Relational IBL in Music with a New Structural Similarity Measure

Asmir Tobudic[1] and Gerhard Widmer[1,2]

[1] Austrian Research Institute for Artificial Intelligence, Vienna,
[2] Department of Medical Cybernetics and Artificial Intelligence, University of Vienna,
asmir|gerhard@oefai.at

Abstract. It is well known that many hard tasks considered in machine learning and data mining can be solved in an rather simple and robust way with an instance- and distance-based approach. In this paper we present another difficult task: learning, from large numbers of performances by concert pianists, to play music expressively. We model the problem as a multi-level decomposition and prediction task. Motivated by structural characteristics of such a task, we propose a new relational distance measure that is a rather straightforward combination of two existing measures. Empirical evaluation shows that our approach is in general viable and our algorithm, named DISTALL, is indeed able to produce musically interesting results. The experiments also provide evidence of the success of ILP in a complex domain such as music performance: it is shown that our instance-based learner operating on structured, relational data outperforms a propositional k-NN algorithm.

Keywords: relational instance-based learning, music

1 Introduction

Instance-based learning has always been very popular within machine learning and data mining. During the long research history on IBL, countless studies have stressed its strong aspects: algorithmic simplicity, incrementality, almost obvious extensions for handling noisy examples and/or attributes, the ability to deal with discrete as well as with continuous attributes, and often surprisingly good performance. Although most research on IBL has been done in a propositional setting, recently there has been a growing interest in transferring the successful IBL framework to the richer first-order logic (FOL) representation. A number of instance-based learners operating in the FOL framework have already been developed — e.g. KBG [1], RIBL [5], STILL [13] — and shown to work well on a number of tasks.

This paper introduces another difficult task for relational IBL, from the area of music research. We would like to automatically build, via inductive learning from 'real-world' data (i.e., real performances by highly skilled musicians), predictive models of certain aspects of performance (e.g. tempo, timing, dynamics, etc). Previous research has shown that computers can indeed find and

describe interesting and useful regularities at the level of individual notes. Using a new machine learning algorithm [17], we succeeded in discovering a small set of simple, robust and highly general rules that predict a substantial part of the note-level choices of a performer (e.g., whether (s)he will shorten or lengthen a particular note) with high precision [16]. However, music performance is a highly complex activity, and the note level is far from sufficient as a basis for a complete model of expressive performance. The goal of our ongoing work is to build quantitative models of musical expression at different levels of abstraction: we would like to learn tempo and dynamics strategies at levels of *hierarchically nested phrases*.

In this paper we show how relational IBL can be applied to learn expressive tempo and dynamics patterns at different phrase levels. We also propose a new similarity measure for structured objects described in FOL, which is a fairly straightforward modification of existing measures and can be regarded as a combination of two techniques: (1) RIBL's [5] strategy for assessing similarity between FOL objects by computing the similarity between objects' properties and the similarity of the objects related to them, and (2) a definition of distance between two sets based on the notion of transport networks, as proposed in [10].

Our similarity measure has been built into a relational instance-based learner named DISTALL and applied to our music task. DISTALL predicts timing and dynamics patterns for phrases in a new piece by analogy to the most similar phrases in the training set. Empirical evaluation shows that the relational IBL approach is viable in this domain and DISTALL is indeed able to achieve musically sensible results. Experiments also show that DISTALL produces clearly better results than a propositional k-NN algorithm, which provides additional evidence for the benefits of relational instance-based learning.

The paper is organized as follows: Section 2 introduces the notion of expressive music performance and its representation via performance curves. We also show how hierarchically nested musical phrases are represented in FOL, and how complex tempo and dynamics curves can be decomposed into well-defined training instances for the instance-based learning algorithm. In Section 3 we describe our distance measure in detail. Section 4 gives empirical results on DISTALL's performance and a comparison with both RIBL and a propositional k-NN algorithm. Section 5 gives a brief conclusion.

2 Real-World Task: Learning To Play Music Expressively

The work presented here is part of a large research project that studies the fundamentals of *expressive music performance* via AI and, in particular, machine learning [18]. Expressive music performance is the art of shaping a musical piece by continuously varying important parameters like tempo, loudness, etc. while playing a piece. Instead of playing a piece of music with constant tempo or loudness, (skilled) performers rather speed up at some places, slow down at others, stress certain notes or passages etc. The way this 'should be' done is not specified

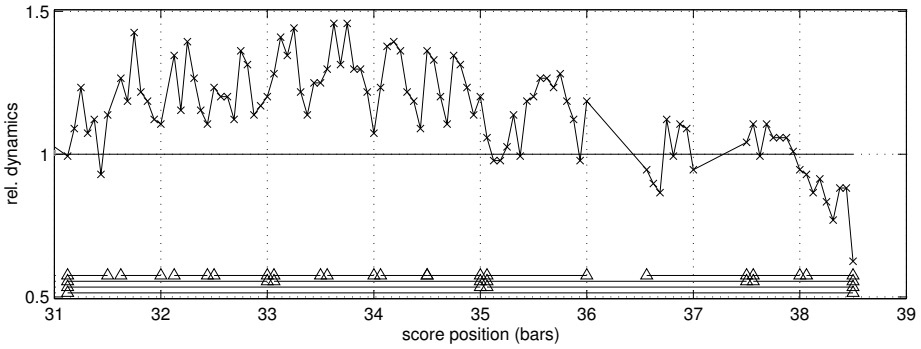

Fig. 1. Dynamics curve (relating to melody notes) of performance of Mozart Sonata KV.279, 1^{st} movement, mm. 31–38, by a Viennese concert pianist.

precisely in the written score[3], but at the same time it is absolutely essential for the music to sound alive. The aim of this work is learning predictive models of two of the most important expressive parameters: *timing* (tempo variations) and *dynamics* (loudness variations).

