 5.2, and the number of hidden units has varied between 50 and 200, settling on 100. The nets were trained using backpropagation.

5.2 Data representation for the neural network

Finding a suitable data representation for the neural network turned out to be one long series of experiments. In a first group of approaches (representations 1-6, see below), we wanted the network to classify chord pairs or successions as good or bad. Inspired by [11], we trained the network with a set of positive instances and a corresponding set of negative instances generated from each of the positive examples in one of the following ways:

- mutating the positive example by stepping one or more randomly chosen voices a random number of semitones up or down
- mutating the positive example by interchanging two randomly chosen voices, repeating this a random number of times
- creating a completely random new sample

The positive instances had target output 1 and the negative instances had target output 0.

A second group of representations (7-8) focused on chord prediction instead of classifying chord pairs as good or bad.

5.2.1 Representation 1: Absolute soprano pitch, voice intervals

[8 number inputs, 1 output]
Sliding a window over the extracted chord progressions, we generated data as chord pairs (c1, c2). Each chord was represented as four numbers:

- the absolute midi pitch value of the soprano voice
- the interval between soprano and alto
- the interval between alto and tenor
- the interval between tenor and bass

These eight numbers (four in each of the two chords) were fed to the network along with 0/1 target values. Results were very disappointing.

5.2.2 Representation 2: Soprano pitch class, voice intervals

[8 number inputs, 1 output]
A second attempt was almost identical, except the soprano voice was represented modulo 12 to allow the network to recognise different notes with octave intervals as members of the same pitch class. This was no improvement at all.

5.2.3 Representation 3: Voice intervals only

[6 number inputs, 1 output]
As a third attempt, we omitted the pitch data and gave only information on the relative inter-voice intervals (soprano-alto, alto-tenor, tenor-bass). No improvement could be observed.

5.2.4 Representation 4: Unary pitch class

[96 boolean inputs, 1 output]
At this point we had to try something different. Changing to a unary pitch class representation helped a lot. Each of the eight voices in the chord pairs was encoded modulo 12 as 12 bits, 11 of which were 0, and the bit corresponding to the appropriate pitch class was 1. At this, the network actually began learning something.

5.2.5 Representation 5: Unary absolute pitch, trimmed (UAPT)

[190 boolean inputs, 1 output]
Feeling that we excluded too much information by modulating all pitches to pitch classes, we made a unary encoding of the absolute pitches of all notes. In principle, this gives 128 bits per note in a unary encoding. But since chorale voices fall in restricted intervals (there are bounds on what a human voice can sing), we could trim this representation to the number of possible notes for soprano, alto, tenor and bass respectively. Running through the Bach chorales, we found that the voices are restriced to:

- $soprano \in [60; 81]$
- $alto \in [53; 74]$
- $tenor \in [48; 69]$
- $bass \in [36; 64]$

where 60 represents the middle C. These bounds are inclusive, so we ended up with 95 bits per chord.

5.2.6 Representation 6: UAPT, larger windows

[95 bits/chord, 3-4 input chords, 1 output]
The “95 bits per chord”-representation seemed reasonably good. Inspired by the HARMONET encoding (see [9]), we enlarged the sliding window to include more chords. The neural net was thus trained to classify chord successions as good or bad.

5.2.7 Representation 7: Chord prediction

[95 bits/chord, 3 input chords, 1 output chord]
Keeping the unary encoding approach, we tried to predict the next chord based on the preceding chords. This gave rise to another question: how to interpret the output of the net as a chord? The neural nets with a single output between 0 and 1, we interpreted as whatever number was closest: 0 or 1. But the prediction networks yield a number between 0 and 1 *for every output unit*, i.e. 95 decimal numbers. A typical output can be seen in Table 2 and its corresponding target output in Table 3 .This example was taken during training.

0.03533	0.02832	0.06512	0.02282	0.02102	0.02887	0.12558	0.00614	0.05689	0.75492
0.08691	0.02489	0.2459	0.00099	0.0101	0.00133	0.06755	0.00343	0.00768	0.01585
0.02396	0.01485	0.02055	0.01877	0.03843	0.0151	0.04214	0.01931	0.0064	0.18088
0.02622	0.05187	0.3197	0.16376	0.02844	0.00362	0.12577	0.00469	0.06975	0.07287
0.00369	0.01607	0.01225	0.02186	0.02229	0.0146	0.02614	0.02202	0.0462	0.06793
0.04082	0.36754	0.00048	0.30422	0.0048	0.00905	0.11816	0.01513	0.04006	0.08143
0.03444	0.02393	0.0009	0.01417	0.02082	0.02127	0.00803	0.00793	0.07309	0.00784
0.04245	0.12278	0.01828	0.00421	0.03094	0.10847	0.04376	0.10241	0.02434	0.00444
0.06928	0.23163	0.0159	0.37061	0.01004	0.00389	0.00922	0.02299	0.03583	0.01292
0.02165	0.01267	0.01129	0.01519	0.02041					

Table 2: Example of typical prediction output

0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0					

Table 3: Target output for example

As can be seen, the 1’s at positions (10,1), (7,6), and (2,8) have a fairly high output value (which they should), while the 0’s at positions (2,6), (3,4), and (4,9) also have a high output value (which they should not).

