

EVOLVING 8-BIT ARRANGEMENTS: A GENETIC ALGORITHM FOR THE NES FOUR-VOICE IDIOM

Giovanni Martinelli

Departamento de Ciência da Computação, Universidade Federal de Minas Gerais

ABSTRACT

An 8-bit cover is a re-composition: the Nintendo Entertainment System (NES) sound chip has only four monophonic tone channels, so chords must become arpeggios, and a bass line and a drum part must be invented. We present a genetic algorithm (GA) that arranges a lead sheet (melody, chord symbols and part labels) for the four NES channels. The genotype holds four integer genes per bar, which select the harmony texture of the second pulse channel, the bass texture of the triangle channel, the drum pattern of the noise channel and the octave of the melody. Deterministic realizers turn every genotype into a valid arrangement, so the chip rules hold by construction, and a weighted mean of seven music-theory criteria scores each genotype. Selection is roulette with windowing and elitism, crossover cuts between bars, and mutation changes each gene with probability 0.01. We arrange two traditional tunes with famous 8-bit versions, *Ievan Polkka* and *Korobeiniki* (the Tetris theme), and six Brazilian songs reduced from multi-track MIDI files, and report the evolution of the fitness, the diversity across seeds, a random-search baseline and a weight ablation.

1. INTRODUCTION

Fans re-create songs as 8-bit music, the sound of 1980s consoles. The NES audio processing unit offers two pulse channels (P1, P2), a triangle channel (TR) and a noise channel (NO), each playing one note at a time, plus a sample channel that we do not use [1]. Within these limits, chiptune composers settled on a few conventions: fast arpeggios that stand for chords, a dedicated bass voice and noise-based percussion [2]. These were aesthetic choices as much as technical necessities [3]. Arranging a song for the chip is therefore a composition task: the melody is given, but the harmony texture, the bass line, the drum part and the register must be decided bar by bar.

We use the NES four-voice idiom as the stylistic constraint of a classic algorithmic composition method, the genetic algorithm. The system receives a lead sheet, written in ABC notation or reduced from a multi-track MIDI file, and outputs a four-track MIDI file that obeys the chip

rules, rendered to audio only for listening. To our knowledge, no earlier GA arranges music for the NES chip; our contribution is to cast the idiom as a searchable space of per-bar textures, while the GA itself is standard. Code, MIDI files, audio (General MIDI and chip renderings), piano rolls and videos of the ten arrangements of eight songs are available at <https://github.com/GiowGlow/chiptune-ga-arranger> (folder `samples/`, one subfolder per piece).

2. RELATED WORK

Genetic algorithms have arranged existing music for an instrument with physical limits, as in the guitar arranger of Tuohy and Potter [4]; we also treat arranging as search, but generate new accompaniment for the four NES channels from the chord symbols instead of selecting notes from a full score. Evolutionary orchestration searches Pareto-optimal combinations of instrument samples that imitate a target sound, under instrumentation constraints [5]; we address a symbolic counterpart of this problem with a single weighted fitness. In melody harmonization, Freitas and Guimarães evolve one chord per bar with a crossover that cuts only at bar boundaries [6], and Phon-Amnuaisuk et al. report that the rule weights, more than the operator rates, decide the quality of evolved four-part harmonies [7]; we adopt the bar-aligned crossover and report a weight ablation. GenJam organizes its genotype in bars and four-bar phrases [8], and AMUSE combines normalized music-theory objectives in a weighted sum [9]; our fitness follows the same scheme, with criteria about the interplay between channels. Evolutionary work on 8-bit music has evolved bytebeat sound formulas with genetic programming and human ratings [10], and LakhNES maps the parts of arbitrary MIDI files onto the NES voices at random to pre-train a Transformer [11]; instead, we search the per-bar textures with explicit musical criteria.