The tempo and loudness variations can be represented as curves which quantify the variations of these parameters for each note relative to some reference value (e.g. average loudness or tempo of the same piece). Figure 1 shows a *dynamics curve* of a small part of the Mozart piano Sonata K.279 (C major), 1st movement, as played by a Viennese concert pianist (computed from recordings on a Bösendorfer SE290 computer-monitored grand piano[4]). Each point represents the relative loudness with which a particular melody note was played (relative to an average loudness of the piece); a purely mechanical (unexpressive) rendition of the piece would correspond to a flat horizontal line at $y = 1.0$. Tempo variations can be represented in an analogous way.

A careful examination of the figure reveals some trends in the dynamics curve. For instance, one can notice an up-down, *crescendo-decrescendo* tendency over the presented part of the piece and relatively consistent smaller up-down patterns embedded in it. This is not an accident since we chose to show a part of the piece which is a musically meaningful unit: a high-level *phrase*. This phrase contains a number of lower-level phrases, which are apparently also 'shaped' by the performer. The hierarchical, four-level phrase structure of this passage is indicated by four levels of brackets at the bottom of the figure. The aim of our work is the automatic induction of tempo and dynamics strategies, at different levels of the phrase structure, from large amounts of real performances by

[3] The *score* is the music as actually printed.

[4] The SE290 is a full concert grand piano with a special mechanism that measures every key and pedal movement with high precision and stores this information in a format similar to MIDI. From these measurements, and from a comparison with the notes in the written score, the tempo and dynamics curves corresponding to the performances can be computed.

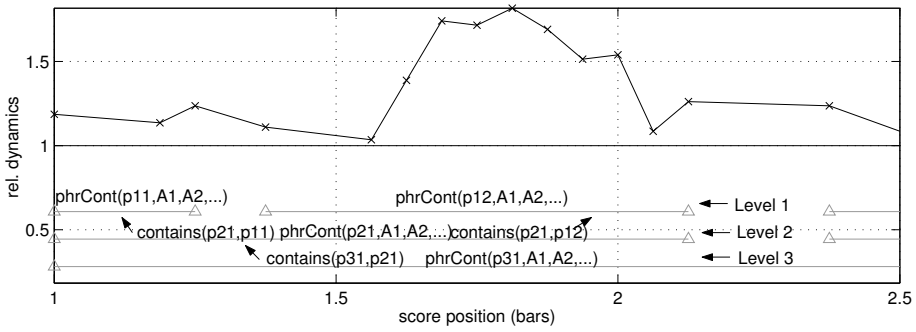

Fig. 2. Phrase representation used by our relational instance-based learning algorithm.

concert pianists. The heart of our system, the relational instance-based learning algorithm described below, recognizes similar phrases from the training set and applies their expressive patterns to a new (test) piece. In this section we will describe the steps which precede and succeed the actual learning: First we show how hierarchically nested phrases are represented in first-order logic. We then show how complex tempo and dynamics curves as measured in real performances can be decomposed into well-defined training instances for the learner. Finally, we discuss the last step: at prediction time, the shapes predicted by the learner for nested phrases at different levels must be combined into a final performance curve that can be used to produce a computer-generated 'expressive' performance.

2.1 Representing Musical Phrases in FOL

Phrases are segments of music heard and interpreted as coherent units; they are important structural building blocks of music. Phrases are organized hierarchically: smaller phrases are grouped into higher-level phrases, which are in turn grouped together, constituting a musical context at a higher level of abstraction etc. The phrases and relations between them can be naturally represented in first-order logic.

Consider Figure 2. It shows the dynamics curve corresponding to a small portion (2.5 bars) of a Mozart sonata performance, along with the piece's underlying phrase structure. For all scores in our data set phrases are organized at four hierarchical levels, based on a manual phrase structure analysis. The musical content of each phrase is encoded in the predicate $phrCont(Id, A1, A2, ...)$. *Id* is the phrase identifier and *A1,A2,...* are attributes that describe very basic phrase properties like the length of a phrase, melodic intervals between the starting and ending notes, information about where the highest melodic point (the 'apex') of the phrase is, the harmonic progression between start, apex, and end, etc. Relations between phrases are specified via the predicate $contains(Id1, Id2)$, which states that the bigger phrase *Id1* contains the smaller one *Id2*. Note that smaller phrases (consisting only of a few melody notes) are described in detail by the

predicate *phrCont*. For the bigger phrases — containing maybe several bars — the high-level attributes in *phrCont* are not sufficient for a full description. But having links to the lower-lever phrases through the *contains* predicate and their detailed description in terms of *phrCont*, we can also obtain detailed insight into the contents of bigger phrases.

In ILP terms, the description of the musical scores through the predicates *phrCont* and *contains* defines the background knowledge of the domain. What is still needed in order to learn are the training examples, i.e. for each phrase in the training set, we need to know how it was played by a musician. This information is given in the predicate *phrShape*(*Id*, *Coeffs*), where *Coeffs* encode information about the way the phrase was played by a pianist. This is computed from the tempo and dynamics curves, as described in the following section.

2.2 Deriving the Training Instances: Multilevel Decomposition of Performance Curves

Given a complex tempo or dynamics curve (see Figure 1) and the underlying phrase structure, we need to calculate the most likely contribution of each phrase to the overall observed expression curve, i.e., we need to decompose the complex curve into basic expressive phrase 'shapes'. As approximation functions to represent these shapes we decided to use the class of second-degree polynomials (i.e., functions of the form $y = ax^2 + bx + c$), because there is ample evidence from research in musicology that high-level tempo and dynamics are well characterized by quadratic or parabolic functions [14]. Decomposing a given expression curve is an iterative process, where each step deals with a specific level of the phrase structure: for each phrase at a given level, we compute the polynomial that best fits the part of the curve that corresponds to this phrase, and 'subtract' the tempo or dynamics deviations 'explained' by the approximation. The curve that remains after this subtraction is then used in the next level of the process. We start with the highest given level of phrasing and move to the lowest. As tempo and dynamics curves are lists of multiplicative factors (relative to a default tempo), 'subtracting' the effects predicted by a fitted curve from an existing curve simply means dividing the y values on the curve by the respective values of the approximation curve.

