Darrell Conklin and Mathieu Bergeron
Department of Computing
City University London
United Kingdom
conklin@city.ac.uk

Feature Set Patterns in Music

Pattern discovery is an important part of computational music-processing systems. The discovery of patterns repeated within a single piece is an important step to segmentation according to thematic structures (Ruwet 1966). Patterns found within a few works may be signatures that can be instantiated for style emulation of novel musical material (Cope 1991; Rowe 1993) and can reveal a deep similarity in musical material. Patterns that are conserved across many pieces in a large corpus can represent structural building blocks and used for comparative style analysis and music genre recognition (Huron 2001; Conklin and Anagnostopoulou 2001; Lin et al. 2004).

Pattern discovery methods can be discussed according to the expressiveness of patterns—in particular, the levels of abstraction permitted by pattern components. Many approaches are restricted to a representation in which every pattern component is described using the same musical attribute: pitch, duration, interval, or fixed combinations of these (e.g., linked interval/duration, etc.). In these approaches, an event has only one possible representation, and therefore patterns can be efficiently found using general string algorithms (Gusfield 1997) after transforming the corpus to strings of attribute values.

Recent methods have considered whether this restriction can be relaxed by allowing patterns with heterogeneous components and subsumption relations among possible pattern components (Lartillot 2004; Cambouropoulos et al. 2005; Conklin and Bergeron 2007). The need for such patterns can be motivated with a few melodic fragments (see Figure 1) from the music of the famous twentieth-century French singer and songwriter Georges Brassens (1921–1981). In both pairs of fragments, the description of events by melodic interval or melodic contour alone is inadequate. Though the fragments within each pair have a common duration pattern, there is no melodic interval pattern that spans the complete fragments, though some events do have conserved melodic intervals.

This article describes a new approach to pattern representation and discovery in music, where pattern components can contain any number of features and can be as general or as specific as required by the data. An important concept in this work is *subsumption,* which provides a natural way to explore the pattern search space in a general to specific manner, pruning entire branches when patterns become infrequent. The space of patterns that may be discovered is very rich, and patterns have highly flexible levels of abstraction, much more than is possible with single attribute patterns.

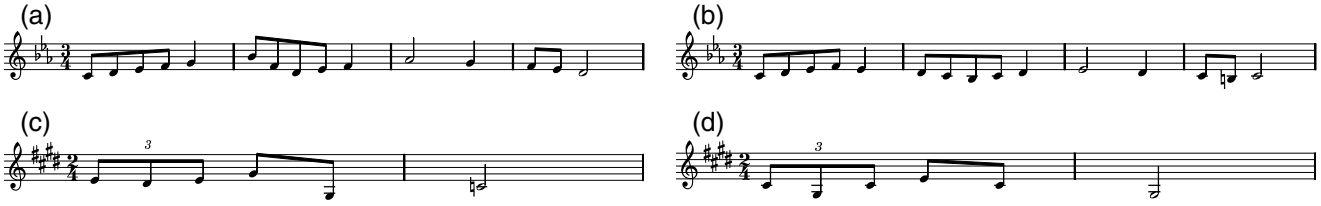
A well-known fact of knowledge representation is that increased expressiveness usually leads to increased time complexity of reasoning (Brachman and Levesque 2004). For pattern discovery in particular, allowing pattern components to contain a flexible and varying number of attributes substantially increases the size of the pattern search space. Though the size of this space is substantially reduced by the restriction to frequent patterns, when there are too many frequent patterns in the search space, heuristic algorithms must be employed. This article presents two algorithms to solve these pattern discovery problems: a complete algorithm and a heuristic probabilistic algorithm.

A large number of patterns can be revealed in a single piece or corpus, and these patterns must somehow be filtered and ranked for presentation. In this article, a statistical measure is used to order patterns according to the deviation of their observed from their expected frequency in the corpus.

The methods developed here are applied to the music of Georges Brassens. The style of Brassens, described by Keefe (2002) as "sophisticated simplicity," is well suited for pattern discovery studies. It did not change substantially throughout his career, and his output was prolific: he set over 170 texts to music, in addition to over a dozen texts of other well-known French poets.

Computer Music Journal, 32:1, pp. 60–70, Spring 2008

Figure 1. (a)–(b): First two phrases of Brassens's Le Bistrot; *(c)–(d): end of the first two phrases of Brassens's* Tonton Nestor.



Methods

This section provides the necessary definitions and an overview of the algorithms for pattern discovery. Some of the definitions and terminology are standard in the data mining literature (Agrawal and Srikant 1995; Ayres et al. 2002; Uno et al. 2003), but there are also a few novelties dealing specifically with music data mining. For reference, Table 1 provides a summary of notation and terminology used in this section.

Events and Viewpoints

An *event* is a music object within a sequence. Music objects can be notes, or structured objects such as simultaneities (Conklin 2002) and sequences (Conklin and Anagnostopoulou 2005). In this article, we focus on melody, and we only consider note objects. A *viewpoint* is a function that computes values for events in a sequence (Conklin and Witten 1995; Conklin 2006). For example, for the melodic interval viewpoint, the values in the range of the viewpoint are integers representing the difference in semitones between an event and its predecessor.

