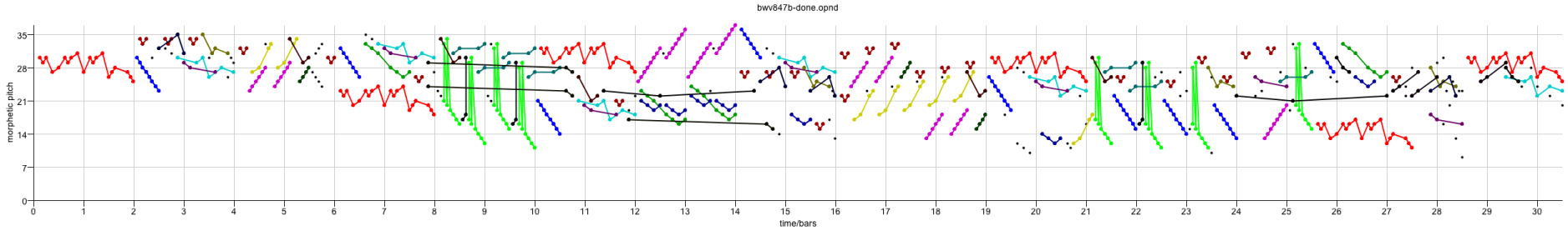
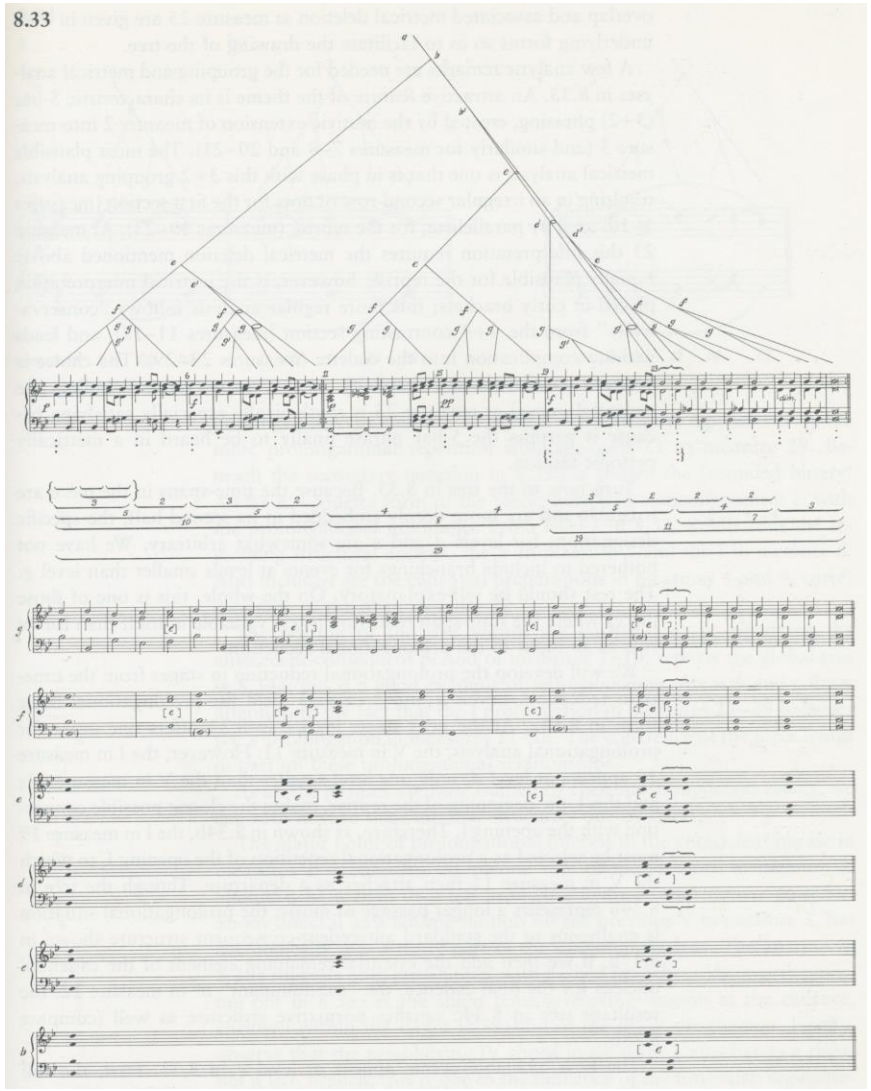


Geometric, Compression-based Pattern Discovery in Music

David Meredith

AALBORG UNIVERSITY
DENMARK

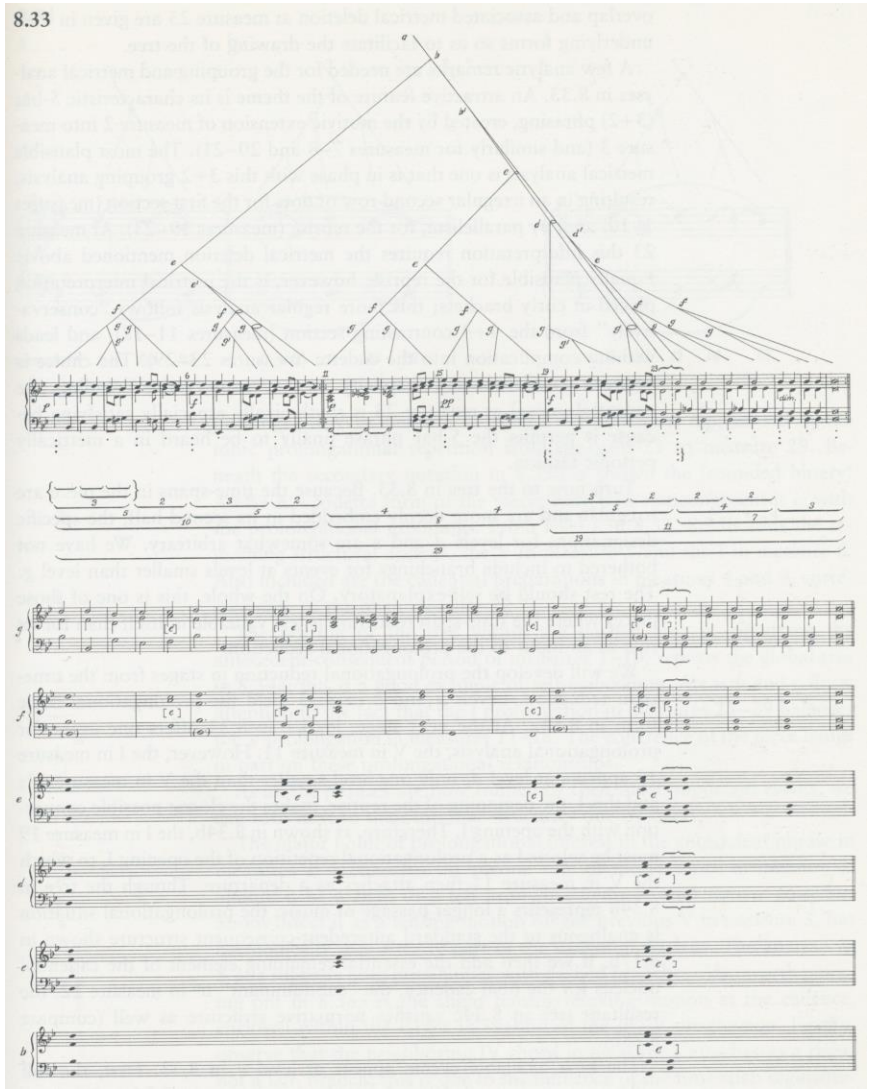
The goal of music analysis



Lerdahl and Jackendoff (1983, p.205)

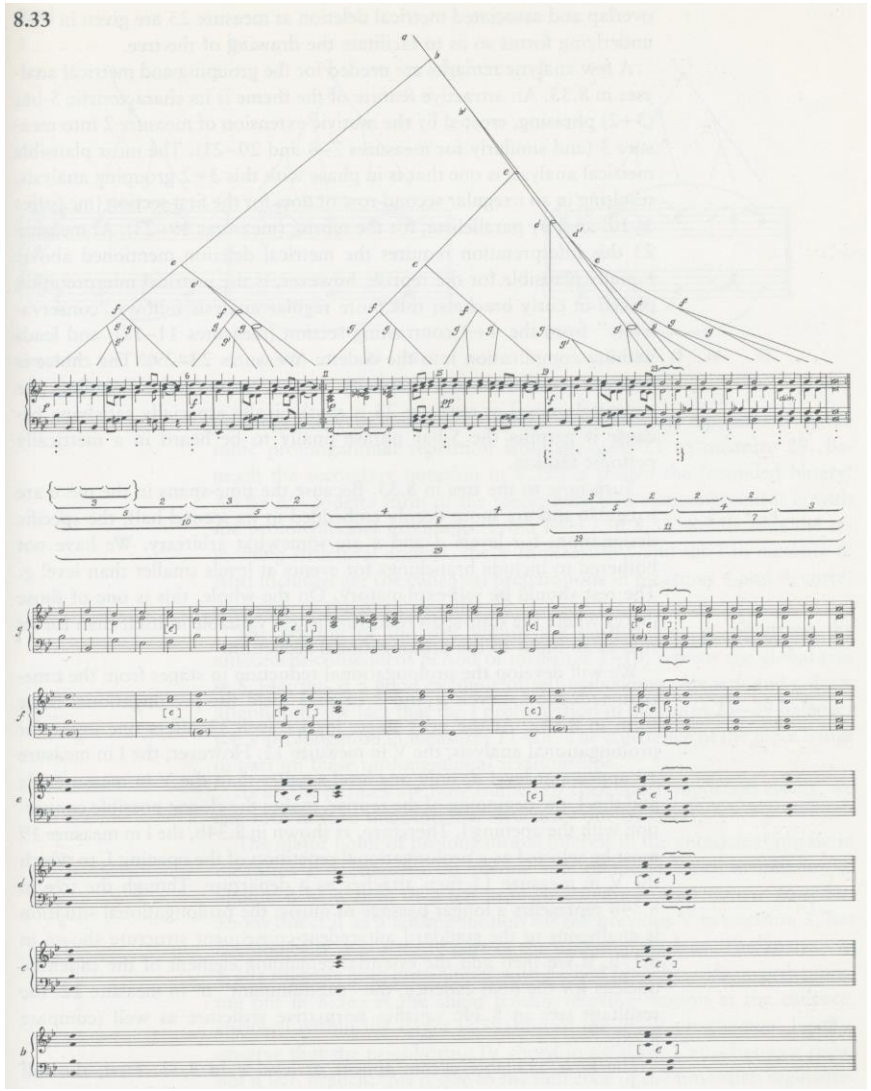
- A ***musical analysis*** is a representation of a way of understanding certain aspects of the structure of a musical object
- A ***musical object*** can be any quantity of music from a single note to a collection of works
- The ***analyst*** constructs a musical analysis from a given representation or encoding of a musical object that we'll call the ***musical surface***
- The goal of music analysis is to find the *best possible* explanations for musical works
- The "best possible" explanation for a musical work is one that allows you to
  - remember it most easily
  - identify errors most accurately
  - predict best what will come next

The musical surface



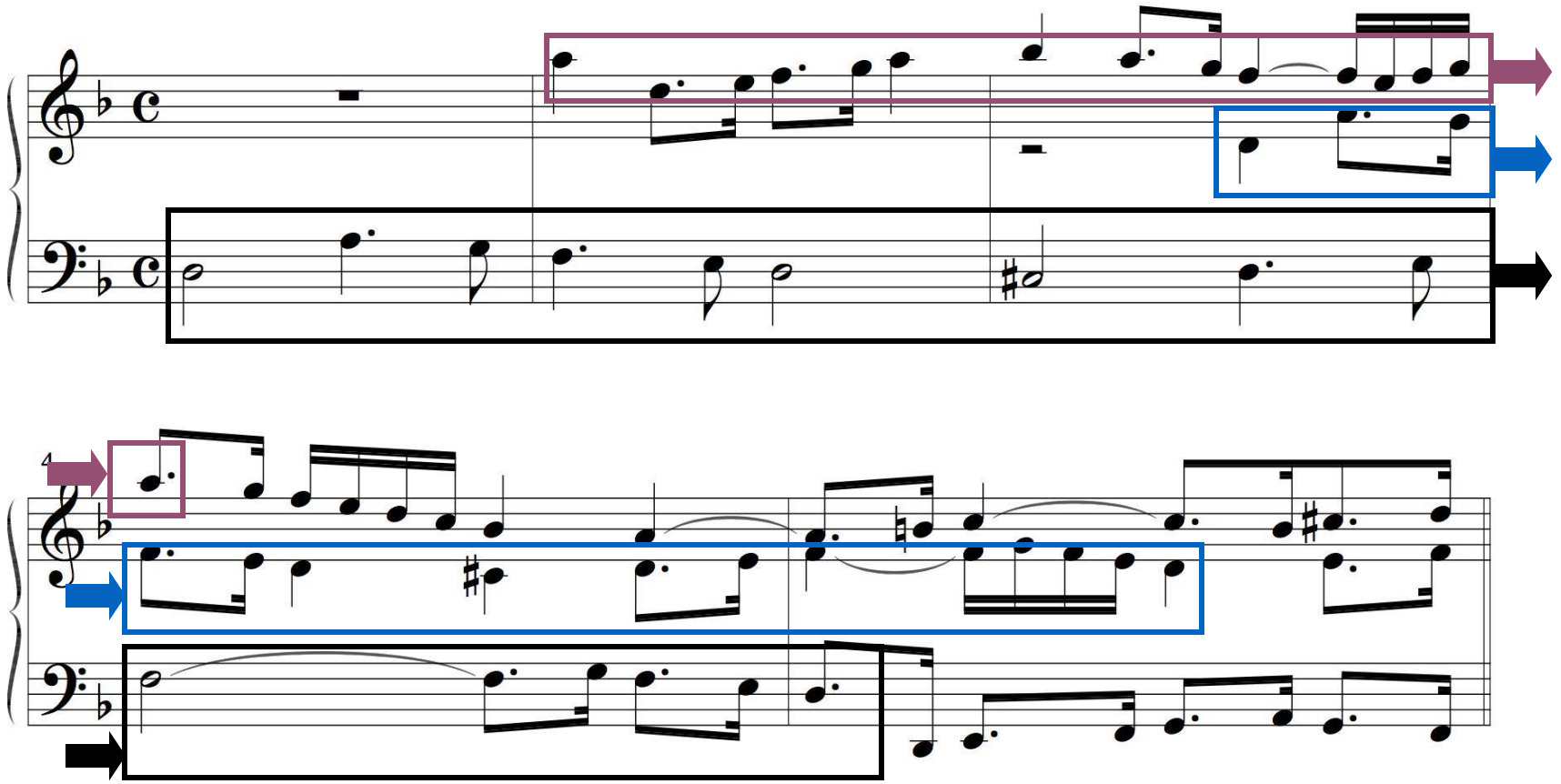
- The musical surface typically takes the form of an ***in extenso*** listing of a set of ***atoms***, in which the values of the properties of each atom are explicitly specified, for example,
  - in a notated score, each atom is a note and the properties of each note, such as when it starts and ends, its pitch and the instrument that should play it, are explicitly specified
  - in a PCM audio file, each atom is a sample and the time and value of each sample are explicitly specified
  - in a MIDI file, each atom is a MIDI event that has various properties depending on its type, e.g., a MIDI "Note On" event represents the start of a new note and specifies the note's onset time, note number and velocity
  - in a chord sequence, each atom is a chord that has properties such as its onset time (which may just be indicated by its position in the sequence), the root of the chord (which may be indicated by a roman numeral giving the root relative to a tonic) and the chord's quality (e.g., major, minor, diminished, dominant seventh)
- If the musical surface is an *in extenso* description of the musical object that consists of a list of independently specified atoms, then it represents neither groupings of these atoms into larger-scale constituents, nor any relationships between such constituents or between the atoms themselves
- In this case, the musical surface represents the musical object as though it is a ***meaningless***, ***randomly-selected*** collection of unrelated atoms

The musical surface



- The musical surface represents the musical object at some particular level of "***granularity***" or detail and typically focuses on only certain ***structural aspects*** of the musical object (e.g., rhythm, chords, melody, instrumentation, voice-leading, motivic/thematic structure)
- We assume the musical surface is a ***full and accurate*** description of those structural aspects of the musical object in which we are interested
- In other words, the musical surface is sufficient for communicating the structural information about the musical object that we're interested in understanding
- The musical surface typically (but not always) provides sufficient information for a *performance* of the musical object to be ***rendered*** either by a human (if the surface is, e.g., a score or lead sheet) or a machine (if the surface is, e.g., a MIDI file, magnetic tape, vinyl record or audio file)

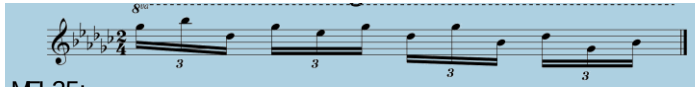
A musical analysis is a compressed description of a musical object



J. S. Bach, Contrapunctus 6 from *Die Kunst der Fuge*, BWV 1080

- Like a musical surface, a musical analysis is a description of a musical object
- The musical surface describes a musical object as though it is a meaningless, randomly-selected collection of independent atoms
- However, a musical analysis will typically group these atoms into meaningful structural units at various scales corresponding to what Lerdahl and Jackendoff call ***groups*** or
  - e.g., an analysis might describe a musical object as being constructed from notes grouped into chords, motives, themes, voices, phrases, sections, etc.
- A musical analysis will also typically describe ***relationships*** between these constituents, or, equivalently, ***transformations*** that map constituents onto each other
  - e.g., an analysis of a fugue might identify the subject and countersubject as important constituents and then enumerate the entries of these themes, specifying for each entry how it is related to the first entry by transformations such as inversion, retrograde, diminution, augmentation, etc.

A musical analysis as a program



```
MEL 25;
n1 = note(0, 90);                          // First note
p = coords(1, -1);                         // Corresponds
n = coords(1, 1);                          // Corresponds
ms1 = maskStructure(2, 2, 3);              // Triad mask
s1 = mask(6, 2, 2, 3, 2, 3);               // Background
s2 = mask(6, 2, 2, 1, 2, 2, 2, 1);         // Gb major sc
T1 = maskSequence(mask(0, 4, 2, 6));       // Background
T2 = maskSequence(mask(0, 1));             // Tatum time
T3 = maskSequence(mask(0, 2));             // Time mask s
P1 = maskSequence(s2, mask(3, ms1));       // Subdominant
P2 = maskSequence(s1);                     // Pitch mask
P3 = maskSequence(s2, mask(0, ms1));       // Tonic triad
S1 = space(T1, P2);                        // Background
S2 = space(T2, P3);                        // Space for f
S3 = space(T2, P1);                        // Space for v
S4 = space(T3, P3);                        // Space for v
v1 = vector(p, S1);                        //  \
v2 = vector(p, S4);                        //   | Vectors
v3 = vector(n, S2);                        //   |
v4 = vector(n, S3);                        //  /
Q1 = repeat(2, v1);                        // Sequence of
Q2 = repeat(2, v2);                        // Sequence of
R1 = product(v2, v3);                      // Cartesian p
R2 = product(Q2, v3);                      // Cartesian p
add(translate(n1,
                product(Q1,
                        sequence(R1,
                                 vectorSumSet(v4),
                                 R2))));
```

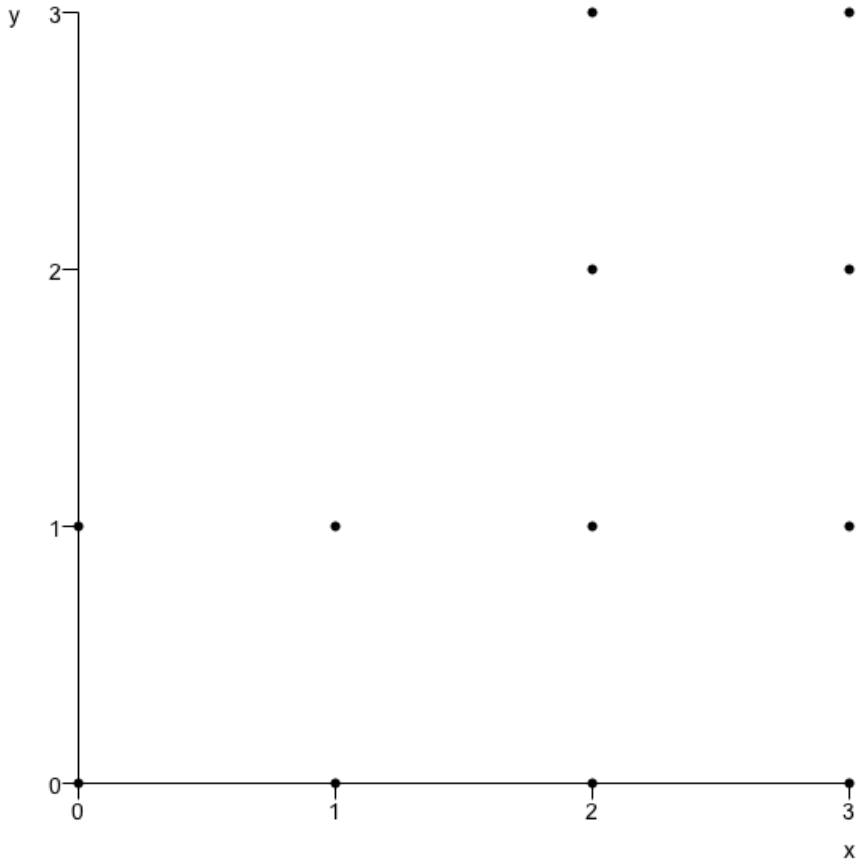
Meredith (2012)

- A musical analysis can be represented (or *encoded*) as a ***program*** or ***algorithm***
  - The program must generate an *in extenso* ***representation*** of the music to be explained as its ***only*** output
- The program is usually a ***compact*** or ***compressed*** encoding of its output
- The program is a ***description*** of its output
- If this description is ***short enough***, it becomes an ***explanation*** of its output
- An object is random and unpredictable if it is incompressible
- Ambiguity arises when there are two or more shortest descriptions

Program *length* as a measure of complexity

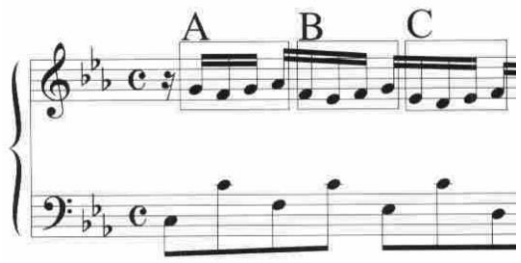
P(p(0,0),p(0,1),p(1,0),p(1,1),p(2,0),p(2,1),p(2,2),p(2,3),p(3,0),p(3,1),p(3,2),p(3,3))

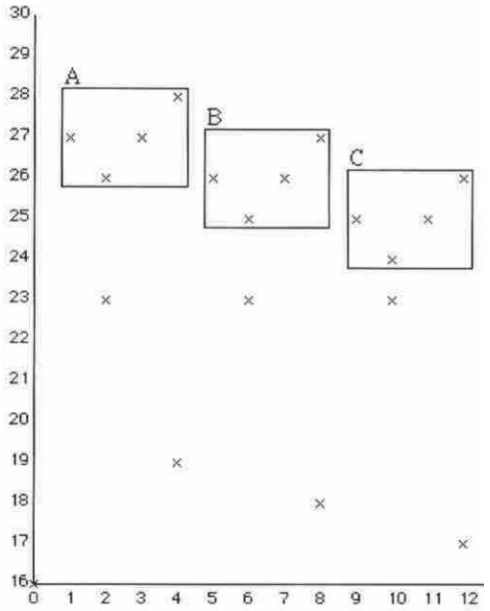
T(P(p(0,0),p(0,1),p(1,0),p(1,1)),V(v(2,0),v(2,2)))



- From Kolmogorov (1965) complexity theory:
  - can use the ***length*** of a program to measure the ***complexity*** of its corresponding explanation
  - The ***shorter*** the program, the ***simpler*** and ***better*** the explanation

Music analysis aims to compress music





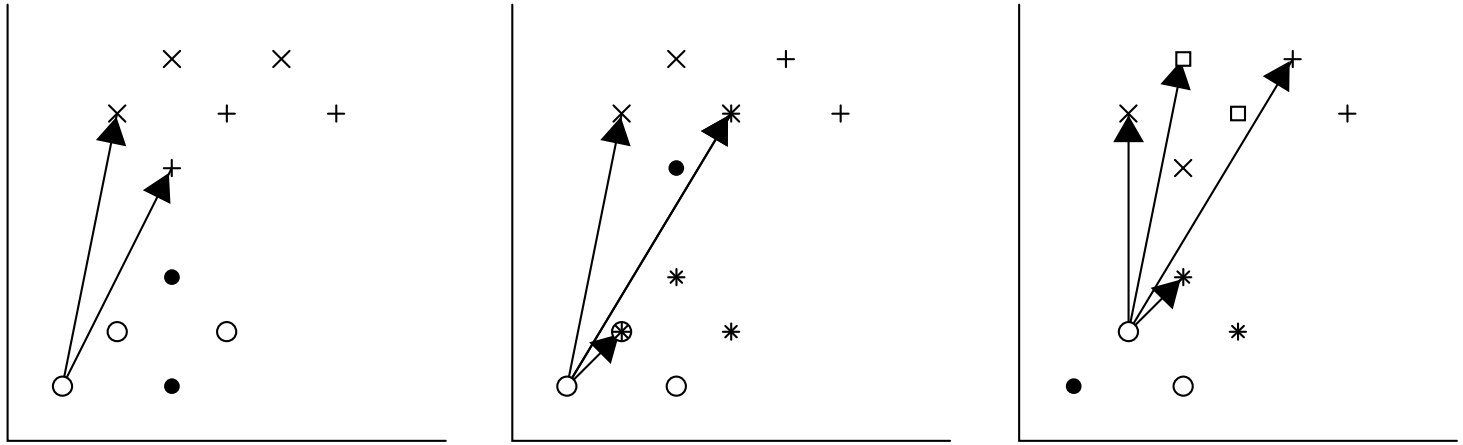
P(p(1,27),p(2,26),p(3,27),p(4,28),p(5,26),p(6,25),p(7,26),
p(8,27),p(9,25),p(10,24),p(11,25),p(12,26))

T(P(p(1,27),p(2,26),p(3,27), p(4,28)),V(v(4,-1),v(8,-2)))

- Since the best explanations are the shortest descriptions, the aim of music analysis is to ***compress*** music as much as possible

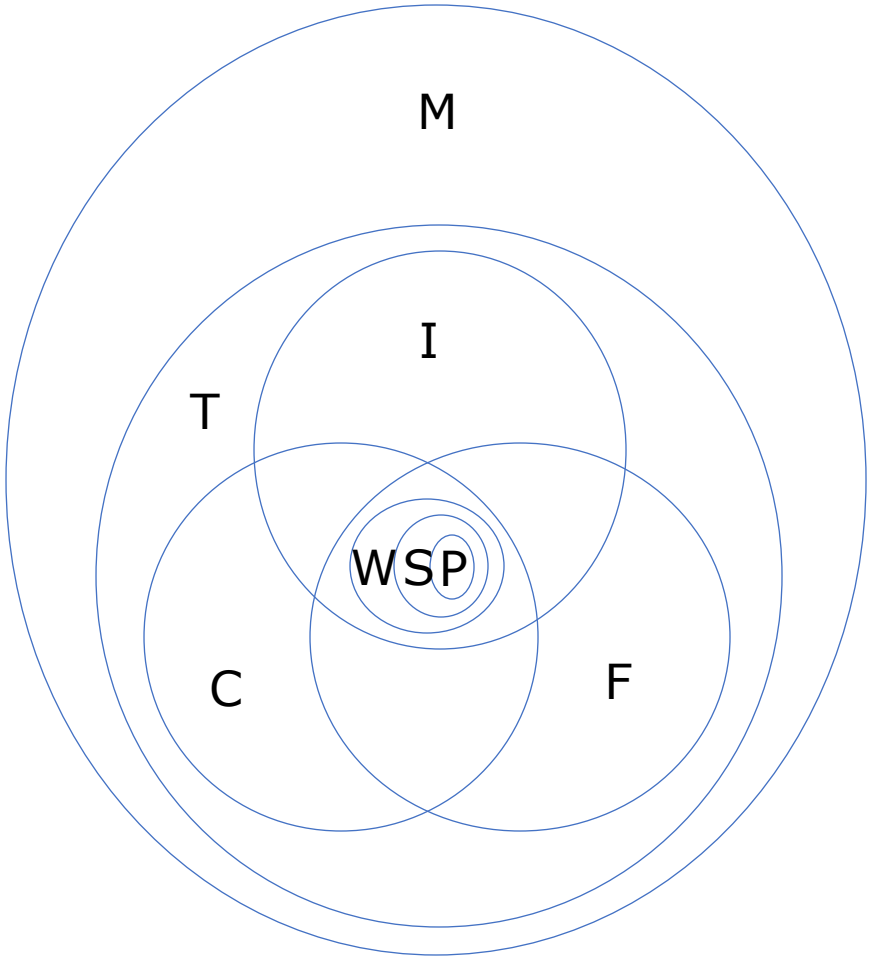
Meredith, Lemström and Wiggins (2002)

Compression and explanation



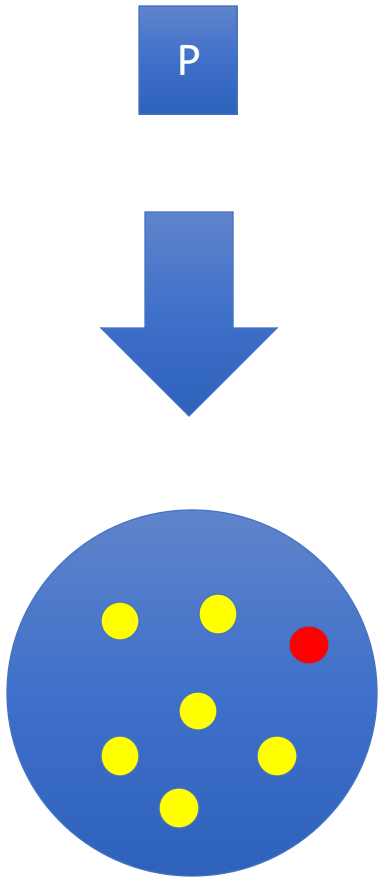
- Hypothesis: The shortest descriptions provide the best explanations
  - Ockham's razor, MDL, MML, Solomonoff's theory of inductive inference
- Suggests that knowledge can be extracted from data by finding short descriptions of it (i.e., by compressing it)
- Identifying the maximal repeated patterns and their occurrences is an excellent strategy for compressing data

Musical objects are interpreted in the context of larger containing objects



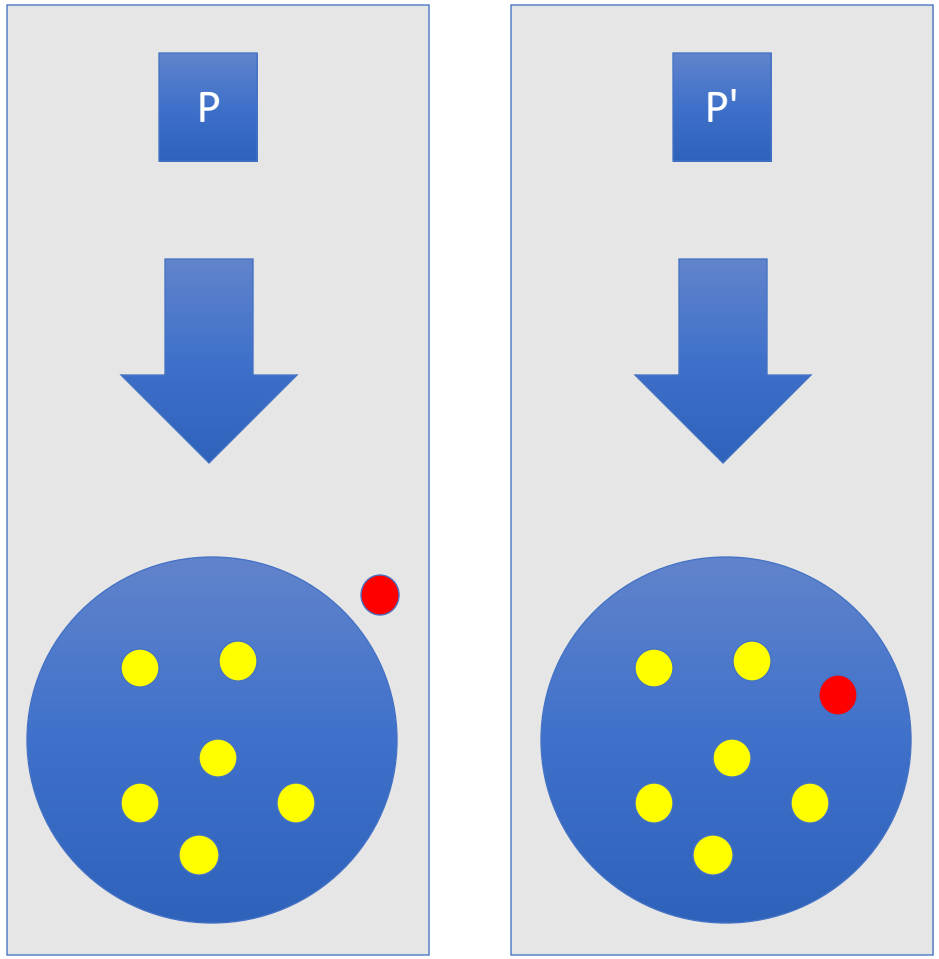
- A *musical object* (phrase, section, movement, work, corpus, ...) is usually interpreted ***within the context of*** some larger object that contains it
  - e.g., a work is often interpreted in the context of its composer's other works (C), or other works in the same genre or form (F) or other works for the same instrument(s) (I)
- Corresponds to a *two-part code*
  - characteristic function of the context set
  - codeword to identify object as member of this set

Music analysts look for short programs that compute *collections* of musical objects



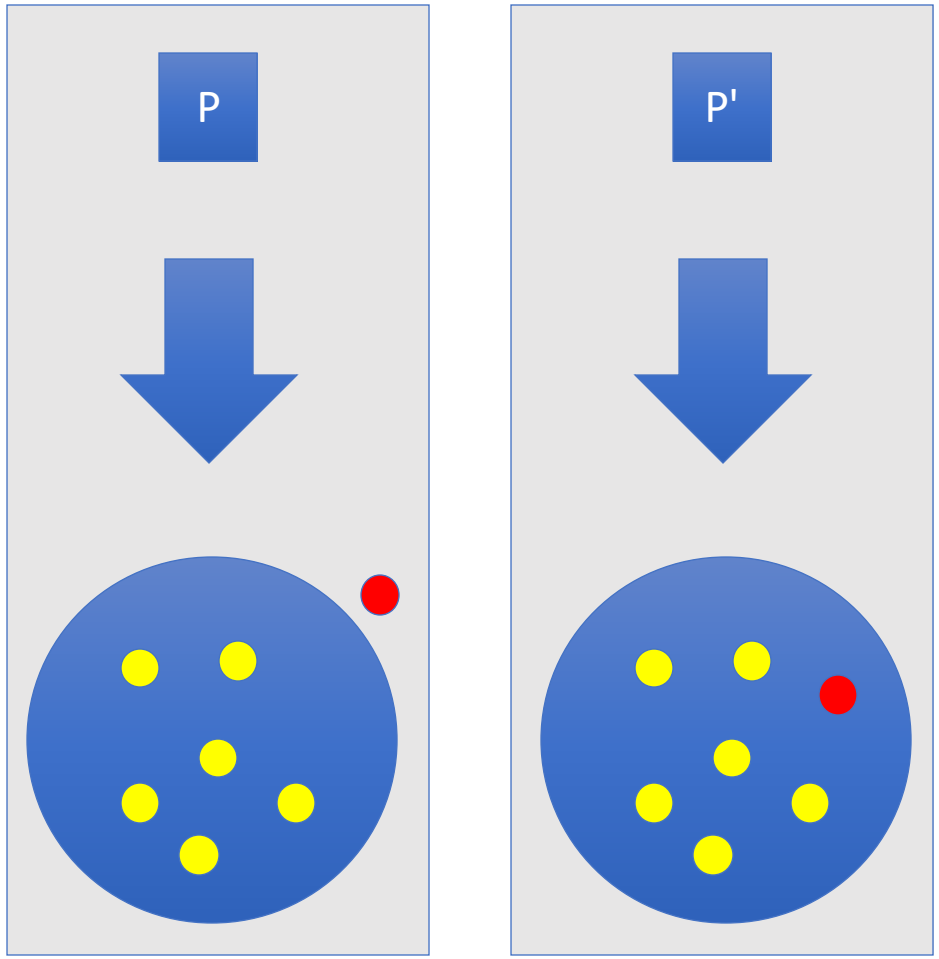
- The analyst tries to find the ***shortest*** program that computes a *set* of *in extenso* descriptions of
  - the object to be explained (the ***explanandum***)
  - other objects, related to the explanandum, defining a ***context*** within which the explanandum is to be interpreted

