DJSTER REVISITED – A PROBABILISTIC MUSIC GENERATOR IN THE AGE OF MACHINE LEARNING

Georg Hajdu
ligeti center
Hamburg University of Music and Drama
georg.hajdu@hfmt-hamburg.de

ABSTRACT

DJster is a probabilistic generator for musical textures based on Clarence Barlow's legacy program Autobusk, further developed by Hajdu since 2008. The 2023 revision for Max and Ableton Live includes new features that improve the versatility of the application and enable data exchange between the synchronous and asynchronous incarnations. The synchronous incarnation of DJster can be used to preview a texture to be further developed as a sketch in the asynchronous one. DJster allows the real-time addition and modification of tonal and metric profiles departing from Barlow's original fixed-input paradigm. This motivated an exploration of metric interpolations by means of self-organizing maps and an extension of Jean-Claude Risset's illusion of an ever-accelerating rhythm. Furthermore, the implementation of a novel *melodic cohesion* parameter allows transitions from a sequence of events to a probabilistic process, the latter being the original modus operandi. Finally, DJster as a style-agnostic music generator can be embedded in machine-learning contexts to make user interaction a more rich and intuitive experience.

1. INTRODUCTION

DJster is a generative music system based on work by the composer and computer musician Clarence Barlow done between 1986 and 2000 [1] whose development was taken over by Hajdu in 2008; its genesis being outlined in a 2016 TENOR conference paper [2]. It is a sophisticated, style-agnostic system for the generation of tonal/atonal and metric/ametric music based on a stochastic principles. In comparison to "opaque" machine learning systems, DJster represents a "transparent" approach controlled by 12 distinct, yet interdependent parameters such as *event density*, *harmoniclarity* or *tonic pitch*. By adopting a mathematical method, DJster also allows the exploration of a large number of microtonal scales without requiring any pretraining of the system. DJster exists in several incarnations for Max (Figure 1) and Ableton Live. The features added recently are concerned with the ability to interpolate between complex meters and pitch sets in real time as well as passing parameter settings to a MaxScore plug-in for the generation of music notation.

In the original version of Autobusk, Barlow made metric and tonal-semblance profiles strictly dependent on the meter and scale files to be used [7]. Their hierarchies were calculated according to the algorithms he developed for this purpose. In contrast, DJster permits arbitrary, user-defined profiles which need not be strictly hierarchical. In combination with the *melodic cohesion* parameter, DJster can thus create transitions between motives and the probabilities assigned to the members of these motives.

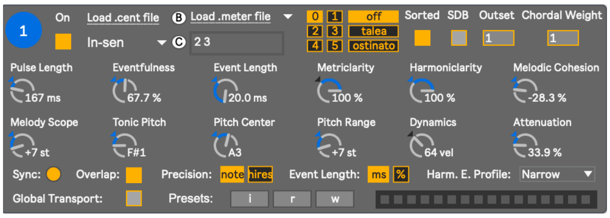

Figure 1. GUI of the 2024 version of DJster.

DJster has been used in countless projects ranging from dance-theater pieces [2] and real-time notation of EEG signals to sound interventions in hospital waiting areas [4]. Exposing all its parameters to the host environment, it can be used as event generator in Ableton Live as well as interactively in a man-machine dialog.

As outlined in [2], parameter settings could also be considered a type of notation for *musical texture*, particularly if they could be represented symbolically via shorthand notation. It requires some intimacy with the system to predict the resulting textures, but experience has shown that the connection between settings and outcome can be learned quickly and applied reliably during the composition and interaction process. Using machine learning (ML), this connection can be inverted for the generation of parameter settings via the creation of a large corpus of music generated by DJster. Upon training the ML system, music fed into it would generate settings resulting in music closely related to the input. This could spawn a recursive and potentially interesting dialog between musical input and output.

2. METER

2023 marks the year of György Ligeti's 100th birthday as well as Clarence Barlow's passing[1]. Both composers have shown great interest in meter and polymeter in their compositions and music-theoretical work. Ligeti prided himself of generalizing the medieval concept of the hemiola apparent in many a piano etude [6] while Barlow undertook quantitative research on metric hierarchy in context of his piano piece *Çoğluotobüsişletmesi* [7]. In his 1980 essay *Bus Journey to Parametron,* he pointed out the lack of available literature on this topic at the time [8]. Barlow's theory of meter is characterized by the mathematical concept of *indispensability* from which a *metric profile* can be derived for any given meter (e.g. [2, 0, 1] for triple meter). Considering prime *stratification divisors,* his formula yields profiles for complex multiplicative and additive meters, e.g. [8, 0, 3, 6, 1, 4, 7, 2, 5] for compound triple meter. While in the original version of Autobusk, the metric profiles had to be calculated and saved to an *.IDP* file, DJster generates metric profiles on the fly and accepts arbitrary *named* profiles that may be modified in real-time. This can be achieved by sending it a message consisting of the message name *append-meter*, a *meter_name* and a *list of values* for the definition of a metric profile referred to as *meter_name*. This feature facilitated the experiments on metric interpolation by means of ML Hajdu undertook during the summer of 2023, aiming to provide a framework for flexible and fluid rhythm generation.

