Melodic Clustering: Motivic Analysis of Schumann's *Träumerei*

Emilios Cambouropoulos and Gerhard Widmer

Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Vienna, Austria
{emilios, gerhard}@ai.univie.ac.at

Abstract

In this paper a formal model will be presented that attempts to organise melodic segments into 'significant' musical categories (e.g. motives). Given a segmentation of a melodic surface, the proposed model constructs an appropriate representation for each segment in terms of a number of attributes (these reflect melodic and rhythmic aspects of the segment at the surface and at various abstract levels) and then a clustering algorithm (the *Unscramble* algorithm) is applied for the organisation of these segments into 'meaningful' categories. The proposed clustering algorithm automatically determines an appropriate number of clusters and also the characteristic (or defining) attributes of each category. As a test case this computational model has been used for obtaining a motivic analysis of Schumann's *Träumerei*.

Keywords: motivic analysis, musical categories, similarity, clustering

1. Introduction

Making sense of a musical work means being able to break it down into simpler components and to make associations between them. Musical analysis is geared towards providing the 'resolution of a musical structure into relatively simpler constituent elements, and the investigation of those elements within that structure' (Bent, 1980: 340).

Paradigmatic analysis (Nattiez 1975, 1990) is an analytic methodology that aims at providing a segmentation of a musical surface and an organisation of the derived musical segments into 'significant' musical categories (or *paradigms)*. This methodology relies heavily on a notion of musical similarity and aims at assisting a human analyst to explicate his/her own similarity criteria for obtaining a certain analysis.

In relation to the above, the current paper addresses the following problem: given a segmentation of a melodic surface, how can a computational system be developed that can arrive at a 'plausible' clustering of the given segments and at the same time provide explicit descriptions of each category/cluster?

In the problem description stated in the previous paragraph, melodic segmentation is taken to be a pre-requisite. This is a simplification so that we can focus on the categorisation problem. As has been shown elsewhere (Cambouropoulos 1998) segmentation is strongly associated with similarity and categorisation, e.g. a 'good' categorisation of musical segments may suggest a 'preferred' segmentation that may actually override local perceptual grouping indications. The relation between segmentation and categorisation is a very complex topic; a study of this relation is currently in progress and some results will appear in a forthcoming paper.

There has been a number of attempts to use clustering techniques for organising melodic segments into paradigms. A brief survey and comparison of some existing formal models is presented in (Anagnostopoulou et al., 1999). Some difficulties of existing clustering approaches for musical purposes will be addressed later on in this text.

The main topics that will be discussed in this paper are: representation of melodic segments, pattern-processing techniques, clustering techniques and intensional description of emerging clusters. As a test case the proposed computational model will used for obtaining a motivic analysis of the melody of R. Schumann's *Träumerei* (this is the 7th piece from *Kinderszenen, op. 15).*

2. Motivation

The construction of a sophisticated computational system for organising melodic segments into 'meaningful' categories, apart from theoretical interest for the domains of musical analysis and musical cognition, provides a potentially very useful tool for a number of applications such as interactive composition and performance, musical data indexing and retrieval, expressive machine performance, and so on. The main argument is that, the more sophisticated musical computer applications one envisages to develop, the more ‘understanding’ of musical structure a computational system should have.

As the current study is undertaken within the framework of a project titled ‘Artificial Intelligence models of musical expression’ we will elaborate on the usefulness of structural analysis for the study of expressive features of performance.

The aim of the project on musical expression is to study expressive features of musical performance (at this stage tempo and loudness micro-variations) and to develop learning algorithms that can induce general rules of expressive music performance from examples of real performances by human musicians.

In a double experiment reported by G.Widmer (1996) expression rules were induced from real melodic performances a) at the note-level and b) at the structural level (motivic and phrase level). The results obtained indicated that learning was significantly improved when structural aspects of the melody were taken into account. It was therefore strongly suggested that musical structural information is indispensable for the development of a machine model of musical expression.

In a detailed study of expressive features in Schumann’s *Träumerei* Bruno Repp (1992) presents at the outset a primarily motivic analysis of the melodic gestures of this piece (see Figure 4). This analysis is an intuitive analysis performed by a human and is used throughout the main body of the paper for studying the micro-timing variations of 28 different performances of *Träumerei*. Again, metric and rhythmic structural information is thought to be paramount for the study of musical expression.

In this paper, we have used the melody of *Träumerei* as a test case to study the proposed clustering model. The segmentation provided in Repp’s paper is taken as an input in the current study. The machine-generated results obtained herein will be compared to and tested against the human analysis provided in Repp’s study.

We believe that research on musical expression can benefit from the development of more sophisticated machine models of musical structure but we hypothesise that the reverse may also be true, namely that the study of musical expression may provide cues to structural analytic models in order to disambiguate complex multi-faceted analytic results and to fine-tune all sorts of parameters and preference rules in such models. This will be a topic of future investigations.