Table 2 provides a catalog of viewpoints used in this study. At the top of the table are the *primitive* viewpoints, namely, those used to select the basic attributes of notes. Following this are *derived* viewpoints that compute values using the primitive viewpoints. They can be further grouped into unary (computable from a single event, e.g., pitch class) and binary (requiring one preceding event to compute a value, e.g., melodic interval or melodic contour). A potentially unbounded set of derived viewpoints is available through the use of *constructors,* which

Table 1. Glossary of Terminology and Notation Used in This Study

Term	*Definition*
viewpoint	a function computing a property of a music event
feature	a viewpoint/value pair
$\perp$	an undefined value
feature set	a set of features representing a logical conjunction
$\{\}$	the empty feature set
closed feature set	no subsumee feature set has the same total count in the corpus
feature set pattern	a sequence of feature sets occurring in the corpus
$C_p(\cdot)$	the number of pieces having at least one occurrence of the specified feature set or pattern
$C_t(\cdot)$	the number of non-overlapping occurrences of a feature set or a pattern in the corpus
$E_t(\cdot)$	pattern expected total count
$I(\cdot)$	pattern interest measure
frequent	occurring in the corpus with at least the specified piece and total count thresholds
infrequent	occurring in the corpus below the specified piece count or total count threshold
maximal frequent pattern	a more specific frequent pattern cannot be found in the corpus

Table 2. Viewpoints Used in This Study

Viewpoint	*Description*	*Range Set*
pitch	pitch of event as MIDI number	{0, . . . , 127}
key	MIDI key signature as number of sharps (+) or flats (–)	{−7, . . . , +7}
onset	onset of event in MIDI ticks	$\mathbb{Z}^+$
duration	duration of event in MIDI ticks	$\mathbb{Z}^+$
ml1	first metric level	{t}
ml2	first or second metric level	{t}
pc	pitch class	{0, . . . , 11}
ref	pitch class of tonic, assuming major mode	{0, . . . , 11}
intref	pitch class interval from ref	{0, . . . , 11}
int	pitch interval	$\mathbb{Z}$
intu	unordered pitch interval	$\mathbb{N}$
intpc	pitch class interval	{0, . . . , 11}
intupc	unordered pitch class interval	{0, . . . , 6}
intscs	scale steps; natural or raised in major mode	$\mathbb{Z} \times$ {n, s}
intuscs	unordered scale steps	$\mathbb{N} \times$ {n, s}
repeat	contour class	{t}
up	contour class	{t}
down	contour class	{t}
step	contour class	{t}
leap	contour class	{t}
ioi	inter-onset interval in MIDI ticks	$\mathbb{Z}^+$
dc3	three-point duration contour	{<, =, >}
dr	duration ratio	$\mathbb{Q}^+$

Here, $\mathbb{N}$ denotes natural numbers (including 0), $\mathbb{Z}$ all integers, $\mathbb{Z}^+$ positive integers, $\mathbb{Q}^+$ positive rational numbers, and t "true."

build new viewpoints from existing ones. In this study, constructors are not employed, but rather a fixed set of derived viewpoints is used.

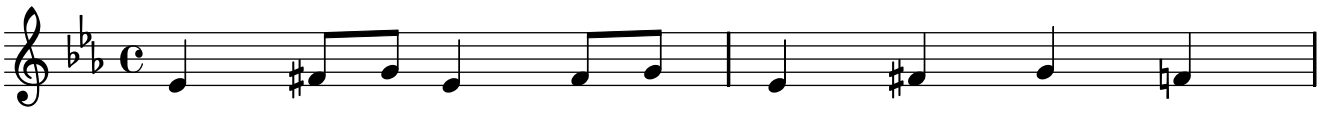
Viewpoints are used to transform sequences of concrete events into more abstract sequences. This is done simply by applying a viewpoint to every event in a sequence. Table 3 illustrates how events in a short melodic fragment (at a resolution of 192 ticks per beat) are transformed using the viewpoints of Table 2. The fragment is notated using three flats in the MIDI key signature (key:−3). Though this fragment (and the MIDI files for the Brassens corpus used in the current study) does not contain indication of mode or spelling of pitches, some quasi-diatonic features can still be computed from the MIDI key signature. To do so, major mode is assumed (e.g., E-flat major or ref:3), and quasi-diatonic intervals are computed between pitches standardized to that major key (whole scale steps receive a natural n and modified scale steps an augmented s sign on their interval).

Table 3 also illustrates how a hierarchical attribute is represented using feature sets. An event can occur in the highest metric level beginning a bar (ml1:t and ml2:t), only in a second level (ml2:t; in triple meter, both the second and third beats are in this level), or in neither (⊥). Note that any binary viewpoint is undefined (⊥) for the first event in the fragment, and that the intscs and inuscs viewpoints are defined only if the first pitch is a major scale degree. Some viewpoints (contour classes, metric level viewpoints) return only a possible value of t (true). Negated classes for these viewpoints (e.g., "not a step", "not in the highest metric level") can easily be handled, but they do not usually represent musically significant pattern components and are not used in this study.