2.1 Self-Organizing Networks

We explored the ability of *self-organizing maps* (SOM) to distinguish between the hemiolas evident in 3/4, 6/8 and 12/16 meters[2], where the metric *stratifications* consist of the permutation of the elements 2 and 3 (and powers thereof). With a Max patch created in 1992 in David Wessel's Introduction to Computer Music class at UC Berkeley, one of the participating students showed that a *multilayer perceptron* (MLP) could classify a random stream of pulses according to eight profiles which the MLP was trained to (**Table 1**). Furthermore, an *inverse* network generated an accompaniment that best fit the random series of pulses.

3x2x2:	1 0 0 0 0 0 0 0	1 0 0.56 0.1 0.82 0.27 0.64 0.18 0.91 0.36 0.73 0.45
3x2:	0 1 0 0 0 0 0 0	1 0 0.170 0 0.670 0 0.330 0 0.830 0 0.500 0
3:	0 0 1 0 0 0 0 0	1 0 0 0 0.330 0 0 0 0.670 0 0 0
2x3x2:	0 0 0 1 0 0 0 0	1 0 0.56 0.1 0.73 0.27 0.91 0.36 0.64 0.18 0.82 0.45
2x3:	0 0 0 0 1 0 0 0	1 0 0.170 0 0.500 0 0.830 0 0.330 0 0.670 0
2:	0 0 0 0 0 1 0 0	1 0 0 0 0 0 0.500 0 0 0 0 0
2x2x3:	0 0 0 0 0 0 1 0	1 0 0.36 0.73 0.18 0.56 0.91 0.1 0.45 0.82 0.27 0.64
2x2:	0 0 0 0 0 0 0 1	1 0 0 0.250 0 0 0.750 0 0 0.500 0 0

Table 1. Training data for the eight profiles used for the generation of meters with 12 pulses (inverse model). The red numbers encode indexes for the eight meters while the black numbers represent associated metric profiles, normalized to the 0–1 range.

[1] Sadly, Clarence Barlow died on June 29, 2023.

[2] We used the Max ml.som object from the ml.star package [9].

Exposing a SOM to the same profiles, the network accurately represents them in a 2-dimensional space (Figure 2) with the 2 axes indicating stratification depth (x-axis) and stratification type (y-axis). The red area denotes the division of the measure by 3 beats getting increasingly brighter as it is gradually subdivided into 12 pulses. The black area transitioning to brown (6/8) and blue (12/16) indicates the division of the measure by 2 (and then further divided into 2 and 2x3 or 3 and 3x2, totaling 12 pulses). The square is roughly divided into eight zones showing the learned profiles as well as the interpolations between them.

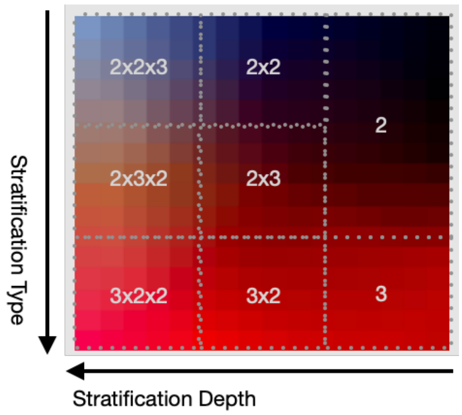

Figure 2. An ml.som object was trained to metric profiles for duple and triple meters and their subdivisions.

2.2 'Bohlen-Pierce' Meter

We then investigated how a SOM would fare if exposed to far more complex meters made of primes 3, 5 and 7. Trained to 16 different vectors with 105 values each – representing a meter made of these primes and their combinations – the SOM produced a result which was more difficult to interpret (see Figure 3). What in the previous example with 12 pulses looked like a logical arrangement in two dimensions, now seemed more like a scattering of profiles along the two axes, with some of the profiles missing. We concluded that the scattering would be the result of a dimensionality reduction performed by the SOM, for the space formed by the stratifications of these three primes may not be accurately mapped in two dimensions. To test this assumption, we calculated the Euclidian distance between the profiles and performed multidimensional scaling which yielded – in three dimensions – a sensible arrangement supporting our hypothesis. We were also able to discern some similarities to the results of the SOM (Figure 4).[3]

[3] For instance, 3 and 7 are at a closer distance to each other than 5, and the six three-strata meters are clustered in three areas (which may explain why the distinction between the pairs of meters within the clusters is blurred or why one member of a pair may have eclipsed the other).

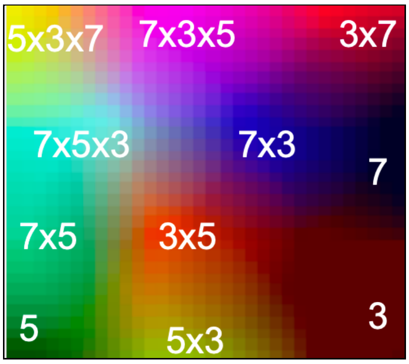

Figure 3. A SOM trained to 16 metric profiles based on primes 3, 5 and 7. In comparison to Figure 2 the zones appear scattered. This is most likely due to dimensionality reduction.