3. Representation of melodic segments

Let us suppose that at the lowest level of representation each melodic segment is represented as a string of notes. The question that arises is whether this representation is sufficient or whether further processing of the melodic segments is necessary before presenting them as input to a clustering algorithm.

The reply to the above question depends on what notion of musical similarity one intends to employ as a basis for pattern processing and clustering. A more detailed discussion on this topic can be found in (Cambouropoulos et al, 1999). Here, we will only present two main strategies:

a) Two musical segments - represented as vectors of notes or intervals - are construed as being similar if they share at least a certain number of their component elements (notes or intervals); approximate pattern-matching algorithms and the *edit distance* are commonly used in this approach (e.g. Rolland et al, 2000).

b) Two musical segments - represented as vectors of a number of patterns for various parameters at many levels of abstraction - are construed as being similar if they share at least a certain number of patterns at the surface level or reductions of it; this strategy requires more sophisticated representation of musical segments; exact pattern-matching techniques and the *hamming distance* can be used in this approach.

Figure 1 illustrates these possibilities with a simple melodic example. Of course, there can be all sorts of variations or even combinations of the above two main strategies.

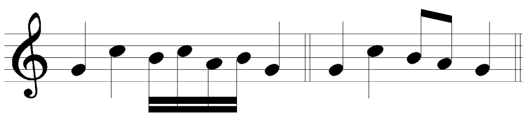


Figure 1. These two melodic segments may be construed as similar either because five of their pitches and onsets match at the surface level (approximate matching) or because they match exactly at the reduced eighth-note metric level.

In this study a strategy very close to the second approach given above has been employed. For each melodic segment of *Träumerei* the following pattern attributes are computed (see example in Figure 2):

a. Surface level
  • Pitch-intervals: exact (p_ex), scale-steps (p_ss), step-leap (p_sl), step-doublestep-leap (p_sdsl), doublestep-leap (p_dsl), and contour (p_contour).
  • Rhythm: exact durations (r_dur), inter-onset interval (r_ioi), short-long (r_sl), inter-onset interval ratios (r_ratio), shorter-longer-equal (r_sel)

b. Quarter-note level plus notes on boundaries of segments.
  • Pitch-intervals: exact (p_ex), scale-steps (p_ss), step-leap (p_sl), step-doublestep-leap (p_sdsl), doublestep-leap (p_dsl).
  • Rhythm: inter-onset interval (r_ioi), short-long (r_sl)

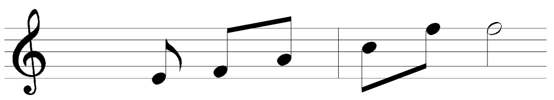
c. Quarter-note level.
  • Pitch-intervals: exact (p_ex), scale-steps (p_ss), step-leap (p_sl), step-doublestep-leap (p_sdsl)

Note: in r_sl, durations longer than a certain value (in this instance a quarter) are represented merely by 'long'.

Further research is necessary to establish a plausible and quite general set of levels of abstraction and reduction. The above set gives gradually less prominence to the more abstract reduction levels and gives a slight preference to parameters related to pitch as it takes into account fewer attributes for gradually more abstract reduction levels and for rhythmic aspects of the melody. The initial set of attributes can be modified by the clustering algorithm - in terms of altering attribute weights - so that more important attribute-values for the specific *context* of the given piece may be highlighted (see next section).

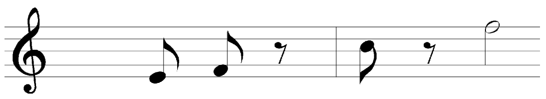
Near-exact matching is an additional technique incorporated in the current proposal. Limited approximate matching is employed in the following manner: two pattern attributes match if they are identical or if they contain a common 'significant' *sub-pattern* (this of course includes the case where one of the two is a sub-pattern of the other). The notion of a 'significant' sub-pattern is controlled by a parameter that defines the relation of the size of the sub-pattern to the two patterns (e.g. sub-pattern should be at least 70% of the size of each pattern). A sub-pattern may consist only of contiguous elements from the given super-pattern. *Exact* matching techniques can be used for determining *near-exact matches*.

Melodic surface



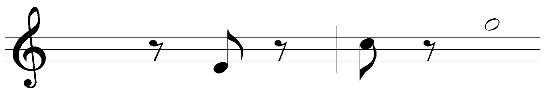
p_ex:	1 - 4 - 3 - 5 - 0
p_ss:	1 - 2 - 2 - 3 - 0
p_sl:	step - leap - leap - leap- 0
p_sdsl:	step - dstep- dstep- leap- 0
p_dsl:	dstep- dstep- dstep- leap- 0
p_contour:	U - U - U - U - 0
r_dur:	1/8- 1/8 - 1/8 - 1/8 - 1/8 - 1/2
r_ioi:	1/8- 1/8 - 1/8 - 1/8 - 1/8 - 1/2
r_sl:	1/8- 1/8 - 1/8 - 1/8 - 1/8 - long
r_ratio:	1 - 1 - 1 - 1 - 4
r_sel:	equal- equal- equal- equal- longer

