Patrik Ohlsson

Computer Assisted Music Creation

A recollection of my work and thoughts on heuristic algorithms, aesthetics, and technology.

Det självständiga, konstnärliga arbetet finns bifogat som partitur.
# CONTENT

1 Introduction................................................................................................................. 3
2.1 Background............................................................................................................. 4
2.2 Hypothesis ............................................................................................................. 5
2.3 Purpose.................................................................................................................. 7
3 Method....................................................................................................................... 8
3.1 Heuristic algorithms .............................................................................................. 8
3.1.1 Backtracking ..................................................................................................... 8
3.1.2 Genetic algorithm ............................................................................................. 11
3.1.3 Practical applications ...................................................................................... 15
3.1.4 Artistic applications ....................................................................................... 18
3.2 Fractal algorithms ................................................................................................ 26
4 Aesthetics .................................................................................................................... 31
4.1 Idealism .................................................................................................................. 32
4.2 Optimized Intelligibility ....................................................................................... 33
4.3 Mathematical-Music unification .......................................................................... 34
5 Result.......................................................................................................................... 36
5.1 *Weights Blows Encounters Motions* – For Choir .............................................. 36
5.2 *Kort Etta* – For Accordion and Acoustic Guitar ............................................... 40
5.3 *KOLOKOL* – For Chamber Ensemble ............................................................... 43
6 Discussion................................................................................................................. 48
7 References ............................................................................................................... 50
8 Appendix.................................................................................................................... 53
8.1 Analysis of Per Nørgård’s Symphony no. 3 bar 61-69 ........................................ 53
8.2 *PrunedSearch* C++ class .................................................................................. 54
8.3 *KOLOKOL* score ............................................................................................... 56
2 INTRODUCTION

This work is the text part of the master’s degree in composition at KMH, it complements the artistic part which is supplied in the form of a musical score. Professor Karin Rehnqvist was supervising the artistic part of this work.

This text is partly a testament to my own journey within computer based composition, in particular for algorithmic music creation. It is also an exploration into regions of the thought-world that emerges as a consequence of the method – discussing aesthetic concepts and implications. I wish to introduce concepts that, in combination with an artistic idea, are enabled through the power of technology. I try to holistically approach these subjects – discussing aesthetics, technique, and realization of some musical idea. This, whilst also supplying plenty of notation examples, some code, and illustrations – specifically, to support the explanation of concepts that are uncommon to most students of music.

The text is, on an introductory basis, presenting concepts of both practical and artistic nature involving computer composition. There are a few sections however, that might require the reader to have some familiarity with a computer composition environment or language (such as Max/MSP or SuperCollider). Not fully grasping these few concepts is not detrimental for the text as a whole, however – some are purposely advanced for completion’s sake, and for inclined readers to return to, if they so wish.
2.1 BACKGROUND

The potential of formalizing the compositional process and using programmable machines dates back to early 19th century. Ada Lovelace, widely regarded as the first computer programmer (Hammerman & Russell, 2015), stated in regards to Charles Babbage’s mechanical computer, the Analytical Engine:

Again, it might act upon other things besides number, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations, and which should be also susceptible of adaptations to the action of the operating notation and mechanism of the engine. Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent. (L. F. Menabrea, 1842)

Though the field of algorithmic composition may be thought of as one involving computers, the core processes and methods predate the computer age. One early example of this is the pitch-time fractal music of Josquin des Prez – the prolation canon, a common form at the time (15-16th century) and it involved the superposition of time-scaled and/or offset versions of a melody. This was superimposed on other harmony related rules. The canon itself is a structure built on the rigorous restraints of these rules that project every choice made by the composer that forces a relation to be considered later in time.

In my own work I have gradually transitioned from a non-algorithmic approach to a fully computer based algorithmic practice in the past five years. Initially this transition was done to spend more time on music making and less on fixing mistakes, but I found almost immediately that this shift in method had deeper repercussions in every part of my music creation.

Beyond solving issues there were several important questions brought up through this transition regarding both my own relationship to technology, the role of the composer, and the nature of music itself. By no means will I try to answer all those questions here, but I hope to provoke discussion and perhaps alleviate some common prejudice (Fowler, 1994) on the subject of computer composition.

The Method section of this text will present some potent tools for working with music composition in both practical and artistic ways. Heuristic search algorithms are the methods that lets us navigate huge spaces of possible states such as combinations of notes, durations, or sounds – to find solutions to a particular musical question. It is when attempting to answer such questions that the connection between a musical quality and the musical material is exposed – a significant effort will be put on just how to formulate such questions in this text.

A brief glance on machine learning and in particular deep learning will be included in a gaze forward in to what the algorithmic composition field may become. I will also present the plans for a research project where heuristic algorithms play a key part – the project is dealing with the entire chain from compositional idea to final rehearsals.
2.2 **HYPOTHESIS**

Composer Per Nörgård made the remark that objects we find and use for artistic purposes (‘objects trouvés’) and the objects belonging to mathematics which is also used for art, that is proportions, the overtone series, and infinity rows – are in kinship to one another.

It may seem surprising that within the same corpus of music, even perhaps within the same work - within the music of the same composer and even within the same work - one encounters on the one hand what have been called ‘objects trouvés’, that is, objects that have been found, and they don't have to be objects, things, like a box of matches, but also objects like birdsong (in the realm of sound), the roaring of the sea, the crash of waves, wind whistling through the grass, and so on, and on the other hand the composer working with proportions which are called Pythagorean, or the systems of the overtone rows, infinity rows and golden sections. There is, as it were, a Platonic, eternal world, and alongside it this world of immediate presence, of sense experiences. Now for me, these two things are not contrasts. Co-existing at the most, because as far as I can see their kinship is in no way different from that found in music, because in the same way in music we find a high level of abstract order linked to a sound that very much appeals to the senses. And if you remove one of these aspects from music and retain the other alone, then what you have is not music, but amputated music. (Nörgård, Kullberg, Mortensen, Nielsen, & Thomsen, 1999)

Nörgård accentuates the importance of the relation of sense and abstract structure, but there is also the reasoning that these are not contrasts. To some extent the “eternal world” (Danish: evig verden) of Pythagorean proportions, overtones, and infinite sequences are discovered objects, repurposed for the sake of art or music – in a similar fashion to Marcel Duchamp’s *Fountain* (Duchamp, 1917).

Now, if music material, form, and every aspect of the music process, would all be “objects trouvés”, harnessed from the eternal world – could a composer be considered a discoverer and not a creator? What is subjectivity in the composition process when the entire structure of the piece is “discovered” and its representation is just a set of practical choices, or even calculations in itself?

The line between representing as, and inventing music, is becoming increasingly blurry in my own work. Is this just a phenomenon of my transition to a programmable digital environment or is music making, at least partially, mathematical structure-representation in sound?

Nörgård’s corpus of music is especially interesting in this case as he has works on both ends of this spectrum – *Voyage into the Golden Screen* is the epitome for such musical mapping of a mathematical concept, the pitch content of the second movement can be reduced to these formulas:

The *infinity row* (OEIS: A004718) definition (0 = G over middle C):

\[
f(n) = f(n - 2) + (-1)^{n+1} [f(n/2) - f(n/2 - 1)] \\
f(0) = 0, f(1) = 1
\]

Instrumentation:

- flute = \(f(n)\)
- oboe = \(f(4n + 3)\)
- clarinet = \(f(2n + 1)\)
- bassoon = \(f(16n + 6)\)
- horn = \(f(4n)\)
- harp = \(f(8n + 7)\)

With some additional instruments accenting longer *wavelengths*\(^1\) of the infinity row, such as every 64\(^{th}\) and 256\(^{th}\) note. The piece is in itself a mathematical object represented in sound by Nörgård’s lush instrumentation and phrasing.

\(^1\) Nörgård uses the term *wavelength* (Danish: Bølgelængder) when referring to every nth number in the *infinity row*. E.g. wavelength 3 would refer to every 3\(^{rd}\) number in the sequence.
There is however a significant difference in being mathematical and being mathematically reducible. It is inspiring to imagine that all music has this elegant logic at the core of their being, just hidden in the sounding representation – but even looking at Nørgård’s other music there is obvious ambiguity where formalism ends and experience begin\(^2\). Sound is described by physics and music structure is to a large extent based on mathematical relations – so every piece could in theory be considered mathematical.

*Voyage into the Golden Screen*, however, is mathematically reducible in that it can be expressed in a single formula generating the content and structure of the piece. Condensing any piece of any composer in to a simple formula\(^3\), is arguably inconceivable\(^4\).

Now, is there even any reason to believe such a mathematical formula exists for any given piece? If general reducibility of this kind would be discovered, how would this change the way we make and analyse music? This will be furthered discussed in the *Mathematical-Music unification*-part later in the text.

---

2 See Appendix – “Analysis of Per Nørgård’s Symphony no. 3 mm. 61-69”

3 That is a function generating (and simultaneously explaining) all parts of the music, and this whilst being disproportionately simple in definition.

4 This is ultimately a discussion on what a piece is. Is it the emitted sound? The experience of the sound? The score? And what about the interpretation factor, the room, and listener preconceptions – all affecting the experience of the piece? Reducibility would only apply to the domain of conception (e.g. the score’s content, the DAW, or the doodles on a sketching paper), but even there – how would one deconstruct the arbitrarily complex layers of ideas and artistic choices of a piece, into a condensed, mathematical form?
2.3 Purpose

There might be no definite answers to the questions stated above – yet I hope to properly introduce these topics and to inspire further research by this text. Also by, in reverse, showing how music can be made on mathematical, or algorithmic ideas – I hope to prove that music and mathematical concepts can share a common ground.

The first part of the text introduces some of the technical concepts such as heuristic and fractal algorithms, that are essential to my own music writing. These concepts are general enough to be applied to practically any musical style, as they are applied prior to, or in concurrence with the conception of the music representation. The text will only present applications that are present in my own work however, that to make it possible to present actual music built on these methods. The structure of the subsections of the Method part is as follows:

- Method description
- Practical applications
- Artistic applications

The second part will deal with some implications of algorithmic music making. Such as the subject of idealism – that is; if it is possible to exhaust the entire search space to find a solution to a musical problem, is there any inherent value (such as beauty) to a perfect solution? We will see that such a solution depends highly on the context of the stated question and the formulation of the question itself. Regarding a solutions value one might have to distinguish between practical and artistic problems but in both cases it is reasonable to favour elegant solutions in contrast to verbose or overly-complicated ones.\footnote{By the principle of Occam’s razor, and Optimized Intelligibility, that will be discussed later.} Now, is elegant the same as beautiful? This remains to be discussed.

In the Aesthetics section we will also discuss the score ↔ musician-relationship and study the effect the visual content of the score has on interpretation. Whilst this could be perceived as non-relatable to the subject of aesthetics it is in the juxtaposition of certain contemporary movements, favouring notational and performative difficulty, and that of optimized intelligibility, a natural outcome of the heuristic workflow (with the goal to maximize simplicity of notation for arbitrarily complex music) – that we might uncover some fundamental aesthetic discrepancies.

These and some other topics discussed in the Aesthetics part are more or less naturally derived from the techniques described in the Methods section and, at least partially, to algorithmic composition as a whole. New possibilities necessitate new theory – these are however only modest, food-for-thought topics, that I hope composers and other inclined readers can experience as inspiring and thought-provoking.
3 METHOD

3.1 HEURISTIC ALGORITHMS

When faced with a difficult combinatorial problem whose optimization may be prohibitively expensive, researchers frequently turn to the study of fast heuristic algorithms in an effort to guarantee near-optimal results. (Langston, 1987, p. 539)

A practical shortcut taken when solving a combinatorial problem is called a heuristic (from Greek: εὑρίσκω, "find" or "discover"). Heuristics are employed when a problem involving combinations is sufficiently complicated and it becomes impossible to solve using a brute force method in a reasonable time frame. A classic example is the travelling salesman problem (TSP) stated as follows: Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?\(^6\) (Flood, 1956, p. 61)

Here each city added expands the problem exponentially and a full brute force search would start to become impractical already at around 11-12 cities. For \(n\) cities there would be \(1 \times 2 \times 3 \cdots \times (n - 1)\) or \((n - 1)!\) combinations to try, e.g. for 10 cities: \((10 - 1)! = 362880\) combinations. Heuristic optimizations and efficient algorithms have made it possible to solve this problem with a million cities (Rego, Gamboa, Glover, & Osterman, 2011, p. 431).