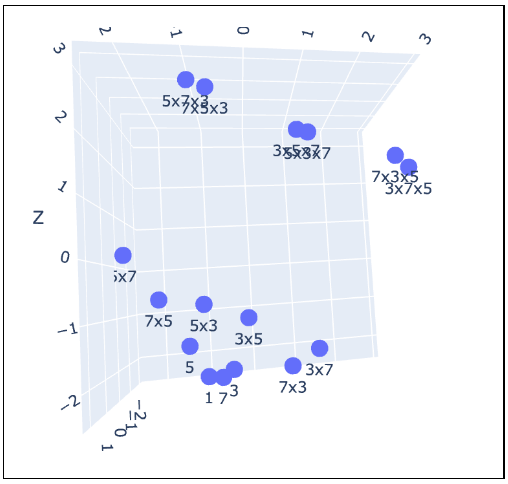

Figure 4. Multidimensional scaling of the Euclidian distances between the metric profiles in 3D, substantiating the assumption of a dimensionality reduction in the SOM.

2.3 Multilayer Perceptrons

The question arose as to how to best control the space formed by these primes with available gestural or fader controllers to provide interpolated profiles to DJster in real-time. One possibility was to use the SOM despite its seemingly "imperfect" arrangement of the profiles; it still yielded intriguing rhythmic patterns. The other was to train a multilayer perceptron to an arbitrary spatial arrangement of the profiles. We conceived of each of the six three-strata profiles as "fader" with a gradual transition from a one-pulse meter (0^{th} stratum) to meters with one, two and finally three strata, e.g. 1→3→3x5→3x5x7, corresponding to fader values 0., 0.33, 0.66 and 1 (Figure 5).

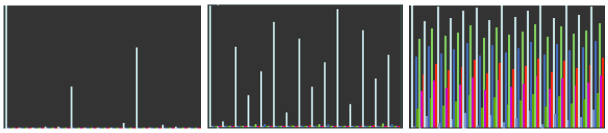

Figure 5. Metric profiles for fader values [.33, 0, 0, 0, 0, 0], [.66, 0, 0, 0, 0, 0] and [1., 0, 0, 0, 0, 0].

For this, we configured the Max-Java nnLists object to six input neurons receiving their input from the faders and 105 output neurons encoding the metric profiles. Interpolations were performed as crossfades between pairs of meters, such as 3x5x7→7x5x3 (Figure 8).

Figure 6. Notation of the interpolation between a 7x5x3 meter and a 3x5x7 meter according to interpolated metric profiles, performed by an MLP trained to 16 different metric profiles. The notehead sizes indicate the velocities of the notes which in turn are dependent on their indispensability via the *attenuation* parameter. While on the top system every third note is stressed, it is every seventh on the bottom system (also revealing the 2+2+3 nature of this additive meter). The systems in between are characterized by metric ambiguity resulting from the interpolation.

