Pattern Processing in Melodic Sequences: Challenges, Caveats & Prospects

Emilios Cambouropoulos,* Tim Crawford,[#] Costas S. Iliopoulos[⊗]

*Austrian Research Institute for Artificial Intelligence, Vienna.

[#]Department of Music, King's College London.

*,[⊗]Department of Computer Science, King's College London.

emilios@ai.univie.ac.at, tim.crawford@kcl.ac.uk, csi@dcs.kcl.ac.uk

Abstract

In this paper a number of issues relating to the application of string processing techniques on musical sequences are discussed. A brief survey of some musical string processing algorithms is given and some issues of melodic representation, abstraction, segmentation and categorisation are presented. This paper is not intended towards providing solutions to string processing problems but rather towards highlighting possible stumbling-block areas and raising awareness of primarily music-related particularities that can cause problems in matching applications.

1. Introduction

There exists a large number of string matching algorithms which are usually applied on text strings or biological strings (e.g. DNA or protein strings) - a plethora of string algorithms is surveyed in (Apostolico and Galil, 1985) and (Crochemore and Rytter, 1994).

It is often hypothesised that a musical surface may be seen as a string of musical entities such as notes, chords etc. on which pattern recognition or induction techniques can be applied. In this text, the term *pattern induction* refers to techniques that enable the extraction of useful patterns from a string whereas *pattern recognition* refers to techniques that find all the instances of a predefined pattern in a given string.

Overviews of the application of pattern processing algorithms on musical strings can be found in (McGettrick, 1997; Crawford et al, 1998; Rolland et al, 1999); a very brief overview of a number of such music pattern processing methods is presented in this paper in Table 1 - see Appendix.

When attempts are made to apply string matching algorithms to musical strings various questions arise that have to do with the particular nature of musical elements. For instance, should a melody be represented at the lowest level as a single string of note tuples (pitch and duration) or should the different parameters be treated as separate strings? Should the melodic surface be considered as a string of absolute pitches, pitch classes, pitches in relation to a tonal centre or pitch intervals? Should rhythmic strings consist of durations or duration ratios? How about more abstract representations such as step-leap and contour pitch strings or shorter-longer-equal rhythm strings? How can structural prominence of some of the musical entities (e.g. more prominent notes in terms of duration length, harmonic content, metrical stress etc.) be taken into account?

Apart from issues relating to the selection of an appropriate representation of the musical surface, other issues arise as well. For instance, although approximate matching seems to be the obvious solution for capturing musical variation (e.g. filling and thinning of thematic material, rhythmic changes, pitch changes, tonal changes etc.), can exact matching account for such a phenomenon? Especially in relation to pattern induction, are exact repetitions and similarity ratings between musical patterns sufficient for extracting 'significant' patterns from a musical string? Should categorisation techniques be considered a necessary or an optional part of pattern induction methods? Is the pre-segmentation of a string necessary or even useful?

In the next sections most of these questions will be addressed and some possible solutions will be presented. First, problems in relation to the representation of musical strings will be discussed, then, some pros and cons of using exact or approximate matching techniques will be presented and, finally, the relevance of categorisation techniques and segmentation in pattern induction problems will be addressed.

2. Musical String Representation

There is a wide range of possible representations of musical strings that researchers can use as input to pattern processing algorithms. Often one representation is chosen as a first test case (e.g. absolute pitch) and then the assumption is made that the same string-matching mechanism can be applied to other representations (e.g. contour or pitch intervals). This assumption is often valid; however there are some caveats that the researcher should be aware of - some of these are discussed below.

2.1 Pitch Representation

Pitch is most often represented - in the western tradition - either by the traditional pitch naming system (e.g. F#4-G#4-A4) or as absolute pitch (e.g. in MIDI: 66, 68, 69). Most computer-aided musical applications adopt the absolute pitch representation. It has been argued in (Cambouropoulos, 1996) that the absolute pitch encoding is insufficient for applications in tonal music as it disregards the hierarchic importance of diatonic scale tones over the 12-tone discrete pitch space (e.g. enharmonic tones that have different tonal qualities are made equivalent).

As far as pattern matching is concerned, applications that use the MIDI representation sometimes resort to what will be referred to as *δ-approximate matching* in order to compensate for the information lost by the use of absolute pitch. In *δ-approximate matching,* equal-length patterns consisting of integers match if each corresponding integer differs by not more than δ - e.g. a C-major (60, 64, 65, 67) and a C-minor (60, 63, 65, 67) sequence can be matched if a tolerance δ=1 is allowed in the matching process (efficient algorithms for *δ-approximate* problems are currently studied by the Algorithm Design Group at the Department of Computer Science, King’s College London).

