Automating algorithmic representations of musical structure using IGME: The Interactive Generative Music Environment

Samuel J Hunt, Tom Mitchell, Chris Nash.

Corresponding author email address: Samuel.hunt@uwe.ac.uk

ABSTRACT (228)

In this paper we explore the recreation of existing musical compositions by representing the music as a series of unique musical bars, and other bars that can be replicated through various algorithmic transformations, inside the Interactive Generative Music Environment software, or IGME. This re-composition approach is intended to explore whether the pre-existing music could have been created using the processed based approaches offered by the IGME software. If music can be expressed by algorithmic processes then we propose that original works of music can be expressed or created in the same way. Such a justification can provide a rationale for creating the unique compositional processes and workflows that IGME affords to those looking to compose with generative and algorithmic music techniques, and avoid many of the pitfalls of generative music.

Music can be imported into IGME and automatically analysed to find unique bars, and bars that have been transformed from them. The overall timeline can be visualised to quickly demonstrate the structure of the music, using colour to differentiate unique musical ideas, and arrow-arcs to show the relationships between different parts. Such a process reduces the overall entropy of the music data and provides an educational insight into macro level music structures. Each of the techniques are explained and examples given. In addition, data sets have been pre-computed for several genres of music, showcasing the distribution of different types of techniques.

1. INTRODUCTION

IGME (the Interactive Generative Music Environment) is a music sequencer that supports the exploration of generative and algorithmic music techniques. Unlike code or patch-based systems, it provides an easy to use interface for exploring generative and algorithmic music techniques, that is built on common music software paradigms. Many existing generative music systems use workflows that are not familiar to non-programmer music composers. A more detailed overview of IGME (previously named IGMSE) is given in (Hunt, Mitchell and Nash, 2017 and 2018). The core design principles of IGME are:

1. Integrates algorithmic techniques for musical composition inside familiar score editing and music sequencing workflows.
2. Provides full version control, for revisiting and comparing material.
3. Uses graphical controls (WIMP) rather than code-based interfaces.
4. Takes a modular approach to composition, while retaining a linear timeline.
5. Uses a multi-layered assembly stage that assembles the final score from individual parts.

An impediment of generative music systems is that they often fail to form high level structure, and are often highly stochastic in nature (Hunt et al, 2017). This is seen in large existing systems such as Jukedeck (Langkjaer-Bain, 2018), Aiva (Zulić, 2019), and Melodrive (Collins, 2018) focusing on replacing the human completely, with cutting edge machine learning. The, overarching aim of IGME is to create a system that supports human and computer composition. The aim is that by combining the best aspects of generative music with the careful control of a human operator that more structured forms of generative music can be created. Therefore, it is worth considering how much of the music should be unique and how these ideas should be developed through the piece. Therefore, the principal aim of this research is to assess whether existing music (composed by humans) can be encoded and represented by algorithms using the tools afforded by IGME. From this we can understand what techniques other general music sequencing software should adopt, for supporting interactive generative music.

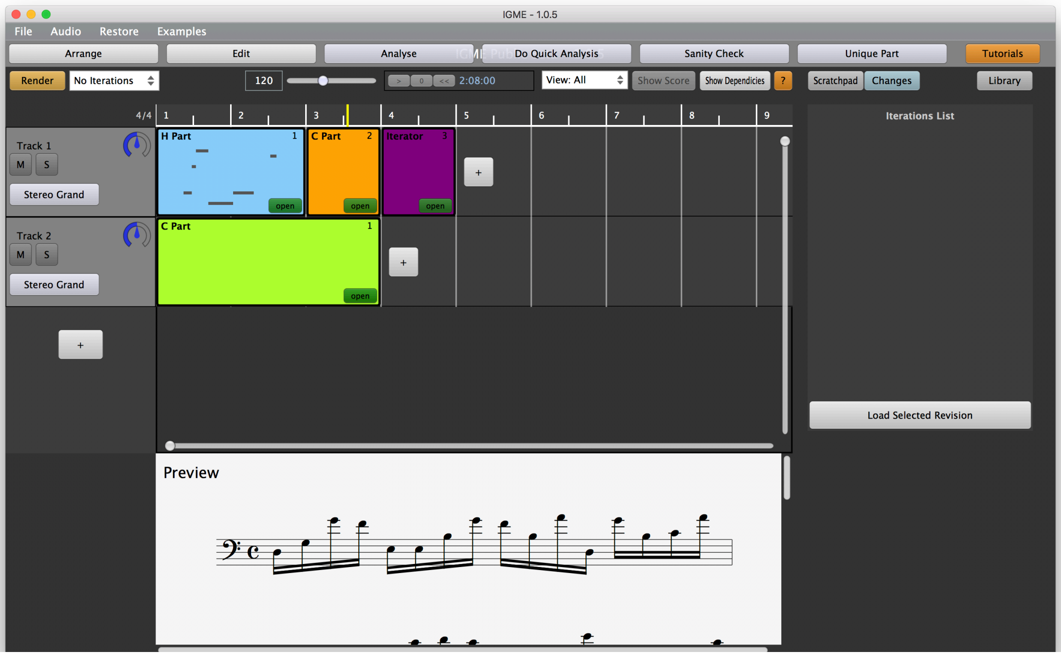


Figure 1: Arrange view inside IGME

IGME considers composition in terms of three distinct musical parts: human created content, computer generated content, or a mixture of both. A part within IGME is similar to the idea of a MIDI clip in other music software (the differences are discussed in section 3). The IGME program is divided into two main views: the arrange view (Figure 1) and edit view (Figure 2). The arrange view focuses on arranging and sequencing individual parts (e.g. MIDI clips), using design principles found in other common music sequencers. The edit view (or detail view) allows the user to edit the individual music sequences, and/or specify the algorithmic effects for each part. A range of algorithmic effects are implemented by IGME, that can either augment human composed music, or generate computer created music.