3. METHOD

3.1 Representation

The lead sheet is parsed into 4/4 bars. Each bar holds its melody notes (MIDI pitch, onset and duration in beats), its chord segments (root and chord tones), a part label (A, B, ...) that the configuration maps to a role (verse or chorus) and a phrase index (phrases of four bars from the start of each part, as in GenJam [8]). The genotype (the arrangement plan) is an integer array with one row per bar and



Gene	Channel	Values (indices 0, 1, ...)
P2	pulse 2	hold : chord root, held; arp16 , arp32 : chord tones cycled up from the root in 16ths or 32nds; echo : the melody an eighth later, softer; third : the highest chord tone at least a minor third under each melody note
TR	triangle	root : held root; octave pump : root and octave above alternating in eighths; root-fifth : quarters root, fifth, root, fifth; walking : quarters root, third, fifth, semitone below the next root
NO	noise	none ; half time : kick 1, snare 3, hi-hat 2 and 4; 8th beat : kick 1 and 3, snare 2 and 4, hi-hat off-beats; 16th beat : 8th beat plus hi-hat on every sixteenth; fill : 8th beat, then four snare sixteenths; one hit per step (kick over snare over hi-hat)
OCT	pulse 1	0, +12, -12 semitones

Table 1. The four genes of each bar and their values. P1 always plays the melody; the GA chooses only its octave.

four genes per row; each gene is an index into its value list in Table 1, so a bar has $5 \times 4 \times 5 \times 3 = 300$ choices and a 24-bar tune 300^{24} genotypes. For each gene value, a deterministic *realizer* writes the notes of the bar from the chord in effect at each onset: P2 roots sit in C3–B3, TR roots in E1–D#2, drum hits fall on a sixteenth-note grid, and a note outside a channel’s range is folded by octaves into it. Every genotype is thus a valid arrangement, and a validator checks the chip rules before export: one note at a time per channel, pulse pitches in MIDI 33–108 (a one-semitone safety margin inside the chip’s 32–108 [1]), triangle pitches in 21–108, P2 and TR chord tones on beats 1 and 3 (except the echo, which replays the melody), and complete bars.

3.2 Fitness

The fitness is $f = \sum_i w_i s_i / \sum_i w_i$, the weighted mean of seven criteria $s_i \in [0, 1]$ with weights w_i , as in AMUSE [9]; a criterion that does not apply to a tune (contrast without a chorus) is left out of both sums. With $a_c(b)$ the activity of channel c in bar b (onsets divided by the 16 sixteenth steps, capped at 1, a note density as in [12]): *complementarity* is the mean of $1 - |a_{P2}(b) - (1 - a_{P1}(b))|$, so P2 is calm under a busy melody and busy under a sparse one; *register* is the share of the sixteenth steps where both P1 and P2 sound in which P2 is lower (1 if they never overlap), a voice-crossing penalty as in GA harmonization [7]; *coherence* is the share of adjacent bars inside a phrase with the same P2, TR, octave and drum genes (a fill may follow any pattern in the bar that ends the phrase), a repetition objective like the pattern matching of AMUSE [9]; *contrast* is $0.5 + \bar{d}_{\text{chorus}} - \bar{d}_{\text{verse}}$ clipped to $[0, 1]$, with d the mean activity of P2, TR and NO; *bass smoothness* is the mean score of every leap of the bass line, inside and across bars, 1 up to a fourth, 0 from an octave and linear in between (the upper notes of the octave pump are an ornament, not leaps), a graded form of the leap rules of [7,9]; *idiom* is the share of good decisions: arpeggios on single-chord bars, a chiptune convention [2], and drum fills exactly at phrase ends; *fidelity* is the share of bars where the melody stays in

Hyperparameter	Default (MIDI songs)
Population size; elite count	60; 2
Crossover	one point, between bars, rate 1
Mutation rate, per gene	0.01
Max generations; stagnation window	3000; 300 (4000; 400)
Phrase length	4 bars
Criterion weights	1, fidelity 0.5

Table 2. GA hyperparameters. Every run uses these values unless stated otherwise.

its written octave, which prevents the degenerate genotype that raises the whole melody to win the register criterion. The leap thresholds and the default weights (1, with 0.5 for fidelity) are our choices. Since every realization depends only on its bar and its gene, the system precomputes per-bar score tables and evaluates a whole population with array indexing.

