On minimal change musical morphologies

Michael Winter

Abstract In this paper, we examine a number of minimal change musical morphologies. Each morphology has an analogous representation in mathematics. Our mathematical objects of study are Gray codes, de Bruijn sequences, aperiodic necklaces, disjoint subset pairs, and multiset permutations with musically motivated constraints that result in several open problems.

Key words: minimal change morphologies, minimalism, de Bruijn sequences, Gray codes, generative music, Lyndon words, aperiodic necklaces, disjoint subsets, multiset permutations, graph theory, constraint programming.

1 Introduction and Preliminaries

Several different minimalist trends exist in art and music. In this paper, we focus on minimal change musical morphologies where the word "minimal" primarily denotes "minimal change" between adjacent elements in a given morphology. Each morphology has an analogous representation in mathematics. Our mathematical objects of study are Gray codes, de Bruijn sequences, aperiodic necklaces, disjoint subset pairs, and multiset permutations with musically motivated constraints that result in several open problems.

First, we discuss a taxonomy of "morphological constraints" used to contextualize the definition of each morphology. Next, we review previous research in order to show the genesis of our current formalization. Section 2 focusses on examples of minimal change musical morphologies and the open mathematical problems that result from musically motivated constraints on the analogous mathematical representations. We conclude with an overview of the open problems and suggestions for further research.

Michael Winter
Independent Researcher, Los Angeles, CA USA, e-mail: `mwinter@unboundedpress.org`

1.1 Scope and complexity of morphological constraints

In this paper, the morphologies and their analogous mathematical representations are defined by a subset of four types of morphological constraints: Combinatorial Constraints (CC), Local Morphological Constraints (LMC), Global Morphological Constraints (GMC), and Optimal Global Morphological Constraints (OGMC). This taxonomy shows a hierarchy of scope (from elements of a morphology to its large-scale form) and computational complexity (from easy to difficult to compute). The complexity at each hierarchical level is based on the satisfaction of that constraint in conjunction with all lower-level constraints.

1.1.1 Combinatorial Constraints (CC)

A CC is a constraint that defines all the elements of a morphology. As no further constraints are imposed (between adjacent/pairwise elements or among sets and sequences of elements), computing a set of elements defined by a CC is generally easy/efficient. For example, in Section 2.1.1, we discuss a morphology where the CC is that each element must be a subset of a set of n sounds. The set of all subsets (the powerset) can be represented mathematically by all binary words of size n where each bit position corresponds to one of the sounds.

1.1.2 Local Morphological Constraints (LMC)

A LMC is a constraint at the next-higher hierarchical level; i.e., between adjacent elements in a morphology. A morphology that satisfies a LMC with no higher-level constraints is generally easy to compute. However, the computation is likely to take more time and resources than generating a set of elements using a CC alone unless the known fastest algorithm that satisfies the CC also satisfies the LMC. Continuing with the example that will be discussed in Section 2.1.1, the LMC is that from subset to subset, only one sound can be added or removed; or framed mathematically, only one bit can flip from word to word.

1.1.3 Global Morphological Constraints (GMC)

A GMC constrains a statistical property of the morphology. For most of the morphologies detailed below, the GMC is that each element defined by the CC occurs only once; e.g., any given subset of sounds or binary word is never repeated. Unless the known fastest algorithm that satisfies all lower-level constraints also satisfies the GMC, finding a morphology that satisfies a GMC is harder than just a LMC and/or CC. Often, the difficulty increases exponentially with respect to the number of elements defined by the CC. For example, some of our morphologies can be generated by finding Hamiltonian paths (where each vertex is traversed only once) in representative graphs. Finding Hamiltonian paths is known to be NP-complete for arbitrary graphs as brute-force search times typically explode exponentially with the size of the graph. Section 1.1.5 further explicates this taxonomy's relation to graph theory.

1.1.4 Optimal Global Morphological Constraints (OGMC)

An OGMC constrains sets or sequences of three or more elements in the morphology (as opposed to just adjacent elements as with the LMC) thus defining a subset of morphologies satisfying all lower-level constraints. In most of the examples in Section 2, the OGMCs are satisfied such that the order of elements minimizes or maximizes some feature/characteristic of the morphology (hence the use of the word "optimal"); e.g., codes that have maximally uniform, long run-lengths and sequences where the running sum is minimized. Depending on the OGMC, finding a satisfactory morphology can be extremely hard with complexity on the order of solving difficult games and puzzles.

1.1.5 Relation to graph theory and Constraint Programming (CP)

Two methods used to generate some of the morphologies detailed in this paper have cogent relations and near-analogs to the above taxonomy: finding paths in representative graphs and searches using Constraint Programming (CP; see [20]).

A graph with vertices defined by CCs and edges induced by LMCs is essentially a structural representation of the morphology. The graph can be used to generate the morphology by finding a path that satisfies any defined GMCs and OGMCs. Generating morphologies using this technique illustrates an important, if not fundamental, link between morphology (or shape) and structure.

In CP, a solver searches for a solution that satisfies a programmed set of "binary" and/or "global" constraints applied over a "domain" by optimizing a set of "objectives" (minimizing or maximizing a set of functions). A domain in CP is equivalent to a set of elements defined by a CC. Binary constraints are similar to LMCs as they both involve only two variables. Global constraints and GMCs relate because they both involve more than two variables and are constraints at a hierarchical level higher than binary constraints and LMCs, respectfully. However, global constraints also relate to OGMCs as both constrain sequences or sets of elements. Thus, a global constraint could be considered as something between a GMC and an OGMC. Notwithstanding the connection between global constraints and OGMCs, CP objectives clearly relate to OGMCs because of the optimization process.

To summarize, Table 1 shows the morphological constraint taxonomy in relation to hierarchical scope, difficulty, and analogs to graph theory and CP.