Feature Sets and Patterns

A viewpoint/value pair is called a *feature.* These are notated by a viewpoint name and value separated by a colon, for example, pitch:63 or int:−4. A *feature set* represents a logical conjunction of features. A feature set is *instantiated* by an event if the event has every feature within the set; the *empty feature*

Table 3. Fragment of Brassens's *Une jolie fleur* Transformed Using the Viewpoints of Table 2



Primitive viewpoints										
pitch	63	66	67	63	66	67	63	66	67	65
key	−3	−3	−3	−3	−3	−3	−3	−3	−3	−3
onset	0	192	288	384	576	672	768	960	1152	1344
duration	192	96	96	192	96	96	192	192	192	192
ml1	t	⊥	⊥	⊥	⊥	⊥	t	⊥	⊥	⊥
ml2	t	⊥	⊥	t	⊥	⊥	t	⊥	t	⊥
Derived viewpoints (unary)										
pc	3	6	7	3	6	7	3	6	7	5
ref	3	3	3	3	3	3	3	3	3	3
intref	0	3	4	0	3	4	0	3	4	2
Derived viewpoints (binary)										
int	⊥	+3	+1	−4	+3	+1	−4	+3	+1	−2
intu	⊥	3	1	4	3	1	4	3	1	2
intpc	⊥	3	1	8	3	1	8	3	1	10
intupc	⊥	3	1	4	3	1	4	3	1	2
intscs	⊥	1s	⊥	−2n	1s	⊥	−2n	1s	⊥	−1n
intuscs	⊥	1s	⊥	2n	1s	⊥	2n	1s	⊥	1n
repeat	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
up	⊥	t	t	⊥	t	t	⊥	t	t	⊥
down	⊥	⊥	⊥	t	⊥	⊥	t	⊥	⊥	t
step	⊥	⊥	t	⊥	⊥	t	⊥	⊥	t	t
leap	⊥	t	⊥	t	t	⊥	t	t	⊥	⊥
ioi	⊥	192	96	96	192	96	96	192	192	192
dc3	⊥	<	=	>	<	=	>	=	=	=
dr	⊥	1/2	1/1	2/1	1/2	1/1	2/1	1/1	1/1	1/1

The onset time of the fragment has been shifted to time 0.

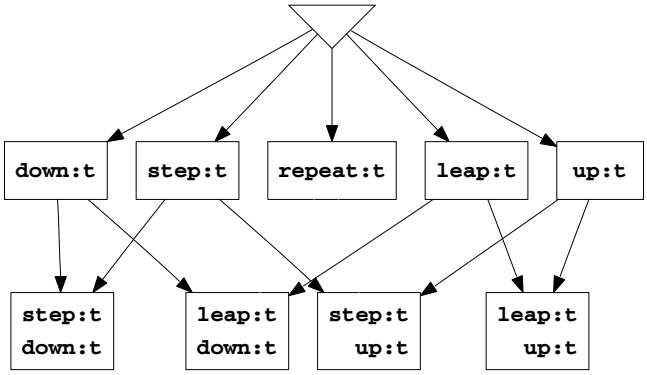
set `{}` is instantiated by any event. For example, the feature set `{pitch:63,int:-4}` is instantiated by the events at positions 4 and 7 of the melody fragment in Table 3. A feature set can be *specialized* by adding one or more features to the set. The more general feature set is said to *subsume* the specialized feature set: all instances of the specialized feature set are also instances of the more general feature set. For example, the feature set `{pitch:63}` subsumes the feature set `{pitch:63, int:-4}` which in turn subsumes `{pitch:63, int:-4, ml1:t}`.

The subsumption relation between feature sets can be visualized as a taxonomy in which nodes of the taxonomy are feature sets, directed links represent subsumption, transitive subsumption links are omitted, and a node is placed between its most specific subsumers and its most general subsumees.

Figure 2 shows a small feature set taxonomy using the five viewpoints `step`, `down`, `repeat`, `leap`, and `up`. These viewpoints are used to represent various levels of abstraction of melodic contour classes (Conklin and Bergeron 2007). Note that this taxonomy is not presented as a full lattice of subsets, because some feature sets (e.g., the set `{step:t, repeat:t}`) cannot be instantiated by any event and are therefore contradictory.

A *pattern* is a sequence of feature sets. A pattern

Figure 2. Taxonomy of melodic contours represented as feature sets. The inverted triangle represents the empty feature set.



occurs in (i.e., is instantiated by) a contiguous event sequence if the component feature sets of the pattern are instantiated by the corresponding events. For example, the pattern `[{int:+3,dc3:<},{int:+1}]` occurs at positions 2 and 5 of the melody fragment in Table 3. In a corpus, the *total count* of a pattern is the number of positions in which it occurs (not counting overlapping occurrences). The *piece count* of a pattern is the number of pieces that have one or more occurrences of the pattern. A *frequent pattern* is one that occurs with at least a specified minimum total count and piece count. Consequently, a frequent pattern will not contain infrequent feature sets as components. A *maximal frequent pattern* is a frequent pattern whose component feature sets cannot be further specialized without the pattern becoming infrequent.