There are several types of heuristic algorithms, from those who simply take shortcuts in a full search effort to those who imitates natural selection and gradually evolve a fitting solution. In general, there are two classes of heuristic methods i.e., those that guarantee an optimal solution and those that do not. When a problem is too large or complicated only an approximate solution might be reasonable to go for – in other cases, such as in the solution of the TSP, a heuristic efficient enough to prune down the search tree to a computable size, could be achieved.

In music there are several combinatorial problems that are similar to the TSP e.g.; deciding accidentals and placing time signatures. We may, as with the TSP, design our own implementation or pick from conventional music praxis to try and solve such problems with heuristic algorithms (see examples in the Practical applications part).

We could also formulate an artistic problem by quantifying a musical quality, defining a problem, and then solving it using heuristics on the resulting search space. One example is finding a combination of sets of pitches (each of size \(n\)) that results in the most occurrences of a specific interval. Another could be finding a combination of a set of sound clips that, when mixed, mimics the spectral structure of a church bell.

In a demonstration I will generate a full instrumentation with notation using: a sample library of instrument recordings, a source sound or sound ideal, a heuristic algorithm, and an exporter. This system can then be extended for microtonal subdivision which also enhances the results (more on this in the Artistic applications section).

3.1.1 Backtracking

A common procedure for solving combinatorial problems is the backtracking algorithm. Imagine picking marbles from some bags and by taking one from each bag you are trying to find a specific

\(^6\) Quote from https://en.wikipedia.org/wiki/Travelling_salesman_problem, see reference for the formal problem statement.
combination, for example; the set of marbles, one from each bag, that are the largest in total. By picking marbles in a certain order we can make sure that we have tried every combination and indeed found the largest set of stones. To visualize this, we imagine a tree structure where each full branch is a complete combination of marbles as shown in Figure 1.

We pick one marble from each bag until we have reached the last bag, there we cycle through all of its stones, after this we backtrack to the second to last bag and exchange our current stone with a new one from this bag – then cycle through the stones in the last bag again. Once we have picked the last stone from the first bag and cycled through all the stones in all subsequent bags we know that all combinations have been shown and can say for sure what the largest combination of marbles is.

This is the full search or brute force method. All combinations are checked with no heuristics for branch pruning, this is not very efficient on a large set of marble bags with many stones in them but it is a good starting point.

As shown in Figure 1, the backtracking algorithm starts at the root of the tree and navigate depth wise down each branch. If we wish to improve the brute force version then each partial solution (Knuth, 1975, p. 125), that is each incomplete combination, could be evaluated against some constraint. If this fails, the algorithm ignores the invalid sub nodes and do an early backtrack up to its parent node to continue with the parents next child node.

A demonstration of the backtracking algorithm for a trivial smallest sum-problem is shown in Figure 2 and Figure 3. We want to determine a sequence of 3 numbers from some groups of positive integers whose sum is the smallest, if the sum is greater than 3 then we have lost.

Figure 1, Marble tree (left-to-right) of three bags (dashed), each with two marbles each. Numbers denoting the order in which the backtracking algorithm picks each marble combination.
In Figure 2 we see a *partial solution/branch* where we will check if the constraints are satisfied. Now as $1 + 3$ is not less than 4 we see that the constraint has failed. Therefore, we *backtrack* up the tree and go to the other possible node instead.

![Figure 2](image2.png)

*Figure 2, Smallest sum < 4.*
Partial solution in grey, this won’t continue to check nodes [1] and [2], as the sum is already = 4.

In Figure 3 we see that there exists a solution all the way down the tree, i.e. $1 + 1 + 1$. Now in this trivial case we would probably stop searching but if we are not certain that this solution is the best or there might be several optimal solutions we would just output and continue the search.

![Figure 3](image3.png)

*Figure 3, Smallest sum < 4.*
Sum is less than 4 and it is the bottom of the tree, we found a solution!

Large trees where each node have multiple sub nodes could obviously require stricter heuristics. In the smallest sum-problem we could, for example, keep track of the scores of each complete solution and backtrack when a partial solution supersedes the best of these values. This could prune down the tree and improve computation times and it would still find the global optimum/optima.

Now, not all problems are of the smallest sum kind, consider what happens if we want the *largest* sum instead. Any partial solution would obviously grow when going further down the tree so we cannot, without some prior information about the nodes, use a threshold constraint. Some problems may not be uniformly increasing or decreasing at all which makes heuristic construction tricky or even infeasible.

The backtracking algorithm is well suited for solving a wide array of musical problems as will be demonstrated in the *Practical/Artistic applications* sections. It works particularly well on smaller combinatorial problems even without strict heuristics. It is easily tailored for optimizations on sequential material such as pitches, scales, durations, number sequences and the like.

My C++ implementation of a version of the backtracking algorithm will be included in the appendix.

---

7 This could be the case when the nodes represent something non-trivial (e.g. a signal, word, object) or the process of evaluation of solutions is cross-domain, e.g. requires a mixing of signals in the time domain and analysis plus constraint evaluation in the frequency domain.
3.1.2 Genetic algorithm

The idea is to start with several random arrangements of components that each represent a complete but unorganized system. Most of these chance designs would fare very poorly, but some are bound to be better than others. The superior designs are then "mated" by combining parts of different arrangements to produce "offspring" with characteristics derived from both their "parents". (Peterson, 1989, p. 346)

Genetic algorithms (GA) are part of a larger group of biologically inspired methods called evolutionary algorithms (EA). These procedures are approximate models of actual natural behaviour such as selection, reproduction, and mutation. These methods are used to solve optimization problems, train artificial neurons, or create self-evolving computer programs.

A GA generates a heap of random (complete) solutions (such as marble combinations) and calculates each solution's fitness, that is, a numerical value signifying how well this particular solution solves the problem (e.g. which marble combination is the largest in total). The most fit solutions are paired and, using a crossover procedure, spliced together to create a new "generation" with a slightly higher average fitness.

The details of the crossover procedure are implementation specific but two common implementations are demonstrated in Table 1. Note that the GA can work on almost any data type and with continuous variables, therefore, designing a custom crossover function might be necessary. Imagine for example what a crossover function on a musical fragment would do – would it splice horizontally, vertically or maybe both?

<table>
<thead>
<tr>
<th>Crossover at random point</th>
<th>Crossover randomly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution 1 (parent):</td>
<td>Solution 1 (parent):</td>
</tr>
<tr>
<td>Solution 2 (parent):</td>
<td>Solution 2 (parent):</td>
</tr>
<tr>
<td>Crossover (child):</td>
<td>Crossover (child):</td>
</tr>
<tr>
<td>aaaaa</td>
<td>aaaaa</td>
</tr>
<tr>
<td>bbbbb</td>
<td>bbbbb</td>
</tr>
<tr>
<td>aaabb</td>
<td>ababb</td>
</tr>
</tbody>
</table>

Table 1, Crossover examples.

A GA with just a crossover method would not work very well in practice. Population diversity would decrease quickly, causing the algorithm to "stall" (average fitness not improving by each generation). This could be mended by 1) including less optimal solutions in the next generation and 2) inducing random mutations on some members of the population. The effect of this is as follows:

1. Helps sustain genetic diversity in the population. In nature, it is not solely the elites that breed the next generation.
2. Simulates the effects of damage, environmental effects and natural variation on individual genes. This also increases variability within the gene pool.

Mutation, as with crossover, is done differently depending on data type and precision. It could be a random alteration applied on all solutions in the population, or just a fraction. It could be a large change, or small. The extent of the mutation could even vary depending on the current generation number – it is all up to the implementer and the problem at hand.

Finally, there are some stopping conditions for the algorithm to seize operation and return the best solution it could find. Some conditions could be superseding a max generations count, improving beyond a fitness tolerance or breaking a time limit (on slow problems).
The GA has proven efficient on optimization problems in a surge of areas ranging from scheduling, product design, computational biology (Ławrynowicz, 2008) (Balakrishnan & Jacob, 1996) (Barta, Flynn, & Giraldeau, 1997), to music and arts (Johnson & Cardalda, 2002).

A significant difference between the GA and the backtrack procedure is that the GA is not guaranteed to return a global optima solution (Forrest, 1993, p. 875). Depending on the starting conditions, crossover and mutation procedures – there is always a risk that the GA gets stuck on a local optimum instead. Another issue is when optima are far apart, solutions might not be able to cross the gap in the search space – this is particularly a risk when the mutation effect is small in comparison to the distances between optima.

These disadvantages are countered by the incredible scalability inherent to the GA – a well implemented algorithm can handle hundreds of variables and achieve high fidelity results.

As mentioned – a GA can be used on many problems of the optimization kind and to end off this section, a GA will be applied to a cross domain fitness problem. Could a ‘tone’ be ‘grown’ in a time domain signal, by looking at it in the frequency domain and evaluate the signal fitness on the spectral content?

The time domain signal consists of 100 samples, these are the variables that the GA will try to optimize. The GA requires a population of several 100-sample sets, or solutions – each solution in the initial population consists of randomly generated values in the range of -1 to 1. This is the initial setup of the program, now on to the optimization procedure.

As mentioned, before calculating each solutions fitness, the time domain signal needs to be converted in to a frequency domain polar signal. This is done using a discrete Fourier transform which is defined as follows:

\[ X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi i kn/N} \]

\( x_n \) is our time domain signal, \( N \) the number of samples in our signal (100) and \( X_k \) the resulting frequency domain signal over frequency \( k \). \( X_k \) consists of complex-valued points in the Cartesian plane but only the amplitude of a particular frequency is of interest in this example – to get this we need to perform a Cartesian to polar conversion (disregarding phase):

\[ A_k = \sqrt{\text{real}(X_k)^2 + \text{imag}(X_k)^2} \]

\( A_k \) contains the amplitude values for frequency \( k \). It is on this frequency domain representation the fitness function will be evaluated.

Now, in this example a ‘tone’ will only be ‘grown’ at a single frequency, say 4 Hz, so what is interesting is the value of \( A_k \) at \( k = 4 \). What would the time domain signal be expected to look like in the end? Well, hopefully like a sine wave oscillating 4 times over the 100 sample signal. And what about the frequency domain signal? A pure sine wave of 4 Hz in the time domain would translate to a single straight peak where \( k = 4 \) in the frequency domain.

The fitness function will not just attempt to maximize \( A_4 \) but it also has to decrease energy at other frequencies, so, the actual fitness formula will be as follows:

\[ \frac{A_4}{\max(A_k, k \neq 4)} \]

A good fitness would mean that the numerator \( A_4 \) is significantly larger than the denominator and the entire expression is approaching infinity.
Figure 4, “Grow a Tone”. Time domain signal found by the GA and optimal solution in dashed line.
The GA was implemented in the Matlab programming environment using the `ga` function (MathWorks, ga, 2016) with mostly default parameters. By default, `ga` uses a “scattered” crossover function, that is equivalent to the “Crossover randomly” in Table 1. For mutation, an adaptive function is used by default – when constraints are present. There are only constraints on the range of values (−1 ≤ x ≤ 1) but Matlab’s `mutationadaptfeasible` function guarantees that mutations will not put any solution values outside these bounds.

The only change to the default parameters of `ga` was population size which was raised to 500 – this proved to be a decent balance between performance and quality on the particular rig this was executed on. After about 500 generations the best solutions in the population were approaching a sine wave, as shown in Figure 4 – the same signal in the frequency domain, the way the GA saw the signal, is shown in Figure 5. Here a clear peak at 4 Hz is apparent and there is not much energy elsewhere.

Obviously, this is not a very artistically interesting problem, but it does demonstrate the power of the algorithm. The GA “figures out” the relationship between a sweeping sine wave in the time domain and a peak in the frequency domain, by just looking at the amplitude at a certain Hz-value, comparing it to the surrounding amplitude level and, gradually evolving a good solution.

It has not been mentioned yet but the time domain sine wave could start at any phase, as this information was discarded in the evaluation process. So there is not just one way for the GA to converge – but infinitely many. The final phase of the sine wave will depend on the initial conditions but with the ability to converge at any phase – it is fascinating that the GA is able to find a particular solution at all.
3.1.3 Practical applications

On to some musical examples where the heuristic algorithms outlined above could be used. The two examples will be on general notation issues that are particularly relevant to melodic, serialist, and polyphonic writing. These techniques are, to some degree, present in my own work – as will be further expanded upon in the Result part.

