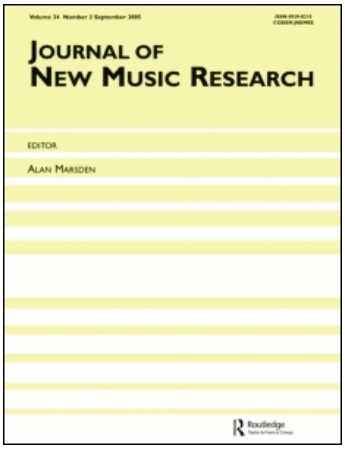


Journal of New Music Research
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713817838

Automated Motivic Analysis via Melodic Clustering

Emilios Cambouropoulos[a]; Gerhard Widmer[a]
[a] Austrian Research Institute for Artificial Intelligence, Vienna, Austria

To cite this Article Cambouropoulos, Emilios and Widmer, Gerhard(2000) 'Automated Motivic Analysis via Melodic Clustering', Journal of New Music Research, 29: 4, 303 — 317
To link to this Article: DOI: 10.1080/09298210008565464
URL: http://dx.doi.org/10.1080/09298210008565464

Journal of New Music Research, 29 (2000), No. 4, pp. 303–317

0929-8215/00/2904-303$15.00

AUTOMATED MOTIVIC ANALYSIS VIA MELODIC CLUSTERING

Emilios Cambouropoulos and Gerhard Widmer

Austrian Research Institute for Artificial Intelligence, Vienna, Austria

ABSTRACT

In this paper a computational 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 algorithm 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 attributes of each category. As a test case this computational model has been used for obtaining a motivic analysis of Schumann's *Träumerei* and Debussy's *Syrinx*.

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' categorisation 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 not only with local discontinuities but also with musical 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 part of ongoing research.

Correspondence: Emilios Cambouropoulos, Austrian Research Institute for Artificial Intelligence, Schottengasse 3, A-1010, Vienna, Austria. E-mail: Emilios@ai.univie.ac.at

There has been a limited number of attempts to use clustering techniques for organising melodic segments into paradigms – a study and comparison of some clustering algorithms for melodic analysis is presented in (Höthker et al., 2001). 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 methods and clustering techniques. As a test case the proposed computational model will be used for obtaining motivic analyses of the soprano part of R. Schumann's *Träumerei* (this is the 7th piece from *Kinderszenen, op. 15*) and of Debussy's *Syrinx* for solo flute.

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' (see Widmer, 2000) 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 particular, information on the motivic structure of a piece is essential in determining the types of expressive patterns that are applied by a performer on different instances of the same melodic pattern. For example, if the same expressive gesture/mood is intended for the different variations of a motive (e.g., a leitmotif) then knowing motivic classes is necessary both for learning expressive patterns from a performance and for predicting expressive patterns of motives in new pieces.

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.

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., 2001). 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).

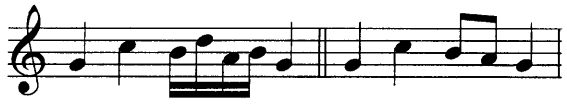


Fig. 1. These two melodic segments may be construed as similar either because five of their pitches and onsets match at the surface level (approximate matching) or because they match exactly at the reduced eighth-note metric level.

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.

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

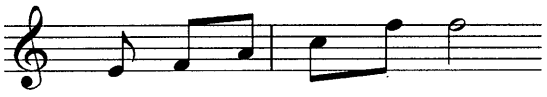
a. Surface level
   - Pitch-intervals: exact (p_ex), scale-steps (p_ss), step-leap (p_sl), step-doublestep-leap (p_sdsl), doubleORsinglestep-leap (p_dssl), and contour (p_contour).
   - Rhythm: exact durations (r_dur), inter-onset interval (r_ioi), short-long (r_sl), inter-onset interval ratios (r_ratio), 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), doubleORsinglestep-leap (p_dssl).
   - 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). NB: in r_sl, durations longer than a certain value (in this instance a quarter) are represented merely by 'long'.