Figure 3 illustrates the result of the decomposition process on the last part (mm.31–38) of the Mozart Sonata K.279, 1st movement, 1st section. The four-level phrase structure our music analyst assigned to the piece is indicated by the four levels of brackets at the bottom of the plot. The elementary phrase shapes (at four levels of hierarchy) obtained after decomposition are plotted in gray.

We end up with a training example for each phrase in the training set — a predicate *phrShape*(*Id*, *Coeff*), where $Coeff = \{a, b, c\}$ are the coefficients of the polynomial fitted to the part of the performance curve associated with the phrase.

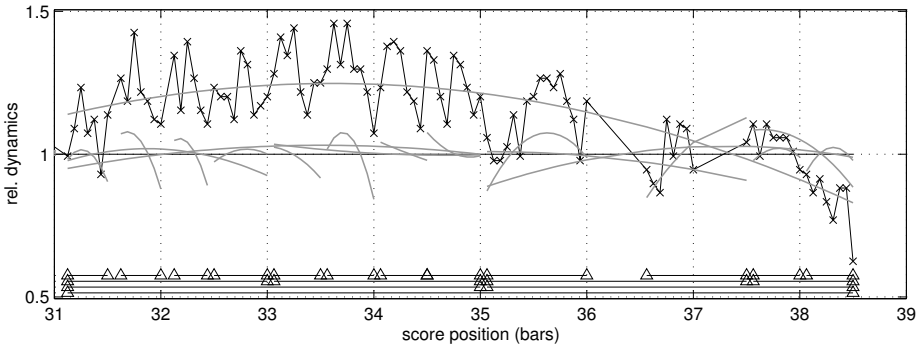

Fig. 3. Multilevel decomposition of dynamics curve of performance of Mozart Sonata K.279:1:1, mm.31-38.: original dynamics curve plus the second-order polynomial shapes giving the best fit at four levels of phrase structure

2.3 Combining Multi-level Prase Predictions

Input to the learning algorithm are the (relational) representation of the musical scores plus the training examples (i.e. timing and dynamics polynomials), for each phrase in the training set. Given a test piece the learner assigns the shape of the most similar phrase from the training set to each phrase in the test piece. In order to produce final tempo and dynamics curves, the shapes predicted for phrases at different levels must be combined. This is simply the inverse of the curve decomposition problem. Given a new piece to produce a performance for, the system starts with an initial 'flat' expression curve (i.e., a list of 1.0 values) and then successively multiplies the current value by the multi-level phrase predictions.

3 Instance-based Learning in FOL

This section presents the relational IBL learner DISTALL and briefly contrasts it with its ancestor RIBL [5], by showing via an example how they implement different notions of structural similarity.

3.1 A Structural Similarity Measure

Before explaining DISTALL in detail, we recall some definitions concerning first order logic and inductive logic programming from [6, 9].

Definition 1 (linked clause). *A clause is linked if all of its variables are linked. A variable v is linked in a clause c if and only if v occurs in the head of c, or there is a literal l in c that contains the variables v and w ($v \neq w$) and w is linked in c.*

Example 1. Clause $p(A) \leftarrow r(B)$ is not linked, while $p(A) \leftarrow q(A,B), r(B,C), u(D,C)$ is.

Definition 2 (level of term). *The level $l(t)$ of a term t in a linked clause c is 0 if t occurs as an argument in the head of c; and 1+min $l(s)$ where s and t occur as arguments in the same literal of c.*

Example 2. The variable F in $father(F,C) \leftarrow male(F), parent(F,C)$ has level 0, the variable G in $grandfather(F) \leftarrow male(F), parent(F,C), parent(C,G)$ has level 2.

The algorithm to be presented here can be regarded as a generalization of the propositional k-NN for examples described in first-order logic. In FOL, examples are usually represented as sets of ground facts. The heart of a relational IBL algorithm is thus a distance function between sets of elements. A number of distances on sets already exist, e.g. the Hausdorff metric, symmetric difference distances, distances based on relations between sets, etc. For our algorithm, we adopt the distance proposed in [10, 11]. The distance between sets of elements is defined in [10] via transport networks (for more information on the concept of transport networks see [7]).

First, the appropriate transport network is constructed. The network has two groups of vertices $\{a_i\}$ and $\{b_i\}$ corresponding to the elements of the two sets A and B; a starting and an ending vertex (source s and sink t); and two additional vertices, let us call them a_- and b_-. For all edges in the network, *capacities* and *weights* are defined, where capacities represent the maximal amount of 'units' which can 'flow' through a connection and the weights are the distances (transport costs) between particular vertices.

The distance between two sets of elements is then defined as the solution of the maximum flow minimal weight problem: one would like to transport as much as possible from s to t with minimal costs. In other words, one wants to maximally match elements from one set with the elements of the other and achieve the minimal possible distance. By setting the weights of the edges between set elements and the two additional vertices a_- and b_- to a big constant (e.g. bigger than all other edge weights), a 'penalty' is modeled: all elements of one set which do not match with any element of the other cause big costs. By associating appropriate capacities with the edges in the network one can generalize the notion of cardinality in such a way that sets of different cardinality can be scaled appropriately (e.g. allowing the elements of the smaller set to match up more than one element of the other and avoiding that the distance of two sets with vastly different cardinalities is expressed mainly through penalty). An example of a distance network is given in Figure 4.