Pattern Interest Measure

Patterns can be ranked according to the difference between observed and expected counts in a corpus (Conklin and Anagnostopoulou 2001; Huron 2001), similar to methods for ranking word collocations in natural language (Manning and Schutze 1999). Large differences between observed and expected counts indicate potentially interesting patterns. Here, a pattern P is given a pattern interest measure $I(P)$ that represents the ratio of observed to expected counts:

$$I(P) = \frac{C_t(P)}{E_t(P)} \qquad (1)$$

where $C_t(\cdot)$ represents the observed total count in a corpus and $E_t(\cdot)$ the expected total count. To define the expected total count, consider a feature set pattern $P = f_1, \ldots, f_n$, and let $N = C_t(\{\})$ (i.e., N is the total number of events in the corpus). The (non-overlapping) total count of the pattern can be no more than N / n (the number of events in the corpus divided by the length of the pattern). The expected total count in the corpus is this maximum quantity multiplied by the probability of the pattern:

$$E_t(P) = \frac{N}{n} \times p(f_1, \ldots, f_n) \qquad (2)$$

The probability of the pattern is computed using a zero-order model (assuming independence of successive events); this is the product of the relative frequencies within the corpus of the component feature sets of the pattern:

$$p(f_1, \ldots, f_n) = \prod_{i=1}^{n} \frac{C_t(f_i)}{N} \qquad (3)$$

The pattern interest can be computed rapidly since the raw feature set counts emerge directly from the first phase of the pattern discovery algorithm, as described below.

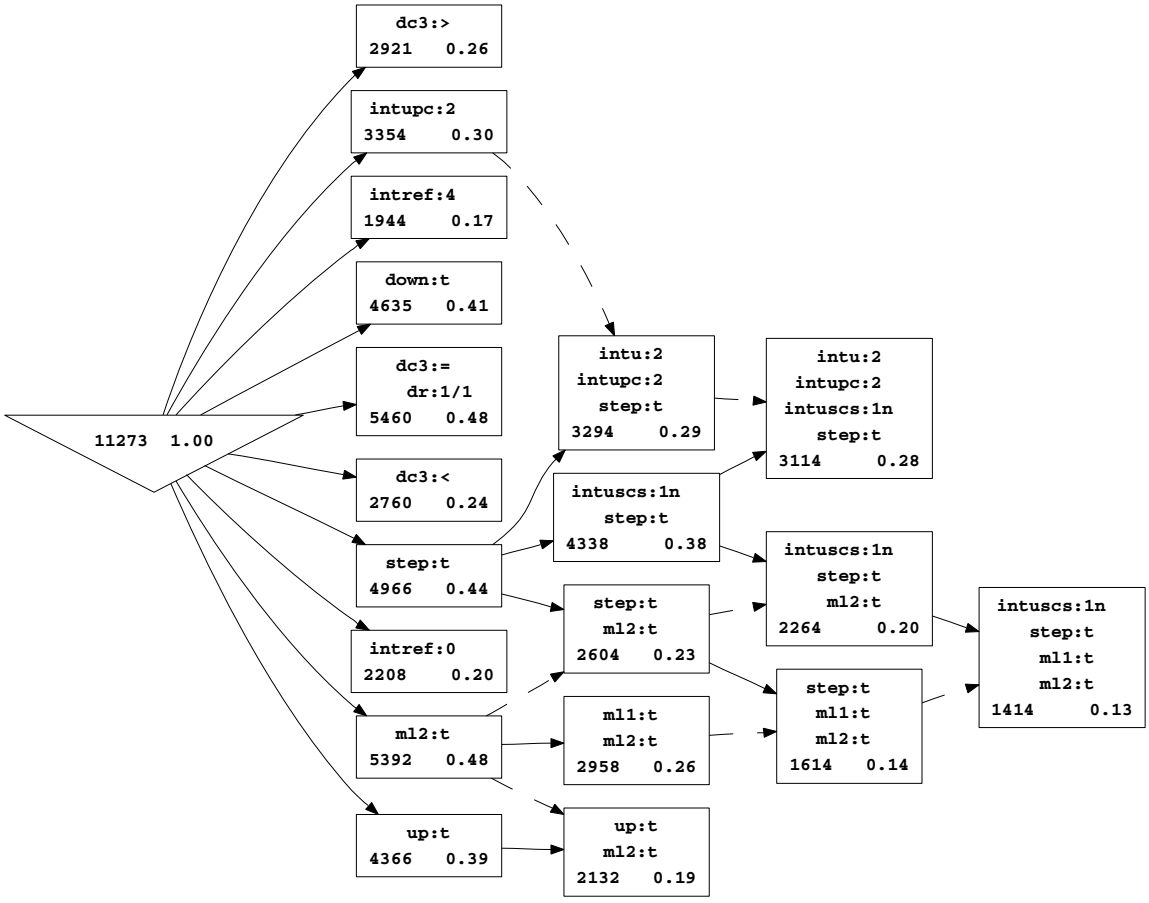
Pattern Discovery Algorithms

In this section, two algorithms for pattern discovery are developed. The first algorithm is complete and finds all maximal patterns, and the second is an optimization algorithm that uses a probabilistic hill-climbing approach and is suited to rapidly find a single interesting pattern in one or two pieces. This algorithm is heuristic and therefore guarantees neither that the pattern found will be maximal nor the most interesting, but it can be used to search large search spaces that may arise when the specified minimum total and piece count are low.