3.1.3.1 Accidentals

When notating a melody, series, or polyphonic part – one will have to decide on a system for setting accidentals. In modal music with no key changes this is not, typically, an issue. If key changes are present there might be some ambiguity on where new accidentals should be introduced – but it is not, unequivocally, a case of right and wrong. Instead, the scenario presented here is when the pitch material is free tonal, atonal, or of unknown structure – as could be the case when mapping a number series to a chromatic scale.

First of all, what could be some of the problems when only picking accidentals of one type (flat or sharp)? In Example 1, a randomly generated 12-tone row with only flat accidentals, some discrepancies are clear. There are augmented intervals from notes 1 and 9, and diminished intervals from note 6 and 7 – this makes the notation a bit harder to read whilst also hinting at a modality that may not be present.

Example 1, All flat 12-tone row.

A neutral representation of the 12-tone row in Example 1 would, preferably, lack augmented or diminished intervals. Also, in an atonal context, it is reasonable to avoid F♭, E♯, C♭, or B♯ – as they impair the readability of the music. So, how could selecting melodically sound accidentals be thought of as a combinatorial problem of the marble-picking kind?

Now, each black key-pitch could be expressed in two ways – sharp low note or flat high note. White key-pitches could, potentially, be written in three ways – for this example only the natural version is allowed. Analogue to the marble problem, we could think of each pitch as a bag of marbles, but instead of stones – each bag contains the different alterations of a given pitch.

![Figure 6, Alteration tree (left to right) - first four pitches in Example 1.](image)

The search tree when dealing with accidentals could look relatively minimal, as it only branches on black key-pitches. For longer melodic segments, however, it may turn prohibitively complex as branching exponentially increases the amount of combinations. In Figure 6 the search tree for the first
four notes in Example 1 is demonstrated, and it is clear how this could be navigated in the same manner as the two previous backtracking examples.

Now, what remains are the backtracking constraints – that is, the rules deciding if a partial solution is defunct and backtracking should happen. This could, for example, happen when a branch appends a diminished/augmented interval or reaches a sharp even though all previous music is notated in flats and flat is a possible alteration on the pitch.

The obvious way of filtering out diminished or augmented intervals would be to just backtrack when encountering one – however, this would not guarantee that a solution could be found. Consider note 5, 6, and 7 in Example 1 – there is no way to avoid a diminished/augmented interval without using double accidentals. By this fact we see that terminating an entire branch due to a single diminished/augmented interval is a bad way of guaranteeing good results.

An alternative approach would be to only backtrack on really bad partial solutions, such as when intervals are notated in a different direction from the sounding pitches (e.g. $E\# \rightarrow F\flat$). For lesser violations a penalty score would be incremented and the search continued. To improve the heuristics, the penalty score of a partial solution is compared against the best complete solution. If the penalty score surpasses the best penalty score of a previous complete solution, it can be concluded that no optimal solutions exist down that branch – backtracking can be performed.

![Example 2, 12-tone row (from Example 1) backtracking results.](image)

In Example 2 the results of this algorithm on the same 12-tone row as in Example 1, are shown. The algorithm was implemented in C++ with the input being the MIDI values of the pitches (middle C = 60) and the output as a vector of number pairs. The first number in each pair being the ordinal (unaltered pitch) and the second number the alteration (-1, 0, or 1).

This is not the only solution, as is clear by the 6th note, that could just as well be $F\#$ – fixing the diminished third from 6 to 7, but creating an augmented unison on 5 to 6. Additional heuristics could be imagined, such as incrementing the penalty when a minority alteration is present in a solution – as would be applicable in a locked-down situation such as the note 5-7 dilemma (Example 1 to Example 2). As flats are in minority, over all, penalizing minority accidentals would then result in an additional penalty score on the 6th note – making $F\#$ a more viable solution.

For this algorithm to work in a polyphonic setting the heuristics would have to be altered somewhat – making the implementation a bit more complex. This won’t be detailed here but I recommend anyone interested in this to try this out themselves. Were one to allow double accidentals but still want the algorithm to prioritize natural notation then this would have to be arranged correctly in the penalty system.

3.1.3.2 Time signatures
Some problems are perhaps not as common in manual music notation as they are in algorithmic composition – this is true for the following technique. This method should, however, be of interest for anyone writing polyphonic music. The problem this method solves is the following: when generating
or composing a rhythmically complex and highly individualized polyphonic texture – how can we ensure that optimal time signatures are selected to maximize on-beat events?

It might seem easy at first to imagine how a backtracking algorithm or GA could solve this problem, but it does in fact introduce some new issues.

1. There are basically infinitely many time signatures – that is if *any* subdivision of meter and *any* compound bar structure is accepted (in practice, however, it doesn’t make sense to group below the shortest note duration or create excessively complex compound bars).

2. As time signatures are laid out consecutively, and may be of varying length, there is also a variable number of time signatures in a complete solution. This comes as the total length of the bars has to be $\geq$ total length of the music, and therefore, the time signature count is dynamic. For the search tree this means that the tree depth, or (complete) solution size, is variable – determined by the content of a particular partial solution.

Besides this, there are no additional complications to what was shown in the accidental problem.

It could be argued that a set of time signatures producing the most on-beat notes is optimal, one might also wish for note ends on beats, as this further simplifies the music-reading\(^8\). A reasonable conclusion would be that on-beat note ends should be at a lesser priority than on-beat note starts or the algorithm might select an unfit solution.

The particular complications in this example does not hinder using a backtracking algorithm to find an optimal solution – it is, however, inherently easier to implement using a GA (at the possible expense of *globally* optimal solutions). It is done in these five steps:

1. Calculate start (and end) positions in absolute time of each note in the music to be grouped.
2. Select a number of time signatures, or possible numerator/denominator combinations that will be tested by the GA.
3. Calculate a fixed variable count (number of consecutive time signatures) by:

   $$\text{variable count} = \frac{\text{total music time}}{\text{shortest bar time}}.$$  

   E.g. if the total music time is 30 quarter notes and the shortest time signature is $3/8$ then:

   $$\text{variable count} = \frac{30/4}{3/8} = 20.$$  

   Solutions overextending the total music time could be discarded by posing a *nonlinear constraint* (MathWorks, Nonlinear Constraints, 2016) – or included and the overflow is ignored.

4. The *fitness function* receives a random complete solution of time signatures that are guaranteed to hold the entire music. The absolute start positions of all beats are calculated from this sequence of measures. Any start (and end) position calculated from the music that does not sit on a beat increments the solutions penalty score.

5. Finally, the GA does its magic of picking from the population of sets of time signatures and gradually improving them until an optimal solution has been found.

---

\(^8\) These rules are good for textural music with little rhythmical accentuation (e.g. György Ligeti’s *Lontano*), but less adaptable, in general, to homophonic parts where melodic movement would influence measure structure.
3.1.4 Artistic applications

Two useful examples were shown on practical\(^9\) notation issues, but, it is in finding solutions to artistic problems, that I personally believe the potential of heuristic algorithms is the greatest. Two problem statements on two, seemingly contrary composition techniques, will be introduced here in short and later be exemplified in actual music – in the Result part.

Here, I wish to emphasize that the composition techniques presented will not be done so in an overly-faithful way to their inventor(s) – as I would rather explore the full potential of each technique (sans archaic aesthetic restrictions). Arguably, any composition technique is merely a vessel of artistic *potential* that should not be ignored or praised on any historical preconception. I do find, however, that it is in the undiscovered cracks within this *known* that the most communicative expression is found.

3.1.4.1 Twelve-tone

I do not have any particular affection for early 20\(^{th}\) century twelve-tone music – yet, I have found that working within a restricted space, such as in dodecaphonc music – can be inspiring at times. A twelve-tone row consists of twelve notes of twelve unique pitch classes. Now, there are guidelines on constructing twelve-tone rows imposed by the inventor, Arnold Schoenberg, that are, in no way, adherent to the technique itself – instead, rather a testimony on his aesthetic views (and, of course, to some of his contemporaries).

The term *emancipation of the dissonance* refers to its [the dissonance’s] comprehensibility, which is considered equivalent to the consonance’s comprehensibility. A style based on this premise treats dissonances like consonances and renounces a tonal center. By avoiding the establishment of a key modulation is excluded, since modulation means leaving an established tonality and establishing another tonality. (Schoenberg, 1950, p. 150)

Schoenberg wished to get rid of any hints of tonality in his dodecaphonic music and keeping true to his ideals now, would have implications beyond the fundamental rule of having twelve different pitch classes. I will respectfully ignore Schoenberg in these specifics, but, is it possible to take his general concept and search for 12-tone rows that are musically dissolved in other ways?

A natural first step would be to not only have 12 unique pitch classes but also 11 unique intervals. This is commonly referred to as an *all-interval twelve-tone row* and it has been used extensively by composers Elliott Carter (Childs, 2006) and Luigi Nono (Il canto sospeso, 1955).

![Example 3, Luigi Nono – Il canto sospeso. All-interval twelve-tone row.](image)

Typically, intervals of the same size but differing direction are still treated as equivalent and should only happen once. This is demonstrated in Nono’s all-interval row for *Il canto sospeso*, shown in Example 3. In disjunction with Schoenberg’s dodecaphonic design principles – Nono’s row also has

---

\(^9\) Practical issues are often indiscernible to artistic issues when discussing a specific implementation, the separation is true only from the *bird’s eye* perspective of this text. That is, that *practical* application considers more general issues, whilst *artistic* applications consider a range of *specific* issues.
cadence like movement (such as in note 4-6), that might hint at a tonality. It is, however, the expanding ranges; the dramatic structure of the row – that are the most striking.

Even with all-interval rows it is questionable if the uniqueness of each interval is really a factor in the perceived dissolution of the music. I.e., the row in Example 3 with its powerful trajectory – does arguably not match the punctualist undertones of its unique pitch classes and intervals. Still, there are conceivable situations where an all-interval row, with its lesser internal “connectedness”, is more appropriate to use than an ordinary twelve-tone row – for example in polyphonic music, if repeating harmony is sought to be avoided.

Are there twelve-tone rows that, on an even deeper level, lacks repetition? Even with the questionable perceptibility of the phenomena could, not only, uniqueness in pitch classes and intervals – but also intervals of intervals, and deeper – be of any artistic value?

An ‘interval of intervals’ is essentially just the difference of consecutive intervals. These could be thought of as the growth / shrinkage of the intervals over time and they can be expressed in the ways written on the last two rows (from the top) of Table 2.

<table>
<thead>
<tr>
<th>Pitch (semitone)</th>
<th>0</th>
<th>1</th>
<th>-1</th>
<th>2</th>
<th>-2</th>
<th>3</th>
<th>-3</th>
<th>4</th>
<th>-4</th>
<th>5</th>
<th>-5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diff 1</td>
<td>-2</td>
<td>3</td>
<td>-4</td>
<td>5</td>
<td>-6</td>
<td>7</td>
<td>-8</td>
<td>9</td>
<td>-10</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diff 2</td>
<td>-3</td>
<td>5</td>
<td>-7</td>
<td>9</td>
<td>-11</td>
<td>13</td>
<td>-15</td>
<td>17</td>
<td>-19</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diff (</td>
<td>Diff</td>
<td>)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2, Number representation of all-interval row in Il canto sospeso. |_ _ | denote the absolute values.

In Table 2 a numerical representation of Nono’s row in Example 3, is shown. The top row is equivalent to the pitches (in semitones) from a centre pitch (0), the 2nd row from the top displays the intervals, and 3rd and 4th rows the interval difference – with the 4th row displaying the difference on the absolute intervals (non-negative). It is clear that the 4th row is a more tangible representation, as it is observable (in Example 3) that each interval grow one semitone at a time. Yet, when considering the bidirectionality of intervals – row three from the top is the more correct.

Obviously the individual values in the 4th row of Table 2 are not unique (they are all 1), and if we wrap musical intervals \( \geq \) an octave (e.g. minor 9 = minor 2, major 10 = major 4) and disregard direction – then, the values in the 3rd row are not unique either. The interval wrapping procedure on the numerical representation is defined by: \( W(x) = |x \mod 12| \), where “\( \mod 12 \)” refers to the remainder after dividing by 12 (retaining the sign of \( x \)), and \( |x| \) meaning the absolute value of \( x \) (removes the sign).