3.3 Genetic Operators

Table 2 lists the hyperparameters. The initial population has 60 random genotypes, every gene uniform over its values. Parents are drawn by roulette selection with windowing (probability proportional to the fitness minus the lowest fitness of the generation), which keeps the selection pressure when the weighted mean gives close values; the two best genotypes pass unchanged (elitism). One-point crossover, applied to every pair of parents, cuts only between bars [6], and mutation replaces each gene by another valid value with probability 0.01, close to the usual rate $1/L$ [9,13] for the $L = 96$ and 128 genes of the folk tunes. The GA stops after 3000 generations or after 300 generations without improvement (4000 and 400 for the longer MIDI songs). All randomness comes from one seeded generator, and each run stores its seed, hyperparameters, input hash and code commit.

3.4 Inputs and Rendering

ABC lead sheets give the melody, the chords and the part labels directly. A multi-track MIDI song is first reduced to a lead sheet. The chosen melody track or tracks are merged by a skyline (the highest note wins) and snapped to a sixteenth-note grid. One chord per half bar is chosen among major, minor, dominant-seventh and minor-seventh chords: the other tracks vote for pitch classes in proportion to how long they sound, bass tracks count twice, and a chord whose root is the bass note gets a bonus. The arrangement is rendered twice from the same MIDI file: with General MIDI timbres (FluidSynth) and with our own NES chip synthesizer (pulse, triangle and noise waveforms).

4. RESULTS

Setup. We arranged two traditional tunes with well-known 8-bit versions: *Ievan Polkka*, with the melody as sung by Loituma, in E-flat minor at 110 BPM, with form A B A’ played twice (24 bars), and *Korobeiniki*, the Tetris theme, in A minor at 150 BPM, with form A A B A (32 bars). The

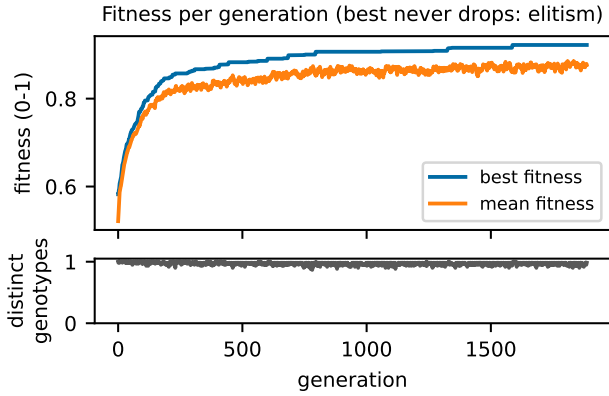


Figure 1. Korobeiniki, seed 1. Top: best and mean fitness per generation. Elitism keeps the best curve non-decreasing. Bottom: share of distinct genotypes in the population.

	Ievan Polkka	Korobeiniki	6 MIDI songs
GA, mean of 5 seeds	0.92	0.90	0.87
GA, worst seed	0.91	0.88	0.82
Random search	0.64	0.62	0.60
Genes differing	28%	32%	36%
<i>Ablation, seed 1: criterion score, default weight → 0</i>			
coherence	0.83 → 0.28	0.92 → 0.63	0.84 → 0.31
contrast	1.00 → 0.55	0.93 → 0.39	0.90 → 0.54
bass smoothness	1.00 → 0.78	0.97 → 0.90	0.93 → 0.87
idiom	1.00 → 0.86	0.94 → 0.69	0.92 → 0.66
fidelity	0.92 → 0.50	1.00 → 0.44	0.92 → 0.27
register	1.00 → 0.94	1.00 → 0.96	0.94 → 0.86
complementarity	0.64 → 0.56	0.73 → 0.71	0.72 → 0.61

Table 3. Fitness with the default weights. Random search scores as many genotypes as the first GA run; genes differing: mean share of genes that differ between two seeds. The MIDI column averages the six songs. Ablation: removing a criterion lowers its own score in every case.