Listener interprets new music in the context of previously heard music



- When the (expert) listener interprets a new piece, the existing explanation (program), *P*, for all music previously heard is *modified* (as little as possible), to produce a new program, *P′*, to account for the new music *in addition*

Perceived structure represented by process by which object is generated

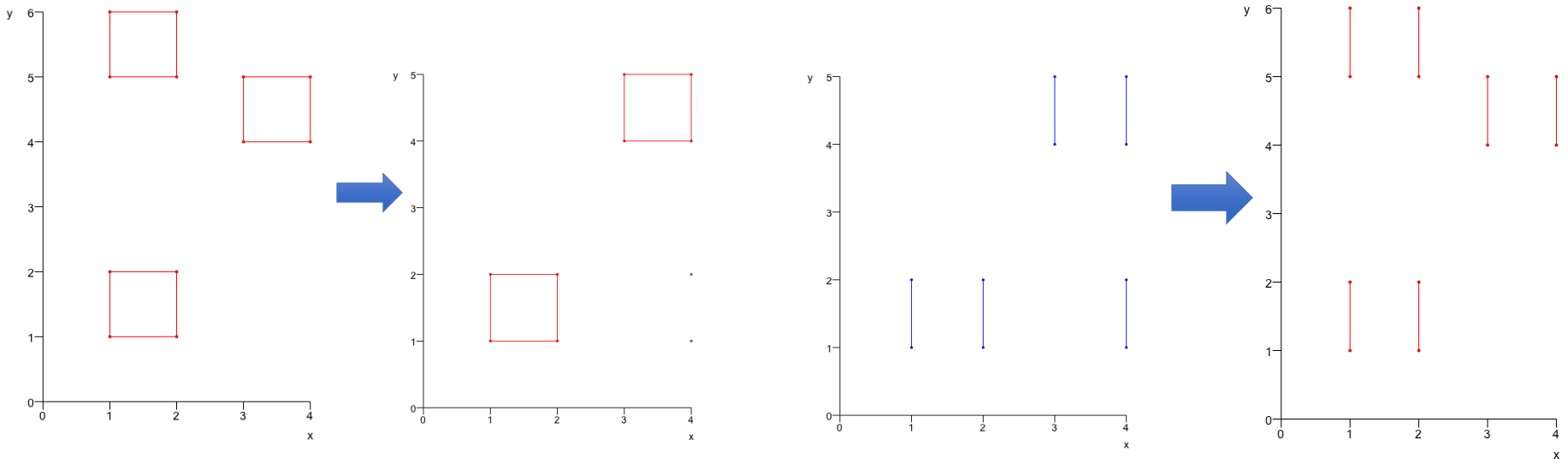


- Perceived structure of new musical object represented by specific way in which P' computes that object
- On this view, both music analysis and music perception are the ***compression*** of collections of musical objects

Perception and analysis are ***non-optimal*** compression

- Both analyst and (expert) listener ***aim*** to find shortest encodings
- Neither analyst nor listener achieve this aim in general
- Hampered by limitations of memory and perceptual system
    - e.g., require recognizable patterns to be fairly *compact* in pitch-time space (Collins *et al*., 2011)

Individual differences depend on order of presentation of context

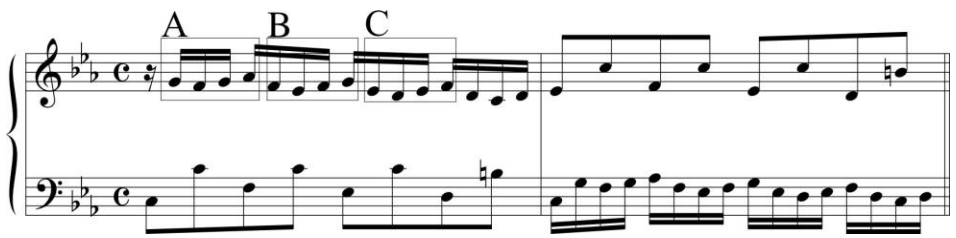


- Prefer to re-use previous encodings wherever possible
  - "greedy algorithm": means that way in which a new object is understood depends on the order of presentation of previous objects
- Implies that each individual will have a different interpretation of the same musical object that depends, not only on *what* previous music has been heard, but also the *order* in which it was encountered

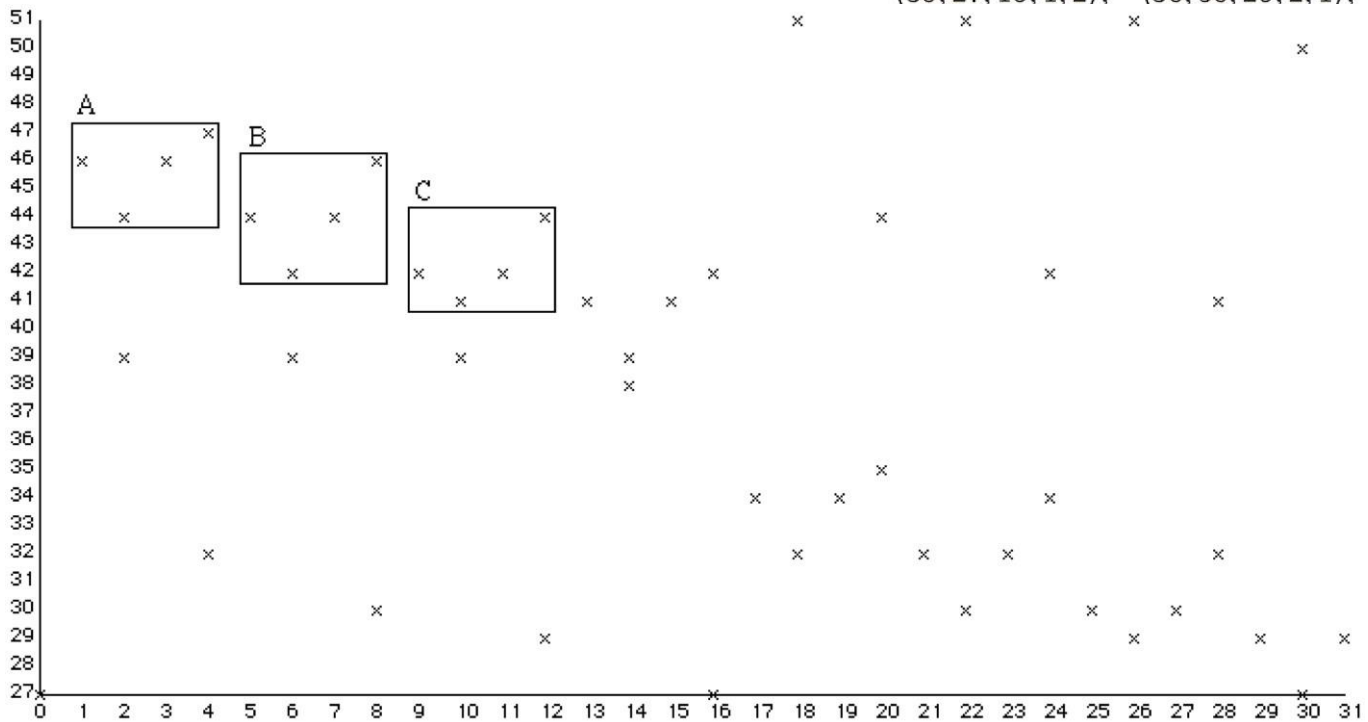
Music-theoretic compression strategies

- Voices
  - Most notes within a voice end at the start of the next note – implies that offsets only have to be encoded in exceptional cases – reduces encoding length
- Metre
  - More frequently occurring time point categories encoded by short labels, corresponding to higher levels in a metrical hierarchy
  - Compare Shannon–Fano coding
- Pitch
  - More frequently occurring pitch classes encoded with shorter labels, corresponding to higher levels in a tonal hierarchy

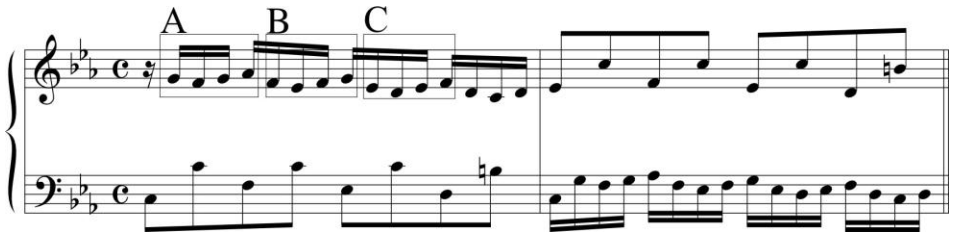
Using multidimensional point sets to represent music (1)



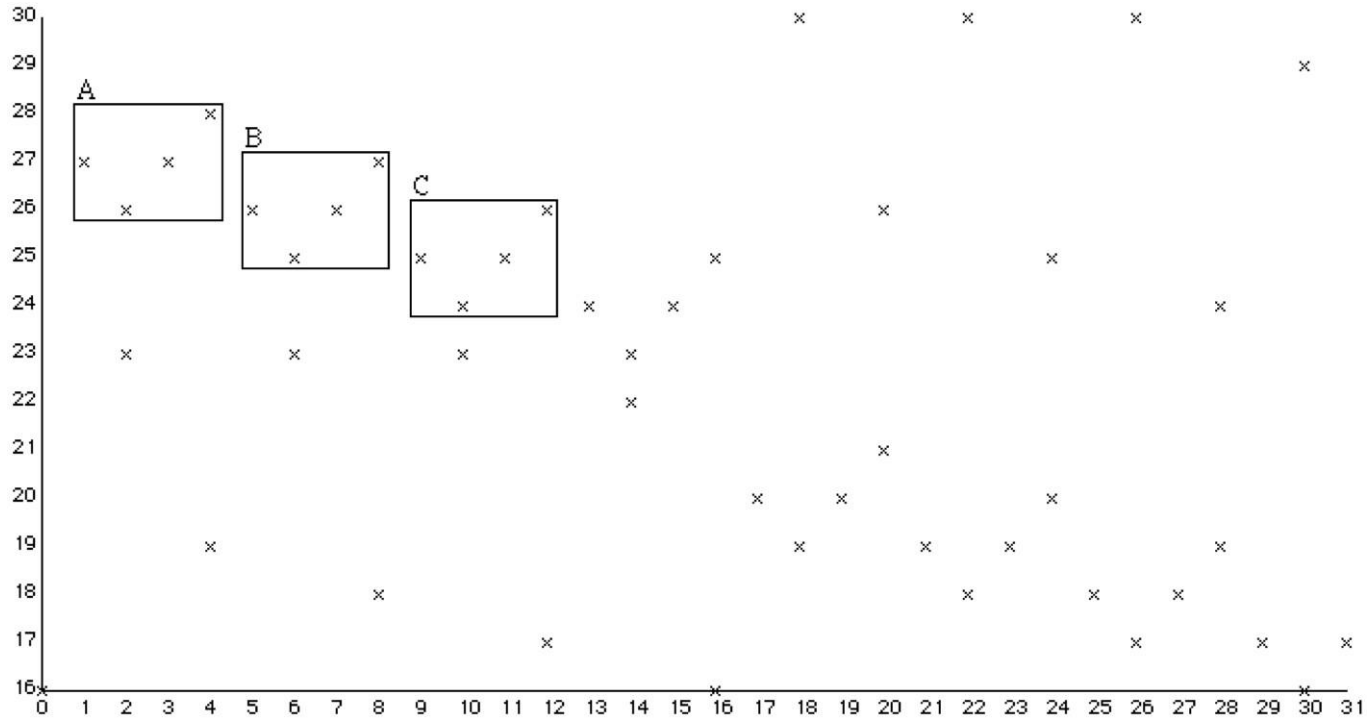
$$\{\ \langle 0,27,16,2,2\rangle,\ \langle 1,46,27,1,1\rangle,\ \langle 2,39,23,2,2\rangle,$$
$$\langle 2,44,26,1,1\rangle,\ \langle 3,46,27,1,1\rangle,\ \langle 4,32,19,2,2\rangle,$$
$$\langle 4,47,28,1,1\rangle,\ \langle 5,44,26,1,1\rangle,\ \langle 6,39,23,2,2\rangle,$$
$$\langle 6,42,25,1,1\rangle,\ \langle 7,44,26,1,1\rangle,\ \langle 8,30,18,2,2\rangle,$$
$$\langle 8,46,27,1,1\rangle,\ \langle 9,42,25,1,1\rangle,\ \langle 10,39,23,2,2\rangle,$$
$$\ldots\ \ \ldots\ \ \ldots$$
$$\langle 26,29,17,1,2\rangle,\ \langle 26,51,30,2,1\rangle,\ \langle 27,30,18,1,2\rangle,$$
$$\langle 28,32,19,1,2\rangle,\ \langle 28,41,24,2,1\rangle,\ \langle 29,29,17,1,2\rangle,$$
$$\langle 30,27,16,1,2\rangle,\ \langle 30,50,29,2,1\rangle,\ \langle 31,29,17,1,2\rangle\ \}$$



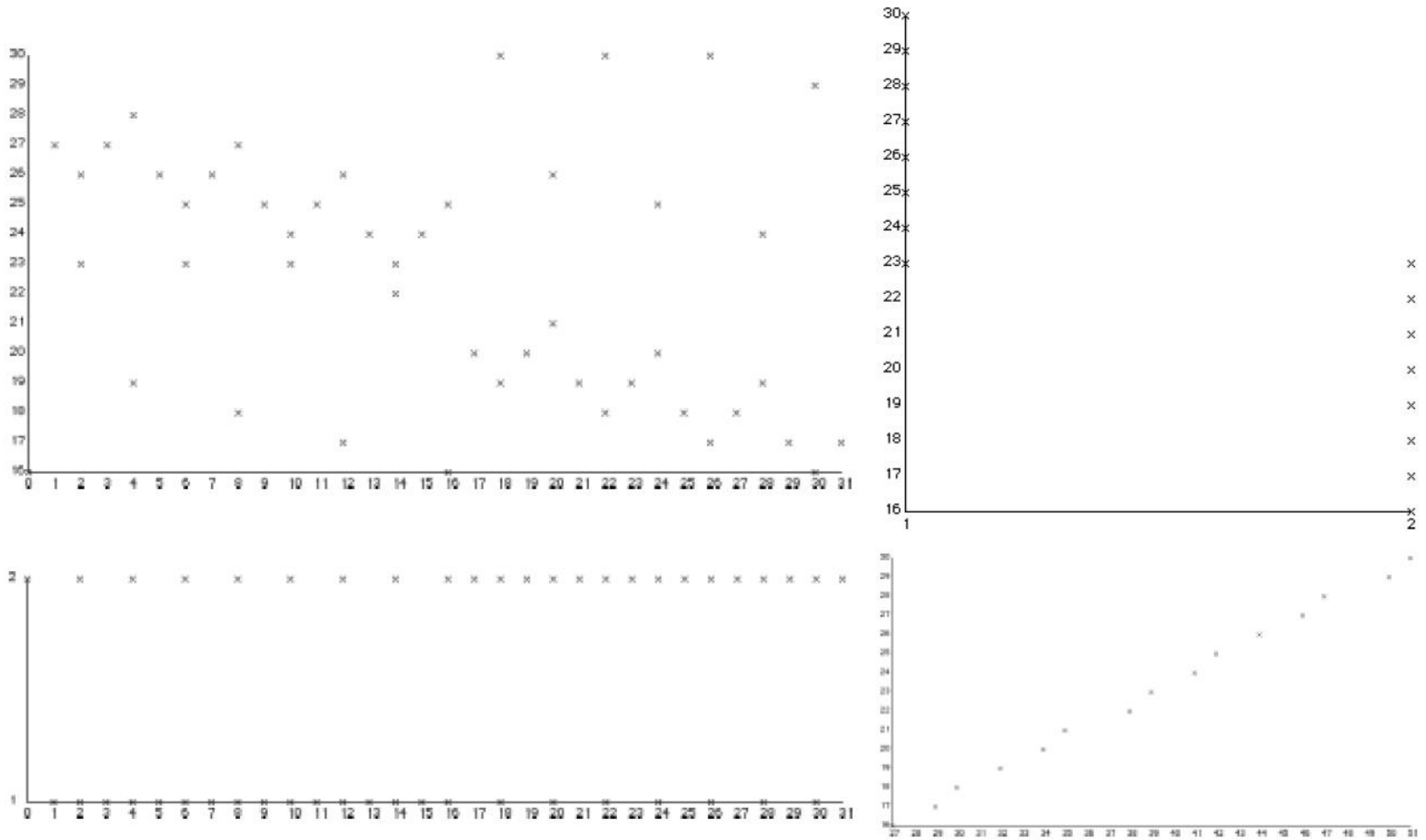
Using multidimensional point sets to represent music (2)



$$\{\ \langle 0,27,16,2,2\rangle,\ \langle 1,46,27,1,1\rangle,\ \langle 2,39,23,2,2\rangle,$$
$$\langle 2,44,26,1,1\rangle,\ \langle 3,46,27,1,1\rangle,\ \langle 4,32,19,2,2\rangle,$$
$$\langle 4,47,28,1,1\rangle,\ \langle 5,44,26,1,1\rangle,\ \langle 6,39,23,2,2\rangle,$$
$$\langle 6,42,25,1,1\rangle,\ \langle 7,44,26,1,1\rangle,\ \langle 8,30,18,2,2\rangle,$$
$$\langle 8,46,27,1,1\rangle,\ \langle 9,42,25,1,1\rangle,\ \langle 10,39,23,2,2\rangle,$$
$$\ldots\ \ \ldots\ \ \ldots$$
$$\langle 26,29,17,1,2\rangle,\ \langle 26,51,30,2,1\rangle,\ \langle 27,30,18,1,2\rangle,$$
$$\langle 28,32,19,1,2\rangle,\ \langle 28,41,24,2,1\rangle,\ \langle 29,29,17,1,2\rangle,$$
$$\langle 30,27,16,1,2\rangle,\ \langle 30,50,29,2,1\rangle,\ \langle 31,29,17,1,2\rangle\ \ \}$$

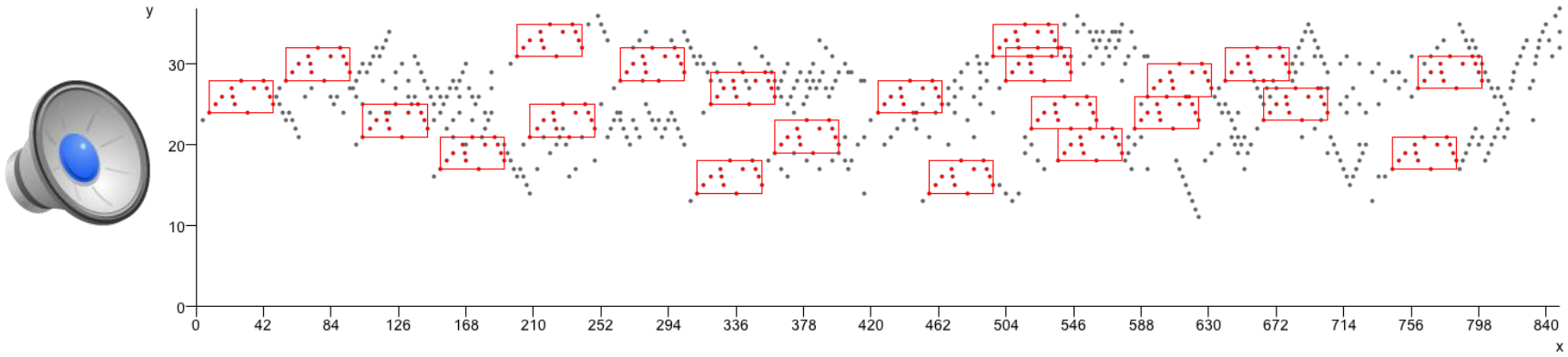


Other projections of a multidimensional representation of a musical object



- Top-left: morphetic pitch against onset time
- Top-right: Morphetic pitch against voice
- Bottom-left: voice against onset time
- Bottom-right: morphetic pitch against chromatic pitch

Repeated patterns in music

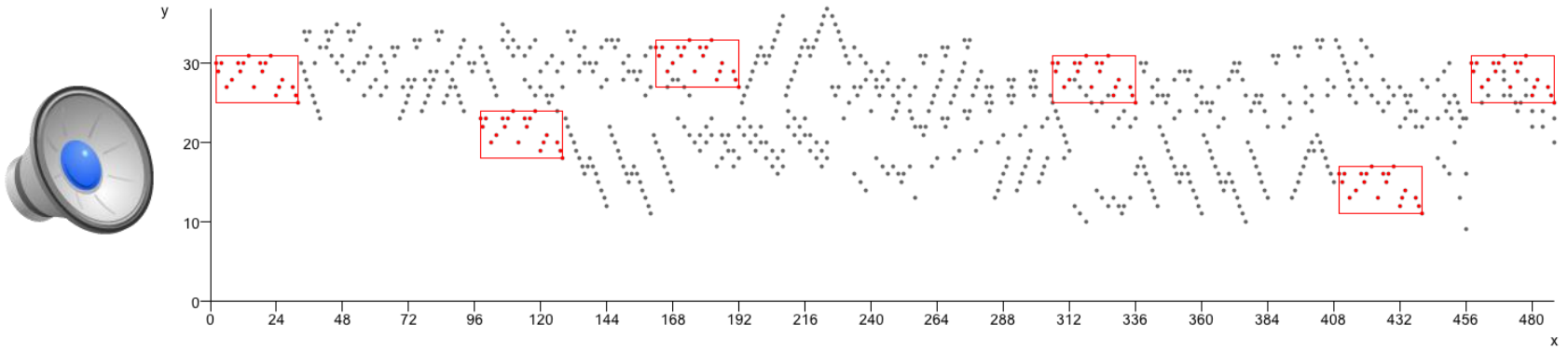


J.S. Bach, Fugue in C major, BWV 846

"the importance of parallelism [i.e., repetition] in musical structure cannot be overestimated. The more parallelism one can detect, the more internally coherent an analysis becomes, and the less independent information must be processed and retained in hearing or remembering a piece"

Lerdahl and Jackendoff (1983, p.52)

Repeated patterns in music

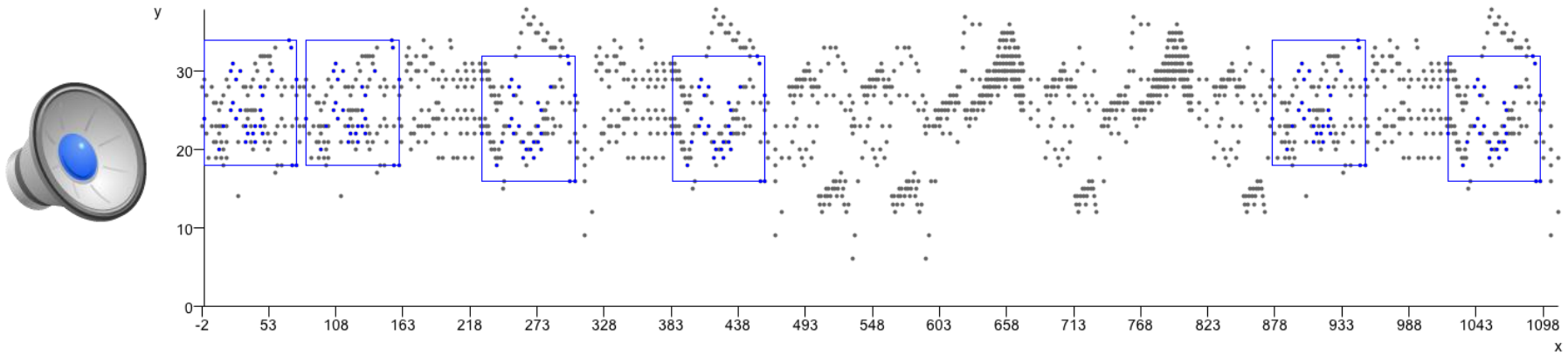


J.S. Bach, Fugue in C minor, BWV 847

repetition "is the basis of music as an art"

Schenker (1954, p.5)

Repeated patterns in music



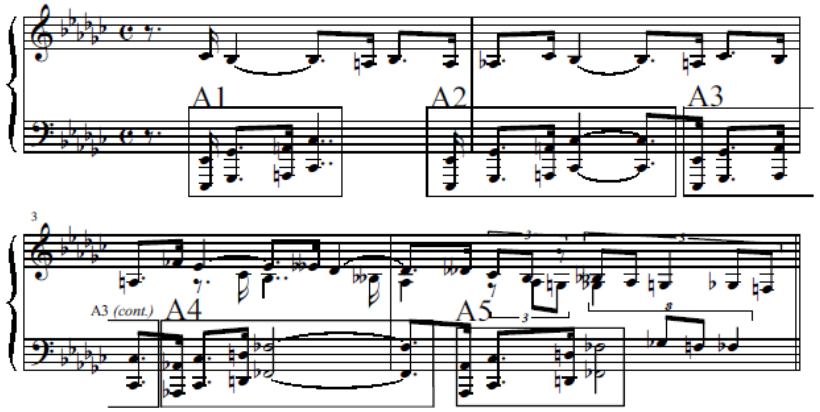
L. van Beethoven, Piano Sonata, Op.2, No.1, 3rd movement

"the central act" in all forms of music analysis is "the test for identity"

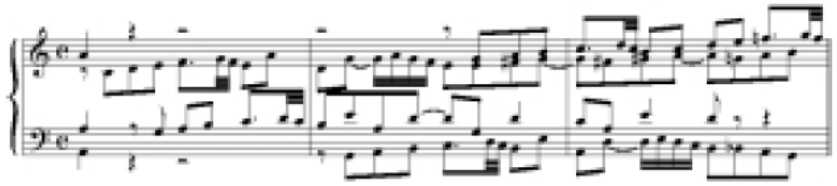
Bent and Drabkin (1987, p.5)

The diversity of musical repetition

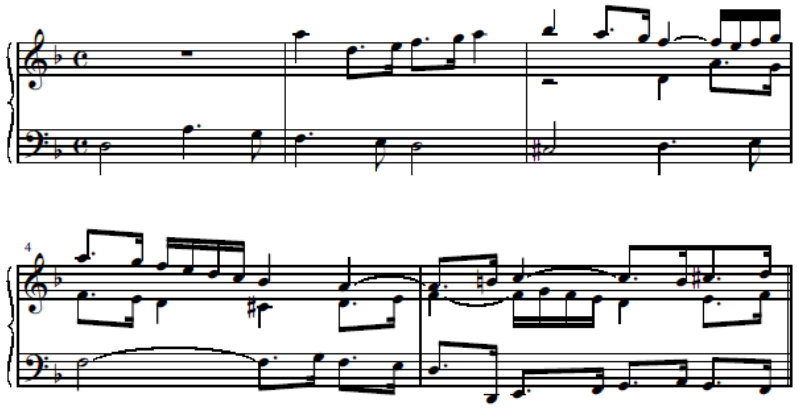
Samuel Barber, Sonata for Piano, Op. 26 (bars 1-4)

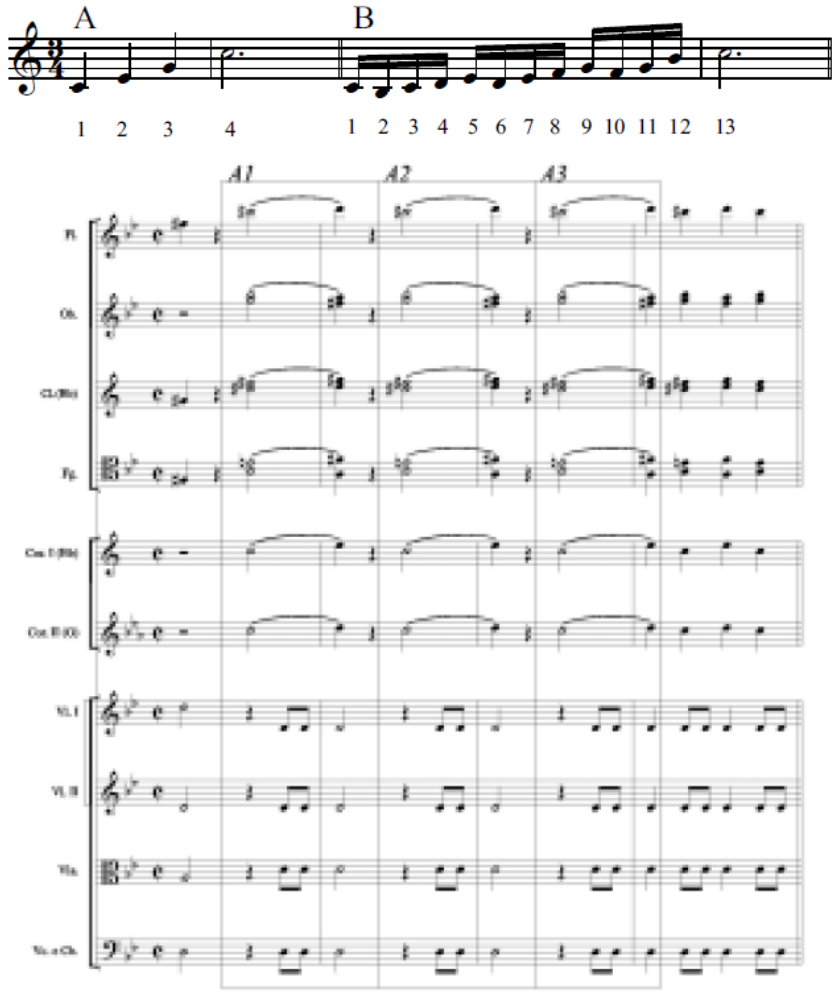


J. S. Bach, Fugue in C major, BWV846 (bars 14-16)



J. S. Bach, Contrapunctus VI from *Die Kunst der Fuge*, BWV1080 (bars 1-5)



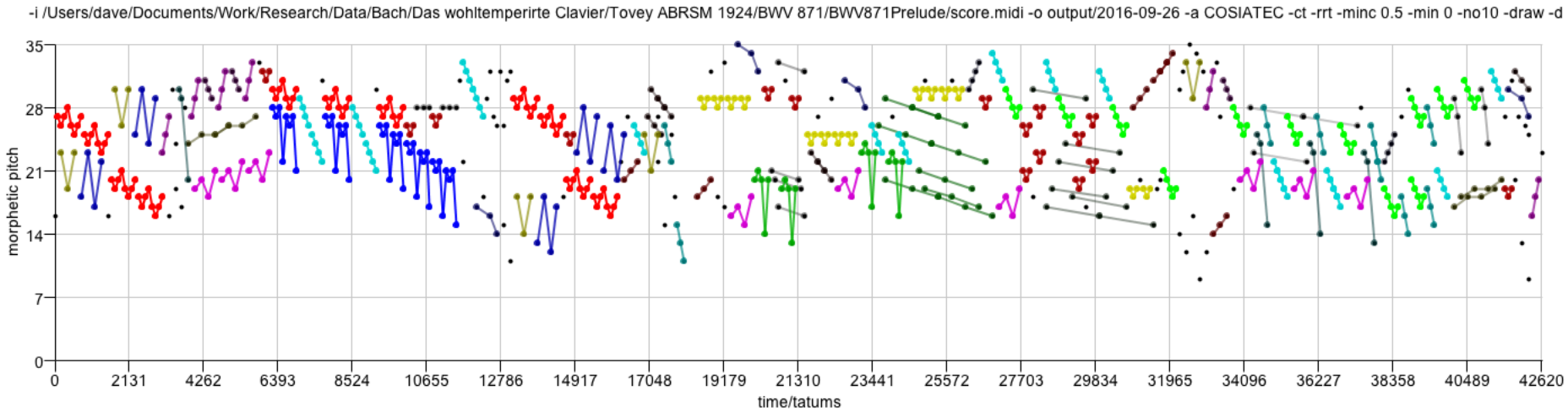


Mozart, Symphony No. 40 in G minor, K.550, bars 16-19

Interesting musical repetitions are structurally diverse

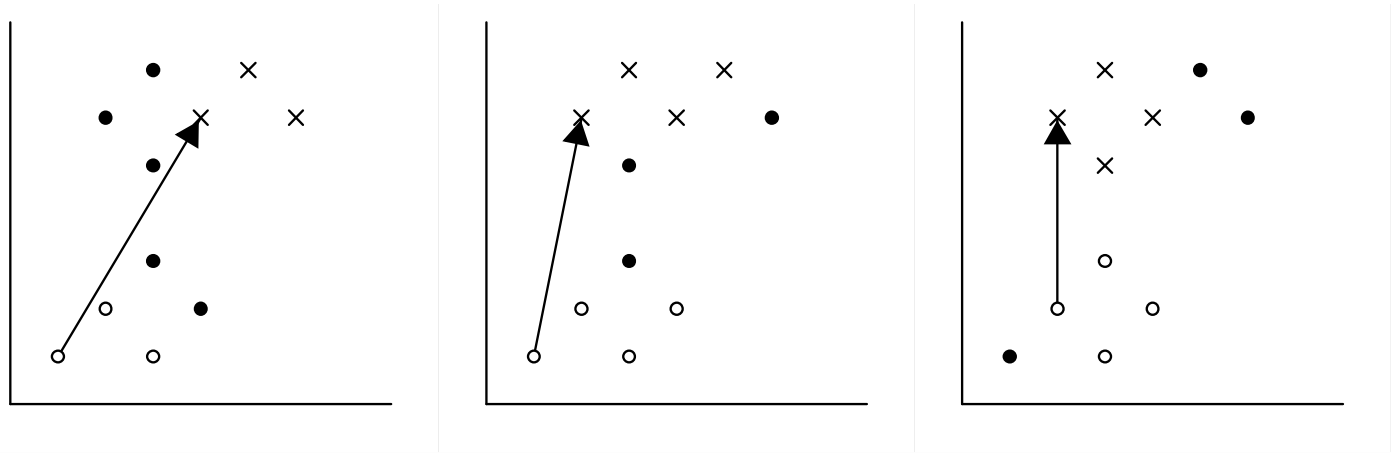
- Want to discover all *and only* *interesting* repeated patterns
  - The ones that give us knowledge that we can use to more effectively carry out musical tasks
- Class of interesting repeated patterns is structurally diverse because
    - patterns vary widely in structural characteristics
    - many ways of transforming a musical pattern to give another pattern that is perceived to be a version of it
      - e.g., truncated, augmented, diminished, inverted, embellished and even reversed

Geometric, compression-based pattern discovery in music



J. S. Bach, Prelude No. 2 in C minor, BWV 871
from Book 2 of The Well-Tempered Clavier
Performed by Angela Hewitt

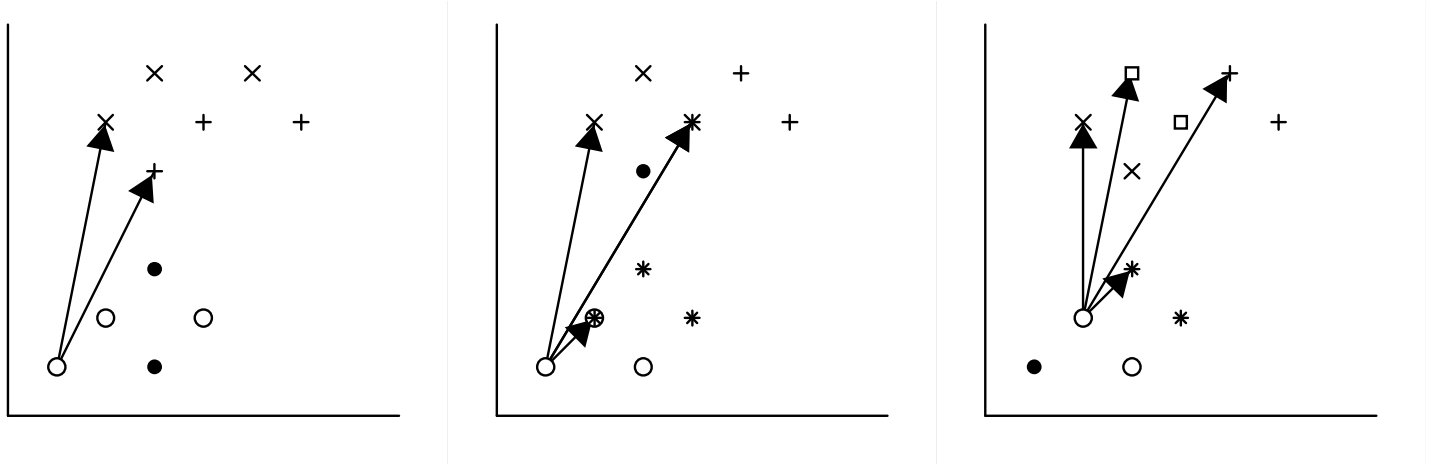
Maximal translatable patterns (MTPs)



$$P_1 \equiv_{\mathrm{T}} P_2 \iff (\exists v \mid P_2 = P_1 + v) \tag{1}$$

$$\mathrm{MTP}(v, D) = \{p \mid p \in D \wedge p + v \in D\} \tag{2}$$

Translational equivalence classes (TECs)