The wrapped numerical representation of the 3rd row is then: [3, 5, 7, 9, 11, 1, 3, 5, 7, 9]. The first interval difference (originally a falling minor third) is now equal to the 7th interval difference (originally a falling minor 10) – in this definition, not all interval differences are unique in Nono’s row.

Now, are there even any all-difference twelve-tone rows so that, not only, all interval differences are unique – but also the differences of the interval differences, the differences of the differences of the interval differences and so forth?

It turns out, and the backtracking algorithm can exhaustively prove, that no complete all-difference twelve-tone rows exist (at least not within an octave’s range) – but, it does get very close. First, let’s look at a complete difference matrix (post-wrapping) for Nono’s row:
Table 3, difference matrix of the 12-tone row in Il canto sospeso.

Each row in table 3 displays the wrapped values of the unwrapped difference of the row above – the same method used in Table 2, but direction excluded. As shown – the first row (the musical intervals) are all unique – but values are repeating in the 2nd to 8th row, which would not be the case in an all-difference twelve-tone row.

Searching for an all-difference twelve-tone row is relatively straight-forward, and similar to what has already been demonstrated on accidentals and time signatures. As in the accidentals-algorithm, I employ a penalty system rather than a terminating one, to ensure that good solutions are not discarded. The penalty score is calculated from the difference matrix – starting at zero, it increments once for each repetition of a number on a row in the matrix.

If all twelve tone-combinations within an octave is tested by the backtracking algorithm using the penalty score for heuristics (see the Accidentals example) – then there are no 0-penalty solutions in the entire search space. More interestingly however, there are exactly four 1-penalty solutions – in fact four versions of one series – that happen to be both all-interval and all-interval-difference twelve-tone rows.

Example 4, Four all-difference 12-tone rows.
Note that each row is transposed to start at C.

Of these four twelve-tone rows shown in Example 4, the second is just the inversion (SI), the third the retrograde (SR), and the fourth the retrograde-inversion (SRI) of the first row (S).
The imperfection is on the 4\textsuperscript{th} difference level (that is the differences of the differences of the interval differences) – a recurring tritone in the, arguably, imperceptible sub structure of the twelve-tone row. This particular repetition of a tritone is present in all four versions of the row – on this same level.

Of 12 factorial, or 479001600 possible combinations – there is, effectively, just one (imperfect) all-difference twelve-tone row. It is, by this definition – the least repeating and most melodically dissolved combination of twelve-tones in an octave, that could exist. If this quality transfers to the realm of perception is another matter of course, and one that is subject to further study.

3.1.4.2 Spectral composition

The Orchidée and Orchids tools, developed at IRCAM in Paris – are two relatively well known spectral composition software, that generate orchestration suggestions based on the analysis of a sound file (Esling, 2014). Orchids uses an interesting combination of partial tracking, psychoacoustic classifiers, and heuristic search algorithms (Esling, 2014, p. 10), to find a matching orchestration to a source sound – the specifics, however, are not well documented (it is a proprietary software, after all).

I decided to construct my own algorithm for generating an instrumentation based on the spectral content of a given sound clip in late summer 2015. This developed in to a miniature suite of orchestration tools for Matlab – not yet released to the public.

At the core is a synth or sample library generating the audio data to be selected from. Most of the tools will work with any VSTi (virtual instrument), but by default they use the 120 gigabyte orchestra library – EWQL Symphonic Orchestra Platinum. The hosted VSTi is programmatically manipulated to construct matrices of audio data, to be used in the combinatorial problem. The final output is a Lilypond notation file (.ly) containing the suggested instrumentation. This file can be further manipulated in any text editor, or in the freely available Frescobaldi notation software (frescobaldi.org, 2015).

There are specific backtracking algorithms for small instrumentations that can exhaustively search through all instrumentation combinations, and genetic algorithms for medium, to large instrumentations. The tools differ in how they compare the source and generated instrumentation sound. Commonly though, a long-term average spectrum (LTAS) is calculated from a few seconds of audio data (from the instrumentation) – this can then be compared with the data from the source sound.

Several methods for comparing temporal and spectral data are already built-in, but the option of supplying a custom comparison method is also supported. In some of the tools the comparison is done directly on the LTAS of the two sounds, whilst others do various manipulations of the data first. Some built-in methods for comparison sounds include partial distance comparison, regression analysis, and fundamental analysis.
Unlike *Orchids*, one can also specify a *sound ideal*, in the form of a custom fitness function – which the algorithm will optimize. There are some psychoacoustic helpers, such as perceived amplitude correction (Fletcher-Munson curves), that assist in improving the result. An objective could be designed for, for example, finding the instrumentation that generates the most audible partials, the most even spectrum, or the most energy on the pitch A♭. That way, it is not only useful for sound *matching*, but for almost any orchestration problem. Some areas this could be used are in spectrally consistent reductions, and adaptations of a piece for a new instrumentation.

All the details on the underlying algorithms driving these tools is beyond the scope of this text, but the toolset was used extensively in composing *KOLOKOL* a piece I made for chamber ensemble. For the obliged, I suggest reading the part on this piece in the *Result* section.

Some of the tools support floating point indices. The integer part of the index could then be used for selecting a sound from the matrix – and the fractional part for calculating a pitch shift on that sound.

---

10 In reality, the tools create a support structure for easily reconstructing the instrumentation (with pitch and dynamic) from the indices in the matrix (generated by the GA).
This way, continuous microtonal solutions could be generated – optional pitch rounding is also possible at any stage of the algorithm (for notation purposes).

The tools are fairly rudimentary, not having any built-in way of dealing with spatialization for example (as is included with Orchids) – but they are programmed on a highly modular principle, meaning that such features could be appended almost anywhere in the algorithm. Spatialization, a custom fitness function, or heuristic could easily be supplied as an extension – via a Matlab file (.m) or anonymous function.

The reasoning behind this extensibility is that the GA is highly sensitive to initial conditions – seemingly small changes in the input data or GA settings, might unexpectedly produce bad results. Being able to control the parameters, extend, and reshape the algorithm for each problem – is therefore of importance on an artistic level.

An interesting phenomenon, that I became aware of when I was writing the piece Weights Blows Encounters Motions for choir, and the piece Variation for 13 musicians (see Example 5) – was the effect of having an extraordinary density of events and voices in close proximity. This seemed, at moments, to have the potential to destroy all perceptive identifiers pertaining to the individual voices, or groups – resulting in the experience of a new, non-divisible sound. The effect is similar to what happens when instruments are playing pitches at harmonic partials from a common fundamental pitch. When balanced perfectly the instruments may no longer be perceived individually, they have become the sum of their parts. An example of this phenomenon, on a harmonic series on E♭ can be heard at the very end of the Prelude to Act I in Richard Wagner’s Parsifal (see Example 6).

Gérard Grisey also described a technique in the article “Structuration des timbres dans la musique instrumentale”, that seem to enable the investigation of this phenomenon. He called it instrumental synthesis – a reconstruction (and extension) of a source sound by orchestrating instruments on the partial pitches of the source spectrum (Grisey, 1991). However, the purpose of this technique was not necessarily to create the illusion of a new sound, but rather to realize a greater theory of spectral harmony – yet, it is reasonable to believe that the perceived illusion of two or more sounds fusing, to create the illusion of a new sound, is in some way a result of overlapping spectra.

A personal goal I set up, in late summer 2015, was to create a music piece consisting only of such sound illusions – never revealing the true sources contained in each polyphonic sound. I started working on the tools presented in the first part of this section, with the hopes of having a workable prototype for the two pieces I would make in the spring of 2016 – one for orchestra, and one for 7 musician Pierrot ensemble. Unfortunately, discovering instrumentations that would generate such sound illusions proved extremely difficult. Having a sound to compare against and then trying to approach its spectral identity, by listing sounds of various instrumentations was one thing – finding a quantifiable measure for the perceived degree of illusion proved a much harder task. There are several potential reasons why it did not consistently succeed in adequately solving the instrumentation fusion problem (IFP), some that I have analysed this far are:

1. Problems pertaining to the heuristic algorithm (GA, backtracking). Resulting in sub optimal solutions even though good solutions existed within the search space.
2. Problems pertaining to the fitness function. Resulting in improper quantifications of relevant perceptual factors in the IFP (several ways of analysing each generated instrumentation’s spectrum was tested).
3. Problems pertaining to the sound generating modules (for the instrumentations). Resulting in an inadequate or incomplete search space.
4. Problems pertaining to external factors and method. Perhaps there are no instrumentations from the selected instruments that would create such an illusion. As the algorithm was mainly testing orchestrations of long notes (with some exceptions), perhaps there were solutions using moving pitches.

Finally, I had to settle with a weaker mimetic form of the software (see Figure 7) to do the analysis for one of the pieces (KOLOKOL for Pierrot ensemble). Neither the fusing or mimetic form of the tool was used for the orchestra piece. Yet, the prospect of utilizing illusions of fused sounds in music composition could, like universally unique structures – reveal an entirely new, spectrally surreal, domain of expression.

Example 5, Patrik Ohlsson, Variation for 13 musicians – trills and expressive notes.
3.2 Fractal algorithms

Leaving heuristic algorithms for now and moving on to another central concept for my own work. Fractal integer sequences, of which Nørgård’s infinity row is one, was the main subject for my bachelor thesis (Ohlsson, 2014). Several methods for generating and composing with self-similar number sequences were exemplified there – therefore, this part will only discuss some recent development regarding self-harmonizing fractal sequences.

First, a clarification – there are many kinds of “fractal sequences”. Any sequence, containing itself as a sub sequence is by definition self-similar / fractal\(^{11}\). Drexler-Lemire and Shallit uses the term “\(k\)-self similar” when the sequence recurs on every \(k\)th element (Notes and Note-Pairs in Nørgård’s Infinity Series, 2014, p. 11) – this is the case with Nørgård’s infinity row, and it is the definition used in this text.

Now, what does self-harmonizing mean in this context? Looking at the infinity row of Nørgård, one of its defining properties is the exact recurrence of the sequence on every \(4\)th, \(16\)th, \(64\)th, and so on, element. This means that two musicians playing this exact same pitch sequence – one on \(16\)th notes and the other on quarter notes – would result in them playing unison intervals on all concurrent notes. Also, a third musician playing the inversion of the sequence on half notes – would play in unison to the others as well.

This is pretty remarkable in itself, yet I wondered – could the cross-voice interval be something else than unison on simultaneous notes? Could series be discovered that recur on musical thirds, or even has a changing relationship to the original voice? Searching for such a sequence could possibly be done through heuristic algorithms, but a more straightforward way is possible if we revisit the original definition of Nørgård’s infinity row.

\[
f(n) = f(n-2) + (-1)^{n+1} [f(n/2) - f(n/2 - 1)]
\]

This shows that the element \(n\), for which to evaluate, is dependent on a value found two steps earlier (\(f(n-2)\)), the interval at the index half the value of \(n\) (\(f(n/2) - f(n/2 - 1)\)), and a sign changing function (\((-1)^{n+1}\)). New values depend on prior values, that themselves depend on prior values, and so forth.

Now, a \(k\)-self similar sequence could simply be constructed by picking \(k\) random numbers, iterating over all other indices \(n = k \cdots m - 1\) and looking up the value on index \(n/k\) on every \(k\)th value. All other values could just be picked at random, as is demonstrated in Example 7.

\[\text{Example 7, Plain k-self-similar sequence for } k=3, 
\text{constructed from random chromatic pitches.}
\]

Sequence recurring on every \(3\)th value, highlighted above the staff

\(^{11}\)Consider the sequence of numbers that result from counting \(1 \cdots n\) for each number \(n\) on the number line in order, i.e. \(s(m) = 1, 1, 2, 1, 2, 3, 1, 2, 3, 4, 1, 2, 3, 4, 5 \cdots\) (A002260, in the OEIS). If we remove the first occurrence of each number, \(1, 2, 3, \ldots\), in \(s(m)\) – this will return the very same sequence \(s(m)\) again. This can be repeated indefinitely.
The sequence in Example 7 only displays self-similarity on one level, that is, every 3\textsuperscript{rd} value from the starting point. In this way it is a fairly uninteresting sequence, any value could occupy each “random” slot so there is a fundamental arbitrariness to this entire process. So, how can several levels of self-similarity be achieved? What would a level be?