More formally, the weighting function for sets and generalization of the notion of cardinality can be defined as follows [10]:

Definition 3 (weighting function). *Let X be a universe. A weighting function W for X is a function that maps each subset $A \in 2^X$ to a function $W[A] : A \rightarrow \Re$.*

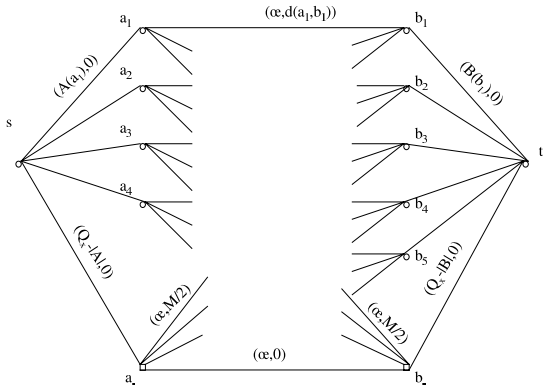

Fig. 4. A distance network (see [11])

Definition 4 (Size under weighting function). *Let W be a weighting function for X. Then the function* $size_w : 2^X \rightarrow \Re^+$ *is defined as* $size_W(A) = \sum_{a \in A} W[A](a)$.

The distance network can be formalized as (see figure 4):

Definition 5 (distance network). *Let X be a set, and d a metric on X. Let M be a constant, W be a weighting function for X and* $Q_X^W = max_{A \in 2^X} size_W(A)$. *Then for all finite* $A, B \in 2^X$ *with* $A = \{a_1, ..., a_m\}$ *and* $B = \{b_1, ..., b_n\}$, *we define a distance network between A and B for d, M and W in X to be N[X,d,M,W,A,B]=N(V,E,cap,s,t,w) with* $V = A \cup B \cup \{s, t, a_-, b_-\}, E = (\{s\} \times (A \cup \{a_-\})) \cup ((B \cup \{b_-\}) \times \{t\}) \cup ((A \cup \{a_-\}) \times (B \cup \{b_-\}))$, $\forall a \in A, \forall b \in B : w(s,a) = w(b,t) = w(s,a_-) = w(b_-,t) = w(a_-,b_-) = 0 \wedge w(a,b) = d(a,b) \wedge w(a_-,b) = w(a,b_-) = M/2$ *and* $\forall a \in A, \forall b \in B : cap(s,a) = W[A](a) \wedge cap(b,t) = W[B](b) \wedge cap(s,a_-) = Q_X^W - size_W(A) \wedge cap(b_-,t) = Q_X^W - size_W(B) \wedge cap(a,b) = cap(a_-,b) = cap(a,b_-) = cap(a_-,b_-) = \infty$ *(see Figure 4).*

The distance between sets of elements is then defined as follows:

Definition 6 (netflow distance). *Let X be a set, d a metric on X, M a constant, and W a weighting function for X. For all* $A, B \in 2^X$, *the netflow distance from A to B under d, M and W in X, denoted* $d^N_{X,d,M,W}(A,B)$, *is the weight of the minimal weight maximal flow from s to t in N[X,d,M,W,A,B].*

In [7] it has been shown that if W has integer values, the maximal flow minimal weight problem can be solved in time polynomial in $size_W(A)$ and $size_W(B)$. If the weights of the netflow distance are normalized (in the interval [0,1]), the netflow distance can also be normalized:

Definition 7 (normalized netflow distance). *Let X be a set and d a normalized metric on X. Then, the normalized netflow distance based on d is a distance function $d^{N,n}_{X,d,M,W}(A,B) : X \times X \to \Re$ defined by:*

- *0 if (X=0 and Y=0)*
- $\frac{2 \times d^{N}_{X,d,M,W}(A,B)}{d^{N}_{X,d,M,W}(A,B)+(size_W(X)+size_W(Y))/2}$ *otherwise*

Although the maximal matching distance defined via transport networks can be calculated in polynomial time, applying it directly on examples described in FOL (containing maybe hundreds of ground facts) would cause impractically high computational costs. Another problem is that given an example described by many facts, the relevance of particular facts is hard to tune (e.g. by weighting them differently). We avoid these problems by collecting the facts derived from background knowledge into hierarchical *subsets*. By doing so we develop a context-sensitive similarity measure where terms with a lower linkage level to the objects whose distance we are interested in can be made to influence the distance more. In the following we describe our algorithm in more detail. The difference between our approach and the RIBL algorithm, which uses a similar strategy for grouping facts, is illustrated on an example from our learning problem in the next subsection.

3.2 The DISTALL Algorithm

The first step of the algorithm is building so-called starting clauses. The building of starting clauses is a well known method in ILP employed for reasons of computational complexity and implemented in many systems (CLINT [2], GOLEM [8], ITOU [12], RIBL [5]). For each (training and test) example we collect literals that contain terms linked to the example and group them into subsets according to linkage levels. The building of starting clauses is in our case guided by types and modes of literals. The distance between a test and training example is computed as the set distance between all literals found in the test and training starting clause at linkage level 1 . In other words, we find the solution of the maximal flow minimal weight problem (see Definition 6), where the transport network vertices are literals containing terms which are directly linked to the examples. The weights of the edges (i.e. distances between individual literals) are computed as the Manhattan distance defined over the literals' arguments (or set to 1 if the literals have different functors). If the arguments are object identifiers, the distance between them is computed by expanding them into a new transport network where the vertices are literals also containing these objects, found one linkage level deeper in the starting clauses. At the lowest level, the distance between objects is calculated as the distance between discrete values.

The *capacities* associated with the edges in the network can be used to control the 'virtual' cardinalities of sets (and, accordingly, the influence of penalty on the set distance). They can also be used to give different importance to the predicates in sets.