Complete Algorithm

A corpus (which may be one, two, or any number of pieces) is first transformed to feature-set sequences by *saturation*, i.e., applying each viewpoint in a catalog to every event in every piece (as illustrated

Figure 3. Taxonomy of closed frequent feature sets in Brassens, piece count 132.



in Table 3). The pattern discovery algorithm then proceeds in two phases. In the first phase, all frequent feature sets are found and configured into a taxonomy using a description logic classification algorithm (Brachman and Levesque 2004), which places each feature set between its most specific subsumers and its most general subsumees. Because a frequent pattern cannot have an infrequent feature set as a component, the potential feature-set space may be restricted to those feature sets that actually occur frequently in the corpus. Furthermore, it suffices to consider only *closed* feature sets, namely, those that do not subsume any other feature set with the same total count. This is because all components of a maximal pattern must by implication be closed feature sets. The set of all closed feature sets that are frequent in their total count can be efficiently computed using the method of Uno et al. (2003). This set is further pruned to those having at least the minimum specified piece count.

Figure 3 displays the subsumption taxonomy of closed feature sets from a corpus of 132 pieces comprising 11,273 events (the Results section describes this corpus in more detail), using the viewpoints provided in Table 2 and restricted to feature sets occurring in all 132 pieces. At the bottom of each node of the taxonomy are the total count of the feature set and the relative frequency of the feature set in the corpus. For example, 48% of all events have the feature `ml2:t`, and 19% of all events are further specialized to include the feature `up:t`.

The second phase of the discovery algorithm explores the specialization space of frequent feature set patterns in the search for maximal frequent pat-

terns. This space can be fully visited by recursively applying two refinement operators (Ayres et al. 2002) beginning at the empty pattern: an *I-step,* which specializes the right-most component of a pattern by walking one step down the feature set subsumption taxonomy (there may be several such specializations), and an *S-step,* which appends the empty feature set `{}` to a pattern. For example, referring to the closed feature set taxonomy of Figure 3, I-steps applied to the single component pattern `[{step:t}]` would yield the three patterns `[{intu:2, intupc:2,step:t}]`, `[{intuscs:1n,step:t}]`, and `[{step:t,ml2:t}]`. The S-step would yield the single pattern `[{step:t},{}]`.

A depth-first search using these refinement operators is used to explore the pattern space. All search nodes have an *instance map,* implemented as an associative array that maps piece/position pairs to a Boolean value indicating whether the pattern at the node occurs in the piece at the indicated position. When an S-step or I-step is applied, a new instance map is generated by intersection with the maps of the empty feature set (S-step) or the specialized feature set (I-step), taking into account an appropriate position offset. The empty feature set `{}` has a fully saturated instance map, as it occurs in every piece at every position. After the initial construction of the instance maps for all closed feature sets, the corpus does not need to be probed again, and pattern counts emerge directly from instance-map intersection. If the pattern counts indicate that a candidate specialization is infrequent, the current branch of the search can be safely terminated. This is because any pattern that is a specialization of an infrequent pattern will also be infrequent.

To avoid redundant computation, it is important to avoid visiting any pattern in the search space more than once. This can be achieved by converting the taxonomy into a tree after the first phase by breaking all but one subsumption link pointing into a node, thereby ensuring that no feature set is visited by more than one chain of I-steps. The dashed links in Figure 3 indicate one possible set of subsumption links that can be removed to produce a tree.

An S-step (appending the empty feature set) applied to a frequent pattern will usually produce only minimal changes to its instance map and will therefore usually lead to a frequent pattern. To avoid reporting maximal patterns that end with chains of empty feature sets, every node in the search space contains a pointer back to the node of the last I-step in the specialization chain, and when a maximal frequent pattern is reported, this link is followed to determine the appropriate pattern for presentation. Avoiding patterns beginning with an empty feature set is accomplished simply by requiring the first refinement applied in the search to be an I-step.