Table 1: Summary of morphological constraint taxonomy.

Constraint	Hierarchical Scope	Difficulty	Graph Theory	CP
CC	elemental	easiest	vertices	domain
LMC	pairwise	easy	edges	binary constraint
GMC	global	hard	path	global constraint
OGMC	global optima	hardest	optimal path	objective

1.2 Precedence of musical thinking with respect to morphological constraints

The two pieces described in this section exemplify a compositional process where musical morphologies are defined by morphological constraints. Both have well-defined CCs, LMCs, and GMCs, but do not have OGMCs. While several of the pieces described later have OGMCs, the concept was theoretically formalized only recently for the purposes of this paper.

The mathematical implications of the following two examples are investigated more thoroughly in "Chordal and timbral morphologies using Hamiltonian cycles" [1], where the authors show the conditions that admit a Hamiltonian path or cycle[1] in representative graphs derived and generalized from the pieces. Section 2 is a focussed extension of the ideas in the aforementioned article: focussed in that we look exclusively at minimal change morphologies and extended in that we also look at examples with OGMCs. "Chordal and timbral morphologies using Hamiltonian cycles" also provides a historical context connecting this work to the work of James Tenney and Larry Polansky among others (specifically, Tenney's definition of form as shape and structure in *Meta+Hodos* [26] and Polansky's definitions of "morphological metrics" [17]). These writings along with the author's dissertation "Structural Metrics: an epistemology" [29] further illustrate the genesis of compositional thinking detailed throughout this paper.

1.2.1 Maximally smooth chordal cycles

In a "maximally smooth cycle", as defined by Richard Cohn [7], one part moves by a semitone or whole step while the other parts remain on the same pitch. Tom Johnson's piece *Trio* (2005)[2] is a variant of this idea that exemplifies well-defined morphological constraints. In *Trio*, each pitch in a four-octave chromatic set is represented by a number 0 to 48 where middle C equals 24. The musical morphology enumerates through all three-note chords satisfying the CC that the numbers repre-

[1] A Hamiltonian cycle is basically the same as a Hamiltonian path except it returns to the start vertex.

[2] Score to this piece available at `http://www.editions75.com` (accessed January 2015).

senting the pitches within each chord are distinct integer partitions without repetitions of 72. The LMC is that from chord to chord, one pitch must remain the same while the other pitches move by a semitone in contrary motion. The GMC is that each chord occurs only once. The morphology is analogous to a Hamiltonian path (which by definition satisfies the GMC) on a graph where the vertices represent the chords defined by the CC and edges are induced by the LMC. Figure 1 shows the first system of the score to *Trio*.

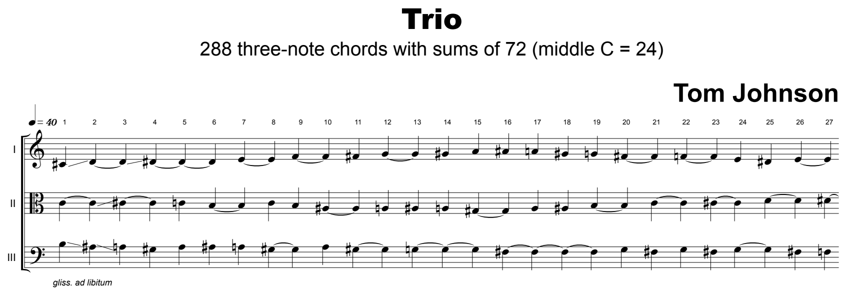

Fig. 1: Excerpt from score of *Trio*.

1.2.2 Maximal change timbral morphologies

In the author's piece *maximum change* (2010),[3] the elements are all timbral possibilities of a chord with 4 pitches using 4 instruments assuming that each instrument can play up to all of the pitches at once (the CC), which framed mathematically are all 4-tuples where the position in the tuple represents the pitch to which an instrument, represented by a number at that position, is assigned. The LMC is that from chord to chord, each pitch is played by a different instrument; or mathematically, from tuple to tuple, each position is assigned a different number. That is, the same chord is repeated but the mapping of instruments to pitches changes as maximally as possible. The GMC is that each timbral possibility occurs only once. The morphology is analogous to a Hamiltonian path in a graph where the vertices represent the timbral possibilities (or mappings of instruments to pitches) defined by the CC and edges are induced by the LMC.[4] Figure 2 shows the first 10 measures of the score to *maximum change*.

[3] Score to all the author's pieces available at `http://www.unboundedpress.org` (accessed January 2015). Unless otherwise specified, all works discussed are that of the author.

[4] This is essentially the opposite of the types of morphologies discussed in this paper. However, in [1], the problem and corresponding graph are generalized such that the number of elements that stay the same from tuple to tuple is specified. *maximum change* is a specific instance where the number of elements that stay the same from tuple to tuple is 0.

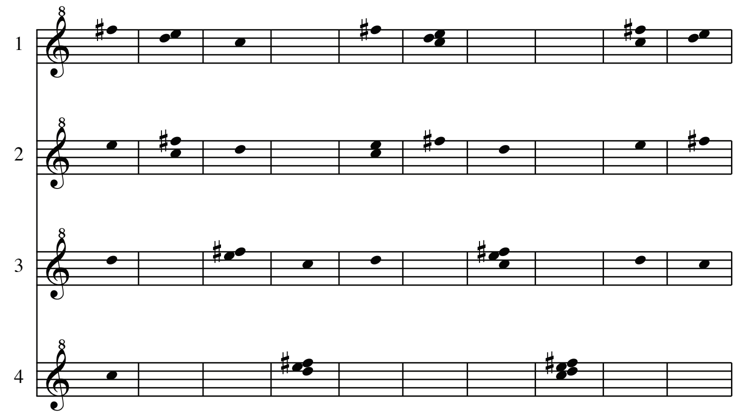

Fig. 2: Excerpt from score of *maximum change*.

