Case-based Relational Learning of Expressive Phrasing in Classical Music

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

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

Abstract. An application of relational case-based learning to the task of expressive music performance is presented. We briefly recapitulate the relational case-based learner DISTALL and empirically show that DISTALL outperforms a straightforward propositional k-NN on the music task. A set distance measure based on maximal matching - incorporated in DISTALL - is discussed in more detail and especially the problem associated with its 'penalty part': the distance between a large and a small set is mainly determined by their difference in cardinality. We introduce a method for systematically varying the influence of the penalty on the overall distance measure and experimentally test different variants of it. Interestingly, it turns out that the variants with high influence of penalty clearly perform better than the others on our music task.

Keywords: relational case-based learning, case-based reasoning, music

1 Introduction

Case-based learning for expressive performance has been demonstrated before in the domain of expressive phrasing in jazz [2, 6], where the promise of CBR was shown, but the evaluation was mostly qualitative and based on relatively small numbers of phrases. In previous work we presented what to our knowledge is the first large-scaled quantitative evaluation of case-based learning for expressive performance (against a high-class concert pianist) [12]. An case-based learning system was presented which was able to recognize performance patterns at various levels of musical abstraction (hierarchically nested phrases) and apply them to new pieces (phrases) by analogy to known performances. While the experimental results in this difficult domain were far from being truly satisfying, some of the resulting expressive performances sounded indeed musically sensible.[3]

One obvious limitation of the presented system was the simple attribute-value representation used to characterize phrases, which did not permit the

[3] A recording of one of the system's expressive performances won second prize at the International Computer Piano Performance Rendering Contest (RENCON'02) in Tokyo in September 2002, behind a rule-based rendering system that had been carefully tuned by hand.

learner to refer to details of the internal structure of the phrases, nor to their broader musical *context*. These problems can be partly overcome by applying a relational representation for phrase description. We briefly recapitulate the relational case-based learning algorithm DISTALL [13], and show how it is indeed able to outperform the straightforward propositional k-nearest neighbor on the 'expressive-performance' task. In the relational setting, where examples are mostly represented as sets of facts, the distance measure between two sets of elements is a crucial part of each case-based learner. DISTALL's rather intuitive set distance measure based on maximal matching (first proposed in [9]) was shown to work well on a number of tasks [10, 13]. One of the problems with this set distance measure is that the distance between two sets with (largely) different cardinalities is mainly determined through a 'penalty' - the difference in cardinalities of both sets. It is thus not clear if the sets should be first scaled in some way to the approximately same cardinalities, e.g. by weighting their elements. In this work we present an experimental study of the impact of varying influence of penalty to the overall results. It turns out that the retention of a high penalty influence is indeed effective: variants of distance measures with high penalty influence produced clearly better results on our data set than the others.

This 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 first-order logic, and how complex tempo and dynamics curves can be decomposed into well-defined training cases for the case-based learning algorithm. Section 2 is a recapitulation of material already published in [13]. Section 3 briefly describes our relational case-based learner DISTALL. Experimental results achieved with DISTALL on the expressive performance learning task and its comparison with a straightforward propositional k-NN are given in Section 4. Experiments with various degrees of penalty influence for the set distance measure based on maximal matching are reported and discussed in Section 5. Section 6 concludes.

2 Real-World Task: Learning To Play Music Expressively

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 precisely in the written score[4], 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

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

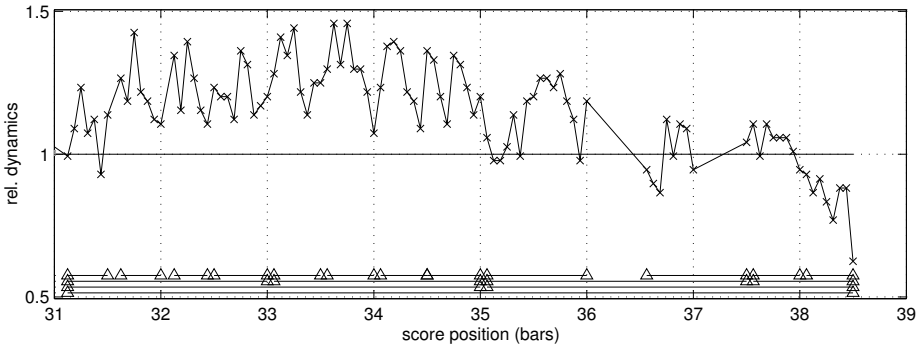

Fig. 1. Dynamics curve (relating to melody notes) of performance of Mozart Sonata KV.279, 1^{st} movement, mm. 31–38, by a Viennese concert pianist.