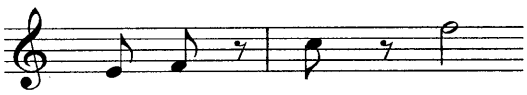
Further research is necessary to establish a plausible and quite general set of levels of abstraction and

Melodic surface



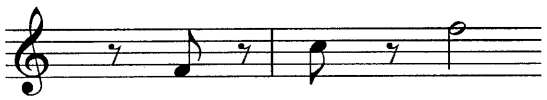
p_ex:	1	-	4	-	3	-	5	-	0
p_ss:	1	-	2	-	2	-	3	-	0
p_sl:	step	-	leap	-	leap	-	leap	-	0
p_sdsl:	step	-	dstep	-	dstep	-	leap	-	0
p_dssl:	dss	-	dss	-	dss	-	leap	-	0
p_contour:	U	-	U	-	U	-	U	-	0
r_dur:	1/8	-	1/8	-	1/8	-	1/8	-	1/8 - 1/2
r_ioi:	1/8	-	1/8	-	1/8	-	1/8	-	1/8 - 1/2
r_sl:	1/8	-	1/8	-	1/8	-	1/8	-	long
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_dssl:	dss	-	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

Fig. 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.

reduction. The above set gives gradually less prominence to the more abstract reduction levels and gives a 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 study. 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.*

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 Fig. 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.

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

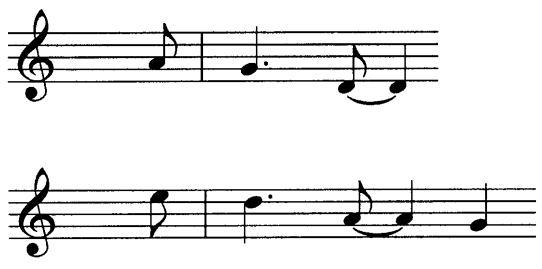


Fig. 3. These two melodic segments (from *Träumerei*) share common sub-patterns for both the pitch interval and rhythm profiles (actually the first is a sub-pattern of the second) – that is, their attribute-values for both pitch and rhythm are *near-exact*.

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 & 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.

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:

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

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:

$$C_k = \{x_1, x_2, \ldots, x_n / x_i \in T\} \quad \text{with the property:}$$
$$\forall i,j \in \{1,2,\ldots,n\},\ s_h(x_i, x_j) = 1. \qquad \text{(II)}$$

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 relation 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 over 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.

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.

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: $[p_1, p_2, \ldots, p_m]$ and object y: $[q_1, q_2, \ldots, q_m]$). 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):

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

where: $\delta(p_i, q_i) = 0$ if $p_i = q_i$ and $\delta(p_i, q_i) = 1$ if $p_i \neq q_i$.

The algorithm

The algorithm proceeds in cycles; in each cycle, firstly, all the possible distance 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 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) – see below.

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 selected 'goodness' value is less or equal to the value that resulted during the immediately preceding run.

Additional fundamentals

The following definitions are also necessary for the algorithm:

'Goodness' of clustering

As the *Unscramble* algorithm 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:

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

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 or 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 0 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 $P = \{p_1, \ldots, p_m\}$ describing the entities, and a grouping of entities in T into k clusters $C = \{c_1, \ldots, c_k\}$, category utility is defined as:

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

where:

$P(c_i)$ is the probability of cluster c_i
$P(p_j)$ is the probability that an entity has property p_j
$P(p_j|c_i)$ 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 application of the algorithm on the two musical test cases (see last section and also Appendices I and II).

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:

$$w = |m/n - m'/(N - n)|, \qquad \text{(V)}$$

where:

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 maximum weights of each property calculated for each category are then selected 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.

Having a weighted set of properties at the end of the process is very useful in that each cluster can be

explicitly described by the set of its defining or most characteristic attributes; such properties can be used for membership prediction tasks. The term *defining properties* refers to properties that have a weight equal to 1; such properties are singly necessary and jointly sufficient in defining a category (classical description of a category). *Characteristic properties* are properties that receive a weight value close to 1 and are typical of a category, but are not necessarily shared by all the members of the category (prototypical description of a category). Examples of such properties are given in the last section.

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 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 (Stix, 2000) 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 (which is particularly useful for musical applications).

Some of these characteristics are accommodated in various clustering algorithms. For instance, *Cluster/2* (Michalski, 1983) is an unsupervised learning algorithm that produces explicit descriptions of emerging clusters determining the conditions/criteria for grouping together members of each cluster (conceptual clustering). *Cobweb* (Fisher, 1987) encompasses all the above characteristics except overlapping; it creates hierarchical categorisations which is something *Unscramble* can do as well (though that is not the topic of the present paper); the number of categories, however, has to be defined in advance and overlapping is also not allowed. *Cobweb* differs from Unscramble in that it is based on a probabilistic approach and also performs categorisation in an incremental manner. *Adclus* (Arabie, 1977) is an indirect clustering model and its main common characteristic with *Unscramble* is that it allows overlapping of categories – this characteristic is rarely accommodated in clustering algorithms – see (Arabie et al., 1981) for potential utility of overlapping approaches to categorisation.