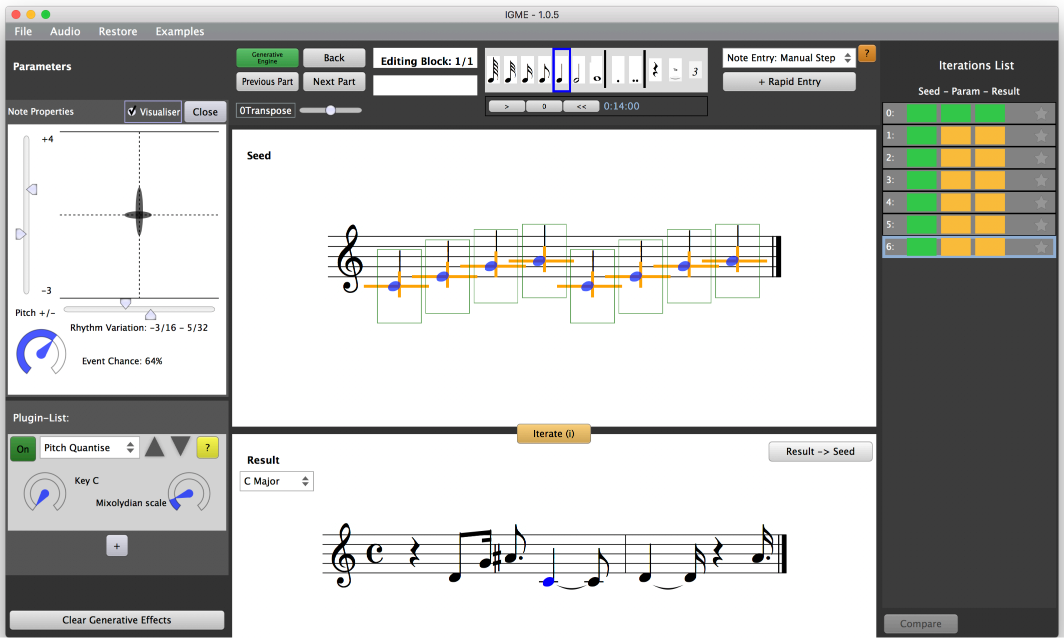


Figure 2: Edit view inside IGME.

The aim of this research is to look at how existing pieces of music can be represented and encoded using the IGME environment. The paper therefore considers two differing but similar research tasks. The first explores specific examples of music in detail, whereby the music can be represented more closely by; original ideas, and a series of transformations. The second looks at automatically analysing larger datasets to provide generalised metadata about musical structure. The concept of expressing music as patterns and processes has been explored previously by Nash (2014) using the *Manhattan environment,* and shares a number of parallels with this work.

The main body of text is broken down into five sections. Section 3 explores some of IGME's unique features that are crucial to this research. The various techniques for transforming and relating groups of musical parts are explained in section 4, each shown with examples of this process. Following this section 5 discusses the data pipeline for computing analysis automatically. Section 6 examines the output of a complete analysis of a single song, revealing insight into its musical structure. Lastly datasets are computed for various types of music and these are summarised in section 7.

2. BACKGROUND

In general, musical structure is often composed by following some set of *rules*. Moore (2001, p 433) notes that it is these rules that characterise different genres of music. Rules can be understood in terms of stylistic choices that determine or constrain elements of the music which are often inferred rather than formally defined (Herremans and Sörensen, 2012). For example, most tonal music is constrained to a given musical key (Temperley, 2007). However, certain genres of music, for example music composed using species counterpoint have well defined formal rules (Fux and Wollenberg, 1992). Furthermore; Minimalism, Serialism and other process-based forms of music are a genre of music that focuses on representing music composition as a process or series of algorithms, the research here considers music that does not identify as belonging to such a genre. In a general sense, this research attempts to find patterns in the structure of composed music.

Lerdahl and Jackendoff's (1983) generative theory of tonal music (GTTM) organises music into a four-level hierarchy; motives, phrases, periods, and larger sections. Several authors have attempted to automate the GTTM, notably Hamanaka (2006), however a full automated implementation of the GTTM remains unexplored. Rothfarb (2010) notes that the phrase level generally considers music to be 4 measures in length. The research here considers segmenting music mostly into motives, where the smallest division of hierarchy is fixed to a single measure.

The *Manhattan* music programming environment (Nash, 2014) uses a pattern-based sequencer paradigm in which code is situated in repeating musical patterns to manipulate the music during playback, as an explicit interaction model that considers music as the synthesis of patterns and processes, sympathetic to the key roles of rules and repetition throughout musical practice and history. Through a series of studies (études), the tool has been used to encapsulate pieces across various genres and eras of Western music (baroque, minimalism, romantic, popular, etc. – from Bach to Stravinsky to Hendrix) – recomposed as expressions of arranged patterns (musical seeds) and transformative or generative processes (procedural code). Used currently as a pedagogical tool, this model is designed to foster analytical thinking in students through manual analysis and reinterpretation through code, but is also the basis of other work on automated analysis and the practical exploration of data models in music.

Formal frameworks for analysing music have existed for a long term, notably Schenkerian analysis. Despite work by Marsden (2010), traditional Schenkerian analysis has only had limited success in being automated and remains too computationally expensive. More cutting-edge research in machine learning has explored automated music analysis in other ways (Huang, 2016). Deep learning, despite its promises of delivering exemplar solutions to the problem, provides a ‘black box’ approach that provides little metadata explaining the process, which is both important and useful for music. Rudin (2019) notes that more emphasis should be placed on making interpretable models for big data, rather than using black box algorithms.

Notable researchers, such as David Cope (Cope, 1991, 1996), has produced multiple works in the area of list programming for generating music. Starting with smaller fragments of music and combining them through various procedures to produce larger works. This work takes the opposite approach (starting at the end result, and working back to the start).

3. TERMINOLOGY:

A part in IGME is very similar to a MIDI clip in other musical sequencing software, however an individual part in IGME is made of 3 distinct sub-components. These are the seed, parameters, and result. The seed is the musical material that is edited by the user. The parameters are a series of processes (effects and algorithms) that are applied to the seed, to produce a result. Note the result is the musical material that is audible to the user. Without specifying any parameters, the result is identical to the seed. The seed material can also be supplied from a previous part’s result (discussed below) or by a seed generator (generative effect).

A reference part in IGME is whereby the content of a given part is referenced (or taken) from another part. In this relationship the seed material of a given part B is specified from part A’s result, therefore part B is referencing part A. Note that the reference part can have exactly the same content as its parent, or modify it (through various transformations). Looking closely at the score in Figure 3, the second bar is a direct duplicate of the first bar. Therefore, inside IGME part 2 could be notated as a part that references (in this case) part 1 (Figure 4 middle). This representation shows more explicitly the structure of the music. This could also be expressed as a *repeat* as shown in Figure 4 right. However, this common music sequencing paradigm fails to work when a bar of music is repeated in a non-consecutive manor, as shown in figure 5. Referencing can be used to represent musical structure in a more visual way, and is therefore argued as crucial concept for this work.

