Cartesian Genetic Programming Parameterization in the Context of Audio Synthesis

Edward Ly, *Graduate Student Member, IEEE*, and Julián Villegas, *Senior Member, IEEE*

***Abstract*—This letter presents an evaluation of the effects of elitism, recurrence probability, and prior knowledge on the fitness achieved by Cartesian Genetic Programming (CGP) in the context of DSP audio synthesis. Prior knowledge was introduced using a probabilistic learning method where the distribution of nodes in the expected solutions was used to generate and mutate new individuals. Best results were obtained with traditional elitist selection, no recurrence, and when prior knowledge was used for node initialization and mutation. These results suggest that the apparent benefits of recurrence in CGP are context-dependent, and that selecting nodes from a uniform distribution is not always optimal.**

***Index Terms*—Audio synthesis, Cartesian genetic programming, digital signal processing, evolutionary algorithms.**

I. Introduction

CARTESIAN Genetic Programming (CGP) was introduced by Miller [1] as a means for more efficient program induction compared to classical Genetic Programming (GP). Evolutionary Algorithms (EAs) such as CGP search through a space of computer programs represented as directed acyclic graphs to find one that closely resembles the desired outputs given a set of inputs. Since its introduction, various implementations in multiple programming languages have been introduced. The most notable of which include `CGP4Matlab` [2], which was successfully used for pitch estimation of piano tones, and `CGP-Library` [3], an LGPLv3-licensed C library provided by Turner and Miller for symbolic regression as well as the automatic programming of Artificial Neural Networks (ANNs). Various extensions to CGP have been proposed as well, with Recurrent Cartesian Genetic Programming (RCGP) [4] being one such extension which allows the directed graphs to be cyclic.

Many open questions still remain regarding the performance and efficacy of CGP [5]. To extend our knowledge on CGP, we evaluated the effects of three parameters in the context of digital audio synthesis: selection method, recurrence probability, and inclusion of prior knowledge in the evolutive process. In other words, we draw the probabilities of node selection and mutation from a known distribution as opposed to a uniform distribution, in a similar manner as informative priors are often preferred over non-informative priors in Bayesian data analysis.

Manuscript received 19 April 2023; revised 2 August 2023; accepted 7 August 2023. Date of publication 10 August 2023; date of current version 23 August 2023. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Daniele Giacobello. *(Corresponding authors: Edward Ly; Julián Villegas.)*

The authors are with the Graduate School of Computer Science and Engineering, University of Aizu, Aizu-Wakamatsu, Fukushima 965-8580, Japan (e-mail: d8222106@u-aizu.ac.jp; julian@u-aizu.ac.jp).

Digital Object Identifier 10.1109/LSP.2023.3304198

II. Background

Many previous approaches to sound synthesis using EAs have been done through Evolutionary Programming (EP), where the structure of the program is fixed to one or more known synthesis techniques and only the parameter values are evolved and evaluated (see, for example, [6] for a survey of basic approaches). As for GP-based approaches, Macret and Pasquier [7] used Mixed-Type Cartesian Genetic Programming (MT-CGP) to evolve DSP synthesizers in Pure Data (Pd) [8], a visual programming language for real-time audio synthesis. The use of mixed typing, however, was only needed because of Pd's distinction between audio and control signals, arbitrarily limiting the solution space of DSP programs. Ly and Villegas [9] were able to remove this distinction by evolving DSP programs in the Functional Audio Stream (FAUST) programming language [10], which operates on a singular "signal" data type, instead. In either case, extensive analysis of the CGP parameters used was yet to be conducted, which is one of the aims of this study.

Arguably, the most common CGP configuration is a $(1+4)$-Evolutionary Strategy (ES), where the best-fit individual in the population is chosen as the parent for the next generation (i.e., elitist selection), and four offspring individuals are generated via mutation-only reproduction [1], [11], although other population sizes—indicated by $(\mu+\lambda)$—are also possible (μ and λ being the number of parents and offspring, respectively). Turner and Miller [4] also showed performance improvements of recurrent CGP over acyclic CGP for finite state automation [12] and time series prediction [13] problems. Thus, $(1+4)$ in recurrent CGP became the de-facto standard in many implementations.

However, a recent study by Kalkreuth [14] shows that other evolutionary strategies, such as tournament selection in combination with subgraph crossover, outperform $(1+4)$ or even $(\mu+\lambda)$ in some symbolic regression and Boolean function problems. His findings suggest that the best CGP configuration may be problem-dependent. This raises the question of whether or not this extends to all other CGP parameters or possible evolutionary strategies as well.