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[5]). 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 concert pianists. The heart of our system, the relational case-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 cases for the learner. Finally, we discuss

[5] 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.

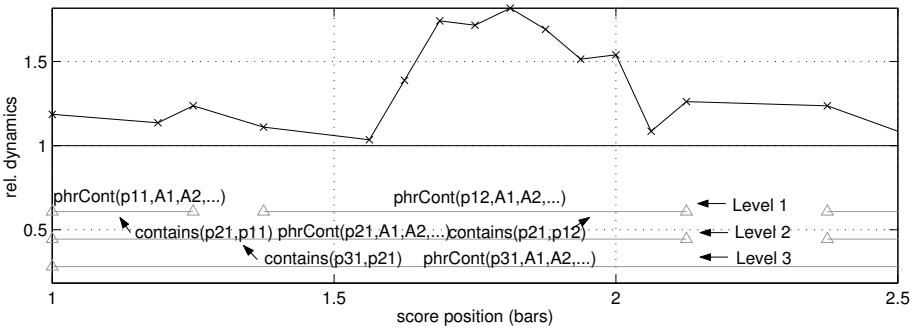

Fig. 2. Phrase representation used by our relational case-based learning algorithm.

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 first-order logic

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.

Predicates *phrCont* and *contains* encode a partial description of the musical score. 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 cases: 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.

2.3 Combining multi-level phrase 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

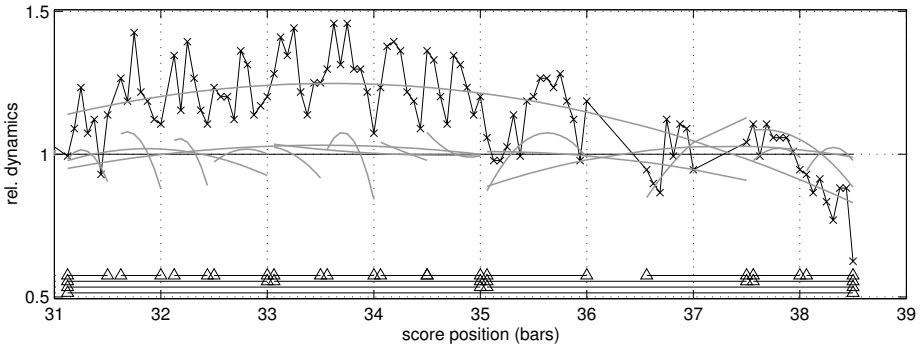

Fig. 3. Multilevel decomposition of dynamics curve of performance of Mozart Sonata K.279:1:1, mm.31-38.: original dynamics curve plus the second-order polynomial shapes giving the best fit at four levels of phrase structure

values) and then successively multiplies the current value by the multi-level phrase predictions.

3 DISTALL And Set Distance Measure Based on Maximal Matching

The following section gives a brief overview of the relational case-based learner DISTALL. The interested reader is referred to [13] for a more detailed description.

3.1 The relational case-based learner DISTALL

DISTALL can be regarded as the 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. The same idea is employed by DISTALL. Figure 4 depicts the basic principle of DISTALL (DIstance on SeTs of Appropriate Linkage Level).

In the example, we are interested in the distance between objects Ob_1 and Ob_2. It is calculated as the distance between two sets of FOL-literals: between the set of all literals from the background knowledge also containing object identifier Ob_1 as one of the arguments, and the set of all literals containing object identifier Ob_2. Most elements of these two sets will typically be literals which describe basics properties of objects Ob_1 and Ob_2 (putting this in the context of our music example from section 2.1, objects Ob_1 and Ob_2 would