The simplest way to interpret this output as a chord is to chose the unit with the highest output value for each voice predicted. In the above example, the first 22 values represent the 22 possible notes that the soprano might sing in this chord (from midi value 60 to 81, inclusive). Of these, 0.75492 at position (10,1) is the highest, thus a value of $60 + (10 - 1) = 69$ should be chosen for the soprano, which is an A. As discussed in 7.2.2, this interpretation could be subject to further development.

In the experiments, we have used a version of this representation where the soprano of the fourth chord was given in the input, and only alto, tenor and bass voices where predicted, since when harmonising a given melody, the soprano is given and cannot be changed.

5.2.8 Representation 8: Chord type analysis, prediction

[60 bits/chord, 3 input chords, 1 output chord]
In this representation, would like to incorporate harmonic analysis. Each chord gets analysed, so that the key, type (major, minor etc.) and which tone of the scale, each of the voices (not soprano) are singing. The scale is a 7 step diatonic scale with the steps: prime, second, third, fourth, fifth, sixth and seventh. The steps are implicitly defined from the chord type, ie. in a major scale the third is the 4th half note and in a minor scale it is the 3rd.

Each chord is then represented as follows: The soprano modulo 12 (12 nodes). Alto, tenor and bass are represented as sclalesteps (prime, second, third ..., 3×7 nodes) and a direction up or down (2 nodes for each voice) from the pitch in the last chord. The key is also represented as 12 nodes, and in our implementation, we so far use 9 different chord types. With the 9 chord types, we are able to analyse 88.733% of all chord we found in the Bach files.

Keep in mind, that the chord types, we are not able to analyse are a product of voiceleading. The sound of the chord makes perfectly sense in the Bach file, but since we have to chop the file into small bites we sometimes lose the overall picture.

Then on 3 chords in this encoding and the soprano modulo 12 in the fourh chord, the network has to learn to output the next chord type and key and which scalesteps the soprano alto and bass are going to sing, and also if they should jump up or down (or stay if possible) to that step.

The hope is that the neural network eventually will learn some connection between all these things: chord type, key, note doubling and voice leading.

6 Experimental results

This section describes some results obtained by generating harmonisations using the trained neural nets. Representations 1-4 showed so bad results just training the neural nets that we did not go any further with them. For the remaining representations, some results have been saved in midi files that are available from the web page:

`http://www.daimi.au.dk/~elmer/harm`

Please note that the generated midi files are slower than the original Bach chorales and also simpler because of the preprocessing. The tempo could have been adjusted to the original tempo, but it is easier to listen to the slower versions. To justly compare the generated harmonisations to the original ones, compare to the preprocessed Bach files instead of the original Bach files.

6.1 Representation 5: EA using NN fitness

6.1.1 Neural net training

First of all, a neural net was trained to recognise good chord pairs. The experiments shown here used a training set of 10000 patterns and a validation set of 5000 patterns, corresponding to approximately one fourth and one eighth of all available patterns after preprocessing the entire Bach collection to this representation.

The Bach chorale on which the midi files were generated was neither included in the training examples, nor in the validation examples. The net consisted of:

- 190 input units (2 chord input)
- 1 hidden layer with 100 units
- 1 boolean output

Figure 3 shows the mean square error of the training set (lower, black line) and the validation set (upper, red line). The harmonised midi files have been generated using nets saved after training 0, 1, 2, 3, 5, 10, 20... epochs etc.

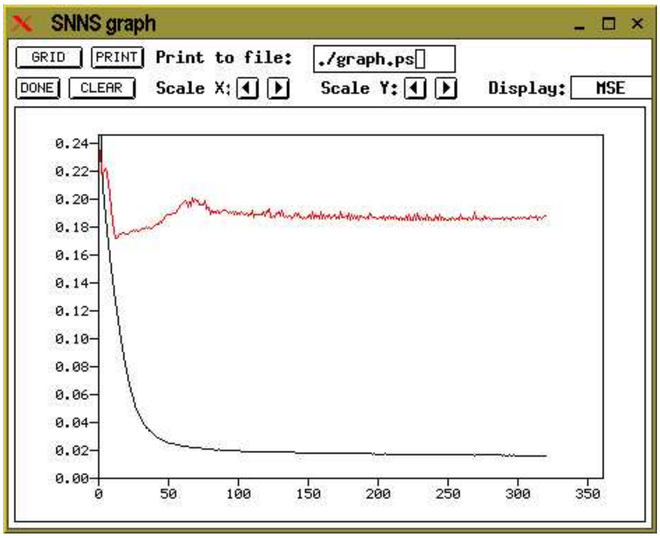

Figure 3: Mean square error training/validation.

