

This article was downloaded by: [Aalborg University]

On: 11 May 2011

Access details: Access Details: [subscription number 912902580]

Publisher Routledge

Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK



Journal of New Music Research

Publication details, including instructions for authors and subscription information:

<http://www.informaworld.com/smpp/title~content=t713817838>

ChordNet: Learning and Producing Voice Leading with Neural Networks and Dynamic Programming

Dominik Hörnel^a

^a Software design and management AG München Germany,

Online publication date: 10 September 2010

To cite this Article Hörnel, Dominik(2004) 'ChordNet: Learning and Producing Voice Leading with Neural Networks and Dynamic Programming', Journal of New Music Research, 33: 4, 387 — 397

To link to this Article: DOI: 10.1080/0929821052000343859

URL: <http://dx.doi.org/10.1080/0929821052000343859>

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: <http://www.informaworld.com/terms-and-conditions-of-access.pdf>

This article may be used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

CHORDNET: Learning and Producing Voice Leading with Neural Networks and Dynamic Programming

Dominik Hörnel

Software design and management AG, München, Germany

Abstract

We introduce a novel method for learning voice leading using neural networks. Unlike earlier approaches for learning chord sequences where chords are predicted from a local context, this method uses a multi-layer neural network for learning chord assessment from music examples. The network employs a special constraint-based topology for transforming the relative comparison of chord pairs into an absolute assessment function. Using this chord assessment function, globally optimized chord sequences can be found in linear time using the dynamic programming technique. The CHORDNET implementation of the approach learns stylistic aspects of voice leading directly from a set of training examples. In combination with the HARMONET system, it presents a powerful framework for solving practical harmonization tasks.

1. Introduction

Voice leading is a well-known and well-explored domain in music theory. It is taught at music schools, and many voice-leading rules have been formulated in music theory books, e.g., in (Grabner, 1967). One of the reasons for its popularity is that voice leading is a rather well-defined task which is also applicable to other musical problems such as musical harmonization and counterpoint. Nevertheless, voice leading can be a difficult and time-consuming task where a solution must be chosen from a very large search space. In (McIntyre, 1994), a rough estimation resulted in a search space of size 29^{n*4} where n is the length of the melody.

With the upcoming of the computer, the idea to automate the voice leading process emerged. Since then, a large number of approaches to automatic music harmonization

have been proposed (e.g., Bellgard & Tsang, 1996; Berger & Gang, 1997; Ebcioglu, 1988; Feulner & Langnickel, 1998; Hild et al., 1991; McIntyre, 1994). The main technical differences between them can be categorized as follows:

- whether the approach is style-independent (e.g., the ChoralHarmonizer system in Feulner and Langnickel, 1998) or style-dependent (e.g., Ebcioglu, 1988)
- whether it is rule-based (Ebcioglu, 1988; McIntyre, 1994) or learning-based (Hild et al., 1991; Bellgard & Tsang, 1996)
- whether it solves the problem by chord prediction (Berger & Gang, 1997) or by chord assessment (McIntyre, 1994).

The style-independent approaches concentrate on general aspects of voice leading, whereas style-dependent approaches model a specific style. Rule-based methods are more appropriate for observing typical voice-leading rules like the parallel fifths rule, but cannot easily capture stylistic properties given in the examples of a specific style under consideration. Prediction-based learning methods like Markov models and neural networks have been successful in dealing with complex musical tasks that are intuitively solved by humans, e.g., music harmonization, melody variation (Hörnel & Menzel, 1998) and melody completion (Hörnel & Höthker, 1998). The results have been weaker for tasks which are more strongly governed by rules, e.g., voice leading and musical counterpoint. For example, it is not straightforward what should be done if a chord predicted by a neural network does not conform to a chord rule. In Bellgard and Tsang (1996), an error-control encoding is used for the Boltzmann machine to encourage syntactically correct chords. In this way, voice leading problems can be reduced but not completely avoided, as shown by the authors.

Accepted: 23 May, 2003

Correspondence: Dominik Hörnel, Software design and management AG, Thomas-Dehler-Str. 27, D-81737 München, Germany. Tel.: +49 89 63812-657; E-mail: dominik.hoernel@sdm.de

Instead of directly predicting a musical structure like a chord, it may therefore be more suitable to define an assessment function. In a rule-based system, the assessment values have to be fitted manually. In a learning-based system, they can be based on the predictions by defining a distance measure between selected and predicted structure. For example, in Hörnel and Höthker (1998) neural networks are used within an evolutionary framework to define a fitness function for assessing the musical structure of folksong melodies. In the learning phase, one neural network is trained to predict the next melody note. During the evolutionary phase, the activations of the network's output neurons are compared with the target values defined by the selected melody note. The distance between selected melody and network predictions is then used to define an appropriate fitness function for the evolution of folksong-style melodies. In this case, the Euclidean distance was chosen as a distance measure. See Hörnel and Höthker (1998) for details.

The CHORDNET system presented here is able to model a style given by a set of examples, and to observe at the same time general voice-leading rules. It combines rule-based filtering and learning-based chord assessment. The assessment function is automatically learned from the training examples. An efficient dynamic programming approach is used to obtain a globally coherent solution. In order to significantly reduce the search space, the underlying harmonies are determined before computing the chord sequence.

2. Task definition

The *harmonization* task is usually defined as follows: For each note of a given soprano melody, complete the chord by filling in a note for each accompanying voice. In the case of a *four-part (chorale) harmonization*, the accompanying voices are an alto, a tenor, and a bass voice. Sometimes the notion of harmonization also includes the addition of ornamentations to the resulting chord sequence (Hörnel & Menzel, 1998). In the present work, ornamentations will not be considered by CHORDNET, but the HARMONET system (Hild et al., 1991) adds eighth notes to the voice leadings produced by CHORDNET.