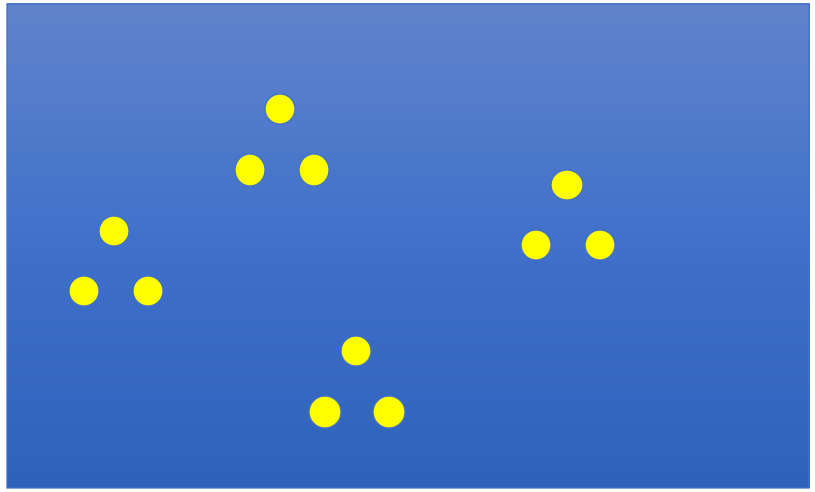


$$\mathrm{TEC}(P, D) = \{Q \mid Q \equiv_{\mathrm{T}} P \wedge Q \subseteq D\} \quad (3)$$

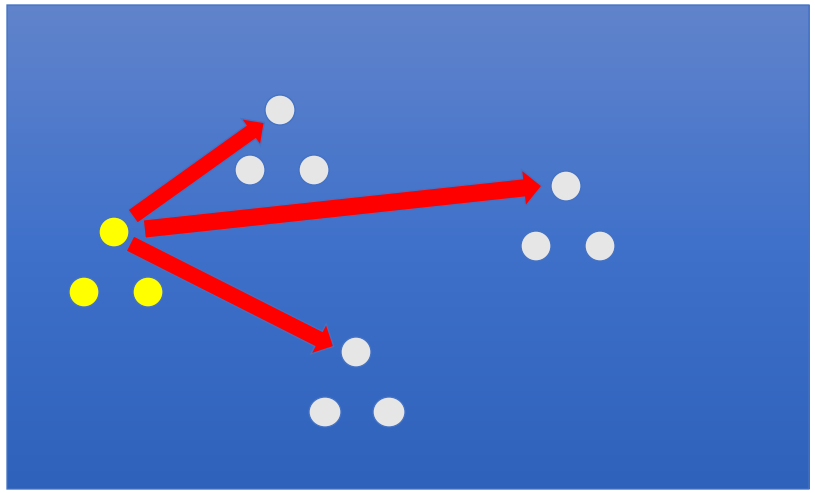
$$\mathrm{COV}(T) = \bigcup_{P \in T} P \,. \quad (4)$$

- Translational equivalence class (TEC) of a pattern *P* is the set of all patterns in the dataset that are translationally equivalent to *P*
- A TEC can be compactly expressed as a <pattern,vector set> pair
- Covered set of a TEC is the set of points that are members of patterns in the TEC

Compressed representation of a TEC



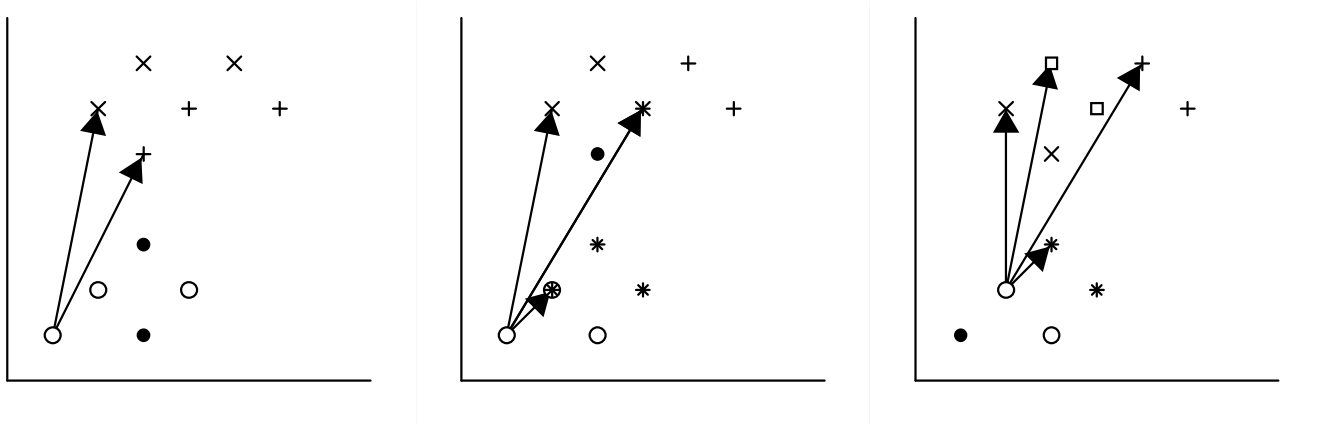
12 points = 24 numbers



3 points + 3 vectors = 12 numbers

- A translational equivalence class is a set of translationally equivalent patterns
- Could encode such a set by simply listing the points in each occurrence
- Can be encoded more parsimoniously by listing the points in *one* occurrence of the pattern along with the vectors that map that occurrence onto the other occurrences
  - i.e., a <pattern, vector set> pair

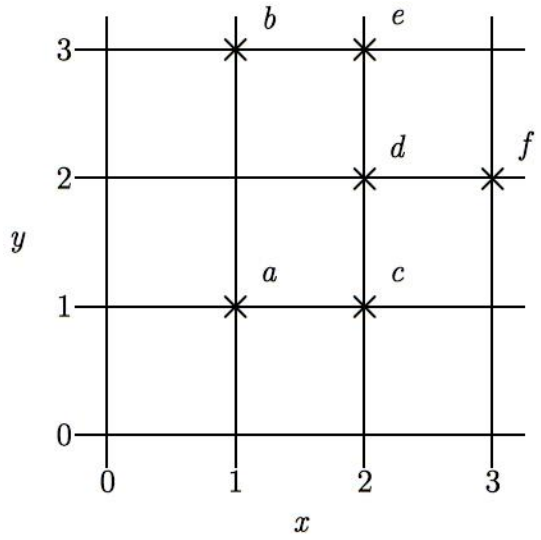
Compression with TECs

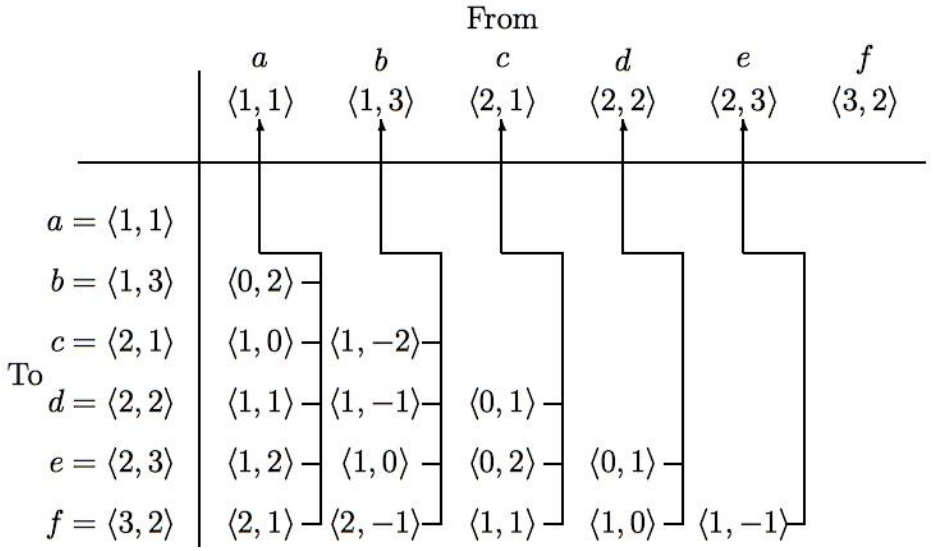


- *In extenso* specification of a TEC uses *kmn* integers where there are *n* repetitions of a *k*-dimensional pattern of size *m*
- $\langle \text{pattern}, \text{vector set} \rangle$ representation uses $k(m + n - 1)$ integers—gives compressed encoding

$$\text{CR}(T) = \frac{|\text{COV}(T)|}{|P| + |V|} \tag{5}$$

SIA
Discovering all maximal translatable patterns (MTPs)





Vector		Datapoint
⟨0, 1⟩	→	⟨2, 1⟩
⟨0, 1⟩	→	⟨2, 2⟩
⟨0, 2⟩	→	⟨1, 1⟩
⟨0, 2⟩	→	⟨2, 1⟩
⟨1, −2⟩	→	⟨1, 3⟩
⟨1, −1⟩	→	⟨1, 3⟩
⟨1, −1⟩	→	⟨2, 3⟩
⟨1, 0⟩	→	⟨1, 1⟩
⟨1, 0⟩	→	⟨1, 3⟩
⟨1, 0⟩	→	⟨2, 2⟩
⟨1, 1⟩	→	⟨1, 1⟩
⟨1, 1⟩	→	⟨2, 1⟩
⟨1, 2⟩	→	⟨1, 1⟩
⟨2, −1⟩	→	⟨1, 3⟩
⟨2, 1⟩	→	⟨1, 1⟩

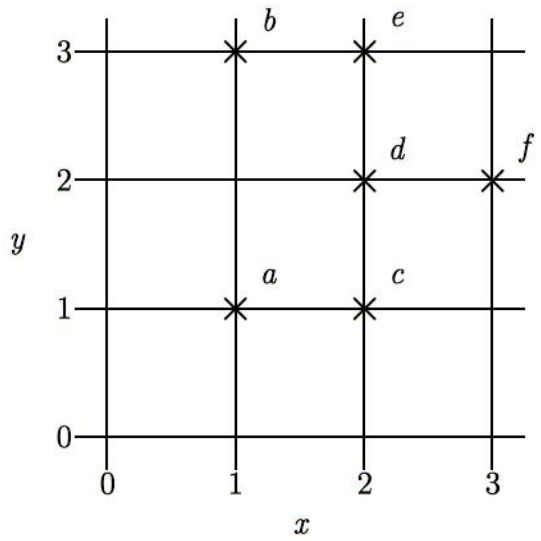
Pattern is *translatable* by vector *v* in dataset if it can be translated by *v* to give another pattern in the dataset

MTP for a vector *v* contains all points mapped by *v* onto other points in the dataset

$O(kn^2 \log n)$ time, $O(kn^2)$ space

$O(kn^2)$ time if use direct address table to store vectors

SIATEC
Discovering all occurrences of all MTPs



		From $a=\langle 1,1\rangle$	$b=\langle 1,3\rangle$	$c=\langle 2,1\rangle$	$d=\langle 2,2\rangle$	$e=\langle 2,3\rangle$	$f=\langle 3,2\rangle$
To	$a=\langle 1,1\rangle$	$\langle 0,0\rangle$	$\langle 0,-2\rangle$	$\langle -1,0\rangle$	$\langle -1,-1\rangle$	$\langle -1,-2\rangle$	$\langle -2,-1\rangle$
	$b=\langle 1,3\rangle$	$\langle 0,2\rangle$	$\langle 0,0\rangle$	$\langle -1,2\rangle$	$\langle -1,1\rangle$	$\langle -1,0\rangle$	$\langle -2,1\rangle$
	$c=\langle 2,1\rangle$	$\langle 1,0\rangle$	$\langle 1,-2\rangle$	$\langle 0,0\rangle$	$\langle 0,-1\rangle$	$\langle 0,-2\rangle$	$\langle -1,-1\rangle$
	$d=\langle 2,2\rangle$	$\langle 1,1\rangle$	$\langle 1,-1\rangle$	$\langle 0,1\rangle$	$\langle 0,0\rangle$	$\langle 0,-1\rangle$	$\langle -1,0\rangle$
	$e=\langle 2,3\rangle$	$\langle 1,2\rangle$	$\langle 1,0\rangle$	$\langle 0,2\rangle$	$\langle 0,1\rangle$	$\langle 0,0\rangle$	$\langle -1,1\rangle$
	$f=\langle 3,2\rangle$	$\langle 2,1\rangle$	$\langle 2,-1\rangle$	$\langle 1,1\rangle$	$\langle 1,0\rangle$	$\langle 1,-1\rangle$	$\langle 0,0\rangle$

Time to find all occurrences of pattern of size $m = O(kmn)$.

Translational Equivalence Class (TEC) is set of all translationally invariant occurrences of a pattern

$$\sum_{i=1}^{l} m_i \leq \frac{n(n-1)}{2}$$

$$O\left(\sum_{i=1}^{l} km_i n\right) \leq O\left(k\frac{n^2(n-1)}{2}\right)$$

Overall worst-case running time of SIATEC $= O(kn^3)$

Need for heuristics to isolate interesting MTPs and TECs

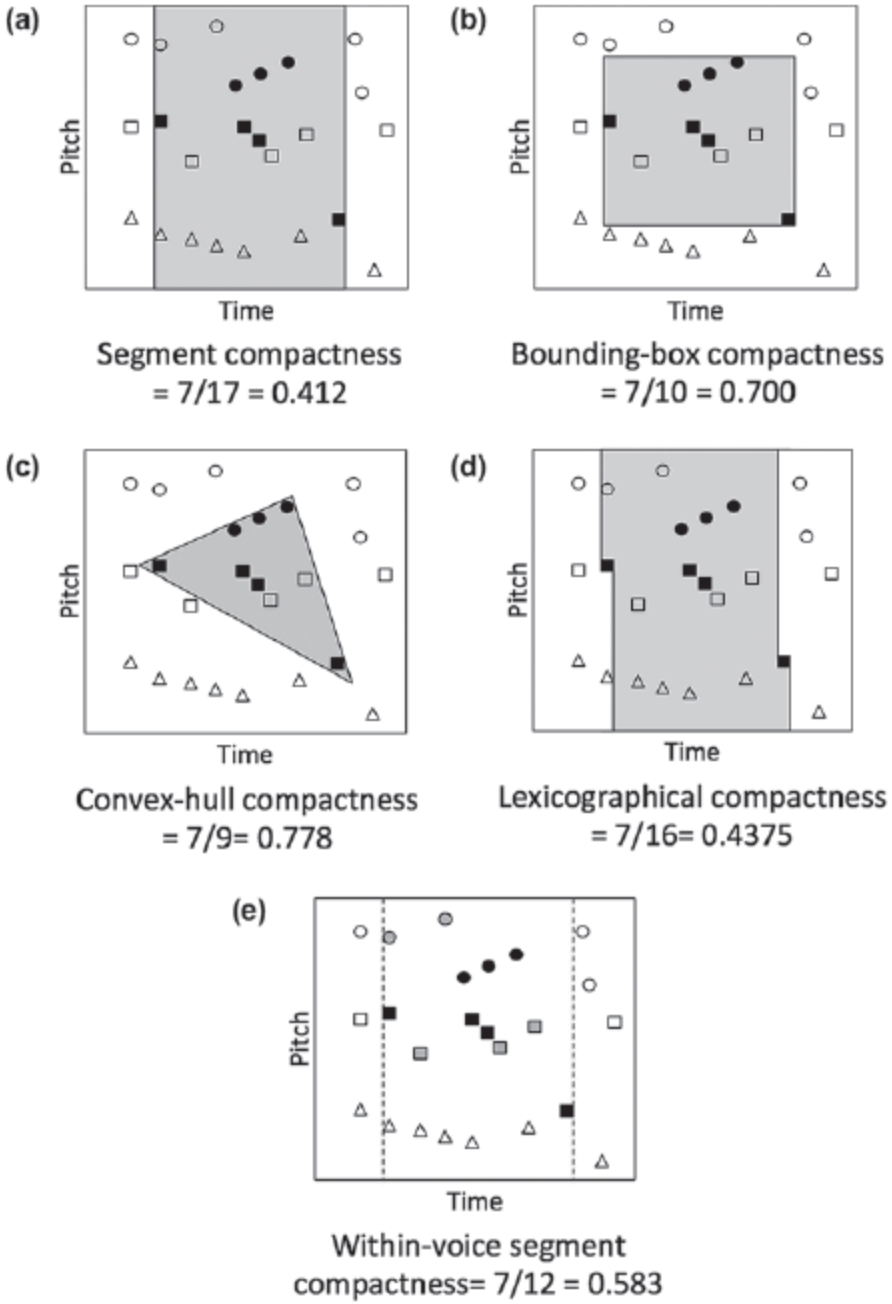
- 2^n patterns in a dataset of size n
- SIA generates < $n^2/2$ patterns
  - sum of cardinalities of MTPs is n(n-1)/2
  - => SIA generates small fraction of all patterns in a dataset
- Many interesting patterns derivable from MTPs found by SIA
  - e.g., bounding box or segment spanned by the patterns
- BUT many of the patterns found by SIA are NOT interesting
    - 17519 patterns found by SIA in the fugue from BWV 847
    - probably about 20 are interesting
- => Need heuristics for isolating interesting patterns in output of SIA and SIATEC

Heuristics for isolating the “best” patterns

$$\text{Compression ratio} = \frac{\text{Coverage}}{\text{Number of points in pattern + Freq. of occurrence of pattern -1}}$$

$$\text{Compactness} = \frac{\text{Number of points in pattern}}{\text{Number of points in region spanned by pattern}}$$

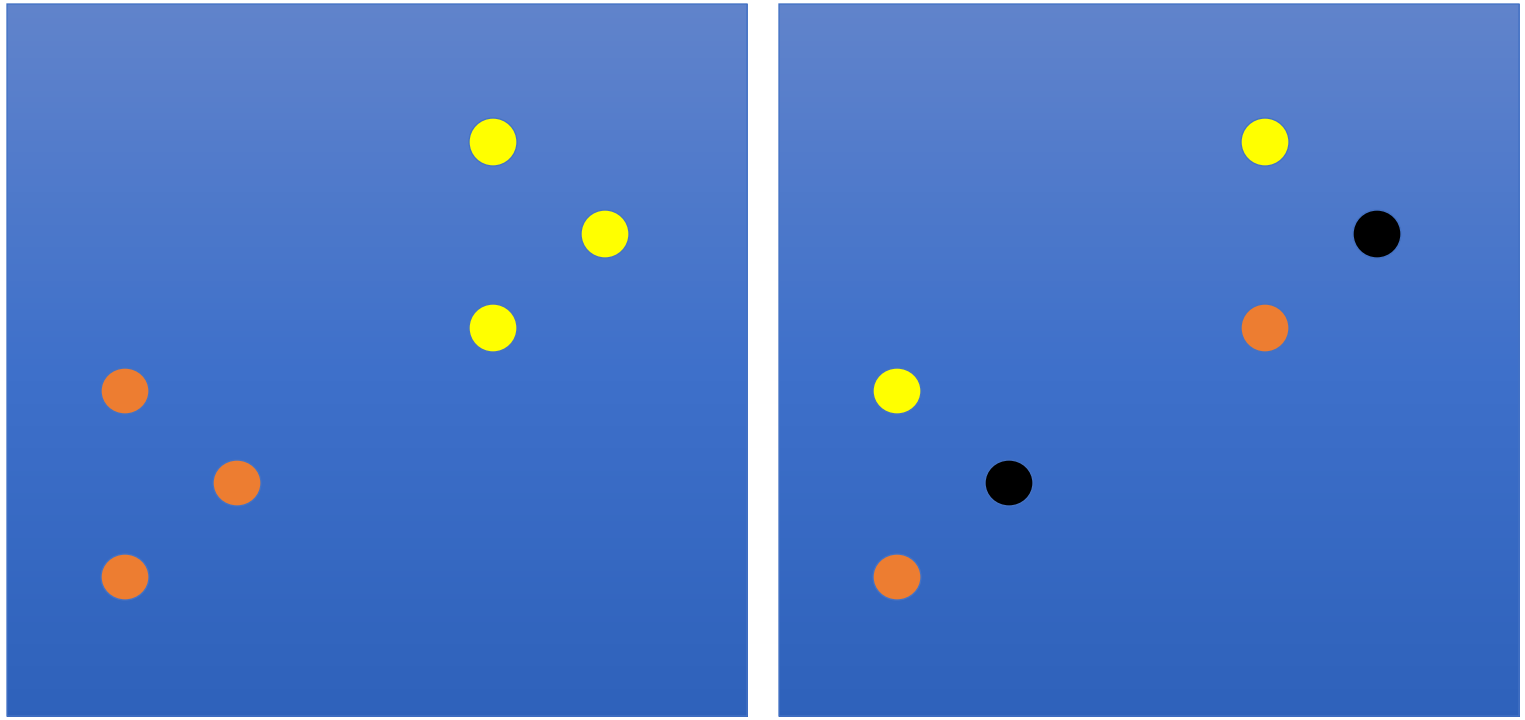
Coverage = Number of points covered by occurrences of the pattern



Comparing TECs

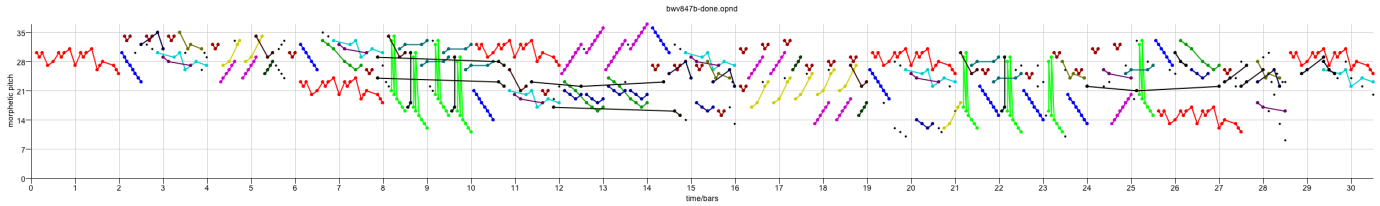
- Given two TECs, the ***better*** one is
  - the one with the ***higher compression ratio***
  - otherwise (i.e., they have the same compression ratio), the one which is ***more compact***
  - otherwise, the one that has the ***better coverage***
  - otherwise, the one that has the ***larger pattern***
  - otherwise, the one whose pattern has the ***shorter duration***
  - otherwise, the one with the ***smaller bounding box***

Dual or conjugate TECs



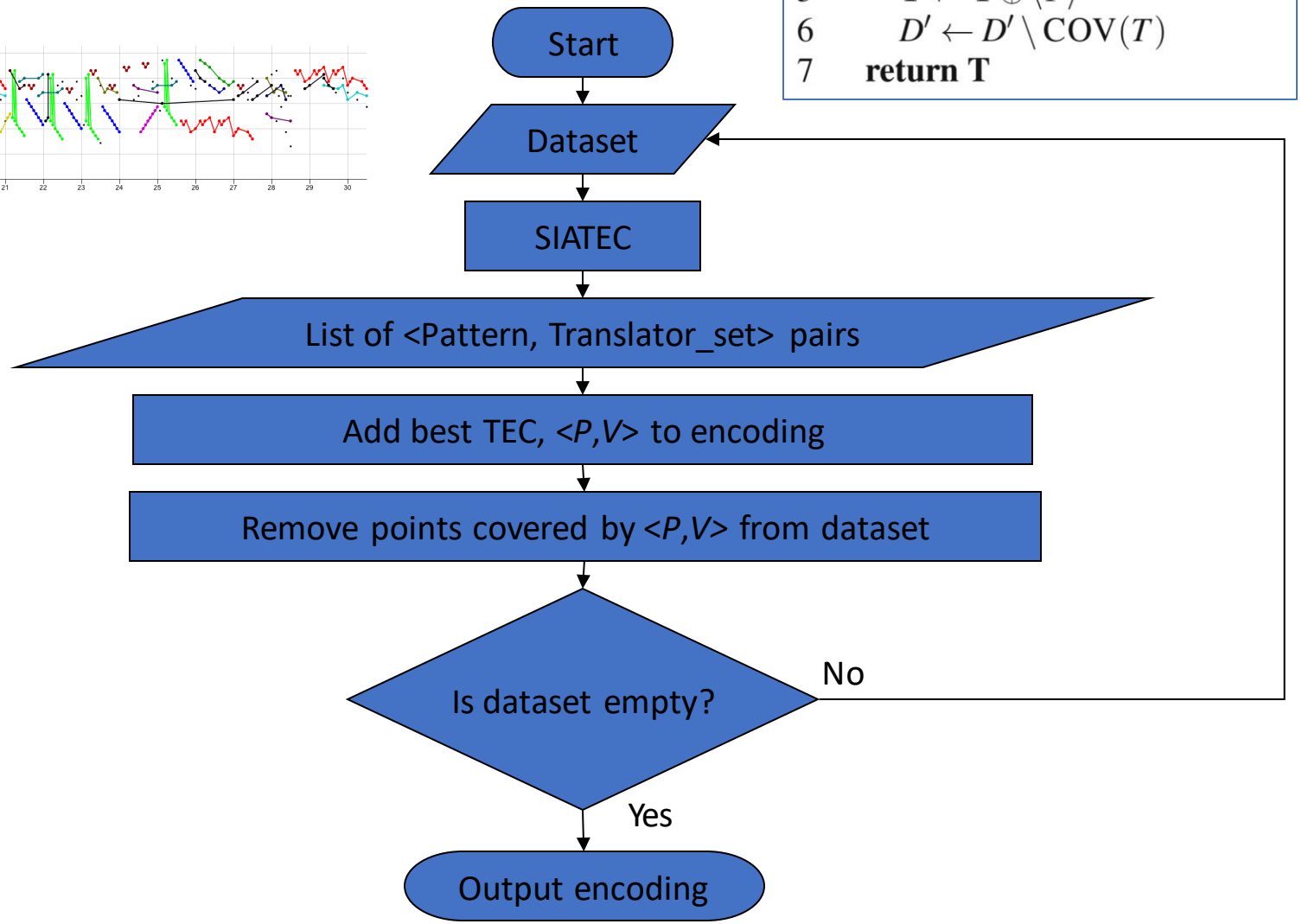
COSIATEC

Compression and covering with MTP TECs

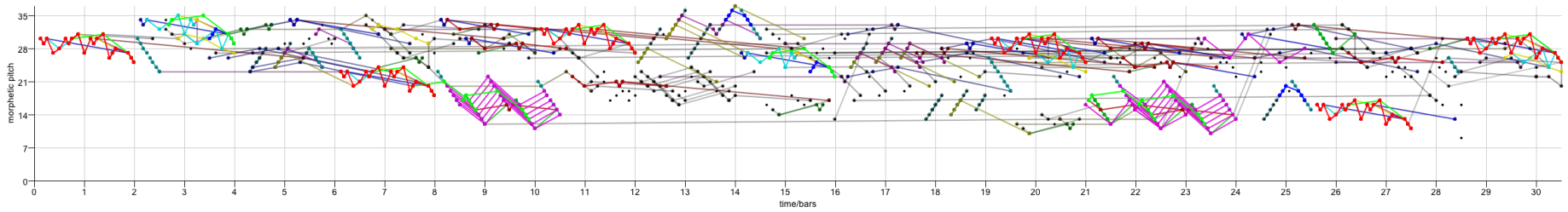


- Computes a set of TECs that collectively cover the input dataset
- The dataset is exclusively and exhaustively partitioned into TEC covered sets
- Typically gives better compression than the other strategies tried

```
COSIATEC(D)
1   D' ← COPY(D)
2   T ← ⟨⟩
3   while D' ≠ ∅
4       T ← GETBESTTEC(D', D)
5       T ← T ⊕ ⟨T⟩
6       D' ← D' \ COV(T)
7   return T
```



SIATECCompress
Compression and covering with MTP TECs




```
SIATECCompress(D)
1    T ← SIATEC(D)
2    T ← SortTECsByQuality(T)
3    D′ ← ∅
4    E ← ⟨⟩
5    for i ← 0 to |T| − 1
6        T ← T[i]
7        S ← COV(T)
         ▶ Recall that each TEC, T, is an ordered pair, ⟨P, Θ⟩
8        if |S \ D′| > |T[0]| + |T[1]| − 1
9            E ← E ⊕ ⟨T⟩
10           D′ ← D′ ∪ S
11           if |D′| = |D|
12               break
13   R ← D \ D′
14   if |R| > 0
15       E ← E ⊕ ⟨AsTEC(R)⟩
16   return E
```

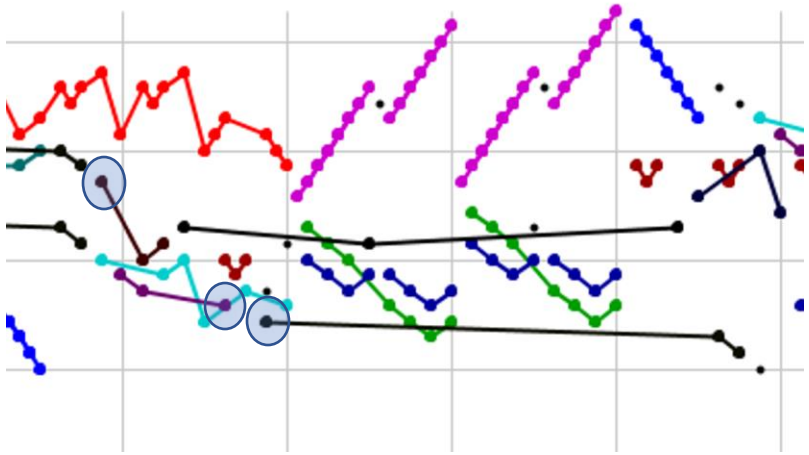
Adds a TEC to encoding if its <P,Θ> representation is shorter than the set of new points covered

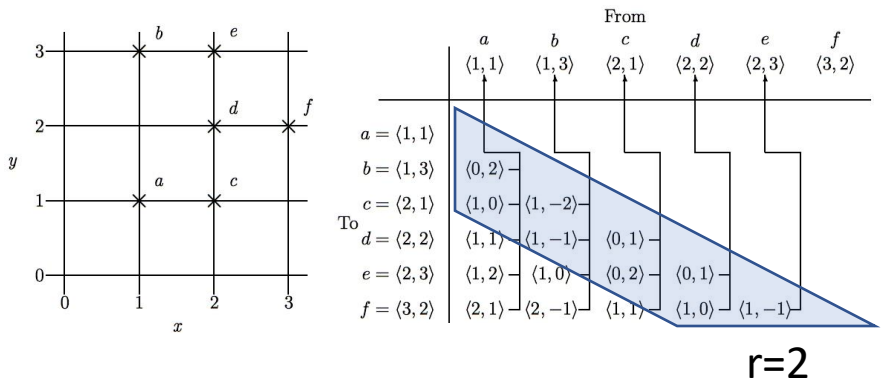
- Sorts TECs into decreasing order by "quality"
  - Quality is a function of compression factor, compactness and coverage
- Scans list, adding TECs to output encoding if they increase encoding length by less than the number of extra points that they cover
- TEC covered sets in generated cover may intersect
- Typically lower compression factor than COSIATEC

Forth's algorithm (Forth 2012, Forth and Wiggins 2009)

- Runs SIATEC once to find all MTP TECs
- Computes a union of TEC covered sets that aims to maximises a "musicological weight" that is a function of compression ratio and compactness
- Does not strictly partition the dataset
- May not completely cover the dataset

SIACT and SIAR (Collins et al. 2010, Collins 2011)





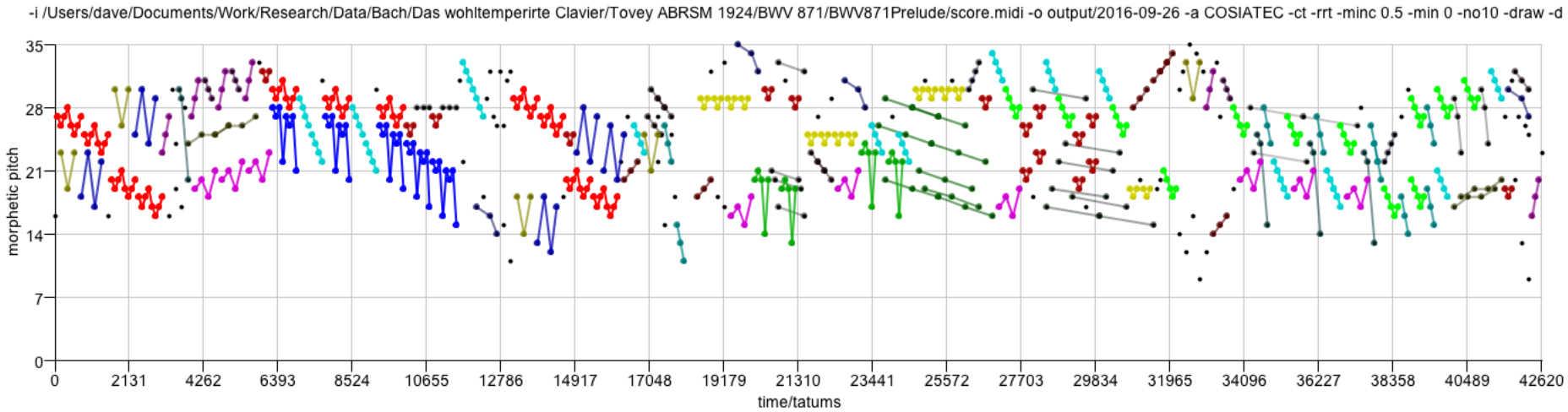
SIACT

- Aims to solve the problem of isolated membership
- "Trawls" inside each MTP for compact subsets
- Can be used to replace SIA in SIATEC, COSIATEC and Forth's algorithm

SIAR

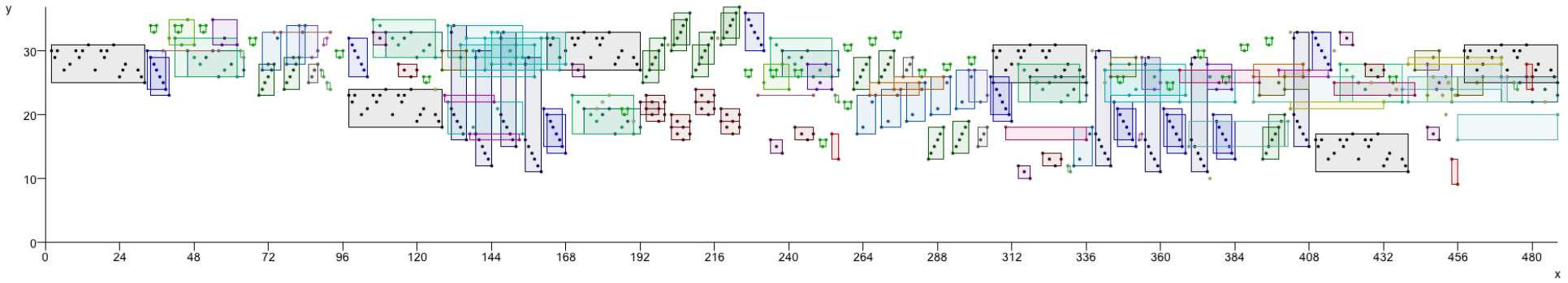
- Computes only a given number, r, of the subdiagonals in the SIA vector table, rather than the whole region below the leading diagonal
- Similar to running SIA with a moving window