6.1.2 EA setup

For each chord to be generated throughout a melody, an EA was used to search through the space of possible chords. Fitness evaluation of each candidate was done by feeding the pair (preceding chord, candidate chord) to the neural net and using the single output value in [0;1] as a fitness value, 0 for bad, 1 for good. As an example, see the evolution of the highest and mean fitnesses in the population in Figure 4.

- Population size: 100
- Generations: 20
- 80 % of next generation created by tournament selection
- 10 % of next generation created by crossover
- 10 % of next generation created by mutation

Crossover of two chords is done by selecting the four voices in the child at random from the two parents. This is a crossover that does not make much musical sense and thus, we think, produces strange jumps in the search space. The crossover operator in this setup therefore has the role of making wild new guesses to probe for new and interesting areas in the search space.

The mutation operator, on the other hand, adds a small positive or negative number δ of semitones to the value of one of the voices in the chord. At $\delta = 1$, this should produce a musically somewhat meaningful small alteration of the chord. At greater values of δ, it is more uncertain what effect this has.

Brief experiments with larger populations or longer runs (more generations) showed no compelling improvements on results. Generally the highest fitness settles on a number between 0.6 and 1.0, often ending above 0.95.

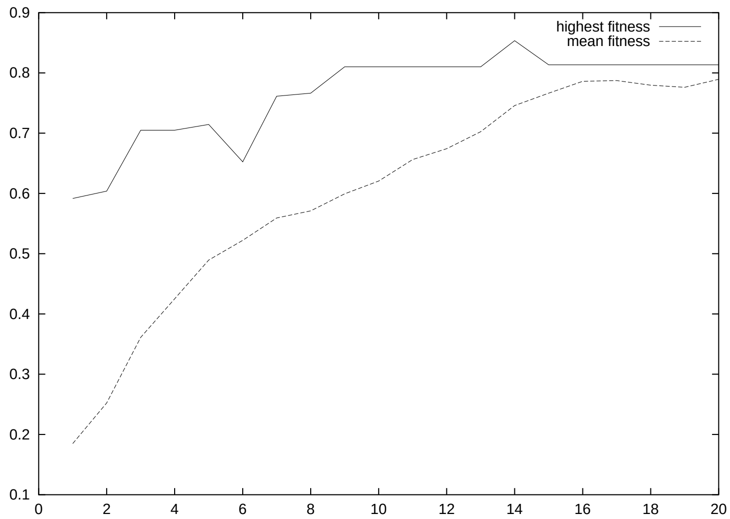

Figure 4: EA fitness evolution while searching for one chord

One could object that using an EA to search for the best chord successor is overkill, because the search space is not that large: $22 \times 22 \times 29 = 14036$ and we could have just searched through the entire space, making 14036 fitness evaluations per chord. With a population of 100 and running 20 generations, we have made only 2000 fitness evaluations for each chord, but more importantly, we would like the non-determinism of the EA in a harmonisation program.

6.1.3 Evaluation of the results

The web page presents different harmonisations of the file `008707b_.mid` after training for 0, 1, 2,... etc. epochs. Please consult the web page to hear the results. The following is an informal discussion of what we can hear from these results.

The untrained network not surprisingly gives an awful sounding random harmonisation.

After one epoch, we note a number of (unwanted) tritone jumps in the bass. The first chord is hard to guess, because the network only has an empty chord as the predecessor to guess from. The same chord is easier to guess at in the repetition of the initial melody.

After twenty epochs, the result seems to improve a little, but there are many places where two voices cross each other, to the effect e.g. that the alto jumps to a note higher than the soprano. Besides breaking a fundamental harmonisation rule, this makes it difficult to hear the melody, which we would normally think of as the highest note throughout the chorale.

After 40 epochs we tend to think there are a little fewer strange dissonances, but some chords that sounded alright earlier have now gone awry, e.g. in the repetition.

80 epochs: we still have a lot of voice crossings, although perhaps there is a little overall improvement in the choice of chords.

This tendency continues after 160 epochs; there are more true chords, but still, they don't fit together to form a cadence. This is perhaps the best harmonisation we have from this representation, and it is clearly unacceptable. After 320 epochs, we find that the number of good chords has decreased again, and the overall harmonisation is rubbish.

Consulting Figure 3, one should perhaps think that the best harmonisation would be made using the net saved at the minimum of the validation graph, say, around 10 epochs. Based on the above observations, we disagree with this. The early attempts, before 20 epochs, are simply too ugly in our ears.

The longer the neural net had been trained, the less likely the EA was to improve in best fitness. Rather, the best fitness immediately settled on a level that was either very close to 0 or very close to 1. This we take to mean that the overfitting of the net to our training set destroyed the smoothness of the fitness landscape and so deteriorated the EA's ability to gradually climb to other maxima.