In tonal music, we can abstract from concrete notes by clustering chords into *harmonies* or *harmonic functions*, e.g., T, D, S, Tp, ... respectively I, V, IV, VI, ... in Roman Numeral notation. Therefore the harmonization task can be divided into two subtasks (see Hörnel & Menzel, 1998): The first subtask is to determine the so-called harmonic skeleton conceiving chords as harmonies. The second subtask is to expand these harmonies to chords. The decomposition of the harmonization process into these subtasks substantially reduces the search space. In HARMONET, the first subtask is learned by a committee of neural networks, predicting the next harmony given information about harmonies at some former times, and the melody from time $t - 1$ to time $t + 1$, learned from examples of a given harmonization style (e.g., the Bach chorale style). The second subtask is solved by a

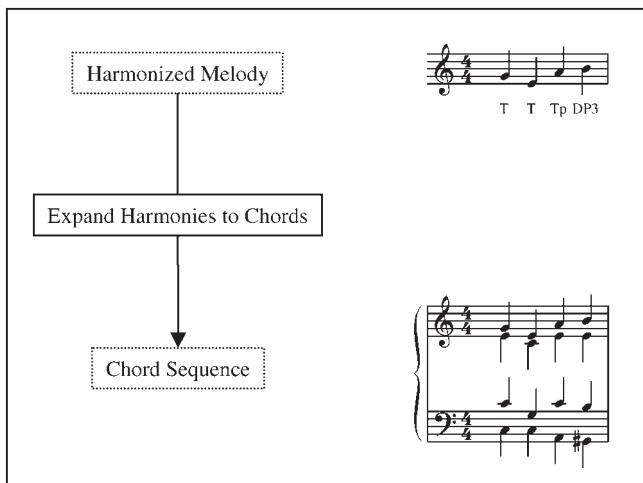


Fig. 1. Voice Leading Task (example from Bach chorale).

symbolic algorithm which selects chords fulfilling general **style-independent** voice-leading rules.

The CHORDNET system creates a solution to the second subtask which is globally coherent and in aesthetic conformance to a given set of training examples of a composer. It solves the following *voice leading task*: Given a soprano melody and a corresponding harmonic skeleton, find a four-part chord sequence that conforms to a given style. For both melody and harmony, a quarter beat rhythm is assumed. If there are two or more melody notes for a quarter beat, the first note is selected. Figure 1 shows an example.

3. The CHORDNET structure

The CHORDNET system solves the voice leading task in four steps (see Fig. 2):

1. For each harmonized melody note, all possible chords that match both melody and harmony are generated recursively.
2. The resulting chord list is filtered according to style-independent voice-leading rules. A rule is accepted as a filtering rule if it is observed in more than 95 percent of the cases, independent of the style under consideration. A filtering rule should be always fulfilled. An example is the *parallel motion rule* which will be described later.
3. The chords of the filtered chord list are then assessed using the neural network described in the next section. The result of chord filtering/assessment is a sequence of chord lists. For clarity, only the chord with the highest assessment value of each chord list is displayed in the "chord list sequence" music example of Figure 2. The resulting chord sequence is said to be locally optimal.
4. After the chord lists have been computed and assessed for all melody notes, a dynamic programming algorithm determines the chord sequence for which the sum of assessment values is globally optimal.

The important subtasks which make the novelty of the approach are solved in step 3 and 4. They are described in