Notes on quarter beats and on boundaries (reduction)



p_ex:	1 - 7 - 5
p_ss:	1 - 4 - 3
p_sl:	step - leap - leap
p_sdsl	step leap leap
p_dsl:	dstep - leap - leap
r_ioi:	1/8 - 1/4 - 1/4 - 1/2
r_sl:	1/8 - 1/4 - 1/4 - long

Notes on quarter beats (reduction)



p_ex:	7 - 5
p_ss:	4 - 3
p_sl:	leap - leap
p_sdsl	leap leap

Figure 2 Attribute-values for one melodic segment (surface and reductions); each row of alphanumeric symbols constitutes a single attribute-value, i.e. this melodic pattern is represented by a vector of 22 attribute-values – see text for the description of abbreviations.

A very simple example of near-exact matching is presented in Figure 3. This kind of partial matching is valuable in that it allows the use of a smaller set of pattern attributes (e.g. the two patterns in Figure 3 could actually be matched exactly at a high enough reduction level such as the 2/4 metric level). It may be, however, problematic as it incorporates in the representation itself a degree of similarity before any distance metrics are applied by the clustering algorithm.

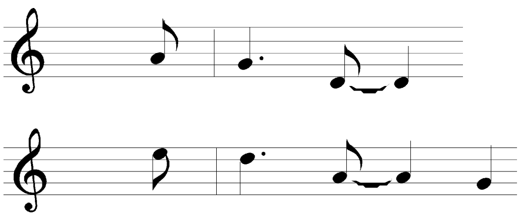


Figure 3 These two melodic segments (from *Träumerei)* share common sub-patterns for both the pitch and rhythm profiles (actually the first is a sub-pattern of the second) - that is, their attribute-values for both pitch and rhythm are *near-exact.*

At the current stage, the computational system requires as input the melodic surface and a selected segmentation, and then it automatically generates the segment attribute-value vectors to be used by the clustering technique.

4. The *Unscramble* clustering algorithm

In this section a brief description will be given of the *Unscramble* clustering algorithm that has been developed primarily for dealing with clustering problems in the musical domain (Cambouropoulos and Smaill, 1997).

The *Unscramble* algorithm is a clustering algorithm which, given a set of objects and an initial set of properties, generates a range of plausible clusterings for a given context. During this dynamically evolving process the initial set of properties is adjusted so that a satisfactory description is generated. There is no need to determine in advance an initial number of clusters nor is there a need to reach a strictly well-formed (e.g. non-overlapping) categorisation. At each cycle of the process weights are calculated for each property according to how characteristic each property is for the emergent clusters. This algorithm is based on a working definition of similarity and category that inextricably binds the two together.

4.1 A Working Formal Definition of Similarity and Categorisation

Let *T* be a set of entities and *P* the union of all the sets of properties that are pertinent for the description of each entity. If $d(x,y)$ is the distance between two entities x and y, and *h* is a distance threshold we define similarity $s_h(x,y)$ as follows:

$$s_h(x,y) \begin{cases} 1 \text{ iff } d(x,y) \leq h \text{ (\textit{similar} objects)} \\ 0 \text{ iff } d(x,y) > h \text{ (\textit{dissimilar} objects)} \end{cases} \qquad \text{(I)}$$

In other words, two entities are *similar* if the distance between them is smaller than a given threshold and *dissimilar* if the distance is larger than this threshold.

The above definition of similarity is brought into a close relation with a notion of category. That is, within a given set of entities *T*, for a set of properties *P* and a distance threshold h, a category C_k is a maximal set:

$C_k=\{x_1,x_2,...x_n/x_i \in T\}$ with the property:

$$\forall i,j \in \{1,2,...n\},\ s_h(x_i,x_j)=1 \qquad \text{(II)}$$

In other words, *a category C_k consists of a maximal set of entities that are pairwise similar to each other for a given threshold h.*

A category, thus, is inextricably bound to the notion of similarity; all the members of a category are necessarily similar and a maximal set of similar entities defines a category.

As the similarity function s_h is not transitive, the resulting categories need not be disjoint (i.e. equivalence classes). In other words, overlap between categories is permitted.

The distance threshold may take values in the range of $0 \leq h \leq d_{max}$ where the distance d_{max} is defined as the maximum distance observed between all the pairs of entities in *T*. For h=0 every object in *T* is a monadic category; for $h=d_{max}$ all the objects in *T* define a single category.

4.2 The *Unscramble* algorithm

The above definitions of similarity and category can be restated in the terminology of graph theory as follows: objects are represented by vertices in an undirected graph, similarity between similar objects is represented by edges, and categories are defined as maximal cliques (a maximal clique is a maximal fully connected sub-graph). We will use this terminology below for the description of the *Unscramble* clustering algorithm.

It should also be noted that in the context of this paper 'properties' are taken to mean 'binary features' that correspond to a particular 'attribute-value' pair.