6.2 Representation 6: EA using NN fitness

This set of experiments is set up much like the experiments with Representation 5. The changes are briefly listed below.

6.2.1 Neural net training

The neural net has twice as many inputs:

- 380 input units (4 chord input)
- 1 hidden layer with 100 units
- 1 boolean output

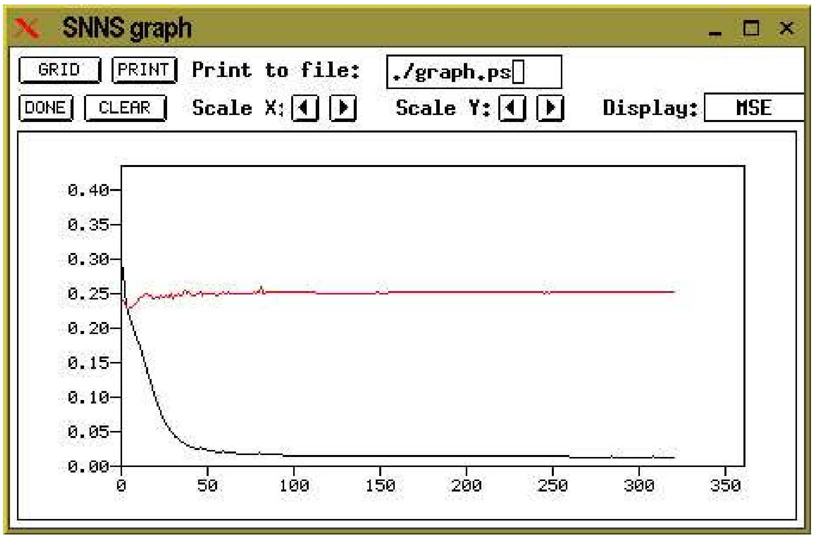

Figure 5: Mean square error training/validation.

6.2.2 EA setup

The EA seemed to have difficulties increasing the highest fitness, so we doubled the crossover and mutation parameters:

- Population size: 100
- Generations: 20
- 40 % of next generation created by tournament selection
- 30 % of next generation created by crossover
- 30 % of next generation created by mutation

6.2.3 Evaluation

Sadly, we don't find much to discuss in these results. If we'd had a dog, we would have spared it the bad experience of listening to these harmonisations. The repetition doesn't seem to get better than the first occurrence of the initial melody line, and we generally have difficulties even hearing the melody.

We obtained a slightly better, but still unacceptable, result by training a similar net for 1000 epochs on a training set of all 43304 available 4-chord successions extracted from the Bach chorales, including the five files harmonised afterwards (see and listen to the web page). The result on `008707b_.mid` is clearly better than those above. Our guess is that this is due not to the longer training but to the larger training set and the fact that the harmonised files themselves were included in the training set.

6.3 Representation 7: NN prediction

In these experiments, we have harmonised melodies from the Bach chorales using only the neural net to predict the next chord. As described in section 5.2.7, the output of the network can be interpreted in several ways. We have used the simplest, choosing for each note the output with the highest value.

6.3.1 Neural net training

The training set consisted of 20000 patterns and the validation set of 1254 patterns, totalling all 21254 patterns available after preprocessing all the Bach chorales.

- 307 input units (3 chord and soprano input)
- 1 hidden layer with 50 units
- 73 boolean output units (predicted alto, tenor and bass)

6.3.2 Evaluation

We shall resort to the same informal evaluation of the harmonisations of the file `008707b_.mid`. We have produced harmonisations on four other melodies, but the overall conclusion we think holds for all five.

After only one epoch, this net is able to produce a lot of true chords and some reasonable chord transitions, aspiring to real cadences. We were surprised at how well this harmonisation stays on the same chord when the melody stays on the same note. The chords are simple compared to those produced after 40-80 epochs, but at least they are chords. There is one clear problem, which appears also in the next harmonisations, after 17 seconds, where the melody jumps 3 semitones up. Let's call it the "up-3-problem".

After two epochs, we note more voice crossings. There are some good cadences, however, and these progressions proposed by the net are different from Bach's own. The up-3-problem remains.

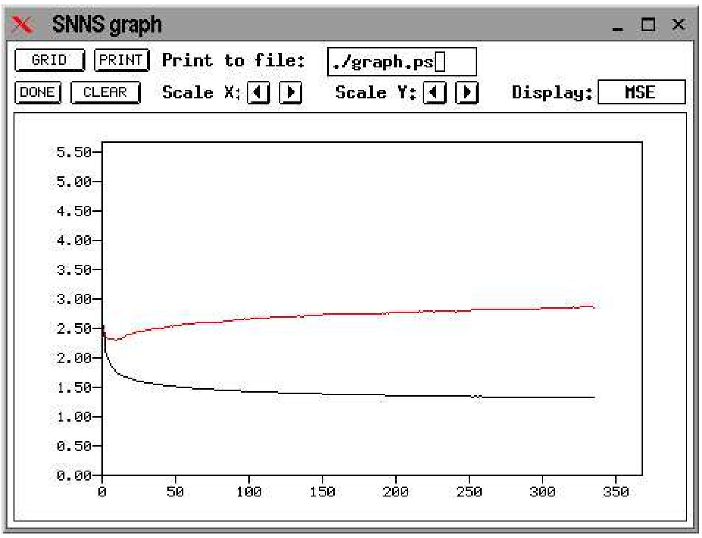

Figure 6: Mean square error training/validation.