Geometric, compression-based pattern discovery in music



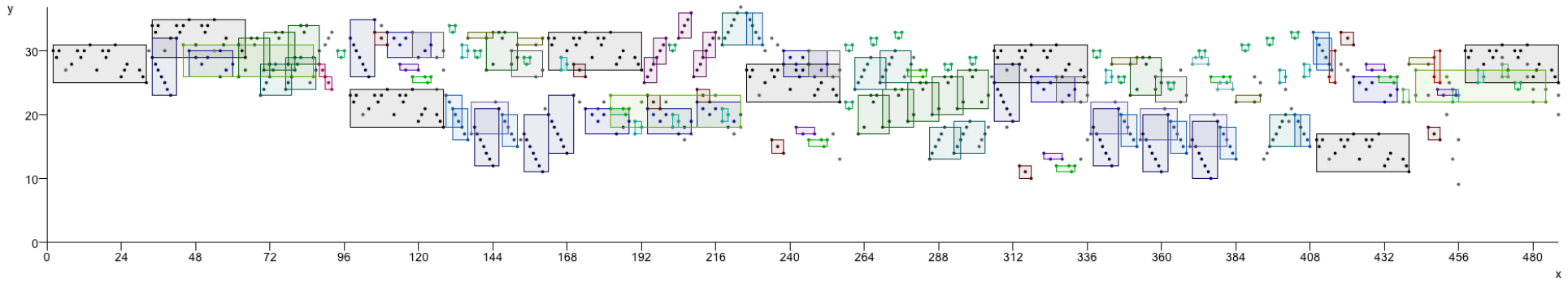
J. S. Bach, Prelude No. 2 in C minor, BWV 871
from Book 2 of The Well-Tempered Clavier
Performed by Angela Hewitt

COSIATEC
J.S. Bach, Fugue in C minor, BWV 847



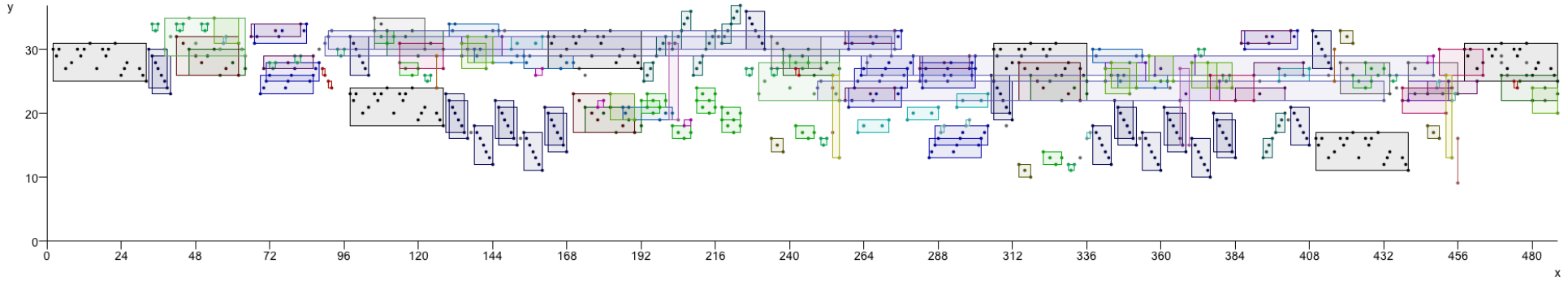
- Number of TECs: 26
- Encoding length: 268
- Number of notes: 751
- Encoding length without residual point set: 248
- Number and proportion of residual points: 20, 2.66%
- Compression ratio: 2.80
- Compression ratio excluding residual point set: 2.95

COSIACTTEC
J.S. Bach, Fugue in C minor, BWV 847



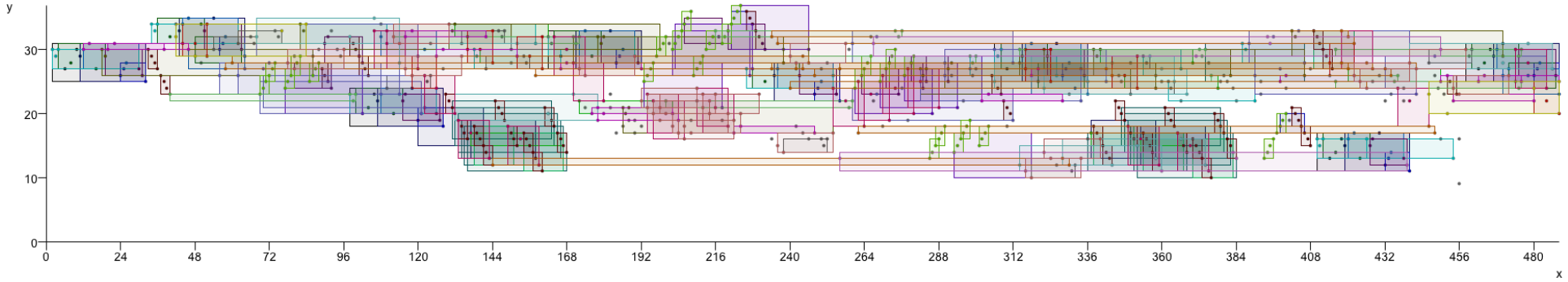
- Number of TECs: 21
- Encoding length: 277
- Number of notes: 751
- Encoding length without residual point set: 210
- Number and proportion of residual points: 67, 8.92%
- Compression ratio: 2.71
- Compression ratio excluding residual point set: 3.26

COSIARTEC
J.S. Bach, Fugue in C minor, BWV 847



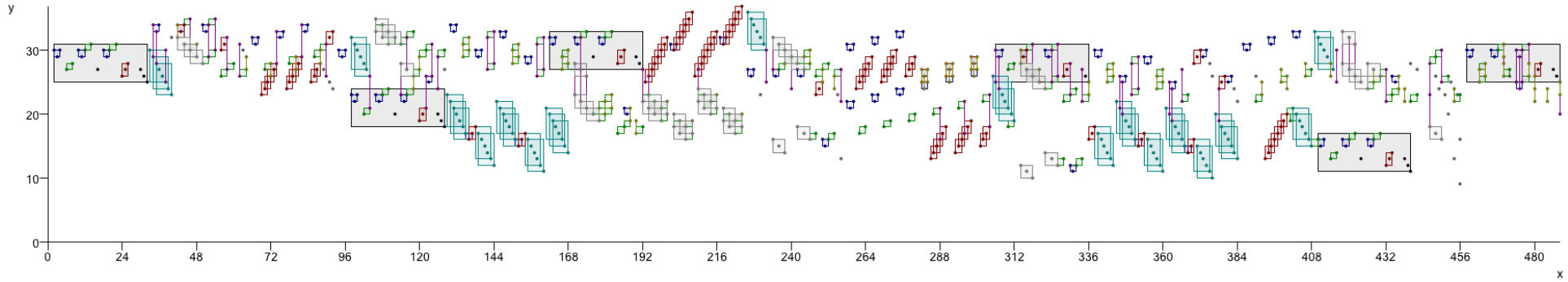
- Number of TECs: 26
- Encoding length: 269
- Number of notes: 751
- Encoding length without residual point set: 248
- Number and proportion of residual points: 21, 2.80%
- Compression ratio: 2.79
- Compression ratio excluding residual point set: 2.94

SIATECCompress
J.S. Bach, Fugue in C minor, BWV 847



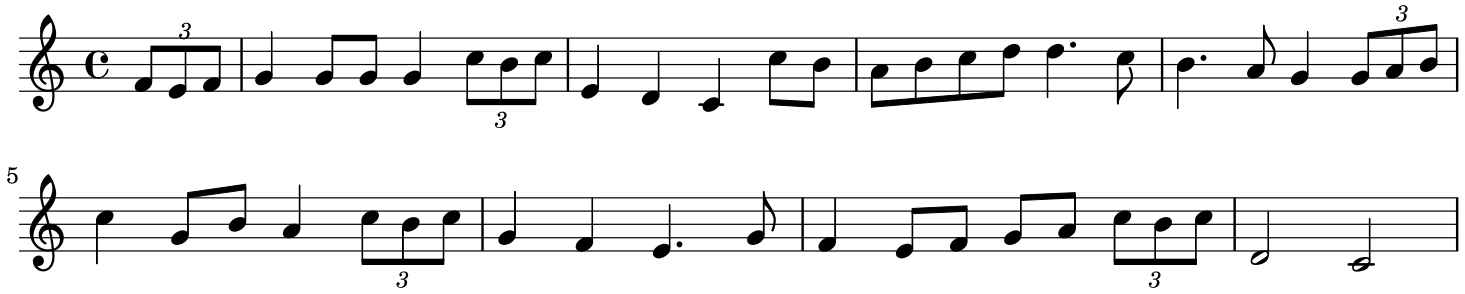
- Number of TECs: 26
- Encoding length: 496
- Number of notes: 751
- Compression ratio: 1.51

Forth's algorithm
J.S. Bach, Fugue in C minor, BWV 847



- Number of TECs: 8
- Compression ratio: 1.4
- Encoding length: 534
- Number of points in dataset: 751
- Total number of points covered: 725
- Total proportion of points covered: 0.97

Using point-set compression algorithms for folk-song classification



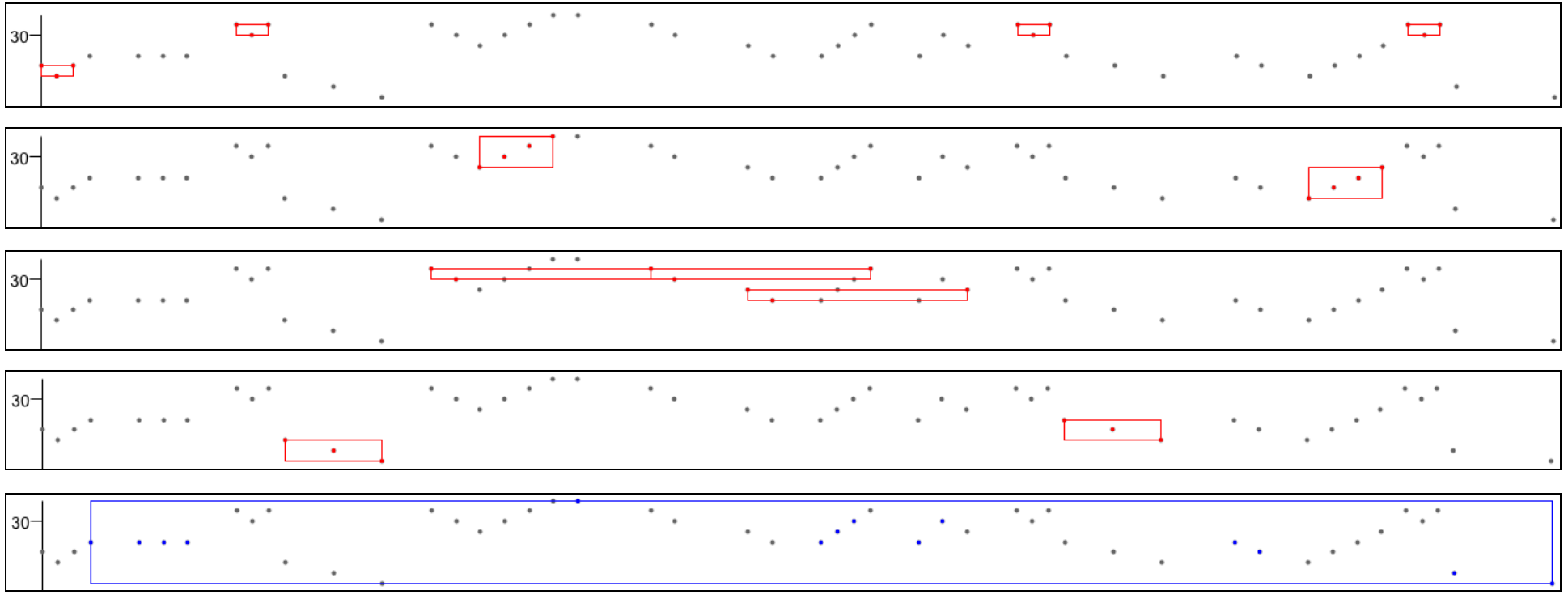
$$\mathrm{NCD}(x, y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}$$

Table 2. Results on Task 1. *SR* is the classification success rate, CR_{AC} is the average compression ratio over the melodies in the *Annotated Corpus*. CR_{pairs} is the average compression ratio over the concatenated pairs of files used to obtain the NCD values.

Algorithm	*SR*	CR_{AC}	CR_{pairs}
COSIATEC	0.8389	1.5791	1.6670
COSIARTEC	0.8361	1.5726	1.6569
COSIARCTTEC	0.7917	1.4547	1.5135
COSIACTTEC	0.7694	1.4556	1.5138
ForthCT	0.6417	1.1861	1.2428
ForthRCT	0.6417	1.1861	1.2428
Forth	0.6111	1.2643	1.2663
ForthR	0.6028	1.2555	1.2655
SIARCTTECCompress	0.5750	1.3213	1.3389
SIATECCompress	0.5694	1.3360	1.3256
SIACTTECCompress	0.5250	1.3197	1.3381
SIARTECCompress	0.5222	1.3283	1.3216
bzip2	0.1250	2.7678	3.5061

Meredith, D. (2015). Music analysis and point-set compression. *Journal of New Music Research*, 44(3), pp. 245-270.

Example COSIATEC encoding of NLB folk song



(NLB015569_01, “Daar zou er een maagdje vroeg opstaan 2”, from the Nederlandse Liederen Bank, http://www.liederenbank.nl. Courtesy of Peter van Kranenburg.)

Example COSIATEC encoding

(NLB015569_01, “Daar zou er een maagdje vroeg opstaan 2”, from the Nederlandse Liederen Bank, http://www.liederenbank.nl. Courtesy of Peter van Kranenburg.)

Input:

T(P(p(0,27),p(40,26),p(80,27),p(120,28),p(240,28),p(300,28),p(360,28),p(480,31),p(520,30),p(560,31),p(600,26),p(719,25),p(839,24),p(960,31),p(1020,30),p(1080,29),p(1140,30),p(1200,31),p(1260,32),p(1320,32),p(1500,31),p(1560,30),p(1740,29),p(1800,28),p(1920,28),p(1960,29),p(2000,30),p(2040,31),p(2159,28),p(2219,30),p(2280,29),p(2400,31),p(2440,30),p(2480,31),p(2520,28),p(2639,27),p(2759,26),p(2939,28),p(3000,27),p(3120,26),p(3180,27),p(3240,28),p(3300,29),p(3360,31),p(3400,30),p(3440,31),p(3480,25),p(3719,24)),V(v(0,0)))

(48 points)

Encoding:

T(P(p(0,27),p(40,26),p(80,27)),V(v(480,4),v(2400,4),v(3360,4)))
T(P(p(1080,29),p(1140,30),p(1200,31),p(1260,32)),V(v(2040,-3)))
T(P(p(960,31),p(1020,30),p(1500,31)),V(v(540,0),v(780,-2)))
T(P(p(600,26),p(719,25),p(839,24)),V(v(1920,2)))
T(P(p(120,28),p(240,28),p(300,28),p(360,28),p(1320,32),p(1920,28),p(1960,29),p(2000,30),p(2159,28),p(2219,30),p(2939,28),p(3000,27),p(3480,25),p(3719,24)),V())

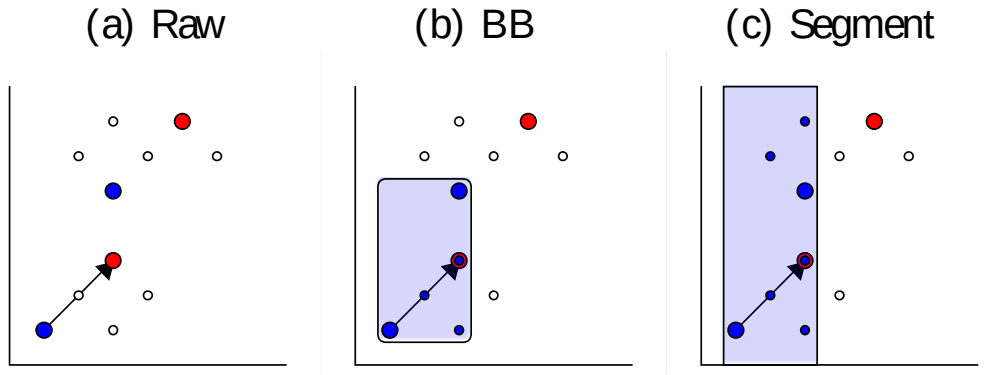
(34 points or vectors)

Compression ratio: 1.41

MIREX 2013:
Discovery of repeated themes and sections

- First MIREX competition on pattern discovery
  - http://www.music-ir.org/mirex/wiki/2013:Discovery_of_Repeated_Themes_%26_Sections
- JKU Pattern Development database
  - https://dl.dropbox.com/u/11997856/JKU/JKUPDD-noAudio-Jul2013.zip
  - Five pieces, each with a set of “ground-truth” patterns (occurrence sets)
    - Patterns come from analyses by well-known analysts (e.g., Schoenberg, Bruhn)

Discovering repeated themes and sections in the JKU PDD

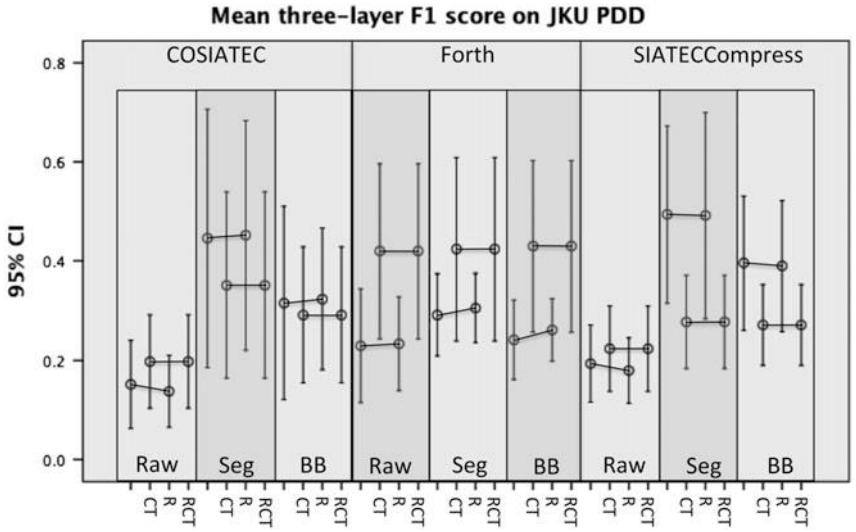


- Five pieces in database – **polyphonic version used!**
- Three basic algorithms, COSIATEC, SIATECCompress and Forth's algorithm
  - with and without CT
  - SIA or SIAR
  - Raw, BB or Segment mode
- Gives 36 algorithms formed by combining values from these 4 variables

Johannes Kepler University Patterns Development Database (JKU-PDD)

- Gibbons, “Silver Swan” (madrigal) (1612)
- J. S. Bach, Fugue in A minor (BWV 889) from 2nd book of the WTC (1742)
- Mozart, Piano Sonata in E flat major (K. 282), 2nd movement (1774)
- Beethoven, Piano Sonata in F minor, Op.2, No.1 (1795)
- Chopin, Mazurka in B flat minor, Op.24, No.4 (1836)

Point-set compression algorithms on JKU-PDD



- Replacing SIA with SIAR had no significant result on precision, recall or F1
- CT had a positive effect with Forth's algorithm but generally a negative effect on COSIATEC or SIATECCompress (except in Raw mode)
- Top-performing algorithms used Segment mode

Algorithm	Chopin	Gibbons	Beethoven	Mozart	Bach	Max	Mean
SIATECCompressSegment	0.56	**0.40**	0.63	0.59	0.29	0.63	**0.494**
SIARTECCompressSegment	**0.60**	0.39	0.65	0.57	0.25	0.65	0.492
COSIARTECSegment	0.44	0.39	0.69	0.55	0.19	0.69	0.452
COSIATECSegment	0.37	0.37	**0.71**	**0.60**	0.18	**0.71**	0.446
ForthCTSegment	0.29	0.35	0.58	0.59	0.31	0.59	0.424
ForthRCTSegment	0.29	0.35	0.58	0.59	0.31	0.59	0.424
ForthCT	0.23	0.35	0.56	0.56	**0.40**	0.56	0.420
ForthRCT	0.23	0.35	0.56	0.56	**0.40**	0.56	0.420
COSIACTTECSegment	0.25	0.31	0.56	0.45	0.19	0.56	0.352
COSIARCTTECSegment	0.25	0.31	0.56	0.45	0.19	0.56	0.352
Forth	0.12	0.33	0.32	0.21	0.17	0.33	0.230
COSIACTTEC	0.09	0.16	0.22	0.29	0.23	0.29	0.198
COSIARCTTEC	0.09	0.16	0.22	0.29	0.23	0.29	0.198
SIATECCompress	0.11	0.16	0.19	0.25	0.26	0.26	0.194
COSIATEC	0.15	0.11	0.18	0.23	0.19	0.23	0.152

Meredith, D. (2015). Music analysis and point-set compression. *Journal of New Music Research*, 44(3), pp. 245-270.

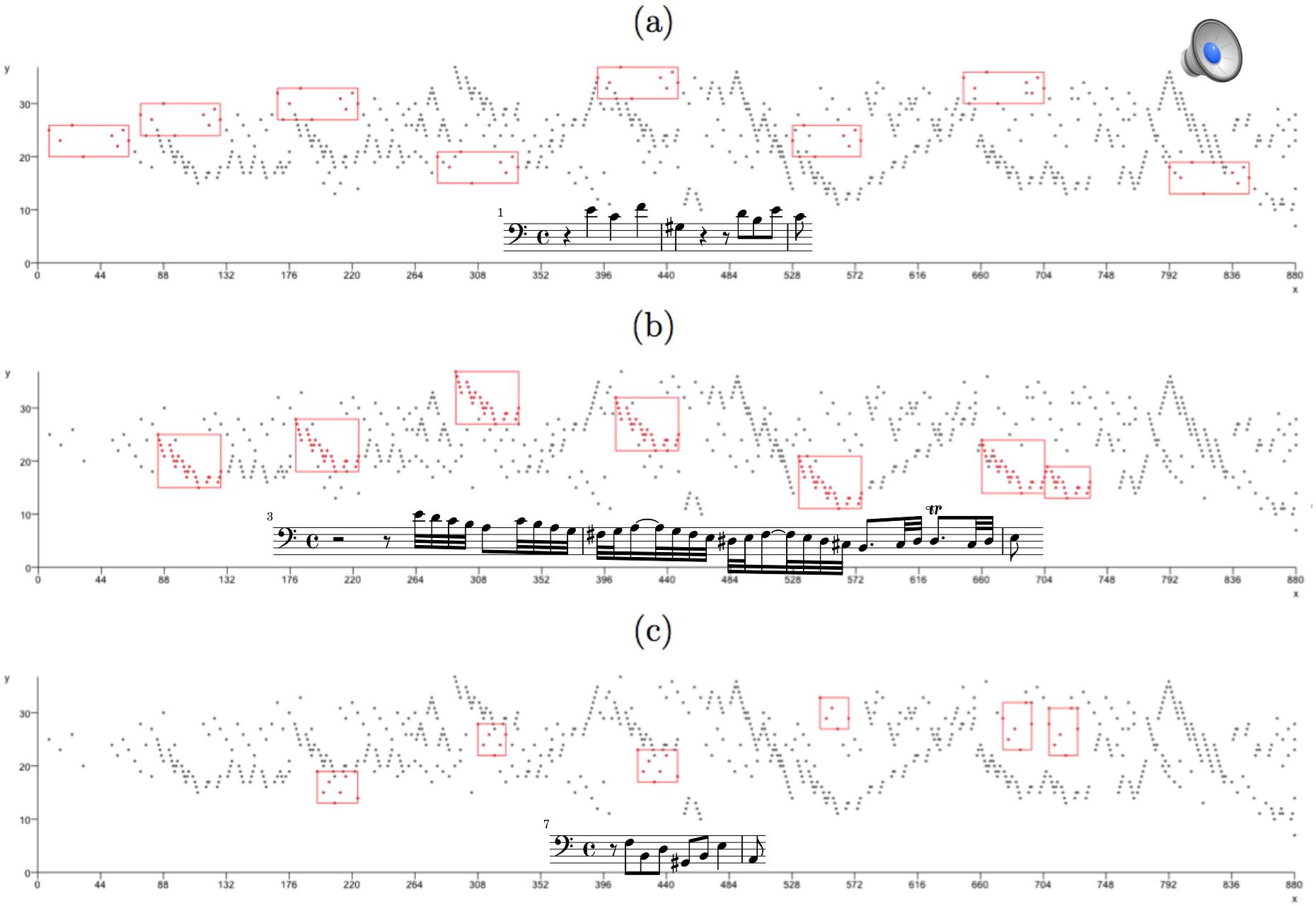


Figure 30: The ground-truth patterns in the JKU PDD for Bach's Fugue in A minor (BWV 889).

Problems with evaluating pattern discovery algorithms by comparison with human analyses

- To what question is the JKU PDD supposed to be providing the correct answer?
  - “What are the noticeable and/or important patterns in these five pieces?”
- Do the analyses in the JKU PDD actually identify all and only the noticeable and/or important patterns in the pieces in the database?
  - No!
    - some important patterns are missing
    - some important occurrences of recorded patterns are missing

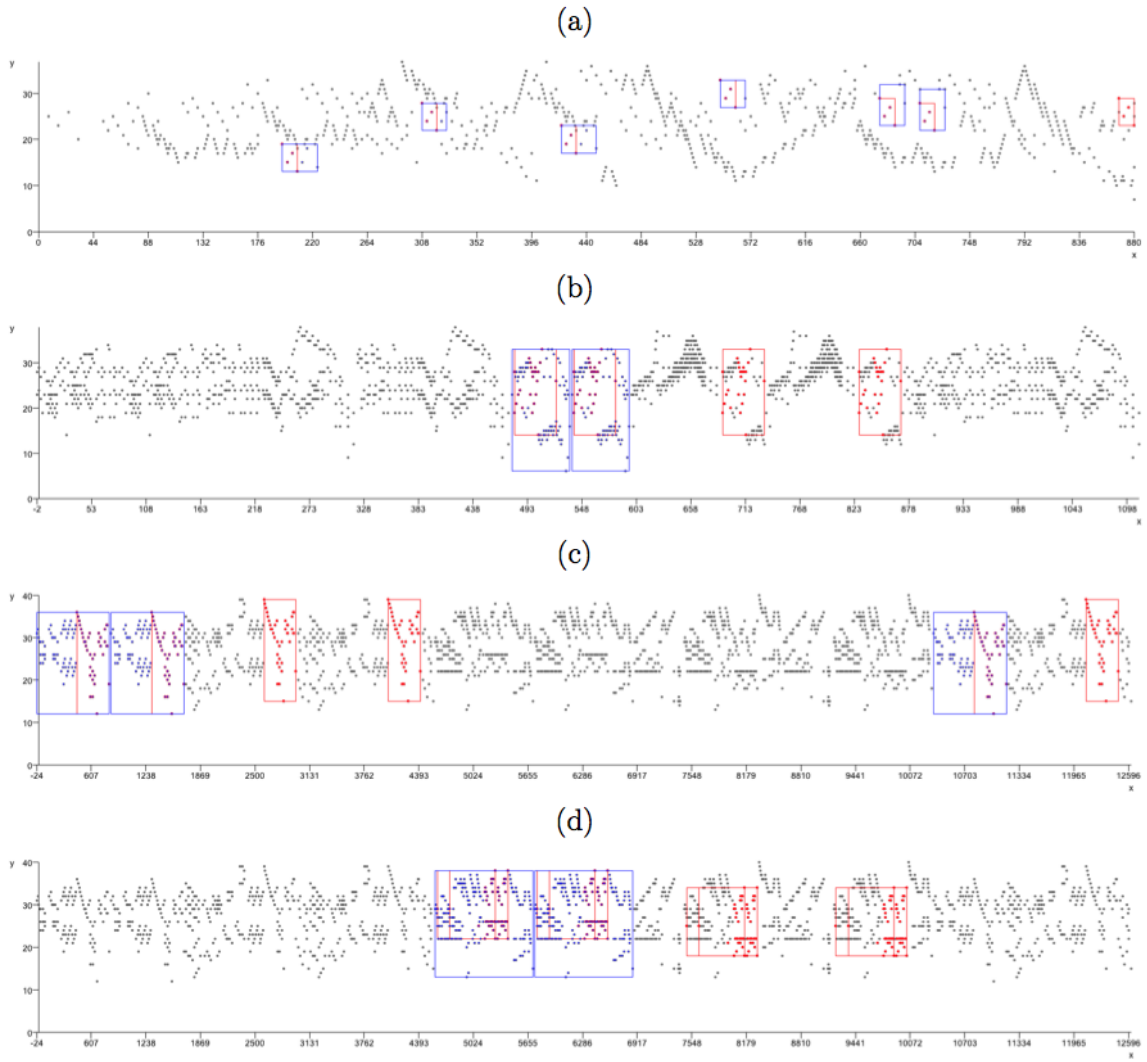
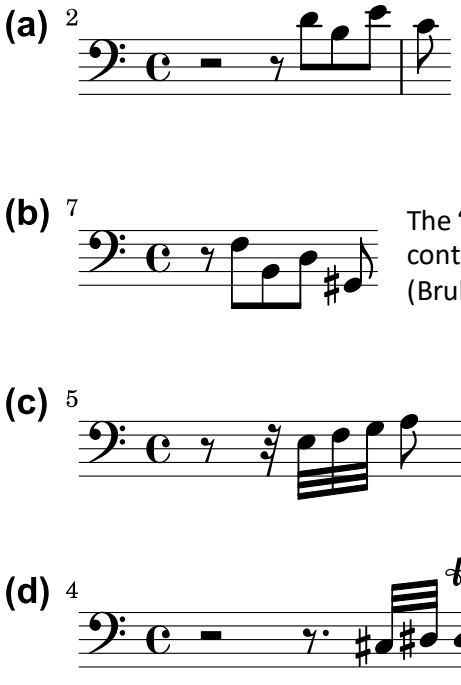


Figure 35: Examples of "missed" occurrences of ground truth patterns in the JKU PDD. The ground truth patterns are in blue, the patterns in red were discovered by COSI-ATEC. (a) In the Bach fugue, a partial occurrence of the second countersubject also occurs at the end of the piece, but is not recorded in the ground truth. (b) In the Beethoven sonata movement, the two unmatched occurrences indicated suggest two further unrecorded occurrences of this ground-truth pattern. (c) The two unmatched occurrences of the discovered pattern indicated suggest two further partial occurrences of the ground-truth pattern in the Mozart minuet and trio. (d) The four unmatched, overlapping discovered occurrences suggest two unrecorded occurrences of this ground-truth pattern in the Mozart minuet and trio.

Some interesting non-ground-truth patterns discovered by COSIATEC

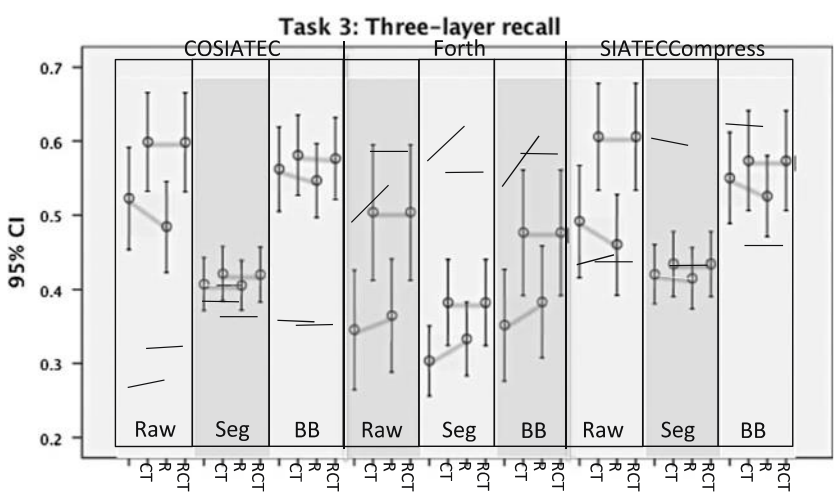


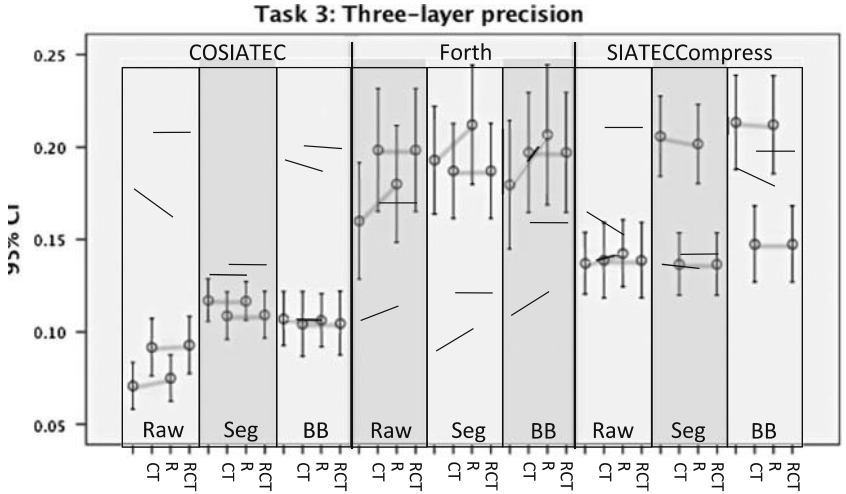
The "most characteristic segment [of the second countersubject] contains the first four eighth-notes'
(Bruhn, 1993, http://www-personal.umich.edu/~siglind/wtc-ii-20.html)

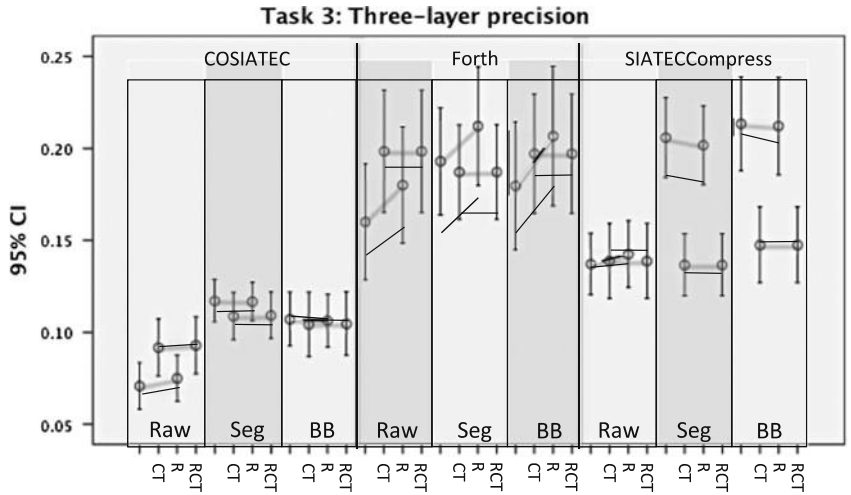
Fig. 15. Examples of noticeable and/or important patterns in Bach's Fugue in A minor (BWV 889), that were discovered by the algorithms tested in Task 2, but are not recorded in the ground truth. Patterns (a), (b) and (d) were discovered by COSIATEC. Patterns (c) and (d) were discovered by SIATECCompress.

Meredith, D. (2015). Music analysis and point-set compression. *Journal of New Music Research*, 44(3), pp. 245-270.