Heuristic Probabilistic Algorithm

The size of the search space of frequent patterns is determined by several factors including the number of viewpoints in the catalog, the size of the corpus, and the specified piece-count and total-count thresholds. For corpus analysis tasks in which patterns are required to occur in a substantial fraction of the pieces, entire branches of the search space can be pruned quite early. For larger search spaces, a heuristic probabilistic hill-climbing method can be used. This method is similar to the Gibbs sampling methods used for protein motif discovery (Lawrence et al. 1993). The interest measure (Expression 1) is computed for all candidate specializations (produced by all possible I-steps and the S-step), and one of these is sampled. The probability p of sampling a pattern P from the set Q of all candidate specializations is

$$p(P,Q) = \frac{I(P)}{\sum_{q \in Q} I(q)} \tag{4}$$

and the sampled pattern then becomes the new current pattern. The process begins at the empty pattern and is iterated until there are no further candidate specializations (i.e., until no candidate specialization is frequent). To reduce the effect of climbing to solutions with low pattern interest, a random restart method is employed. The probabilistic hill-climbing search is restarted a fixed number of times, and the most interesting pattern arising from all iterations is reported. In the heuristic algorithm, the taxonomy of frequent feature sets is not converted to a tree, because it is necessary to in-

spect all specializations of a pattern to compute the set Q (used in Expression 4).

Results

Georges Brassens's songs (Brassens 1993) in MIDI format were collected from various online sources and prepared for analysis. The onsets and durations of notes were appropriately quantized, the melody line was isolated, instrumental introductory and concluding material were removed, and repetitions of stanza and chorus material were also removed. Internal key and time signature changes were correctly handled. This process led to a corpus of 132 melodies, comprising a total of 11,273 notes. The corpus was then saturated as described herein with all 23 viewpoints presented in Table 2.

Three types of analysis are illustrated here using the pattern discovery method: corpus analysis (patterns found in many pieces), comparative analysis (patterns found in two pieces), and intra-opus analysis (patterns occurring two or more times in a single piece). Patterns were allowed to have at most one empty feature set, and they were required to have a pattern interest of greater than 1. For comparative and intra-opus analyses, the piece count was set to 2 and 1, respectively, the minimum total count was set to 2, and the heuristic probabilistic algorithm was used with 10 iterations.

Corpus Analysis

With a piece-count threshold of 132 pieces (i.e., requiring patterns to be present in all pieces in the corpus), there are 20 closed frequent feature sets (Figure 3) and only three general patterns are reported (the top three patterns of Table 4). With a piece-count threshold of 125 pieces (i.e., 95% of the corpus), there are 134 closed frequent feature sets, and 100 patterns with interest greater than 1 are found; three selected patterns are displayed in Table 4. The first pattern refers only to duration features, the second to a mix of duration and pitch features, and the last only to pitch features. The last pattern is chosen here to illustrate the results. This pattern has length 4 and covers an event sequence of length 5, because the first feature set contains the binary feature `up:t`. Instances of the pattern begin with a melodic motion upwards, then any event can follow and the pattern ends with an event having a downward melodic motion. In addition, the penultimate event in the sequence is further restricted according to a scale step feature and must be a step of exactly one scale degree. The pattern occurs in 126 pieces of the corpus and has a total count of 760.

Table 4. Some Feature Set Patterns Found in the Brassens Corpus

Pattern	C_p	C_t
`[{dc3:>},{dc3:<}]`	132	2114
`[{down:t},{up:t}]`	132	1935
`[{down:t},{ml2:t}]`	132	1852
`[{dc3:>},{dc3:<},{},` `{dc3:=, dr:1/1}]`	129	988
`[{step:t},{down:t,` `ml2:t},{dc3:<}]`	125	602
`[{up:t},{},{intuscs:1n,` `step:t},{down:t}]`	126	760

The top half shows three patterns found in all pieces; the bottom half contains three patterns found in at least 125 pieces.

The musical relevance of the pattern can be illustrated by two instances of the pattern within the song *Le vieux Léon* (see Figure 4: the pattern covers the last five events of each fragment). Note how the empty feature set in the second position is filled by a leap up in the first instance and by a step movement down in the second. Though the presented pattern is maximal for the corpus, it is not for the particular instances of Figure 4. The instances from *Le vieux Léon* could be captured more precisely by specialization, for example, adding the feature `intscs:-1n` to the last feature set. The resulting specialized pattern, however, would not be maximal, as it would be found within less than 95% of all pieces in the corpus. Finally, note that the third component of the pattern could be presented more succinctly, because the feature `intuscs:1n` implies the feature `step:t`. (This is further highlighted by the fact that the feature set

Figure 4. Two instances of the last pattern of Table 4 in Brassens's Le vieux Léon.

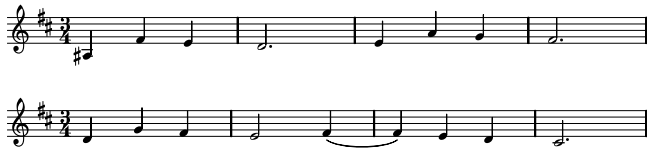
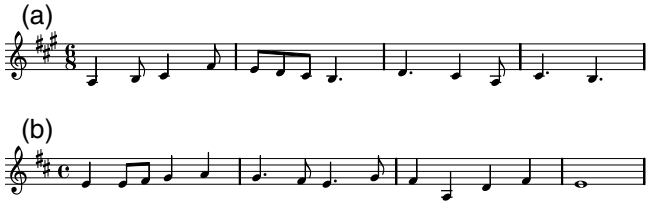


Figure 5. First two phrases of Brassens's (a) Les quatre bacheliers *and (b)* Rien à jeter.