The aim of this research is to determine whether the hitherto default settings of CGP achieve better accuracy than other configurations in the context of DSP. Chiefly, we re-examine the elitist selection method used in $(\mu+\lambda)$ with two other (less elitist) alternatives, as the former has initially been shown to perform best in other contexts [15]. We also compare acyclic vs. recurrent CGP via the recurrent connection probability parameter (0 for acyclic CGP and 1 for only recurrent connections), as non-zero probabilities have been shown to perform better in other contexts [4].

Finally, we compare the traditional way of initializing and mutating programs by drawing the function set primitives from a uniform distribution with drawing them from a prior known distribution. In a similar way that there are more function words (articles, conjunctions, etc.) than content words (nouns, adjectives, etc.) in speech, assigning functions to nodes in CGP could be done to reflect an underlying known distribution different from the uniform distribution. While our approach may reduce the likelihood that unexpected (but better) solutions are found, introducing prior knowledge to CGP could improve its efficiency (in convergence time) and its accuracy (in achieved fitness). A similar approach was used by Ardeh et al. [16] for classical, tree-based GP, using Probabilistic Prototype Trees (PPTs), structures which encode different probability distributions for each tree node, to generate new GP individuals. They also demonstrated that using such distributions improves accuracy over traditional GP, especially in the initial population and subsequent generations. Because the depth of each function node is not fixed in CGP, however, our approach differs from theirs in that a single probability distribution is used for all nodes, and that such a distribution is based on that of known or expected solutions rather than prior evolved solutions.

III. Methods

Our simulations were performed with a customized version of `FaustCGP` [9], a software application that generates DSP synthesizers in FAUST using a modified version of `CGP-Library`.[1] The customized version allows the setting of any recurrence probability, and implements a modified function set that satisfies closure in the interval $[-1, 1]$, values that can be represented in a floating-point audio signal. As a signal generated via additive synthesis is merely a sum of sinusoids (each with their own frequency, phase, and amplitude), a minimum viable function set may consist of nothing more than addition, multiplication, a sine wave oscillator, and Ephemeral Random Constants (ERCs), random values in $[-1, 1]$ which remain constant unless the mutation operation is applied to them. Still, it may be desirable to include additional operations so that more complex or harmonically rich sounds can be produced with fewer nodes. A sawtooth oscillator, for instance, is one example of an encapsulated function which consists of an infinite sum of sine waves. Thus, our entire function set is comprised of arithmetic operations (`sum`, `sub`, `mul`, and `div`); 1st-order Butterworth low- and high-pass filters (`lop` and `hip`, respectively); sine, sawtooth, square, and triangle wave oscillators (`sinp`, `sawp`, `sqrp`, and `trip`, respectively) whose frequency and phase arguments are set as fractions of the Nyquist frequency and 2π, respectively; and ERCs (`const`).

[1]The source code is publicly available at: https://git.sr.ht/led/faustcgp.

TABLE I
Proportion of Functions in an Additive Synthesizer ($\boldsymbol{p}_1$), and Probabilities for Each Function to be Selected in the Weighted Function Set ($\boldsymbol{p}_2$)

Function	$\boldsymbol{p}_1$	$\boldsymbol{p}_2$
`const`	0.500	0.400
`sum`, `mul`, `sinp`	0.167	0.133
`sub`, `div`, `sawp`, `sqrp`, `trip`, `lop`, `hip`	0	0.029

This software application features two mechanisms to deal with "plateaus" after a local minimum is reached. After a number of generations defined by a "soft plateau," the mutation rate increases 5% each generation, broadening the search through the solution space until either a new best-fit individual is found, resetting the mutation rate back to its original value, or the mutation rate reaches unity. After a number of generations defined by a "hard plateau," the CGP algorithm terminates to reduce computation time. The default thresholds for the "soft" and "hard" plateaus were set to 10 and 200 generations, respectively.

Different selection methods can also be set. In addition to elitism (dubbed `Elite` here), two more methods are available: one where the best-fit individual among the offspring is selected as the parent for the next generation (`Child`), and "roulette wheel" selection (`Rand`), where a parent is chosen with a probability proportional to its fitness relative to other individuals in the same generation. In any case, the best solution across all generations is output.