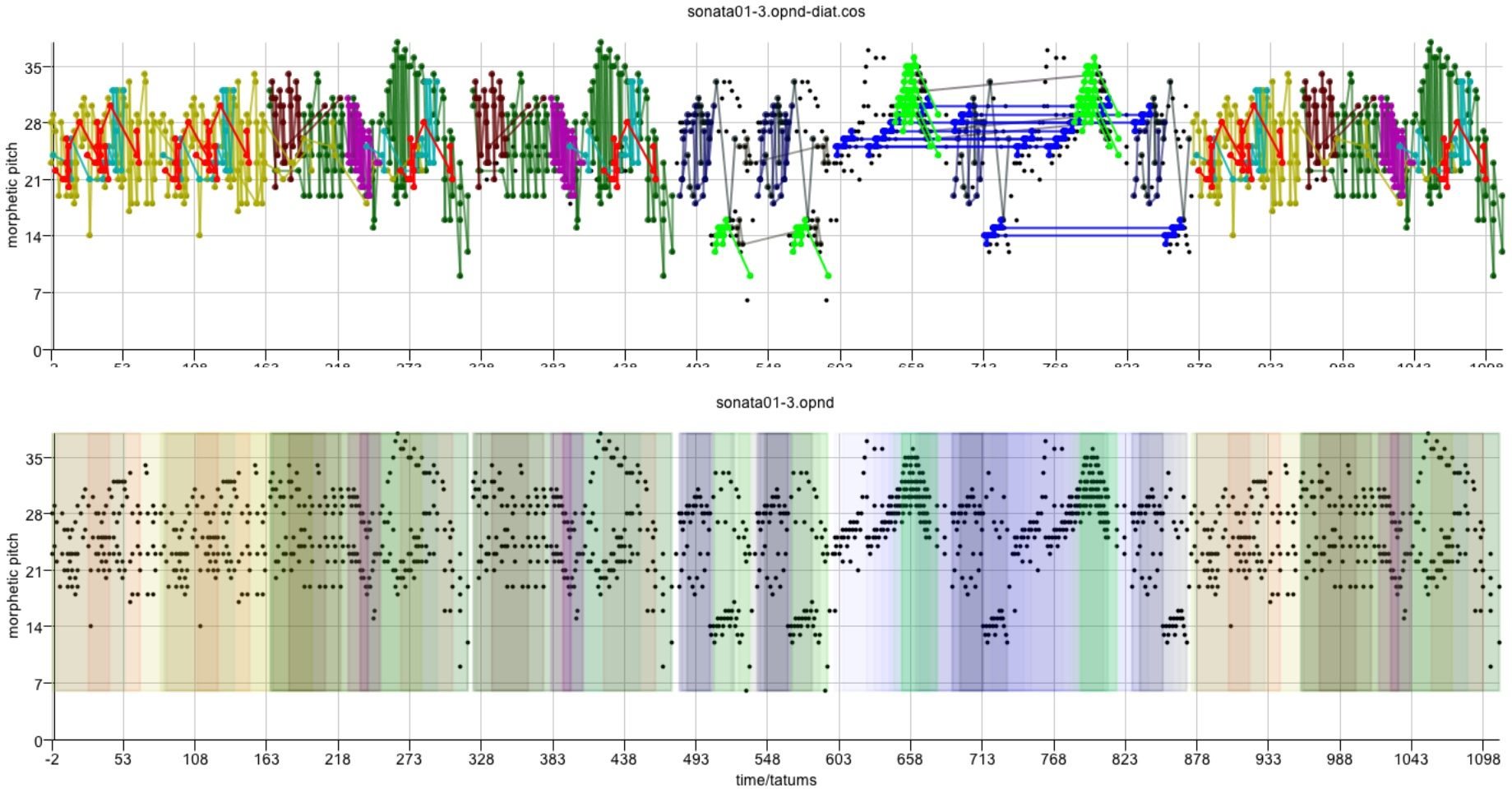
Discovering fugue subjects and counter subjects in WTC1



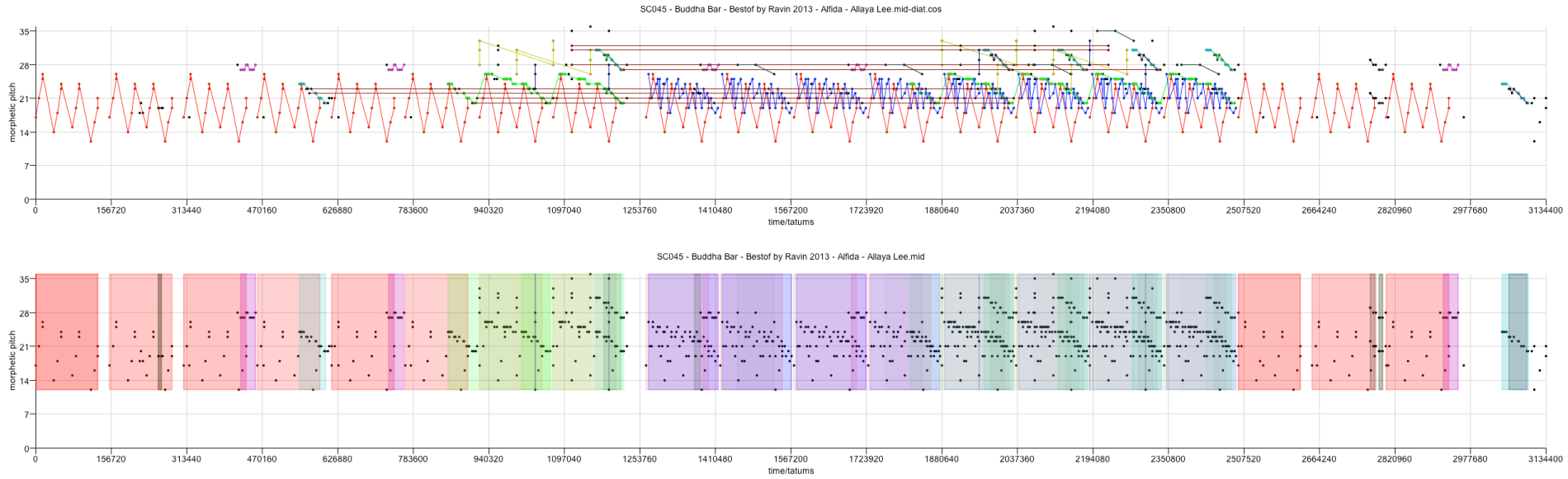




From point-set analyses to segmentations

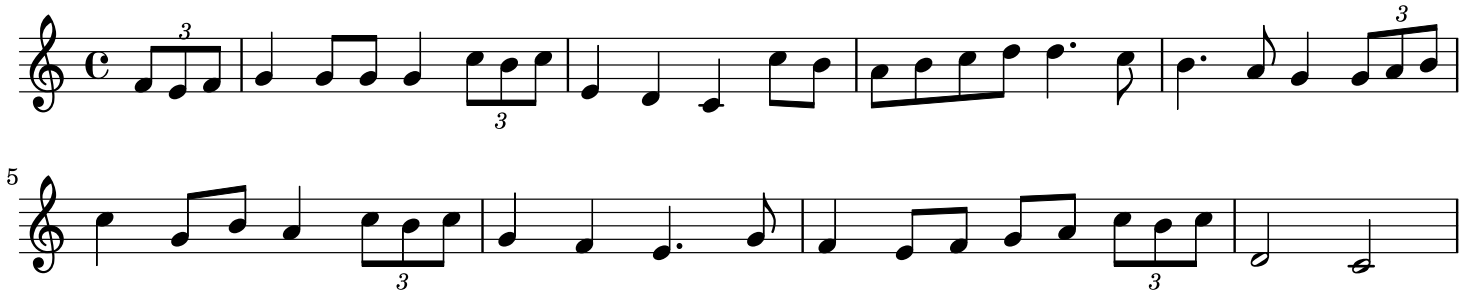


Generating segmentations from point-set analyses



- Using segmentations generated from point-set analyses for
  - improving expressive performance (with OFAI)
  - automatically generating large-scale structures of lounge music pieces to constrain generation (with SONY)

Results of point-set compression algorithms does not fully support parsimony principle – why?



$$\mathrm{NCD}(x, y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}$$

Table 2. Results on Task 1. *SR* is the classification success rate, CR_{AC} is the average compression ratio over the melodies in the *Annotated Corpus*. CR_{pairs} is the average compression ratio over the concatenated pairs of files used to obtain the NCD values.

Algorithm	*SR*	CR_{AC}	CR_{pairs}
COSIATEC	0.8389	1.5791	1.6670
COSIARTEC	0.8361	1.5726	1.6569
COSIARCTTEC	0.7917	1.4547	1.5155
COSIACTTEC	0.7694	1.4556	1.5138
ForthCT	0.6417	1.1861	1.2428
ForthRCT	0.6417	1.1861	1.2428
Forth	0.6111	1.2643	1.2663
ForthR	0.6028	1.2555	1.2655
SIARCTTECCompress	0.5750	1.3213	1.3389
SIATECCompress	0.5694	1.3360	1.3256
SIACTTECCompress	0.5250	1.3197	1.3381
SIARTECCompress	0.5222	1.3283	1.3216
bzip2	0.1250	2.7678	3.5061

Louboutin, C. and Meredith, D. (2016). Using general-purpose compression algorithms for music analysis. *Journal of New Music Research*, 45(1), pp. 1-16.

Using general-purpose compression algorithms for music analysis

- Compared LZ77, LZ78, Burrows-Wheeler and COSIATEC on classifying folk songs in the NLB
  - *k*-nearest neighbour
  - combined representations (ensemble) classification method
  - novel similarity metric, *corpus compression distance,* based on NCD
- Compared LZ77 and COSIATEC on discovery of fugue subject and countersubject entries in WTC1

Louboutin, C. and Meredith, D. (2016). Using general-purpose compression algorithms for music analysis. *Journal of New Music Research*, 45(1), pp. 1-16.

Viewpoints

Name	*Description*
basic	The basic *pitch-time* representation—i.e., a string of $(onset,\ pitch)$ points
int	A string of $(onset, pitch\ interval)$ points: $$int(p_0) = p_0$$ $$int(p_n) = (p_n.onset,\ p_n.pitch - p_{n-1}.pitch)$$
*int*0	A string of $(onset, pitch\ interval\ from\ first\ note)$ points: $$int0(p_0) = p_0$$ $$int0(p_n) = (p_n.onset,\ p_n.pitch - p_0.pitch)$$
pp	A string of $(onset, pitch\ pointer)$ points: $$pp(p_0) = p_0$$ $$pp(p_n) = \begin{cases} (p_n.onset,\ p_n.pitch), & \text{the first time the pitch occurs; and} \\ (p_n.onset,\ j - n), & \text{otherwise;} \end{cases}$$ where j is the index of the most recent occurrence of the pitch $p_n.pitch$.
ioi	Inter-onset interval: $$int(p_0) = p_0$$ $$ioi(p_n) = (p_n.onset - p_{n-1}.onset,\ p_n.pitch)$$
oip	Same as *pp* but for onset-intervals: $$oip(p_0) = p_0$$ $$oip(p_n) = \begin{cases} (p_n.onset - p_{n-1}.onset,\ p_n.pitch), & \text{the first time the IOI occurs; and} \\ (j - n,\ p_n.pitch), & \text{otherwise;} \end{cases}$$ where j is the index of the most recent occurrence of the IOI, $p_n.onset - p_{n-1}.onset$.

Corpus compression distance

- NCD is not constrained to being between 0 and 1
- For two different compression algorithms, the range of NCD values is different
- We wanted to combine NCD values from different compressors to get a new measure of similarity
- We therefore defined *corpus compression distance* as the *normalized* NCD over the corpus:

$$\mathrm{CCD}(s, s', Z, C) \;=\; \frac{\mathrm{NCD}(Z, s, s') - \min \mathcal{D}(s, C)}{\max \mathcal{D}(s, C) - \min \mathcal{D}(s, C)}$$

$$\mathcal{D}(s, C) = \{\mathrm{NCD}(Z, s_1, s_2) \mid s_1, s_2 \in C \cup \{s\}\}$$

Single-viewpoint NLB classification

Compressed viewpoint	*1-NN Leave-one-out SR*	CF_{AC}	CF_{pairs}
$(\text{COSIATEC}, basic)$	0.8528	1.5794	1.6670
$(\text{LZ77}, int \circ ioi)$	0.8222	1.4597	1.6735
$(\text{LZ77}, ioi \circ ioi)$	0.8222	1.2108	1.3547
$(\text{LZ77}, ioi)$	0.8194	1.3075	1.4915
$(\text{LZ77}, int0 \circ ioi)$	0.8139	1.3769	1.5690
$(\text{LZ77}, oip)$	0.7944	1.1188	1.2629
$(\text{LZ77}, int0 \circ oip)$	0.7861	1.1806	1.3306
$(\text{COSIATEC}, int)$	0.7556	1.5266	1.6226
$(\text{LZ77}, ioi \circ oip)$	0.7472	1.0088	1.1127
$(\text{LZ77}, int \circ oip)$	0.7444	1.2389	1.4062
(BW, ioi)	0.7333	1.9627	2.2768
$(\text{BW}, int0 \circ ioi)$	0.7194	2.0732	2.3853
$(\text{BW}, int0 \circ oip)$	0.7111	1.4192	1.5436
$(\text{LZ78}, ioi)$	0.6361	1.7542	1.9292

Louboutin, C. and Meredith, D. (2016). Using general-purpose compression algorithms for music analysis. *Journal of New Music Research*, 45(1), pp. 1-16.

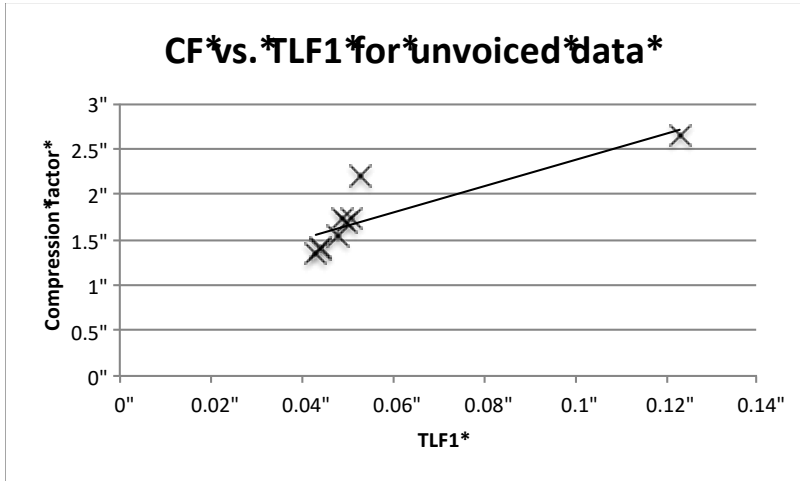
Combined viewpoints NLB classification (Ensemble classifiers)

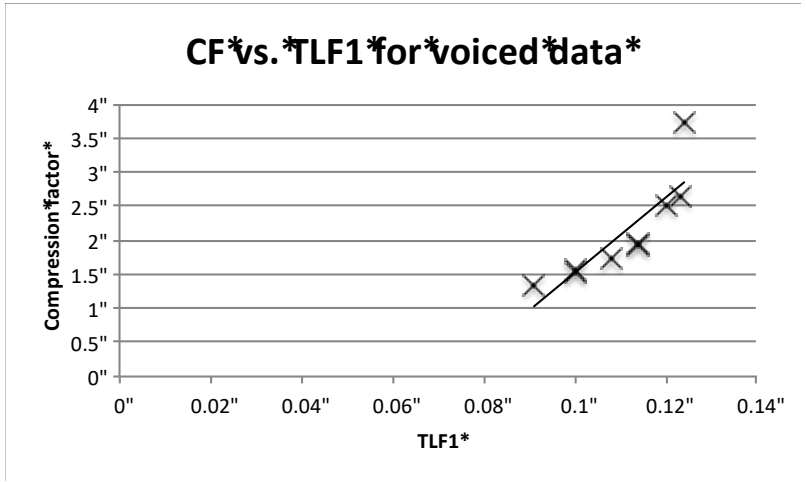
First viewpoints	*Leave-one-out SR*
2	0.8833
3	0.9139
4	0.9250
5	0.9083
6	0.9083
8	0.9194
10	0.9333
12	0.9139
14	0.9139
10′	0.9444

- 10’ is obtained by combining 8 of the 10 best compressed viewpoints, omitting (LZ77, int0 * ioi) and (COSIATEC,int)

Louboutin, C. and Meredith, D. (2016). Using general-purpose compression algorithms for music analysis. *Journal of New Music Research*, 45(1), pp. 1-16.

Comparing LZ77 and COSIATEC on discovery of fugue subject and countersubject entries





Compressed viewpoint	*TLF1*	*TLP*	*TLR*	*CF*
(COSIATEC, *basic*)	0.123	0.071	0.523	2.6404
(LZ77, $int \circ ioi$)	0.053	0.035	0.125	2.1976
(LZ77, $int \circ oip$)	0.051	0.034	0.118	1.7464
(LZ77, ioi)	0.050	0.032	0.097	1.6876
(LZ77, $int0 \circ ioi$)	0.049	0.033	0.097	1.7305
(LZ77, $ioi \circ ioi$)	0.048	0.033	0.095	1.5397
(LZ77, oip)	0.044	0.030	0.087	1.3929
(LZ77, $int0 \circ oip$)	0.044	0.030	0.087	1.4255
(LZ77, $ioi \circ oip$)	0.043	0.030	0.080	1.3374

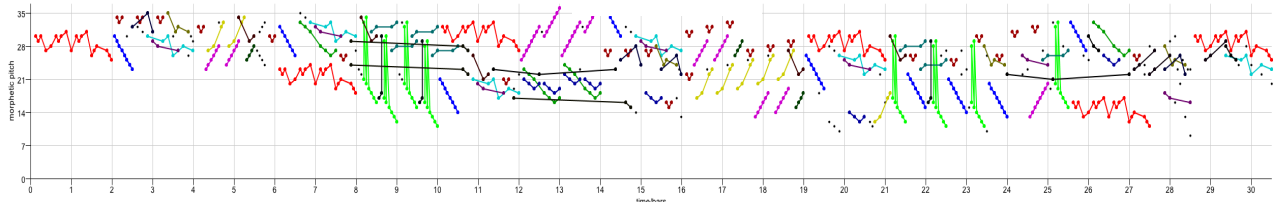
Compressed viewpoint	*TLF1*	*TLP*	*TLR*	*CF*
(LZ77, *int* ¶*ioi*)	0.124	0.073	0.521	3.7142
(COSIATEC, *basic*)	0.123	0.071	0.523	2.6404
(LZ77, *int* ¶*oip*)	0.120	0.072	0.441	2.5008
(LZ77, *ioi*)	0.114	0.073	0.298	1.9374
(LZ77, *int*0 ¶*ioi*)	0.114	0.073	0.298	1.9553
(LZ77, *ioi* ¶*ioi*)	0.108	0.068	0.280	1.7152
(LZ77, *oip*)	0.100	0.065	0.234	1.5428
(LZ77, *int*0 ¶*oip*)	0.100	0.065	0.234	1.5584
(LZ77, *ioi* ¶*oip*)	0.091	0.063	0.202	1.3424

Correlation between CF and TLF1: r= 0.86, N = 9, p = 0.003

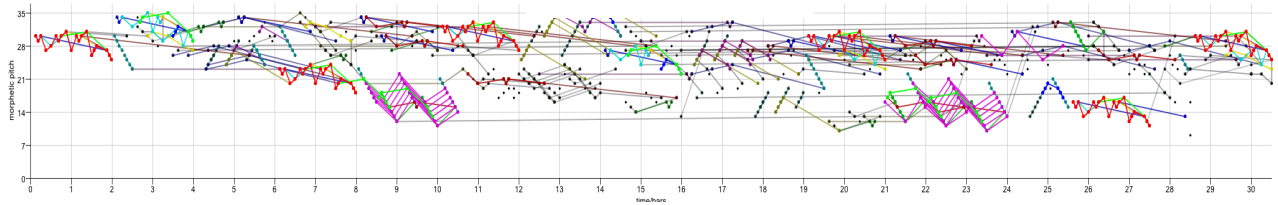
(NB: SIATECCompressBB achieved TLF1 of 0.301)

Compression-driven point-set pattern discovery in music

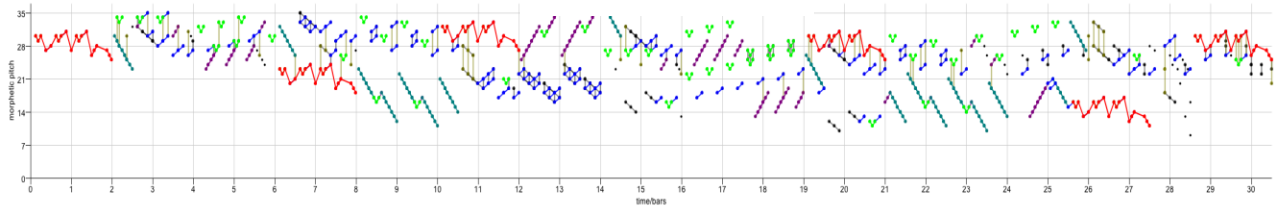
COSIATEC



SIATECCompress

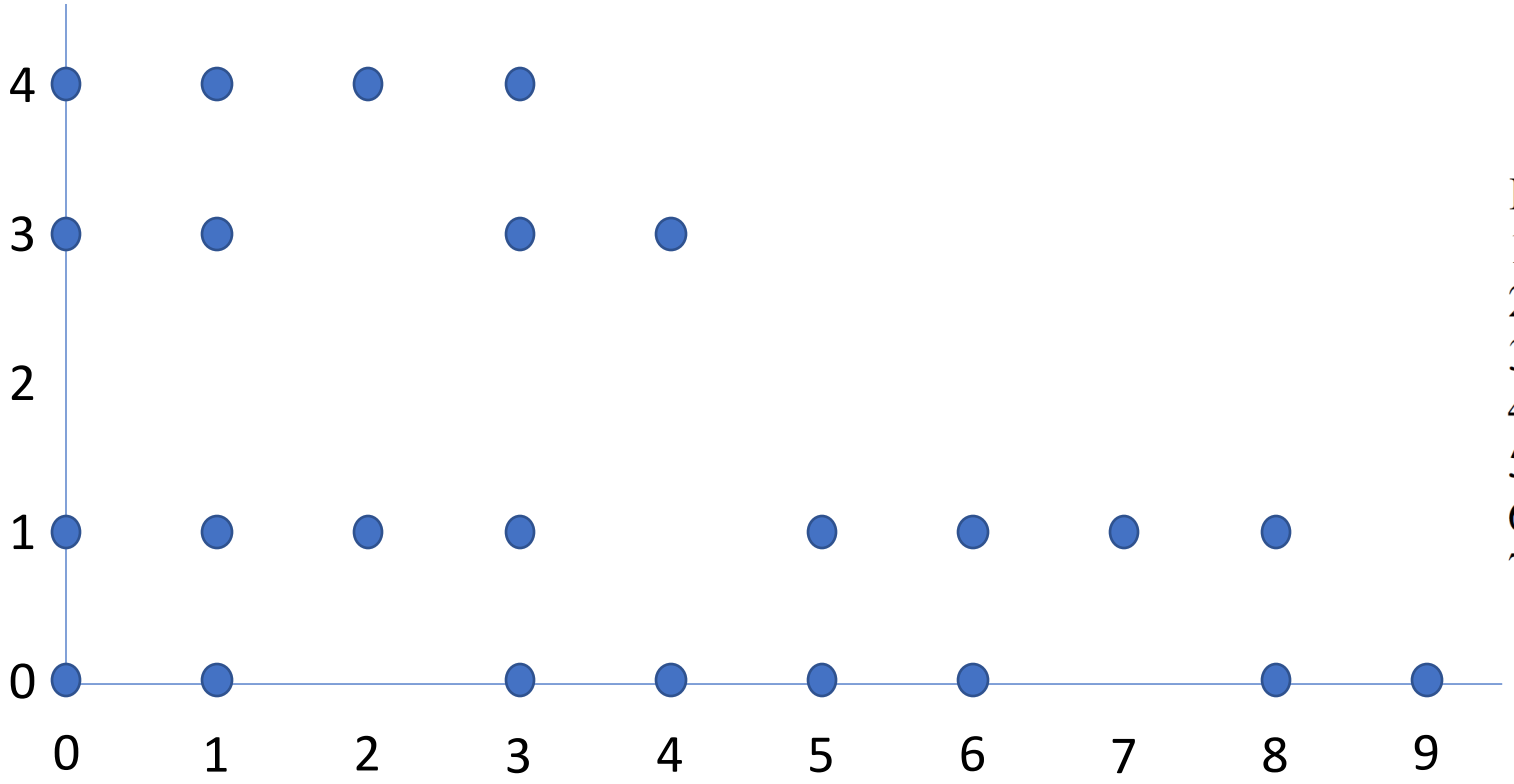


Forth's algorithm



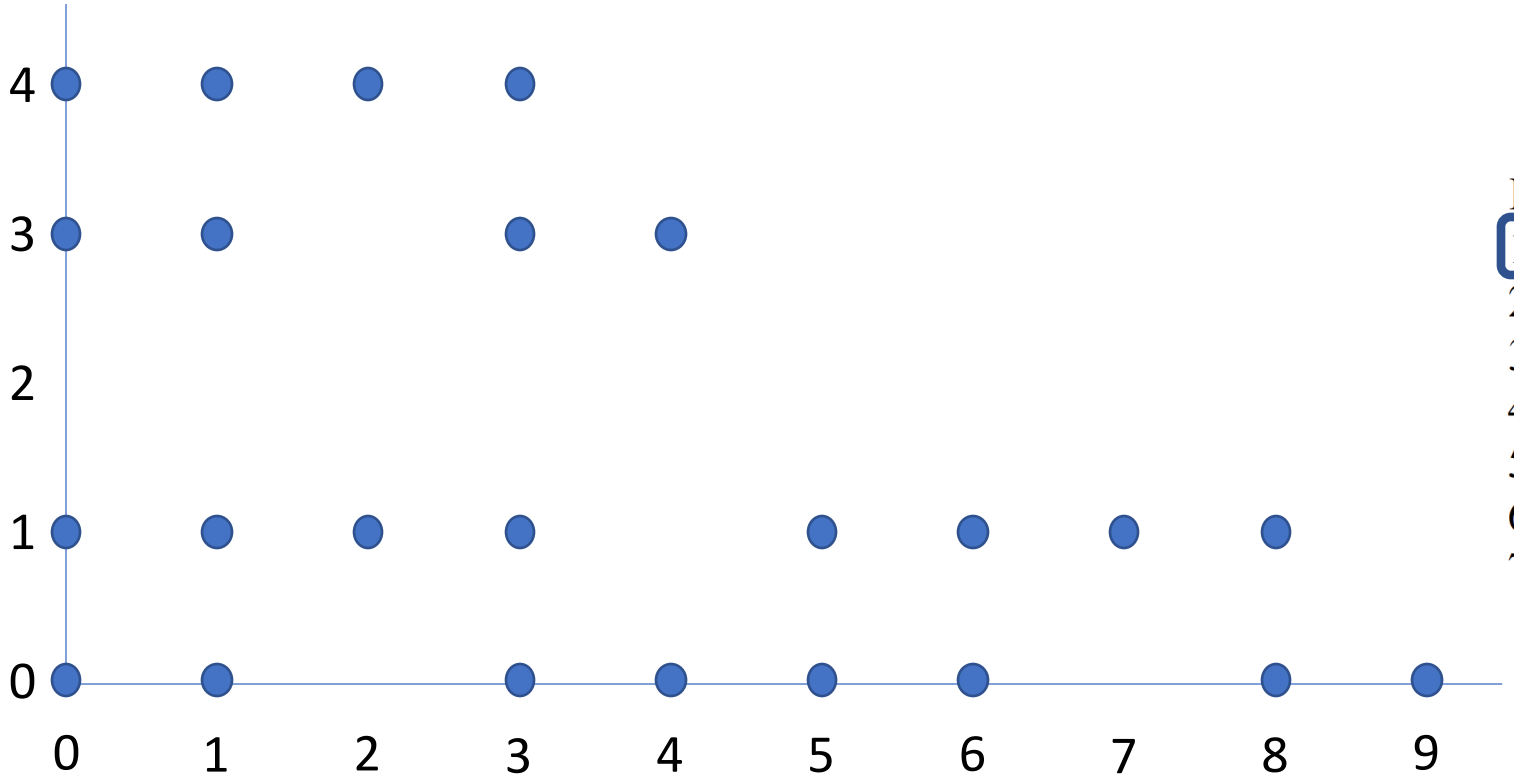
- Principle of parsimony:
  - Given two models that equally accurately describe the data, the simpler one is less likely to be an accurate description by chance
- Have applied compression-based point-set cover algorithms to a number of different musicological tasks with some success
  - classification of folk songs, simulation of human analyses, discovery of subjects and counter-subjects in fugues
- Some evidence that better compression is correlated with better performance (Louboutin and Meredith, JNMR, 2016) and better simulation of cognition (Collins *et al.,* Music Perception, 2011)
- Motivated to develop techniques that give us better compression, in the hope that the more compressed encodings of musical objects will represent better ways of understanding them

RecurSIA: Recursive translatable pattern discovery



RECURSIA$(\mathcal{A}, D)$
1 $\mathbf{E} \leftarrow \mathcal{A}(D)$
2 **if** $|\mathbf{E}| = 1 \wedge |\mathbf{E}[0][1]| = 1$ **return** $\mathbf{E}$
3 **for** $i \leftarrow 0$ **to** $|\mathbf{E}| - 1$
4 $\quad \mathbf{e} \leftarrow$ RECURSIA$(\mathcal{A}, \mathbf{E}[i][0])$
5 $\quad$ **if** $|\mathbf{e}| > 1 \vee |\mathbf{e}[0][1]| > 1$
6 $\quad\quad \mathbf{E}[i][0] \leftarrow \mathbf{e}$
7 **return** $\mathbf{E}$

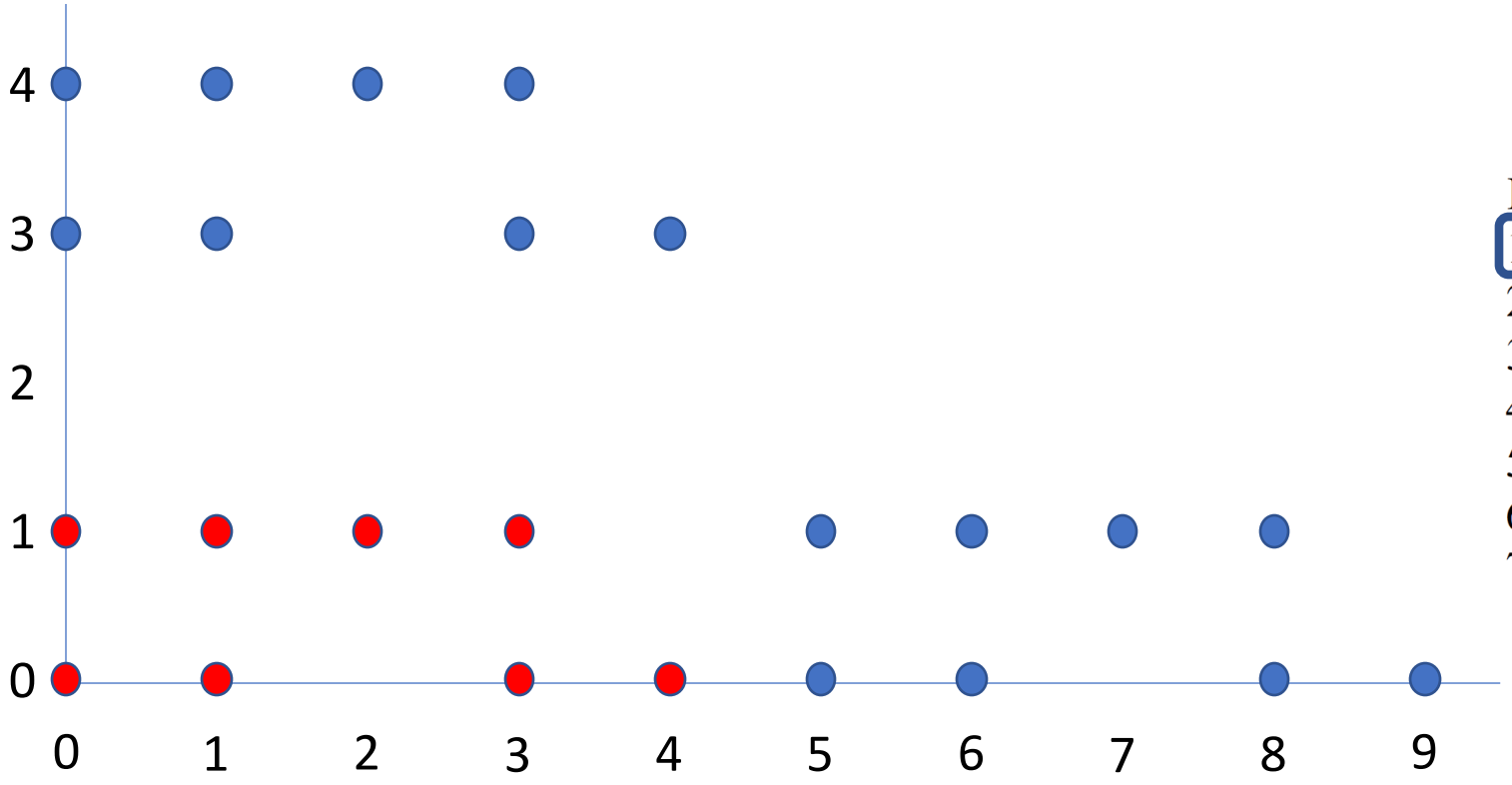
RecurSIA: Recursive translatable pattern discovery




```
RecurSIA(𝒜, D)
1    E ← 𝒜(D)
2    if |E| = 1 ∧ |E[0][1]| = 1 return E
3    for i ← 0 to |E| − 1
4        e ← RecurSIA(𝒜, E[i][0])
5        if |e| > 1 ∨ |e[0][1]| > 1
6            E[i][0] ← e
7    return E
```

```
T(P(p(0,0),p(0,1),p(1,0),p(1,1),p(2,1),p(3,0),p(3,1),p(4,0)),V(v(0,0),v(0,3),v(5,0)))
```

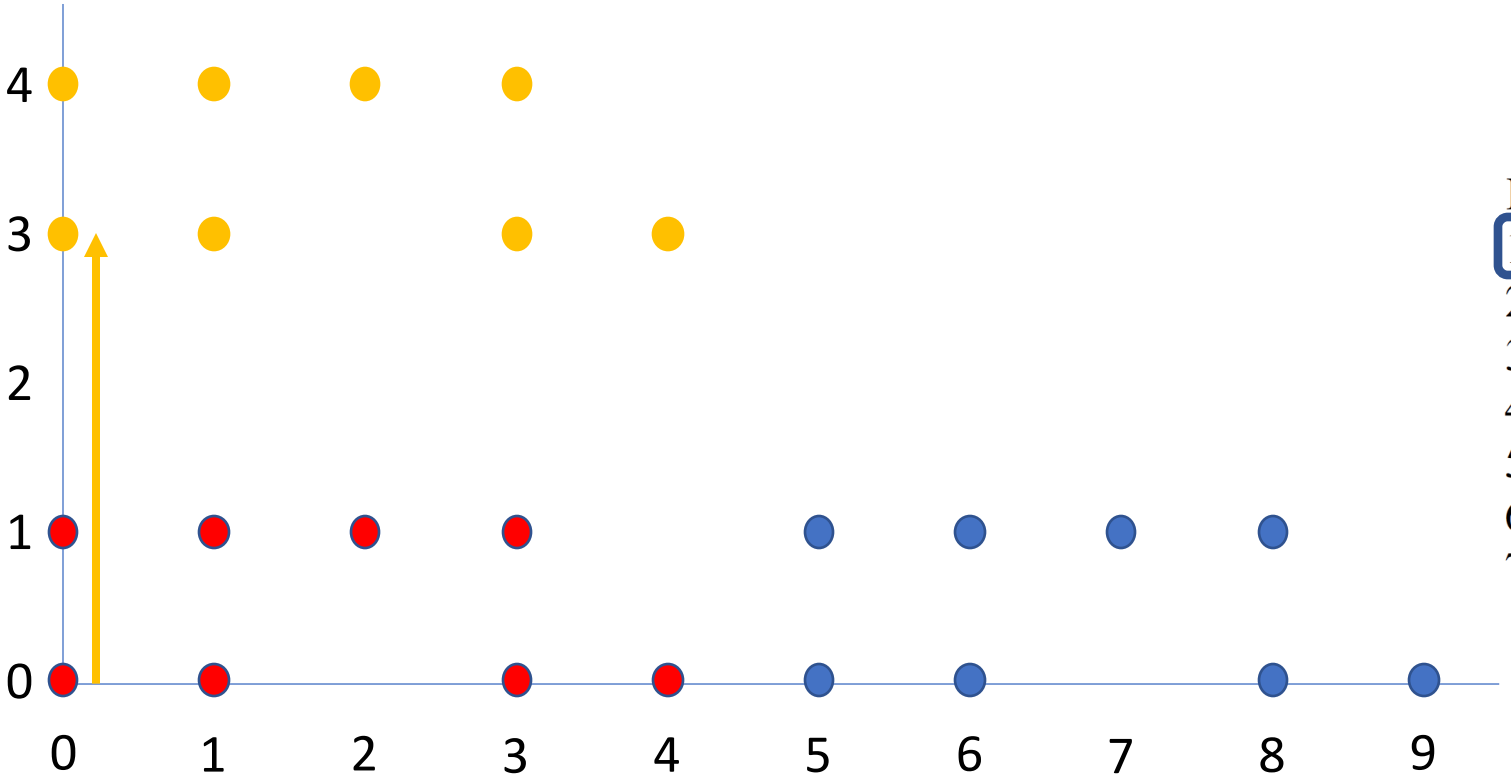
RecurSIA: Recursive translatable pattern discovery




```
RecurSIA(𝒜, D)
1   E ← 𝒜(D)
2   if |E| = 1 ∧ |E[0][1]| = 1 return E
3   for i ← 0 to |E| − 1
4      e ← RecurSIA(𝒜, E[i][0])
5      if |e| > 1 ∨ |e[0][1]| > 1
6         E[i][0] ← e
7   return E
```

```
T(P(p(0,0),p(0,1),p(1,0),p(1,1),p(2,1),p(3,0),p(3,1),p(4,0)),V(v(0,3),v(5,0)))
```