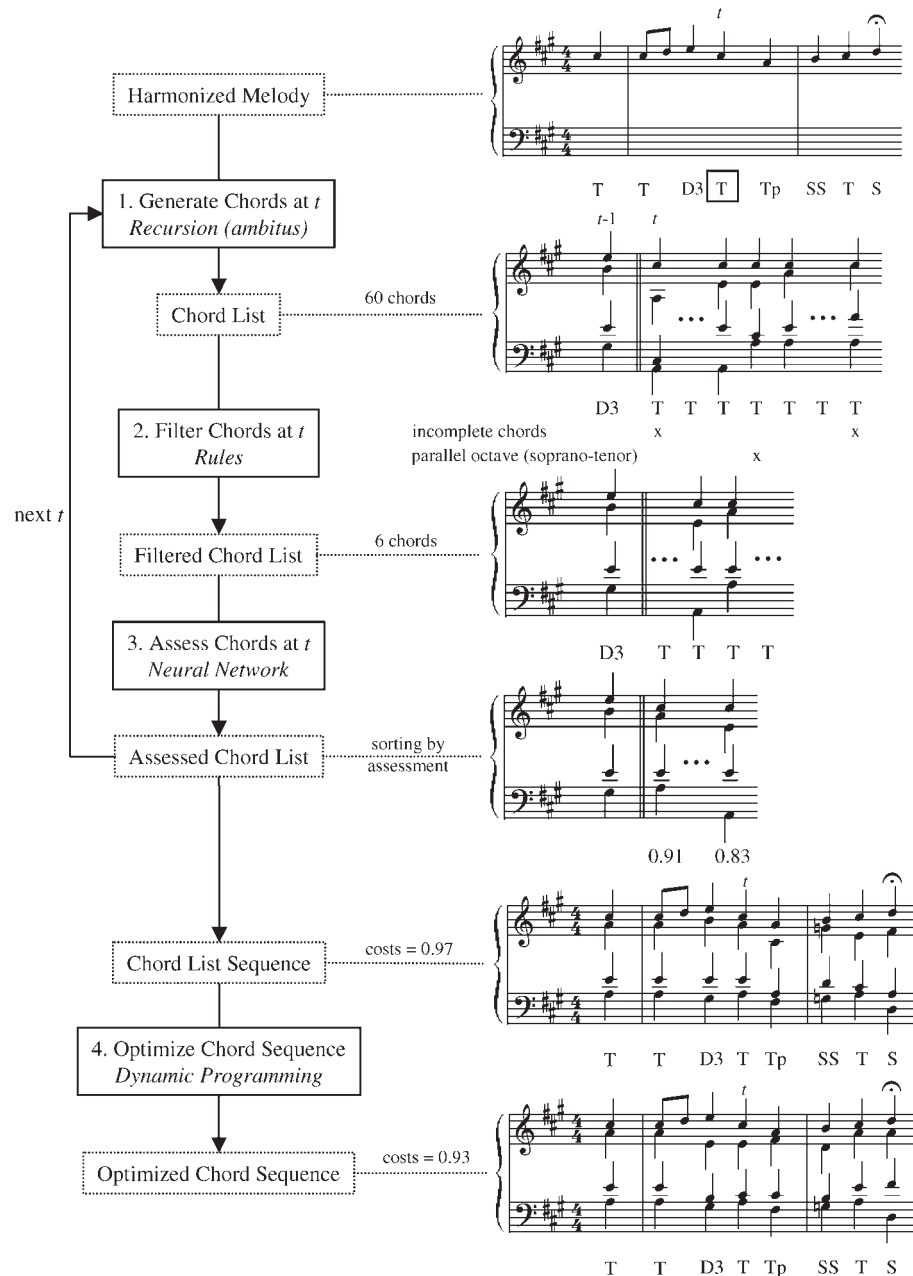


Fig. 2. CHORDNET structure. For each melody note, a list of chords is generated, filtered by rules, and assessed by a neural network. The result of chord filtering/assessment is a sequence of chord lists. For clarity, only the chord with the highest assessment value of each chord list is displayed. Afterwards, a dynamic programming algorithm selects the chord sequence which is globally optimal.

detail in the following sections. The notes for generating chords in step 1 are selected from an ambitus which defines the pitch range for each voice by the style under consideration. Recursion is used to compute all possible combinations of chord notes.

4. Learning chord assessment

Once all possible chords have been computed for a given melody note and harmony, they have to be filtered and

assessed. In HARMONET (Hörnel & Menzel, 1998), four types of voice-leading rules are distinguished for this purpose:

- chord structure filtering rules (e.g., the Completeness Rule)
- chord structure preference rules (e.g., the Close Position Rule)
- chord transition filtering rules (e.g., the Parallel Motion Rule)
- chord transition preference rules (e.g., the Neutrality Rule)

Completeness Rule:	A chord is complete iff it contains all triad tones (fundamental tone, third and fifth) of its underlying harmony. All incomplete chords are filtered. Example: Given the tonic harmony in C major, the chord (c,e,c,e) will be filtered because the fifth g is missing.
Close Position Rule:	The position of a chord is measured by the interval distance between soprano and bass note. Three types of positions are distinguished: close position (≤ 1 octave), neutral position (between 1 and 1,5 octave), open position ($>1,5$ octave). In four-part music, open position is generally preferred to close position. Example: The chord (S,A,T,B) = (c1,g0,e0,c0) is in close position, the chord (e2,g1,c1,c0) is in open position.
Parallel Motion Rule:	Parallel motion in fifths or octaves of two voices is forbidden. Example: The chord transition from (c2,e1,g0,c0) to (a1,f1,c1,f0) is forbidden because there is a parallel fifth between tenor and bass voice.
Neutrality Rule:	The neutrality value of a chord is measured by summing up the contour values (-1, 0 or 1) from the previous chord for each voice and computing the absolute value. Usually, large neutrality values are avoided. Example: The neutrality value of the chord transition from (c2,a1,f1,f0) to (b1,g1,d1,g0) is $\text{abs}(-1-1-1+1) = 2$.

Fig. 3. Examples of chord structure/transition filtering/preference rules.

Examples of chord structure/transition filtering/preference rules are displayed in Figure 3.

The chord rules filter/assess a single chord, the chord transition rules filter/assess a transition between two neighboring chords. In HARMONET, the assessment values for the preference rules (e.g., the values of a chord in close, open and neutral position for the *close position rule*) are determined manually based on a set of examples. The assessment value of a chord is then determined by summing up the values over all preference rules.

The manual fitting of the assessment values is a very difficult and time-consuming process. For every new style, the values must be re-estimated. Furthermore, the summing up of the assessment values assumes a linear relationship between the criteria on which the preference rules are built. In order to learn an appropriate assessment function, one could associate an assessment value with every chord. But how can this value be determined since from the training examples, we only know the one chord that was selected by the composer in a specific situation?

A potential alternative is to learn relative assessment instead of absolute values. Given two chords, the task is to decide which is the more appropriate one. In a specific situation, the chord selected by the composer is then preferred to all other chords that have not been selected. In a neural network setting, an appropriate network structure for learning relative judgement could be the following (see Fig. 4).

The input neurons represent the two chords to be compared. The output neuron should be 0 if chord 1 is preferred to chord 2, 1 otherwise.

This approach has several shortcomings:

- In the test phase, once the network has been trained, one chord must be chosen from a set of possible chords. This requires a chord selection strategy because the network has learned only relative judgement and can not directly find

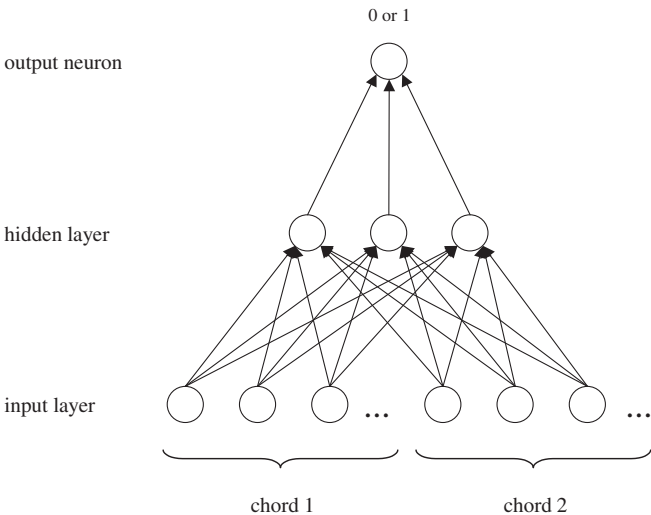


Fig. 4. Neural network for learning relative chord assessment.

out which one is the best chord. When choosing a survival strategy in which the chords outperform each other, the outcome of this strategy will also depend on the order in which the chords are compared.