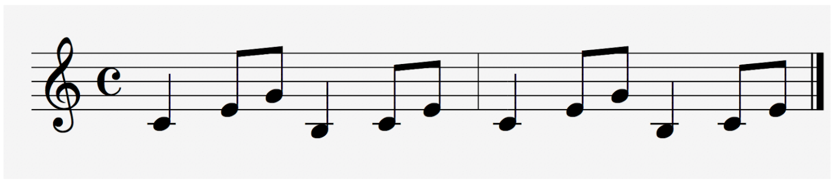


Figure 3: musical score.

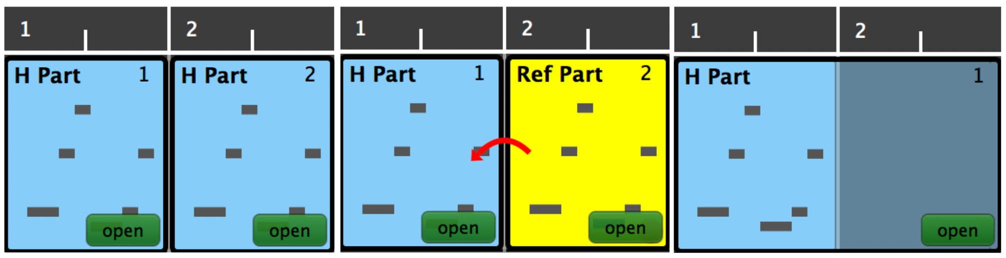


Figure 4: Left: Two parts the same. Middle: Part 2 referencing part 1. Right: Part 1 repeated once.

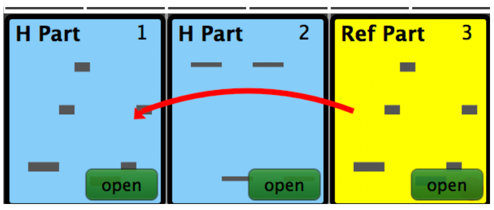


Figure 5: Part 3 is referencing part 1.

With reference parts there is a one-to-many mapping, whereby an individual part may be referenced many times, in figure 6, part 1 is referenced by parts 2, 3, and 4.

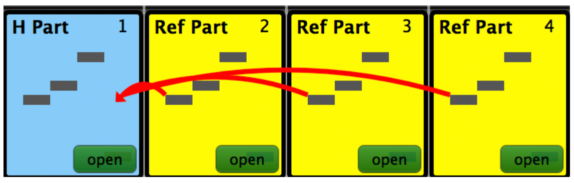


Figure 6: One part being referenced multiple times.

4. TECHNIQUES

IGME is a music composition environment, and consequently there are a range of tools for generating music through both stochastic and algorithmic techniques. Many of the stochastic techniques are not relevant in this research, as existing music cannot be expressed statistically, as the musical decisions would have been fixed during composition. Instead, a subset of the tools offered by IGME are used to determine and express musical structure. Namely the following techniques; duplication, transposition, transformations, arpeggiation and note-mapping can be automated. Each of these tools will be described in the next section alongside a working example. All of these techniques (except arpeggiation) make use of part referencing, whereby a part's initial content is taken from a previous part and then has some further process applied.

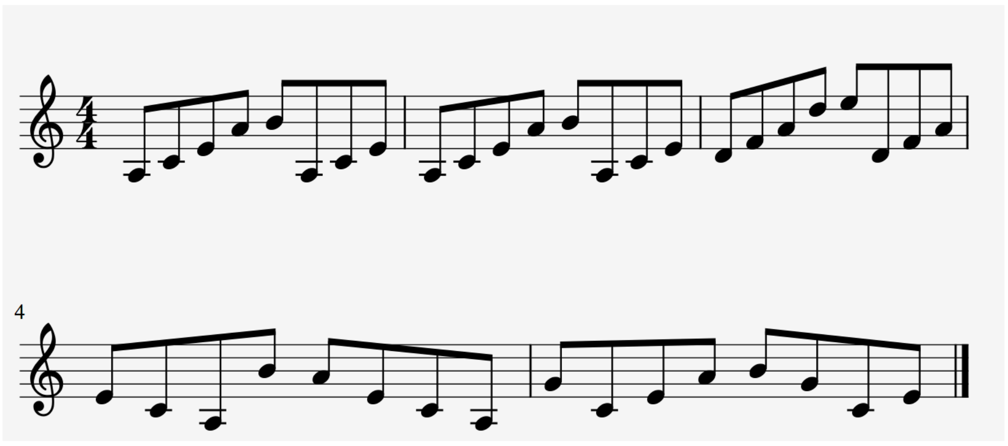


Figure 7: Score for the examples shown in this section.

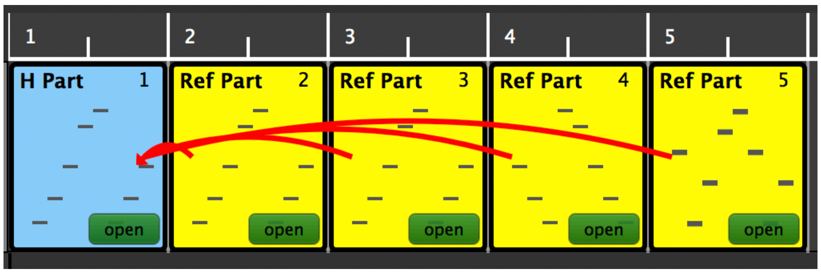


Figure 8: IGME part representation of the score in figure 3.

4.1. DUPLICATION

Is the technique of duplication (repeating) previous musical ideas, these are expressed in IGME through simple references (previous section).

4.2. TRANSPOSITION

Transposition is simply the process by which all notes are chromatically transposed by a given value. In Figure 7, the second third bar of music is the first bar of music repeated and transposed by +5 semitones. Therefore, more structural semantics can be shown, if this is expressed as a reference part with a transpose process applied.

4.3. TRANSFORMS:

A transform process applies one of 4 simple procedures to a given part, these are; retrograde (playing the sequence backwards), inversion (inverting the pitches), retrograde-inversion (both together), rotate left or rotate right (note that rotating left and right together is nullified). In Figure 8 part 4 is set to reference part 1 and then have the retrograde plugin added.

4.4. NOTE MAPPING:

Note mapping is analogous to find and replace. A note map is simply a mapping that defines what notes in one sequence are replaced by in another sequence. For example, in figure 7 the pattern of notes in bar 5 is similar to bar 1, however the 2 A4's are replaced with G5's. This is expressed with a reference part and the note map plugin applied (Figure 9). With this process all occurrences of note A4 are replaced.