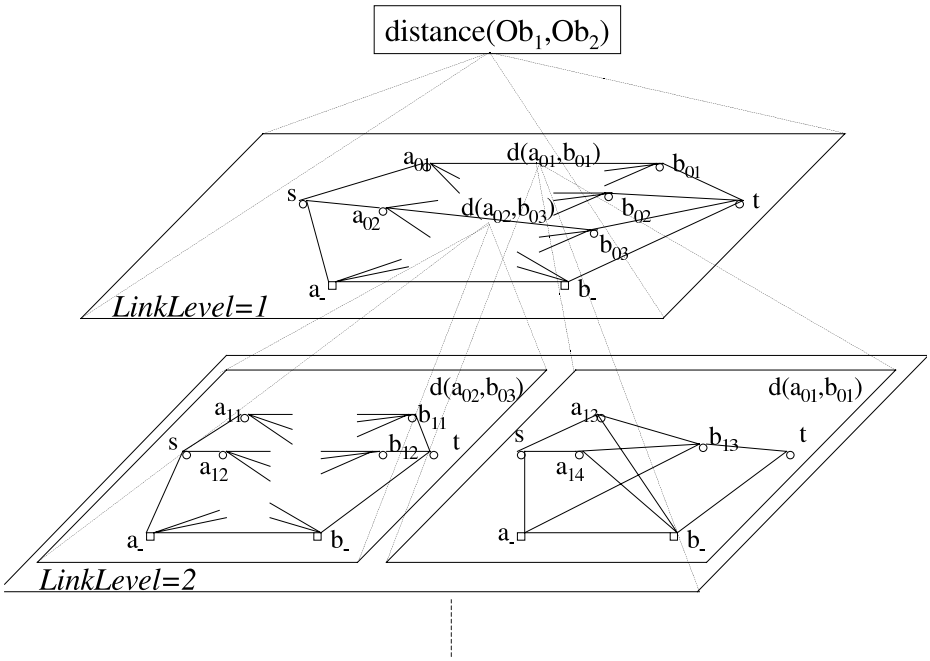

Fig. 5. Basic principle of DISTALL's similarity measure

The basic principle of the algorithm, let us call it DISTALL (DIstance on SeTs of Appropriate Linkage Level), is illustrated in Figure 5. In the example, the distance between objects Ob_1 and Ob_2 is calculated as the solution of the maximal flow minimal weight problem on the sets of literals found at $LinkLevel = 1$ in the starting clauses built for Ob_1 and Ob_2. The weights $d(a_i, b_j)$ of edges connecting literals containing no object identifiers are computed directly (via Manhattan distance, see above). In the example, the literals a_{01} and b_{01} as well as a_{02} and b_{03} have same functors and object identifiers as arguments. The weights of edges between them are thus defined as distance network problems involving the literals containing these objects, found one linkage level deeper in the starting clause. The procedure continues recursively, until the depth of the starting clauses is reached. The computational cost is kept small, since the algorithm solves many hierarchically nested transport network problems with a small number of vertices in one network.

Notice that the distances between vertices are normalized. With a normalized distance between vertices we can apply Definition 7 and normalize the netflow distance. Normalized netflow distance is in turn used in the computation of the (normalized) distance between literals in the 'higher-level' transport network.

3.3 DISTALL vs. RIBL

DISTALL can be regarded as a continuation of the line of research initiated in [1], where a clustering algorithm together with its similarity measure was presented. This work was later improved in [5], in the context of the relational instance-based learning algorithm RIBL. The main idea behind RIBL's similarity measure is that the similarity between two objects is determined by the similarity of their attributes and the similarity of the objects related to them. The similarity of the related objects depends in turn on their attributes and related objects.

DISTALL combines this idea with a set distance function based on the notion of transport networks, which was proposed in [10]. While RIBL's similarity measure permits several literals in one example to match the same literal in the other example, DISTALL's netflow distance strongly favors matchings that are as complete as possible and penalizes literals left unmatched. We argue that the so defined distance is more natural in structured domains and works better in practice. First we show the main difference between RIBL's and DISTALL's behavior in one constructed situation from our musical application domain. In the next section we present DISTALL's empirical results and provide a direct comparison with RIBL.

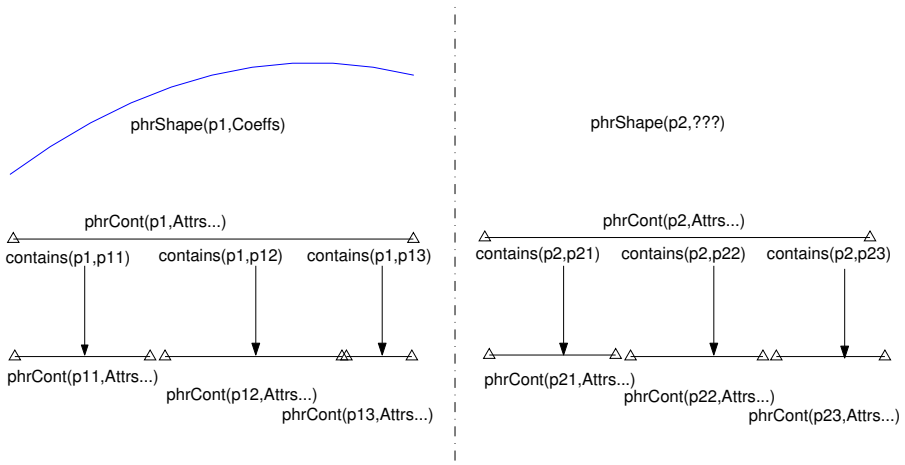

Fig. 6. An example of relational learning situation: training example (left) and new test case (right)

Consider the situation given in Figure 6. We are interested in predicting the 'expressive shape' of the high-level phrase $p2$ and thus want to calculate the distance between phrases $p1$ and $p2$. Each of the two phrases is described via the attributes stored in the phrase-content predicate *phrCont*. They also contain lower-level phrases (predicate *contains*), which are in turn described with their phrase-content predicates. RIBL would compute the similarity between

$p1$ and $p2$ as a (weighted) sum of the similarities between the *phrCont* and *contains* predicates, where for each *contains* predicate of $p2$, the most similar $contains(p1, X)$ predicate is found (by finding the most similar *phrCont* and *contains* predicate at the lower level). Imagine the situation where the short lower-level phrase $p13$ is a *prototype* of all lower-level phrases of $p2$ (i.e. the sum of the distances between $p13$ and all $p2x$ phrases is minimal, and the other two lower-level phrases $p11$ and $p12$ are completely different from all phrases $p2x$). By matching all $p2x$ phrases to $p13$ RIBL would obtain a relatively high similarity between $p1$ and $p2$. This is not what we want, as the internal details of the *whole* high-level phrase $p1$ are 'responsible' for the expressive shape and not just a small fraction. In DISTALL, on the other hand, such a matching that leaves two of three subphrases unmatched would receive a high penalty and thus result in a low similarity rating.[5]

4 Experiments

In the following we present detailed empirical results achieved with DISTALL on a complex real-world dataset derived from piano performances of classical music and compare these with results achieved by RIBL. We also provide a comparison with a simpler propositional approach.