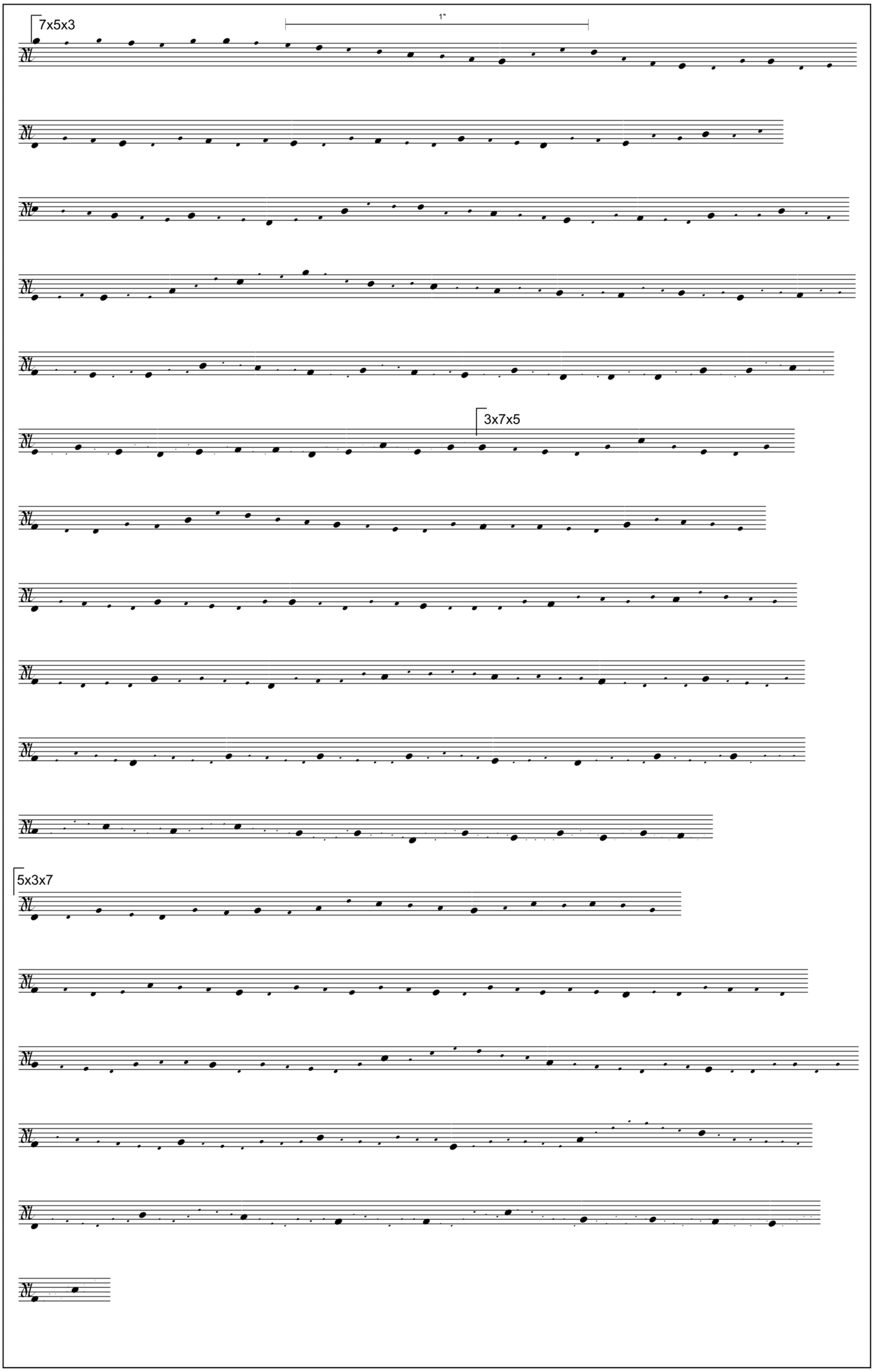

Figure 7. Notation of the Triple Moebius-Band illusion created by Hajdu. The size of a notehead depends on its velocity, which in turn is contingent on its indispensability. Values below a certain threshold will successively fade out while the sequence is sped up. Once the fade-out is complete, a new cycle begins with a rotation of the meter's lowest stratum to the top. The music is in the Bohlen-Pierce scale and notated in its N clef.

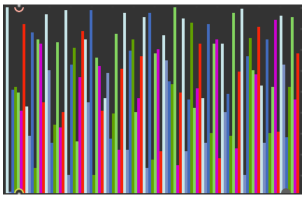

Figure 8. Interpolated profile for meters 3x5x7 and 7x5x3 (fader values [.5, 0, 0, 0, 0, .5]).

Feeding the interpolated profiles into DJster yielded fascinatingly irregular patterns that showed similarity to the rhythms employed by Ligeti in his piano etudes (Figure 6).

2.4 Triple Moebius Band Illusion

Barlow's theory of meter was also validated by an extension of Jean-Claude Risset's illusion of an ever-accelerating rhythm which Hajdu demonstrated as an example for *self-similarity* in music at the 2023 symposium in Hamburg entitled *Ligeti's Labyrinths of Wonderland.*[4] Risset's original illusion consist of fading out (or in) pulses while gradually doubling (or halving) the tempo. In Barlow's terms, it is based on a gradual removal (or addition) of a metric stratum with the stratification divisor of 2. Hajdu modified the illusion by employing meters consisting of three strata with the stratification divisors of 3, 5 and 7. The strata are rotated by gradually fading out the lowest stratum and accelerating the stream of pulses by a factor corresponding to its stratification divisor, and eventually adding the faded-out stratum back to the top.

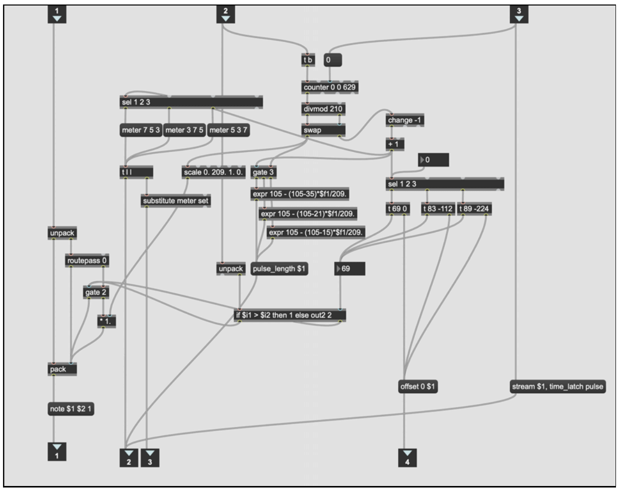

Figure 9. The Max patch driving the Triple Moebius Band illusion. Messages coming out of outlet 1 are sent to a playback device while the messages coming out of outlet 2 control DJster. The second inlet receives indispensability values from DJster.

This can be likened to moving along a Moebius band with three surfaces. The fadeout is correlated to the indispensability values of the metric profiles: The velocities for pulses above a certain indispensability value are kept while the ones equal or below the threshold are being faded out (see Figure 7 for the notation of the illusion which thus also becomes apparent on a visual level). Figure 9 shows how Max communicates with DJster to enable the illusion.