A wider comparison with these and other relevant clustering algorithms will be necessary for establishing and assessing the relative usefulness of *Unscramble*. The proposed algorithm itself can benefit from other approaches, such as, for instance, using Cobweb's category utility criterion for evaluating the quality of categorisation descriptions (see previous section on 'goodness' of clustering).

Some examples of the usefulness of the special characteristics of *Unscramble* will be presented in the musical test cases in the next section.

TWO ANALYSES OBTAINED BY *UNSCRAMBLE*

In this section the *Unscramble* algorithm will be used to obtain motivic analyses of Schumann's *Träumerei* and Debussy's *Syrinx* (for application of the algorithm on other melodies see also Cambouropoulos

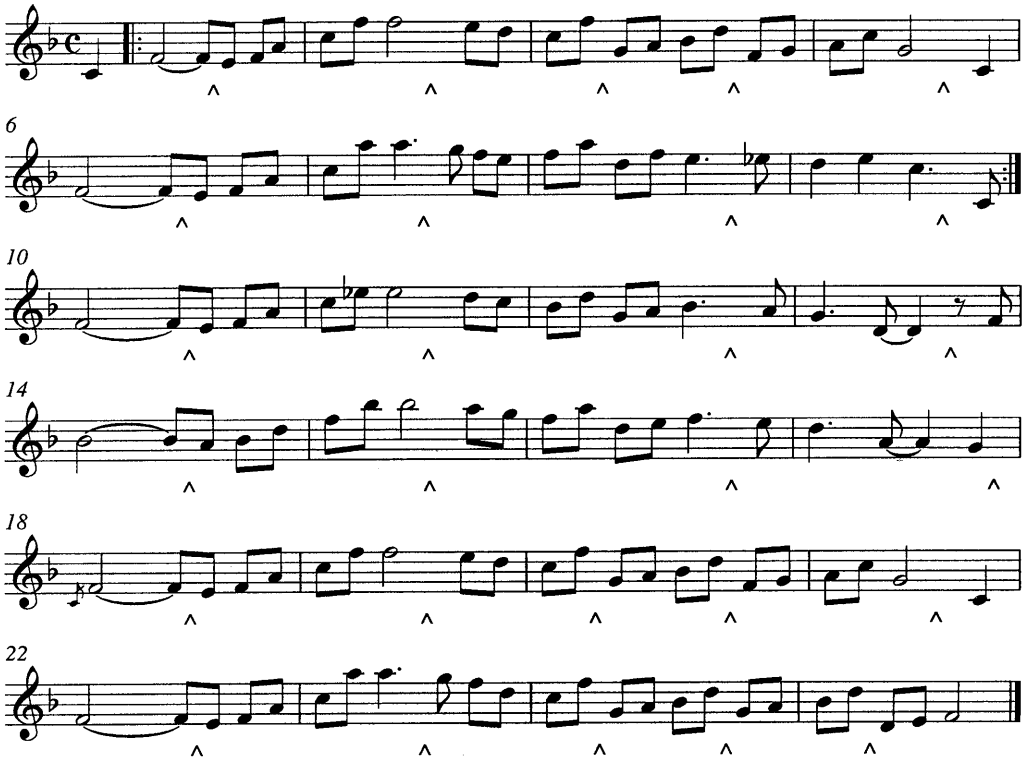


Fig. 4. The soprano part of Schumann's *Träumerei*, along with segmentation provided by B. Repp (1992) – points of segmentation indicated by upward arrows.

et al., 1999; Cambouropoulos, 2001). The aim of applying the algorithm on each of these pieces is merely to show that, given a certain 'reasonable' segmentation, the algorithm can organise the emerging melodic segments into categories that are comparable to the motivic categories given by a human. No musicological judgement is intended neither for the initial segmentations, nor for the motivic classes given in the human analyses.

Motivic analysis of *Träumerei*

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. 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*. The machine-generated results obtained herein will be compared to and tested against the human analysis provided in Repp's study.