- By doubling the input layer, the number of weights significantly increases and impairs the generalization of the network.
- A large number of training examples has to be generated. For each melody note, the chord selected by the composer is compared to all other possible chords (the average number of melody notes in Bach chorales is about 50, an estimation of the number of possible chords per melody note will be presented in the evaluation section). Additionally, the network does not only have to learn the information that chord 1 is better than chord 2, but also that chord 2 is worse than chord 1. This is done by swapping

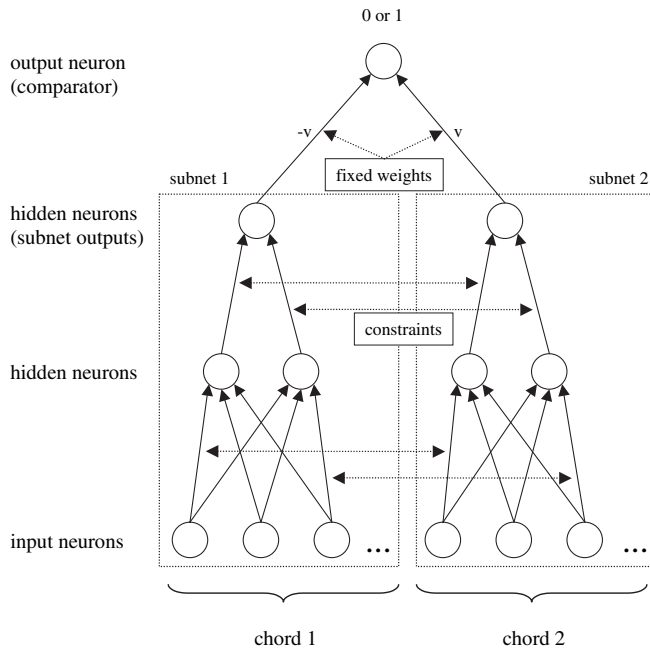


Fig. 5. ChordNet network topology for learning absolute chord assessment.

chord 1 and chord 2 in the input layer and learning 0 instead of 1 in the output neuron. Taking into account these considerations, the number of training patterns is given by

$$\# patterns = 2 \cdot (\# possible_chords(t) - 1) \cdot \# melody_notes \quad (1)$$

The CHORDNET approach avoids some of these shortcomings. Its basic idea is to transform relative comparison of chord pairs into an absolute assessment function. For this purpose, a neural network with a special constraint-based topology is constructed. This topology was proposed in Braun et al. (1991) where a similar approach is used to learn an assessment function for the game of Nine-Men's Morris. Figure 5 sketches the network structure.

The CHORDNET network is composed of two identical subnets, one for each chord. The output neuron of each subnet is connected to an additional neuron which computes the output of the CHORDNET network and acts as a comparator: The threshold (bias) of the output neuron is fixed at 0, the weights of the two incoming connections have the same size v and opposite sign. They amplify the signals of the subnet outputs by factor $v > 1$. All hidden and output neurons of the CHORDNET network use the logistic activation function

$$f_{\log}(x) = \frac{1}{1 + e^{-x}}.$$

If the output of the first subnet is greater

than the output of the second subnet, the output neuron will produce a value near 0; if the output of the first network is smaller than the output of the second subnet, it will produce a value near 1. The value of v is fixed, not learned. It guarantees that even small differences between the assessments of the two chords produces output values near 0 or 1. (Braun

et al., 1991) have shown that learning is very robust respective the choice of v but not insensitive. It should not be too small or too big. For the game of Nine-Men's Morris, the backpropagation algorithm got stuck in a local minimum for $v = 1$ and $v = 1000$. Good results were obtained for $v \in [30, 300]$.

During the learning phase, all connections of the two subnets are restricted in pairs by constraints: Each weight of subnet 1 always has the same value as its corresponding weight in subnet 2. When using gradient descent training methods like backpropagation, this can be achieved by summing up the weight deltas of both weights before computing the weight update (see Rumelhart & McClelland, 1988). Thus the learning goal is twofold: The weights are modified to achieve correct judgement of the chord pairs, and the weight constraints force the network to produce identical weights for both subnets.

After learning, the output neuron of both subnets will produce a value within the interval $[0, 1]$ for each chord. This value reflects an assessment of a given chord since learning has arranged it on a linear scale relative to other chords. Therefore, one of the (identical) subnets can be used for chord assessment. Mathematically speaking, a partial ordering given by relative chord comparison has been transformed into a total ordering defined by a chord assessment function.

The CHORDNET approach to learning chord assessment has several advantages:

- No chord selection strategy is necessary. The best chord is the one which produces the highest subnet output.
- The number of different weights for learning is reduced by factor of 4 compared to the relative judgement approach, thereby avoiding overfitting, if the number of hidden neurons of the relative judgement network is equated with the number of hidden neurons of the CHORDNET network. During the test phase, the performance of the network is significantly faster because only one subnet is used.
- Learning two chords with target 1 is equivalent to exchanging the chords and learning a target of 0 (Ullrich, 1991). Therefore the network only has to learn that chord 1 is better than chord 2. Using this result, the number of training examples can be halved.
- Due to the stochastic nature of neural networks, the approach can also deal with conflicting patterns. These conflicting patterns arise because a composer usually has the option to select one of several musically sensible chords.
- The assessment values can be used to define a cost function for global optimization. This will be shown in the next section.