4.2.1 Algorithm input

The input to the *Unscramble* algorithm is a set of N objects each described by an m-dimensional property vector (e.g. object x: $[p_1,p_2,...p_m]$ and object y: $[q_1,q_2,...q_m]$). Each property has a corresponding initial weight $w_p=1$. The distance between two objects is given by the following function (based on the Hamming distance):

$$d(x,y)= \sum_{i=1}^{m} w_{p_i} \cdot w_{q_i} \cdot \delta(p_i,q_i) \qquad (III)$$

where: $\delta(p_i,q_i) = 0$ if $p_i=q_i$

$\delta(p_i,q_i) = 1$ if $p_i \neq q_i$

4.2.2 The algorithm

The algorithm proceeds in cycles; in each cycle, firstly, all the possible thresholds are calculated, then for each threshold an undirected graph is constructed (edges connect similar objects), then for each graph maximal cliques are enumerated, then for each clustering a 'goodness' value is computed and finally the clustering with the highest 'goodness' value is selected and new weights for all the properties are computed. A more detailed description is given below:

Step 1. All the possible threshold values h are calculated. The number of thresholds l is equal to the number of possible distances between the N objects of set T; $l_{max} = N \cdot (N-1)/2$ - it often is smaller as some entities are equidistant.
Step 2. For each of these thresholds, all the similar objects are computed according to definition (I) and (III) and an undirected graph for each threshold is created where edges connect similar objects.
Step 3. All the maximal cliques (II) are computed for each of these graphs, resulting in l different clusterings.
Step 4. For each of the l clusterings a 'goodness' value is calculated according to function (IVa,b).
Step 5. The clustering that rates highest according to the 'goodness' function is selected and new weights are calculated according to function (V).
Step 6. The algorithm is repeated from step 1 for the new weights.
Step 7. The algorithm terminates when the newly calculated 'goodness' value is less or equal to the value that resulted during the immediately preceding run.

4.2.3 Additional fundamentals

The following definitions are also necessary for the algorithm:

4.2.3.1 'Goodness' of clustering

As the *Unscramble* algorithm (section 4.2.1) generates a large number of clusterings (one for each possible similarity threshold) it is necessary to define some measure of 'goodness' for each clustering so as to select the best. Two such measures have been considered:

a. Overlap Function
One simple criterion for selecting preferred clusterings is a measure for the degree by which clusters overlap. The less overlapping between clusters the better. An overlap function OL could be defined as:

$$OL = 1 - N / \sum_{i=1}^{k} n_i \quad \text{where:} \qquad (IVa)$$

k = number of clusters
N = number of objects in T
n_i = number of objects in cluster C_i

The problem with such a measure is that in the extreme cases where each object is a cluster of itself and where all the objects form a single category overlapping is necessarily zero. It is thus necessary either to set *ad hoc* limits for minimum and maximum allowed number of clusters or to multiply the overlapping value by a function that has values close to 1 for a preferred range of number of clusters and values close to zero for the extremes of either too many or only one cluster. This *ad hoc* parametric bias is avoided in the next measure.

b. Category Utility
Category Utility (Gluck & Corter, 1985; Fisher, 1986) is a measure that rates the homogeneity of a clustering. Given a universe of entities T, a set of (binary) properties $P=\{p_1,...,p_m\}$ describing the entities, and a grouping of entities in T into k clusters $C=\{c_1,...,c_k\}$, category utility is defined as:

$$CU(\{c_1,\ldots,c_k\}) = \frac{\sum_{i=1}^{k} P(c_i) \cdot [\sum_{j=1}^{m} P(p_j/c_i)^2 - \sum_{j=1}^{m} P(p_j)^2]}{k} \qquad \text{(IVb)}$$

where:

$P(c_i)$ is the probability of an entity belonging to cluster c_i,
$P(p_j)$ is the probability that an entity has property p_j, and
$P(p_j/c_i)$ is the conditional probability of p_j, given cluster c_i.
Probabilities are estimated by relative frequencies.

CU favours categorisations with high uniformity (in terms of properties) within individual clusters ('intra-class similarity') and strong differences between clusters ('inter-class dissimilarity').

Another way of interpreting this is that category utility measures the *prediction potential* of a categorisation: it favours clusterings where it is easy to predict the properties of an entity, given that one knows which cluster it belongs to, and vice versa. The main advantages of this measure are its firm grounding in statistics, its intuitive semantics, and the fact that it does not depend on any parameters.

Both of these 'goodness' functions have been applied and compared in the example in section 5 (see also Appendices I and II).

4.2.3.2 Weighting function.

When a clustering is selected, then the initial weights of properties can be altered in relation to their 'diagnosticity', i.e. properties that are unique to members of one category are given higher weights whereas properties that are shared by members of one category and its complement are attenuated. A function that calculates the weight of a single property *p* could be:

$w = |m/n - m'/(N-n)|$ where: (V)

m = number of objects in category C_k that possess property p
m' = number of objects *not* in category C_k that possess property p (i.e. objects in T-C_k)
n = number of objects in C_k
N = number of objects in T

The weights of each property calculated for each category can then be averaged and normalised for a given clustering. The whole process may be repeated for the new set of weighted properties until the terminating conditions of *Unscramble* are met.