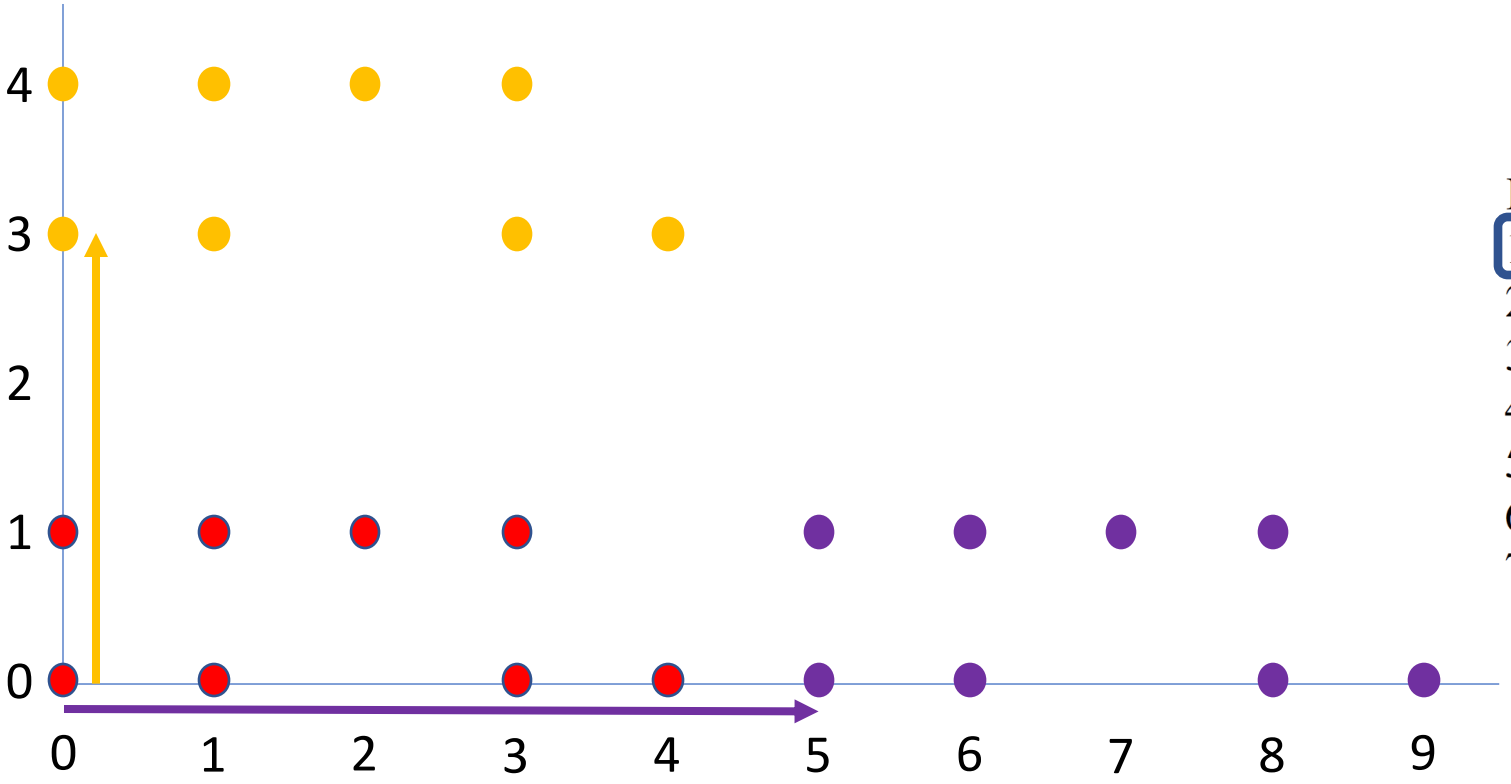
RecurSIA: Recursive translatable pattern discovery




```
RecurSIA(𝒜, D)
1    E ← 𝒜(D)
2    if |E| = 1 ∧ |E[0][1]| = 1 return E
3    for i ← 0 to |E| − 1
4        e ← RecurSIA(𝒜, E[i][0])
5        if |e| > 1 ∨ |e[0][1]| > 1
6            E[i][0] ← e
7    return E
```

T(P(p(0,0),p(0,1),p(1,0),p(1,1),p(2,1),p(3,0),p(3,1),p(4,0)),V(v(0,3),v(5,0)))

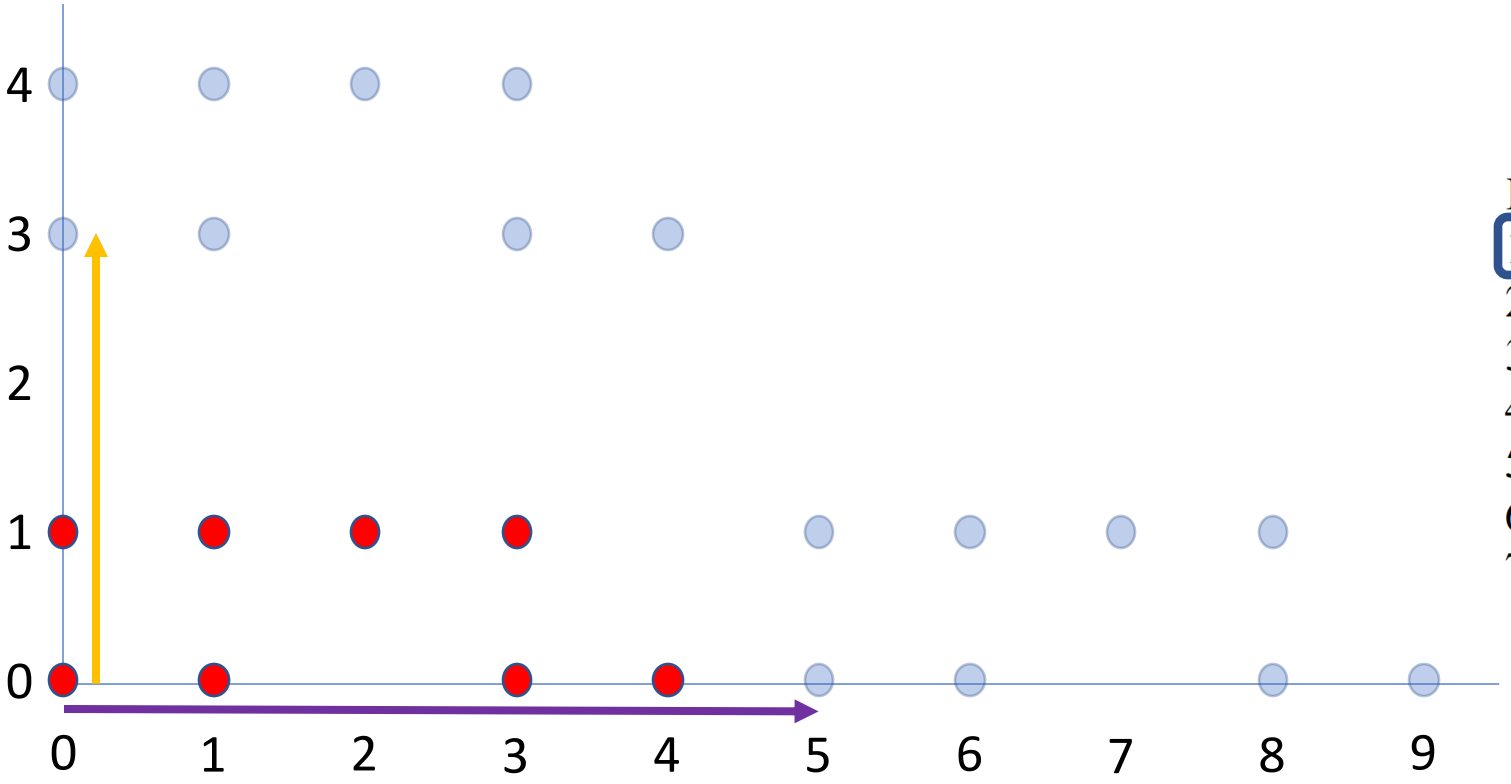
RecurSIA: Recursive translatable pattern discovery




```
RecurSIA(𝒜, D)
1    E ← 𝒜(D)
2    if |E| = 1 ∧ |E[0][1]| = 1 return E
3    for i ← 0 to |E| − 1
4        e ← RecurSIA(𝒜, E[i][0])
5        if |e| > 1 ∨ |e[0][1]| > 1
6            E[i][0] ← e
7    return E
```

T(P(p(0,0),p(0,1),p(1,0),p(1,1),p(2,1),p(3,0),p(3,1),p(4,0)),V(v(0,3),v(5,0)))

RecurSIA: Recursive translatable pattern discovery

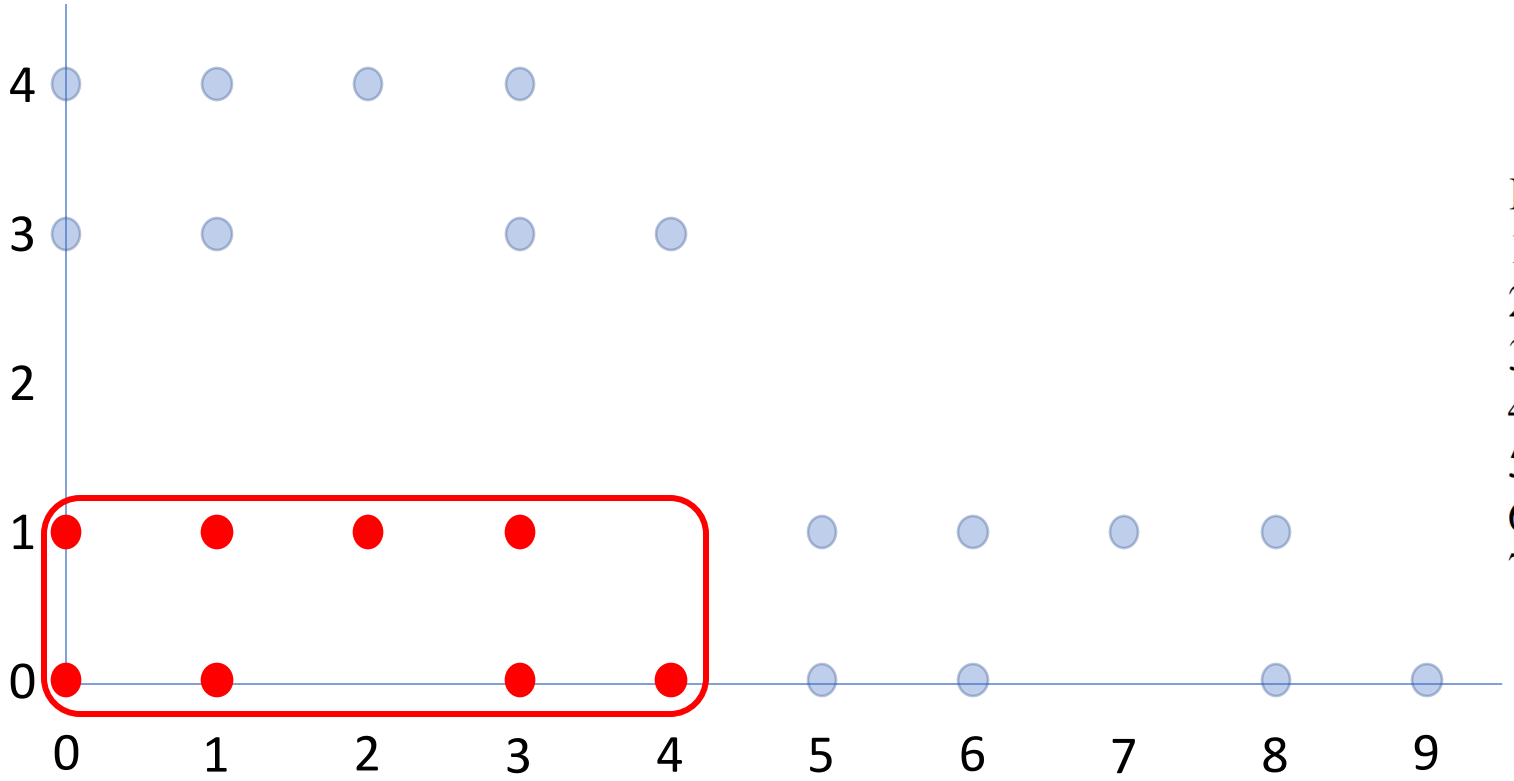



```
RecurSIA(A, D)
1    E ← A(D)
2    if |E| = 1 ∧ |E[0][1]| = 1 return E
3    for i ← 0 to |E| − 1
4        e ← RecurSIA(A, E[i][0])
5        if |e| > 1 ∨ |e[0][1]| > 1
6            E[i][0] ← e
7    return E
```

T(P(p(0,0),p(0,1),p(1,0),p(1,1),p(2,1),p(3,0),p(3,1),p(4,0)),V(v(0,3),v(5,0)))

Compression factor without RecurSIA = 24/(8+2) = 2.4

RecurSIA: Recursive translatable pattern discovery

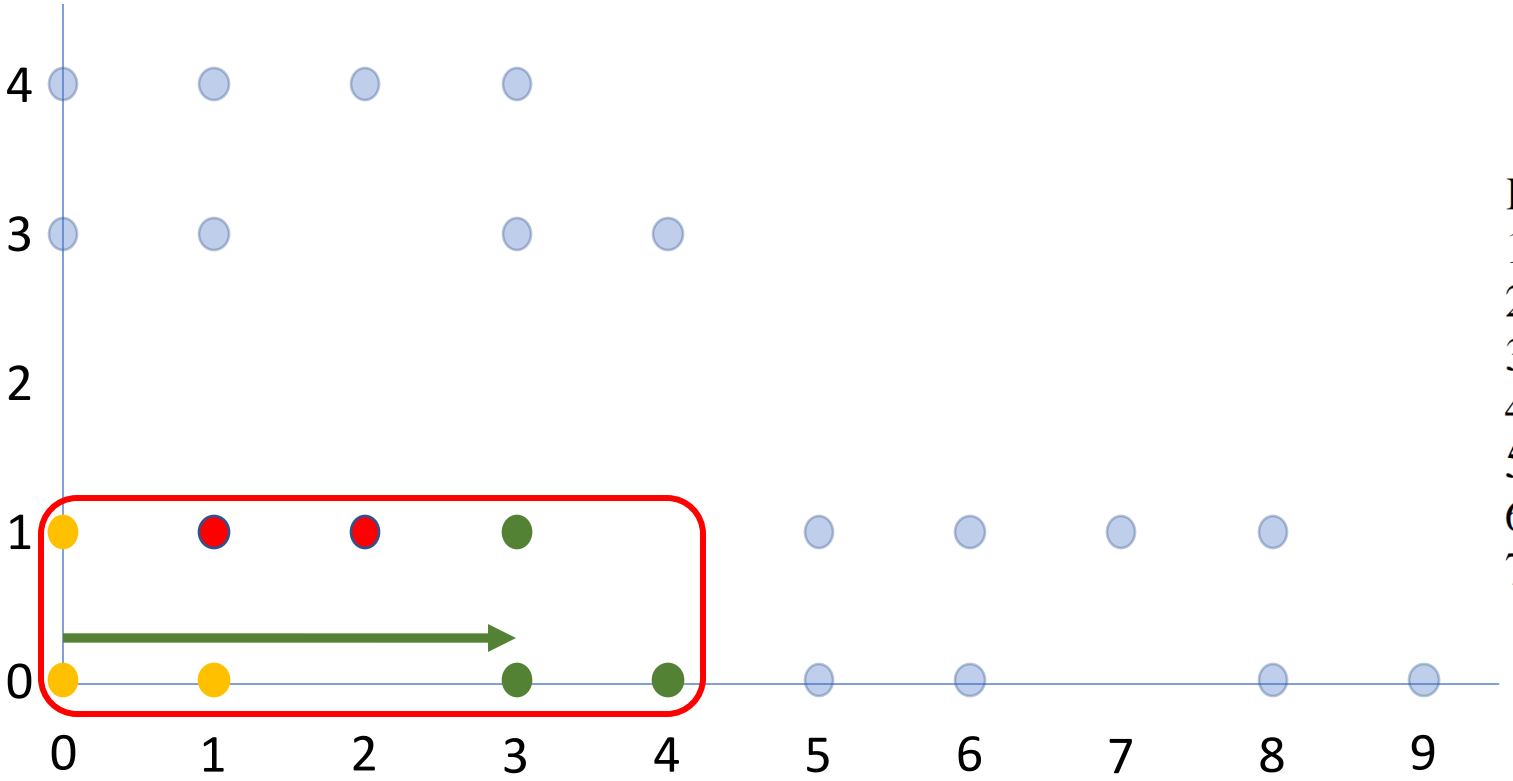


RECURSIA($\mathcal{A}, D$)
1 $\mathbf{E} \leftarrow \mathcal{A}(D)$
2 **if** $|\mathbf{E}| = 1 \wedge |\mathbf{E}[0][1]| = 1$ **return** $\mathbf{E}$
3 **for** $i \leftarrow 0$ **to** $|\mathbf{E}| - 1$
4 $\mathbf{e} \leftarrow$ RECURSIA($\mathcal{A}, \mathbf{E}[i][0]$)
5 **if** $|\mathbf{e}| > 1 \vee |\mathbf{e}[0][1]| > 1$
6 $\mathbf{E}[i][0] \leftarrow \mathbf{e}$
7 **return** $\mathbf{E}$

```
T(P(p(0,0),p(0,1),p(1,0),p(1,1),p(2,1),p(3,0),p(3,1),p(4,0)),V(v(0,3),v(5,0)))
```

```
T(P(T(P(p(0,0),p(0,1),p(1,0)),V(v(3,0))),P(p(1,1),p(2,1))),V(v(0,3),v(5,0)))
```

RecurSIA: Recursive translatable pattern discovery

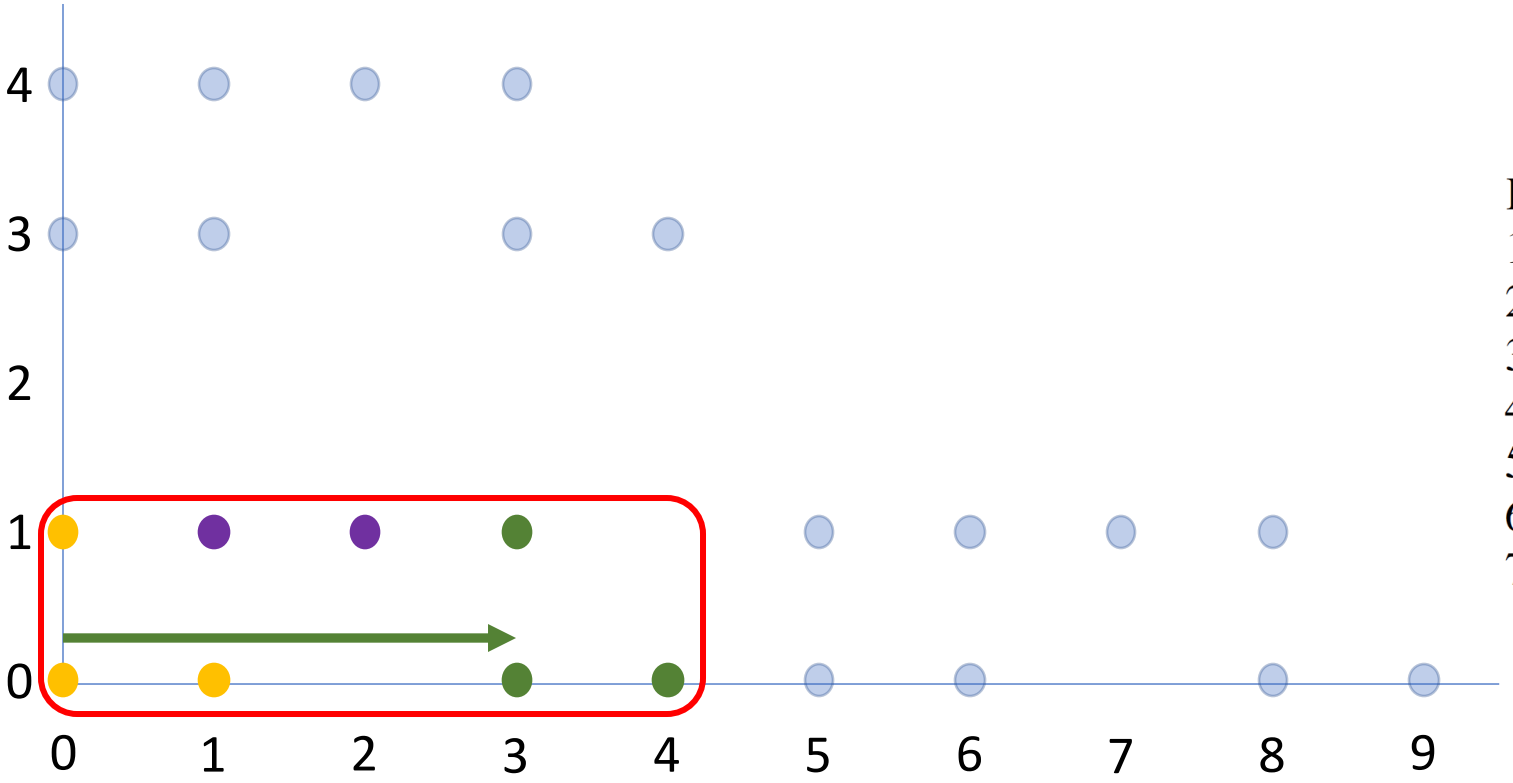



```
RecurSIA(𝒜, D)
1   E ← 𝒜(D)
2   if |E| = 1 ∧ |E[0][1]| = 1 return E
3   for i ← 0 to |E| − 1
4     e ← RecurSIA(𝒜, E[i][0])
5     if |e| > 1 ∨ |e[0][1]| > 1
6       E[i][0] ← e
7   return E
```

```
T(P(p(0,0),p(0,1),p(1,0),p(1,1),p(2,1),p(3,0),p(3,1),p(4,0)),V(v(0,3),v(5,0)))
```

```
T(P(T(P(p(0,0),p(0,1),p(1,0)),V(v(3,0))),P(p(1,1),p(2,1))),V(v(0,3),v(5,0)))
```

RecurSIA: Recursive translatable pattern discovery



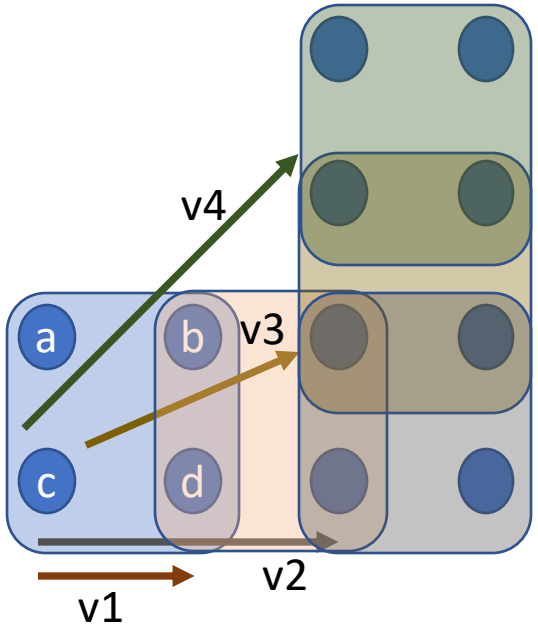

```
RecurSIA(A, D)
1   E ← A(D)
2   if |E| = 1 ∧ |E[0][1]| = 1 return E
3   for i ← 0 to |E| − 1
4     e ← RecurSIA(A, E[i][0])
5     if |e| > 1 ∨ |e[0][1]| > 1
6       E[i][0] ← e
7   return E
```

T(P(p(0,0),p(0,1),p(1,0),p(1,1),p(2,1),p(3,0),p(3,1),p(4,0)),V(v(0,3),v(5,0)))

T(P(T(P(p(0,0),p(0,1),p(1,0)),V(v(3,0))),P(p(1,1),p(2,1))),V(v(0,3),v(5,0)))

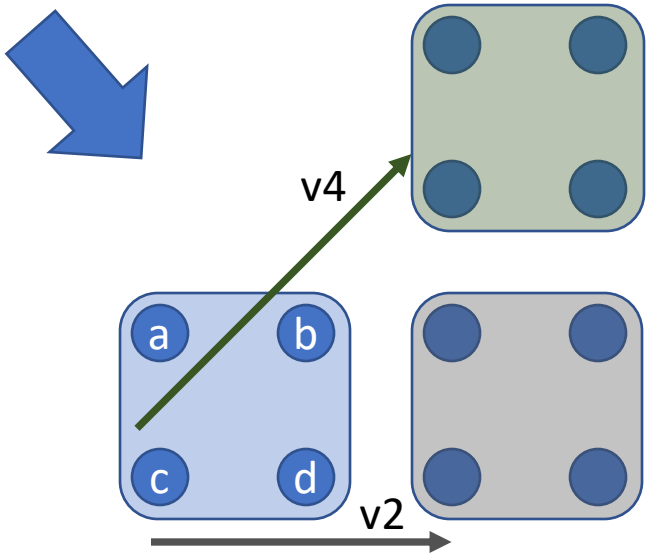
Compression factor with RecurSIA = 24/(3+1+2+2) = 3.0

RRT: Removing redundant translators



T(P(a,b,c,d),V(v1,v2,v3,v4))

Length = 8



T(P(a,b,c,d),V(v2,v4))

Length = 6

RRT(T)
1 $\mathbf{F} \leftarrow$ ComputePointFreqSet(T)
2 **if** $\mathbf{F}[|\mathbf{F}|-1][0] = 1$ **return** T
3 $\mathbf{S} \leftarrow$ ComputeSIAMVectorTable($T, \mathbf{F}$)
4 $\mathbf{R} \leftarrow$ ComputeRemovableVectors($T, \mathbf{S}$)
5 $M \leftarrow$ ComputeMaxPoints($T, \mathbf{R}, \mathbf{F}$)
6 **if** $M = \emptyset$ **then** $T[1] \backslash\leftarrow \mathbf{R}$, **return** T
7 $\mathbf{V} \leftarrow$ ComputeVectorMaxPointSetPairs(M)
8 $\mathbf{Q} \leftarrow$ ComputeRetainedVectors($\mathbf{V}$)
9 **return** RemoveRedundantVectors($T, \mathbf{Q}, \mathbf{R}$)

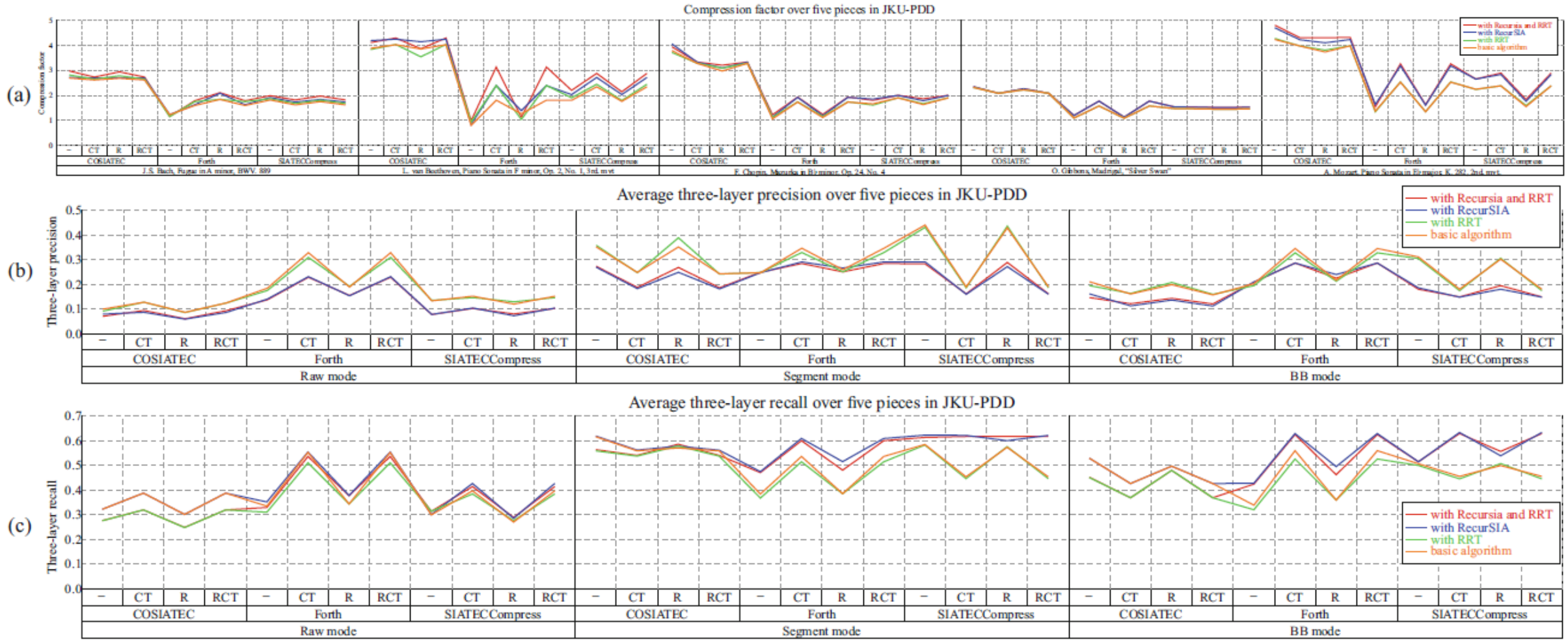
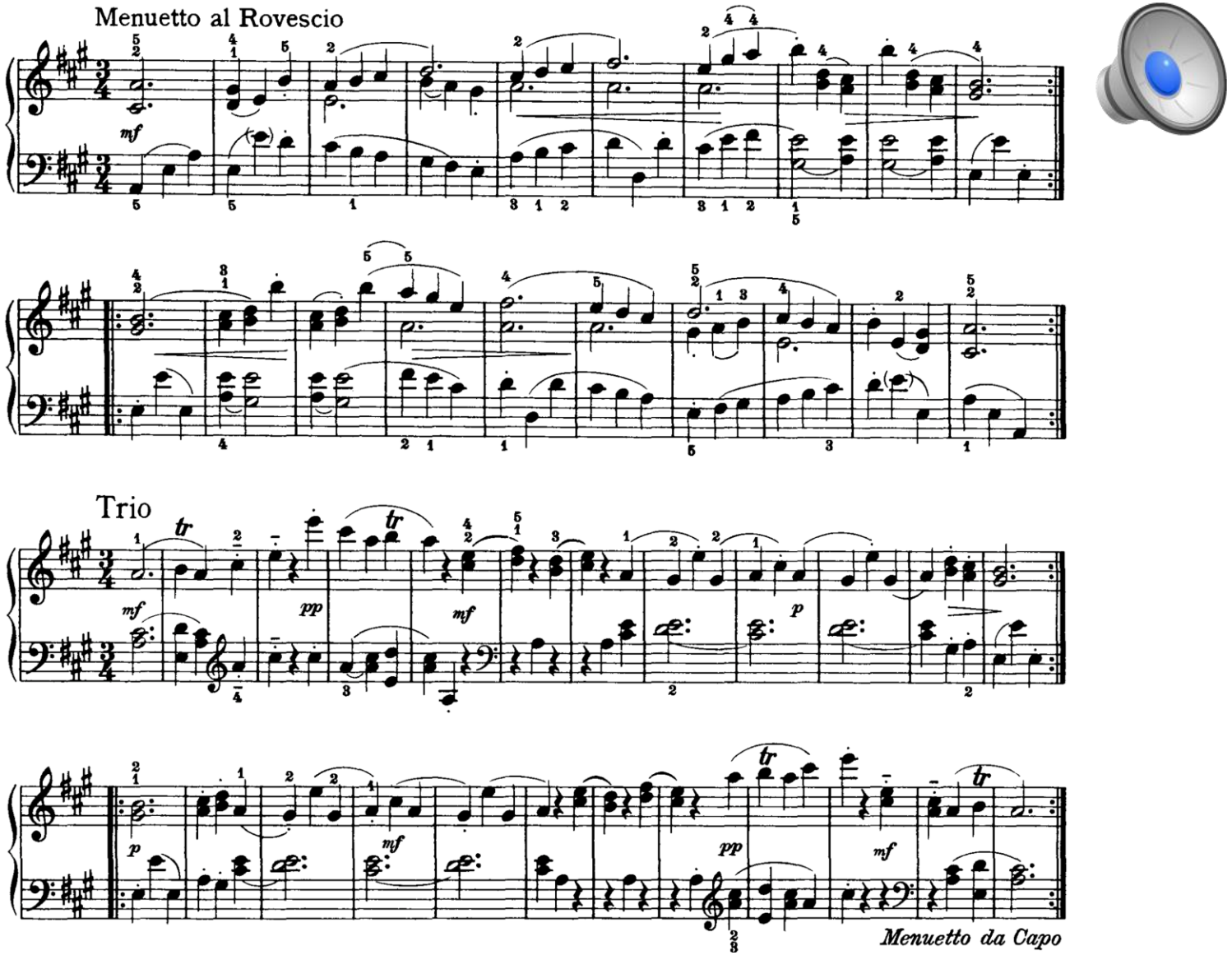
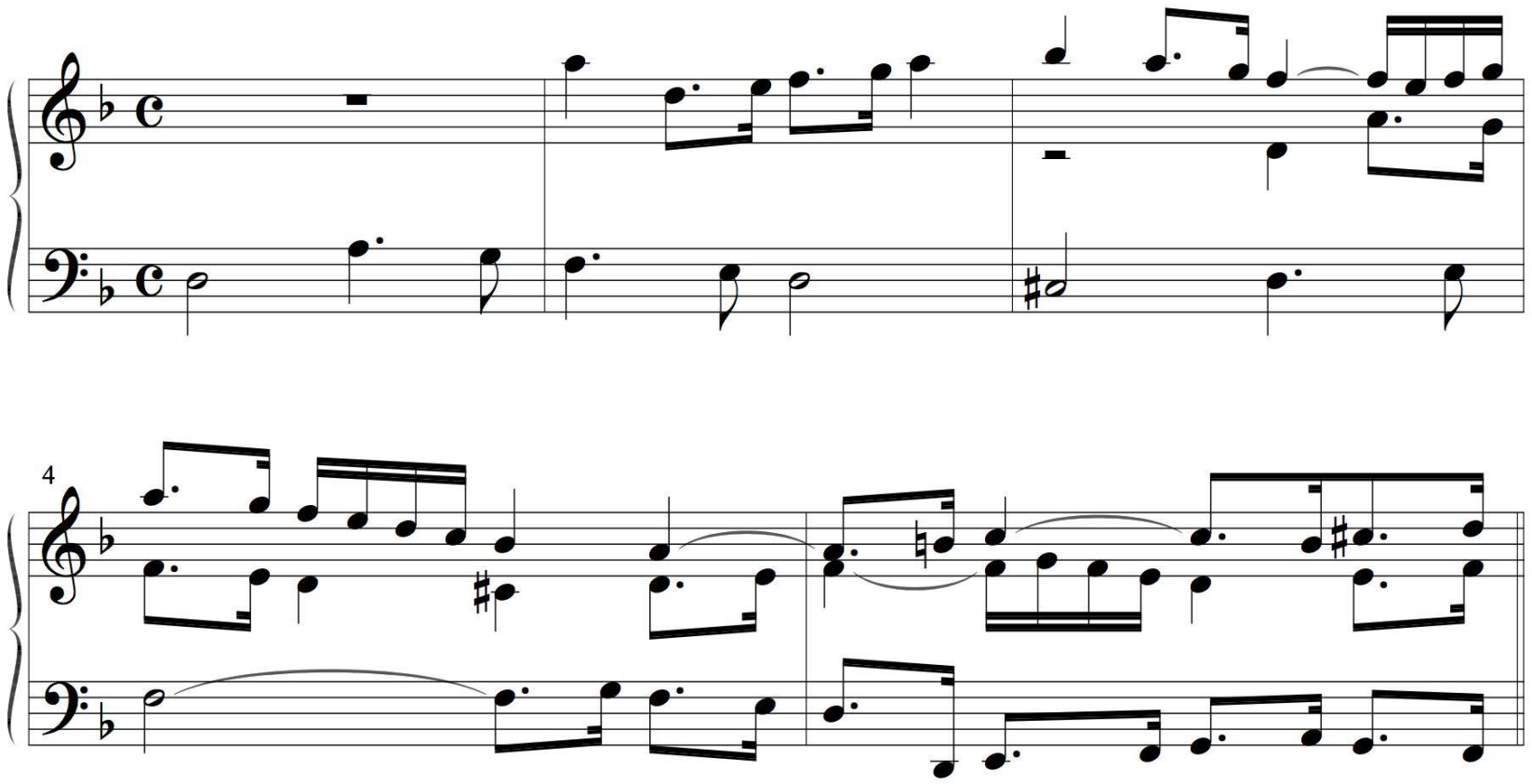