4.1 The Data

The data used for the experiments was derived from performances of Mozart piano sonatas by a Viennese concert pianist on a Bösendorfer SE 290 computer-controlled grand piano. A multi-level phrase structure analysis of the musical score was carried out manually by a musicologist. Phrase structure was marked at four hierarchical levels; three of these were finally used in the experiments. The sonatas are divided into sections, which can be regarded as coherent pieces. The resulting set of annotated pieces is summarized in Table 1. The pieces and performances are quite complex and different in character; automatically learning expressive strategies from them is a challenging task.

4.2 A Quantitative Evaluation of DISTALL

A systematic *leave-one-piece-out* cross-validation experiment was carried out. Each of the 16 sections was once set aside as a test piece, while the remaining 15 pieces were used for learning. DISTALL uses one nearest neighbor for prediction,

[5] One could avoid RIBL's 'mismatching' of the lower-level phrases, e.g. by introducing a new predicate $relPos(Phrase, Position)$, which describes the relative position of lower-level phrases within higher-level phrases. In this case, the phrases with the same relative position would be matched with higher probability. We do not want to make use of this information, since there are relational learning problems which do not have such well defined alignment information between objects.

Table 1. Mozart sonata sections used in experiments (to be read as <sonataName>:<movement>:<section>); *notes* refers to 'melody' notes.

sonata section	notes	phrases at level 1	2	3	4
kv279:1:1 fast 4/4	391	50	19	9	5
kv279:1:2 fast 4/4	638	79	36	14	5
kv280:1:1 fast 3/4	406	42	19	12	4
kv280:1:2 fast 3/4	590	65	34	17	6
kv280:2:1 slow 6/8	94	23	12	6	3
kv280:2:2 slow 6/8	154	37	18	8	4
kv280:3:1 fast 3/8	277	28	19	8	4
kv280:3:2 fast 3/8	379	40	29	13	5
kv282:1:1 slow 4/4	165	24	10	5	2
kv282:1:2 slow 4/4	213	29	12	6	3
kv282:1:3 slow 4/4	31	4	2	1	1
kv283:1:1 fast 3/4	379	53	23	10	5
kv283:1:2 fast 3/4	428	59	32	13	6
kv283:3:1 fast 3/8	326	52	30	12	3
kv283:3:2 fast 3/8	558	78	47	19	6
kv332:2 slow 4/4	477	49	23	12	4
Total:	5506	712	365	165	66

with the starting clause depth set to 3 (i.e. just those phrases whose relationship order to the phrase in question is ≤ 3 can influence the distance measure).

The expressive shapes for each phrase in a test piece were predicted by DISTALL and then combined into a final tempo and dynamics curve, as described in section 2.3. The following performance measures were computed: the *mean squared error* of the system's predictions on the piece relative to the actual expression curve produced by the pianist ($MSE = \sum_{i=1}^{n} (pred(n_i) - expr(n_i))^2/n$), the *mean absolute error* ($MAE = \sum_{i=1}^{n} |pred(n_i) - expr(n_i)|/n$), and the *correlation* between predicted and 'true' curve. MSE indicates those cases where the learner produces rather extreme 'errors'. MSE and MAE were also computed for a *default* curve that would correspond to a purely mechanical, unexpressive performance (i.e., an expression curve consisting of all 1's). That allows us to judge if learning is really better than just doing nothing. The results of the experiment are summarized in table 2, where each row gives the results obtained on the respective test piece when all others were used for training. The last row ($WMean$) shows the weighted mean performance over all pieces (individual results weighted by the relative length of the pieces).

We are interested in cases where the *relative errors* (i.e., $\text{MSE}_L/\text{MSE}_D$ and $\text{MAE}_L/\text{MAE}_D$) are less than 1.0, that is, where the curves predicted by the learner are closer to the pianist's actual performance than a purely mechanical rendition. In the dynamics dimension, this is the case in 12 out of 16 cases for MSE, and in 14 out of 16 for MAE. The results for tempo are dramatically

Table 2. Results, by sonata sections, of cross-validation experiment with DISTALL (*depth=2*, *k=1*). Measures subscripted with D refer to the 'default' (mechanical, inexpressive) performance, those with L to the performance produced by the learner. The cases where DISTALL is better than the default are printed in bold.

	dynamics					tempo				
	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$
kv279:1:1	.0383	**.0214**	.1643	**.1100**	.6714	.0348	.0375	.1220	.1257	.3061
kv279:1:2	.0318	.0355	.1479	**.1384**	.5744	.0244	.0291	.1004	.1133	.3041
kv280:1:1	.0313	**.0195**	.1432	**.1052**	.6635	.0254	**.0188**	.1053	**.0934**	.5611
kv280:1:2	.0281	.0419	.1365	.1482	.4079	.0250	.0290	.1074	.1111	.3398
kv280:2:1	.1558	**.0683**	.3498	**.2064**	.7495	.0343	.0373	.1189	**.1157**	.5888
kv280:2:2	.1424	**.0558**	.3178	**.1879**	.7879	.0406	.0508	.1349	.1443	.4659
kv280:3:1	.0334	**.0168**	.1539	**.0979**	.7064	.0343	**.0260**	.1218	**.1179**	.5136
kv280:3:2	.0226	.0313	.1231	.1267	.4370	.0454	**.0443**	.1365	.1388	.3361
kv282:1:1	.1076	**.0412**	.2719	**.1568**	.7913	.0367	.0376	.1300	**.1196**	.3267
kv282:1:2	.0865	**.0484**	.2420	**.1680**	.7437	.0278	.0474	.1142	.1436	.2072
kv282:1:3	.1230	**.0717**	.2595	**.2172**	.6504	.1011	**.0463**	.2354	**.1575**	.8075
kv283:1:1	.0283	**.0263**	.1423	**.1067**	.7007	.0183	.0202	.0918	.1065	.3033
kv283:1:2	.0371	**.0221**	.1611	**.1072**	.7121	.0178	**.0171**	.0932	.0960	.4391
kv283:3:1	.0404	**.0149**	.1633	**.0928**	.8247	.0225	**.0183**	.1024	**.0954**	.4997
kv283:3:2	.0424	**.0245**	.1688	**.1156**	.6881	.0256	.0308	.1085	.1184	.2574
kv332:2	.0919	.0948	.2554	**.2499**	.3876	.0286	.0630	.1110	.1767	.2389
WMean	.0486	**.0360**	.1757	**.1370**	.6200	.0282	.0326	.1108	.1202	.3600