4.2.4. Complexity issues and merits of *Unscramble*

The enumeration of maximal cliques in an undirected graph is known to be an NP-complete problem; an extended overview of algorithms for maximum and maximal clique finding algorithms is presented in (Bomze et al, 1999). However, for small graphs this need not be a serious problem. Most musical categorisation tasks would involve tens or maybe a few hundreds of musical segments rather than thousands. It is also possible to consider using a semi-incremental version of the algorithm whereby objects are clustered in small chunks rather than all at once (this option is considered for further research).

An additional problem is that in each cycle of *Unscramble* all maximal cliques have to be computed for all the graphs that correspond to each threshold (maximum number of thresholds: $l_{max} = N \cdot (N-1)/2$ where N = number of objects) - i.e. the maximal clique enumeration algorithm has to be applied $\sim N^2$ times in the worst case! A solution to this problem has been given by E.Bomze and his colleagues (at the department of Statistics, University of Vienna) that is based on the observation that this problem is equivalent to finding all the maximal cliques in a single gradually evolving graph. According to this description of the problem edges are added one-by-one in an empty graph of N vertices until a fully connected graph is reached (or the reverse); each newly added edge is examined as to how it modifies the previously determined cliques. This evolving clique finding algorithm provides an efficient solution to the aforementioned problem.

The most useful characteristics of the *Unscramble* algorithm - depending on the task at hand - are the following:

- there is no need to define in advance a number of categories
- the prominence of properties is discovered by the algorithm
- categories may overlap.

Some examples of the usefulness of such clustering characteristics will be presented in the musical test case in the next section.

5. Motivic analysis of *Träumerei*

A detailed melodic and rhythmic analysis of R.Schumann's *Träumerei* is presented in (Repp, 1992). In this study we limit ourselves strictly to the soprano voice ignoring melodic gestures that appear in other voices. The soprano part of *Träumerei*, along with the segmentation provided by Repp, is depicted in Figure 4.

Table 1 presents the clustering of melodic segments that is used in Repp's study and Tables 2 and 3 present the clustering produced by the proposed computational system. The tables in this section are taken to be schematic representations of the score in Figure 4. In the bold outlined tables, rows correspond graphically to staffs in Figure 4; vertical lines correspond to segmentation points in staffs; table entries correspond to melodic segments; alphabet symbols correspond to names of melodic clusters. The first column, outside the bold part of the tables, indicates the structure at the level of melodic periods, i.e. 8-bar sections, as proposed by Repp; the second column indicates the phrase structure, i.e. 4-bar sections.

In this experiment, the *Unscramble* algorithm has been applied on the melodic segments in Figure 4; each segment is represented by attributes of equal initial weight at the surface level and reductions of it for various pitch and rhythmic parameters as described in section 3. The number of relevant clusters is computed by the algorithm – there's no need to predefine the number of clusters. Both of the 'goodness' functions described in section 4.2.3.1 have been used yielding in this case the same results (see appendices I and II).

When the attributes for each segment are calculated employing only *exact* matching then the results given in Table 2 are obtained. The main two clusters (*a* and *b*) emerge successfully along with clusters *e* and *f;* cluster *c* is selected because its members share an identical rhythmic pattern of 4 eighth-notes which is characteristic of the category. This clustering captures some of the main strong melodic categories that are given in Repp's analysis but over-emphasises the rhythmic aspects of cluster *c* and also totally misses some other important categories (see empty table entries).

A	A_1	a	b	c	d	e	
	B_1	a	b	f		g	
B	B_2	a	b	f		g	
	B_3	a	b	f		g	
A'	A_1	a	b	c	d	e	
	A_2	a	b	c	d	e	h

Table 1 Organisation of *Träumerei's* melodic segments according to Repp (1992).

A	A_1	a	b	c	c	e	
	B_1	a	b				
B	B_2	a	b	f			
	B_3	a	b	f			
A'	A_1	a	b	c	c	e	
	A_2	a	b		c	c	

Table 2 Machine organisation of *Träumerei's* melodic segments when *exact* matching is employed at the surface level and at reduced levels (empty entries signify monadic categories).

A	A_1	a	b	c	d	e	
	B_1	a	b				
B	B_2	a	b	f		g	
	B_3	a	b	f		g	
A'	A_1	a	b	c	d	e	
	A_2	a	b	c	d	d	d,e

Table 3 Machine organisation of *Träumerei's* melodic segments when *near-exact* matching is employed at the surface level and at reduced levels.

When *near-exact* matching is introduced then the results given in Table 3 are obtained. In this case, there are some additional clusters: cluster *c* is now the same as the one in Table 1; cluster *d* includes the last two segments of the melody and also overlaps on the last segment with cluster *e;* cluster *g* is also discovered; there are only two segments which remain monadic.