A level of self-similarity can be thought of as a $k$-power-scaling at a particular offset in the sequence, where the original sequence is recurring (in Example 7 on every 3\textsuperscript{rd}, with offset 0). In Nörgård’s infinity row the full sequence recur transposed, and/or inverted – on any offset value if $k$ is a power of 2 (> 1) (Ohlsson, 2014, p. 14). This means that, even if we start at say the 26\textsuperscript{th} value in the series and take every 4\textsuperscript{th} value on to infinity, this sequence would already have occurred starting at offset 26/4 (rounded), i.e. index 7, 8, 9, 10 … in the row.

$$s(n) = \begin{cases} a, & \text{if } n = 0; \\ b_m s(n/k) + c_m, & \text{if } n \mod k = m. \end{cases}$$

$m = 0 \cdots k - 1$

A general method for constructing $k$-self similar sequences like Nörgård’s row is shown above, the variable $a$ is the starting value of the sequence $s(n)$, $b_m$ is the coefficient of a recurrence level $m$ (e.g. $-1$ creates an inversion to $s(n)$). Finally, $c_m$ determines the transposition of each level.

For Nörgård’s row: $a = 0$, $b = \{-1, 1\}$, $c = \{0, 1\}$ (Drexler-Lemire & Shallit, 2014, p. 1).

Now, it is possible to map the numbers to pitches in a scale, create $k$ voices – one for the original sequence, and one for each level with offsets ($1 \cdots k - 1$) – and do the corresponding offset in time for the offset voices.

In Nörgård’s row the offset voices would have to be transposed correctly to actually overlap, and here, the first harmonizing possibility becomes apparent. What if the sequence’s formula was altered so that, without transposition, certain given intervals would occur between the offset voices and the original?

Consider what happens when $k = 2$, $a = 0$, $b = \{1, 1\}$ and $c = \{0, 4\}$. If the resulting sequence is mapped on chromatic 16\textsuperscript{th} notes with 0 = E over middle C, and an offset voice that is the original sequence at half the speed, offset one 16\textsuperscript{th} duration, is created – the results shown in Example 8 is generated. Note that all concurrent pitches are now at a major third interval, not unison like before.

This sequence is trivial, but it demonstrates the effect of the transpositions ($c$) on the self-similar levels. As $k = 2$, there are only two unique levels (offsets $\frac{0}{16}$ and $\frac{1}{16}$, scale 2) to generate.

It is possible to create harmonized self-similarity on the even notes of the series in Example 8 too. To modify the series above to generate a minor third above the original sequence on every odd eighth
note and keep the major third below on every even eighth note, you would just set \( c = \{-3, 4\} \). The result is shown in Example 9.

It is obviously impossible to create anything other than a unison on the first note of the 0-offset level (voice 2) as the original pitch sequence (voice 1) is completely identical – on all other concurrent notes a minor third above the original sequence is maintained. To work around this inconsistency regarding this first note in voice 2 – set \( c_0 = 0 \), this will create a 0-offset (2-scaled) voice that is unison on concurrent notes, this could then simply be transposed up a minor third.

Now, the sequences in Example 8 and Example 9 quickly collapse in to a repeating pattern of little variation. An easy fix is inverting one, or several of the levels, so that the inter-level relationships turn more complex and variation is preserved. One risk of doing this is that intervals in the original sequence could expand very quickly – this phenomenon is shown in Example 10\(^\text{12}\).

Static harmony in a chromatic context like this could quickly get boring, even if the base pitch is changing in interesting ways. Now, it is possible to initialize the sequence in different ways by setting some of the pitches to fixed values and then expanding – this way the unfolding could be controlled in

---

\(^{12}\) Those familiar with Nörgård’s sequence will notice that the shape of the sequence in example 6 is very similar to his row – the values of the coefficients in \( b \) are here the same as Nörgård’s \( \{-1, 1\} \), but \( c \) differs as it is kept at \( \{-3, 4\} \).
detail and the self-similar structure would transform around it. Some examples of this will be discussed in conjunction to the part on my piece *Weights Blows Encounters Motions* for choir, under the Result section.

So far, only contiguous $k$-power scales have been explored. The effects of having static intervals on concurrent notes on levels with nonadjacent scalings (such as 1 and 4) is a bit more complex. By copying the first voice in Example 10, making it four times slower, and playing them together – it is clear that they would be in unison, not in the expected thirds, on simultaneous notes.

The harmonized self-similarity did not transfer to scaling 1 and 4; this happens as a result of transpositions stacking for each scaling, and including an inversion, these will cancel out. If the inversion is removed ($b = \{1, 1\}$), then the 4-scaling would be at $2 \times c_0$ transposition – in this case a tritone ($c = \{-3, 4\}$) below the unscaled voice.

Finally, $c_m$ does not have to be constant. For adjacent scalings each transposition $c_m$ could be a function of the pitch at that particular level (i.e. $c_m \rightarrow c(m)$). This enables the infusion of external harmony concepts (e.g. tonality, free tonality, atonality). Defining the sequence in an arbitrary way, from concurrent pitches at an adjacent scaling, will, typically, break self-similarity at other nonadjacent scalings.

Even applying a harmonic idea whilst trying to preserve self-similarity on adjacent scalings, could be tricky – and it is not incomparable to the challenges faced when composing prolation canons (Ohlsson, 2014, pp. 10-11). The offset projections and the chromatic scale, however, produce some challenges that are not typical in canon writing\(^\text{13}\).

By choosing each note in an $k = 2$-sequence, based on the interval that would occur to its 0-offset 2-scaled, 1-offset 2-scaled, and 0-offset 4-scaled level, a sequence such as the top voice in Example 11, could be constructed. Here a semi-tonal approach is applied to try and keep the sequence in a key. This is rather difficult in the presence of inverted voices. The second voice brings forth pitch classes that are foreign to the established key, whilst the third and fourth voice are conserving the key by restating the old pitch classes.

---

\(^{13}\) Yet, relating voices based on indexes in a sequence is, arguably, not as tricky as relating time scaled variations on a voice – as is done in prolation canon writing.
the scope of this text. In fact, as this method breaks self-similarity on bigger scales it cannot really be considered a fractal\textsuperscript{14} method.

A combination of static and dynamic harmonizing sequences were used in my piece \textit{PARALLEL} for symphony orchestra. I had no explicit tonal or modal harmony in mind, rather, I was exploring sets of repeating relationships between scaled voices, as offset parts on the same scale could have their own, independent, set of intervals – this was shown in Example 11. Formally, this mean that every transposition in $c$, could be a repeating sequence in itself, e.g.: if the first offset level would shift between minor third and pure fifth interval to the unscaled voice then $c_1 = 3, 7, 3, 7, 3, 7 \ldots$, this is demonstrated in Example 12.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{example12.png}
\caption{Example 12, Unscaled and 1-offset, 2-scale voice being a minor third apart on even notes and a pure fifth on odd.}
\end{figure}

\textit{PARALLEL} is similar in method to another piece, \textit{Weights Blows Encounters Motions}, I composed for choir, in that melodic fractal sequences are at the core of the musical structure. The orchestra piece has not yet been performed, so therefore I find it appropriate to only detail the choir piece in the Result part.

\textsuperscript{14} Imagine in example 7, that we introduced an inverted 0-offset 8-scaled voice, the equivalent of a twice as slow inversion of voice 4. The notes would be G and F over middle C. Now, the F would not match up to well with the C\textsuperscript{7} chord in the first beat of bar 2. As “fractal” in this text is defined as; \textit{maintaining a relationship across scales} – this sequence has already failed. This does not exclude the possibility that self-similarity could be preserved in combination with an external harmony system – it is just subject to further study.
4 AESTHETICS

I believe that the human-tool relationship is well described using the term abstraction level, a concept common to programming. A tool lacking any understanding of the task it is put to do, yet having the mechanical prerequisites to perform the task, is considered at a low level of abstraction (i.e. pen and paper). A tool that has some understanding of the task at hand and therefore is able to hide (abstract) some of the work in sub tasks – is considered of a higher abstraction level (i.e. notation software).

A tool that only exposes certain interactions through its interface (i.e. high abstract level) could derail the execution of a task, peripheral to the area-of-use considered in the tools’ design. On the opposite side – an unsophisticated tool that directly and inefficiently manipulates the medium might prove unreasonably slow.

On this premise I find the greatest danger to be in the use of a higher abstraction level-tool for the promise of benefits, such as speed and efficiency – if one is not concerned about the possibly detrimental effects of (unknowingly) subscribing to the toolmaker’s design concept. Will and discipline can counteract and perhaps neutralize the will of the tool, but what effects are in place with regards to the accumulation of minor choices? Those that might seem insignificant in the moment, perhaps left at the software’s default value – yet prove significant at a later stage.

Through the tool’s facilitation of an aesthetic idea, in the subtlety of default values and interface design, stock choices might subliminally be favoured. Ultimately, leading to an overwhelming propagation of the toolmaker’s aesthetic concept, allowed to shape the image of our time’s music and style. In an era of relatively few commercial choices for digitally editing notated music, is it even more relevant to question aesthetic freedom in notated music creation? Arguably, this is not an issue of freedom of expression, plenty of options are freely available (e.g. Lilypond, MusicXML, pen, etc.) – it is always reasonable, however, to question the creative limit of one’s tool, on the premise that it might influence choice.

The pen could be considered an extension of the body – it is an enabling tool, representing thought in symbolic scribbles on a paper. Personally, I seek this same relationship with technology – as enabling of artistic thought realization, and as an extension of the mind. Yet obviously, there is no simple mechanical relationship between the movement (key presses), and the musical result, on a computer, as it is with a pen. A computer interface however, whilst also being an abstract representation, is typically programmable – this enables interactions of great fidelity, although costing plenty of time and will to realize. For myself this cost was infinitesimal in relation to the enabling benefits, of an artistic human-computer extension.

Pertaining to the computer extended composer – there are several aesthetic concepts more or less specific to algorithmic composition. In particular concepts requiring or excessively benefitting from computing power, or concepts where the synthesis of human perception and computation is necessary. The possibility to extend or enhance our sense of the world through interacting with technology, is transferable to aesthetic exploration – this, in part, is what will be discussed further in the next section.
4.1 IDEALISM

One question, prompted in particular by heuristics and the prospect of optimal solutions is if there exists only one global solution to a particular well-defined\textsuperscript{15} artistic problem (and it is provable using, for example, the methods presented in this text), is there any inherent aesthetic value to such a solution?

Unique expressions often prove defining for an entire era, such as the silent music of John Cage or the bombastic expression of Ludwig van Beethoven – but is the singular structure generated from the heuristic search, really corresponding to a unique aesthetic expression? Not \textit{invariantly} so. This would imply that there exists a unique aesthetic expression for every possible structure, so that even the slightest change in structure would generate a unique set of aesthetic attributes – arguably, this is not the case. It is reasonable, however, to state that expression is causally dependent on structure (Levinson, 1980, p. 436).

The question is if any distinction can be made between an aesthetic idea and its structural realization, if it is, explicitly, the only structure that could exist (to realize this expression)? Consider the example of an origami artist, seeking to express his/hers emotions by folding regular polygonal shapes in a way that each corner connects the same amount of faces. The artist uses plain paper and struggles to make as many unique geometric bodies as possible. He/She would soon realize that only five solutions are possible (the Platonic solids), and no matter how the papers were folded – no other geometric body would appear. This process could obviously be repeated by other artists, forming the same shapes perhaps using different colours – yet, the originality of their work would likely be questioned.

This origami example suggests some entwinement of structural and aesthetic uniqueness – at least when the structure is \textit{universally} unique and the artistic idea is bound to it (as shape is bound to our perception of the shape). The origami example exposes some additional dimensions of expression (texture, material, size, weight), yet, any work of art deriving from the same structure, would have to eradicate the perception of the geometric bodies, shifting focus on to some other expression dimension – for the work to be aesthetically unique.

Even if no inherent \textit{aesthetic} value (like beauty) pertain to structural uniqueness, the quality of being \textit{unique} is arguably one of great significance in itself. The Platonic solids of the origami artist brings to light a mathematical limit of our physical reality – one cannot go beyond that limit, there is no sixth Platonic body! Even if the purpose of the origami artist was not to explore the mathematical fabric of our universe, the medium (the folded paper) is part of it, and by simple manipulation this hidden world is unveiled. Therefore, an aesthetic value that could adhere to these universally unique structures is \textit{perceptualizing}\textsuperscript{16}. They serve to “perceptualize” the peripherals of mathematical possibility, and thereby expand our sense of the world.