A. Evaluation

We attempted to replicate the steady state spectra of 250 entries randomly chosen from the Sandell Harmonic Archive (SHARC) [17]. This timbre database is comprised of 1338 entries, each containing amplitudes and phases of all harmonics up to 10 kHz from recordings of 24 orchestral instruments played with different styles at various pitches. The three aforementioned selection methods (`Elite`, `Child`, and `Rand`) were considered for the evaluation to compare levels of elitism. Three recurrent connection probabilities were also selected to measure different levels of recursion: 0 for acyclic CGP, 0.5 which removes any bias for or against recurrent connections, and 0.25 for a moderate amount of bias against recurrent connections. Finally, two function set weight distributions were used to evaluate the effects of prior knowledge on the evolved solutions: equally probable (`Eq`), and one that is based on the proportions of an additive synthesizer (`Add`). For the weighted function case, functions present in the additive synthesizer (`sum`, `mul`, `sinp`, and `const`) were chosen 80% of the time. `Const` is three times as likely as the other functions to be chosen to ensure that a variety of numerical values available as arguments to the other functions. The remaining 20% was equally divided among functions absent in the additive synthesizer. Table I summarizes the proportions and probabilities of each function in the additive synthesizer and the weighted function set case.

Fitness was measured as the Euclidean distance between 40 Mel-Frequency Cepstral Coefficients (MFCCs) [18], as minimizing the perceptual differences between the synthesized

TABLE II
ANOVA Type II Wald Chi-Squared Test Results for the Fitted LMM

Factor	χ^2	*df*	*p*
`pitch`	120.1	69	< 0.001
`rec`	1012.6	2	< 0.001
`select`	813.4	2	< 0.001
`weights`	33.3	1	< 0.001
`pitch:rec`	241.4	138	< 0.001
`pitch:weights`	121.0	69	< 0.001
`select:weights`	10.9	2	0.004
`select:rec`	23.4	4	< 0.001
`weights:rec`	14.4	2	< 0.001

sounds is desired. These coefficients were obtained from the CGP-generated audio and those of an additive synthesizer programmed in FAUST that perfectly replicates each SHARC tone. Such coefficients were taken from 48 triangular filter banks equally spaced in the mel scale from 20 to 10240 Hz. Other parameters such as the genotype length, initial mutation rate, distance threshold, and the maximum number of generations to evaluate remained constant throughout the evaluation (64, 0.05, 0.1 and 1000, respectively). Finally, a total of 4500 seeded simulations were conducted, one for each combination of timbre, selection method, function set weights, and recurrence probability.

IV. Results

The experiment was executed on a Mac mini (running Arch Linux), with 8GB of RAM and an 8-core Intel i7-2635QM CPU. Each CGP process was given its own CPU core via GNU `parallel` [19] to minimize computation time. Despite this, the entire experiment still required 117.6 h to complete. The mean time of each CGP process was ∼12.5 min, although the vast majority of this time is actually due to the compilation of candidate FAUST programs rather than to the evolutionary process itself.

Fitness results were analyzed with a series of Linear Mixed-Effects Models (LMMs) eased with the `lme4` library [20] in R [21]. LMMs are suitable for this analysis as we want to measure the effects of both fixed and random factors [22]. The MFCC distance was set as the dependent variable, while the selection method (`select`), prior knowledge initialization (`weights`), recurrence probability (`rec`), and the MIDI note corresponding to a given timbre (`pitch`, ranging from C1 = 32.73 Hz to F#7 = 2.963 kHz), were considered as explanatory variables. Starting with a model only including timbre (entries in the SHARC database) as a random factor, more complex models were built and accepted if the additional complexity resulted in significantly better fit (with a 95% significance level), as assessed with pairwise ANOVA tests. The final model included significant effects of all four explanatory variables and their two-way interactions except that between `select` and `pitch` ($\chi^2_{138} = 128$, $p = 0.720$), as summarized in Table II. All three-way and four-way interactions were not significant.

A post-hoc analysis based on Tukey's Honest Significant Difference (HSD) of the Estimated Marginal Means (EMMs) [23] shows that best mean fitness (shortest distance) was achieved when the elitist selection was applied to acyclic CGP, as shown in Fig. 1(a). Fitness worsened with recurrence probability, and differences in fit within levels of recurrence probability increased with less elitist selection methods; the roulette wheel selection yielded the greatest differences. Fig. 1(b) shows that the default, equally distributed, function set yielded worse fit for recurrence probabilities > 0, and that the differences in fitness achieved by the two function sets also increased with recurrence probability. Fig. 1(c) shows that prior knowledge initialization yielded better fit for the elitist selection method.