After three and five epochs, the nets seem to lose some variation in the choice of chords for the first half of the `008707b_.mid` chorale. Epoch 10 has a nice long cadence in the end. Epoch 20 produced a good start but was generally worse because it introduced more strange or dissonant chords. The up-3-problem remains.

After 40 epochs, we observe to our satisfaction that the net has found some sort of solution to the up-3-problem. Unfortunately, it does not repeat the success in the repetition of the melody, where things get more messy. Note the nice resolved dissonance after 28 seconds. We will take a closer look:

Note the 4th, 5th and 6th chords. They make a well prepared dissonanse in the alto on the first beat of measure 8, and is resolved downwards on the next beat. Just as the literature describes. That is: the alto on d stays on d and is resolved to the db (which is really a c#) to form an A-major chord. It would have been much better to continue with a D-minor chord, but instead it chooses a Bb-major7. The two chords have 3 notes in common. A d in the bass could have done the difference.

On the first three chords, we however don't know what is going on. There is no musical connection: F-major, F-major with an added fourth in the bass (nonsense), C7 with the added seventh in the bass. A seventh in the bass should always continued downwards, but not this time. Instead we have an third doubled D-minor, which fits very well the next chords as described.

The program now tend to change chord more often, than after only a few training epochs.

The increase in complexity also holds true after 80 epochs. The up-3-problem is still solved on the first occurrence, but this harmonisation is generally messed up in the repetition. We still think of it, however as one of the best overall attempts with only a very few ugly non-chords.

After 160 epochs, we observe more voice crossings. It seems as though the net has become more bold, or daring, succeeding sometimes in finding new and intricate chord changes, improving also on the up-3-problem solution, but at other times it fails and produces some awful dissonances that don't fit anywhere.

The 320 epoch version is noticeably worse. There are problems with individual chord changes, but also, the earlier quite good connection with the melody seems to be lost to this harmonisation. The up-3-problem is not solved any longer.

Generally speaking, the neural nets trained on Representation 7 do not try to make chords which are far away from the general key, but sometimes add some unappropriate notes.

6.3.3 Generation with Bach input

On the web page is another set of files that were created by cheating. We generated them by predicting chords on the basis of Bach's own harmonisations. In other words, the three chords given as input to the network were the original Bach harmonies, and the output is thus free of accumulated error from prediction based on other error-prone predictions. The files generated in this way are better than the files generated on the basis of their own output, but of course, they have had more help than should be expected when harmonising a melody. Still, they give an idea of how good the network is when given the perfect input. Even then, it makes errors and some strange chords now and then.

6.4 Representation 8: analysis and prediction

The neural network has been trained to output the values 0 or 1, in each group of information (the a,t and b steps, up and down, key and chord type) we would like to recognise. For interpreting the result, we have chosen simply to take the output node in each group with the largest value.

So given the chord type, key, voicesteps and directions it is easy to make the next chord. However it was necessary to make sure that each voice stays in it's pitch range, so sometimes we have to overrule the up and down directions given.

As an example, the following output from the program shows the deciding process (the up and down is omitted):

```
making chord # 9
sop%12=9
aStep=8, with value: 0.7042
tStep=5, with value: 0.43065
bStep=8, with value: 0.95173
root=5, with value: 0.44757
chordType=0, with value: 0.81529
new chord is a: 5-dur
```

The chordType 0 is a major (3 notes), and the key should be 5 (an f). The soprano mod 12 is 9, that is the note a – a third in the f-major. The alto and bass are going to sing the key tone (8 is not the pitch here) which is f, and the tenor the fifth which is c. So totally that gives us a perfect keynote-doubled F-major.

Unfortunately, that is not always the case. In the next example we do not get a real chord back.

```
making chord # 10
sop%12=11
```

```
aStep=5, with value: 0.39717
tStep=5, with value: 0.40925
bStep=8, with value: 0.55927
root=5, with value: 0.27421
chordType=0, with value: 0.52679
new chord is a: ?
```

Soprano: h, alto: fifth in f major is a c, the same as the tenor, and the bass: f. that is alltogether not a chord found in major-minor tonality. We can not analyse it, and the network which have been trained on real chords must have problems with recognising and using it as input for making the next chord.

The music from this representation is not impressive at all. It is not below acceptable. We tried severel network structures, but the network does not seem to learn which chord types, keys and soprano notes that make sense.