The main problem however with applying a pattern processing algorithm on an absolute pitch string is that transpositions are not accounted for (e.g. the repeating pitch motive in bars 1 & 2 in figure 1). And there is plenty of evidence, both theoretical and experimental, that transposition is paramount in the understanding of musical patterns. One partial solution that has sometimes been devised is to transpose different musical works (e.g. folk melodies) to the same key - this approach, however, does not account for transpositions of a pattern within the same piece and of course the whole idea of a musical work being in *one* key is problematic. The obvious solution to this problem is the use of relative pitch, mainly through the derivation of pitch intervals from the absolute pitch surface.

2.2 Pitch Interval Representation and Abstractions

Pitch intervals are adequate for representing relations between absolute pitches. Most commonly, computer systems make use of intervals that consist of a number of semitones. Cambouropoulos (1996) argues that this is insufficient for tonal music and proposes the *General Pitch Interval Representation (GPIR)* that can encode intervals according to the relevant set of scales in a given musical idiom. For instance, in figure 1 pitch-class intervals are inappropriate for revealing the repetition of the first two bars whereas name-class intervals *(nci)* - i.e. diatonic intervals in scale steps - are more adequate (see below for problems in this example).

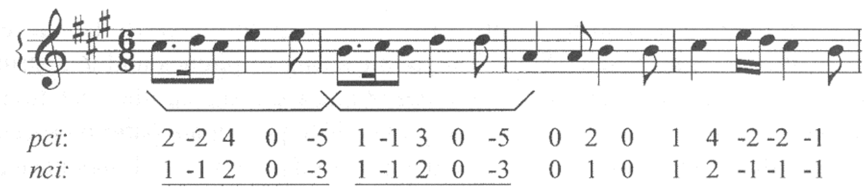

pci: 2 -2 4 0 -5 1 -1 3 0 -5 0 2 0 1 4 -2 -2 -1
nci: 1 -1 2 0 -3 1 -1 2 0 -3 0 1 0 1 2 -1 -1 -1

Figure 1 Beginning of theme of the A major sonata KV331 by Mozart (*pci:* pitch-class intervals, *nci:* name-class intervals). See text for discussion on the 'incorrect' pattern depicted here.

There exists a somewhat 'peculiar' relationship between pitch strings and pitch interval strings. As Rowe (1995) points out, if *one* note is altered within a string of notes then *two* corresponding intervals change. The converse also needs attention: if one pitch interval in a string of pitch intervals is altered then *all* the succeding notes are altered (transposed). So a change in a string of pitches and in a string of pitch intervals is not exactly the same thing. Take, for instance, the 'deletion' (or 'insertion') transformation commonly employed in approximate pattern processing techniques: the deletion of a pitch or the deletion of a pitch interval may have quite different effects on the transformed musical sequence (e.g. if the second pitch of the first bar of the melody in figure 1 is deleted a not very different pitch pattern C#-C#-E-E occurs; if the second pitch interval is deleted a rather more 'radical' change in the resulting pitch pattern C#-D-F#-F# occurs).

In terms of pattern induction techniques, the following problem arises as well: successive contiguous non-overlapping patterns in a string of pitch intervals result in overlapping patterns (by a single pitch) in the corresponding string of pitches. For instance, if a pattern induction algorithm that attempts to find an 'economic' non-overlapping description of the string is applied to the *nci* string of figure 1 - e.g. minimal length description methods such as (Annunziata et al, 1995) or grammar-induction-based compression methods such as (Nevill-Manning and Witten, 1997) - then the underlined pattern illustrated in figure 1 appears; at the pitch interval level these two patterns do not overlap wheras at the absolute pitch level they overlap by one note (see brackets in figure 1)! If a whole melody could be described in terms of contiguous non-overlapping pitch interval patterns then, at the note level, these consecutive patterns would overlap by one note resulting in a rather implausible description.

Pitch interval encodings readily lend themselves for constructing a number of more abstract representations of musical strings such as contour strings. Intervals can be categorised in a number of classes according to their sizes (e.g. **r**epeat: *nci*=0, **s**tep: *nci*=1, **l**eap: *nci*>1 and a string can be constructed from the alphabet {-l, -s, r, +s, +l} or according to the signs of intervals in which case contour can be represented as a string from the alphabet {-, +, =}. This way exact matching techniques can be applied for revealing 'approximate' matches.