The interactions with pitch are shown in Fig. 2. This figure indicates that best fits were achieved at low and high pitches, but not for mid-pitches. The weighted function set was more likely to yield better fitness than the default function set for mid-pitches, as shown in Fig. 2(a). In a similar manner, Fig. 2(b) indicates that acyclic CGP was more likely to yield better fitness within the same frequency range.

V. Discussion

Our results are in agreement with previous findings for which the conventional elitist selection method outperforms others. In contrast, we found that acyclic CGP outperformed recurrent CGP in our problem, suggesting that the apparent benefits of RCGP are problem-dependent. Note that despite feedback loops being commonplace in DSP programming, such as in the building of Infinite Impulse Response (IIR) filters, sawtooth waves, etc., they worsen the fitness achieved in our case. We speculate that acyclic graphs may be more suitable structures for additive synthesis and steady state tones in particular, as additive synthesizers do not require any recurrent connections at all. An increase in the solution space by introducing feedback loops into candidate programs, then, may also increase the possibility of exploring and converging towards sub-optimal solutions. Such feedback loops also introduce the possibility that non-steady state sounds are returned by the evolution, which are not desirable in our case. Thus, in such cases where the optimal solutions are known to be acyclic, excluding cyclic graphs from the solution space seems to be advisable.

When recursion was allowed, we observed better fitness for the weighted function set compared to the default function set. The largest differences in fitness occurred with elitist selection and a recurrence probability of 0.5. Thus, our findings support the hypothesis that assigning functions to nodes in CGP based on an underlying known distribution (the proportions found in an additive synthesizer in our case) is more beneficial than assigning them based on a uniform distribution as is traditionally performed. However, such benefits are more pronounced with increasing recurrence probability and more elitist selection methods. One possible explanation for this phenomenon is the elimination of length bias in recurrent CGP, whereby each function node is equally likely to be activated regardless of their position in the chromosome [24]. This in turn allows for longer program phenotypes, and thus more complex tones, to be produced, a finding that is supported by our evaluation: the mean number of active nodes was 7.56 when acyclic CGP was

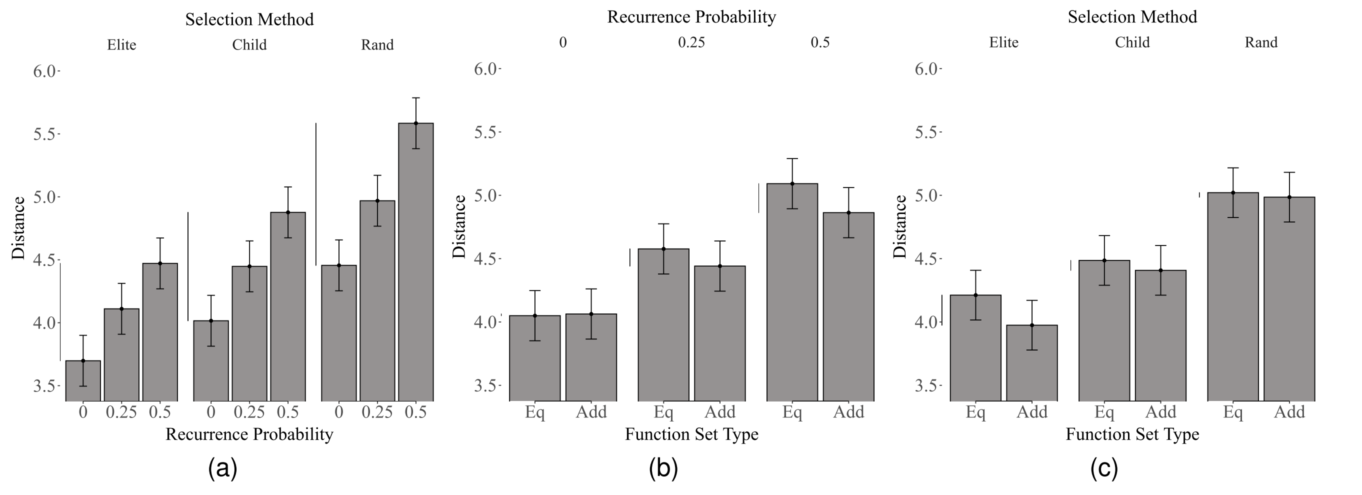

Fig. 1. Mean MFCC distances and 95% confidence intervals: (a) by recurrence probability and selection method, (b) by function set type and recurrence probability, and (c) by function set type and selection method.