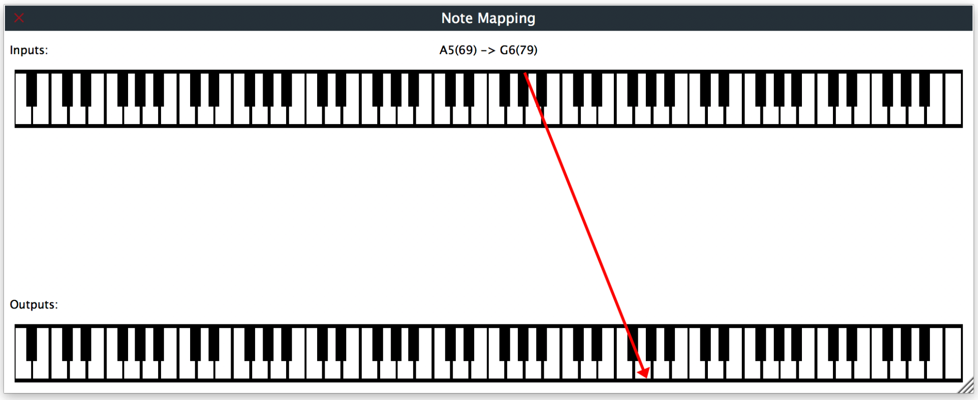


Figure 9: note mapping interface.

The famous guitar hook introduction of Guns N' Roses' *Sweet Child O' Mine* (1987), provides ample opportunity to demonstrate the note map technique. The arpeggiated sequence of notes in bar 1 repeats in a block of 8. Bars 3 and 4 take the initial idea and replace the low D with an E. Bars 5-6 substitute the same note with a G, a score is provided in figure 10. Applying the note map process for bars 3 and 5, and applying reference duplication for bars 2,4,6,7, and 8, we end up with just 1 unique part, and 7 bars of transformations. This is visualised in Figure 11.

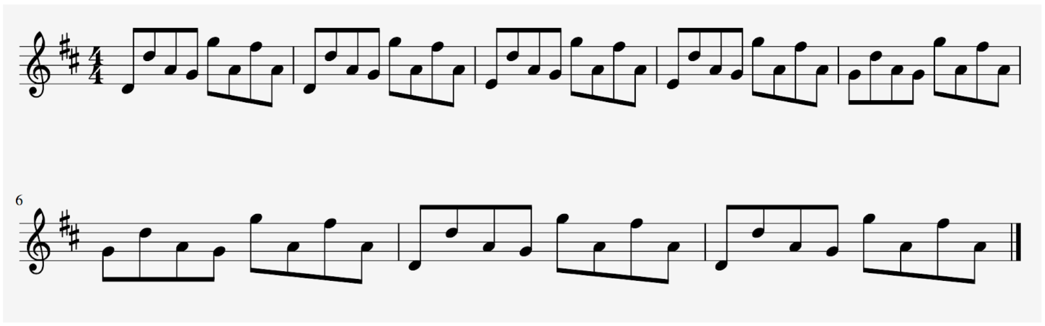


Figure 10: First 8 bars from Sweet Child O' Mine.

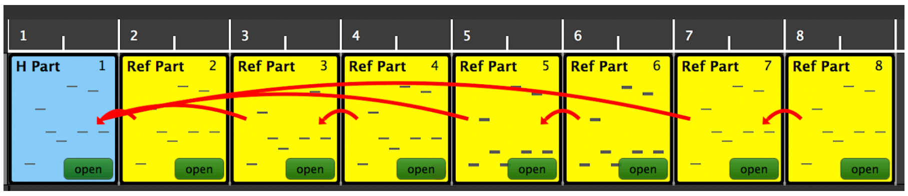


Figure 11: IGME representation of Figure 14.

4.5. ARPEGGIATION:

Unlike the other techniques discussed so far, arpeggiation attempts to reduce the overall musical data in a single part by expressing it as a collection of pitches, and the settings for an arpeggiator. For example, the sequence in Figure 12 can be encoded as 4 notes and the arpeggiator plugin with up as the play order, 1/16 for the speed, for 1 bar, and in 1 octave. Figure 13 shows the editor set up in IGME to replicate this.

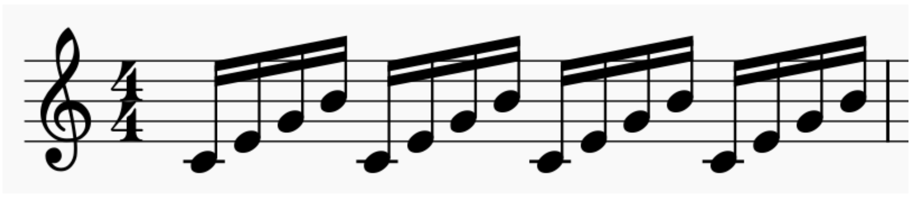
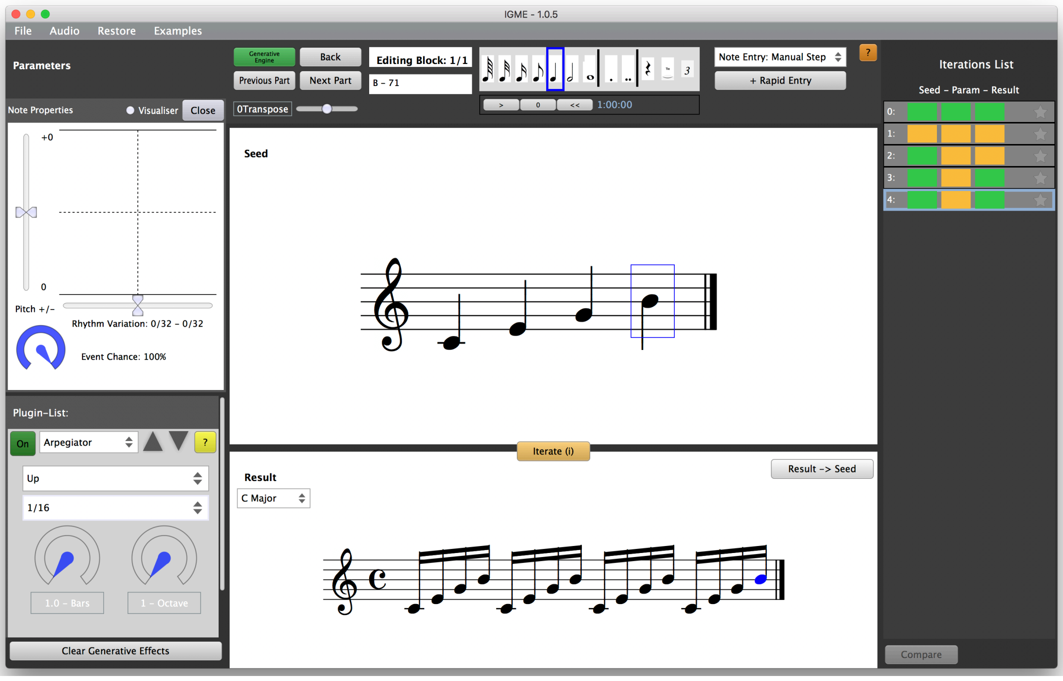


Figure 12: Simple arpeggiated idea.