As mentioned above, all hidden and output neurons use the logistic activation function. For training the RPROP learning algorithm (Riedmiller & Braun, 1993) was used. The basic principle of RPROP is to eliminate the harmful influence of the size of the partial derivative on the weight step in backpropagation. As a consequence, only the sign of the deriva-

tive is considered to indicate the **direction** of the weight update. The **size** of the weight change is exclusively determined by a weight-specific, so-called *update value* Δ_{ij} :

$$w_{ij}^{new} := w_{ij}^{old} \begin{cases} -\Delta_{ij} & \text{if } \frac{\partial E}{\partial w_{ij}} > 0 \\ +\Delta_{ij} & \text{if } \frac{\partial E}{\partial w_{ij}} < 0 \\ -0 & \text{otherwise} \end{cases} \quad (2)$$

where $\frac{\partial E}{\partial w_{ij}}$ denotes the summed gradient information over all patterns of the pattern set. In this way, RPROP achieves much faster convergence than backpropagation.

The RPROP learning rule was modified in order to be able to deal with weight constraints. This can be achieved by summing up the weight deltas $\frac{\partial E}{\partial w_{ij}}$ for each weight w_{ij} in one subnet and its corresponding weight $w_{i'j'}$ in the other subnet. Then the modified weight update is

$$w_{ij}^{new} := w_{ij}^{old} \begin{cases} -\Delta_{ij} & \text{if } \frac{\partial E}{\partial w_{ij}} + \frac{\partial E}{\partial w_{i'j'}} > 0 \\ +\Delta_{ij} & \text{if } \frac{\partial E}{\partial w_{ij}} + \frac{\partial E}{\partial w_{i'j'}} < 0 \\ -0 & \text{otherwise} \end{cases} \quad (3)$$

The weights of the connections from the subnet outputs to the output neuron are never modified. For the experiments, the amplification value v was set to 100. Full shortcut connections from the subnet inputs to the subnet output neuron were added (not shown in Fig. 5) such that the networks tend not to require too many hidden neurons.

A special chord representation is used to obtain a good generalization of the network. A chord is not represented by its notes, but by a binary encoding of the features used to express the preference rules. Both information about chord structure (given by structure preference rules) and information about the relationship of a chord to its predecessor (given by transition preference rules) are given to the network.

Let us consider the “Close Position Rule” and the “Neutrality Rule” displayed in Figure 3 as examples. The “Close Position Rule” distinguishes three categories of chord positions (close, neutral and open). Since these chord positions are ordered, they are best modeled by an ordinal variable with the range $\{0, 1, 2\}$. An appropriate binary coding uses two bits: $\{“00”, “10”, “11”\}$. The neutrality value defined in the “Neutrality Rule” can be modeled by an ordinal variable with the range $\{0, 1, 2, 3, 4\}$. An appropriate binary coding uses four bits: $\{“0000”, “1000”, “1100”, “1110”, “1111”\}$.

Similar codings can be defined for all preference rules. Table 1 summarizes all preference rules and how they are encoded in the CHORDNET network. The first four rules assess the structure of current chord, the other four rules

assess the transition from previous to current chord. If a chord under consideration has no predecessor because it occurs at the beginning of a chord sequence, all bits representing transition preference rules are set to zero. Afterwards the binary strings for the two chords to be compared are concatenated to form an input pattern for the network. The resulting input size is $2 * 43 = 86$. The output pattern is 0 or 1, dependent on whether the first or the second chord was chosen in the training example.

5. Optimizing chord sequences

Once the network has been trained, it can be used to select the best chord according to the assessment value computed by the CHORDNET subnet. This decision strategy can be suboptimal, however. Sometimes, after the selection of the (locally) best chord, it is not possible to find a musically sensible continuation. One could even run into dead ends where all chords are filtered from the chord list. To avoid this, one can give a very low assessment value to chords that break a filtering rule instead of filtering them from the list. This will not prevent the breaking of the rule, however. It would have been better to choose a suboptimal chord in order to find a better continuation afterwards.

One possible solution to avoid dead ends is to use backtracking if a musically sensible continuation is impossible. However, a much more efficient and elegant solution to this problem can be found by using the dynamic programming technique (Bellman & Dreyfus, 1962).

Let A be a chord, A_t the chord list computed at position t . As mentioned in the previous section, a chord is assessed by

- its *structure* which can be measured by a cost function $c_1: A_t \rightarrow [0, 1]$
- the *transition* from the previous chord measured by $c_2: A_{t-1} \times A_t \rightarrow [0, 1]$

Then the total costs of a chord are

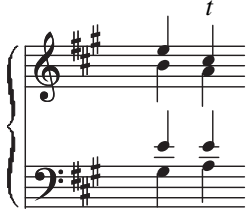
$$c(B, A) = c_1(A) + c_2(B, A), \quad A \in A_t, B \in A_{t-1} \quad (4)$$

This cost value can be computed as 1 minus the assessment value of the CHORDNET subnet which encodes both information about the structure of a chord A and the relationship to its predecessor B . Choosing the chord with minimal costs is only locally optimal. A globally optimal chord sequence can be found by formulating the chord sequence optimization problem as a multistage graph problem and applying the *Principle of Optimality*:

“An optimal sequence of decisions has the property that whatever the initial state and decision are, the remaining decisions must constitute an optimal decision sequence with regard to the state resulting from the first decision” (Horowitz & Sahni, 1978).

The *multistage graph problem* is a standard problem in computer science. Given a weighted multistage graph, the task is to find a minimum cost path through the graph. A multistage or k -stage graph is defined as follows:

Table 1. Chord structure/transition preference rules and their encodings.