Fig. 4. Effect of RecurSIA and RRT on compression factor (a), three-layer precision (b) and recall (c), over the pieces in the JKU-PDD. (Color figure online)



Haydn, Menuet from Sonata in A Major (Hob.VXI:26)



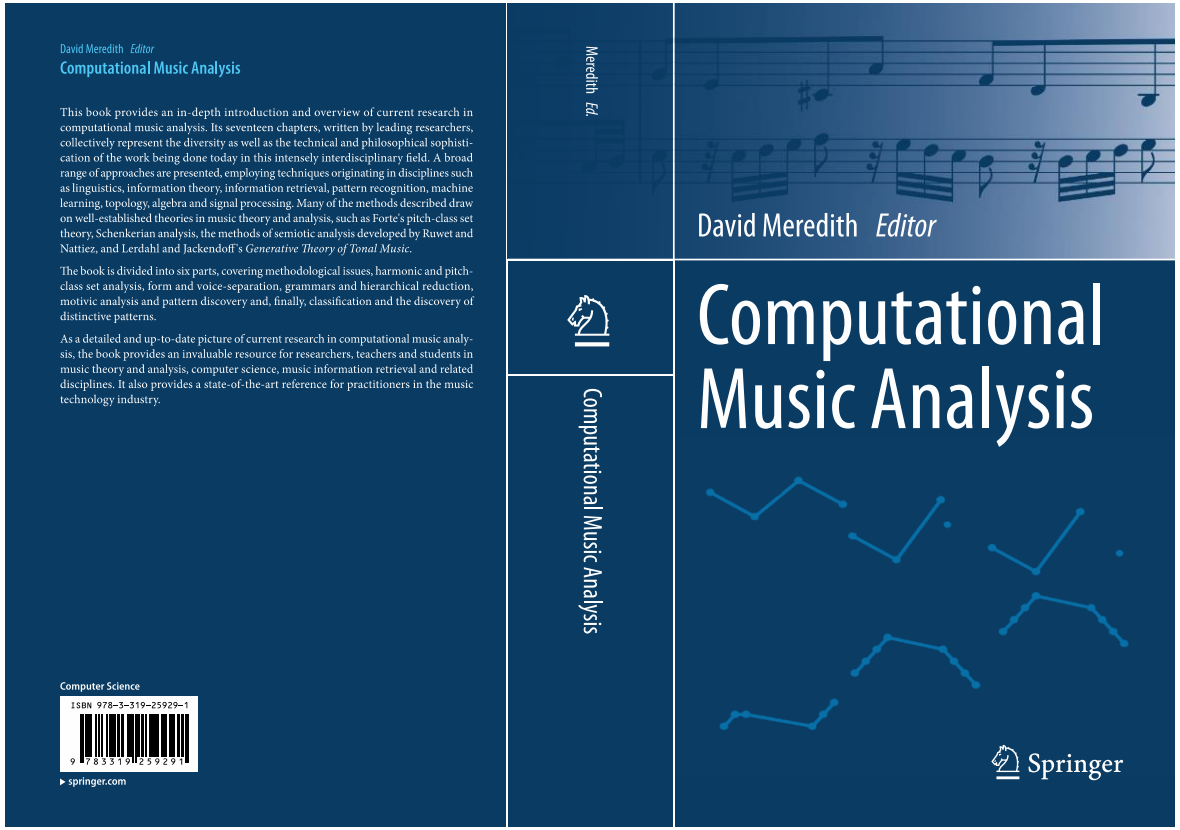
J. S. Bach, Contrapunctus 6 from *Die Kunst der Fuge*, BWV 1080

Scalable patterns

- Concerned with pairs of patterns related by augmentation, diminution, inversion and retrograde

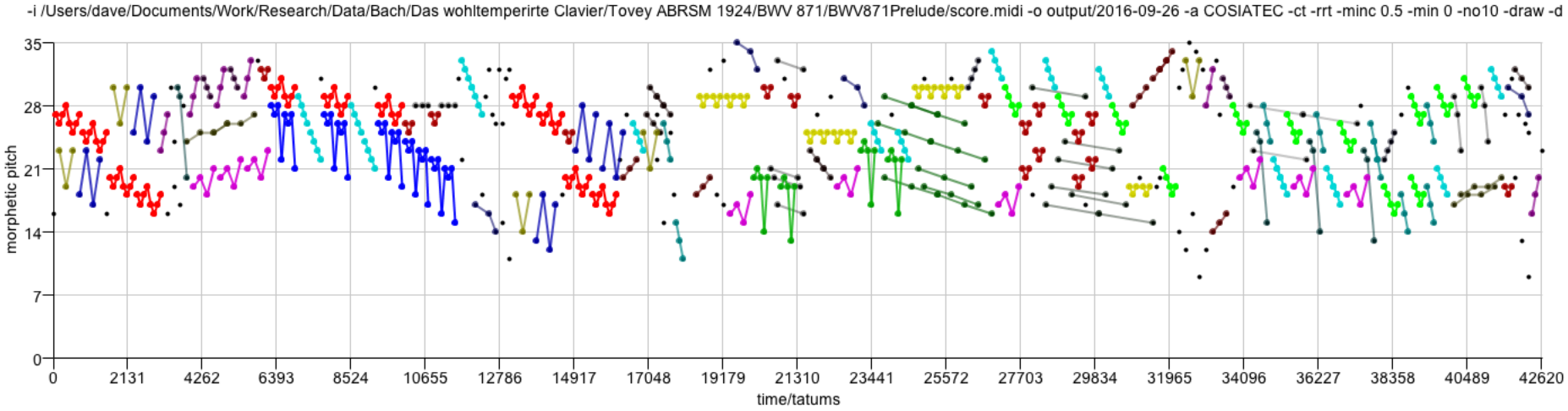
Musical transformation	Geometric transformation in pitch-time space	x scale factor	y scale factor
Repeat with transposition	Translation	1	1
Inversion	Reflection in axis || re time axis	1	-1
Diminution	Directional horizontal scaling	< 1	1
Augmentation	Directional horizontal scaling	> 1	1
Retrograde	Reflection in axis || re pitch axis	-1	1

Springer book on Computational Music Analysis



- http://link.springer.com/book/10.1007/978-3-319-25931-4

AALBORG UNIVERSITY
DENMARK



OMNISIA

David Meredith

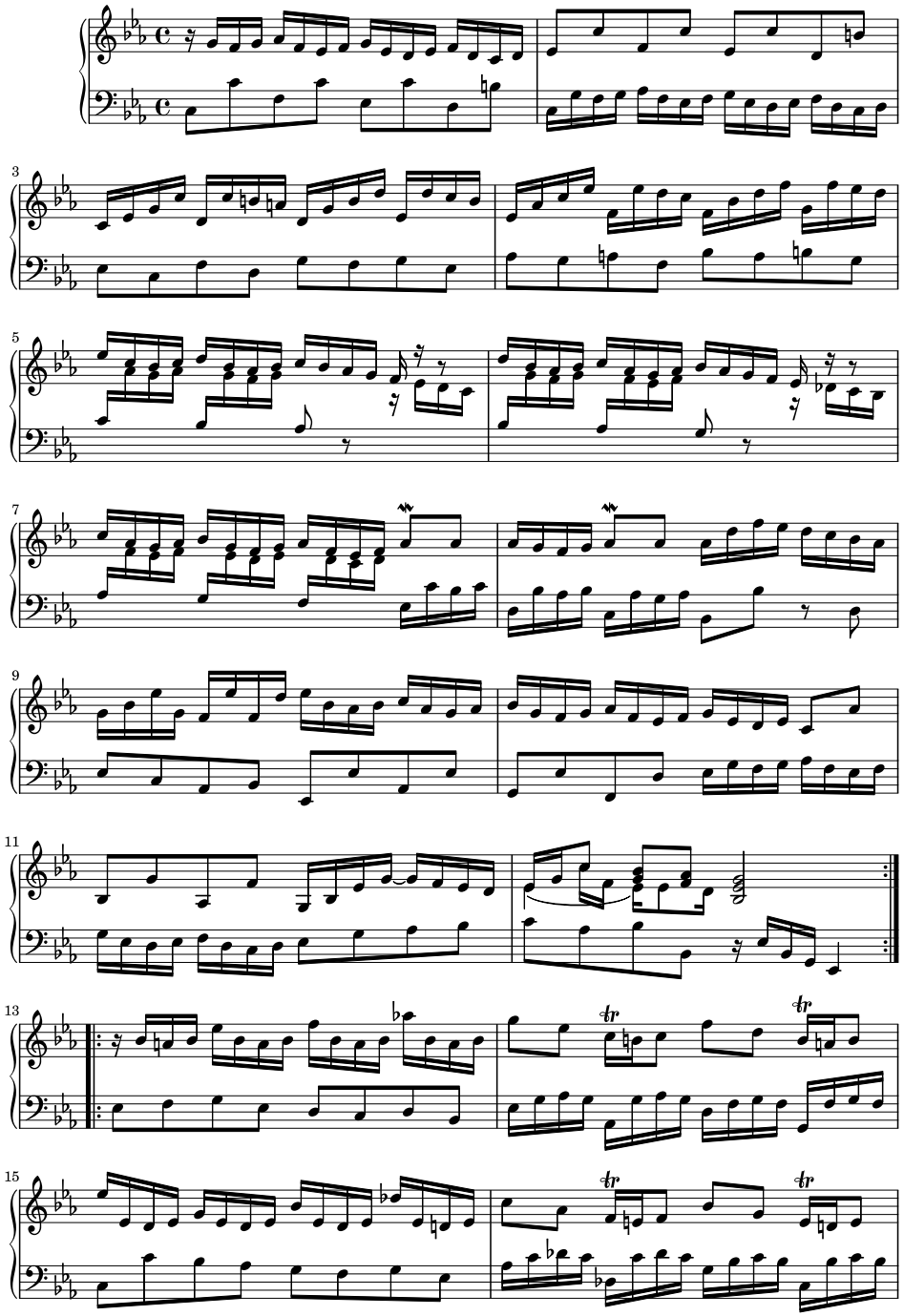
OMNISIA

- OMNISIA is an implementation in Java of the following SIA-based algorithms
  - SIA
  - SIATEC
  - COSIATEC
  - Forth’s algorithm
  - SIATECCompress
  - RecurSIA

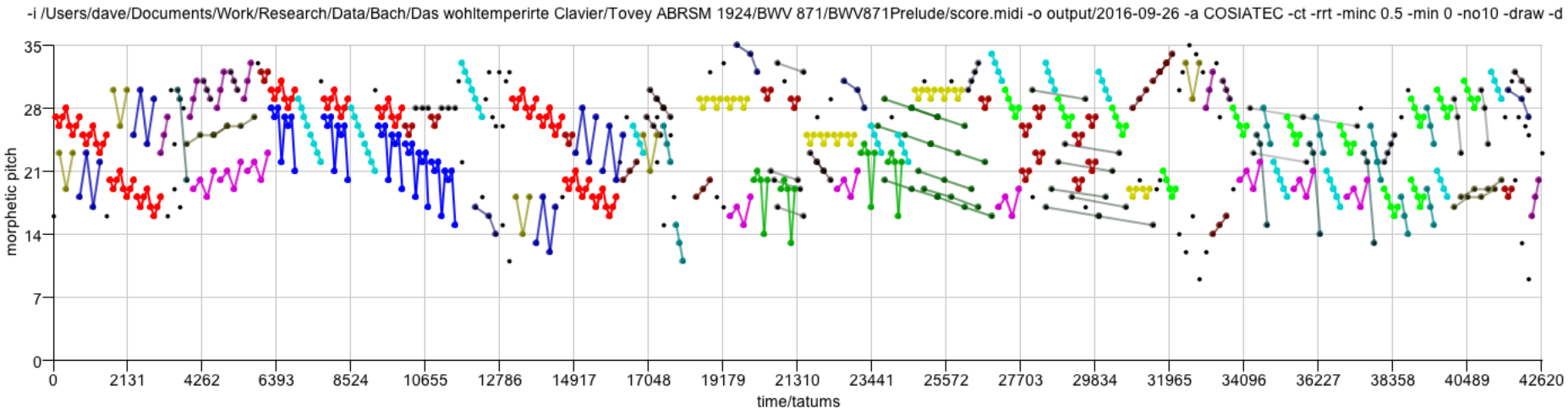
Switches

- -a Basic algorithm to use. Possible values are: SIA, SIATEC, COSIATEC, SIATECCompress, Forth, RecurSIA. Default is COSIATEC.
- -i Path to input file (REQUIRED).
- -o Path to output directory. Default is same directory as input file.
- -d If present, then use morphetic (diatonic) pitch instead of chromatic pitch. If morphetic pitch is not available in the input data (e.g., MIDI format), then input data is pitch-spelt using the PS13s1 algorithm.
- -h Help. If present, then this help screen to be printed. This happens if the program is called with no arguments or if it is unable to determine the values of all necessary parameters from the arguments provided.
- -m If present, generates output in MIREX format.
- -ct If present, uses Collins' compactness trawler, as used in his SIACT and SIARCT-CFP algorithms.
- -cta The variable which Collins et al call 'a'. It is the minimum compactness permitted in the trawled patterns.
- -ctb The variable which Collins et al call 'b'. It is the minimum size of the patterns trawled by the compactness trawler.
- -rsd If present, limits SIA to r superdiagonals, as used in Collins' SIAR algorithm. Number of superdiagonals determined by the -rswitch.
- -r Number of superdiagonals to analyse if limited with -rsd switch. Default value is 1.
- -rrt If present, redundant translators are removed.
- -minc Threshold value for minimum TEC compactness (default is 0.0).
- -min Minimum allowed pattern size. Default is 0.
- -max Maximum allowed pattern size. Default is 0, which means that patterns of all sizes are allowed.
- -no10 If present, channel 10 (drum channel) is removed if input is in MIDI format.
- -draw If present, generates an image file containing a visualization of the analysis.
- -crlow Minimum compression ratio in Forth's algorithm. Default is 0.2.
- -crhi Maximum compression ratio in Forth's algorithm. Default is 1.0.
- -comlow Minimum compactness threshold in Forth's algorithm. Default is 0.2
- -comhi Maximum compactness threshold in Forth's algorithm. Default is 1,0
- -cmin c_min threshold in Forth's algorithm. Default is 15
- -sigmin sigma_min threshold in Forth's algorithm. Default is 0.5.
- -bbcomp If present, use bounding-box compactness in Forth's algorithm instead of within-voice segment compactness.
- -nodate If present, then does not append date to output directory names.
- -bbmode If present, then uses BB mode when generating output in MIREX format.
- -segmode If present, then uses Segment mode when generating output in MIREX format.
- -out If present, overrides -o and prints a single output encoding to the given path.
- -top If present, limits output to top N patterns.
- -recalg If RecurSIA is main algorithm used, then value of this switch determines which basic algorithm is used on each pattern. Possible values are COSIATEC, SIATECCompress or Forth.
- -sortpat When using COSIATEC, getBestTEC sorts TECs with preference given to TECs with larger patterns.

Prelude in C minor from WTCII BWV871

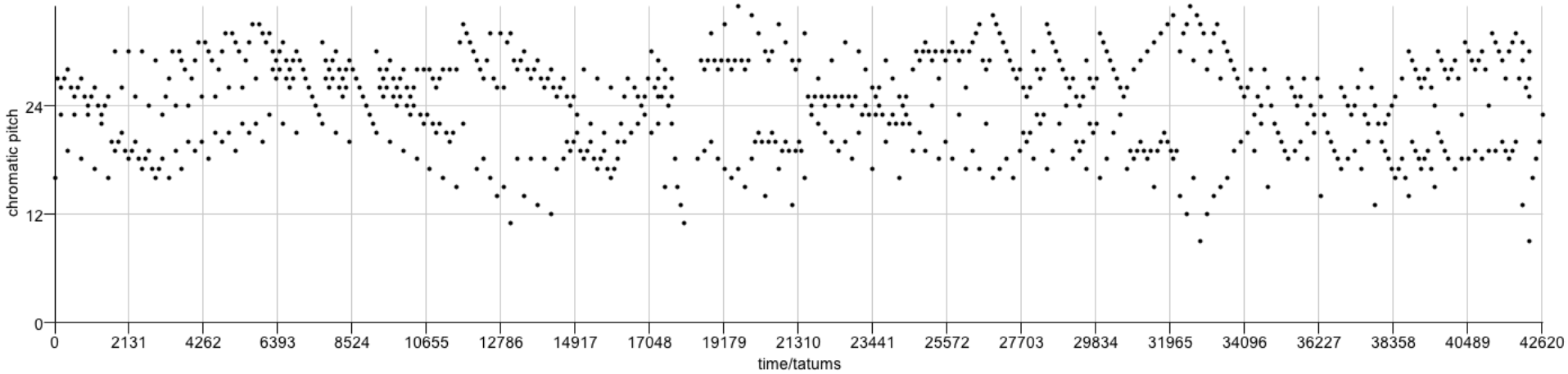


Combining switches

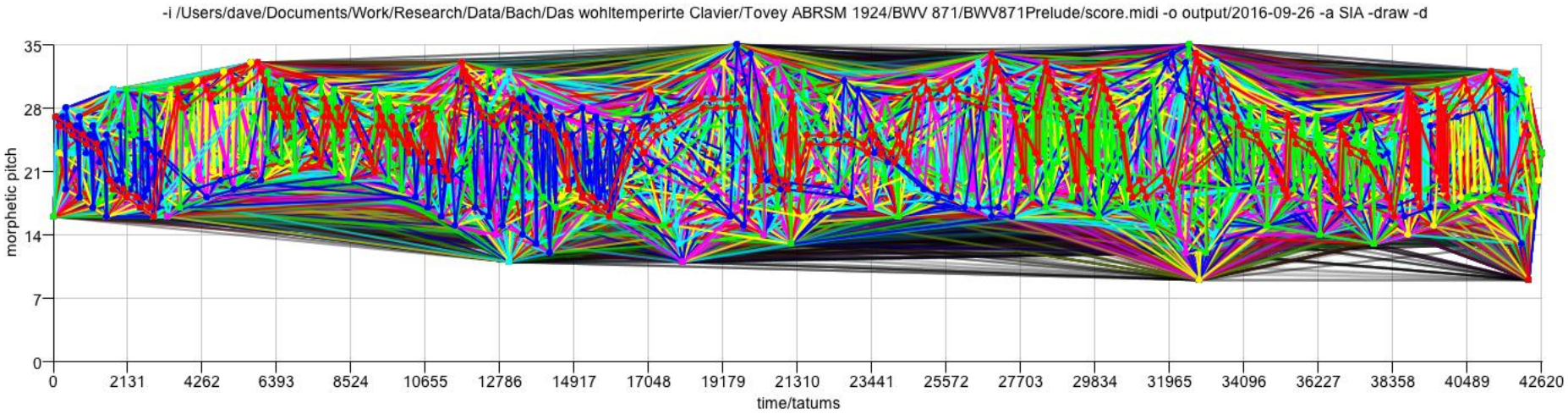


- -a COSIATEC -ct -rrt -minc 0.5 -min 0 -no10 -draw -d
- numberOfNotes 692
- compressionRatio 2.283828382838284
- runningTime 3661
- encodingLength 303
- encodingLengthWithoutResidualPointSet 231
- numberOfResidualPoints 72
- percentageOfResidualPoints 10.404624277456648
- compressionRatioWithoutResidualPointSet 2.683982683982684
- numberOfTECs 26
- isDiatonic true

Input point set (Prelude in C minor from WTC II BWV 871)

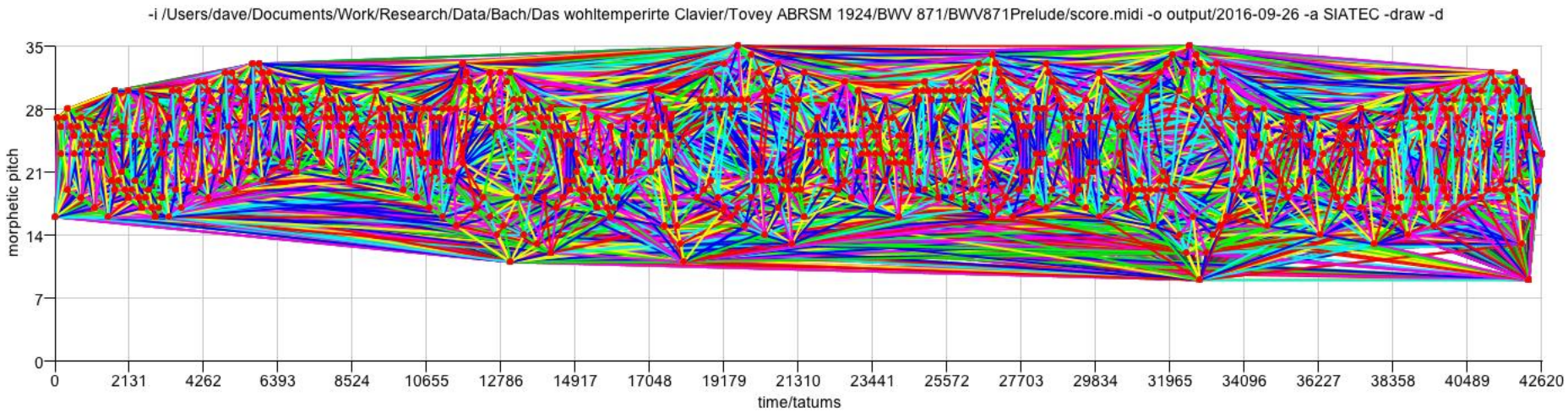


SIA



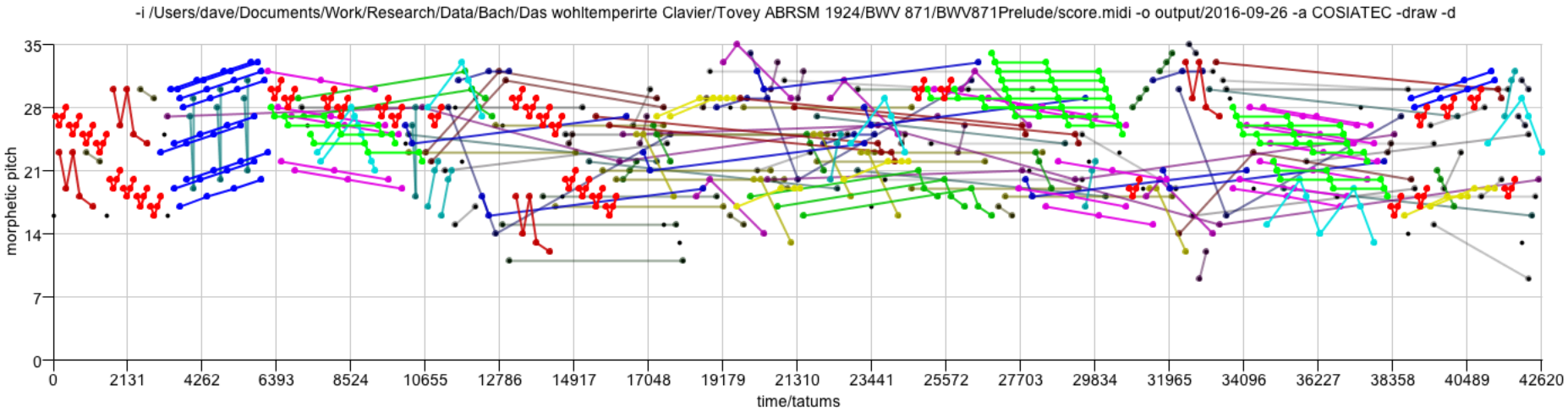
- numberOfNotes 692
- compressionRatio 0.0027371360538567 12
- runningTime 1021
- encodingLength 252819
- numberOfTECs 13947
- isDiatonic true

SIATEC



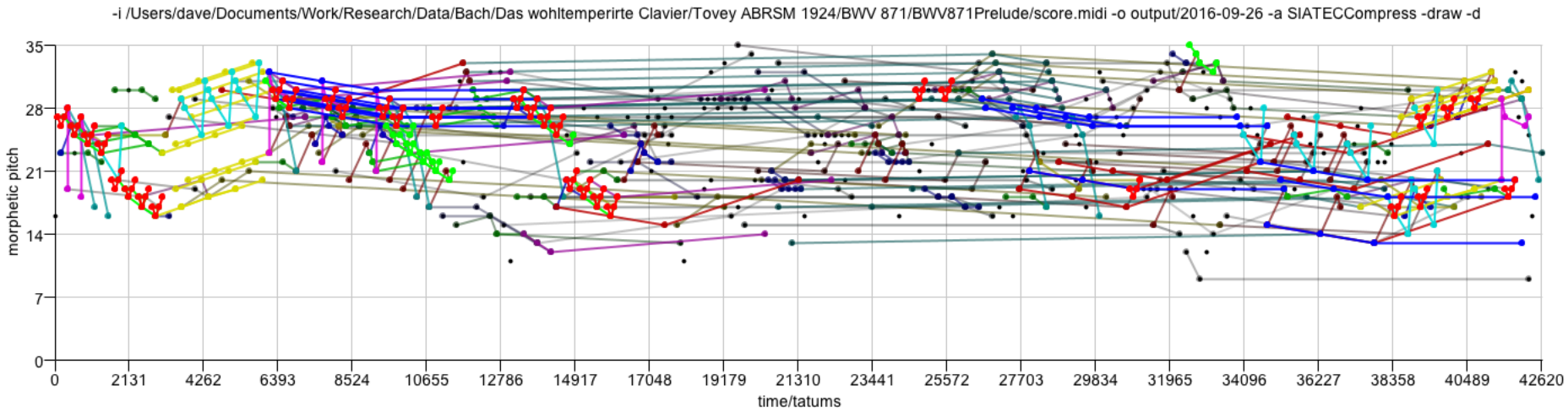
- numberOfNotes 692
- compressionRatio 0.0024844809696656363
- runningTime 3401
- encodingLength 278529
- encodingLengthWithoutResidualPointSet 278529
- numberOfTECs 13079
- isDiatonic true

COSIATEC



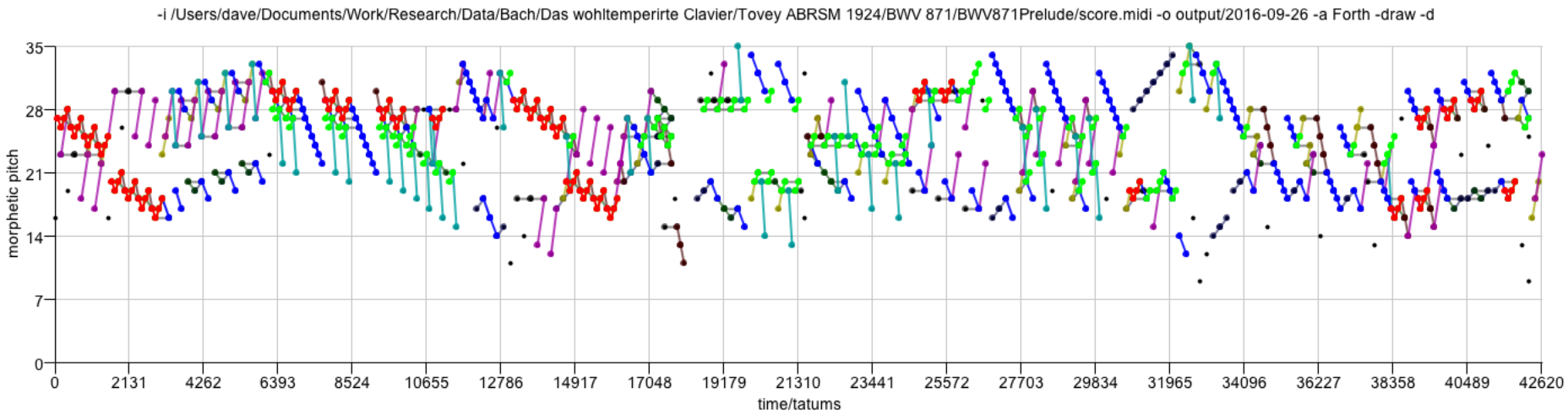
- numberOfNotes 692
- compressionRatio 2.276315789473684
- runningTime 13813
- encodingLength 304
- encodingLengthWithoutResidualPointSet 279
- numberOfResidualPoints 25
- percentageOfResidualPoints 3.61271676300578
- compressionRatioWithoutResidualPointSet 2.390681003584229
- numberOfTECs 37

SIATECCompress



- numberOfNotes 692
- compressionRatio 1.3568627450980393
- runningTime 7271
- encodingLength 510
- encodingLengthWithoutResidualPointSet 422
- numberOfResidualPoints 88
- percentageOfResidualPoints 12.716763005780347
- compressionRatioWithoutResidualPointSet 1.4312796208530805
- numberOfTECs 29
- isDiatonic true

Forth

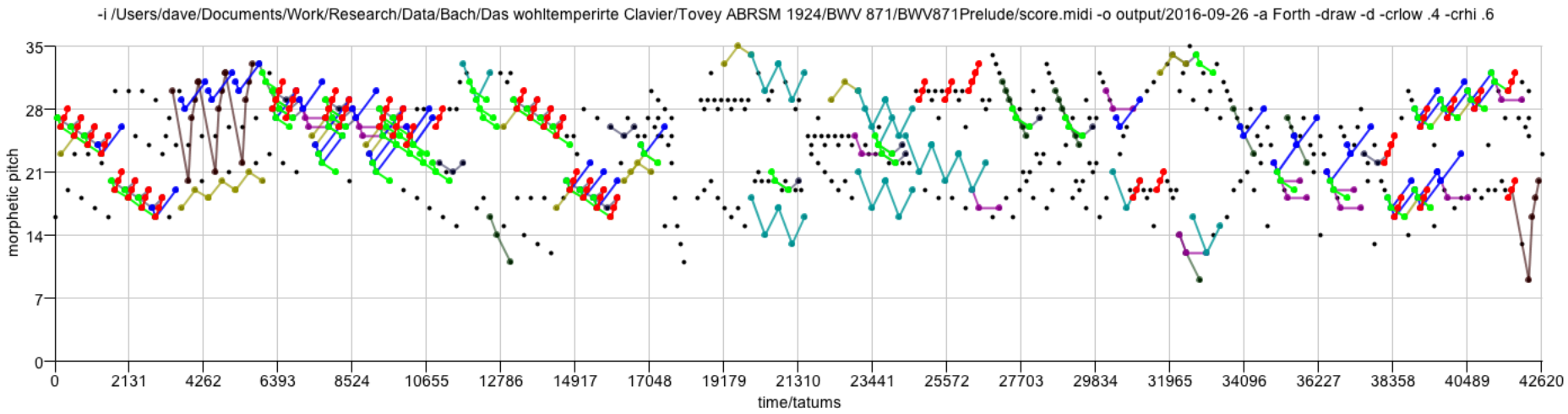


- numberOfNotes 692
- compressionRatio 0.9153439153439153
- runningTime 4842
- encodingLength 756
- encodingLengthWithoutResidualPointSet 726
- numberOfResidualPoints 30
- percentageOfResidualPoints 4.335260115606936
- compressionRatioWithoutResidualPointSet 0.9118457300275482
- numberOfTECs 11
- isDiatonic true

Switches on Forth’s algorithm

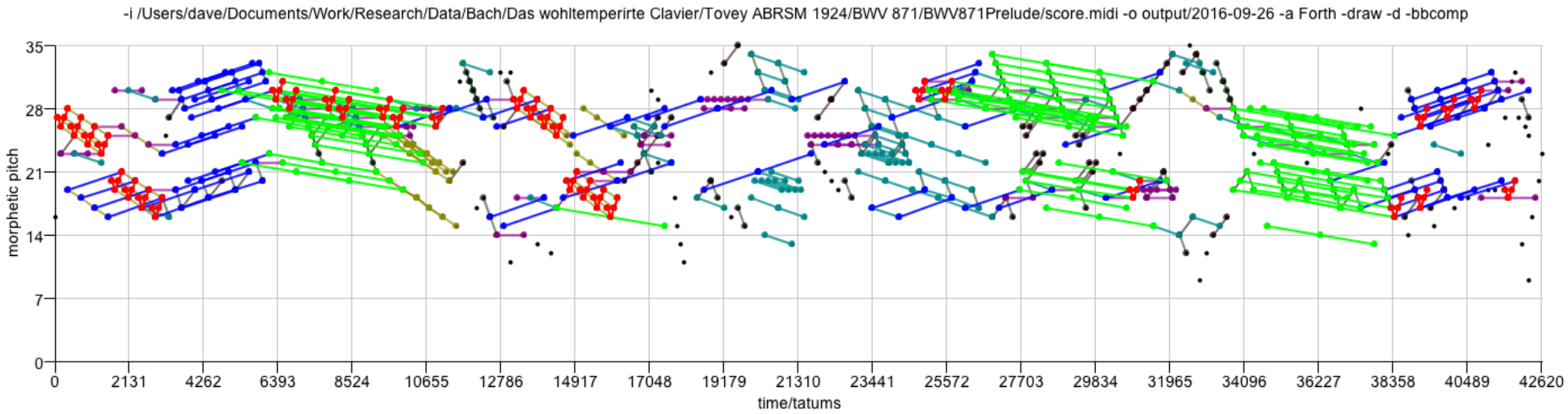
- -crlow Minimum compression ratio in Forth's algorithm. Default is 0.2.
- -crhi Maximum compression ratio in Forth's algorithm. Default is 1.0.
- -comlow Minimum compactness threshold in Forth's algorithm. Default is 0.2
- -comhi Maximum compactness threshold in Forth's algorithm. Default is 1,0
- -cmin c_min threshold in Forth's algorithm. Default is 15
- -sigmin sigma_min threshold in Forth's algorithm. Default is 0.5.
- -bbcomp If present, use bounding-box compactness in Forth's algorithm instead of within-voice segment compactness.