[4] https://ligeti2023.hfmt-hamburg.de/index.php/toolbar-2-2-2-2-2-2-2-3-2-2-3-2-3-2-2-3-3-2-2-2-2-2/

3. PITCH AND MELODY

In *Çoğluotobüsişletmesi* and many of his later pieces, Barlow employed the hierarchical nature of meter and scales to generate music defined by the resulting texture of melodies. To this aim, he introduced the concepts of the *indigestibility* of a number and, closely related, the *harmonicity* of an interval formed by the ratio of two numbers. As the latter term is widely used in psychoacoustics to denote the deviations of partials from an ideal harmonic spectrum, we are using the term *tonal semblance* throughout this paper instead.

3.1 Tonal-Semblance Profiles

We are defining *tonal-semblance* as the inverse of *tonal-distance*: the higher the value, the more similar a tone is to the tonic of a scale[5]. DJster contains preloaded tonal-semblance profiles for 36 scales stored in a Max dictionary. New scales can be added in three ways:

1. As a message to DJster consisting of the message name *add-scale*, a *scale name* and *list of cents values*, the first and the last items forming the replication interval when calculating the tonal-semblance profile (e.g. *add-scale tritones 0 600*). Optionally, a *reference* to four harmonic-energy profiles (narrow, wide, odd-narrow and odd-wide[6]) can be appended to the message. They are used for determining a ratio for a given cent value[7].
2. As an entry in a .cents file containing a *scale name*, a *list of cent values* and, optionally, a *reference* for the harmonic-energy profiles.
3. By dragging a *scala* file from the MaxScore Scala browser (accessible by clicking on the small, encircled B next to "Load .cent file") onto the DJster Scales menu (see Figure 1).

For instance, the cent values for the pentatonic scale with the intervals 0, 200, 400, 700, 900 and 1200 cents are expanded into the following profile ranging from -9600 to 9600 cents, consisting of a *key* and array with *cent value*, *ratio numerator*, *ratio denominator* and *tonal-semblance* value for every pitch:

[5] This similarity has often been explained by how the partials line up between the tones.

[6] The harmonic-energy profiles differ to the extent by which tonally distant intervals are considered or whether even numbers are taken into consideration in the ratios they form, as it is the case with the Bohlen-Pierce and related scales.

[7] If no harmonic-energy profile is given, the profile indicated in the "Harm. E. Profile" menu is used by default.

"pentatonic" :
```
{
"1" : [ -9600, 1, 256, -126 ],
"2" : [ -9400, 9, 2048, -62 ],
…
"36" : [ -1200, 1, 2, -1001 ],
"37" : [ -1000, 9, 16, 108 ],
"38" : [ -800, 5, 8, 107 ],
"39" : [ -500, 3, 4, 215 ],
"40" : [ -300, 27, 32, 77 ],
"41" : [ 0, 1, 1, 1112 ],
"42" : [ 200, 9, 8, 121 ],
"43" : [ 400, 5, 4, 120 ],
"44" : [ 700, 3, 2, 273 ],
"45" : [ 900, 27, 16, 84 ],
"46" : [ 1200, 2, 1, 1001 ],
…
"78" : [ 8800, 162, 1, 86 ],
"79" : [ 9100, 192, 1, 116 ],
}
```

Note that this large range is necessary to account for some extreme parameter settings where tonic pitch is in the low range while the pitch window – formed by pitch center and pitch range – is in the high range and vice versa.

Disregarding the large absolute differences between values, the tonal-semblance values (which can be either negative or positive due to the *polarity*[8] of the ratio) are processed to obtain pitch and index pairs and sent to a Max table object with the index determining the likelihood for the associated pitch to occur[9] (Figure 10). The Max *table* object does not actually receive a pitch value, but rather (another) index for an index-to-pitch lookup occurring in the *funbuff* object.

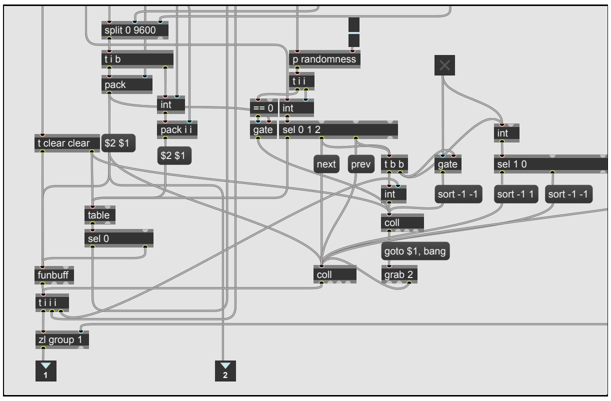

Figure 10. A section from the DJster subpatch implementing the *melodic cohesion* parameter.