In the example of figure 2, if the patterns are represented by absolute pitch no interesting matches occur; if encoded as pitch intervals in semitones then the first 5 intervals are matched; if encoded as step-leap strings then the whole patterns are matched (of course contours match as well but step-leap matching is more accurate). This pitch pattern repeats 12 times in this piece, each time transposed upwards by one semitone and at the same time the second-to-last and last pitches are transposed downwards by one semitone - 'evolution' algorithms such as (Crawford et al., 2000 – this issue) may be used to capture such gradually evolving transformations.

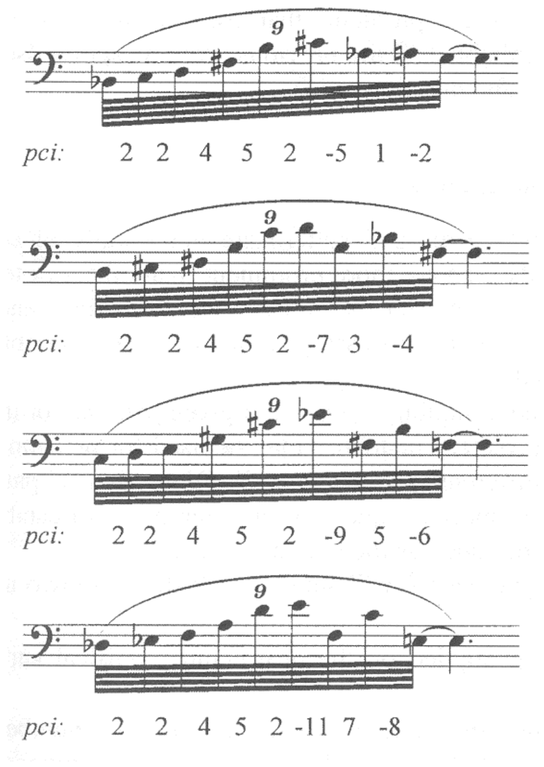
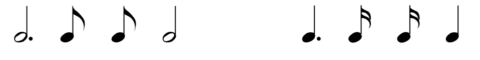


Figure 2 The first 4 occurrences of a motive from Messiaen's *Vingt Regards sur l'Enfant Jésus (III-L'échange).*

2.3 Rhythm Representation

In terms of the rhythmic component of musical strings, string processing algorithms are most commonly applied to strings of durations (or inter-onset intervals). This type of matching can be very effective, but one should also consider encoding rhythm strings as strings of duration relations such as duration ratios or shorter/longer/equal strings. Duration ratios encapsulate the observation that listeners usually remember a rhythmic pattern as a relative sequence of durations that is independent of an absolute tempo. Duration ratios can reveal augmentations or diminutions of a rhythmic pattern (figure 3).



dur.	12	2	2	8	6	1	1	4
ratios	1/6	1	4		1/6	1	4	

Figure 3. The above two rhythmic patterns match at the level of duration ratios.

It should be noted, however, that the problems that arise between pitch and pitch-interval representations (high-lighted in the previous section) apply also for the relationship between durations and duration ratios.

3. Matching of Structured Musical Patterns

The musical entities that constitute a musical pattern are not usually of equal salience, i.e. some notes (or chords etc.) are more prominent than others in terms of metrical position, duration length, register, harmony, tonal hierarchies and so on. In this section, ways in which pattern processing techniques may account for structured strings will be examined.

Exact pattern matching is aimed at finding instances of given patterns (or inducing identical patterns). However, pattern matching may be used for revealing or establishing *similarity* between different patterns as well. What kind of pattern matching methodology, though, is most adequate when attempting to establish similarities between complex entities such as melodic passages?

Simplifying for the sake of argument we will suppose that there are two main approaches:

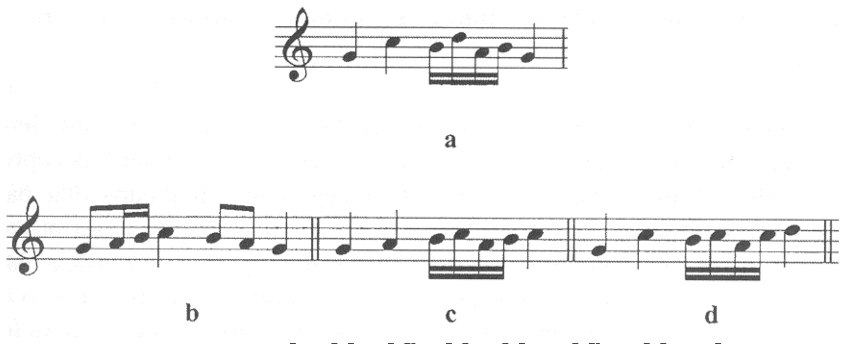
a) approximate pattern-matching applied on the unstructured musical surface and,

b) exact pattern-matching applied on the musical surface and on a number of reduced versions of it that consist of structurally more prominent components.