Preference Rule	Short Description	Encoding	Example
			
Dissonance Rule	Assesses whether a harmonic dissonance (e.g., sixth, seventh) appears in the chord	1 if a harmonic dissonance appears, 0 otherwise	0
Tone Doubling Rule	Assesses which tone appears how often in the chord	4 * 2 Bits (number of appearances) for fundamental tone, third, fifth and dissonance	1 1 1 0 1 0 0 0
Unisono Rule	Assesses whether two chord notes are identical	1 if unisono appears, 0 otherwise	0
Close Position Rule	Assesses the interval distance between soprano and bass note	2 Bits (close, neutral and open position)	1 0
Dissonance Resolution Rule	Assesses how dissonances in the previous chord are resolved in the current chord	3 * 3 Bits (no dissonance, semitone resolution, whole tone resolution) for alto, tenor and bass voice	1 0 0 1 0 0 1 0 0 1 0 0
Interval Rule	Assesses which intervals from previous to current chord appear	14 Bits (all intervals from prime to octave, or interval > octave) where a bit is set if the corresponding interval appears	1 1 1 0 0 0 0 0 0 0 0 0 0 0
Neutrality Rule	Assesses how the voice motions from previous to current chord neutralize each other	4 Bits to encode the neutrality value (= absolute value of the sum of the contours)	1 0 0 0
Motion Rule	Assesses how much motion takes place from previous and current chord	4 Bits to encode the motion value (= sum of the absolute values of the contours)	1 1 1 0

Definition [weighted multistage graph]

A k -stage graph $G = (V, E)$, $k \geq 2$ is a directed graph in which the vertices are partitioned into k disjoint layers of vertices V_i , $1 \leq i \leq k$, where $V_1 = \{s\}$, $V_k = \{z\}$ contain exactly one vertex. Edges are only allowed from a layer to its successor:

$$\forall \langle u, v \rangle \in E \exists i \ 1 \leq i < k: \quad u \in V_i \wedge v \in V_{i+1}$$

A path $P(v)$ from s to v is a sequence of vertices v_1, \dots, v_i with $v_j \in V_j$ where

$$v_i = v \wedge \forall j \ 1 \leq j < i: \quad \langle v_j, v_{j+1} \rangle \in E$$

A weighted k -stage graph $G = (V, E)$ is a k -stage graph where a weight $c(u, v)$ is associated with every edge $\langle u, v \rangle \in E$. The

costs of a path $P(v)$ are computed as the sum of the costs of the edges on the path:

$$\text{costs}(P(v)) = \text{costs}(v_1, \dots, v_i) = \sum_{j=1}^{i-1} c(v_j, v_{j+1})$$

A model of a weighted 4-stage graph is displayed in Figure 6.

The search for the best chord sequence consists of a sequence of T decisions where T is the length of the chord sequence. For the i -th decision, one has to decide which chord of the filtered chord list at time $i + 1$ lies on the optimal path. All subpaths of an optimal path are again optimal because chord assessment only depends on the previously selected chord. Therefore the principle of optimality can be

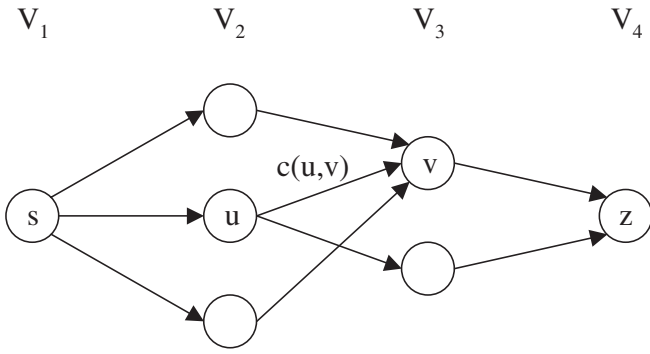


Fig. 6. Model of a weighted 4-stage graph.

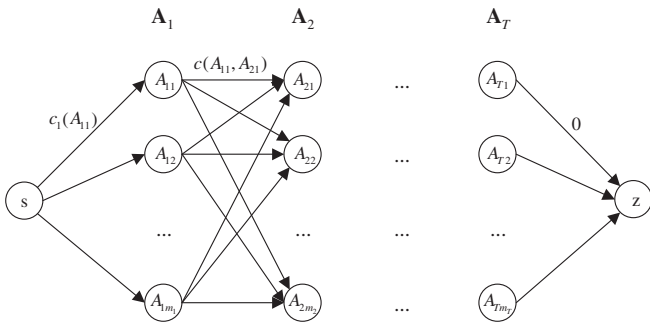


Fig. 7. Representation of the chord sequence as a weighted multistage graph.

applied to the chord sequence optimization problem. The chord sequence to be computed can be represented as a weighted $(T + 2)$ -stage graph (see Fig. 7). The set A_t of all filtered chords at time t corresponds to the vertex set V_{t+1} at stage $t + 1$. Two sequencing chord sets A_t and A_{t+1} are fully connected and the connections are weighted by the cost function c . To this graph a source s and a sink z are added. The weights of the edges between the source and the chords in A_1 are given by the cost function c_1 for the chord structure, the weights between the chords in A_T and the sink are weighted by 0.

In Horowitz and Sahni (1978) an efficient algorithm for computing the path with minimal costs is presented. Let $P(v)$ be a path with minimal costs from s to $v \in V_i$. Then the costs of the path are calculated by minimizing for all predecessors u of v the sum of the costs of $P(u)$ and the costs of the edge $\langle u, v \rangle$:

$$\text{costs}(P(v)) = \min_{u \in V_{i-1}, \langle u, v \rangle \in E} \{ \text{costs}(P(u)) + c(u, v) \} \quad (5)$$

In Horowitz and Sahni (1978) it was also shown that the time complexity for computing the optimal path according to (5) is $\mathcal{O}(|V| + |E|)$. This means that the computational complexity increases linearly with the number of edges. The following corollary estimates the time complexity of the chord sequence optimization problem:

Corollary [time complexity of chord sequence optimization]

The expense for computing the optimal chord sequence using dynamic programming increases only **linearly** with the length T of the chorale.

Proof

Let $m := \max_{1 \leq i \leq T} m_i = \max_{1 \leq i \leq T} |A_i|$ be the maximal size of a chord list. The number of vertices in a $(T + 2)$ -stage graph in the worst case is $|V| = mT + 2$, the maximal number of edges is $|E| = m^2(T - 1) + 2m$. Therefore the worst case time complexity for computing a globally optimal chord sequence according to (5) is given by

$$\mathcal{O}(m^2T) \quad (6)$$

Since m is constant, the expense for computing the optimal chord sequence increases linearly with the length T of the chorale.