worse: in only 6 cases is learning better than no learning (for both MSE and MAE). This can partly be explained by the fact that quadratic functions may not be as reasonable a model class for expressive timing as it has been believed in musicology (see also [19]). On some pieces DISTALL is able to predict expressive curves which are surprisingly close to those actually produced by the pianist — witness, e.g., the correlation of 0.8247 in kv283:3:1 for dynamics.[6]

On the other hand, DISTALL performs poorly on some pieces, especially on those that are fully different in character from all other pieces in the training set (e.g. correlation of 0.2389 by kv332:2 for tempo).

4.3 DISTALL vs. RIBL

In order to put these results into context, we present a direct comparison with RIBL [5]. Since RIBL is not publicly available, in the experiments of this section we used a self-implemented version. While the *leave-one-piece-out* cross-validation procedure stayed the same as in the previous section, for the direct comparison of the two learners we chose a somewhat different evaluation procedure. Rather than comparing entire composite performance curves, we compare

[6] Such a high correlation between predicted and observed curves is even more surprising taking into account that kv283:3:1 is a fairly long piece with over 90 hierarchically nested phrases containing over 320 melody notes.

Table 3. Direct comparison between RIBL and DISTALL. The table shows absolute numbers and percentages of the phrases where the predictions of both learners are equal, and where one learner is closer to the actual phrase shape than the other. The parameters of both learners are $k = 1$ and $depth = 3$. All attribute and predicate weights used by both learners (and especially capacities of the distance networks used by DISTALL) are set to 1.

	dynamics			tempo		
	MSE	MAE	CORR	MSE	MAE	CORR
equal	901 (73%)	901 (73%)	901 (73%)	901 (73%)	901 (73%)	901 (73%)
RIBL closer	153 (12%)	157 (12%)	165 (13%)	155 (12%)	159 (13%)	153 (12%)
DISTALL closer	188 (15%)	184 (15%)	176 (14%)	186 (15%)	182 (14%)	188 (15%)

the learners' performance directly at the phrase level. That is, for each phrase, we compare RIBL's and DISTALL's predictions with the 'real' phrase shapes, i.e. those obtained by decomposing tempo and dynamics curves played by the pianist. We then check whose prediction was closer to the actual shape (again in terms of MSE, MAE and correlation). That gives us much more test instances (1240 phrases instead of 16 pieces) and thus more detailed insight into differences between the algorithms. The results are given in Table 3.

A look at Table 3 reveals that RIBL and DISTALL agree in more than 70% of all cases. For those cases where the predictions differ, DISTALL's prediction are closer to the pianist's shapes in more cases than vice versa in terms of each error measure (i.e., lower MSE, lower MAE, higher correlation), although the difference is not large. Since DISTALL solves the maximal flow minimal weight problem hierarchically, expanding unknown weights into transport network problems at a lower level, the number of elements in each network — and accordingly, the computational complexity of solving the network distance problem — is kept low, making DISTALL's runtimes only slightly higher than RIBL's (for the presented experiment involving 1240 phrases and 16-fold cross-validation, the approximate runtimes on our system are 4h and 5h for RIBL and DISTALL, respectively).

4.4 DISTALL vs. Propositional k-NN

One desirable property of relational learners is performing as well on propositional data as the 'native' propositional learners [5, 4]. Being generalizations of the propositional k-NN, both DISTALL and RIBL share this property. It would however be interesting to compare the performance of DISTALL, given the relational data representation, with the performance of the standard propositional k-NN,[7] since it has been shown that a richer relational representation need not always be a guarantee for better generalization performance [4]. We can

[7] For a detailed study on the performance of the straightforward propositional k-NN on the same learning problem see [15].

Table 4. Direct comparison between standard k-NN and DISTALL. The table shows absolute numbers and percentages of the phrases where the predictions of both learners are equal, and where one learner is closer to the actual phrase shape than the other. Both learners use one nearest neighbor for prediction. DISTALL's depth parameter is set to $depth = 3$. Capacities of the distance networks are set to 1.

	dynamics			tempo		
	MSE	MAE	CORR	MSE	MAE	CORR
equal	697 (56%)	697 (56%)	697 (56%)	697 (56%)	697 (56%)	697 (56%)
prop. closer	240 (19%)	243 (20%)	259 (21%)	249 (20%)	252 (20%)	241 (19%)
DISTALL closer	305 (25%)	302 (24%)	286 (23%)	296 (24%)	293 (24%)	304 (25%)

represent phrases in propositional logic by describing each phrase in the data set with the attributes $A1, A2, ...$ from the predicate $phrCont(Id, A1, A2, ...)$ together with the 'target' polynomial coefficients *Coeffs* from the predicate $phrShape(Id, Coeffs)$. By doing so we lose information about hierarchical relations between phrases and obtain an attribute-value representation which can be used by the k-NN algorithm. The results of the direct comparison are shown in Table 4.

Experimental setup and evaluation were the same as in the previous section. Again, both learners predict the same shapes in a lot of cases — more than 55% of the test set. For the second half however, DISTALL outperforms its propositional counterpart in terms of all error measures.