2 Minimal Change Musical Morphologies: Applications and Resulting Mathematical Problems

2.1 Gray codes

A Gray code (after Frank Gray [11]) is an enumeration of all binary words of a given length n (the CC) such that only one bit changes from word to word (the LMC) and each word occurs only once (the GMC). We reserve a discussion of OGMCs for the following subsection. For a comprehensive overview of Gray codes, see Carla Savage's "A survey of combinatorial Gray codes" [21].

2.1.1 Maximally balanced, maximally uniform long-run Gray codes

The musical composition *gray codes* (2009) is an exploration of all subsets of a set of sounds such that the overall sound changes as minimally/gradually as possible over time. Each instrument (or sound) follows one bit position in a Gray code. An instrument is sounding when 'on' (or 1) and not when 'off' (or 0). The score gives the following description of an OGMC desired to generate a version of the piece.

> "Ideally, a particular type of Gray code is desired to achieve this effect [of minimal change]. That is, a Gray code where the standard deviation of all run-lengths plus the standard deviation of bit flips across the positions is as close to 0 as possible."

The score then includes a version for orchestra with an 8-bit Gray code. Every subset of instruments from 8 groups—flutes, oboes, clarinets, bassoons, horns, vibraphones, strings I (violins and violas), and strings II (cellos and basses)—sounds together once at some point in the piece. A realization is played exclusively on one pitch with a gradually changing overall timbre.

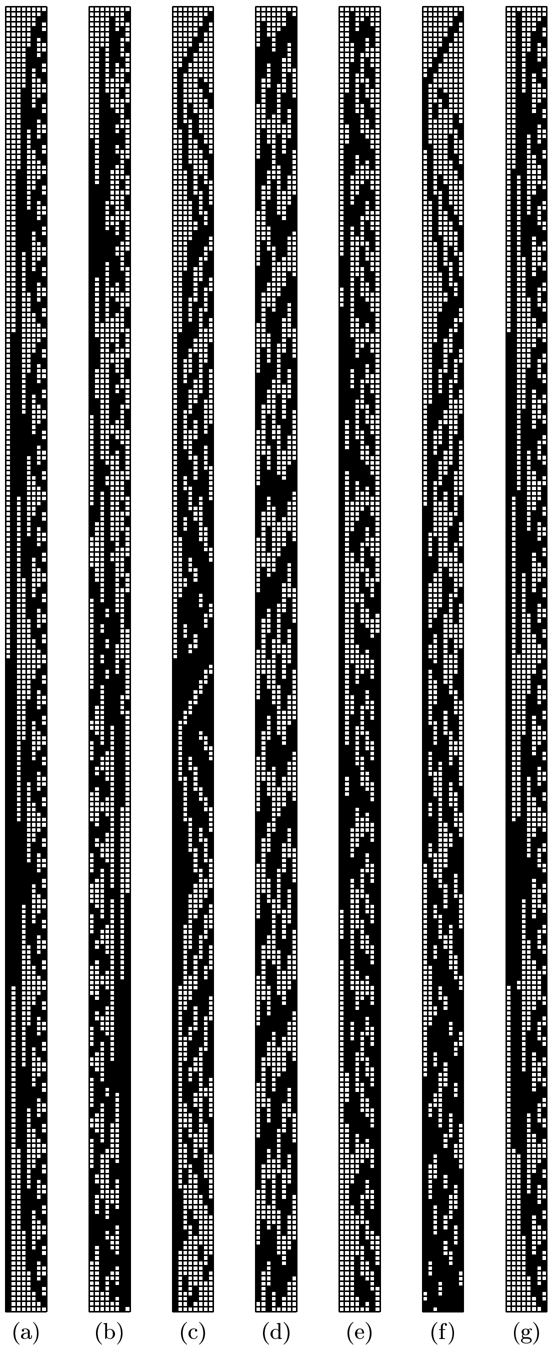

Fig. 3: Examples of 8-bit Gray codes from Donald Knuth's *The Art of Computer Programming*. a) standard; b) balanced; c) complimentary; d) long-run; e) nonlocal; f) monotonic; g) trend-free.

While Gray codes are defined by the LMC that only one bit changes from word to word, standard Gray codes are not at all balanced (in that each bit position in the enumeration has similarly many bit flips as all others). The musical consequence of using an unbalanced code is that some sounds will come in and out more frequently than others. Also, run-lengths may vary tremendously in standard Gray codes. For examples of both these factors, see Figure 3a as compared to Figure 3b and/or Figure 3d from Donald Knuth's *The Art of Computer Programming* [13]. The OGMC given in the score aims to ensure that each part changes as infrequently as all others by assuming that balanced Gray codes generally have uniformly long run-lengths.

Knuth's long-run Gray code (Figure 3d) is the one used for the orchestral version of the piece *gray codes*. The canonical transition sequence for Knuth's code is given in Figure 4. Each number represents the bit position where the bit flip occurs in each successive binary word. For the orchestral realization of *gray codes*, 0, 1, 2, 3, 4, 5, 6, 7 represent clarinets, strings I, flutes, vibraphones, oboes, strings II, bassoons, and horns, respectively.

```
1062351742501635207145263150273514620517325016452371052631542705
1362051742531605237145260153270514623517025316452071352601542735
1062351742501635207145263150273514620517325016452371052631542705
1362051742531605237145260153270514623517025316452071352601542735
```

Fig. 4: The canonical transition sequence for Knuth's long-run Gray code.