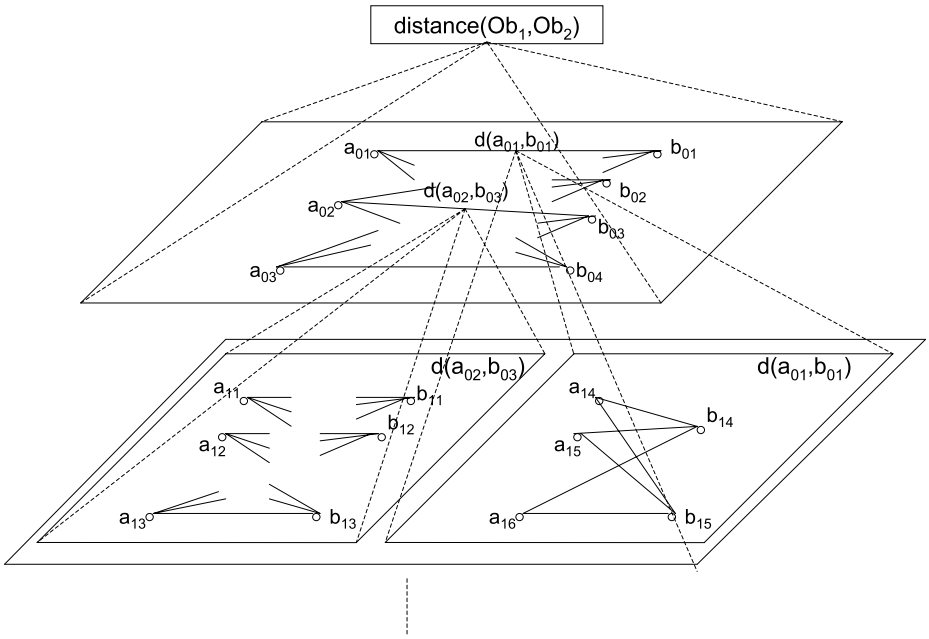

Fig. 4. Basic principle of DISTALL's similarity measure

represent two phrases and literals describing their basic musical properties would be predicates *phrCont*). The distance between such literals can be defined in a straightforward way, e.g., as the Euclidean or Manhattan distance defined over the literals' arguments (or set to 1 if the literals have different functors). But there may also be literals which state relations of objects Ob_1 and Ob_2 to some other objects (these would be *contains* predicates in our music example). In the figure 4 these are literals a_{01}, a_{02}, b_{01} and b_{03}. The distance between such literals is calculated as the set distance between sets of literals from the background knowledge which have the same object identifiers as one of the arguments. The procedure continues recursively, until a user defined depth is reached. At the lowest level, the distance between objects is calculated as the distance between discrete values. From the above it is apparent that a distance measure between *sets of elements* is essential for our learner. It is discussed in the following.

3.2 Set distance based on maximal matching

Given a normalized (i.e. in interval [0,1]) distance measure between individual points, the distance between two sets of points A and B based on maximal matching is proposed in [9] as:

$$d^m(A,B) = \min_{r \in MM(A,B)} \left\{ \sum_{(x,y)\in r} d(x,y) + \frac{|\#A - \#B|}{2} \right\} \quad (1)$$

where $MM(A,B)$ is the set of all possible maximal matchings between A and B, and $d(x,y)$ are distances between individual matched points.

Stated informally, one wants to maximally match elements from one set with the elements of the other and achieve the minimal possible distance. A penalty for each element of one set which does not match to any element of the other is also added (second term in the formula). In [9] it has been shown that such a metric satisfies the reflexivity, similarity and triangle inequality relations and can be calculated in time polynomial in $\#A$ and $\#B$.[6]

Although rather intuitive, the set distance measure based on maximal matching has two main weaknesses. The first is its computational infeasibility. As stated above, examples in FOL are describes as sets of facts, where, for typical learning tasks, it is not uncommon for examples to have fifty or more facts. Computing the distance between such two examples would have a time complexity of 100^3. For IBL and clustering tasks distances between each pair of examples have to be calculated, with databases typically containing tens of thousands of examples. For this reason DISTALL does not apply the set distance measure directly, but splits initial sets into hierarchically nested subsets (see Figure 4). The computational cost is kept small by applying the distance measure to subsets only, each having a small number of elements (see [13] for details).

The second problem of the presented distance measure is that the distance between a large and a small set is largely determined by their difference in cardinality, i.e. if $A >> B$, then $d^m(A,B) \approx |\#A - \#B|/2$. In order to avoid that, methods for weighting elements in sets are proposed (see [9] for details): E.g. one could assign a weight $W[A](a)$ to each element a in smaller set A and define a size of a set (or virtual cardinality) as: $size(A) = \sum_{a\in A} W[A](a)$. If the weights are chosen appropriately, the sets can be scaled to the same virtual cardinality thus reducing (or eliminating) the penalty part in formula 1. Still it is not clear if the set distance measure should be more influenced by the 'point' distances between already matched set points or by the fact that there is a number of points in one set which do not match any of the points in the other set. For example, suppose there are three families A, B_1 and B_2. Family A consists of mother, father and a child, the same as family B_1. It turns out that each member of family A is rather dissimilar from corresponding person from family B_1. Family B_2 has five children but mother,father and one of the children from B_2 are rather similar to the appropriate members from A. Which family, B_1 or B_2 is more similar to family A?

