A Clustering Algorithm for Melodic Analysis

Emilios Cambouropoulos,[*] Alan Smaill[†] and Gerhard Widmer[#]

[*,#]Austrian Research Institute for Artificial Intelligence, Vienna, Austria
[#]Department of Medical Cybernetics and Artificial Intelligence, University of Vienna
[†]Division of Informatics, University of Edinburgh, Scotland
{emilios, gerhard}@ai.univie.ac.at, smaill@dai.ed.ac.uk

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 three melodies from diverse musical styles.

1. Introduction

Paradigmatic analysis (Nattiez 1975, 1990) is an analytic methodology that aims at organising musical entities (e.g. melodic 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 the current paper a computational system is described that can arrive at a 'plausible' categorisation of a given set of melodic segments (segmentation is taken to be a pre-requisite) and at the same time provide an explicit description of each category.

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).

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.

The main topic that will be discussed in this paper is clustering techniques (mainly the *Unscramble* algorithm) for musical applications. As a test case the proposed computational model will used for obtaining a motivic analysis of three melodies from diverse musical styles

2. 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) categori-sation. 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.

2.1 A Working Formal Definition of Similarity and Categorisation

Let T be a finite 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:

$$s_h(x,y) \begin{cases} 1 \text{ iff } d(x,y) \leq h \text{ (similarity)} \\ 0 \text{ iff } d(x,y) > h \text{ (dissimilarity)} \end{cases} \quad \text{(I)}$$

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:

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

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.

2.2 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.

2.2.1 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):

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

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

2.2.2 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.

2.2.3 Additional fundamentals

The following definitions are also necessary for the algorithm:

2.2.3.1 '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:

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

where: 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:

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

$P(c_i)$ 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.

2.2.3.2 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.

2.2.4. 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). In our experiments so far, NP-completeness has not turned out to be a practical problem, because our graphs happen to be not so dense in maximal cliques (and just to safeguard against that, the algorithm abandons a graph when the 'overlapping' criterion reaches a certain threshold); the graph sizes in our experiments have not exceeded 50 vertices. One possible way to deal with computational complexity, is to consider using a semi-incremental version of the algorithm whereby objects are clustered gradually 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 analyses given in the next section.

3. Three melodic analyses

In this section the *Unscramble* algorithm is applied on three melodies selected from diverse musical idioms. The segmentation of each melody is taken as a given - segmentation techiques are discussed extensivelly in (Cambouropoulos, 1998).

Each melodic segment is represented as a vector of attribute-values. The following attributes are computed for each segment: exact pitch interval pattern (semitones), scale-step intervals, pitch contour, exact duration pattern, relative duration pattern (i.e. sequence of shorter, longer, equal onset intervals), all the previous attributes for a reduced version of the surface that consists of metrically stronger notes (repeated notes are merged when no strong boundary appears between them), and, finally, exact pitch interval between first and last note of each segment, and register of each segment (high or low). Obviously there are a large number of other attributes that may be considered important but these should suffice for the purposes of this exercise (more extended discussion on adequate representations of melodic segments can be found in Cambouropoulos et al, 1999). As a first rough approximation rhythmic attributes have been given half the weight of the attributes relating to pitch.

Figures 1, 2 and 3 illustrate the segmentations and the resulting clusterings for the following three melodies: a) the finale theme of Beethoven's *9th Symphony*, b) *L'homme armé* (a 15th century melody) and c) the a melody of the first song from Webern's *Fünf Lieder Op.3.*

a)

b)

Figure 1 a) Segmentation of the finale theme of Beethoven's *9th Symphony*,
b) The 'best' clustering revealed by the *Unscramble* algorithm
(identical segments are represented by the same symbol).

a)

b)

Figure 2 a) Segmentation of the 15th century melody *L'homme armé*,
b) The 'best' clustering revealed by the *Unscramble* algorithm.

a)

b)

Figure 3 a) Segmentation of the melody of the first song from Webern's *Fünf Lieder Op.3,* b) The 'best' clustering revealed by the *Unscramble* algorithm.

The number of clusters in each example is automatically determined by *Unscramble.* As can be seen (e.g. Figures 1 and 2) the algorithm allows a limited amount of overlapping which can be useful for representing a notion of musical ambiguity. Finally, the *Unscramble* algorithm highlights the characteristic or defining properties of each segment that may be used for further melodic pattern prediction tasks (for instance, the most characteristic - actually defining - attributes of category {a,c,j} in Figure 1 are: the scale-step and contour pitch interval pattern for the reduced surface and the first-last note interval; for category {a,h} of Figure 2 the most characteristic attributes are the shared rhythmic pattern, the pitch contour of the reduced surface and the exact pitch interval between the first and last note of the segments).

These three computer-generated analyses have been examined by a number of musical experts and have been judged to be at least 'acceptable', if not, in some instances, 'interesting'.

We are also currently applying our algorithm to more extended pieces of music. Preliminary results appear to be highly interesting (Cambouropoulos and Widmer, forthcoming)

Conclusion

In this paper a formal model that organises melodic segments into 'significant' musical categories was presented. Given a segmentation of a melodic surface and 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) 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 analyses of three melodies selected from diverse musical idioms. It is suggested that the proposed model is capable of producing analyses that are 'acceptable' or 'plausible' when judged by a human analyst.

Acknowledgements

This research is part of the project Y99-INF, sponsored by the Austrian Federal Ministry of Science and Transport in the form of a START Research Prize.

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.

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. and Widmer, G. Melodic Clustering: Motivic Analysis of Schumann's *Träumerei* (forthcoming).

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.