The first approach is based on the assumption that musical segments construed as being parallel (similar) will have *some* of their component elements identical (for example, two instances of a melodic motive will have a 'significant' amount of common notes or intervals but not necessarily all) - some approximate pattern-matching algorithms based on this approach are described in (Bloch and Dannenberg, 1985; Cope, 1990, 1991; Rowe and Li, 1995; Stammen and Pennycook, 1993; Rolland, 1998 - see Appendix). The second approach is based on the assumption that parallel musical segments are necessarily identical in at least one parametric profile of the surface or reduction of it (for example, two instances of a melodic motive will share an identical parametric profile at the surface level or some higher level of abstraction, e.g. pattern of metrically strong or tonally important notes/intervals and so on) - computational techniques based on this approach are described in (Cambouropoulos, 1998a; Hiraga, 1997).

What are the pros and cons of each of the above pattern-matching methodologies? Perhaps an example will help clarify the relative merits of each approach. Consider the tonal melodic segments of figure 4. How similar are segments *b, c, d* to segment *a*? Let us suppose, for convenience, that each melodic segment is represented as a sequence of pitch and onset time note tuples (figure 4, bottom).

Approximate pattern matching would show that each of the segments *b,c,d* is 71% identical to segment *a* as 5 out of 7 note tuples match. Depending on the threshold that has been set, the three melodic segments are equally similar - or dissimilar - to segment *a*. It is quite clear however to a musician that segment *b* is - for most tonal contexts - much more similar to segment *a* than any of the other segments because segments *a* & *b* match in exactly the 'right' way, i.e. more prominent notes match and less important ornamentations are ignored.



segment a:	[g,0],[c,4],[b,8],[c,9],[a,10],[b,11],[g,12]
segment b:	[g,0],[a,2],[b,3],[c,4],[b,8], [a,10], [g,12]
segment c:	[g,0],[a,4],[b,8],[c,9],[a,10],[b,11],[c,12]
segment d:	[g,0],[c,4],[b,8],[c,9],[a,10],[c,11],[d,12]

Figure 4. How similar are melodic segments *b, c, d* to segment *a*?

In order for the second pattern matching methodology to be applied, a significant amount of pre-processing is required - for instance, the melodic segments are not simply examined at the surface level but various more abstract levels of representation that reflect structural properties of the melodic segments have to be constructed (e.g. longer notes, metrically stronger notes, tonally important notes etc.). It should be noted, however, that it is possible to take account of structural prominence in approximate matching techniques by introducing *weights* to the matches of pattern elements - e.g. similarity contributions for each transformation especially in relation to duration length and pitch distance as proposed and implemented by Mongeau and Sankoff (1990) and Rolland (1998).

Both methodologies can handle musical similarity and parallelism. One advantage, however, of the second pattern-matching methodology is that the reasons for which two musical segments are judged to be parallel/similar are explicitly stated, i.e. the properties common to both are discovered and explicitly encoded. Such explicit knowledge may be used constructively for further analytic - or compositional - tasks.

4. Segmentation and Categorisation in Relation to Pattern Induction

4.1 Segmentation

Pre-segmentation of a musical work can increase significantly the efficiency of pattern induction techniques (see table 1 for researchers who favour this approach). However, committing oneself to a particular segmentation means that patterns crossing over boundaries are excluded *a priori*. This can be a serious drawback especially if one takes into account that often significant musical patterns contribute to the segmentation process itself, i.e. although there may be no strong indication for a point of segmentation, due, for instance, to a relatively long note or a relatively large melodic interval, a recurring musical pattern may indeed suggest a strong boundary at that point (see, for instance, boundary between first two bars of *Frère Jacques*).