`intuscs:1n` is not closed in the corpus; see Figure 3.)

The patterns in Table 4 all have an observed to expected ratio greater than 1, computed using the zero-order model of expectation (Expressions 2 and 3). To verify this against a different model of expectation, the Essen folksong collection (Schaffrath 1995), comprising approximately 6,000 songs containing a total of about 300,000 events, was scanned with each pattern in Table 4. All patterns were found to have higher relative total count in the Brassens corpus. Interestingly, the fifth pattern in Table 4 has a relative total count three times that of the Essen corpus. This pattern covers four events, with a melodic motion of a step, a motion down to the second metric level, followed by a note of shorter duration.

Comparative Analysis

Comparative analysis, taken here to mean the comparison of two pieces for shared patterns, is a very challenging task outside of the typical application of comparison of melodic variations of a theme. Figure 5 shows two phrases from Brassens's *Les quatre bacheliers* and *Rien à jeter.* The pieces have a similar overall form: four phrases, each having initial notes in a convex shape, followed by a few cadential notes. A pattern was found that aligns the first two phrases of the pieces; the feature set of one component of the pattern, covering the B3/E4 in the second bar, is

```
{intuscs:1n, intref:2, down:t, step:t, dc3:>,
  dr:3/1, ml2:t, duration:288, ioi:96,
  int:-2, intu:2, intpc:10, intupc:2,
  intscs:-1n}
```

This pattern component refers to several specific features: an interval of two semitones from the major tonic (`intref:2`), a melodic interval of –2 occurring at the second metric level, with a duration of a dotted eighth, three times that of the preceding note. Note again that this pattern component is not presented in the most compact form, as the melodic interval feature `int:-2` implies several other features (e.g., `intu:2`, `step:t`, `down:t`) that could be omitted to improve readability.

Intra-Opus Analysis

The motivating musical fragments presented earlier in Figure 1 were both identified using the probabilistic heuristic, pattern discovery method. In *Le Bistrot,* a pattern of length 15 is discovered, covering exactly the first two phrases of the piece. For example, the pattern components aligning the first and last events of the *Le Bistrot* fragments are the feature sets

```
{pitch:60, pc:0, intref:9, key:-3, ref:3,
  ml1:t, ml2:t, duration:96}
{key:-3, ref:3, step:t, dc3:>, ml2:t, ioi:96,
  intu:1, intupc:1}
```

The last feature set is satisfied by any event involving one scale step (either up or down) a key of –3 (major tonic 3, or E-flat major), a longer duration than previous note, and an inter-onset interval of one eighth-note in duration. This and all feature sets appearing in the pattern are naturally quite specific, as they are required only to occur twice in the piece.

In *Tonton Nestor,* a pattern of length 14 is discovered, covering the two fragments presented in Figure 1. For example, the pattern components aligning the first and last events of the *Tonton Nestor* fragments are the feature sets

```
{intuscs:r, key:4, ref:4, repeat:t, dc3:<,
  dr:1/3, ml1:t, ml2:t, duration:64,
  ioi:192, int:0, intu:0, intpc:0, intupc:0,
  intscs:0}
{key:4, ref:4, leap:t, dc3:>, dr:4/1, ml1:t,
  ml2:t, duration:384, ioi:96}
```

The last feature set refers to a key of 4 (major tonic E major), a duration of a half-note, and a leap to a longer event (with a duration of four times the previous note) on a strong beat.

Discussion and Future Work

This article has developed and applied a new pattern representation for music data mining. The representation, inspired by sequential data mining representations (Agrawal and Srikant 1995), allows any number of features in components of patterns, without restriction to a single viewpoint. The feature-set representation captures and extends recent innovations (Lartillot 2004; Cambouropoulos et al. 2005; Conklin and Bergeron 2007) in the field of musical pattern discovery in a simple and elegant way. For pattern discovery, pattern components may contain as many or as few features (including the possibility of an empty feature set) as are necessary to ensure that the whole pattern is frequent yet also maximal. The methods were illustrated on melodies by the French composer and songwriter Georges Brassens.

The tasks of melodic similarity (Mongeau and Sankoff 1990) and intra-opus pattern discovery in music are closely related; conserved patterns can indicate melodic similarity. In contrast to the work on melodic similarity and on sequential pattern mining (Agrawal and Srikant 1995), the pattern-discovery method developed here does not permit insertions or deletions of events while computing pattern occurrences. Such a facility could be introduced, but care should be taken in the algorithm to handle binary features that refer to previous events and to manage the increase in the complexity of pattern-occurrence counting.

The method used for ranking patterns is a statistical measure based on the deviation of observed from expected pattern total count in the corpus. In this study, for simplicity, the analysis corpus itself was used to construct a zero-order model of pattern component frequencies, which, though not optimal, does in practice yield some interesting patterns toward the top of the report. More realistic background models and interest measures employing relative frequencies in comparison repertoires could be explored.

The aim of this research is that an analyst should have available a large feature catalog and should not need to restrict *a priori* the features used to describe the data nor to predict how they will combine into interesting feature sets and patterns. The algorithm and results presented here suggest the practicality of moving towards this more expressive representation for music patterns.