To help the network we have tried to make a dataset which is made from chords in a row, which we everyhing about (that we are able to analyse in the way described). We removed everything else. In this way, we don't get wrong information from chords we are not able to analyse. This didn't seem to help either, but experiments are still going on.

This representation was supposed to be the "musical" representation, giving the neural network all the information a chorale maker would care about, so we expect this one to be able to do a little better with some changes.

The next representation will be with the key note and the soprano represented in relative steps to the previous chords. Then we are totally independent of the key, and the information the network will have to learn should be reduced by a factor 12, since all chord connections and voice behaviours are treated with no relations to pitch.

7 Discussion

The capabilities of the neural network have to be learned by experimenting. In the beginning we used very simple representations and hoped, that the network easily could see what was going on. That was not the case. The network has to be used in the right way.

A first improvement was to use more output nodes than one. The chord prediction in stead of chord evaluation led to the more proper use of the networks capabilities.

Our best results come from the network which was trained to recognise the chord progression copied directly from somewhere in the Bach chorales (Representation 7). The trained network then in a way represents some standard cadences in the Bach files, controlled by the melody. The network is quite successful in doing this. However we have no knowledge of whether the network have properly generalised Bach in the parts of the music, it has not been trained on. An algorithmic way of generating the chord progressions (for example a lookup in the files) is a more direct way to do what the network does. But this approach would not give a result if the chord progression is not found in the files. So our network of course does generalise – the question is the degree of success. The effect of training on Representation 7 is very audible, if one hears the midi files on the web page. There is a clear improvement after each of the first epochs of training.

In our representations, we still are not able to represent that two voices are singing the exact same note in pitch or the same note in octave(-s) distance. The network of course remembers it all as patterns. Another representation could therefore be to represent the chords as all notes possible to sing by the human voice (piano-like representation), and then simply let "1" indicate that a voice is singing this note and "2" if two voices are singing the same note.

7.1 Conclusion

Chorale harmonisation *is* a difficult problem.

We have not obtained any truly good results. If the HARMONET results may be compared to an improvising organist, our best Representation 7 results might be compared to a drunk improvising organist whose nose is itching all the time.

The harmonisation problem should be solved using some amount of domain knowledge. The GA approaches have had trouble because it is hard to encode the knowledge in GA operators that still permit a flexible search. Neural net critics as fitness evaluators are not good enough and fare better with the support of rule-based critics. The HARMONET project had successful results, but that setup included dividing training sets into musically relevant categories and also dividing the overall task into smaller problems that were musically separable. In this way the structuring of the experimental setup included some amount of domain knowledge implicitly. Our attempts to use a fairly unbiased neural network evaluator or predictor have had only moderate success, and the inclusion of harmonic analysis in the generation of training sets should improve the results considerably.

7.2 Room for improvement

7.2.1 Preprocessing and simplifying

As mentioned in 4.3, we have completely ignored the issue of rhythm in the individual voices, and their rhythmic interplay, which could be studied separately or perhaps in some relation to our study of harmonic progressions. Much of the tension of dissonances in the music are heavily dependent on the accentuations of the music. Therefore we cannot get real structure in our music. The melody has to make it all.

Bach has the habit of using the 1/8 notes also as stressed and unstressed, and he often uses 1/8 notes to resolve dissonances. To get a closer look on Bach, we must take into consideration the effect of the 1/8 notes. They form the harmonic progression in all details, and indeed the melodic lines for the voices. As mentioned our dataset is a simplification of Bach, since we don't get the in-between harmonics, so we don't expect our program not to make some mistakes!

7.2.2 Interpretation of prediction network output

This interpretation could be enhanced by using the chord analysis to evaluate a number of combinations of the most plausible notes (i.e. the outputs with the highest values).

7.2.3 EA improvements

We could have introduced musically meaningful or even smart crossover and mutation operators, e.g. mutating to a chord that is close by on the circle of fifths. But including musical knowledge in the operators also involves the danger of restricting search to certain areas of the search space.