Some comments regarding the differences with Repp's melodic/rhythmic organisation of this melody will now be presented. Firstly, Repp chooses to group the second-to-last melodic segment with 'similar' segments in cluster *e* although this segment is *identical* – at least in regard to the melodic/rhythmic properties that are examined in this study - to segments in cluster d; we conjecture that this choice is taken by Repp because of top-down considerations – for instance

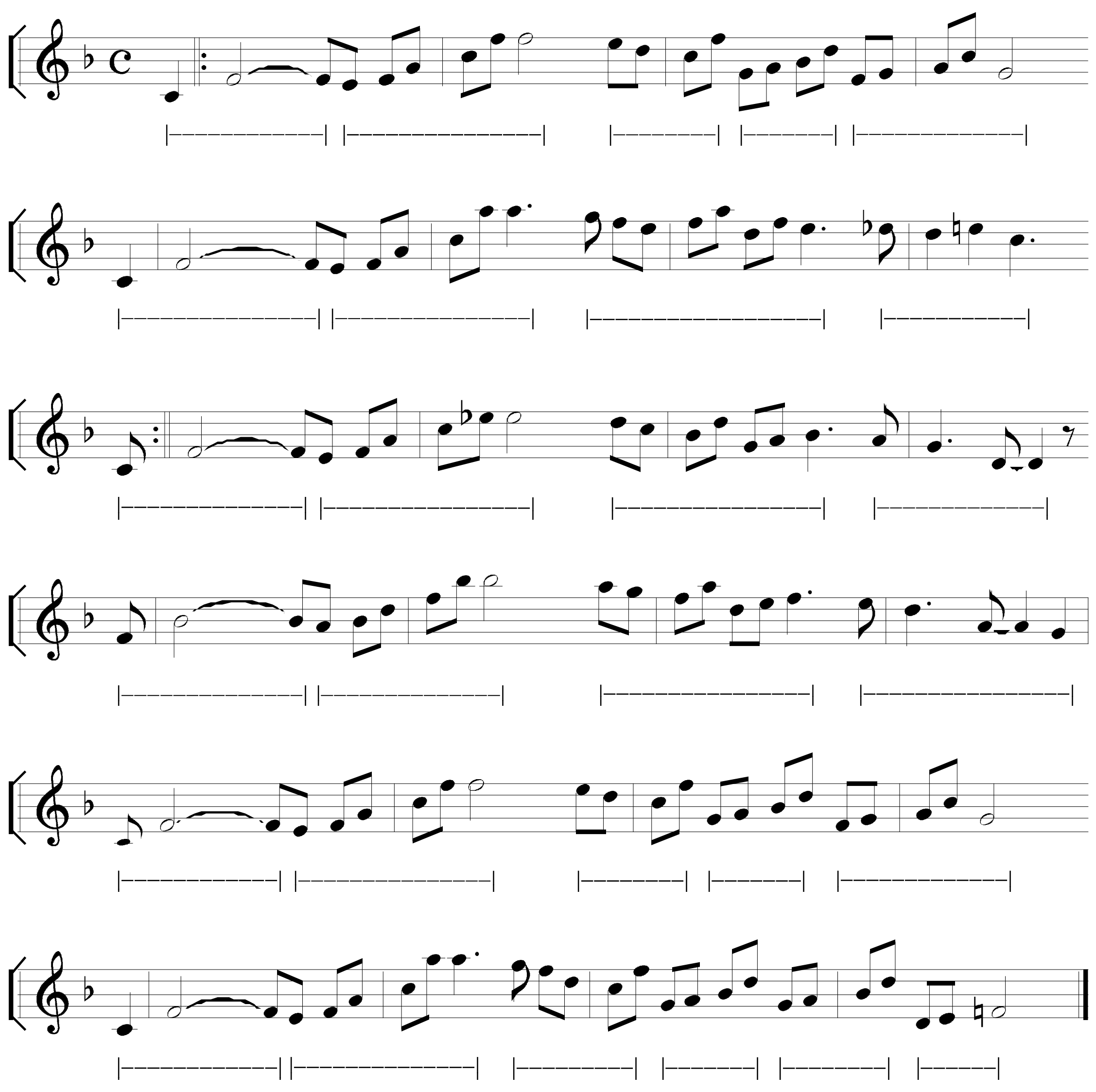


Figure 4 The soprano part of Schumann's *Träumerei,* along with segmentation provided by B. Repp (1992)

phrases 1, 5 and 6 may be seen a being overall 'parallel' (A_1, A_2, A_1) so their last parts are also considered as being similar. The *Unscramble* algorithm looks only at the selected internal attributes of segments so places this segment in cluster *d*. Secondly, in Repp's analysis the last segment is regarded as an independent monadic cluster *h* whereas *Unscramble* places it with clusters *d* and *e*. Thirdly, *Unscramble* fails to group the two 'left-over' segments (see empty entries in table 3) with clusters *f* and *g* respectively (a more sophisticated representation could capture the similarity for cluster *f* – however, it seems that for cluster *g* higher-level considerations of similarity at the phrase-level may be necessary). Overall the two analyses are quite similar; one may even claim that *Unscramble* yields some new 'intuitions' about the melodic/rhythmic organisation of this melodic part such as the overlap of the two clusters on the last segment of the piece or the grouping of the second-to-last segment with members of cluster *d.*