\textsuperscript{15} As in problems that are expressible, and/or solvable through some logical process.

\textsuperscript{16} Structural uniqueness could be discovered, that is not easily made tangible through either visual or auditive art, yet, when an artistic concept is the catalyst – then \textit{tangibility} would arguably be prioritized.
4.2 Optimized Intelligibility

The idea of creating comprehensible scores and trying to maximize the artistic output would not likely be considered an aesthetic standpoint. Yet, somewhat of an antithesis to this, of the aesthetic kind, has gained significant ground – that is, to boost artistic output by obfuscating the score representation – through encumbering amounts of technical and expressional instructions. The movement pertaining to this idea, commonly referred to as New Complexity, does not represent a homogeneous aesthetic ideology – the grouping is more of a superficial one, relating composers who indulge in elaborate rhythms and performative difficulty (Ulman, 1994, pp. 202-203). What is significant is the belief that performative difficulty will improve the expression itself – by demanding more time, effort, and concentration from the performer. Brian Ferneyhough, a forerunner of this philosophy, also stresses that this performative difficulty does not naturally translate to musical difficulty.

What many players often fail to realize is that most of the textures in my works are to a large degree relatable to gestural conventions already familiar from other contexts. What is unfamiliar is, firstly, the unusual rapidity with which these elements unfold and succeed one another; secondly, the high level of informational density in notational terms; and, thirdly, the extreme demands made throughout on the performer’s technique and powers of concentration. (Ferneyhough & Boros, 1990, p. 8)

Whether or not the performative benefits are real is hard to prove, yet, it is quite obvious that a piece rehearsed for six months (Ferneyhough & Boros, 1990, p. 8), should have a significantly better performance, than the same piece, notated simpler, and rehearsed in a week. This is simply due to the engagement of working on anything for that long. New Complexity’s attitude is perhaps more of a socio-political statement, but from the premise of intelligibility it is an odd one.

Besides the hyper technical music, at what point would the complexity of notation be considered ridiculous in relation to the simplicity of the underlying musical structure? The indulgence in complex notation and pitch/time division is an aesthetic position in itself, which is not invariantly combinable with any other aesthetic paradigm.

In contrast to the notation complexity-ideal, it could also be argued that performance quality is bound to other representational factors – such as score intelligibility, that is; the effectiveness by which the symbolic script could be translated in to the appropriate actions and sounds. Obviously, if one acknowledges that music, and music representation are separate to the extent that multiple representations might exist for a single piece, then it is imaginable that one or more expressions and interpretation intelligibility optima exist for each music piece, and that these are bound to specific symbolic representations.

Practically speaking, one cannot be certain that a particular representation is optimal for a certain piece. Imagine, finding a representation that is optimally intelligible – one would need a representation space with one dimension per unique representational sub object. That means that this space would consist of every single combination of symbols that could uniquely represent the music piece. Each one of these combinations would have to be assessed (rehearsed, performed) to get a measure on expression and interpretation intelligibility. Just determining all viable representation combinations would likely be near-impossible.

Constructing a concept around notation principles that benefit the intelligibility of one’s music is obviously benefited by experience of actual work with performers on various pieces. However, studying the psychological, interpretive, and aesthetic information-transfer, through the symbols of a score – could be of interest for researchers in general (in symbolic communication, psychology, or the
4.3 Mathematical-Music unification

Looking back at the question stated in the hypothesis, it has been shown how music can be made using logical processes, numbers, and ratios – but, is music mathematics? A weaker statement; that music can be mathematics, is easier to argue, so this will be the starting point – but first, how is mathematics even defined?

When all human made symbol systems, used to represent or express mathematical statements, are removed – what is left is only cardinals and relationships (Tegmark, 2014, p. 266). A cardinal can be thought of as the “number of” something, such as the number of moons around Jupiter. Relationships or ratios are simply the comparative number representing the cardinal of one thing to the cardinal of another. These kinds of numbers are referred to as pure (Tegmark, 2014, p. 251), in that they do not have a unit, like kilogram, or centimetre – they are purely numbers.

To believe that cardinals exist one must only assume countability, that is that objects can be considered to exist separately from one another. If this is true, then we can group them in a set of things where they can be counted. Then by induction, ratios exist as well – as these only consist of a cardinal divided by another cardinal.

If we strip the symbolic representation from the second movement of Voyage into the Golden Screen (Nörgård, 1968), what is left is only the infinity row – which in itself is a mathematical object, defined by ratios and cardinals. Many of my own pieces have mathematical objects at the core of all musical parameters and processes – the musical object is then indiscernible from the mathematical object. What differs, is merely in representation convention – math is represented by drawing numbers on a board, music by pulling a bow on a string, or writing symbols on a sheet. Obviously the reason for using one representation or the other is fundamentally different – but still, they are discipline-dependent representations, of the very same mathematical object.

In the representation of music, there are always cardinals of events and ratios of pitches, durations, form elements and such. It is not always known why these objects are there to begin with and if composers can compose without mathematically expressible patterns that reveal why they are there – then, it is hard to argue for the general case; that music is math by axiom.

Even if music structure and representation consists of the very same mathematical atoms, it is perhaps reasonable to argue that, for music to be mathematical, there has to be a significantly reduced representation – like the infinity row-formula.

Now, the unisolvence theorem states that; given any \( n \) points, there is always a polynomial of at most degree \( n - 1 \), that will pass exactly through all of these points. This means that any choice of pitch, note value, or proportion set – in fact, any information that could be expressed as a series of numbers – can be written as a polynomial equation of the form: 

\[
    f(x) = c_n x^{n-1} + c_{n-1} x^{n-2} \ldots c_1 x^0.
\]

If there exists a formula for any musical process, expressible by numbers – is math literally in the fabric of all music? Whilst this is true in theory, by Occam’s razor\(^\text{17}\), it is an unsatisfying claim to merely represent

---

\(^{17}\) The scientific principle that among several hypotheses, trying to explain the same thing, the simpler one should be selected.
music in mathematical symbols and therefore call it mathematical – by this statement, almost anything could\textsuperscript{18} be mathematical.

To support a statement saying that a music piece is mathematical, it would be reasonable to prove how the dimensionality of the musical structure representation can be reduced, by expressing and simplifying a mathematical representation of the structure. This does not prove anything about the composer’s intention, it is just telling about the nature of the music structure itself. Imagine, thousands of years from now, an archaeologist unveils a dusty print of \textit{Voyage into the Golden Screen} by Per Nørgård. If any understanding of ancient musical script is still around, then it could be demonstrated how all the internal relationships of the musical structure is derivable from the simple formula: \( f(n) = f(n - 2) + (-1)^{n+1}[f(n/2) - f(n/2 - 1)] \).

That said, the \textit{infinity row} material used in Nørgård’s entire oeuvre, is just an infinitesimally small speck in this everlasting number sequence. The finite stretch of the infinite row cannot explicitly be distinguished as results of the \textit{very formula above}, forever associating it with the composer’s intention. From the archaeologist’s perspective it could just be the result of \textit{chance} or, less likely, \textit{polynomial calculations}. However, by the \textit{disproportional} simplicity and elegance of this mathematical representation – it could be assumed that this association would still be done. At least if \textit{Occam’s razor} stands the test of time.

\textsuperscript{18} And perhaps \textit{is} mathematical. At least if one can prove the Mathematical Universe Hypothesis (MUH) of Max Tegmark, MIT professor of cosmology (Tegmark, 2014, s. 254).
5 Result

5.1 Weights Blows Encounters Motions — For Choir

Autumn 2014, I was given the opportunity to compose a piece for the venerable, Swedish Radio Choir — arguably Sweden’s greatest classical choir, with plenty of experience in, and devotion to, performing contemporary art music. This was my primary project for the first year as a master student at the Royal College of Music in Stockholm (KMH).

Rather than reconciling, by composing a piece that I knew any choir could sing — I tried to use the full potential of the Radio Choir. This meant having sections with solo, and/or highly individualized voices, to having the full, 32 voice, tutti in other sections (and everything in between). It also meant that I could realize my plan of including polytonal and rhythmically challenging expressions, in combination with fractal serialism (of the kind explained in Fractal Algorithms).

In early October 2014, I discovered a fractal sequence, that stood out among roughly six thousand others\(^{29}\) I was considering for this piece. This sequence was peculiar in that it had a very distinct repetition of its three initial values, in direct succession. This only seemed to happen in the very beginning (first six notes of every voice, in Example 13). Another characteristic was that it did not, unlike many other sequences, collapse in to highly repetitive patterns — instead, expanding and contracting in waves.

As is clear in Example 13, the sequence \(S\) is self-similar on \(3^n = 3, 9, 27, 81 \cdots\) scalings — although, imperfect on the very first note. Just like Nörgård’s sequence, or the examples discussed under Fractal Algorithms, this sequence is also offset-self-similar by:

- Every third note of \(S\), creating an inverted version of \(S\) transposed down a step.
- Every third, starting from the second note of \(S\), reproducing \(S\).
- Every third, starting from the third note, reproducing \(S\) transposed down a step.

This would give the constant values \(b = \{-1, 1, 1\}\), and \(c = \{-1, 0, -1\}\), for the sequence-generating function described in Fractal Algorithms. Although, anyone daring enough to try these constants themselves would notice that this does not create \(S\) at all. To achieve the exact \(S\) in Example 13, one would have to supply the first three values \(\{0, -1, -2\}\). In previous examples, only the very first value of each sequence was necessary (typically 0).

\(^{29}\) An illustration of such a dataset of fractal sequences, although a lot smaller, is shown in Figure 8.
The pitch material was extracted solely from the numbers in this sequence, these would then be mapped on both the chromatic scale (mm. 1-56, Ohlsson, 2015), and a diatonic Phrygian scale (mm. 83-111, Ohlsson, 2015).

Initially, I planned to adapt this fractal technique to the lyrics as well\(^\text{20}\). For reasons beyond this text, this proved to not transfer very well to the other material, and I finally went with a different method. The idea of having fractal order in lyrics and melody, is an interesting subject for further investigation, however.

Instead, I went looking for a text that would have the three following, and rather disparate, qualities/subjects:

1. Be about natural science or on some other scientific subject.
2. Be poetically written.
3. Be an old text.

I ended up finding two, completely different authors and texts, that both suited the description above.

The first text I found was by Margaret Cavendish, Duchess of Newcastle-upon-Tyne, a 17th century aristocrat, scientist, and writer. Famous for publishing under her own name in a time where most other women were publishing anonymously, and for being the first woman to, in 1667, attend a meeting at Royal Society of London. The text in question was Of Many Worlds in This World, a rhymed poem, that (as the title suggests) is reflecting upon the existence of a fractal world of sorts, with creatures the size of atoms that are themselves hosts of other, minuscule worlds.

The other text came from an English translation of De rerum natura, originally written by Roman poet Titus Lucretius Carus, approximately 50 BCE. Lucretius writings tells of how our world consists of moving atoms, in a foreshadowing way, prophesizing atomic theory. Yet, it is in no way a scientific text but rather, a colorful didactic poem. I was particularly interested in finding extracts from De rerum natura that pertained to sound, as well as those that thematically compliment the Cavendish text – for example, having some connection to self-similarity, infinity, etc.

---

\(^{20}\) Choosing a relatively simple text, in which semantics could be distorted – by splitting it up in syllables, ordering them in some new way, and using a fractal sequence to pick syllables in a self-similar order
Just like as in a nest of boxes round,
Degrees of sizes in each box are found:
So, in this world, may many others be
Thinner and less, and less still by degree:
Although they are not subject to our sense,
A world may be no bigger than two-pence.
Nature is curious, and such works may shape,
Which our dull senses easily escape:
For creatures, small as atoms, may there be,
If every one a creature’s figure bear.
If atoms four, a world can make, then see
What several worlds might in an ear-ring be:
For, millions of those atoms may be in
The head of one small, little, single pin.
And if thus small, then ladies may well wear
A world of worlds, as pendants in each ear.

Margaret Cavendish
*Of Many Worlds in This World*
17th century

For my mind
Now seeks the nature of the vast Beyond
There on the other side, that boundless sum
Which lies without the ramparts of the world,
Toward which the spirit longs to peer afar,
Toward which indeed the swift elan of thought
Flies unencumbered forth.

... 
Deep in the eternal atoms of the world.

... 
To stablish darkness by his clouds, to shake
The serene spaces of the sky with sound.