In this paper 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 categorisation of melodic segments that is given in Repp's study and Tables 2 and 3 present the clusterings produced by the *Unscramble* algorithm. 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

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

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

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*	*e*	*d*, *e*

initial weight at the surface level and reductions of it for various pitch and rhythmic parameters as described in the section on melodic representation. 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 previously 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).

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, phrases 1, 5 and 6 may be seen a being overall 'parallel' (A_1, A_1, A_2) 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 'hints' 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 previously 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_up - leap_up - leap_up - leap_up - 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 ascending *doubleORsinglestep-leap* attribute-value 'dss-dss-dss-leap-equal' which is shared by all members except the third instance ('dss-dss-dss-dss-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 *doubleORsinglestep-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).

Motivic Analysis of Debussy's *Syrinx*

J.-J. Nattiez provides two paradigmatic analyses of Debussy's *Syrinx* for solo flute – the first consists of segments roughly at the bar level and the second at the beat level. (Nattiez, 1975; see brief intoduction in english and presentation of this analysis in Cook, 1990). Below we will use *Unscramble* in an attempt to reconstruct Nattiez's second analysis.

Paradigmatic analysis relies to a large extent on the segmentation of a musical work. However, segmentation is not merely a prerequisite to paradigmatic analysis but is implicitly affected by the taxomonic process itself. The exact manner in which categorisation processes affect segmentation has not been described by Nattiez. For practical reasons, we will take the segmentation provided in Nattiez's second analysis as a given in this study (see Fig. 5).

The set of attributes described in the section on melodic representation is used in the current experiment with the following modifications (grace notes have been omitted in the description of segments): (a) attributes for the reduced quarter-note (beat) levels are omitted as the current segmentation is roughly at the beat level and reduction would produce mostly trivial single-note segments, and (b) two of the rhythmic attributes are omitted so as to maintain the initial overall balance that gives preference to pitch parameters (NB: the attributes for quarter-note reductions were strongly biased towards pitch). So, the following set of attributes is used for the surface level:

- Pitch-intervals: exact (p_ex), scale-steps (p_ss), step-leap (p_sl), step-doublestep-leap (p_sdsl), doubleORsinglestep-leap (p_dssl), and contour (p_contour).
- Rhythm: inter-onset interval (r_ioi), short-long (r_sl), inter-onset interval ratios (r_ratio).

As this piece is longer and consists of many segments (total number: 76) that cannot be aligned neatly as in the *Träumerei* example, the above table representation for clustering will not be used. Instead, melodic categories will be presented as columns of actual melodic segments (exact repetitions of segments have been omitted). For convenience, only the first four main clusters, which account for approximately two-thirds of the segments, will be illustrated and discussed. In Figure 6 the four clusters (named A, B, C and D) given by Nattiez are presented (plus cluster E also identified by Nattiez).

The overall clustering produced by *Unscramble* is very similar to the human paradigmatic analysis. More specifically, the main four clusters produced by *Unscramble* are indeed very close to those of Nattiez's analysis (represented by the four full columns of Fig. 6 – see caption for details). The algorithm not only discovers these clusters but also describes them in terms of defining and characteristic attributes.

The first cluster (17 segments including exact repetitions) is identical to Nattiez's class A. This is the most prominent motivic class for this piece. The algorithm discovers that a defining attribute for this cluster is the *scale-step* attribute: 'step_down, step_up'. The most characteristic attribute is the exact pitch interval attribute: 'minor_2nd_down, major_2nd_up'. In terms of rhythm, the most characteristic attribute (not very strong) is for *exact inter-onset intervals*: 'dotted_