Our chord sequence optimization implementation allows to apply the algorithm to subsequences of variable length. In this way, one can interpolate between local and global optimization. An optimization length of 1 corresponds to local optimization, an optimization length of T means global optimization. It is also possible to define optimization lengths at a flexible time scale, e.g., at a musical phrase level.

6. Evaluation

6.1 Plausibility experiment

The performance of the chord assessment network was first tested on a simple non-musical task: a partial ordering of natural numbers had to be transformed into a total ordering. From the set $N_{11} = \{1, \dots, 11\}$ of natural numbers all pairs (a, b) with $a < b$ were generated and encoded in a 1-of-11 coding form: A number i is represented by 11 neurons where the activation of neuron i is 1, and the activation of the other neurons is 0. The resulting $11 * 10/2 = 55$ patterns were presented to the network. The network topology was 22-2-1 which means that no hidden layer was used in the subnets. Because of its symmetric structure, the network does not have to learn the information that $b > a$. Therefore a target value of 1 is presented for every pair. After 50 epochs of network training, the patterns were perfectly learned. Then the subnet output was computed for all numbers. The subnet output activations produced by the network are shown in Table 2.

The network has perfectly constructed a total ordering from the training examples. The learned assessment function is $f(x) = 0.1(x - 1)$ where neighboring numbers are equidistant to each other.

6.2 Music experiment

For learning voice leading, a network was trained with 20 Bach chorales. The generated number of patterns was 6901,

Table 2. Subnet output activations after learning the natural number pairs.

Natural number	1	2	3	4	5	6	7	8	9	10	11
Subnet output activation	0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00

Table 3. Performance comparison of several CHORDNET networks and the symbolic algorithm used in HARMONET on an independent test set of 15 Bach chorales.

Topology	Classification rate Training Set (%)	Classification rate Test Set (%)	Hit rate (%)	Deviation
86 – 2 – 1	89.09	89.21	59.30	0.014
86 – 10 – 2 – 1	90.82	89.40	57.93	0.014
86 – 20 – 2 – 1	91.98	89.70	60.05	0.014
86 – 40 – 2 – 1	94.00	89.87	56.43	0.014
86 – 60 – 2 – 1	92.64	90.11	58.93	0.014
86 – 80 – 2 – 1	96.28	89.82	58.80	0.015
86 – 100 – 2 – 1	94.37	89.89	58.80	0.014
Symbolic Algorithm	–	–	52.06	0.021

the number of melody notes was 1005. This means that the average size of the filtered chord list for each melody note was about 7 chords. Several network topologies and parameters were tested. The performance of the networks was evaluated using the following measures:

- classification rate: the percentage of correctly classified patterns, i.e., chord pairs. A pattern is correctly classified if the target value of the output neuron is 1 and the output value is ≥ 0.5 , or the target value is 0 and the output value is < 0.5 , otherwise it is misclassified. The classification rate computes the percentage of correctly classified patterns. In CHORDNET, a pattern corresponds to a chord pair which is correctly classified if the target value is 1 and chord 2 is preferred to chord 1.
- hit rate: the percentage of chords selected by the composer which get the highest assessment value by the network.
- deviation: a more differentiated evaluation of the hit rate. For each melody note s_i , the zero-based position $rank(i)$ of the composer's chord in the sorted assessed chord list is determined. This value is then normalized for the size of the chord list $size(i)$ and the number n of melody notes:

$$deviation = \frac{1}{n} \sum_{i=1}^n \frac{rank(i)}{size(i)} \quad (7)$$

The resulting value within the interval $[0, 1)$ indicates how close the network's chord rankings are to the composer's selections. A deviation of 0 means that all composer's chords are ranked first by the network, a deviation near 1 means that the composer's chords got low rankings by the network.

Table 3 shows classification rates, hit rate and deviation for various networks on an independent test set of 15 Bach chorales. Each network was trained 50 epochs. For four-layer networks, a weight decay parameter of 0.5 was introduced to

Table 4. Performance comparison of local and global optimization on a test set of 15 Bach chorales.

	With Local Optimization	With Global Optimization
Average costs	7.73	7.56
Voice leading errors	11	1

avoid overfitting. The last row shows the hit rate and deviation for the HARMONET approach using manually fitted assessment values. All networks clearly outperform the symbolic algorithm. The classification rates of the three-layer network (first row) can be slightly improved by adding hidden neurons to the subnets. The best hit rate is produced by the 86-20-2-1 network. This network is used in the following.

After testing the networks the effect of the global optimization strategy was examined (see Table 4). The locally optimized chord sequence was constructed by simply selecting the top-rated chord from the chord list associated with each melody note. Global optimization using dynamic programming reduces the average costs of the produced chord sequences and prevents voice leading errors. Only one voice leading error was found in the 15 optimized chord sequences.

From the musical results invented by CHORDNET, the impression was confirmed that global optimization produces much smoother voice leadings than local optimization. However, the voice leading seems to be even slightly better if it is not applied to the whole chord sequence, but to musical phrases. This agrees with the observation that musical phrases are perceived as entities such that voice leading needs to be smooth within a phrase but not necessarily between two phrases.

filter rule violation:
overlapping soprano/alto voices

close position

open position

T D DD7 D D3 Tp3,7 D T T Tp Dp D3 Tp T3 Sp T

T D Sp D D3 Tp3,7 D T T Tp Dp D3 Tp T3 Sp T

T D S3 D5,7 T3 S D T T T T T T T S T

Fig. 8. Harmonization/Voice leading of Bach chorale melody “Vom Himmel hoch” produced by HARMONET/CHORDNET.
Top: Bach style with local optimization.
Middle: Bach style with global optimization.
Bottom: Scheidt style with global optimization.