Titus Lucretius Carus
*De rerum natura*
Written 50 B.C.E
Translation by William Ellery Leonard

The first step of combining melodic sequence and lyrics, was “dissolving” the dominant text structure of Cavendish’s text. What interested me, musically, in *Of Many Worlds in This World* was not so much its inherited meter, by the rhyme and verse – rather, it was the syllabic variety and density, the semantic playfulness, and of course the content\(^\text{21}\). Yet, distortions of the text’s metric structure, by rhythmical or polyphonic rearrangement, could not be allowed to destroy the proportional relationships of the melodic sequence.

This resulted in me using only one level (scale and offset) of the sequence for the start, and selecting duration values from all possible length three *arrangements with repetition* of three possible note durations: \(\frac{1}{8}, \frac{1}{4}, \text{and} \frac{3}{8}\). This meant that every possible constellation of these three note values would happen only once\(^\text{22}\), resulting in 27 arrangements and 81 values in total.

Not skewing distributions in favor of certain note values – which would likely destroy the hierarchy of the more expressively significant melodic sequence – proved effective in preserving the pitch sequence relations.

I wanted to start the piece using only sopranos, singing the melodic sequence with the rhythmic structure on Cavendish’s text. In this part, as in large chunks of the piece, I used full divisi – having

\(^{21}\) Message and semantics.
\(^{22}\) i.e. \(\{1 \ 8 \ 1\}, \{1 \ 8 \ 1\}, \{1 \ 8 \ 1\}, \{1 \ 1 \ 1\}, \{1 \ 8 \ 3\}, \{1 \ 8 \ 3\}, \{1 \ 4 \ 1\}, \{1 \ 4 \ 1\}, \{1 \ 4 \ 1\} \ldots \{3 \ 3 \ 3\}\).
eight sopranos with individual melodic lines singing the equivalent of $S$ in Example 13 (although, starting a halftone up from the example).

To create the eight voice polyphony from just the melody and rhythm, I merely copied the original melody to each voice, and shifted notes randomly around their original starting points. This guaranteed that the total duration would stay roughly the same, and further muddled the text’s metric. In regards to the fractal sequence; I knew that it was resilient to layering with offsets and scaled versions, retaining shape and gesture – the effective expression of superposing randomly shifted versions proved rather similar.

De rerum natura appear later in the piece, and contrary to the Cavendish text, I now let text influence certain aspects of the melodic sequence. A significant difference to Cavendish’s text, of a rather strict meter of ten syllables per verse (fairly consistently), the meter of Lucretius’ text seemed more unrestrained – of course also a trait of it not having ending rhymes. The significant aspects of Lucretius’ text, I perceived as being the poetic expression, not any particular structural or sounding attributes. This meant shaping the melodic phrasing slightly to adapt to the text, rather than vice versa – an example of this being the initial two verses “For my mind (Now) Seeks the Nature” which I repeatedly assigned to the initial, repeating group of three pitches in the sequence, yet splitting the last note to fit the seven syllable text (Example 14, mm. 57-58).

Example 14, splitting and repeating a pitch in the (inverted) melodic sequence (S), in benefit for the text.

I decided on having a rhapsodic form for the piece – as another balancing factor – here, to counter the inherent continuity and implied process of the fractal sequence. Generally, yet not invariantly, a fractal sequence accentuates every $k^n$ element (in this piece every $3^n$, $n > 0$), by some gestural movement, for example; a culmination of a range expansion.

In Example 13, it is relatively clear that this applies to the sequence $S$, as well – average interval size expand in the onset before every $3^{rd}$, $9^{th}$, $27^{th}$, ... note. I did not consider this kind of emphasis to be bad – but, accentuating the melodic (fractal) process – in a way similar to Voyage into the Golden Screen (Nörgård, 1968) or Symphony no. 2 (Nörgård, 1970) – I felt, could distract from other non-melodic developments in the piece.

The rhapsodic form containing various expressions and representations of the fractal sequence, had the desired effect that, even on contiguous segments where the sequence process is unfolding cross parts, it was making it possible to bring forth varying expressions and techniques for each part – whilst conserving some coherence through the, now decentralized, sequence development.

---

23 The melodic development is, undeniably, the most significant structural element in the referenced Nörgård pieces.
5.2 **KORT ETTA – FOR ACCORDION AND ACOUSTIC GUITAR**

I was asked to write a short piece for a contemporary music festival in Milano 2015, by accordionist Francesco Moretti, and classical guitarist Michael Barletta. The deadline was in about a month or so, therefore I decided on just two, or three techniques that would be both fun to experiment with from a composer’s perspective, and challenging, yet rewarding to perform.

The first concept, came to me via an article that was given to me by composer, and composition teacher, Lars Ekström – who sadly, and very suddenly, passed away later that very semester. He handed me the article in the hallway at KMH, the article, a master thesis by Huw Belling (Thinking Irrational, 2010), on the music of Thomas Adès – was specifically dealing with the role of meter, in complex rhythmic polyphony, and it illustrated how irrational time signatures\(^{24}\), could be exploited to possibly simplify notation in polymetric music.

I had no particular interest, at this time, in the aforementioned application. Rather, I felt that this concept would be better applied to single meter music and, in particular, as a way of notating, something similar to what jazz musicians call, *swing*.

My principal experience of swing was not from jazz music however, but Scandinavian folk music. Studying folk music back in 2008, I became aware of *slängpolska*, a specific variant of *polska* which is a triple time dance, common to parts of Scandinavia. Slängpolskan would, in some Swedish regions, be performed, not in *straight* triple time, however, but with a short first beat and (in some cases) with an extended second beat (Näslin, 2008).

The expressive potential for time-expanding/contracting beats, even outside the folk music context, I found inspiring for this piece. The final realization was a synthesis of the concept found in Adès music and the *slängpolska*. Defining a compound irrational time signature, where each beat is of different size, I could notate tuplets in straight note values (see Example 15).

\[ \text{Example 15, Patrik Ohlsson, Kort Etta – ”In sync”}^{25}. \]

Tempo indication is for the second beat.

The particular choice of time signature was the inspiration from *slängpolska*. Interpreting the time signature: \(1_6 + 1_4 + 1_5\), the first beat is the short beat, second beat the long beat, and third in the middle. If this was written in straight 16\(^{th}\)s the first beat would be in tempo quarter note = 60, second beat 40, and third in 50 BPM.

\[^{24}\text{That is, time signatures that does not have powers of 2 in the denominator, such as: } \frac{4}{6} \text{ or } \frac{7}{10}.\]

\[^{25}\text{Irrational compound meters are supported in the Lilypond notation software, producing correct notation and playback. Any irrational compound time signature could be set like this:}\]

```latex
\begin{verbatim}
\compoundMeter '((1 6) (1 4) (1 5)) \set Timing.beamExceptions = '\'
\set Timing.baseMoment = #(ly:make-moment 1/60)
\set Timing.beatStructure = '#(10 15 12)
\tuplet 6/4 \{ b'4 \} b'4 \tuplet 5/4 \{ b'4 \}
\end{verbatim}
```

---

40
Even if this is a direct translation of the *abstract* beat concept of some versions of slängpolska, I did not have any specific region or musician in mind for the choice of beat factors. Rather, I was trying to maximize the perceptual “bouncy ball”-effect of the slängpolska. Similar to throwing a rubber ball in the air that bounces as it hits the ground and then gradually bounces less and less (only to be thrown again). This was the intended effect of the time signature and its relation to slängpolska. The first beat corresponding to the *flight* of the ball, second beat the primary *bounce* of the ball, and third beat a lesser but faster bounce.

The second significant concept in this piece was the derivation of the pitch material. I decided on using twelve-tone rows for two reasons:

1. This posed some particularly interesting challenges in adapting the material to guitar.
2. This fit in well with the triple time being divisible into twelve “16th”-notes.

The absolute pitches themselves were however of less importance, as will be explained shortly.

I wished to emphasize the rhythm and beats, by not having the two voices diverge too much. To achieve this, they were playing the same pitch classes in approximately the same range, with the accordion sustaining certain notes (see Example 16).

There are plenty of, both low and high, natural harmonics in the guitar part. These will theoretically produce the correct pitch in the twelve-tone row, but when played has a very percussive and distinct sound, and not much tone. The sounding range of the guitar part was also large and varied enough, to effectively abolish any melodic association. The accordion, whilst smoother and more melodic, still primarily had a rhythmic function.

As there was relatively little time to compose the piece, and as I had not written for classical guitar before, a major uncertainty was – regardless of material – if preparing material for the guitar would be too time consuming to do anything beyond the trivial.

Rather than dwelling on this, I tried to find a way of expressing the specific challenges concerning the guitar, as a combinatorial problem. My primary concern was in writing chords or intervals that would require impossible fret jumps. This was particularly worrisome as I wanted to have a relatively fast and rhythmical guitar part.

The heuristic algorithm I designed would check all possible ways of playing each note, so that a minimum of frets would be traversed in total. I made several versions of this algorithm simply referred to as “guitaroptim”, but finally settled for this behaviour:
1. A solution would consist of sets of string and fret indicators, representing a unique way of playing the given pitch class (meaning that any octave was allowed).

2. From such a solution, the number of frets having to be traversed could be counted. This would be the penalty number that the algorithm tries to minimize.

3. Include the option to simultaneously try to maximize the number of ringing strings at all times.

4. Include the option to prioritize solutions close to a certain area on the fretboard, for example near the head of the guitar.

5. Include the option to allow natural harmonics.

The absolute pitches of the twelve-tone rows were not significant, but enabling the guitar to play the pitch classes and at moderately high speed, was. Me and my friend Jesper Nielsen, a skilful guitarist and composer in his own, tested some of the output from the algorithm to see if it produced viable results. The output from an early version of the algorithm could look like in Example 17. This was however, before any options regarding open strings, harmonics, or fretboard regions were implemented.

Initially the algorithm was inclined to put pitches quite far down on the fretboard, which was not always ideal, but, by testing extensively and iteratively improving the algorithm, it became quite a powerful tool – with has uses outside this specific piece\textsuperscript{26}.

\begin{example}
\includegraphics[width=\textwidth]{example17.png}
\end{example}

\textit{Example 17, test output from an early version of the guitar algorithm (guitaroptim), with string indicators above each note. The second staff notating each string separately.}

\textsuperscript{26} E.g. transcribing music for guitar.
5.3 **KOLOKOL – FOR CHAMBER ENSEMBLE**

For the last year of my master studies in 2015/2016, I was planning on composing a piece for seven musician Pierrot ensemble, **Norrbotten NEO**. I had written two pieces prior to this for NEO, so I was confident that they would be able to play almost anything I threw at them. Still, the challenges of what I wished to compose would not be in virtuoso rhythms or gestures – rather, in the expression of dynamics and intonation.

I wanted to create a *spectral* piece of sorts, and at least to find a way of having the collective sound of the ensemble being united in one *body*, external to the specific quality of each instrument. This led to the development of the tools described under *Spectral Composition*, many of which were specialized for the instrumentarium of NEO.

The *source* sound (see flow chart in Figure 7) for the majority of all the pitch material, is a large *funeral church bell*, from a freely available recording I acquired online. Which church or cathedral it was taken from was unfortunately not documented, but the quality was good. I did not need more than a second or two however, as I was primarily interested in the static spectral attributes of the sound.

![Figure 9](image)

*Figure 9, Spectrum of first 44100 samples of the funeral bell sound.*

The amplitude spectrum and partial frequencies of the bell (Figure 9) would not be analysed directly, instead this information was passed on to the heuristic algorithm for comparison against the generated instrumentations (see *Spectral Composition* for details). The bell sound was fairly low pitched, with a hum note on 104 Hz, the most prominent partial being 227 Hz (a slightly high A under middle C).

Now, the genetic algorithm that was searching for instrumentation solutions, is initialized to a random state. This meant that every time the algorithm ran, there was the potential for new instrumentations to be discovered. I used this property to create an entire chain of chords for the strings and woodwinds, the difficulty being to guarantee persistent high quality results. To do this I increased the GA population count and let it search for longer (see *Genetic Algorithm* for details), in the end the total time of the multiple GA runs from start to finish would take about 30 minutes. This produced the full instrumentation for strings and woodwinds, throughout the entire piece.