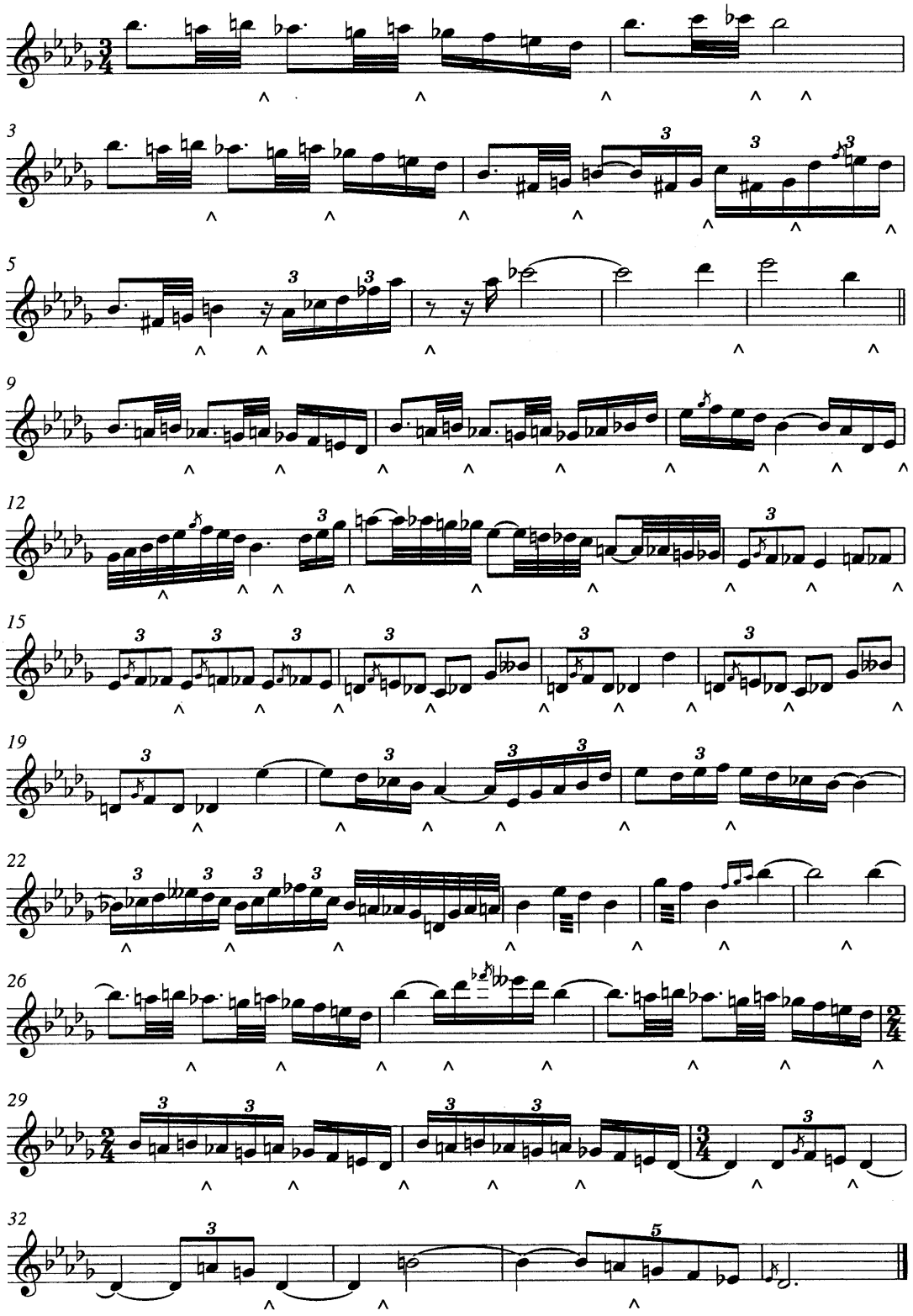


Fig. 5. Debussy's *Syrinx* for solo flute. Points of segmentation, according to Nattiez's analysis, are indicated by upward arrows.