The *Unscramble* algorithm not only clusters melodic segments but also highlights their characteristic or defining properties. The weighting function described in section 4.2.3.2 reinforces 'diagnostic' properties for discovered clusters and attenuates properties that are less characteristic. For instance, members of cluster *b* share the same defining *step-leap* pitch attribute-value ('step-leap-leap-leap-equal') at the surface level (common to all members of this cluster and shared by no members of other clusters) and thus this attribute-value is a *defining* attribute (weight=1); this cluster is also characterised by the *doublestep-leap* attribute-value 'dstep-dstep-dstep-leap-equal' which is shared by all members except the third instance ('dstep-dstep-dstep-dstep-equal') and receives a strong weight w=0.83; a similar description for rhythm is given by *Unscramble*. As another example, *Unscramble* finds that members of cluster *c* share the same pattern attribute for the *doublestep-leap* representation (the first two instances are sub-patterns of the last); they also share the same rhythmic pattern but this receives a much lower weight as this pattern is also shared by members of cluster *d*.

Such a weighting mechanism can be used effectively for the description of clusters in terms of defining and characteristic attributes, which in turn can be used for further melodic pattern prediction tasks (not as yet fully investigated).

Conclusion

In this paper a formal model that organises melodic segments into 'significant' musical categories was presented. Given a segmentation of a melodic surface, the proposed model constructs an appropriate representation for each segment in terms of a number of properties (these reflect melodic and rhythmic aspects of the segment at the surface and at various abstract levels) and, then, the *Unscramble* algorithm organises these segments into 'meaningful' categories. The *Unscramble* clustering algorithm automatically determines an appropriate number of clusters, allowing some overlapping between clusters and also highlights the characteristic or defining attributes of each category.

As a test case, this computational model was used for obtaining a melodic and rhythmic analysis of the soprano-part of Schumann's *Träumerei*. This machine-generated analysis was compared with an analysis performed by a human music analyst; it was shown that the proposed model is capable of producing an 'acceptable' analysis that has a significant amount of resemblance to the human analysis

Acknowledgements

This research is part of the project Y99-INF, sponsored by the Austrian Federal Ministry of Education, Science, and Culture in the form of a START Research Prize. The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Education, Science, and Culture.

References

Anagnostopoulou, C., Hörnel, D. and Höthker, K. (1999) Investigating the Influence of Representations and Algorithms in Music Classification. In *Proceedings of the AISB'99 Convention (Artificial Intelligence and Simulation of Behaviour)*, Edinburgh, U.K.

Bent, I.D. (1980) Analysis. In *The New Grove Dictionary of Music and Musicians, Vol.1*. Macmillan, London.

Bomze, I.M., Budinich, M., Pardalos, P.M. and Pelillo, M. (1999) The Maximum Clique Problem. In *Handbook of Combinatorial Optimisation*, D.-Z. Du and P.M. Pardalos (Eds.), Kluwer Academic Publishers, Dordrecht, The Netherlands.

Cambouropoulos, E. (1998) Musical Parallelism and Melodic Segmentation. In *Proceedings of the XII Colloquium of Musical Informatics*, Gorizia, Italy.

Cambouropoulos, E., Crawford, T. and Iliopoulos, C.S. (1999) Pattern Processing in Melodic Sequences: Challenges, Caveats and Prospects. In *Proceedings of the AISB'99 Convention (Artificial Intelligence and Simulation of Behaviour)*, Edinburgh, U.K.

Cambouropoulos, E. and Smaill, A. (1997) Similarity and Categorisation Inextricably Bound Together: The *Unscramble* Machine Learning Algorithm. In *Proceedings of the Interdisciplinary Workshop on Similarity and Categorisation*, University of Edinburgh, Edinburgh.

Fisher, D.H. (1986) Knowledge Acquisition via Incremental Conceptual Clustering. *Machine Learning* 2(2):139-172.

Gluck, M.A., and Corter, J.E. (1985) Information, Uncertainty, and the Utility of Categories. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates, Irvine (Ca).