B parts are labeled chorus. All runs on these two tunes stopped by stagnation, after 717 to 1986 generations.

Evolution. Figure 1 shows a typical run. The best fitness rises from 0.58 in the random generation 0 to 0.92, and the mean fitness follows it closely. The share of distinct genotypes stays above 0.86: the genotypes become similar, but mutation keeps most of them different. Between generation 0 and the final genotype, the coherence of Korobeiniki goes from 0 to 0.92 and its fidelity from 0.41 to 1.

Search against chance. Table 3 compares five seeds with a random search that scores as many random genotypes as the first GA run did (43,080 and 113,220 for the two folk tunes). Random search stays near 0.6 because it never finds coherent phrases (coherence near 0), while every GA run on the folk tunes reaches 0.88 to 0.94. The seeds end on different genotypes: on average, 28% (Ievan Polkka) and 32% (Korobeiniki) of the genes differ between two seeds. The same configuration therefore yields different arrangements, as the assignment requires.

Ablation and hyperparameter variation. Setting one weight to 0 lowers that criterion (Table 3). Without fi-

delity, the GA moves about half of the bars of the melody by an octave; without coherence, it changes textures inside phrases; without contrast, the chorus loses its denser accompaniment. Register and complementarity move least: most P2 textures sit below the melody, and the texture vocabulary limits how closely P2 can mirror the activity of the melody. The submitted pieces vary the seed (samples korobeiniki-s1 and korobeiniki-s2) and the fidelity weight (ievan-polkka-s1 and ievan-polkka-s1-fidelity0). With fidelity at 0, the GA raises 12 of the 24 melody bars by an octave, so the melody jumps between registers, and the A parts move to held chords over the octave pump.

Listening. In Ievan Polkka (seed 1), the A parts hold a chord tone under the melody over a root bass and light drums, the B parts switch to fast arpeggios over the octave pump, a typical NES bass, with a sixteenth-note beat, and a fill closes every phrase. Korobeiniki arpeggiates throughout, with a root bass and fills only in the A parts, and plays the octave pump and a sixteenth-note beat in the bridge (B). The noise channel sounds 36% to 48% of the time, close to the 39% measured in NES game music, but our P2 and TR channels almost never rest (99–100%, against 84% and 70% in that corpus [1]).

MIDI songs. We also arranged excerpts (32–56 bars) of six Brazilian songs from fan-made MIDI files: *Voa Voa*, *Mulher de Fases*, *Amor I Love You*, *Por Você*, *Aquarela* and *Fico Assim Sem Você*. In four files, the embedded lyric syllables identify the sung melody track; in the other two, the melody is the monophonic line that plays throughout. These runs use the MIDI settings of Table 2 and stopped after 1182 to 4000 generations (4 of the 30 seed runs reached the cap). We repeated the experiments on every song (last column of Table 3). The GA beats random search on each song (five-seed means 0.84–0.89 against 0.57–0.62), the seeds differ more (36% of the genes), and removing a criterion lowers its own score on every song. The songs score lower than the folk tunes on average (0.87 against 0.91). Their melodies often sit in octaves 3–4, so keeping the written register conflicts with keeping P2 below the melody (seed 1: fidelity 0.75–1.00, register 0.87–1.00), and coherence is lower (0.69–0.92, against 0.83 and 0.92 for the folk tunes at seed 1).