Another possible Gray code that could be used to generate the piece is called a Beckett-Gray code (after the playwright Samuel Beckett). Beckett defined this particular type of Gray code for his work *Quad* (1981), where he wanted all combinations of performers to be on stage at some point throughout the work such that the one who has been on stage the longest will always be the next to exit. Mathematically speaking, the OGMC of a Beckett-Gray code is that the position with the current longest 'on' bit run will always be the next to flip 'off'. By definition, a Beckett-Gray code should be quite balanced and have reasonably uniform, long run-lengths.

It turns out that a 4-bit Beckett-Gray code does not exist, which is why Beckett was unable to implement his original idea and altered it in order to finish the piece. Recently, an 8-bit Beckett-Gray code was found by Brett Stevens, et al. [24]. Shortly after, a fast algorithm to generate Beckett-Gray codes was defined by Joe Sawada, et al. [22]. The canonical transition sequence for the 8-bit code presented in [24] is given in Figure 5.

Several open questions arise from the need of a Gray code that is highly balanced and has uniformly long run-lengths such as how to define and encode the OGMCs. It is unclear if the OGMC given in the score of the piece *gray codes* is adequate as it relies on the assumptions that balance will result in uniformly long run-lengths and that minimizing the standard deviation of the number of bit flips across the positions will balance the code. The difficulty of the optimization problem is compounded by

```
0123456070121324356576071021353462670153741236256701731426206570
1342146560573102464537571020435376140736304642737035640271327505
4121027564150240365425013602541615604312576032572043157624321760
4520417516354767035647570625437242132624161523417514367143164314
```

Fig. 5: The canonical transition sequence for the 8-bit Beckett-Gray code found by Stevens, et al.

the fact that there are three potential optima: maximal uniformity of run-lengths, maximality of run-lengths, and balance.[5] How these optima might relate or conflict both perceptually in the resulting sound and with respect to modelling the problem warrants further investigation.

2.2 *de Bruijn Sequences*

Next, we examine de Bruijn sequences (after Nicolaas Govert de Bruijn [4]). Formally defined, a de Bruijn sequence $B(k,n)$ is a cyclic sequence of a given alphabet A of size k in which every word of length n in A appears uninterrupted only once. Essentially, using a de Bruijn sequence is the fastest way to brute force hack a combination lock[6] with combination size n because the last $n-1$ symbols of a word in the sequence will always overlap with the first $n-1$ symbols of the next word.

The morphological constraints of a de Bruijn sequence are nicely illustrated by a particular type of directed graph referred to as a de Bruijn graph. In a de Bruijn graph, the vertices are all words of length n from a given alphabet A of size k (the CC) and two vertices are connected by a directed edge if the last $n-1$ symbols of the out-vertex overlap with the first $n-1$ symbols of the in-vertex (the LMC; for an example, see Figure 6). The sequence itself can be constructed by finding a Hamiltonian cycle (the GMC) on such a graph.

There are several known algorithms to generate de Bruijn sequences. Notably, Harold Fredricksen and James Maiorana have defined an algorithm that generates a lexicographic least de Bruijn sequence by concatenating the lexicographic sequence of Lyndon words of length divisible by n [10]. A Lyndon word is a string that is smaller in lexicographic order than all its rotations. Not only are Lyndon words useful in efficient generation of de Bruijn Sequences, they are also representatives of aperiodic necklaces; the topic of Section 2.3.

[5] In CP, this is called a multi-objective problem (see [19, 12, 8]). For example, one could weight and sum the objectives to prioritize conflicting optima.

[6] This is assuming that one does not need to reset anything after entering each combination.

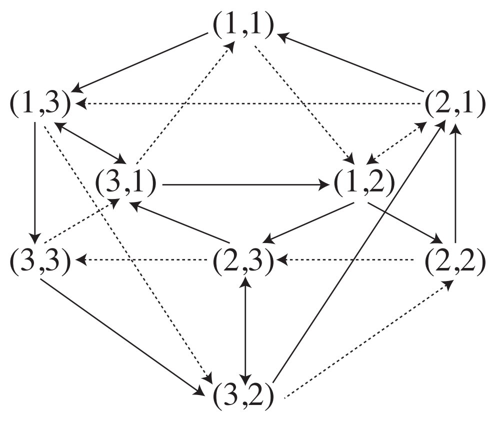

Fig. 6: $B(3,2)$ with Hamiltonian cycle (indicated by dashed lines).

2.2.1 Spatial de Bruijn sequences

In the piece *room and seams* (2008), 4 groups of performers are located in a room as far as possible from all other groups. The piece enumerates all spatial sequences of size 4 in the shortest morphology possible. The representative de Bruijn sequence has both an alphabet size and word length of 4. Each symbol in the alphabet represents a location in space articulated by the sounding of a tone from the group at that location. As no OGMCs were defined, the de Bruijn sequence used to generate the piece was computed with a program written by Hakan Kjellerstrand that implements the algorithm by Fredricksen and Maiorana mentioned above.[7] An excerpt from the score is provided in Figure 7. Despite lacking an OGMC, this example still demonstrates one of many ways even a standard de Bruijn sequence can be of interest musically. Our next example extends the standard de Bruijn sequence with an OGMC necessitated by a musical practicality.

2.2.2 Space-limited contour de Bruijn sequences

The piece *dissection and field* (2008) enumerates all melodic contours of size 6 in the shortest morphology possible. The representative de Bruijn sequence has an alphabet of three numbers $\{-1, 0, 1\}$ that indicate direction in a pitch morphology: down, same, up, respectively.

[7] The program for this de Bruijn sequence generator written in Java is available at `http://www.hakank.org/comb/deBruijn.java` (accessed January, 2015). It is a port of Frank Ruskey's C and Pascal versions, which are available upon request from the Combinatorial Object Server at `http://theory.cs.uvic.ca/inf/neck/NecklaceInfo.html` (accessed January, 2015).