Alternatively, an analytical methodology that relies solely on pattern recurrence is bound to find patterns that are cognitively and analytically implausible (e.g. a frequently repeating pattern may end on a very short note, or contain a long rest in the middle, and so on). It is suggested that pattern induction techniques should not rely heavily on a pre-segmented musical surface, but they should take into account methods that are geared towards finding perceptually-pertinent local boundaries as such boundaries can facilitate the selection process of 'significant' musical patterns. An integrated approach that takes into account both low-level discontinuities in the musical surface and higher-level emerging patterns has been proposed by Cambouropoulos (1998b)

4.2 Similarity and Categorisation

A further serious consideration regarding pattern induction is finding suitable criteria (e.g. weights for parameters such as pitch, rhythm and so on) for comparing musical sequences and setting an appropriate threshold for defining similarity between them. Most commonly such criteria and thresholds are selected in an *ad hoc* manner by the user/programmer.

Regarding parametric weights for contribution functions in musical sequence comparison tasks Rolland et al. (1996a) apply a supervised technique whereby the analytic results given by a human analyst are used to optimise the system's performance by finding the most appropriate weights for pitch and rhythm parameters.

Defining an appropriate threshold for determining which musical excerpts are similar - along with deciding which parameters contribute most in similarity judgements - is a very difficult issue. It has been proposed by Cambouropoulos and Smaill (1997) that similarity is always dependent on context and that it is essentially meaningless unless it is seen in association with processes of categorisation (usually the term similarity is merely considered to be inversely related to distance – an important further difference between the two is that the former requires a threshold). It is suggested that the notions of categorisation, similarity and the representation of entities/properties are strongly inter-related. It is not simply the case that one starts with an accurate description of entities and properties, then finds pairwise similarities between them and, finally, groups the most similar ones together into categories. It seems more plausible that as humans organise their knowledge of the world, they alter their representations of entities concurrently with emerging categorisations and similarity judgements.

Following this discussion on similarity and cate-gorisation the *Unscramble* algorithm (Cambouropoulos and Smaill, 1997) has been devised which, given a set of objects and an initial set of properties, generates a range of plausible classifications 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 classes or a specific similarity threshold or the relative prominence of properties. At every stage of the process both the extension and the intension of the emerging categories are explicitly defined.

5. Conclusions

In this paper, a number of issues relating to the application of pattern processing techniques on melodic strings have been addressed. Special emphasis was given to the various options and difficulties a researcher faces when trying to select an adequate representation of the melodic surface for pattern processing. Issues relating to the application of exact or approximate techniques on structured sequences were briefly discussed. Finally, the relevance of pre-segmentation and categorisation processes for pattern processing was addressed.

References

Apostolico, A. and Galil, Z. (eds) (1985) *Combinatorial Algorithms on Words.* Springer-Verlag, NATO ASI Series.

Bakhmutova, I.V., Gusev, V.D. and Titkova, T.N. (1997) The Search for Adaptations in Song Melodies. *Computer Music Journal,* 12(1):58-67.

Cambouropoulos, E. (1998a) *Towards a General Computational Theory of Musical Structure*. Ph.D. Thesis, University of Edinburgh.

Cambouropoulos, E. (1998b) Musical Parallelism and Melodic Segmentation. In *Proceedings of the XII Colloquio di Informatica Musicale*, Gorizia, Italy.

Cambouropoulos, E. (1996) A General Pitch Interval Representation: Theory and Applications. *Journal of New Music Research,* 25 (3): 231-251.

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.

Chou, T.C., Chen, A.L.P. and Liu, C.C. (1996) Music Database: Indexing Technique and Implementation. In *Proceedings of IEEE Intl. Workshop on Multimedia Data Base Management System.*

Crawford, T., Iliopoulos, C.S. , Winder, R. and Yu, H. (2000) Approximate Musical Evolution. *Computers in the Humanities*, 34:4.

Crawford, T., Iliopoulos, C.S. and Raman, R. (1998) String Matching Techniques for Musical Similarity and Melodic Recognition. *Computing in Musicology*, 11:71-100.

Cope, D. (1990) Pattern-Matching as an Engine for the Computer Simulation of Musical Style. In *Proceedings of the International Computer Music Conference*, Glasgow.

Coyle, E.J. and Shmulevich (1998) A System for Machine Recognition of Musical Patterns. In *Proceedings of IEEE ICASSP'98.*

Crochemore, M. (1981) An Optimal Algorithm for Computing the Repetitions in a Word. *Information Processing Letters,* 12(5):244-250.

Crochemore, M. and Rytter, W. (1994) *Text Algorithms*. Oxford University Press, Oxford.

Hiraga, Y. (1997) Structural Recognition of Music by Pattern Matching. In *Proceedings of the International Computer Music Conference,* Thessaloniki, Greece.