[6] Technically, finding a maximal matching and achieving the minimal possible distance between A and B is accomplished by finding a solution to the maximum flow minimal weight problem of adequately constructed transport network (see [7] for more information on the concept of transport networks) and has complexity cubic in $\#A$ and $\#B$.

This question is subject of our experiments in section 5. We gradually vary the influence of the penalty part to the overall distance between two sets (from 0 to the full influence given by equation 1) and test the learning performance of each variant of the distance measure on our music task.

4 Experiments: Relational vs. Propositional Learning

In the following we present detailed empirical results achieved with DISTALL on a complex real-world dataset derived from piano performances of classical music. We also provide a comparison with a straightforward propositional k-NN. Note that in the experiments we present in this section we do not intend to learn performance curves as close to the pianist's curves as possible (see [12] for this type of problem). Rather, we want to investigate if employing a richer relational representation adds some performance gain compared with a simpler propositional representation.

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.

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 experiment setup is similar to the one already published in [13]. The evaluation procedure is somewhat different. What was compared in [13] was the tempo or dynamics curve produced by the learner with the curve corresponding to the pianist's actual performance. This is somewhat unfair, since the learner was given not the actual performance curves but an *approximation*, namely the polynomials fitted to the curve at various phrase levels. Correctly predicting these is the best the learner could hope to achieve. Thus, in this work we use the following performance measures: the *mean squared error* of the system's prediction on the piece relative to the *approximation* curve – i.e., the

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	53	30	12	3
kv283:3:2 fast 3/8	558	79	47	19	6
kv332:2 slow 4/4	477	49	23	12	4
Total:	5506	714	365	165	66

curve implied by the three levels of quadratic functions – of 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 'approximated' curve. 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 approximation of the pianist's 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 worse: in only 6 cases for MSE and 10 for MAE is learning better than no learning.

On some pieces DISTALL is able to predict expressive curves which are surprisingly close to the approximations of the pianist's ones — witness, e.g.,

Table 2. Results, by sonata sections, of cross-validation experiment with DISTALL (*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	.0341	**.0173**	.1571	**.0929**	.7093	.0161	.0189	.0879	**.0820**	.4482
kv279:1:2	.0282	.0314	.1394	**.1243**	.6162	.0106	.0151	.0720	.0799	.4691
kv280:1:1	.0264	**.0143**	.1332	**.0895**	.7289	.0136	**.0062**	.0802	**.0526**	.7952
kv280:1:2	.0240	.0380	.1259	.1363	.4394	.0125	.0160	.0793	**.0752**	.4983
kv280:2:1	.1534	**.0655**	.3493	**.2022**	.7569	.0310	.0326	.1128	**.1025**	.6360
kv280:2:2	.1405	**.0534**	.3170	**.1854**	.7951	.0323	.0427	.1269	.1293	.5206
kv280:3:1	.0293	**.0125**	.1452	**.0809**	.7575	.0188	**.0104**	.0953	**.0629**	.6971
kv280:3:2	.0187	.0274	.1124	.1151	.4816	.0196	**.0157**	.1033	**.0884**	.5862
kv282:1:1	.0956	**.0297**	.2519	**.1228**	.8367	.0151	.0172	.0905	**.0700**	.4718
kv282:1:2	.0781	**.0397**	.2277	**.1436**	.7839	.0090	.0284	.0741	.0974	.3435
kv282:1:3	.1047	**.0520**	.2496	**.1867**	.7134	.0938	**.0388**	.2236	**.1269**	.8400
kv283:1:1	.0255	**.0236**	.1379	**.0985**	.7377	.0094	.0115	.0664	.0756	.4106
kv283:1:2	.0333	**.0183**	.1560	**.0948**	.7506	.0097	**.0092**	.0691	**.0651**	.5860
kv283:3:1	.0345	**.0099**	.1482	**.0715**	.8818	.0116	**.0077**	.0696	**.0534**	.6847
kv283:3:2	.0371	**.0192**	.1572	**.1002**	.7358	.0100	.0153	.0745	.0757	.4099
kv332:2	.0845	.0869	.2476	**.2398**	.4059	.0146	.0492	.0718	.1498	.2582
WMean	.0437	**.0310**	.1664	**.1225**	.6603	.0141	.0182	.0811	.0823	.5089

the correlation of 0.88 in kv283:3:1 for dynamics.[7] On the other hand, DISTALL performs poorly on some pieces, especially on those that are rather different in character from all other pieces in the training set (e.g. correlation of 0.26 by kv332:2 for tempo).