A Midi files

A.1 List of four-voice files used for training

000206b_.mid
000306b_.mid
000408b_.mid
000507b_.mid
000606b_.mid
000907b_.mid
001306b_.mid
001405b_.mid
001606b_.mid
001707b_.mid
001805b_.mid
001805ba.mid
001907ch.mid
002007b_.mid
002011b_.mid
002406bs.mid
002506b_.mid
002806b_.mid
002908ch.mid
003206b_.mid
003604b2.mid
003706b_.mid
003806b_.mid
003907b_.mid
004003b_.mid
004006b_.mid
004008b_.mid
004311b_.mid
004407b_.mid
004606bs.mid
004705b_.mid
004803b_.mid
004807b_.mid
005505b_.mid
005708b_.mid
006005b_.mid
006206b_.mid
006402b_.mid
006408b_.mid
006502b_.mid
006507b_.mid
006606b_.mid
006704b_.mid
006707b_.mid
007007b_.mid
007011bc.mid
007305b_.mid
007408b_.mid
007706b_.mid
007807b_.mid
008008b_.mid
008107b_.mid
008305b_.mid
008405b_.mid
008506b_.mid
008606b_.mid
008707b_.mid
008807b_.mid
008906b_.mid
009005b_.mid
009106b_.mid
009209b_.mid
009307b_.mid
009408b_.mid
009606b_.mid
009906b_.mid
010207b_.mid
010306b_.mid
010406b_.mid
010806b_.mid
011007b_.mid
011308b_.mid
011407b_.mid
011606b_.mid
012206b_.mid
012306b_.mid
012506b_.mid
012606b_.mid
012705b_.mid
013306b_.mid
013506b_.mid
013906b_.mid
014007b_.mid
014403b_.mid
014406b_.mid
014500ba.mid
014505b_.mid

014608b_.mid
014806b_.mid
015105b_.mid
015301b_.mid
015305b_.mid
015309b_.mid
015403b_.mid
015408b_.mid
015505b_.mid
015606b_.mid
015705b_.mid
015804b_.mid
016206b_.mid
016406b_.mid
016606b_.mid
016806b_.mid
016907b_.mid
017405b_.mid
017606b_.mid
017807b_.mid
018007b_.mid
018305b_.mid
018400bx.mid
018707b_.mid
018806b_.mid
019406b_.mid
019406bg.mid
019412b_.mid
019705b_.mid
019707ba.mid
019710b_.mid
022602b_.mid
022701b_.mid
022711b_.mid
022902b_.mid
024403b_.mid
024410b_.mid
024415b_.mid
024425b_.mid
024432b_.mid
024437b_.mid
024440b_.mid
024444b_.mid
024446b_.mid
024454b_.mid
024462b_.mid
024511b_.mid
024515b_.mid
024517b_.mid
024522b_.mid
024526b_.mid
024528b_.mid
024537b_.mid
024540b_.mid
024812b2.mid
024823bs.mid
024833b3.mid
024842bs.mid
024846b5.mid
024853b5.mid
024859b6.mid
025200b_.mid
025300b_.mid
025400b_.mid
025500b_.mid
025600b_.mid
025700b_.mid
025800b_.mid
025900b_.mid
026000b_.mid
026100b_.mid
026200b_.mid
026300b_.mid
026400b_.mid
026500b_.mid
026600b_.mid
026800b_.mid
026900b_.mid
027000b_.mid
027100b_.mid
027200b_.mid
027300b_.mid
027400b_.mid
027500b_.mid
027600b_.mid
027700b_.mid
027800b_.mid
027900b_.mid
028000b_.mid
028100b_.mid
028300b_.mid
028400b_.mid

028500b_.mid
028600b_.mid
028700b_.mid
028800b_.mid
028900b_.mid
029000b_.mid
029100b_.mid
029200b_.mid
029300b_.mid
029400b_.mid
029500b_.mid
029600b_.mid
029700b_.mid
029800b_.mid
029900b_.mid
030000b_.mid
030100b_.mid
030200b_.mid
030300b_.mid
030500b_.mid
030600b_.mid
030700b_.mid
030800b_.mid
030900b_.mid
031000b_.mid
031100b_.mid
031200b_.mid
031300b_.mid
031400b_.mid
031500b_.mid
031600b_.mid
031700b_.mid
031800b_.mid
031900b_.mid
032000b_.mid
032100b_.mid
032200b_.mid
032300b_.mid
032400b_.mid
032500b_.mid
032600b_.mid
032700b_.mid
032800b_.mid
032900b_.mid
033000b_.mid
033100b_.mid
033200b_.mid
033300b_.mid
033400b_.mid
033500b_.mid
033600b_.mid
033700b_.mid
033800b_.mid
033900b_.mid
034000b_.mid
034100b_.mid
034200b_.mid
034300b_.mid
034400b_.mid
034500b_.mid
034600b_.mid
034700b_.mid
034800b_.mid
034900b_.mid
035000b_.mid
035100b_.mid
035200b_.mid
035300b_.mid
035400b_.mid
035500b_.mid
035600b_.mid
035700b_.mid
035800b_.mid
035900b_.mid
036000b_.mid
036100b_.mid
036200b_.mid
036300b_.mid
036400b_.mid
036500b_.mid
036600b_.mid
036700b_.mid
036900b_.mid
037000b_.mid
037100b_.mid
037200b_.mid
037300b_.mid
037400b_.mid
037500b_.mid
037600b_.mid
037800b_.mid
037900b_.mid