Editing Block: 1/1
B – 71
Note Entry: Manual Step
1:00:00
Pitch +/-
Rhythm Variation: 0/32 – 0/32
Event Chance: 100%
Arpeggiator
Up
1/16
1.0 – Bars
1 – Octave
C Major

Figure 13: Part with an arpeggiator plugin applied.

5. DATA PIPELINE AND AUTOMATED ANALYSIS

An individual song in IGME is analysed by the following automated procedure. A song selected for analysis is first imported in MIDI format and decoded into IGME's internal representation (note that for the purpose of this study songs are limited to those in a 4/4-time signature, discussed further in section 8). Each MIDI track is given a track comprising of a sequence of individual parts. A track is split into parts based on bar lines, so the smallest possible part is a single bar, notes that cross bar lines are expressed such that the part takes up multiple bars of music. Note length and note onsets are quantised so that they are rounded to the nearest 1/32 note. Other details are lost by the process such as dynamic markings. Without making these modifications the complexity of doing this analysis would be implausible.

The general expression of two parts A and B is the relationship that relates B to A. This process therefore tries to find the set of procedures that modifies part A so that it produces the exact same musical output as part B. The processes outlined below automate this process using the techniques (discussed in section 4) to discover musical structure.

5.1. DUPLICATION ANALYSIS

The duplication analysis tried to find and group parts that have identical content. This works from left-to-right from the first track to the last. The process starts by taking the first part on track 1, and comparing it with every other part on the timeline. Note only parts that have the same number of events are compared, greatly reducing the overall complexity.

As the process works from left-to-right the overall number of comparisons decreases. When the first track is complete the process repeats starting with the first part on track 2. Only parts on track numbers greater than the current track need to be compared. When a match is found a reference is made between the two parts.

5.2. TRANSPOSITION ANALYSIS

The transposition analysis is similar to the duplication analysis, but the part is chromatically transposed incrementally from -12 semitones to +12 semitones before being compared with other parts. Essentially running the duplication test 24 times. This is expressed as a reference part, with a transpose plugin.

5.3. TRANSFORMS ANALYSIS

The transform analysis stage checks to see if the relationship between the two clips can be represented by a simple musical transform. This process is quicker than the first two analysis stages as parts that have already been marked as duplicate or transposed are removed from the task queue. Only parts that have the same number of events are compared, as the transformation process does not alter the quantity of different events. Given two parts, A and B, this process would iteratively compute the 4 transforms on part A, and check to see if the output matches part B, if this is true then part B is set to reference part A, and the relevant transform parameter is added to part B.

5.4. NOTE MAPPING ANALYSIS

This process looks to see if a given part can be represented by referencing another part, and substituting certain notes so that the result is the same. A number of checks are first made, these include, ensuring the two parts have the same total number of notes, the same rhythmic structure, and the part cannot be expressed by other simpler techniques. A list of notes that occur in each part are first computed, (we will call these list L1 and L2). A set of possible combinations are computed, by iteratively taking a single note from L1, and each note from L2, whereby the total number of comparisons is the size of L1 multiplied by the size of L2. A recursive function is then used to test each of the transforms on sequence A (original) and comparing it with sequence B (target). If a match is found the function exits and returns a list of the note mappings that transform part A to part B. If this process is successful then part B is set to reference part A, and the relevant note map parameter is added to part B. The note mapping process is CPU intensive and is run last.

5.5. ARPEGGIATION ANALYSIS

The arpeggiation analysis checks to see if a given part can be expressed as a smaller set of core notes and settings for an arpeggiator. The automated analysis first takes a given part and removes all the duplicate notes from the sequence, therefore leaving only a set of unique notes. The arpeggiator effect is then added, and the settings are iteratively worked through. At each iteration the output is computed and compared to the original, if it matches the original then the part is converted to an arpeggiated part.

5.6. OVERALL ANALYSIS

Pieces within IGME can either be analysed individually or in bulk. When computing an individual analysis, the parts are given a unique colour and the entire composition can be visualized. As an additional feature, once the analysis of the piece is computed. IGME can remove all but the unique ideas, therefore revealing just the raw building blocks that make up the rest of the song.

The analysis computes and represents the overall music into 2 overall categories, these are unique parts and representable parts. Within representable parts, several variants are grouped, these are duplicated parts, transposed parts, transformed parts, arpeggiated parts, and note map parts.

Duplication has a higher priority than the other techniques. In many instances a part can, be expressed as either a duplication of the same part previously, or as a transposition. For example, in a given 4 bar section there might be 2 unique parts and 2 parts that are expressed as transpositions of the first 2. When this 4-bar section repeats again, the 4 parts would be expressed as duplications in respect to the first 4 bar section.

6. EXAMPLES

6.1. SECRET OF THE FOREST

To give an overview of the automated analysis process discussed previously (before discussing analysing large corpuses), this section looks at a single piece of music in detail. Secret of the forest is a song composed by Yasunori Mitsuda (Mitsuda, 1995). The song has previously been deconstructed and analysed by Yu (2016). There are a number of sections in this piece that can be expressed and represented using the tools offered by IGME, that disseminate musical structure in the piece. Overall the piece has roughly 10% unique parts, and the remaining 90% can be expressed through the processes discussed in sections 4 and 5. Table 1 shows the overall distribution of parts found by this analysis process. No transformation parts were discovered so these are excluded from table 1. Figure 18 shows a visualisation of the overall piece. Duplicate parts are given the same colour, making patterns in the structure easier to distinguish, light blue is used to show unique parts, which are mostly present at the start of the piece.