It is important to note that calculated fitness value cannot invariantly be trusted as a good measure of source similarity. Yet, in the process I noticed that, using a specific fitness function, it is possible to estimate a fitness threshold where both the computed and the perceptual results are good. This would of course be based on my own perception, but for the sake of the piece, I was very picky.

After days of testing and refining, the results of the GA focused in on a fairly small sub space of combinations and variations of similar-looking instrumentation solutions – this, in combination with me *perceiving* that the similarity to the source sound was good, persuaded me that these in fact were close to a *global* optimum.
The chunk of orchestrated chords generated by the GA had one major issue however; they were all fairly similar. In the random order they came out, there were instruments playing the same note repeatedly ten or more times – for the sake of the musicians and texture, this was not ideal. Besides that, quarter tones were also allowed, so the random chord order meant that there were passages where a single musician, would play multiple repeated quarter tones – not having any individual reference pitch for bars on end. These issues, I realized, could also be stated as a combinatorial problem – the orchestrated chords could probably be sorted so that no consecutive quarter tone would occur in each part, and per-part unison intervals could be minimized to ensure a constant textural movement, inside the emulated bell-like sound.

A backtracking implementation was constructed with the aforementioned objectives. The no quarter tone repetition-rule was implemented as a constraint, whilst the minimize partwise unison intervals-rule was implemented as a penalizing score system (see Backtracking Algorithm, for the details on backtracking). A solution would then be a particular combination of indices 1⋯number of chords, that has no repeated quarter tones in any part, and as few repeated per-part unisons as possible too.

I separated the full list of 60 chords in to four segments of 15 chords each, and tested two variations of the sorting-algorithm on each segment. Variation one, was not only minimizing per-part unison interval count, but also maximizing total per-part interval sizes, to create as much in-texture movement as could possibly happen. Variation two was having as much stepwise motion per voice as possible (whole or half tone intervals). Even with these variations, the material was so uniform that this only resulted in a subtle textural fluctuation – yet, significantly improved interpretability.

In similarity to Weights Blows Encounters Motions, I distorted the onsets of every instrument so that the end of one chord, and start of another would be blurred. The woodwinds were also delayed with an average of a half note, with roughly half the note lengths of the strings. This to guarantee that they had plenty of time to breathe in between onsets, whilst further distributing the content of each chord over time, creating more of a continuum than a series of chords (see Example 18). The only limit on the dispersion of onsets, were restrictions guaranteeing that, at some point, all notes in each chord would sound at the same time.
Also visible in Example 18, are the dynamic curves – the only “extended” notation in the piece besides the quarter tones. I rarely use extended notation in my scores, as is apparent from my arguments in Optimized Intelligibility. There have to be some meaningful expression or performer benefit for me to elect to do so, and in this case there were!

I had problems reconciling with the (in my opinion) limited ways of expressing highly localized changes in dynamic, with classical notation – in particular for shaping the expression of short phrases or notes. I had done some experiments with alternative dynamic hairpins for a workshop with the Swedish Radio Choir the year before, but then resigned to using normal hairpins in the end. This time, with KOLOKOL, I was certain that there were some obvious benefits and expressive potential in extending dynamic notation.

Besides that, I also felt that there were some, largely overlooked, effects on the perception of tempo – connected to the dynamic shape or expression of a note.

I conducted some experiments, generating sounds with an artificial dynamic curve progressively shifting from distinct attack to gradual crescendo (see Example 19). Besides that, I tried changing the curvature from exponential, to linear, to logarithmic – meaning that the peak of the curve would be approached in a soft, sharp, or even way. I found that these variations had drastic effects on the way I perceived the tempo, even though a synthesizer was playing the exact same note length. Sharp attacks and distinct peaks generated a sense of rush, whilst more bulging curvatures gave me the impression
of the notes dragging along, slower than the actual tempo. Even if this was only my perception27, I saw the expressive potential of using this tempo of dynamic shape, in KOLOKOL.

The actual curve was done by writing a Scheme28 extension in the Lilypond notation language29. The exposed parameters for each curve are:

1. Incline curvature point, values between 0 and 1. This determines the shape of the incline curvature to the peak of the curve. A value of zero for the curvatures Y-position would mean a highly exponential growth of the curve, whilst a value of one would mean a very fast growth which then deaccelerates when approaching the peak.
2. Peak X-position, between 0 and 1. This determines where on the horizontal axis the peak is.
3. Decline curvature point, same as “1.” but for the declining segment of the curve.
4. Optional, minimum and maximum, musical dynamic text. The dynamic range was indicated on the left of the curve, to indicate the expressive characteristic before the performer started playing the note.

In KOLOKOL the curve peaks were shifted back and forth, in a sine wave shape. The frequencies of this movement were in harmonic relationship among voices (1,2,3,4⋯), yet, the way I did this had no particular significance to the structure of the piece. The effect I tried to create, was constant homogeneous variation of expression and dynamic shape. The process itself was completely independent of everything else.

Finally, the piano and vibraphone had a slightly different gesture and pitch material. The recurring part found in the beginning (mm. 1-6, Ohlsson, 2016), was derived from the sounds of a tubular bell instrument, playing a chromatic scale from G, under middle C, to the F# over. By feeding these notes through the same analysis method used for the funeral bell, a kind of “Shepard tone”30 occurred. The pitch of the tubular bells was rising and rising, but the algorithm found solutions for the piano and vibraphone where higher partials now overlapped with the source spectrum better than some lower partials had done before – causing them to stay in a fairly narrow range (with some offshoots).

27 I suggest anyone to test this for themselves.
28 As in the Scheme programming language.
29 Lilypond code required an override of the Hairpin-object’s stencil:
\override Hairpin.stencil = #(curvedar-hairpin 0.25 '(0 . 0.25) 0.75 '(0 . 0.25) "p" "ff"
30 A sound illusion creating the sense of pitch is perpetually rising or falling (Braus, 1995).
The algorithm had the option to choose up to 6 notes per chord for the piano, and 4 for vibraphone, in the matching against each bell note. The resulting number of notes was then distributed evenly over a half notes length, resulting in the tuplets shown in Example 20.

This material would disappear after it was played in full at the beginning (a kind of *exposition*), and then additively be brought in again, one half note at a time later in the piece. This was done to balance attention between all ongoing processes. The regularity and slight accentuation of the half notes stood out in relation to the pseudorandom positioning of woodwind and strings – by not overexposing this material and dozing entries out in a processual manner, an appropriate balance was reached.
6 DISCUSSION

In this text it was demonstrated how some heuristic algorithms work, how they can be applied to artistic and practical problems relating to music composition, and what some of the aesthetic implications are – working with a computer as an assistant or extension of the creative mind.

This should, of course, only be seen as an introduction to these subjects, there are an unthinkable number of ways of working with just heuristic algorithms, that is not covered here. Hopefully, these descriptions and methods will be useful for those interested in starting with or expanding their knowledge in, algorithmic, or computer assisted composition, and that the discussions were thought-provoking, for anyone interested in this subject.

The future for algorithmic composition in general is developing quickly. I believe that there are some revolutionary artistic opportunities emerging within the computer science subfield of machine learning. In particular, deep learning – a way of teaching a computer complex relationships between seemingly unrelatable data, such as the pixels of an image and what or who the image is portraying (Erhan, Szegedy, Toshev, & Anguelov, 2014). This is done through training an artificial neural networks – a simplified computer abstraction of biological neural networks – on massive data sets of labelled images.

Deep learning has already shown some potential for visual art (Gatys, Ecker, & Bethge, 2015), but has not had the same breakthrough in music composition yet – even if it is starting to catch on.

The future development for heuristic algorithms in music composition is where I personally am the most active. I have plans on doing an artistic research project or dissertation, involving heuristic algorithms that will cover every stage from initial conception to interpretation and rehearsals.

The project will involve analysing audio data using physical modelling synthesis (Smith, 2010), rather than using sample libraries (as was demonstrated in the Spectral Composition part). In regards to this, I will focus on how specific interactions with the physical instrument models could be interpreted as grips or instrument specific instructions – that ultimately could be expressed in actual notation. From this, there is the potential to use heuristic algorithms to interact and generate audio data, in a similar fashion to the model in Figure 7, and comparing to a source sound or ideal.

The most significant difference is in the input variables of the GA, prior, these were simply row indices in a pre-recorded matrix of audio data, now they could be changed to represent actual interactions with a physical instrument. Such as: percent covered of a tone hole on the clarinet (Smith, 2010, p. 422), or roll angle of the cello bow to the string (Smith, 2010, p. 429) – the algorithm translates a complete set of interactions, to audio data – through the computer models of these instruments. From this the audio data could be compared to another sound, or evaluated by some supplied objective.

A significant part of this project would be testing and refining this process, in collaboration with experienced musicians. This, to ensure that the modelled behaviour is transferable to a real instrument, and to evaluate the notation of such extended performance instructions. Finally, it is important for the last stage, the objective of the heuristic. This, to see that it is fulfilled to a satisfying degree – producing the aesthetic and perceptive output that was expected.

---

31 Including fields such as: artificial intelligence, artificial neural networks, and deep learning.
32 Such as with this amusing, and impressive network by Bob L. Sturm and João Felipe Santos, trained on 23,000 traditional Irish songs – it has produced almost 36,000 new tunes as of this moment.  
http://www.eecs.qmul.ac.uk/~sturm/research/RNNIrishTrad/index.html
From an artistic standpoint, it is particularly exciting to consider the applications of this model – imagining a sound ideal (e.g. spectral similarity to a bell, overtone richness), using several computer instrument models to generate instructions and synthesized renditions for any particular instruments, and directly receiving *intelligible* music notation from this.
7 REFERENCES


OEIS. (2016, April 9). Retrieved April 9, 2016, from A004718: https://oeis.org/A004718


Example 21, Per Nørgård, Symphony no. 3, mm. 61-69.

One single discrepancy is found in the very last note of the sequence – almost like a protest against the infinity row.
8.2 **PRUNEDSEARCH C++ CLASS**

```cpp
/*
   RECURSIVE PRUNE SEARCH ALGORITHM
   An extended version of the backtracking algorithm.

   Feel free to use and modify for your own program.
   The code does is distributed without any warranty at all.
   Patrik Ohlsson
*/

#pragma once

/*
   Necessary headers.
*/
#include <iostream>
#include <vector>
#include <limits>

template <class T, class V>
class RPruneSearch
{
public:
    /*
       Fitness limit is initialized to whatever minimum value
       of the chosen data type.
    */
    RPruneSearch() : fitlimit(std::numeric_limits<V>::min())
    {
    }

    /*
       This is called from the implemented child class to
       start the search.
       First argument is the set of sets of values to go
       through.
       Second argument is the length of the solution.
    */
    void Run(const std::vector<std::vector<T>>& vals, int N)
    {
        this->N = N;
        this->vals = vals;
        this->bestscore = std::numeric_limits<V>::max();
        RunRec(std::vector<T>(), 0);
    }

    // Abstract header for constraint function (implemented by child)
    virtual int consf(const std::vector<T>& state) = 0;

    // Abstract header for scoring determination function (implemented by child)
    virtual bool doscoref(const std::vector<T>& state) = 0;

    // Abstract header for solution scoring function (implemented by child)
    virtual V scoref(const std::vector<T>& state) = 0;

    /*
       Abstract header, with default implementation, of
       outputting solutions. (overriden by child)
    */
    virtual void outf(const std::vector<T> currentState, const V score)
    {
        for (int i = 0; i < currentState.size(); i++)
        {
            std::cout << currentState[i] << ", 
        }
        std::cout << std::endl;
    }

private:
    // Values of current solution.
    std::vector<std::vector<T>> vals;
    int N; // Final solution size
    V bestscore; // Best score overall
    std::vector<T> bestset; // Best solution overall
    V fitlimit; // Fitness limit property
};
```
// Internal recursive algorithm for performing backtracking.
void RunRec(std::vector<T> acc, int n)
{
    for (int i = 0; i < vals[n].size(); i++)
    {
        std::vector<T> s(acc.begin(), acc.end());
        s.push_back(vals[n][i]);
        if (consf(s) > 0)
        {
            continue;
        }
        else if (doscoref(s))
        {
            V score = scoref(s);
            if (score <= bestscore)
            {
                bestscore = score;
                bestset = s;
                outf(s, score);
            }
        }
    }
}
if (bestscore <= fitlimit)
    return;
if (s.size() < N)
    { RunRec(s, n + 1); }
}
8.3 KOLOKOL SCORE
See attachment.