4.3 DISTALL vs. propositional *k*-NN

One desirable property of relational learners is performing as well on propositional data as the 'native' propositional learners [4, 5]. Being generalizations of the propositional k-NN, DISTALL shows this property. It is however interesting to compare the performance of DISTALL, given the relational data representation, with the performance of the standard propositional k-NN,[8] since it has been shown that a richer relational representation need not always be a guarantee for better generalization performance [4]. We can represent phrases in propositional logic by describing each phrase in the data set with the attributes $A1, A2, ...$

[7] 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.

[8] For a detailed study where propositional k-NN is optimized on our learning problem see [12].

Table 3. Comparison between standard k-NN and DISTALL. The table shows weighted mean errors over all test pieces. Measures subscripted with D refer to the 'default' (mechanical, inexpressive) performance, those with L to the performance produced by the learner. The results for DISTALL are repeated from table 2 (last row).

	dynamics					tempo				
	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$
propositional k-NN	.0437	.0335	.1664	.1309	.6369	.0141	.0177	.0811	.0878	.4628
DISTALL	.0437	.0310	.1664	.1225	.6603	.0141	.0182	.0811	.0823	.5089

Table 4. Comparison between propositional k-NN and DISTALL at the level of phrases. 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 learner use one nearest neighbor for prediction.

	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%)

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. Table 3 shows the performance of k-NN on our learning task in terms of weighted mean errors over all test pieces. The equivalent results for DISTALL are repeated from table 2 (last row).

DISTALL performs better than propositional k-NN in both domains (reducing the errors and increasing the correlation), the only exception being MSE for tempo. Although meaningful, comparing such high-level error measures is somewhat unfair. The actual task for both learner is to predict elementary phrasal shapes and not the composite performance curves. One mispredicted shape at the highest level can 'ruin' the whole composite curve even if all other shapes at lower levels are predicted perfectly. For this reason it is instructive to compare the learners' performance directly at the phrase level.

The following experiment was performed. The *leave-one-piece-out* cross-validation procedure stayed the same as in the previous section, but now we compare predictions of propositional k-NN and DISTALL 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 cases

Table 5. Comparison between variants of set distance measure based on maximal matching with different influences of penalty part (see section 3.2). The table shows weighted mean errors over all test pieces. Measures subscripted with D refer to the 'default' (mechanical, inexpressive) performance, those with L to the performance produced by the learner.

	dynamics					tempo				
penInfl	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$	MSE_D	MSE_L	MAE_D	MAE_L	$Corr_L$
0	.0437	.0376	.1664	.1391	.6091	.0141	.0303	.0811	.1186	.3203
0.2	.0437	.0378	.1664	.1374	.6297	.0141	.0286	.0811	.1107	.3313
0.4	.0437	.0347	.1664	.1306	.6481	.0141	.0253	.0811	.1002	.4043
0.6	.0437	.0350	.1664	.1292	.6514	.0141	.0223	.0811	.0936	.4563
0.8	.0437	.0319	.1664	.1226	.6687	.0141	.0200	.0811	.0865	.4736
1	.0437	.0310	.1664	.1225	.6603	.0141	.0182	.0811	.0823	.5089

(1240 phrases instead of 16 pieces) and thus more detailed insight into differences between the algorithms. The results are given in table 4.

A first look at table 4 reveals that both learners predict the same shapes in a lot of cases — more than 55% of the test set. For the second half however, DISTALL predicts shapes which are closer to the pianist's (i.e., lower MSE and MAE, higher correlation) in more cases than vice versa.