	Number of Parts	Unique Parts	Duplicate	Transposed	Arpeggiated	Note Mapped
Counts	579	60	496	14	6	3
Percentage	100.00%	10.36%	85.66%	2.42%	1.04%	0.52%

Table 1: Analysis results for Secret of the Forest.

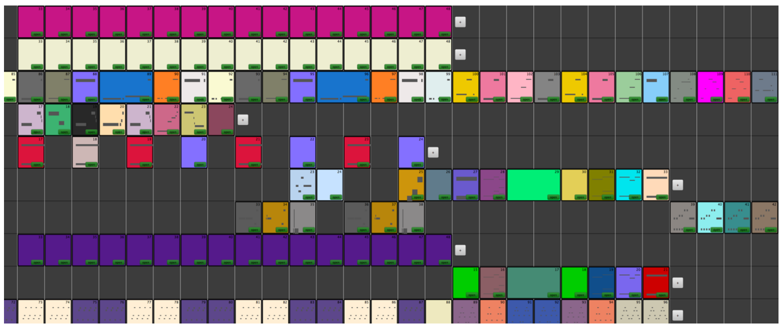


Figure 18: Visualisation of a section from the piece.

Although the analysis process performs optimally on this piece of music, there are a number of reasons why the piece cannot be analysed further. Tracks 1, 2, and 8 are percussion, and repeat a single idea throughout. Track 3 is a bass part and contains a lot of representable content, some of the content on this track is similar in structure however it is not easy to represent within IGME using current techniques, the same conclusion is true of tracks 4 also. Track 5 is almost exclusively 2 note chords, as all of the parts have the same structure this can all be represented through transforms. Tracks 6, 7, and 9, contain the bulk of the material, the unique ideas are different enough that they are mostly inexpressible by other means. 6 of the 7 parts on track 10 can all be expressed as arpeggiated parts. The final 2 tracks are mostly melodic ideas. From these 60 parts the rest of the song can be assembled.

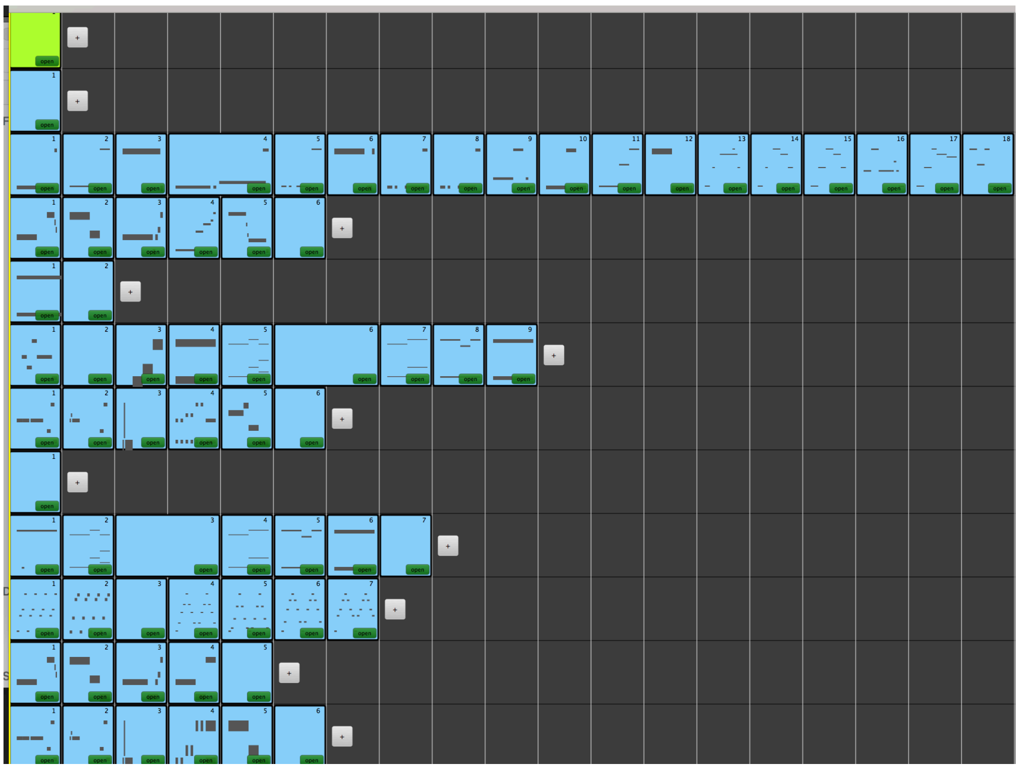


Figure 19: All of the unique ideas (building blocks) for secret of the forest.

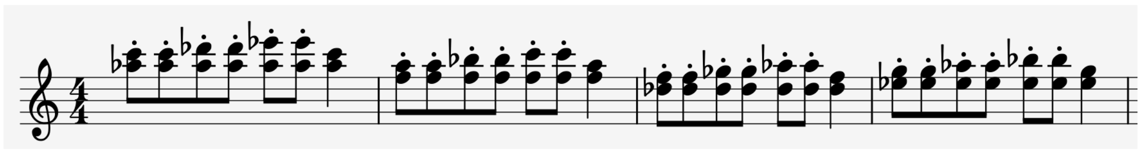


Figure 20: Musical score of repeating idea.

One of the more interesting structural ideas found when analysing the song was the repeating idea shown in figure 20, this is first used from bar 33. The same idea is repeated 4 times, but is chromatically transposed each time. The 4-bar section is then repeated 6 times throughout the piece. The repeated 4-bar sections are expressed as duplication (of the earlier section) rather than 1 part and 3 transpositions, as they did on the first occurrence.

7. DATA SETS

7.1. HUMAN COMPOSED MUSIC

To understand how the processes discussed in sections 4 and 5 can be applied to existing music it is important to analyse a large selection of it. To compute the datasets for this research, a large selection of MIDI files in different genres were gathered from various free online MIDI databases. These were then grouped by genre and analysed in bulk using the pipeline discussed in Section 5. Section 6 looked at analysing a single piece whereas this section applies the same process but for multiple pieces of work. Table 2 shows the results.