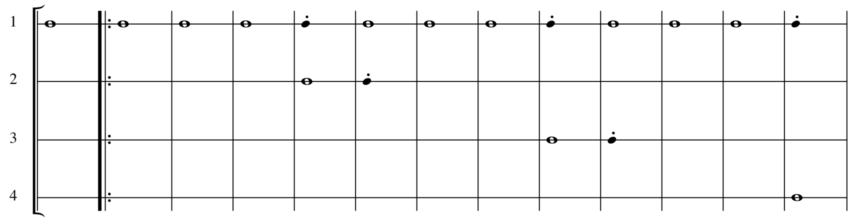

Fig. 7: Excerpt from score of *room and seams*.

In order for pitches not to get extremely high or low, the range of the sequence's running sum is constrained. We denote a de Bruijn sequence constructed from an alphabet of integers that sum to 0 with an OGMC that constrains the range of the running sum as "space-limited". Due to this added constraint, the algorithm used for *room and seams* was not suitable. The final contour morphology for *dissection and field* was found by Kjellerstrand. After unsuccessfully trying to find a solution by brute-force searches for Hamiltonian cycles in a de Bruijn graph, Kjellerstrand turned to CP in order to limit the solution space for greater efficiency by defining an objective that minimized the difference between the extremal values in the running sum while satisfying the fundamental de Bruijn sequence constraints.[8] The final pitch morphologies[9] for *dissection and field* were created from several composed melodic fragments such that, when pieced together, ultimately conformed to the contour sequence as a whole. That is, the contour sequence was reconstructed from the melodic fragments. Both the sequence and an excerpt from the score are provided in Figures 8 and 9, respectively.

While Kjellerstrand found a solution satisfactory for the creation of *dissection and field*, it remains an open question whether or not it is optimal. That is, whether it is the de Bruijn sequence $B(3,6)$ with alphabet $A = \{-1,0,1\}$ where the differ-

[8] The program that generated the final solution for *dissection and field* is available at `http://www.hakank.org/minizinc/debruijn_space_limited.mzn` (accessed January, 2015). It was written in a CP language called MiniZinc available at `http://www.minizinc.org/` (accessed January, 2015). Kjellerstand has several other implementations using various CP languages to find traditional de Bruijn sequences with the running sum constraint relaxed at `http://hakank.org/common_cp_models/#debruijn` (accessed January, 2015). Also, while MiniZinc is considered a general high- or medium-level CP language, there is also a music specific CP language written by Torsten Anders called Strasheela available at `http://strasheela.sourceforge.net` (accessed January, 2015). Anders and others have produced interesting results modelling music theories using CP (see [2, 3, 27]).

[9] The piece integrates the space-limited de Bruijn sequence with other formal concerns. Two morphologies were constructed from the contour sequence; one for each of two groups. That is, the groups play the same contour but on different notes (with one group always higher than the other). In the score (see Figure 9), the notes and rests (the latter of which were arbitrarily inserted using the caret symbol) have numbers and tick marks that indicate general durations (which were also arbitrarily/intuitively assigned). Each performer plays independently of the others, which blurs the sequence to some extent. Also, one of the performers from the first group departs from the sequence for a significant portion of the piece to sustain a high-pitched tone.

```
+ + x − − + + − − x − − + x x − x + + + x + − − − + −
− − + x x + + x x x + − x x − x x − − − + + − x − + +
+ − x + − + − + + − + − x − x x x x x x + − − + x x x
x − x x + x − + + + x + + x + + − − x x + − − − x − −
x x + + − + x + x − + x − + − − x x − x + x x x − x −
x − x x + + x + − x x x x + + − x − x − + − − + + + +
− − x + + + + + + x x − x x x + − + − − x + x x − + −
+ + x − − − x x + x x + x + x x − − x x − − + − + − −
+ x − + x + + + x − x + − x + − x − + − + − + − x + +
x + x x x x x − + − x x x + x x − x − − x + − x x + x
+ − − x + − − + + x + − + + − x x + − + x − − − + − +
+ + x − + − + x − x x − + x + − − + − + x + − + − x x
− − x + x + x − − x x x − − − − x − + − x + x x + − x
− x + − + x x + − + + + + x + x + + x − − x − x − − −
+ x + x + x + − + x + + − x + x − + − x − − + + + − +
x x x − − + + x − + x x − − + x − x + + − + − + x x −
+ x − − + − x + − − x x x + + x − + + x − x x x − + +
x x − − − x − x + + x − x − − + − − + − − x − + + x +
+ + + − + + − − − x x − + − − − x + − + + x x + + + x
x x x + x − − − − + x − − x + + x x + − − x − x x − x
− + + − x + + + − x x − + + − − + x + − x − − − x + +
− − + − x x + + + − − + + − + x − + + − + + + − − − −
− − x + x − x x + − x + x + + − + + x + x − x + x + −
x + + − x x x − x + − − − − + − x − + x + x x + + − −
− + + x x x − + x x + x − x − + x − x − x + x − − + x
+ + x x − + + + + + − x − − x − − − − − + + + x x + x
x x + x + + + − + − − − − x x x x − − x − + x x x + +
+ + x − −
```

Fig. 8: de Bruijn sequence used for *dissection and field* where −1, 0, and 1 map to −, x, and +, respectively.

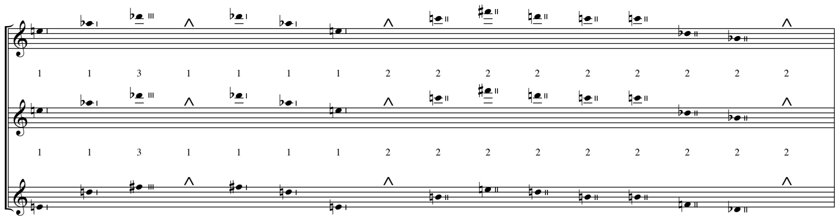

Fig. 9: Excerpt from score of *dissection and field*.

ence of the extremal values of the running sum (which is 8 for the sequence used in *dissection and field*) is smaller than all other such de Bruijn sequences. As an extension, the general case of lower bounds on the range of extremal values in the running sum of optimal solutions for space-limited de Bruijn sequences also remains open.