3.2 Melodic Cohesion

Autobusk was conceived as a near stateless system, i.e. while doesn't keep a history of past events (e.g. to filter out repetitions), the outcome of each drawing is influenced by the previous one. By introducing the parameter of *melodic cohesion*, we create a continuum between sequential readout and the stochastic drawing of pitches from the tonal semblance profiles, where a value of -100% denotes a descending pitch set, a value of 0% the default probabilistic behavior and +100 an ascending pitch set[10]. This determines, for every event, the likelihood of whether to follow the sequence or draw a pitch stochastically. The pitch sets can be ordered according to the gestalt principles of *proximity* and *similarity*, i.e. to either pitch height or tonal semblance. In the patcher shown in Figure 10, the *randomness* subpatcher sends for every event a number between 0 and 2, their probability controlled by the melodic coherence parameter. These numbers represent the modes of the system (0 = random; 1 = next entry; 2 = previous entry). Note that for each event the complete processed tonal-semblance profile is copied to both *coll* objects and sorted according to pitch height, if the gate, controlled by the *sorted* parameter, is on[11]. If mode is either 1 or 2, the *coll* on the right serves to identify, within the sequence, the index for the pitch of the current event (fed by the red patch chord and stored in the *int* object) in the *coll* underneath, which in turn sends a new pitch to be played. This is how DJster ascertains that a profile be played sequentially despite intermittent random leaps.

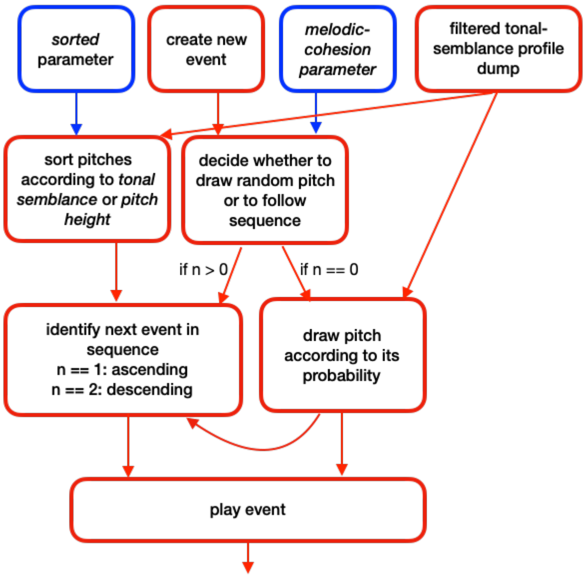

Figure 11. Diagram of the data flow in Figure 10.

3.3 Replacing a Tonal-Semblance Profile

The content of a tonal-semblance profile can be replaced or, when the scale doesn't exist, created by sending a message consisting of the message name *replace-pitchset* and a *scale_name,* as well as a *dictionary* con-

[8] Barlow uses the term to indicate whether the top or the bottom note of an interval is to be considered more stable. His approach convincingly derives the "anti-polarity" of the just fourth or the minor third mathematically but is somewhat arbitrary when applied to tonally distant intervals.

[9] The number of intervals to be considered are affected by the parameters pitch center, pitch range, melody scope, harmoniclarity, and metriclarity values as well as the indispensability for the current pulse.

[10] The transition from a strict sequence to a Markov process of 0th order can be likened to the transition of a solid to a gaseous state where connections between molecules are increasingly broken in favor of free and stochastic movement. This phenomenon has been musically explored by composer Iannis Xenakis from whom the term *stochastic music* originates.

[11] As this is computationally rather expensive, we are hoping to harness the power of the array object recently introduced in Max 8.6.

taining any number of key and value pairs; value being an array consisting of four items *cent value*, *ratio numerator*, *ratio denominator and tonal-semblance value*, such as the following example:

```
{
        "1" : [ 0, "-", "-", 7 ],
        "2" : [ 100, "-", "-", 6],
        "3" : [ 700, "-", "-", 5],
        "4" : [ 600, "-", "-", 4],
        "5" : [ 1100, "-", "-", 3],
        "6" : [ 1200, "-", "-", 2],
        "7" : [ 600, "-", "-", 1]
}
```

Since ratio numerator and ratio denominator are just used for reference and not for calculation, they can be replaced by any placeholder such as "-" or -1. However, the tonal-semblance values ought to be treated as *user-assigned* priority indexes determining the order of the pitches associated with them. With the *Sorted* switch turned off (see Figure 1) and melodic cohesion set to its extremes, motives can be formed which may contain repeated notes.

Note that the tonic pitch and the range parameters (determining the size of the pitch window) should be set to include the motive, whose cent values are relative to tonic pitch (e.g. with C3 as tonic pitch and 6 for pitch range, F#3 would form the center of the window ranging from 0 to 1200 cents). With *Sorted* turned on, the motive would be treated as a scale consisting of 0 100 600 700 1100 1200 cents.

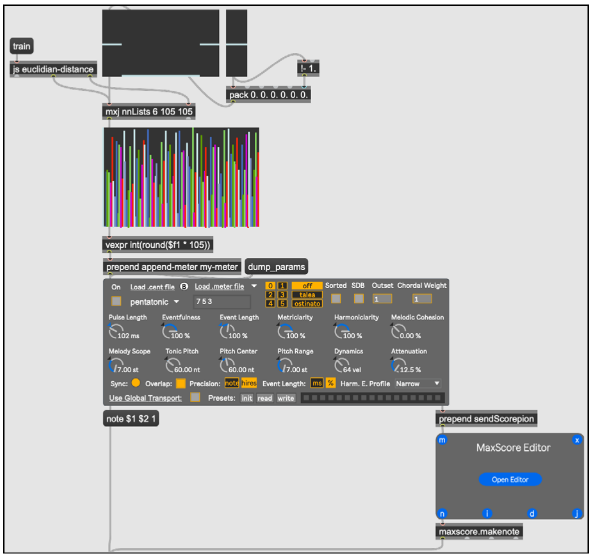

Figure 12. Metric interpolation performed by the nnLists object, a simple MLP object with one hidden layer. Parameter settings—including the stored tonal-semblance and metric profiles—are sent as dictionaries to the DJster Scorepion, a MaxScore Editor plugin via the *dump_params* message.