5. DISCUSSION AND FUTURE WORK

The search, not chance, produces the arrangements, but the approach has clear limits. The fitness is a weighted mean of hand-written heuristics, a common choice whose weights are hard to tune [14] and that hides the conflicts between objectives. A Pareto front [5, 6] would expose these trade-offs. Automatic chord detection on MIDI songs is approximate, and we did not measure its errors. The textures come from a fixed vocabulary, so the GA chooses between arranging gestures but does not invent new ones; the bass never rests, and only 4/4 is supported. Our chip synthesizer has one fixed duty cycle per pulse channel and no envelopes or pitch effects. Future work: a fitness learned from the NES Music Database [1], evolved variations of the P2 and TR lines, and an A/B listening test as in [11].

6. REFERENCES

- [1] C. Donahue, H. H. Mao, and J. McAuley, “The NES music database: A multi-instrumental dataset with expressive performance attributes,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018. [Online]. Available: <https://arxiv.org/abs/1806.04278>
- [2] K. Driscoll and J. Diaz, “Endless loop: A brief history of chiptunes,” *Transformative Works and Cultures*, vol. 2, 2009.
- [3] K. Collins, “In the loop: Creativity and constraint in 8-bit video game audio,” *Twentieth-Century Music*, vol. 4, no. 2, pp. 209–227, 2007.
- [4] D. R. Tuohy and W. D. Potter, “GA-based music arranging for guitar,” in *2006 IEEE International Conference on Evolutionary Computation (CEC)*. IEEE, 2006, pp. 1065–1070.
- [5] G. Carpentier, G. Assayag, and E. Saint-James, “Solving the musical orchestration problem using multiobjective constrained optimization with a genetic local search approach,” *Journal of Heuristics*, vol. 16, no. 5, pp. 681–714, 2010.
- [6] A. R. R. Freitas and F. G. Guimarães, “Melody harmonization in evolutionary music using multiobjective genetic algorithms,” in *Proceedings of the 8th Sound and Music Computing Conference (SMC)*, Padova, Italy, 2011.
- [7] S. Phon-Amnuaisuk, A. Tuson, and G. Wiggins, “Evolving musical harmonisation,” in *Artificial Neural Nets and Genetic Algorithms*. Springer Vienna, 1999, pp. 229–234.
- [8] J. A. Biles, “GenJam: A genetic algorithm for generating jazz solos,” in *Proceedings of the International Computer Music Conference (ICMC)*. Aarhus, Denmark: International Computer Music Association, 1994, pp. 131–137. [Online]. Available: <https://genjam.org/wp-content/uploads/2019/07/bilesicmc94.pdf>
- [9] E. Özcan and T. Erçal, “A genetic algorithm for generating improvised music,” in *Artificial Evolution*, ser. Lecture Notes in Computer Science, vol. 4926. Springer, 2008, pp. 266–277.
- [10] M. A. Kaliakatsos-Papakostas, M. G. Epitropakis, A. Floros, and M. N. Vrahatis, “Interactive evolution of 8-bit melodies with genetic programming towards finding aesthetic measures for sound,” in *Evolutionary and Biologically Inspired Music, Sound, Art and Design (EvoMUSART)*, ser. Lecture Notes in Computer Science, vol. 7247. Springer, 2012, pp. 141–152.
- [11] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, “LakhNES: Improving multi-instrumental music generation with cross-domain pre-training,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019. [Online]. Available: <https://arxiv.org/abs/1907.04868>
- [12] M. Towsey, A. Brown, S. Wright, and J. Diederich, “Towards melodic extension using genetic algorithms,” *Educational Technology & Society*, vol. 4, no. 2, pp. 54–65, 2001. [Online]. Available: <https://eprints.qut.edu.au/169/>
- [13] M. Scirea, J. Togelius, P. Eklund, and S. Risi, “Affective evolutionary music composition with MetaCompose,” *Genetic Programming and Evolvable Machines*, vol. 18, no. 4, pp. 433–465, 2017.
- [14] J. D. Fernández and F. Vico, “AI methods in algorithmic composition: a comprehensive survey,” *Journal of Artificial Intelligence Research*, vol. 48, pp. 513–582, 2013.