Acknowledgments

An earlier report of this work was presented at the IJCAI 2007 Music-AI Workshop (Bergeron and Conklin 2007), and we thank the workshop organizers for making the event possible. Taxonomies were drawn with the Graphviz tool and music layout performed with the Lilypond package. Mathieu Bergeron is supported by a scholarship from Fonds Québécois de la Recherche sur la Nature et les Technologies (FQRNT) and a Ph.D. fellowship from City University London.

References

Agrawal, R., and R. Srikant. 1995. "Mining Sequential Patterns." *Proceedings of the Eleventh International Conference on Data Engineering.* New York: IEEE Computer Society Press, pp. 3–14.

Ayres, J., et al. 2002. "Sequential Pattern Mining Using a Bitmap Representation." *Proceedings of the International Conference on Knowledge Discovery and Data Mining.* New York: ACM Press, pp. 429–435.

Bergeron, M., and D. Conklin. 2007. "Representation and Discovery of Feature Set Patterns in Music." *Proceedings of the International Workshop on Artificial Intel-*

ligence and Music. San Francisco, California: Morgan Kaufmann, pp. 1–12.
Brachman, R., and H. Levesque. 2004. *Knowledge Representation and Reasoning.* San Francisco, California: Morgan Kaufmann.
Brassens, G. 1993. *Poèmes & Chansons.* Paris: Éditions du Seuil.
Cambouropoulos, E., et al. 2005. "A Pattern Extraction Algorithm for Abstract Melodic Representations That Allow Partial Overlapping of Intervallic Categories." *Proceedings of the Sixth Annual International Conference on Music Information Retrieval.* London: Queen Mary University of London, pp. 167–174.
Conklin, D. 2002. "Representation and Discovery of Vertical Patterns in Music." In C. Anagnostopoulou, M. Ferrand, and A. Smaill, eds. *Music and Artificial Intelligence: Lecture Notes in Artificial Intelligence 2445.* Berlin: Springer-Verlag, pp. 32–42.
Conklin, D. 2006. "Melodic Analysis with Segment Classes." *Machine Learning* 65:349–360.
Conklin, D., and C. Anagnostopoulou. 2001. "Representation and Discovery of Multiple Viewpoint Patterns." *Proceedings of the 2001 International Computer Music Conference.* San Francisco, California: International Computer Music Association, pp. 479–485.
Conklin, D., and C. Anagnostopoulou. 2005. "Segmental Pattern Discovery in Music." *INFORMS Journal on Computing* 18(13):285–293.
Conklin, D., and M. Bergeron. 2007. "Discovery of Generalized Interval Patterns." *Proceedings of the Fourth Sound and Music Computing Conference.* Lefkada, Greece: National and Kapodistrian University of Athens, pp. 149–152.
Conklin, D., and I. H. Witten. 1995. "Multiple Viewpoint Systems for Music Prediction." *Journal of New Music Research* 24(1):51–73.
Cope, D. 1991. *Computers and Musical Style.* Madison, Wisconsin: A-R Editions.
Gusfield, D. 1997. *Algorithms on Strings, Trees, and Sequences.* Cambridge, UK: Cambridge University Press.
Huron, D. 2001. "What is a Musical Feature? Forte's Analysis of Brahms's Opus 51, No. 1, Revisited." *Music Theory Online* 7(4). Available online at www.societymusictheory.org/mto/.
Keefe, S. 2002. "Sophisticated Simplicity: Text and Music in the Early Songs of Georges Brassens." *Tijdschrift voor Muziektheorie* 7(1):11–23.
Lartillot, O. 2004. "A Musical Pattern Discovery System Founded on a Modeling of Listening Strategies." *Computer Music Journal* 28(3):53–67.
Lawrence, C., et al. 1993. "Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment." *Science* 262(5131):208–214.
Lin, C.-R., et al. 2004. "Music Classification Using Significant Repeated Patterns." *Proceedings of the Eighth International Conference on Database Systems for Advanced Applications.* Berlin: Springer, pp. 508–518.
Manning, C., and H. Schutze. 1999. *Foundations of Statistical Natural Language Processing.* Cambridge, Massachusetts: MIT Press.
Mongeau, M., and D. Sankoff. 1990. "Comparison of Musical Sequences." *Computers and the Humanities* 24(3):161–175.
Rowe, R. 1993. *Interactive Music Systems: Machine Listening and Composing.* Cambridge, Massachusetts: MIT Press.
Ruwet, N. 1966. "Méthodes d'Analyse en Musicologie." *Revue Belge de Musicologie* 20:65–90.
Schaffrath, H. 1995. "The Essen Folksong Collection in the Humdrum Kern Format." Stanford, California: Center for Computer Assisted Research in the Humanities at Stanford University.
Uno, T., et al. 2003. "LCM: An Efficient Algorithm Enumerating Frequent Closed Item Sets." *Proceedings of the Workshop on Frequent Itemset Mining Implementations.* Aachen, Germany: CEUR Workshop Proceedings.