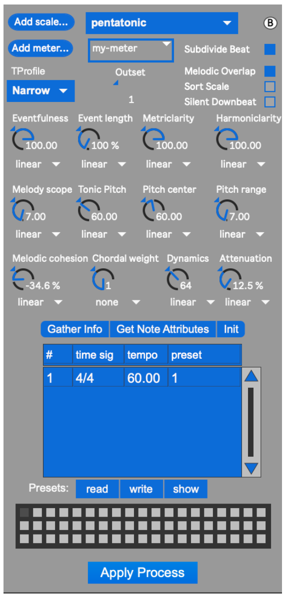

Figure 13. GUI of the DJster Scorepion. Note that the parameter settings have been set according to the sendScorepion message received by the MaxScore Editor.

4. SKETCHING

While DJster was devised as a real-time (synchronous) event generator for Max (and Ableton Live through its Max for Live API), there is also a MaxScore plugin *(Scorepion)* for the asynchronous generation of events. These incarnations can now easily be synced via parameter dumps. When used to compose music, the real-time version of DJster (referred to as DJster RT) can be used for synchronously "auditioning" a stream of events with its parameters set to the desired output. A *dump_params* message causes a dictionary containing the current parameter settings as well as the metric and tonal-semblance profiles to be sent out to a connected MaxScore editor with the DJster *Scorepion* loaded. When prepended by a *sendScorepion* message, the plugin will receive the dictionary and set its buffers and parameters accordingly (Figure 12). In the Scorepion, the user can generate music notation representing the input from DJster RT. In [9] we have presented an earlier version of the Scorepion which is now compatible with the parameter names and ranges of the current version of DJster RT (Figure 13). Since the Scorepion uses the tempo/time signature paradigm of MaxScore, it is recommended that DJster RT make use of the Max *Global Transport* which also uses tempo and time signature rather than absolute time. The pulse length and pulse number are automatically set by the current state of the transport, and the strata of the meter are treated as *subdivisions* of the beat. However, if Global Transport is not being used, *Subdivide Beat* will automatically be switched off in the Scorepion (see Figure 13). This is how the music in Figure 6 was generated.

After performing a score dump, the resulting score can, for instance, be stored in a maxscore.icanvas abstraction, another MaxScore Editor or a Max dict object.

The music can then be used in an arrangement or collage of the material generated: Instead of slavishly following the output of the music generator, the composition thus turns into an interactive and iterative process of auditioning and editing, emphasizing the agency of the composer who thus remains in the "driver's seat."

5. ABLETON LIVE IMPLEMENTATION

The 2023 version of DJster was also turned into an Ableton Live device with Push support. It consists of a wrapper around the Max abstraction with two significant adaptations which we shall describe below.

5.1 Subdivisions

Since Live follows the paradigm of measures, beats and tempo rather than absolute time, the *event length* GUI element is disabled by default, and the value is calculated from the current Live settings. Therefore, a meter chosen in the meter menu is treated as a *subdivision* of the beat rather than defining the entire meter. Subdivisions can be changed within measures: DJster will wait to the next beat and calculate the corresponding pulse number by taking the eclipsed time since the beginning of the measure into consideration (Figure 14). This allows several instances of DJster to be in sync with each other despite switching subdivisions in the interim.

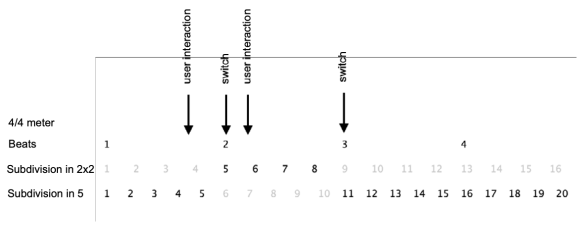

Figure 14. The subdivision of a beat can be changed repeatedly within a measure affecting the pulse number, fundamental for the calculation of indispensability values.

6. OUTLOOK

6.1 Application in Medical Environments: Healing Soundscapes

In 2014, a team from Hamburg consisting of music therapists, music psychologists, medical doctors and composers founded a project called *Healing Soundscapes* with the aim to install and explore the effects of musical soundscape interventions[12] in hospital waiting areas [10]. From the onset, DJster (in conjunction with the MaxScore Sampler) was chosen as an engine to realize the immersive soundscapes, requiring a minimum of four speakers placed in the corners of the (waiting) room. A considerable number of compositions consisting of parameter presets for five instances of DJster and select sound banks have already been created by graduate students of the HfMT.