038000b_.mid
038100b_.mid
038200b_.mid
038300b_.mid
038400b_.mid
038500b_.mid
038700b_.mid
038800b_.mid
038900b_.mid
039000b_.mid
039100b_.mid
039200b_.mid
039300b_.mid
039400b_.mid
039500b_.mid
039600b_.mid
039700b_.mid
039800b_.mid
039900b_.mid
040000b_.mid
040100b_.mid
040200b_.mid
040300b_.mid
040400b_.mid
040500b_.mid
040600b_.mid
040700b_.mid
040800b_.mid
040900b_.mid
041000b_.mid
041100b_.mid
041200b_.mid
041300b_.mid
041400b_.mid
041500b_.mid
041600b_.mid
041700b_.mid
041800b_.mid
041900b_.mid
042000b_.mid
042100b_.mid
042200b_.mid
042300b_.mid
042400b_.mid
042500b_.mid
042600b_.mid
042700b_.mid
042800b_.mid
042900b_.mid
043000b_.mid
043100b_.mid
043200b_.mid
043300b_.mid
043400b_.mid
043500b_.mid
043600b_.mid
043800b_.mid

A.2 List of rejected files

The following files were weeded out.

A.2.1 Not four parts

000106b_.mid
000603b_.mid
000806b_.mid
001106b_.mid
001207b_.mid
001907b_.mid
002406b_.mid
002606b_.mid
002706b_.mid
002908b_.mid
003006b_.mid
003109b_.mid
003306b_.mid
003405b_.mid
004106b_.mid
004207b_.mid
004507b_.mid
004606b_.mid
005206b_.mid
005605b_.mid
005903b_.mid
006106b_.mid
006404b_.mid
006906b_.mid
006906ba.mid
007011b_.mid
007205b_.mid
007507b_.mid

007607b_.mid
007614b_.mid
007903b_.mid
007906b_.mid
009207b_.mid
009501b_.mid
009507b_.mid
009709b_.mid
009801b_.mid
010006b_.mid
010107b_.mid
010506b_.mid
010602ba.mid
010707b_.mid
011205b_.mid
011506b_.mid
011704b_.mid
012008ba.mid
012106b_.mid
012406b_.mid
012805b_.mid
012905b_.mid
013006b_.mid
013606b_.mid
013701b_.mid
013702b_.mid
013703b_.mid
013705b_.mid
013807b_.mid
014001b_.mid
014004b_.mid
014706b_.mid
014907b_.mid
015905b_.mid
016106b_.mid
017106b_.mid
017206b_.mid
017206ch.mid
017206vn.mid
017507b_.mid
017705b_.mid
017906b_.mid
018405b_.mid
018506b_.mid
019007b_.mid
019506b_.mid
022703b_.mid
024310b_.mid
024401bb.mid
024429bb.mid
024503b_.mid
024505b_.mid
024514b_.mid
024805b1.mid
024809b1.mid
024809bs.mid
024817b2.mid
024835b3.mid
024842b4.mid
024864b6.mid
025000b_.mid
025100b_.mid

A.2.2 Overlapping phrases

000707b_.mid
004106bs.mid
011106b_.mid
012006b_.mid
016506b_.mid
024417b_.mid
024828b3.mid
026700b_.mid
026700ba.mid
037700b_.mid
038600b_.mid

A.2.3 Incomplete rests

001007b_.mid
011909b_.mid
022707b_.mid
022709b_.mid
028200b_.mid
030400b_.mid
036800b_.mid
043700b_.mid

References

[1] Inge Svendsen. *Harmonisering – enkel dur/mol-koral.* systime, 1986.

[2] Heinrich Taube. Automatic tonal analysis: Toward the implementation of a music theory workbench. *Computer Music Journal*, Winter 1999.

[3] Bradley J. Clement. Learning harmonic progression using markov models.

[4] G. Wiggins, G. Papadopoulos, S. Phon-Amnuaisuk, and A. Tuson. Evolutionary methods for musical composition, 1998.

[5] Geraint Wiggins Somnuk Phon-Amnuaisuk, Andrew Tuson. Evolving musical harmonisation.

[6] Somnuk Phon-Amnuaisuk and Geraint A. Wiggins. The four-part harmonisation problem: A comparison between genetic algorithms and a rule-based system.

[7] D.Lehmann D.Gang and N.Wagner. Harmonizing melodies in real-time: the connectionist approach. In *Proceedings of the International Computer Music Conference*, Thessaloniki, 1997.

[8] D. Lehmann D. Gang and N. Wagner. Tuning neural network for harmonizing melodies in real-time. In *International Computer Music Conference*, Ann-Arbor, Michigan, 1998.

[9] Wolfram Menzel Hermann Hild, Johannes Feulner. Harmonet, a neural net for harmonizing chorales in the style of j. s. bach, 1992.

[10] David B. Fogel. *Blondie24: Playing at the Edge of AI.* Morgan Kaufmann Publishers, 2001.

[11] Lee Spector and Adam Alpern. Induction and recapitulation of deep musical structure. In *Proceedings of International Joint Conference on Artificial Intelligence, IJCAI'95 Workshop on Music and AI*, Montreal, Quebec, Canada, 20-25 1995.