Figure 8 shows harmonizations and voice leading sequences of the beginning of the chorale melody “Vom Himmel hoch” in the style of J. S. Bach and Samuel Scheidt, typical examples invented by the combined HARMONET – CHORDNET system. The first chord sequence was produced by CHORDNET with local optimization, the second and the third chord sequence were produced with global optimization. The networks were trained with 20 Bach chorales and 23 Scheidt chorales, respectively.

One can notice that the first chord sequence starts in close position and runs into a dead end where a filter rule violation cannot be avoided. Due to global optimization, CHORDNET prefers an open position in the second chord sequence. This leads to a much smoother and more elegant voice leading and avoids the rule violation. The harmonization and voice leading of the example in the style of the early baroque composer Samuel Scheidt are much simpler than the two Bach style examples.

7. Conclusion

We have presented CHORDNET, a novel approach to the learning and generation of voice leading using neural networks and dynamic programming. In combination with the HARMONET system, it presents a powerful framework able to learn stylistic aspects of chorale harmonization directly from a set of training examples. Because of its flexibility, the system can be used for both on-line and off-line harmonization.

Without global optimization, it can be used for improvising harmonizations in real time, as shown in Höthker and Hörnel (2000). With global optimization, smooth and stylistically coherent harmonizations and voice leadings can be generated efficiently.

The generality of the CHORDNET approach makes it applicable to other (also non-musical) problems. However, two main conditions should be fulfilled:

- Assessment learning: The number of learning patterns does not only depend on number and size of the sequences, but also on the size of the filtered list (see pattern size estimation (1)). If the maximal size m of the filtered chord list as given in (6) is too large, the pattern set might be very big and it will be difficult to learn an appropriate assessment function.
- Sequence optimization: It must be checked whether the principle of optimality is fulfilled. This is true if the problem can be formulated in such a way that the assessment only depends on the previously selected position (otherwise the graph cannot be partitioned into stages). For example, the prediction of a harmonic function in HARMONET depends on three previous harmonies. Therefore the approach can not be directly applied to the assessment of harmonic functions.

One promising perspective is to use the approach for learning musical counterpoint. Like voice leading, musical counterpoint seems to be dominated by rules at first glance. However, there are many aspects of counterpoint that cannot be appropriately described by rules. Up to now, only a small number of ideas for learning musical counterpoint exist. Some of the learning aspects of counterpoint might be covered by the presented approach.

References

- Bellgard, M.I., & Tsang, C.P. (1996). On the Use of an Effective Boltzmann Machine for Musical Style Recognition and Harmonisation. *Proceedings of the 1996 International Computer Music Conference*. International Computer Music Association, pp. 461–464.
- Bellman, R.E., & Dreyfus, S.E. (1962). *Applied Dynamic Programming*. Princeton University Press.
- Berger, J., & Gang, D. (1997). A Neural Network Model of Metric Perception and Cognition in the Audition of Functional Tonal Music. *Proceedings of the 1997 International Computer Music Conference*. International Computer Music Association, pp. 23–26.
- Braun, H., Feulner, J., & Ullrich, V. (1991). Learning strategies for solving the problem of planning using backpropagation. *Proceedings of the Fourth International Conference on Neural Networks*. Nimes, pp. 671–685.
- Ebcioğlu, K. (1988). An expert system for harmonizing four-part chorales. *Computer Music Journal*, 12, 43–51.
- Feulner, J., & Langnickel, J. (1998). Informationsstrukturen in der Musik: Analyse und Modellierung mit Methoden der Informatik an der Universität Karlsruhe. *Musik und Neue Technologie, KlangArt-Kongreß 1995*. B. Enders & N. Knolle (eds.). Osnabrück: Universitätsverlag rasch, pp. 295–313.
- Grabner, H. (1967). *Handbuch der funktionellen Harmonielehre*. Bosse musik paperback.
- Hild, H., Feulner, J., & Menzel, W. (1991). HARMONET: A Neural Net for Harmonizing Chorals in the Style of J. S. Bach. *Advances in Neural Information Processing Systems 4 (NIPS 4)*. R. P. Lippmann, J. E. Moody & D. S. Touretzky (Eds.). Morgan Kaufmann, pp. 267–274.
- Hörnel, D., & Ragg, T. (1996). Learning Musical Structure and Style by Recognition, Prediction and Evolution. *Proceedings of the 1996 International Computer Music Conference*. International Computer Music Association, pp. 59–62.
- Hörnel, D., & Menzel, W. (1998). Learning musical structure and style with neural networks. *Computer Music Journal*, 22, 44–62.
- Hörnel, D., & Höthker, K. (1998). A Learn-Based Environment for Melody Completion. *Proceedings of the XII Colloquium on Musical Informatics*. AIMI Association, pp. 121–124.
- Horowitz, E., & Sahni, S. (1978). *Fundamental of Computer Algorithms*. Computer Science Press.
- Höthker, K., & Hörnel, D. (2000). Harmonizing in Real-Time with Neural Networks. *Proceedings of the 2000 International Computer Music Conference*. International Computer Music Association, pp. 527–530.
- McIntyre, R.A. (1994). Bach in a Box: The Evolution of Four Part Baroque Harmony Using the Genetic Algorithm. *Proceedings of the 1994 IEEE Conference on Evolutionary Computation*. Orlando, Florida.
- Riedmiller, M., & Braun, H. (1993). A Direct Adaptive Method for Faster Backpropagation Learning: The Rprop Algorithm. *Proceedings of the ICNN*. IEEE, pp. 586–591.
- Rumelhart, D.E., & McClelland, J.L. (1988). *Explorations in Parallel Distributed Processing*. Cambridge, USA: MIT Press.
- Ullrich, V. (1991). *Erlernen von Spielstrategien für Mühle durch neuronale Netze*. Diploma Thesis, University of Karlsruhe.