5 Expreriment: Influence of the Mismatch Penalty on the Overall Set Distance Measure

In this section we experimentally examine the influence of the penalty part of the set distance (see section 3.2). We introduce a parameter penalty influence (*penInfl*) which allows us to control the influence of the penalty part in the formula 1 as follows:

The parameter *penInfl* can be varied in the range [0,1]. Before computing the set distance, the elements of the smaller set are weighted according to the formula $w = R + (1 - R) * penInfl$, where $R = max(\#A, \#B)/min(\#A, \#B)$. By setting $penInfl = 0$ (we want to neglect the penalty part), each element of the smaller set is assigned a weight $w = R$. In this case both sets have the same 'virtual cardinality' and each matching of the smaller set to one element of the bigger set is weighted with $w = R$. In setting $penInfl = 1$ elements of the smaller set are weighted $w = 1$ and formula 1 applies. For values of $penInfl$ between 0 and 1, the weights are linearly spread in the range (R,1) (e.g. if $\#A = 2$, $\#B = 4$ and $penInfl = 0.5$, each element of A would be weighted $w = 1.5$ and 'virtual cardinality' of A would be 3).

For each value of *penInfl* in the range [0,1] with step size 0.2 we repeated the same *leave-one-piece-out* cross-validation experiment from section 4.2. The learning performance of each variant of the distance measure in terms of weighted mean errors over all test pieces is given in table 5. Interestingly, it turns out that

the variants of the distance measure with high penalty influence perform clearly better on our learning task than those with reduced influence of penalty. The learners' performances in terms of each error measure (lower MSE and MAE, higher correlation) get monotonically better with increasing values of parameter *penInfl*. Although the learning performance increases in both domains, the gain in the tempo domain is more dramatic (e.g. mean correlation of 0.36 vs. 0.24 for variants $penInfl = 1$ and $penInfl = 0$ respectively). It seems that variants with high penalty basically 'filter out' phrases with different structure (e.g. phrases with a (largely) different number of smaller phrases) and choose the nearest phrase based on 'fine tuning' from the subset of phrases with approximately the same structure. Still it is not clear if this strategy is effective on datasets with even larger differences in cardinalities between examples.

6 Conclusion

We have presented an application of cased-base reasoning on 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. We showed how hierarchically nested phrases can be naturally described in first-order logic and briefly presented the relational case-based learner DISTALL which can be seen as a generalization of k-NN learner for data described in FOL. Experimental analysis showed that our approach is in general viable. In addition to 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. Experimental results also showed that DISTALL outperforms a straightforward, propositional k-NN learner on the music task.

The set distance measure based on maximal matching, incorporated in DISTALL, was discussed in more detail. Specially, the problem of high influence of the penalty by assigning the distance to sets with largely different cardinalities was discussed. We presented a way to systematically vary the influence of the penalty part on the overall set distance measure. Interestingly, experimental results showed that variants of the set distance measure with high penalty influence perform better than those with reduced influence of penalty. Future experiments should show if reducing influence of penalty is more effective on datasets with examples with larger differences in cardinalities than our music dataset.

Future work with DISTALL could also have an impact on musicology. The rather 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.

Acknowledgments

This research is supported by a START Research Prize by the Austrian Federal Government (project no. Y99-INF). The Austrian Research Institute for Artificial Intelligence acknowledges basic financial support by the Austrian Federal Ministry for Education, Science, and Culture, and the Federal Ministry of Transport, Innovation, and Technology. 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. Arcos, J.L. and López de Mántaras (2001). An Interactive CBR Approach for Generating Expressive Music. *Journal of Applied Intelligence* 14(1), 115–129.
3. De Raedt, L. (1992). *Interactive Theory Revision: an Inductive Logic Programming Approach*. Academic Press.
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. López de Mántaras, R. and Arcos, J.L. (2002). AI and Music: From Composition to Expressive Performances. *AI Magazine* 23(3), 43–57.
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. 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.
10. 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.
11. Rouveirol, C. (1992). Extensions of inversion of resolution applied to theory completion. In S. Muggleton, (ed.), *Inductive Logic Programming*. Academic Press, London.
12. 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.
13. Tobudic, A. and Widmer, G. (2003). Relational IBL in Music with a New Structural Similarity Measure. In *Proceeding of 9th International Conference on Inductive Logic Programming (ILP'03)*. Szeged, Hungary. Berlin: Springer Verlag.
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. Widmer, G. and Tobudic, A. (2003). Playing Mozart by Analogy: Learning Multi-Level Timing and Dynamics Strategies. *Journal of New Musical Research*, 32(3), 259-268.