Data set genre	*Unique Parts*	*Duplicated*	*Transposed*	*Arpeggiated*	*Transformed*	*Note Mapped*	*Representable*	N
Classical Mixed	65.82%	25.12%	7.69%	0.44%	0.03%	0.90%	34.18%	363
Classical Bach	70.80%	18.58%	9.30%	0.32%	0.03%	0.96%	29.20%	224
Blues	51.70%	45.68%	2.02%	0.16%	0.01%	0.44%	48.30%	62
Country	45.24%	51.02%	3.02%	0.02%	0.00%	0.70%	54.76%	105
Dance	30.33%	67.44%	1.79%	0.09%	0.00%	0.35%	69.67%	268
Folk	22.02%	76.70%	0.55%	0.01%	0.00%	0.72%	77.98%	142
Jazz	55.89%	38.14%	4.60%	0.05%	0.03%	1.30%	44.11%	745
Rap	23.65%	75.67%	0.56%	0.05%	0.00%	0.07%	76.35%	98
Video Game	12.45%	84.97%	2.12%	0.17%	0.00%	0.28%	87.55%	218
Classic Rock	44.39%	52.37%	2.53%	0.12%	0.01%	0.58%	55.61%	1000
Pop hits	34.67%	62.46%	2.29%	0.13%	0.03%	0.42%	65.33%	1000

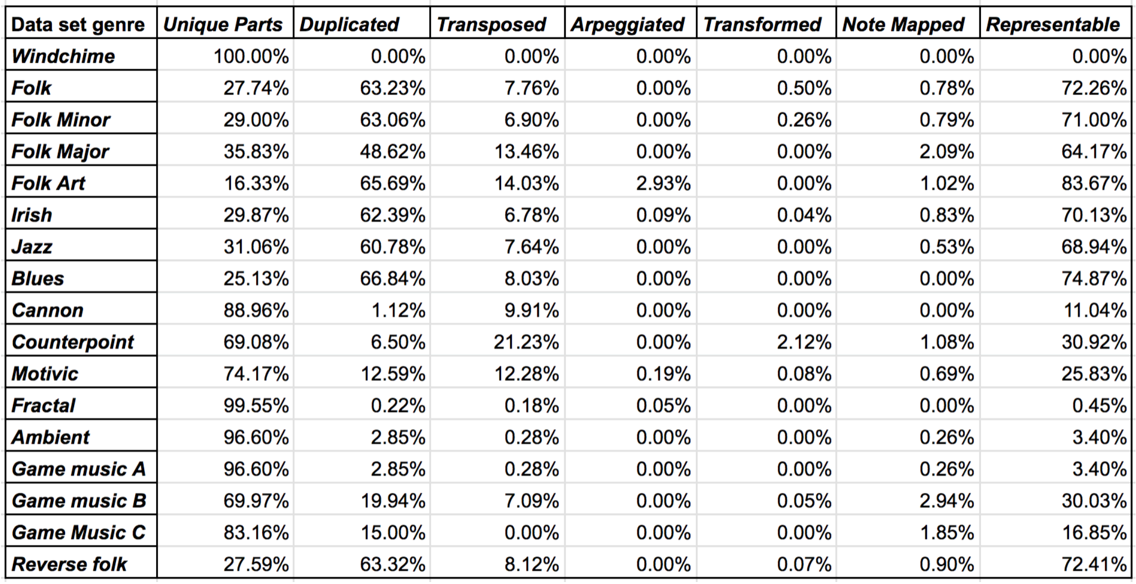
Table 2: Analysis results for various genres of music, whereby N is the number of files analysed.

The representable value is expressed as the percentage of parts that can be computed from another part (i.e. not unique). Initial observations of the data revealed that the Classical dataset scored the lowest for representability, unlike other styles of music is often instrumental, meaning that the music is perhaps more complex to accommodate for the lack of vocals. Pop, Rap, and dance music have high representable scores, perhaps as these genres of music make use of loops. Jazz has slightly more representability than classical but less than pop and rock. Video game music scores the highest overall. The results could be interpreted, that more popular forms of music (rock and pop) tend to express music in a simpler structure that conforms to the bar level hierarchy, whereas jazz and classic tend to follow more nuanced levels of structure that is not sufficiently captured by this process. Despite their low overall scores, Classical and Bach contain more transposed and arpeggiated parts then any of the other datasets.

The transformation (retrograde, inversion, rotation) technique is clearly in its current configuration either; unable to represent the music (incorrect model), or is just simply not used that often as a technique. Given the relatively high duplication score for almost all datasets, it is worth considering if current musical composition software makes this duplication either; easier to do, or make its representation obvious. Additionally, we suggest that these duplicated ideas are intentional and crucial for developing structured (non-stochastic) music.

7.2. GENERATIVE MUSIC

In addition, the output of a series of generative music programs was captured and used to create a large dataset of generative music. Based on work by (Francis, 2018) a selection of the example programs provided were compiled and run 1000 times each to produce 1000 pieces of music for each technique (see table 3). Even though the focus of this study was to test whether or not existing music could be represented by the techniques discussed in this paper, it is worth considering if and how generative music (composed by other programs) fits with this model.



Data set genre	*Unique Parts*	*Duplicated*	*Transposed*	*Arpeggiated*	*Transformed*	*Note Mapped*	*Representable*
Windchime	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Folk	27.74%	63.23%	7.76%	0.00%	0.50%	0.78%	72.26%
Folk Minor	29.00%	63.06%	6.90%	0.00%	0.26%	0.79%	71.00%
Folk Major	35.83%	48.62%	13.46%	0.00%	0.00%	2.09%	64.17%
Folk Art	16.33%	65.69%	14.03%	2.93%	0.00%	1.02%	83.67%
Irish	29.87%	62.39%	6.78%	0.09%	0.04%	0.83%	70.13%
Jazz	31.06%	60.78%	7.64%	0.00%	0.00%	0.53%	68.94%
Blues	25.13%	66.84%	8.03%	0.00%	0.00%	0.00%	74.87%
Cannon	88.96%	1.12%	9.91%	0.00%	0.00%	0.00%	11.04%
Counterpoint	69.08%	6.50%	21.23%	0.00%	2.12%	1.08%	30.92%
Motivic	74.17%	12.59%	12.28%	0.19%	0.08%	0.69%	25.83%
Fractal	99.55%	0.22%	0.18%	0.05%	0.00%	0.00%	0.45%
Ambient	96.60%	2.85%	0.28%	0.00%	0.00%	0.26%	3.40%
Game music A	96.60%	2.85%	0.28%	0.00%	0.00%	0.26%	3.40%
Game music B	69.97%	19.94%	7.09%	0.00%	0.05%	2.94%	30.03%
Game Music C	83.16%	15.00%	0.00%	0.00%	0.00%	1.85%	16.85%
Reverse folk	27.59%	63.32%	8.12%	0.00%	0.07%	0.90%	72.41%

Table 3: Output for different generative program types analysed.