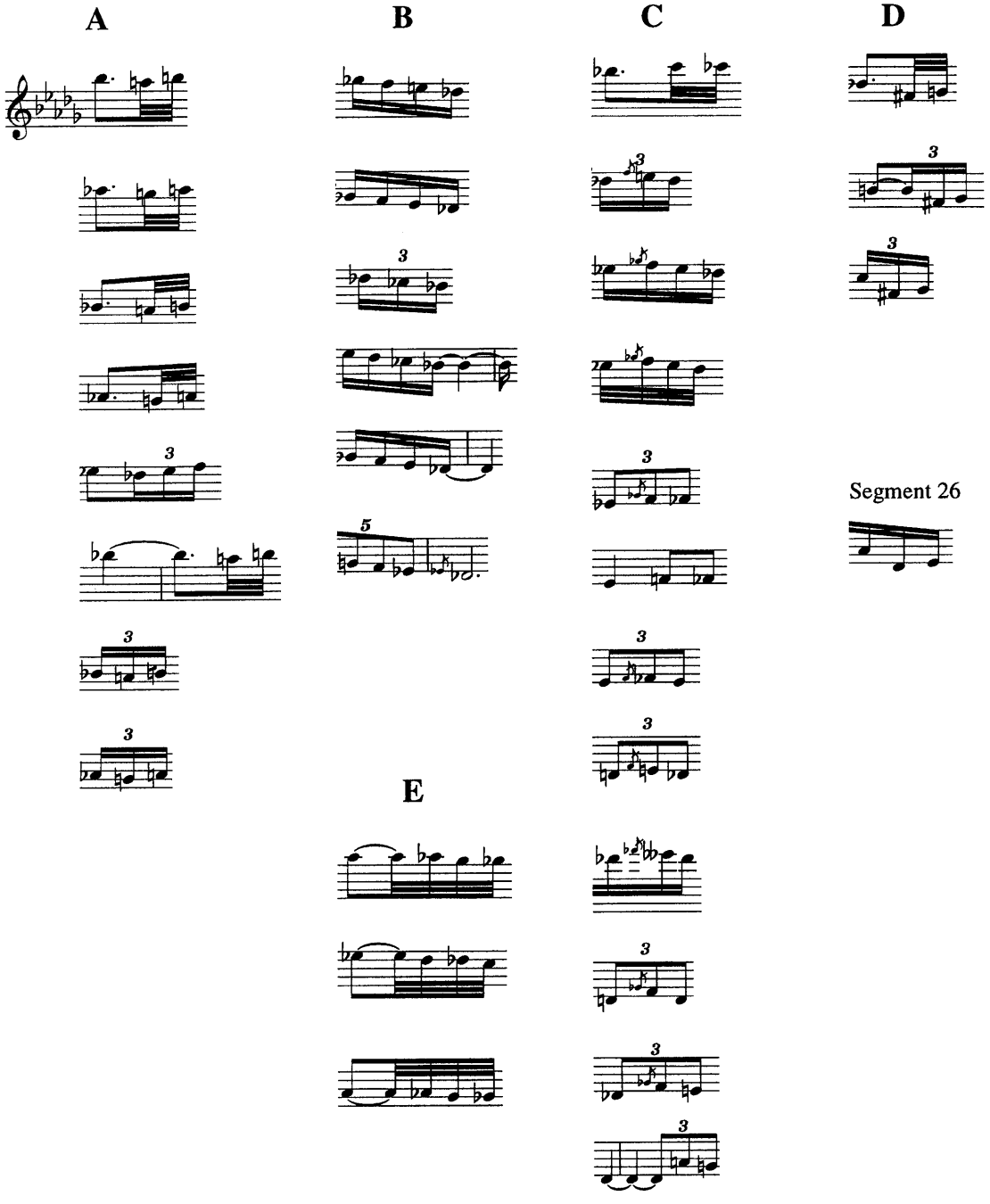


Fig. 6. First four main melodic classes (A, B, C and D) given in Nattiez's paradigmatic analysis of Debussy's *Syrinx* (plus class E also given in the same analysis). The *Unscramble* algorithm gives exactly the same results with the following exceptions: class E is merged with class B (defining attribute: descending melodic steps), the last three members of class C are not placed in this category (see discussion in text), and class D contains an extra segment (segment 26).

eighth, thirty-second, thirty-second' (it appears 10 times in members of this category and three times in members of other categories).

The second cluster discovered by the algorithm (13 segments) contains all the members of both Nattiez's classes B and E. Defining attribute value for this clus-

ter is: descending steps; in the current implementation there is no distinction between chromatic and diatonic steps at the scale-step abstraction level. The representation should be refined further to account for chromatic passages (a chromatic semitone is different from a minor 2nd – however, in the case of ascending or descending chromatic passages this distinction is not important).

The third cluster (14 segments) is the same as with Nattiez's class C with the exception of the last three members which are placed in other categories. The reason is that the last three members contain 'leaps' and that the threshold that was selected by the algorithm as producing the optimal 'goodness' value was somewhat too narrow for them to be included in this category. A richer representation in terms of a broader range of attributes may provide a more relaxed threshold that would allow these segments to be included as well. Defining attribute for this cluster is: 'step_up, step_down'. The most characteristic rhythmic attribute is for the inter-onset interval ratio attribute (not very strong). Inclusion of the grace notes should strengthen this category.

The fourth cluster (5 segments) includes additionally segment 26. It is not clear why Nattiez did not included this segment in this paradigmatic class – he has actually stretched this segment placing it in between two clusters (the first two notes in one class and the third note in another class!). The defining attribute for this cluster is: 'leap_down, step_up'.