[12] https://tinyurl.com/5c753mrc

In 2023, the Healing Soundscapes project became an integral part of the newly founded *ligeti center*, a transdisciplinary joint venture between HfMT, the University of Applied Sciences, the Technical University and the UKE [11]. The project consists of four strands with the following objectives:

1. Establishment of an audio network at the University Heart Center with a central computer delivering streams of generative music created by multiple instances of DJster running on a server.
2. Development of a hardware standalone version of DJster.
3. Development of an intelligent, portable speaker system with each speaker featuring a built-in single-board computer running with the ability to exchange data with the other speakers.

6.2 AI for Analysis and Generation

Current Deep Learning (DL) networks, and Large Language Models (LLM) in particular, are powerful tools which have added an important facet to computer music since about 2016 [12] [13]. One might wonder how a probabilistic music generator might fare in the future considering the current developments of LLMs such as Google's MusicLM [5]. A few experiments with ChatGPT 3.5 prompted to compose a melody for the Bohlen-Pierce scale have shown that it is difficult to achieve sensible results with such general-purpose AI systems and it remains to be seen how soon we will see models capable of capturing cognitive and music-theoretical principles from the music presented to them, to the extent that they can extrapolate from their internal representations of a music corpus and not just create derivative music. In contrast, DJster is not particularly suited to recreating music in specific historical or ethnic styles, but, with its built-in rules grounded in cognition and mathematics, it permits one to venture into the uncharted territories of microtonal music. Considering the strengths of both approaches, the near future might therefore favor a combination of connectionist and stochastic approaches for music generation.

7. CONCLUSION

In this paper, we have shown how a nearly 40-year-old piece of software can be put to good use by embedding it in modern authoring environments and the context of current machine-learning approaches. Its successful application in various use cases validates the approach Barlow took at the time of its inception and the effort undertaken by Hajdu going into further developing this software.

In the introduction, we have outlined the possibility of establishing a recursive feedback loop consisting of the analysis and synthesis of musical material. The idea behind this is not fueled by the desire to reproduce an input as faithfully as possible but rather by a process of *resonance* where a generator responds to a musical input according to its own affordances. This could also be seen as a metaphor for intercultural or interspecies dialog. This approach could be expanded by using text prompts describing the preferred character of the music or an emotional state, very much like the way Dall-E, ChatGPT or MusicLM create content based on a given text input.

When pondering the question of using AI for the creation of music, it dawns on us that it may not be as straightforward as it might seem. Technological progress doesn't follow an exponential curve, but rather a sigmoid curve. This implies that progress can be made fast, but perfection takes a long time to achieve. Since in art and science, its practitioners strive towards perfection, we postulate that it may take an extended amount of time to train a DL network to achieve comparable depth and perfection that we have grown to admire in the art of the great masters.

Acknowledgments

We would like to thank Lin Chen, James Tsz-Him Cheung and Alessandro Anatrini for their contributions to this paper as well as acknowledge the Federal Ministry of Education and Research in Germany (BMBF) and the Hamburg Authorities for Science, Research, Equality and Districts (BWFGB) for their support of this research.

Examples

Supporting materials are available at:
https://tinyurl.com/bdkc2ass.

8. REFERENCES

[1] C. Barlow's AUTOBUSK user manual "AUTOBUSK: A real-time pitch and rhythm generator" for the Atari ST version is available as report no. 44 (2001) in the series "Musikinformatik & Medientechnik".

[2] G. Hajdu. "Resurrecting a Dinosaur – the adaptation of Clarence Barlow's legacy software AUTOBUSK," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'16,* 2016, pp. 181–186.

[3] X. Fu, *Kommunikation und Technologie in dem Tanztheaterstück enactment::interchange*. Hofheim: Wolke, 2023.

[4] G. Hajdu, B. Carey, G. Lazarevic, and E. Weymann, "From atmosphere to intervention: the circular dynamic of installations in hospital waiting areas," in *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME), 2017*, pp. 364–369. https://doi.org/10.5281/zenodo.1176284

[5] A. Agostinelli et al., "Musiclm: Generating music from text," *arXiv* preprint arXiv:2301.11325, 2023.

[6] S. A. Taylor, "Hemiola, maximal evenness, and metric ambiguity in late Ligeti," *Contemporary Music Review* 31, no. 2–3, 2012, pp. 203–220.

[7] C. Barlow, "Two Essays on Theory," *Computer Music Journal,* vol. 11, no. 1, 1987, pp. 44–60.

[8] C. Barlow, "Bus journey to parametron," *Feedback Papers*, vol. 21–23, 1980.

[9] B. D. Smith and G. E. Garnett, "Unsupervised Play: Machine Learning Toolkit for Max," *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, 2012. https://doi.org/10.5281/zenodo.1178419

[10] E. Weymann, "Healing Soundscape," *Lexikon Musiktherapie* 3, 2020, pp. 239–243.

[11] G. Hajdu et al., "The ligeti center: Ligeti and his vision of a center for computer music in Hamburg," *Proceedings of the International Computer Music Conference*, Shenzhen, 2023, pp. 81–85.

[12] A. van den Oord et al., "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[13] A. Roberts et al., "Magenta studio: Augmenting creativity with deep learning in ableton live.", 2019. [Online]. Available: http://research.google/pubs/pub48280