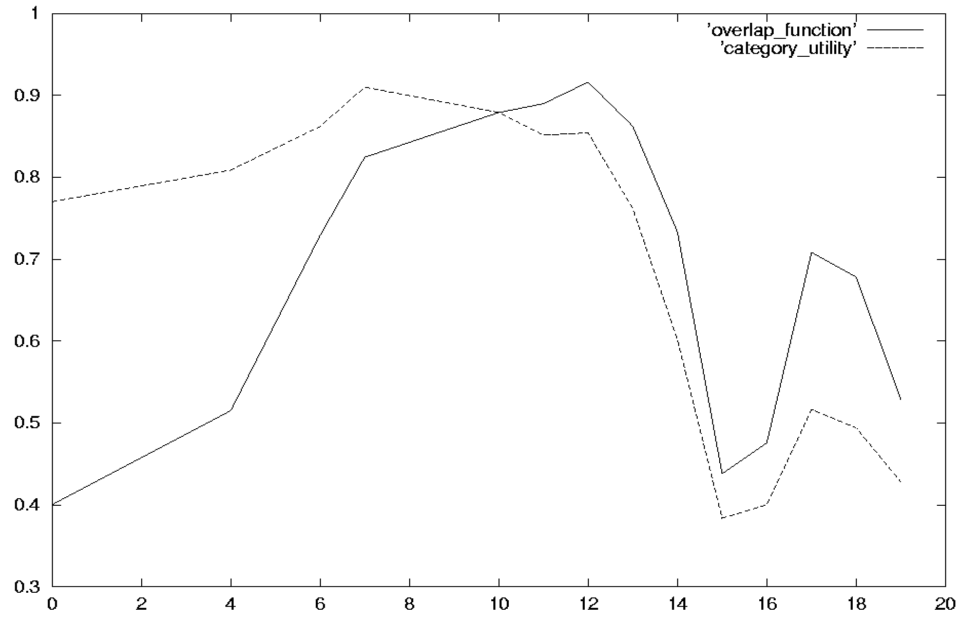
Nattiez, J-J. (1990) Music and Discourse: Towards a Semiology of Music. Princeton University Press, Princeton.

Nattiez, J-J. (1975) *Fondements d'une Sémiologie de la Musique*. Union Générale d'Editions, Paris.

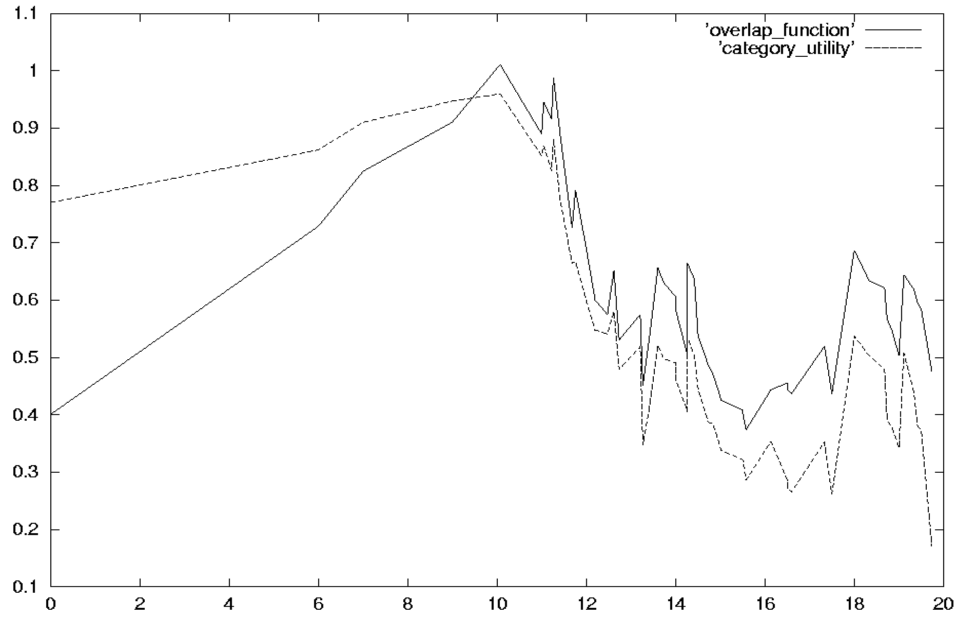
Repp, B. (1992) Diversity and commonality in music performance: An analysis of timing microstructure in Schumann's *Träumerei. Journal of the Acoustical Society of America*, 92(5): 2546-2568.

Rolland, P.Y., and Ganascia, J.G. (2000) Musical Pattern Extraction and Similarity Assessment. In *Readings in Music and Artificial Intelligence*. E. Miranda. (ed.). Harwood Academic Publishers (in press).

Widmer, G. (1996) Learning Expressive Performance: The Structure-Level Approach. *Journal of New Music Research* 25(2): 179-205.



Appendix I Graphs of the clustering 'goodness' values that occur during the *first* cycle of the *Unscramble* algorithm when applied to the melodic segments of *Träumerei*. The x-axis indicates threshold values (that correspond to individual clusterings); the y-axis indicates 'overlap' and 'category utility' values. During the first cycle, the two 'goodness' value functions select different clusterings (different peaks of the graphs).



Appendix II Graphs of the clustering 'goodness' values that occur during the *last* cycle of the *Unscramble* algorithm when applied to the melodic segments of *Träumerei*. The x-axis indicates threshold values; the y-axis indicates 'overlap' and 'category utility' values. During the last cycle, the two 'goodness' value functions select the same final clustering and have a very similar overall outlook (peaks at the same positions).