2.3 Aperiodic necklaces

The piece *necklaces* (2014) is a minimal change ordering of unique picking patterns using a set of plucked strings where the resultant pitch of each string is the same as all others. Each pattern in the piece is analogous to a representative of an aperiodic necklace, which is an equivalency class on aperiodic strings under rotation and permutation of the symbols. For example, $(0,1,0,2,3,3)$ is equivalent to $(1,0,2,3,3,0)$ under rotation and $(2,3,2,1,0,0)$ under symbol permutation. Therefore, they are all representatives of the same aperiodic necklace. In the last section, we discussed how de Bruijn Sequences can be generated efficiently by concatenating Lyndon words ordered lexicographically. The morphology for *necklaces* is related because each aperiodic necklace contains one Lyndon word which means Lyndon words form representatives of aperiodic necklaces.

To demonstrate how equivalencies of aperiodic necklaces relate to the uniqueness of picking patterns, lets map the example above to a traditionally tuned soprano ukulele: $\mathrm{IV} \to \mathrm{G}_4, \mathrm{III} \to \mathrm{C}_4, \mathrm{II} \to \mathrm{E}_4, \mathrm{I} \to \mathrm{A}_4$. $(0,1,0,2,3,3)$ could represent $(\mathrm{I}_0, \mathrm{II}_5, \mathrm{I}_0, \mathrm{III}_9, \mathrm{IV}_2, \mathrm{IV}_2)$ which all sound A_4.[10] Under permutation of symbols (the strings of the ukulele in this case), $(\mathrm{III}_9, \mathrm{IV}_2, \mathrm{III}_9, \mathrm{II}_5, \mathrm{I}_0, \mathrm{I}_0)$ results in the same pattern because all the strings still sound A_4 and the rhythm remains unchanged. In the piece, each pattern may be repeated several times successively. This is why equivalency under rotation is also necessary. When repeated successively, the rhythmic character of the repeated pattern remains the same regardless of which rotational representative is used.

necklaces enumerates through all unique picking patterns of length 6 or less using 4 strings (the CC) such that from pattern to pattern one element is added, removed, or changed (the LMC). Not considering the immediate repetitions, each pattern occurs only once (the GMC) except for the patterns of length 1 and 2 (explained below). The OGMC is that the morphology submit to an "arc" form where the lengths of the patterns generally increase then decrease. An excerpt of the score is provided in Figure 10.

[10] In this example, the letters indicate note names with subscripts that indicate octaves whereas the Roman numerals indicate string numbers of the ukulele with subscripts that indicate frets; 0 being the open string. It is also understood that this fingering, even on the ukulele, is very difficult. We use it just for demonstrative purposes. All the performances to date (as of January 2015), have been with a ukulele where the open strings are tuned to the same pitch. Minor variations in tuning, string tension, and string gauge contribute to the overall sound of the piece even though conceptually, the strings are assumed to be equivalent.

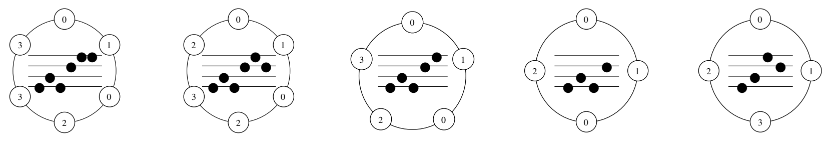

Fig. 10: Excerpt from score of *necklaces*. Each cell indicates a picking pattern given by a tablature where each line represents one of 4 plucked strings and the horizontal axis shows order. The ring around a cell is the necklace representation where the numbers, starting from the number centered above the tablature and moving clockwise, correspond to the strings of the picking pattern.

A solution that satisfies the morphological constraints outlined above can be generated by finding a Hamiltonian path on a graph where the vertices are representatives of the aperiodic necklaces and edges are induced by the LMC. Note that the graph can change substantially based on which representatives are chosen. By definition, adjacent necklaces cannot differ in length by more than one because of the LMC. Clearly the graph cannot submit a Hamiltonian cycle since the trivial necklace, (0) and its equivalencies, can only connect to one other necklace: that of length 2, $(0,1)$ and its equivalencies. However, the graph does submit a Hamiltonian cycle if the trivial case is excluded. Intuitively, it can be seen that a Hamiltonian cycle instead of just a path is more likely to satisfy the OGMC because the only way to return to the start vertex would mean that you would have to generally increase then decrease the length of the necklaces. Otherwise, all vertices adjacent to the start vertex would already be traversed.

The final morphology of the piece was found by brute force (implemented in the programming language Mathematica) as follows. Generate a graph where a vertex represents all representatives of a given aperiodic necklace (excluding the trivial case) and two vertices are connected if any of their representatives satisfies the LMC (note that this graph is highly connected). Starting at the vertex representing the necklace of length 2, randomly choose one of its representatives. Then from all adjacent vertices remove any representatives that no longer satisfy the LMC (which might eliminate some of the edges altogether). Then randomly choose one of the remaining adjacent representatives. Repeat this process until either a Hamiltonian cycle is found or until the path cannot extend any further, in which case, start over.

As a Hamiltonian cycle was found, the necklace of length 2 is repeated since it was the start and end vertex. Then, the trivial case was added at the beginning and end.