Forth with crlow = 0.4, crhi = 0.6



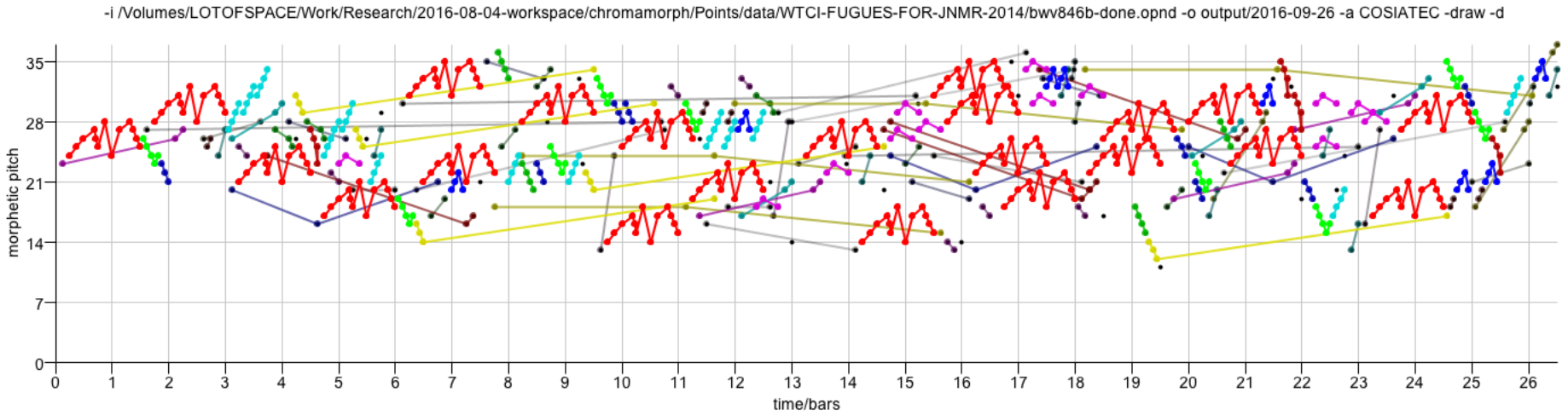
- numberOfNotes 692
- compressionRatio 1.291044776119403
- runningTime 5205
- encodingLength 536
- encodingLengthWithoutResidualPointSet 241
- numberOfResidualPoints 295
- percentageOfResidualPoints 42.630057803468205
- compressionRatioWithoutResidualPointSet 1.6473029045643153
- numberOfTECs 10
- isDiatonic true

Forth with bbcomp



- numberOfNotes 692
- compressionRatio 1.0581039755351682
- runningTime 4571
- encodingLength 654
- encodingLengthWithoutResidualPointSet 607
- numberOfResidualPoints 47
- percentageOfResidualPoints 6.791907514450867
- compressionRatioWithoutResidualPointSet 1.0626029654036244
- numberOfTECs 9
- isDiatonic true

RecurSIA with COSIATEC



- COSIATEC

```
T(P(p(8,24),p(12,25),p(16,26),p(22,27),p(23,26),p(24,25),p(28,28),p(32,24),p(36,27),p(42,28),p(4
T(P(p(50,26),p(52,25),p(54,24),p(56,23),p(58,24)),V(v(0,0),v(144,-7),v(230,-1),v(256,7),v(306,4)
T(P(p(224,20),p(226,21),p(228,22),p(230,20)),V(v(0,0),v(162,7),v(336,12),v(342,12),v(458,10),v(5
T(P(p(98,27),p(100,28),p(102,29),p(104,30)),V(v(0,0),v(8,2),v(16,4),v(54,-3),v(56,-2),v(64,0),v(
T(P(p(160,23),p(164,24),p(172,23)),V(v(0,0),v(236,-5),v(276,-1),v(312,3),v(316,6),v(328,4),v(388
T(P(p(136,31),p(138,30),p(140,29),p(304,34)),V(v(0,0),v(34,-4),v(68,-15),v(164,-9),v(482,-17)))
```

- RecurSIA with COSIATEC

```
T(P(T(P(p(8,24),p(24,25)),V(v(0,0),v(24,0))),T(P(p(16,26),p(22,27)),V(v(0,0),v(6,1),v(20,1))),T(P(p(12,25),p(23,26),p(44,27),
T(P(p(50,26),p(52,25),p(54,24),p(56,23),p(58,24)),V(v(0,0),v(144,-7),v(230,-1),v(256,7),v(306,4),v(594,-3),v(662,-8),v(736,9)
T(P(p(224,20),p(226,21),p(228,22),p(230,20)),V(v(0,0),v(162,7),v(336,12),v(342,12),v(458,10),v(568,0),v(584,1),v(612,13)))
T(P(p(98,27),p(100,28),p(102,29),p(104,30)),V(v(0,0),v(8,2),v(16,4),v(54,-3),v(56,-2),v(64,0),v(80,-6),v(158,-6),v(192,-6),v(
T(P(p(160,23),p(164,24),p(172,23)),V(v(0,0),v(236,-5),v(276,-1),v(312,3),v(316,6),v(328,4),v(388,11),v(392,7),v(420,8),v(552,
```

RecurSIA with COSIATEC

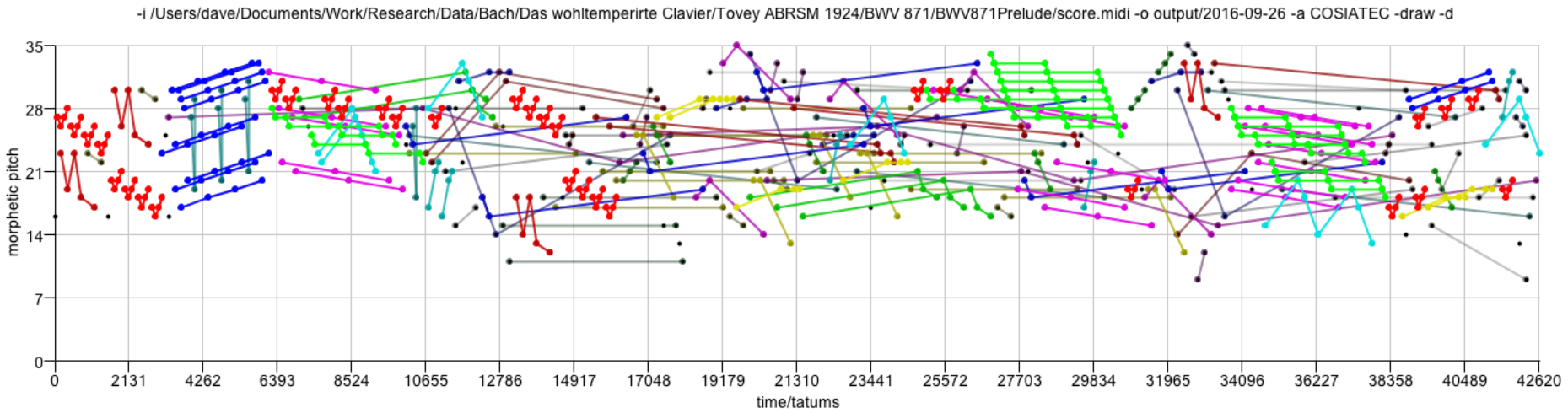
- COSIATEC
- numberOfNotes 729
- compressionRatio 2.904
- runningTime 10026
- encodingLength 251
- encodingLengthWithoutResidualPoint Set 225
- numberOfResidualPoints 26
- percentageOfResidualPoints 3.566
- compressionRatioWithoutResidualPoi ntSet 3.124
- numberOfTECs 30

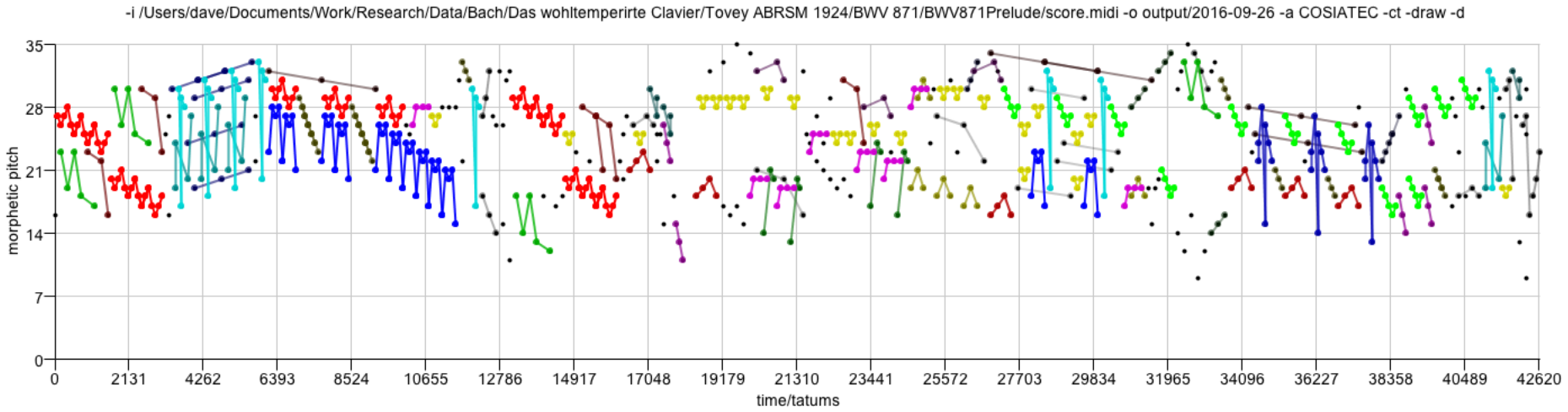
- RecurSIA with COSIATEC
- numberOfNotes 729
- compressionRatio 2.927
- runningTime 9846
- encodingLength 249
- encodingLengthWithoutResidualPoint Set 223
- numberOfResidualPoints 26
- percentageOfResidualPoints 3.566
- compressionRatioWithoutResidualPoi ntSet 3.152
- numberOfTECs 30

Collins et al.’s Compactness Trawler

- -ct If present, uses Collins' compactness trawler, as used in his SIACT and SIARCT-CFP algorithms.
- -cta The variable which Collins et al call 'a'. It is the minimum compactness permitted in the trawled patterns.
- -ctb The variable which Collins et al call 'b'. It is the minimum size of the patterns trawled by the compactness trawler.

COSIATEC with CT

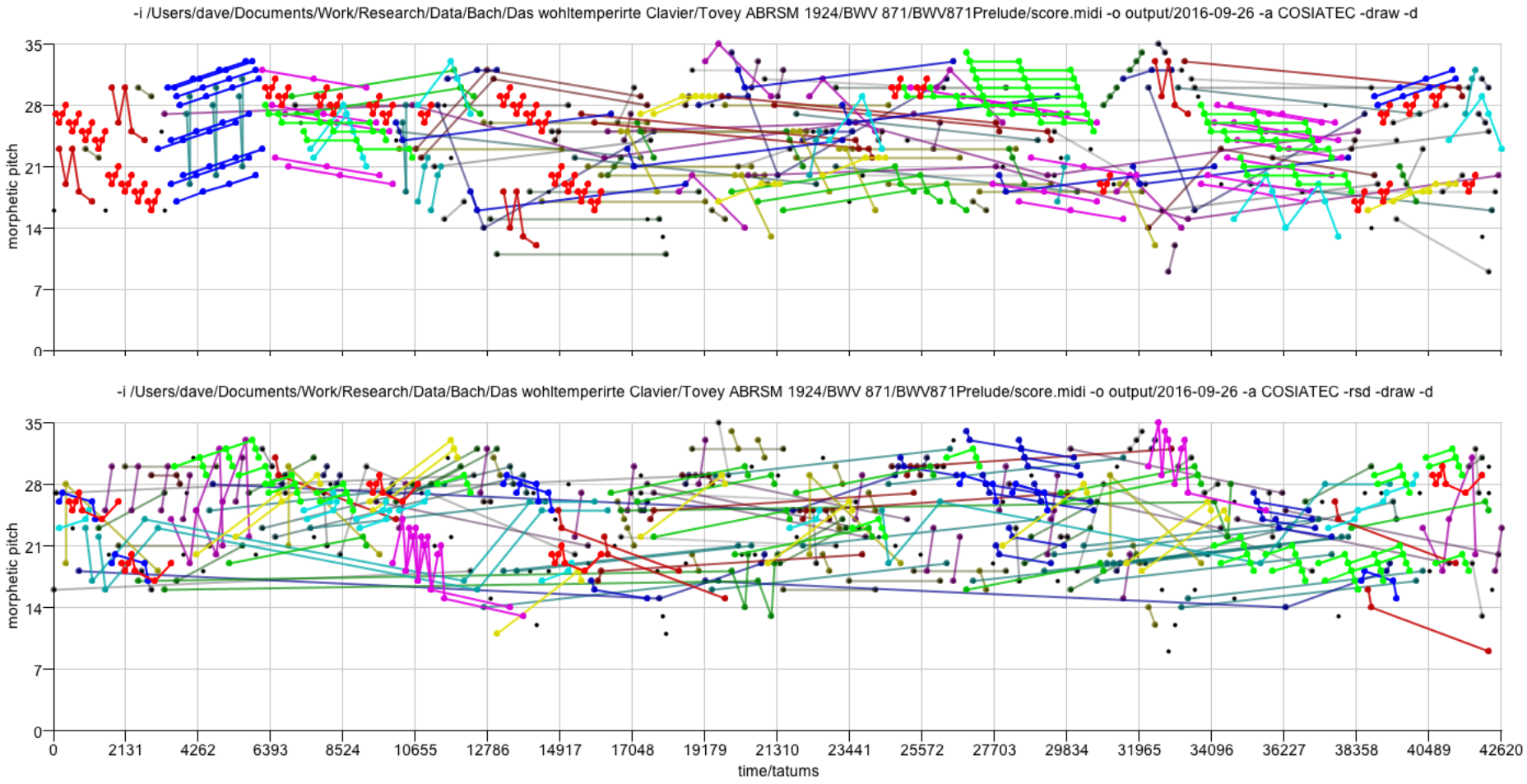




- numberOfNotes 692
- compressionRatio 2.096969696969697
- runningTime 3865
- encodingLength 330
- encodingLengthWithoutResidualPointSet 238
- numberOfResidualPoints 92
- percentageOfResidualPoints 13.294797687861271

Collins’ SIAR algorithm

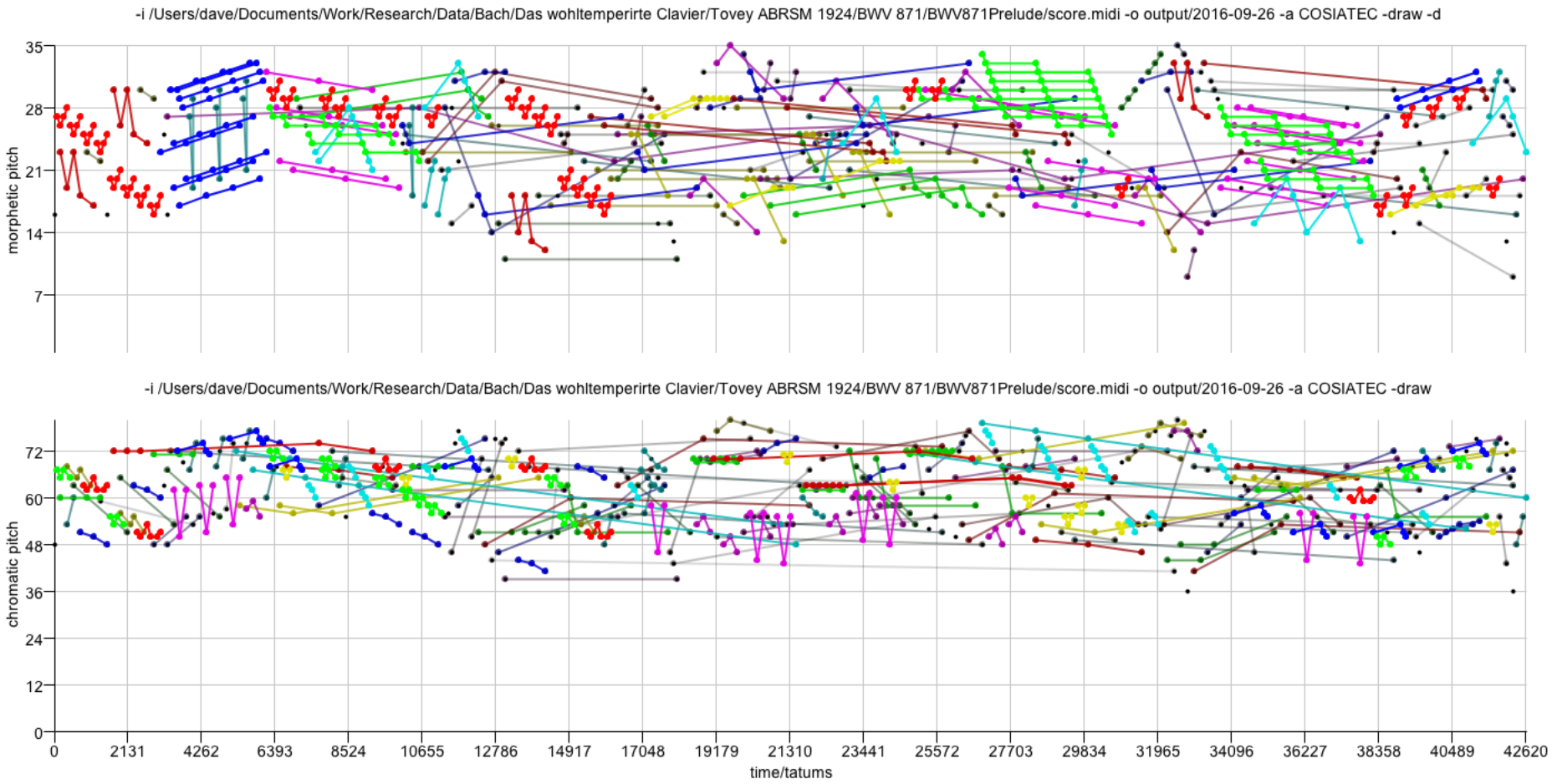
- -rsd If present, limits SIA to r superdiagonals, as used in Collins' SIAR algorithm. Number of superdiagonals determined by the –r switch.
- -r Number of superdiagonals to analyse if limited with -rsd switch. Default value is 1.



- numberOfNotes 692
- compressionRatio 1.9329608938547487
- runningTime 7831
- encodingLength 358
- encodingLengthWithoutResidualPointSet 301
- numberOfResidualPoints 57
- percentageOfResidualPoints 8.236994219653178
- compressionRatioWithoutResidualPointSet 2.1096345514950166
- numberOfTECs 35
- isDiatonic true

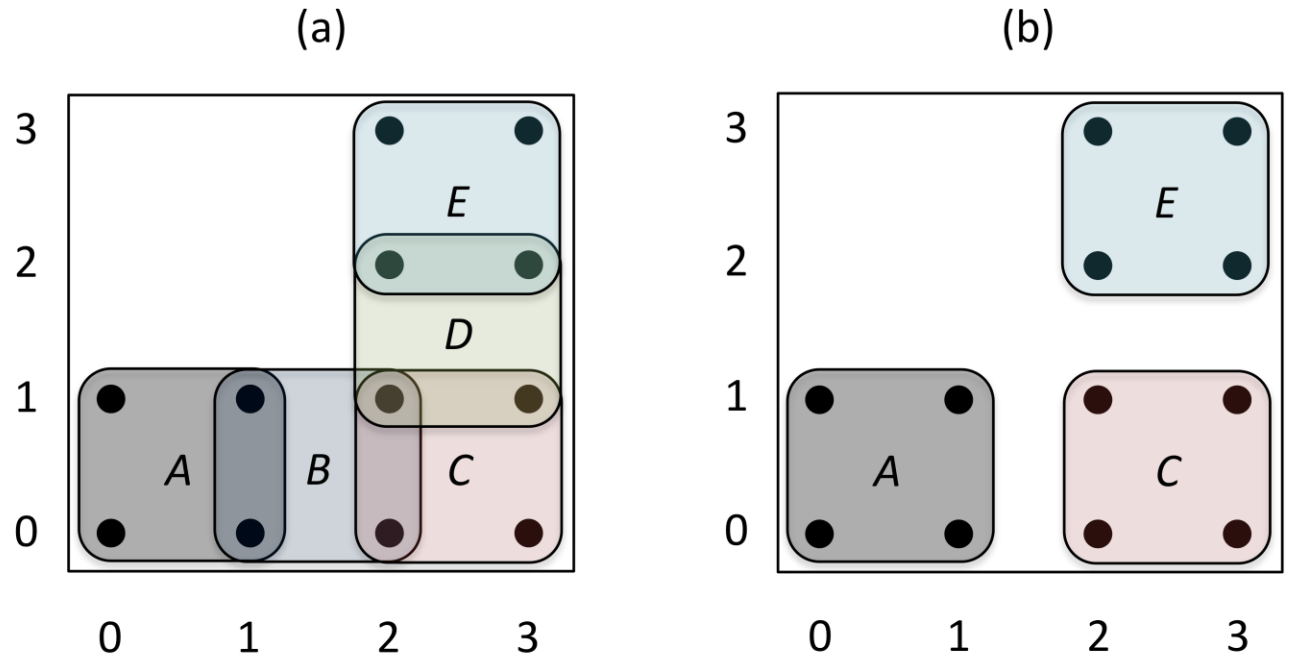
COSIATEC with SIAR

Chromatic or morphetic pitch



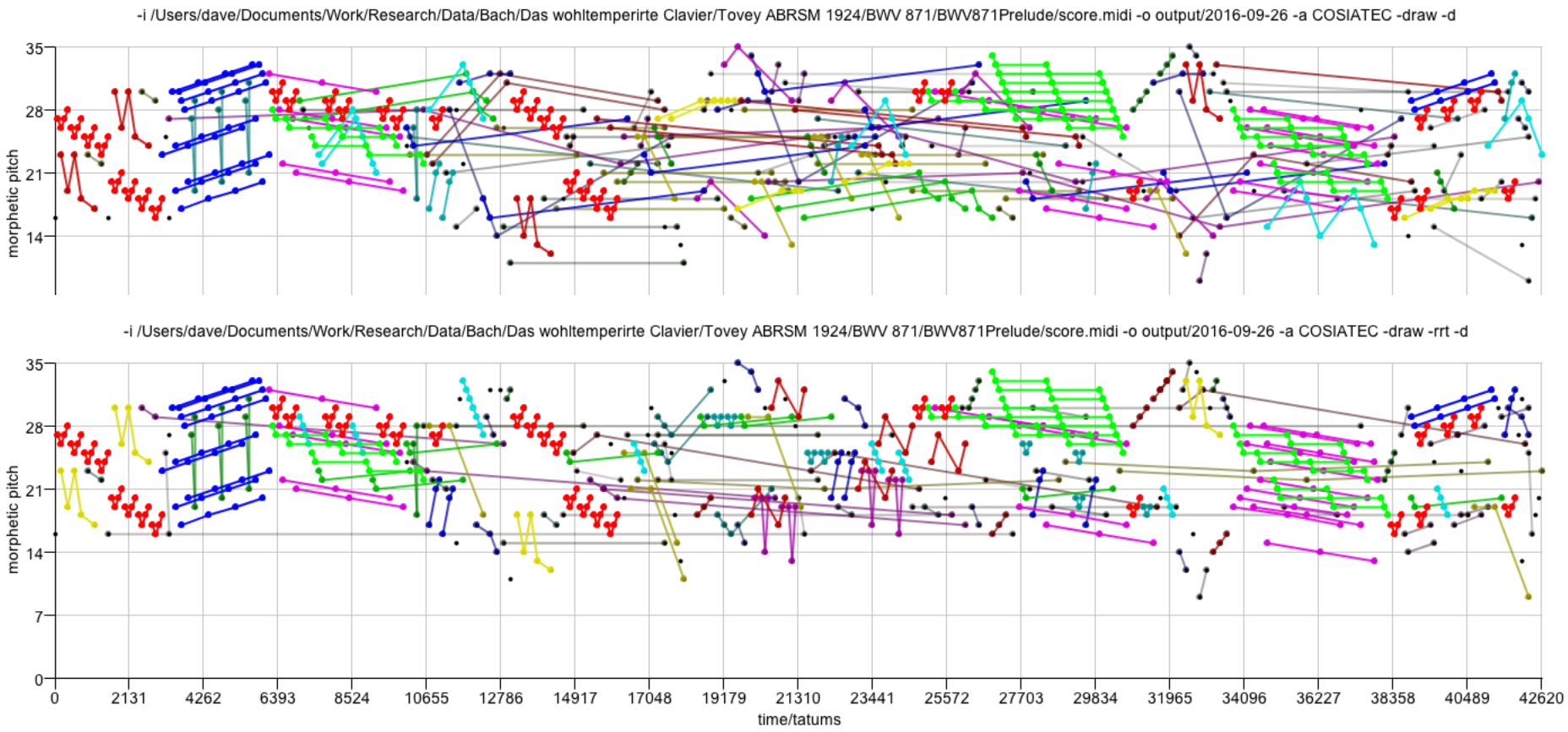
- -d If present, then use morphetic (diatonic) pitch instead of chromatic pitch. If morphetic pitch is not available in the input data (e.g., MIDI format), then input data is pitch-spelt using the PS13s1 algorithm.

Removing redundant translators



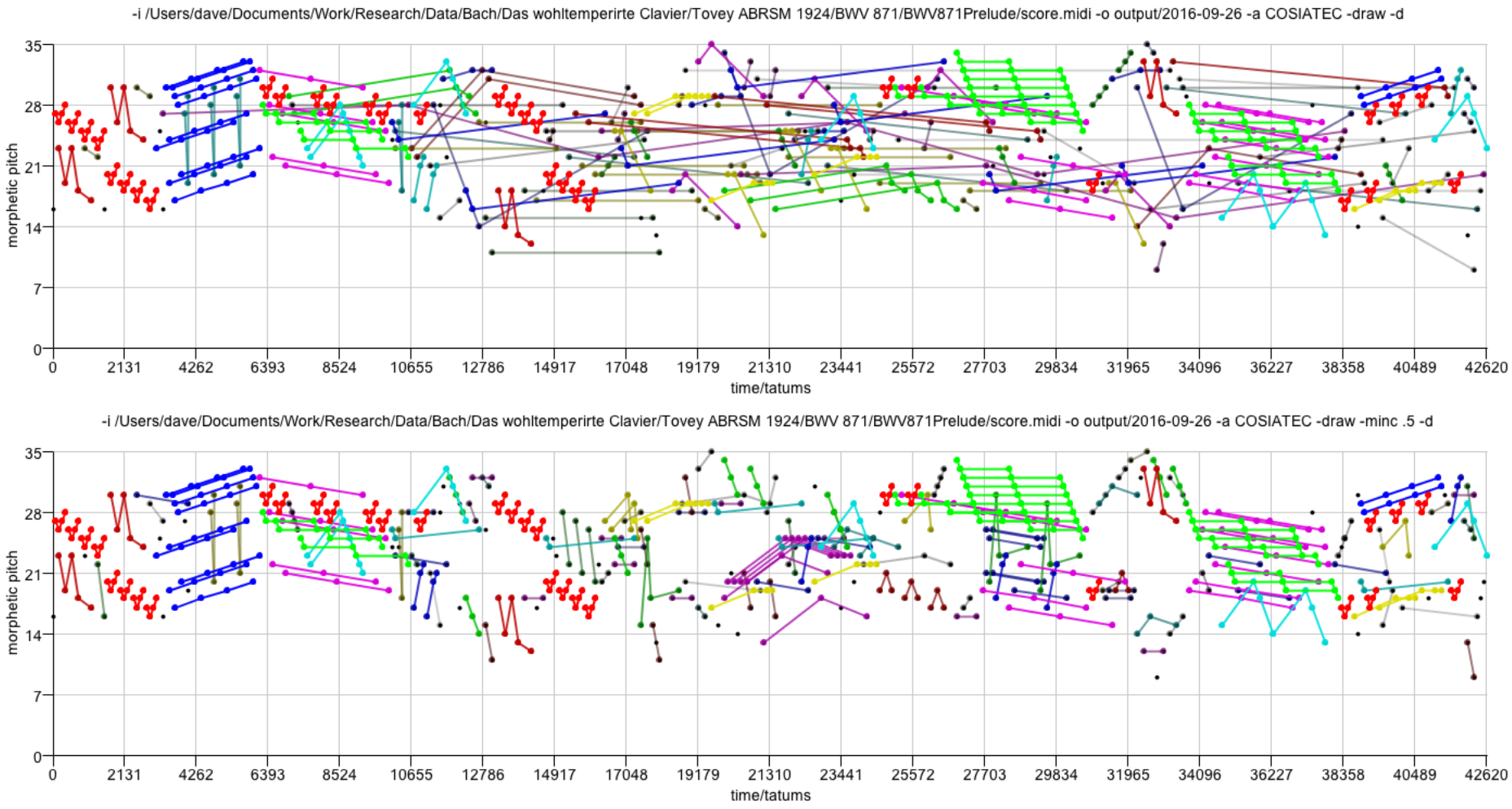
- -rrt If present, redundant translators are removed.

COSIATEC with -rrt



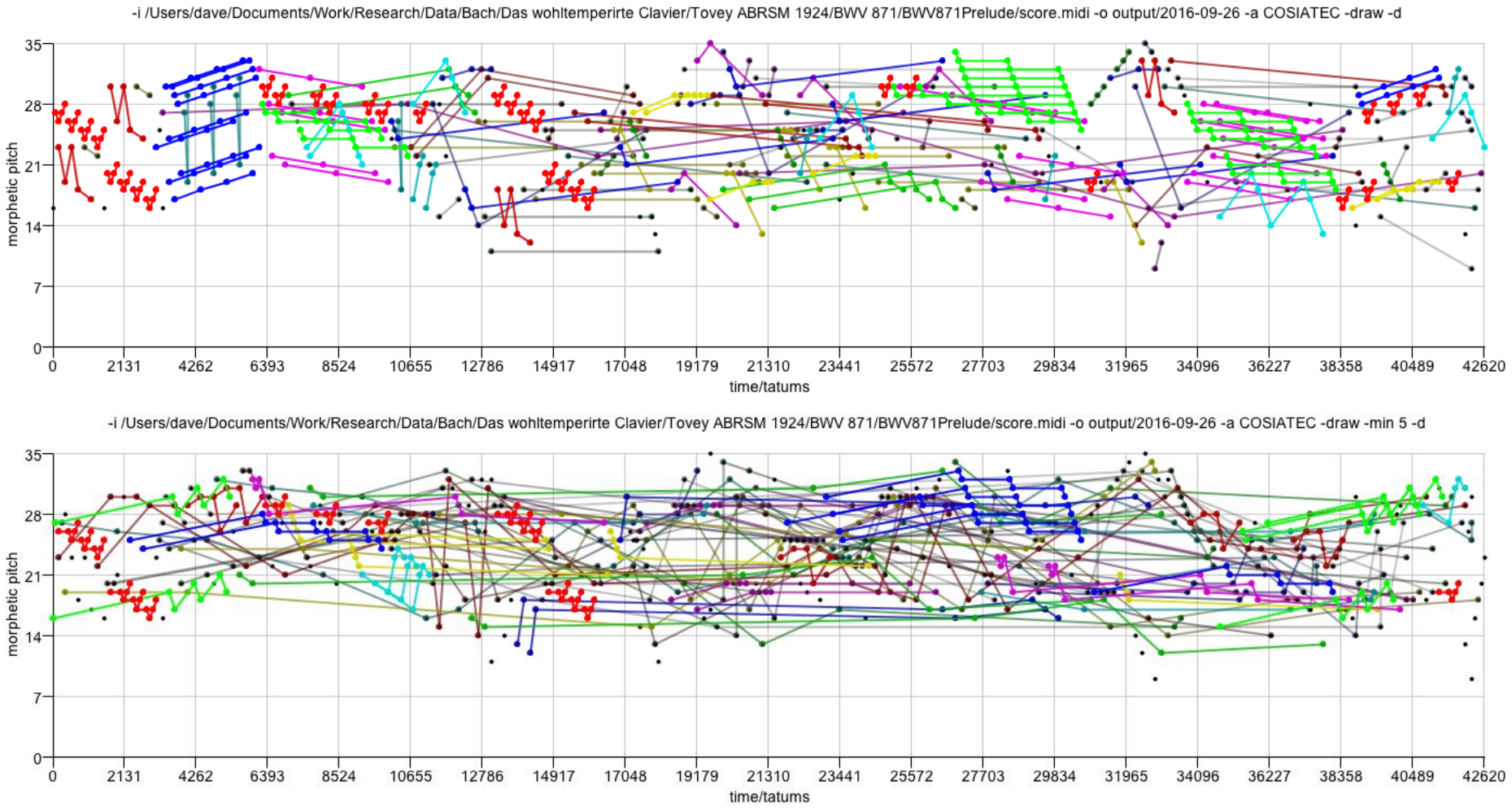
- numberOfNotes 692
- compressionRatio 2.3221476510067114
- runningTime 14443
- encodingLength 298
- encodingLengthWithoutResidualPointSet 265
- numberOfResidualPoints 33
- percentageOfResidualPoints 4.76878612716763
- compressionRatioWithoutResidualPointSet 2.486792452830189
- numberOfTECs 32
- outputFileExtension cos

Minimum compactness



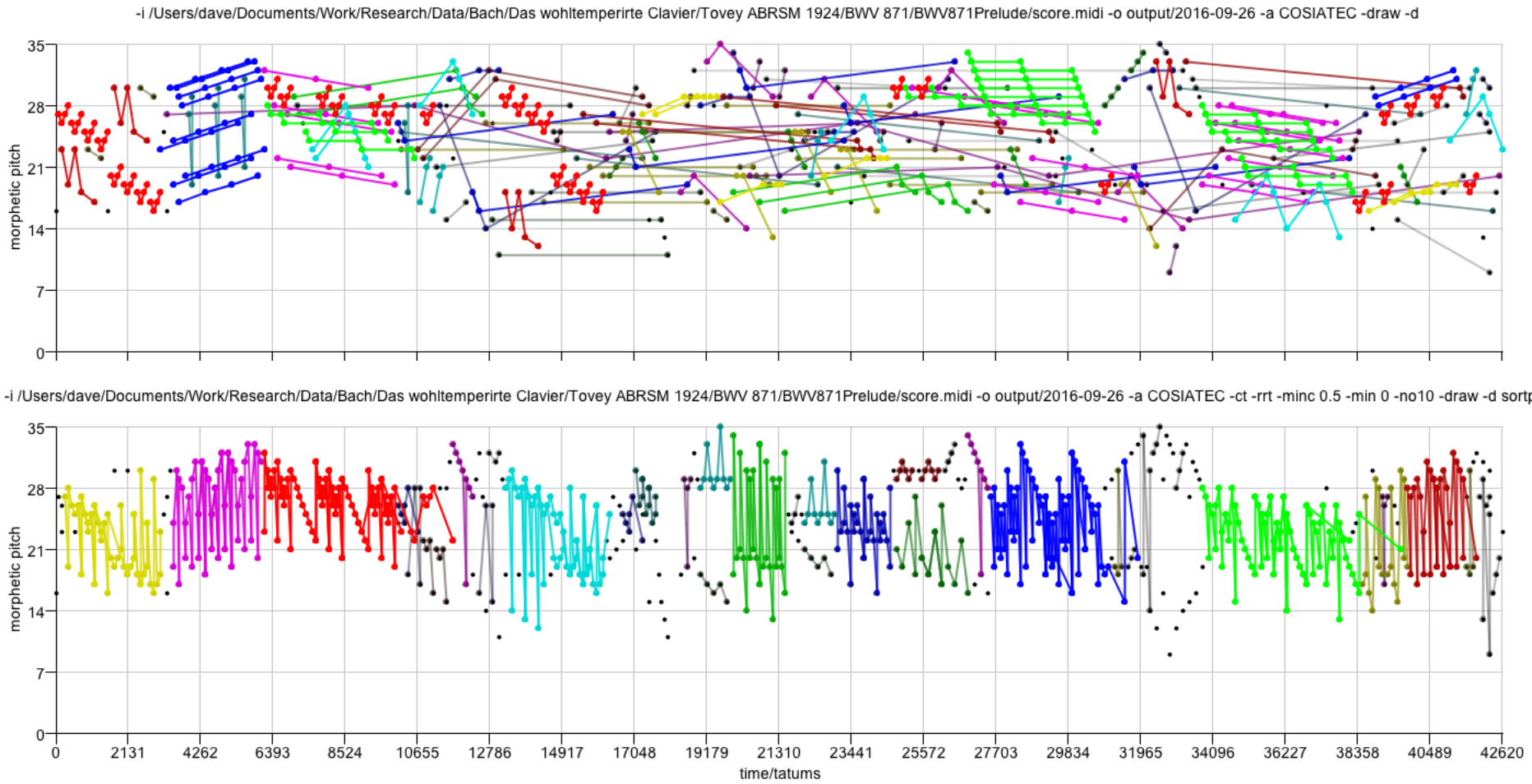
- -minc Threshold value for minimum TEC compactness (default is 0.0).

Controlling pattern size



- -min Minimum allowed pattern size. Default is 0.
- -max Maximum allowed pattern size. Default is 0, which means that patterns of all sizes are allowed.

Sorting TECs with priority given to pattern size



- -sortpat When using COSIATEC, getBestTEC sorts TECs with preference given to TECs with larger patterns.