5 Conclusion

We have presented a complex learning task from the domain of classical music: learning to apply musically 'sensible' tempo and dynamics variations to a piece of music, at different levels of the phrase hierarchy. The problem was modelled as a multi-level decomposition and prediction task and attacked via relational instance-based learning. A new structural similarity measure, based on a combination of two existing techniques, was presented and implemented in a learning algorithm named DISTALL. An experimental analysis showed that the algorithm produces slightly better results in this domain than the related algorithm RIBL, and that it is more effective than a propositional k-NN learner on the music task.

In addition to such quantitative evaluations, listening to the performances produced by the learner provides additional qualitative insight. Some of DISTALL's performances — although being the result of purely automated learning with no additional knowledge about music — sound indeed musically sensible. We hope to demonstrate some interesting sound examples at the conference. On the other hand, some elementary musical 'errors' clearly show that fully automated expressive performance at the level of human pianists is still far from being feasible, and certainly not with such a knowledge-free approach.

In [5] it was argued that attribute weighting is an important issue in propositional IBL and even more important in a relational setting. DISTALL currently lacks a data-driven predicate/attribute weighting module. We suspect that the quality of our similarity measure would become even more evident it the predicates were weighted adequately (e.g., if the detailed structure of the phrases were given higher weight).

Future work with DISTALL could also have an impact on musicology. The poor results in the tempo domain (see section 4.2) suggest that other types of approximation functions may be worth trying, which might lead to better phrase-level tempo models. We also plan to try to empirically prove that a concert pianist plays similar phrases in similar ways (by showing a high correlation between expressive shapes for those phrases for which DISTALL suggests high similarity). One could also turn the question around and take the high correlation between expressive shapes attached to phrases which are suggested to have high similarity as a reliability proof of DISTALL's similarity measure.[8]

Acknowledgments

This research is supported by a START Research Prize by the Austrian Federal Government (project no. Y99-INF) and by the European project HPRN-CT-2000-00115 (MOSART). The Austrian Research Institute for Artificial Intelligence acknowledges basic financial support from the Austrian Federal Ministry for Education, Science and Culture. Thanks to Werner Goebl for performing the harmonic and phrase structure analysis of the Mozart sonatas.

References

1. Bisson, G. (1992). Learning in FOL with a Similarity Measure. In *Proceedings of the 10th AAAI*, 1992.
2. De Raedt, L. (1992). *Interactive Theory Revision: an Inductive Logic Programming Approach.* Academic Press.
3. Duda, R. and Hart, P. (1967). *Pattern Classification and Scene Analysis.* New York, NY: John Wiley & Sons.
4. Dzeroski S., Schulze-Kremer, Heidtke K.R., Siems K., Wettschereck D., and Blockeel H. (1998). Diterpene structure elucidation from 13C NMR spectra with inductive logic programming. *Applied Artificial Intelligence: Special Issue on First-Order Knowledge Discovery in Databases*, 12(5):363-384, July August 1998.
5. Emde, D. and Wettschereck, D. (1996). Relational Instance-Base Learning. In *Proceedings of the Thirteen International Conference on Machine Learning (ICML'96)*, pages 122-130. Morgan Kaufmann, San Mateo.
6. Helft, N. (1989). Induction as nonmonotonic inference. In *Proceedings of the 1st international Conference on Principles of Knowledge Representation and Reasoning*, pages 149-156. Kaufmann.

[8] Certainly, it would be more reliable if we had an objective judgment of similarity between two phrases from some independent third party, but there is no procedure known in musicology that would quantify similarity between phrases.

7. Mehlhorn, K. (1984). Graph algorithms and NP-completeness, volume 2 of *Data structures and algorithms*, Springer Verlag.
8. Muggleton, S. H. and Feng C. (1990). Efficient Induction of Logic Programs. In *Proceedings of the First Conference on Algorithmic Learning Theory*,Tokyo.
9. Muggleton, S. and de Raedt, L. (1994). Inductive Logic Programming: Theory and Methods. *Journal of Logic Programming*, 19,20:629-679.
10. Ramon, J. and Bruynooghe, M (1998). A Framework for defining distances between first-order logic objects. In D. Page, (ed.), *Proceedings of the 8th International Conference on Inductive Logic Programming*, volume 1446 of Lecture Notes in Artificial Intelligence, pages 271–280. Springer-Verlag.
11. Ramon, J. and Bruynooghe, M. (2000). A polynomial time computable metric between point sets. Report CW 301, Department of Computer Science, K.U.Leuven, Leuven, Belgium.
12. Rouveirol, C. (1992). Extensions of inversion of resolution applied to theory completion. In S. Muggleton, (ed.), *Inductive Logic Programming*. Academic Press, London.
13. Sebag, M. and Rouveirol, C. (1997). Tractable induction and classification in FOL Via Stochastic Matching. In *Proceedings of IJCAI-97*. Morgan Kaufmann.
14. Todd, N. McA. (1992). The Dynamics of Dynamics: A Model of Musical Expression. *Journal of the Acoustical Society of America* 91, 3540–3550.
15. Tobudic, A. and Widmer, G. (2003). Playing Mozart Phrase By Phrase. In *Proceedings of 5th International Conference on Case-Based Reasoning (ICCBR'03)*, Trondheim, Norway. Berlin: Springer Verlag..
16. Widmer, G. (2002). Machine Discoveries: A Few Simple, Robust Local Expression Principles. *Journal of New Musical Research* 31(1), 37–50.
17. Widmer, G. (2003). Discovering Simple Rules in Complex Data: A Meta-learning Algorithm and Some Surprising Musical Discoveries. *Artificial Intelligence* 146(2), 129–148.
18. Widmer, G., Dixon, S., Goebl, W., Pampalk, E., and Tobudic, A. (2003). In Search of the Horowitz Factor. *AI Magazine*, in press.
19. Widmer, G. and Tobudic, A. (2003). Playing Mozart by Analogy. *Journal of New Musical Research*, in press.