Quite different results arise if more initial prominence is given to the rhythmic attributes. For instance, the first segment of cluster C may be placed in cluster A or the two clusters may overlap over this segment. Determining an adequate initial set of attributes requires further research.

In this case as well, *Unscramble* gives exactly the same results for both the Overlap Function and for Category Utility. The algorithm produces the final result after 4 cycles.

The *Unscramble* algorithm provides a very useful computational method for studying the task of generating a 'meaningful' paradigmatic analysis of a musical work as the whole analytic process has to be explicit and consistent. The commonalities and differences between the human and machine analyses may highlight possible additional intuitive musical knowledge that the analyst has used during the analytic process, or, of course, shortcomings of the computational method.

A computational analysis of this piece using an unsupervised neural network is mentioned in (Anagnostopoulou et al., 1997). However, no extended results are presented in this paper so a comparison with the results of the current experiment was not possible. One advantage of the current algorithm is that an explicit description of the emerging clusters is given in terms of defining and characteristic attributes (this is very difficult to achieve using a neural net).

CONCLUSION

In this paper a computational 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 melodic and rhythmic analyses of the soprano-part of Schumann's *Träumerei* and of Debussy's *Syrinx* for solo flute. This machine-generated analysis was compared with the analysis performed by human music analysts; it was shown that the proposed model is capable of producing an 'acceptable' analysis that has a significant amount of resemblance to the human analyses.

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. & Westermann, G. (1997). Classification in music: A computational model for paradigmatic analysis. In *Proceedings of the International Computer Music Conference 1997*. San Francisco: International Computer Music Association.

Arabie, P. et al. (1981). Overlapping clustering: A new method for product positioning. *Journal of Marketing Research, 18*, 310–317.

Arabie, P. (1977). Clustering representations of group overlap. *Journal of Mathematical Sociology, 5*, 113–128.

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

Bomze, I. M., Budinich, M., Pardalos, P. M. & Pelillo, M. (1999). The maximum clique problem. In D.-Z. Du & P. M. Pardalos (Eds.), *Handbook of Combinatorial Optimisation*. The Netherlands, Dordrecht: Kluwer Academic Publishers.

Cambouropoulos, E. (2001). Melodic cue abstraction, similarity and category formation: A formal model. *Music Perception, 18*(3). (forthcoming).

Cambouropoulos, E. (1998). Musical parallelism and melodic segmentation. In *Proceedings of the XII Colloquium of Musical Informatics*, Gorizia, Italy.

Cambouropoulos, E., Crawford, T. & Iliopoulos, C. S. (2001). Pattern processing in melodic sequences: Challenges, caveats and prospects. *Computing and the Humanities, 35*(1), 9–21 (forthcoming).

Cambouropoulos, E., Smaill, A. & Widmer, G. (1999). A clustering algorithm for melodic analysis. In *Proceedings of the Diderot Forum on Mathematics and Music*. Vienna, Austria.

Cambouropoulos, E. & 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.

Cook, N. (1987). *A Guide to Musical Analysis*. London: J. M. Dent and Sons Ltd.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning, 2*(2), 139–172.

Gluck, M. A. & Corter, J. E. (1985). Information, uncertainty, and the utility of categories. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*. Irvine (Ca): Lawrence Erlbaum Associates.

Höthker, K., Hörnel, D. & Anagnostopoulou C. (2001). Investigating the influence of representations and algorithms in music classification. *Computing and the Humanities, 35*(1), 65–79.

Michalski, R. S. (1983). Learning from observation: Conceptual clustering. In R. S. Michalski et al. (Eds), *Machine Learning: An Artificial Intelligence Approach*. Berlin: Springer-Verlag.

Nattiez, J.-J. (1990). Music and discourse: Towards a semiology of music. Princeton: Princeton University Press.