Hsu, J-L., Liu, C-C and Chen, A.L.P. (1998) Efficient Repeating Pattern Finding in Music Databases. In *Proceedings of the Conference in Information and Knowledge Management (CIKM'98),* Bethesda, Maryland.

McGettrick, P. (1997*) MIDIMatch: Musical Pattern Matching in Real Time.* MSc Dissertation, York University, U.K.

Mongeau, M. and Sankoff, D. (1990) Comparison of Musical Sequences. *Computer and the Humanities*, 24:161-175.

Nevill-Manning, C.G. and Witten, I.H. (1997) Compression and Explanation using Hierarchical Grammars. *The Computer Journal*, 40(2/3):103-116.

Rolland, P.Y., Ganascia, J.G. (1999) Musical Pattern Extraction and Similarity Assessment. In *Readings in Music and Artificial Intelligence*. E. Miranda. (ed.). Harwood Academic Publishers (forthcoming).

Rolland, P-Y. (1998) FlExPat: A Novel Algorithm for Musical Pattern Discovery. In *Proceedings of the XII Colloquium in Musical Informatics,* Gorizia, Italy.

Rolland, P-Y. and Ganascia, J-G. (1996a) Automated Motive-Oriented Analysis of Musical Corpuses: a Jazz Case Study. In *Proceedings of the International Computer Music Conference*, Hong-Kong.

Rolland, P-Y. and Ganascia, J-G. (1996b) Automated Extraction of Prominent Motives in Jazz Solo Corpuses. In *Proceedings of the 4th International Conference on Music Perception and Cognition (ICMPC'96),* Montreal.

Rowe, R. (1995) Artificial Intelligence and Musical Interaction. In *Proceedings of the International Congress in Music and AI,* University of Edinburgh, Edinburgh.

Rowe, R. and Li, T.C. (1995) Pattern Processing in Music. In *Proceedings of the Fifth Biennial Symposium for Arts and Technology*. Connecticut College, New London, Connecticut.

Smith, L.A., McNab, R.J. and Witten, I.H. (1998) Sequence-based Melodic Comparison: A Dynamic-Programming Approach. *Computing in Musicology*, 11:101-117.

Stammen, D.R. and Pennycook, B. (1993) Real-time Recognition of Melodic Fragments Using the Dynamic Timewarp Algorithm. In *Proceedings of the International Computer Music Conference (ICMC'93).*

Appendix

	pitch representation	rhythm representation	other structural factors	pre-segmentation required	type of pattern processing	type of matching	string matching algorithm
Mongeau & Sankoff, 1990	pitch/degree difference from tonal centre	durations	weights based on degree of consonance		comparison	approximate	dynamic programming
Stammen & Pennycook 1993	intervals in semitones	duration ratios	no	yes	recognition	approximate	dynamic programming
Smith,McNab Witten, 1997	intervals in semitones & contour	durations	no	no	recognition	exact & approximate	dynamic programming
Rowe, 1995 Rowe & Li 1995	intervals in semitones	durations	no	yes	recognition & induction	approximate	dynamic programming
Rolland 1996a,b 1998	absolute pitch & intervals	durations & ratios	contribution weights (e.g. long dur.)	no	induction	approximate	dynamic programming
Bakhmutova, Gusev, Titkova 1997	scale-step intervals		metric position	no	induction	approximate	dynamic programming
Cope 1990, 1991	intervals in semitones	durations	elimination of very short notes	no	m-length pattern induction	near-exact	brute-force algorithm
McGettrick 1997	abs. pitch, intervals in semitones	duration ratios	accented notes	no	recognition	exact	Boyer-Moore algorithm
Coyle & Shmulevich 1998	intervals in semitones (+error)	duration ratios (+error)	key-finding algorithm		comparison	exact (+ error absolute and perceptual)	equal length comparison
Hsu, Liu & Chen, 1998	absolute pitch	durations	elementary chords and metre	no	induction	exact	dynamic programming (only exact)
Hiraga 1997	interv: sem, scale-steps, step-leap, contour	durations: exact, log-ratio, shorter-longer-equal	reduction of surface	yes (tentative)	induction	exact (emphasis in immediate repetition)	not described
Cambouro-poulos 1998a,b	interv: sem, scale-steps, step-leap, contour	durations: exact, ratio, shorter-longer-equal	reduction of surface	no	induction	exact	Crochemore (1981)

Table 1 Left column indicates a number of musical pattern processing methods. Top row indicates some useful aspects of these methods (at least as far as this paper is concerned); first four entries refer to melodic representation issues and last three entries to aspects of pattern processing.