The representability scores for the generative dataset vary widely. Firstly, the mostly stochastic techniques (fractal and windchime) have low representability scores, meaning these music types sound predictably chaotic in nature. On the other end of the scale Jazz and Blues have higher representability scores than their real-world counterparts. The counterpoint data set has a number of differences with the Bach data set, notably the generative set has less duplication, but much more transposition.

The findings in the section perhaps confirm why generative music is seen as either structureless (too stochastic) or repetitive. Therefore, a balance needs to be struck between repeating and developing existing ideas, and creating new ones. From the findings above, it would seem that this remains a challenge. It could also be suggested that generative music techniques are mostly suited to generating lower levels of musical hierarchy and that the human composers should focus on developing and arranging these lower levels to form higher level structure.

8. CONCLUSION

8.1. LIMITATIONS

The analysis processes used in the research considers musical structure to be grouped into bars, while this works for certain styles of music, much of the musical structure operates at smaller micro (rather than macro) levels. This is true of Classical music in which sequences may by asynchronous with bar lines. However as discussed previously this coarse resolution approach is designed to reduce the complexity of the research objectives. However, this automated approach paves the way for more formal methods of analysis such as Schenkerian analysis. Nash's (2014) *Manhattan* software (which has many parallels with this work) provides the ability to encode music at the micro and macro level thus providing a more complete representation of musical structure, although such representation must be encoded manually.

The system itself is still in a beta development stage and some limitations do present themselves. A major limitation is the inability to work with time signatures other than 4/4, and of course pieces that modulate to and from a different time signature. Such pieces are omitted from the data sets discussed in section 8. Many pieces of music cannot be represented sufficiently using IGME and there are two principle reasons for this. Firstly, the pattern-recognition capabilities of IGME are themselves limited, and are demonstrated here as a proof of concept. Developing these techniques for future work will undoubtedly increase the representable score of music, further highlighting the importance of patterns in music. Secondly, and for perhaps good reason certain music cannot be simply compounded into primitive rules.

8.2. FUTURE WORK

This research has several novel uses. Firstly, it allows a user interacting with the software to analyse music in a visually stimulating way. We can also use generalised metrics about music to assess why for example generative music is often seen as structureless, by analysing and comparing it with a style it is trying to replicate. This also might be used as a tool for learning a piece of music. Whereby a student can extract the individual pieces of music and practice these over, later slotting them the logical timeline to realise the full piece of music.

Ultimately the focus of this research was to assess whether existing pieces of music can be represented by a series of unique musical bars, and subsequent representations. As this paper has demonstrated this is indeed the case, with stronger emphasis for dance, folk and rap genres of music. Therefore, it is entirely possible to compose new pieces of music that intentionally use these types of processes. The number of techniques explored in this paper fail to capture all aspects of musical structure, however future work may look to address some of the shortcomings of this research.

9. REFERENCES

Collins, N. (2018) ‘there is no reason why it should ever stop’: large-scale algorithmic composition, *Journal of creative music systems*, Vol. 3, No. 1.

Cope, D. (1991) Computers and Musical Style, *Oxford University Press*, Oxford.

Cope, D. (1996) Computers and Musical Style, *AR editions Madison WI*, Wisconsin.

Cynthia, R. (2019), Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, *Nature Machine Intelligence,* Vol. 1, No. 5, pp. 206-215.

Francis, J, R. (2018) *Algorithmic Computer Music,* 6th Ed, unpublished.

Fux, J.J. and Wollenberg, S (1992) 'Gradus ad Parnassum'(1725): Concluding Chapters, *Music Analysis*. Vol. 11, No. 2/3, pp. 209-243.

Hamanaka, M., and Hirata, K., and Tojo, Satoshi. (2006) Implementing A Generative Theory of Tonal Music*, Journal of New Music Research,* Vol. 35, No. 4, pp. 249-277.

Herrenabsm, D. and Sorensen, K. (2012) Composing first species counterpoint with a variable neighbourhood search algorithm: *Journal of Mathematics and the Arts.* Vol. 6, No. 4, pp. 169-189.

Hunt, S., Mitchell, T., & Nash, C. (2018, May). A cognitive dimensions approach for the design of an interactive generative score editor. *Paper presented at Fourth International Conference on Technologies for Music Notation and Representation,* Montréal, Canada.

Hunt, S., Mitchell, T., & Nash, C. (2017, September). Thoughts on interactive generative music composition. *Paper presented at 2nd Conference on Computer Simulation of Musical Creativity*, The Open University, Milton Keynes, UK.

Huang, A., and Wu, R (2016) Deep learning for music, Cornell University (website), available online from https://arxiv.org/abs/1606.04930 [accessed August 2019]

Langkjaer-Bain, R . (2018) Five ways data is transforming music, *Significance,* Vol. 15, No. 1, pp. 20-23.

Lerdahl, F. and Jackendoff, R. (1985) A generative theory of tonal music, Massachusetts, MIT Press.

Marsden, A. (2010) Schenkerian analysis by computer: A proof of concept, *Journal of New Music Research,* Vol. 39, No. 3, pp. 269-289.

Moore, A,F. (2001) Categorical conventions in music discourse: Style and genre, *Music and Letters*, Vol. 82, No. 3, pp. 432-442.

Moylan, W. (1987). A systematic method for the aural analysis of sound in audio reproduction/reinforcement, communications and musical contexts, *Proceedings of the 83rd Convention of the Audio Engineering Society*, New York.

Nash, C. (2014, June) Manhattan: End-User Programming for Music. *Paper presented at New Interfaces for Musical Expression (NIME),* London, UK.

Rothfarb, L, A. (2010) *Basic Formal Units (Motive, Phrase, Period), University of California, Santa Barbara (website)* available online from http://rothfarb.faculty.music.ucsb.edu/courses/160A/formal-units.html [accessed August 2019].

Temperley, D. (2007) Music and probability, Massachusetts, MIT Press.

Yu, J.M. (2016). Deconstructing: "Secret of the Forest" from Chrono Trigger, *jasonyu* (website), available online from http://jasonyu.me/secret-of-the-forest/ [Accessed November 2019].

Zulić, H. (2019) How AI can Change/Improve/Influence Music Composition, Performance and Education: Three Case Studies, *INSAM Journal of Contemporary Music, Art and Technology, Vol. 1, No. 2, pp. 100-114.*

10. DISCOGRAPHY

Guns N' Roses. (1987), [CD], Appetite for destruction, *Universal Music.*

Mitsuda, Yasunori. (1995), [video game music], *secret of the forest*, *Square.*