It is unknown to the author whether or not enumerations through aperiodic necklaces similar to the one above can be generated more efficiently. Also, it is unknown under what conditions there exists a graph that submits a Hamiltonian cycle when the minimum and maximum lengths of the necklaces are changed.

2.4 Disjoint subset pairs

Our final example is the piece *partition and gate* (2014) for sustaining instruments and computer. While the morphology is not constrained by an OGMC, it has a LMC of minimal change and uses an algorithm to iterate through the elements defined by the CC that suggests an interesting non-deterministic method to search for Hamiltonian paths in a graph.

The piece works as follows. Two microphones are placed equidistant from a single speaker such that performers, who repeatedly play long sustained tones, can move freely in the space among the microphones and the speaker. At any given time, the microphones map to disjoint subsets (including the empty set) of four sources: a high frequency sine tone, a low frequency sine tone, and two recordings. The subset of sources mapped from the microphone with the louder signal (as tracked by an amplitude follower) is output to the speaker while the other subset is muted. Every 15 to 30 seconds, the mapping from one of the microphone changes such that a source is added or removed (minimal change) while the other microphone maps to the same subset of sources (no change); always favoring mappings that have occurred less. In this case, the musical motivations are both situational and perceptual. By changing the mappings over time, the players' expectations of how they are effecting the system are continually shifting while different combinations of the sources are promoted.

Mathematically, the CC defines all disjoint pairs of subsets of the superset $\{1,2,3,4\}$ (where the numbers indicate the sources). The LMC is that between any two pairs, one subset must either add or remove a number while the other stays the same. The GMC is that pairs that have occurred less are favored.

In a realization of the piece, a computer generates the morphology in real-time by a quasi-random walk with statistical feedback on a graph where the vertices are the subset pairs and edges are induced by the LMC (see Figure 11). The algorithm is derived from James Tenney's dissonant counterpoint algorithm (see [18]), which he used as a defacto quasi-random element chooser for many of his computer generated pieces after 1985. Tenney's algorithm works as follows. A set of elements are initialized to some arbitrary set of probabilities. After each trial, the probability of the chosen element is set very low or to 0 and the probabilities of all other elements are incremented. Simply put, the longer an element has not been chosen, the more likely it will be chosen. Based on the increment function, the algorithm can generate evenly distributed quasi-random choices over a limited number of trials.

Similarly in *partition and gate*, when a vertex in the graph is chosen, its probability is set to 0 and the probabilities of all other vertices in the graph are incremented. Therefore, the walk is generally directed towards vertices depauperate in the morphology up to that point. An example sequence of choices is given in Table 2.[11]

[11] The vertices of the graph (shown in Figure 11) represent the subset pairs such that order of subsets within the pair does not matter. However, the computer program that generates the random walk tracks which microphone is mapped to which subset. This does not prohibit any particular subset being mapped from either of the microphones. For example, the first and last mapping in Table 2 are represented by the same vertex in the graph.

Note that the *partition and gate* algorithm would be exactly the same as Tenney's if the graph were completely connected.

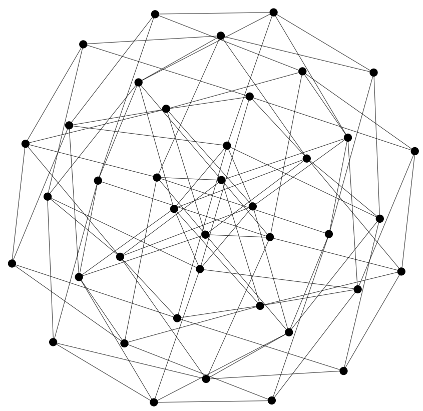
Fig. 11: Graph of *partition and gate*.

Table 2: Example walk of *partition and gate*.

mic 1	mic 2
{1,2}	{3,4}
{1}	{3,4}
{1}	{2,3,4}
{1}	{2,3}
{1,4}	{2,3}
{1,4}	{3}
{1,2,4}	{3}
{1,2,4}	{}
{1,2,3,4}	{}
{1,3,4}	{}
{1,3,4}	{2}
{3,4}	{2}
{3,4}	{1,2}

We leave as an open question whether or not this algorithm (or something similar) might be of use in trying to find Hamiltonian paths in arbitrary graphs.

3 Conclusion

We have examined several types of minimal change musical morphologies. These investigations, particularly with respect to the additional, musically motivated constraints on the analogous mathematical representations, have provided several open questions.

1. How exactly would OGMCs be defined mathematically and encoded computationally for a maximally balanced, maximally uniform long-run Gray Code? How do the three optima of maximal run-length uniformity, run-length maximality, and balance relate to or conflict with each other?
2. What are the lower bounds on the range of extremal values in the running sum of optimal solutions for space-limited de Bruijn sequences?
3. Given a graph with vertices that are representatives of aperiodic necklaces of length *n* to *m* (one representative per necklaces) with edges induced between two representatives if one element is added, removed, or changed, under what conditions is the graph Hamiltonian? Further, if the graph does submit a Hamiltonian cycle, does there exist an efficient algorithm to generate the enumeration?
4. Can Tenney's dissonant counterpoint algorithm be used and/or altered to non-deterministically find Hamiltonian paths in arbitrary graphs?

To add, the author is currently working on a piece derived from multiset permutations where only one transposition/swap occurs between adjacent permutations in the morphology (the LMC) and specifically where only one element in the multiset repeats (the CC). As with most of the morphologies we have discussed, the GMC is that each element occurs only once. Similarly to the piece *gray codes*, a highly balanced morphology with maximally uniform, long run-lengths is desired (the OGMC). And like standard Gray codes, there exists several algorithms for generating minimal change mulitset permutations (e.g., see [14, 16, 28, 25, 23]), but the resulting morphologies have highly varying run-lengths and are unbalanced. While the example was not included in detail because the composition is yet unfinished, the idea has already led to an interesting discussion about the character of multiset permutations and all the questions posed for Gray codes apply.