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

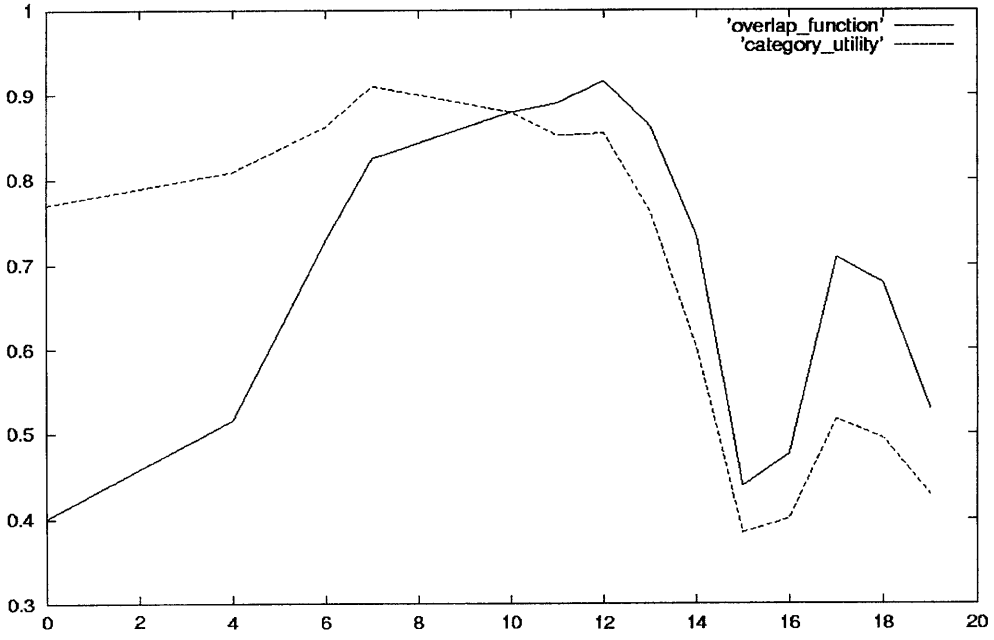
Repp, B. (1992). Diversity and commonality in music performance: An analysis of timing microstructure in Schumann's *Träumerei*. *Journal of the Acoustical Society of America, 92*(5), 2546–2568.

Rolland, P. Y. & Ganascia, J. G. (2000). Musical pattern extraction and similarity assessment. In E. Miranda (Ed.). *Readings in Music and Artificial Intelligence*. London: Harwood Academic Publishers (in press).

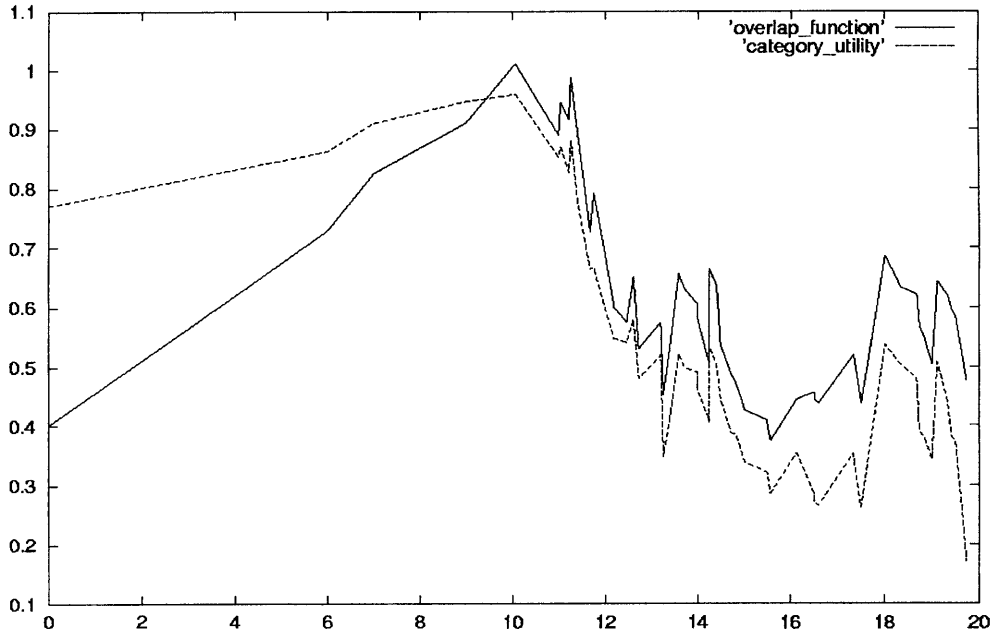
Stix, V. (2000). Finding all maximal cliques in evolving graphs. Research Report, Department of Statistics, University of Vienna, Austria.

Widmer, G. (2000). Large-scale induction of expressive performance rules: First qualitative results. In *Proceedings of the International Computer Music Conference 2000*. San Francisco: International Computer Music Association.

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).