Finally, it might be of interest to investigate these ideas and objects more deeply with respect to algorithmic complexity [15, 5, 6], graph metrics [9, 30], Polansky's morphological metrics [17], other structural metrics [29], and music perceptual measures (such as perceived rate of change as discussed by Tenney in [26]). Does the number of morphologies that satisfy a given set of morphological constraints relate to complexity? How do morphologies that satisfy the same set of morphological constraints compare under various metrics? Can the taxonomy of morphological constraints presented in this paper as a generative tool also prove useful as an analytical tool? Addressing such questions might give us a better understanding of these types of morphologies and suggest further research.

Acknowledgments

I would like to thank Hakan Kjellerstand. His generous help has led to a number of new pieces and he was essential in the formulation of the ideas presented in this paper, especially with respect to CP paradigms and implementations. I would also like to thank Larry Polansky for helping me clarify many of the ideas. Finally, I would like to thank Juan Sebastián Lach Lau and Gabriel Pareyón for encouraging me to submit these ideas for presentation at the 2014 International Congress on Music and Mathematics.

References

1. Akhmedov, A., Winter, M.: Chordal and timbral morphologies using Hamiltonian cycles. Journal of Mathematics and Music **8**(1), 1–24 (2014)
2. Anders, T., Miranda, E.: Constraint application with higher-order programming for modeling music theories. Computer Music Journal **34**(2) (2010)
3. Anders, T., Miranda, E.: A survey of constraint programming systems for modelling music theories and composition. ACM Computing Surveys **43**(4) (2011)
4. de Bruijn, N.G.: A combinatorial problem. Koninklijke Nederlandse Academie van Wetenschappen **49**, 758–764 (1946)

5. Chaitin, G.: A theory of program size formally identical to information theory. J. ACM **22**(3), 329–340 (1975)
6. Chaitin, G.: Meta Math!: The Quest for Omega. Vintage (2004)
7. Cohn, R.: Maximally smooth cycles, hexatonic systems, and the analysis of late-romantic triadic progressions. Music Analysis **15**(1), 9–40 (1996)
8. Emma, R., Javier, L.: Constraint optimization techniques for exact multi-objective optimization. In: V. Barichard, M. Ehrgott, X. Gandibleux, V. T'Kindt (eds.) Multiobjective Programming and Goal Programming, *Lecture Notes in Economics and Mathematical Systems*, vol. 618, pp. 89–98. Springer Berlin Heidelberg (2009)
9. Fernández, M.L., Valiente, G.: A graph distance metric combining maximum common subgraph and minimum common supergraph. Pattern Recognition Letters **22**(6-7), 753–758 (2001)
10. Fredricksen, H., Maiorana, J.: Necklaces of beads in k colors and k-ary de Bruijn sequences. Discrete Mathematics **23**(3), 207–210 (1978)
11. Gray, F.: Pulse code communication. U.S. Patent 2,632,058 (1947)
12. Klamroth, K., Jørgen, T.: Constrained optimization using multiple objective programming. Journal of Global Optimization **37**(3), 325–355 (2007)
13. Knuth, D.: The Art of Computer Programming. Addison Wesley (1997)
14. Ko, C.W., Ruskey, F.: Generating permutations of a bag by interchanges. Information Processing Letters **41**(5), 263–269 (1992)
15. Kolmogorov, A.: Three approaches to the quantitative definition of information. International Journal of Computer Mathematics **2**(1), 157–168 (1968)
16. Korsh, J., LaFollette, P.: Loopless array generation of multiset permutations. The Computer Journal **47**(5), 612–621 (2004)
17. Larry, P.: Morphological metrics. Journal of New Music Research **25**(4), 289–368 (1996)
18. Polansky, L., Barnett, A., Winter, M.: A few more words about James Tenney: dissonant counterpoint and statistical feedback. Journal of Mathematics and Music **5**(2), 63–82 (2011)
19. Rollon, E., Larrosa, J.: Multi-objective propagation in constraint programming. In: Proceedings of the 17th European Conference on Artificial Intelligence, pp. 128–132. IOS Press (2006)
20. Rossi, F., Beek, P.v., Walsh, T.: Handbook of Constraint Programming (Foundations of Artificial Intelligence). Elsevier Science Inc. (2006)
21. Savage, C.: A survey of combinatorial gray codes. SIAM Review **39**(4), 605–629 (1997)
22. Sawada, J., Wong, D.C.H.: A fast algorithm to generate Beckett-Gray codes: Extended Abstract. Electronic Notes in Discrete Mathematics **29**, 571–577 (2007)
23. Shen, X.S., Williams, A.: A 'hot potato' Gray code for permutations. Electronic Notes in Discrete Mathematics **44**(0), 89 – 94 (2013)
24. Stevens, B., Cooke, M., North, C.: Beckett-Gray codes (2007). Discrete Mathematics, submitted
25. Takaoka, T.: An $O(1)$ time algorithm for generating multiset permutations. In: Algorithms and Computation, *Lecture Notes in Computer Science*, vol. 1741, pp. 237–246. Springer Berlin Heidelberg (1999)
26. Tenney, J.: META + HODOS and META Meta + Hodos, 2nd edn. Frog Peak Music (1986)
27. Truchet, C., Assayag, G.: Constraint Programming in Music. Wiley (2011)
28. Vajnovszki, V.: A loopless algorithm for generating the permutations of a multiset. Theoretical Computer Science **307**(2), 415–431 (2003)
29. Winter, M.: Structural metrics: an epistemology. Dissertation, University of California, Santa Barbara (2010)
30. Zager, L.A., Verghese, G.C.: Graph similarity scoring and matching. Applied Mathematics Letters **21**(1), 86–94 (2008)