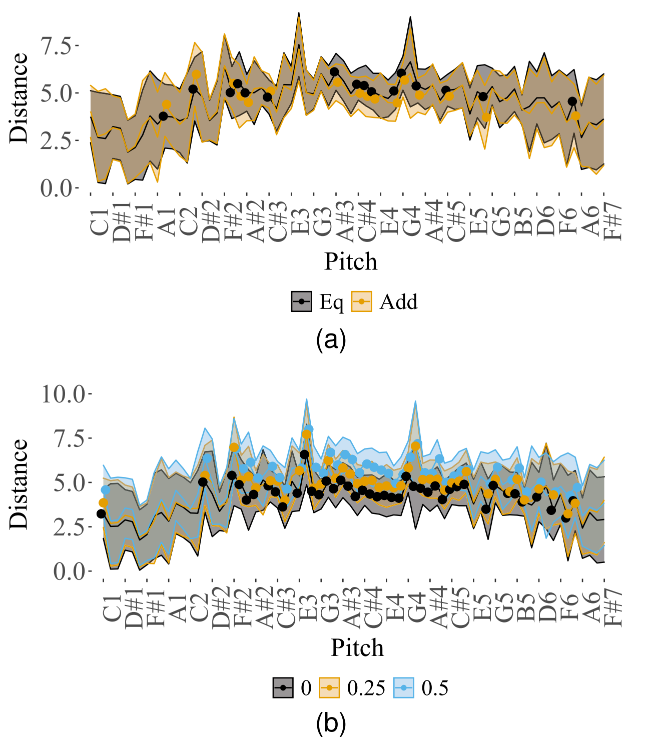

Fig. 2. Mean MFCC distance by pitch and function set type (a), and recurrence probability (b). Shaded areas around lines indicate 95% confidence intervals. Dots indicate where significant differences within pitch were found. Distances between pitch marks are not scaled.

used, compared to 15.63 and 21.18 for recurrence probabilities of 0.25 and 0.5, respectively.

Such short phenotypes, as in our case, may still adversely affect the fitness achieved by the algorithm, as additive synthesis requires a number of nodes proportional to the number of desired frequency components. Assuming the minimum viable function set for an additive synthesizer (`sum`, `mul`, `sinp`, `const`), a chromosome length of 64 can ideally hold up to 10 frequency components, which is well insufficient for more complex tones, especially those corresponding to low pitches that may contain hundreds of frequency components. We argued that including harmonically rich functions (as those indicated in Table I) could help to compensate for the lack of nodes. However, our results show no significant differences between the two function sets at low pitches, while the weighted function set was more likely to trump the default function set at mid-pitches. Recall that the latter could assign harmonically rich functions to nodes more freely than the weighted function set. Since the inclusion of these harmonically rich functions did not yield fitness gains for the default function set, we argue that the observed fitness gains of the weighted function set must come from the prior knowledge used in this case. The arbitrary 80% to 20% split we imposed in the weighted function set could then hinder the capabilities of this set to find even better fitness. Whether to eliminate such split or modify it is deferred to future research.

Our findings are encouraging and far from complete. Future investigations will explore the synthesis of non-steady state sounds to determine other CGP parameters that may be context-dependent, or whether other CGP variants perform better for DSP programming in general. Though our focus is on applications of CGP in audio DSP, we believe that the ideas introduced here (such as the inclusion of prior knowledge through weighting of the function set primitives) may benefit other applications of graphical evolution in general.

VI. Conclusion

We determined that acyclic CGP, elitist selection, and inclusion of prior knowledge in the initialization and mutation of candidate programs significantly outperformed other evaluated options for steady state additive synthesis, indicating that the superiority of RCGP over CGP cannot be generalized to all problem domains. At least for our audio synthesis problem, increasing the recurrence probability worsens the resulting fitness, in a similar manner that selection methods less elitist than the $(1+4)$-ES also does. Modifying the distribution of function set primitives based on prior knowledge also shows promising results as it improves the fitness of candidate programs over the traditional (equal) distribution in our case.

REFERENCES

[1] J. F. Miller, "An empirical study of the efficiency of learning Boolean functions using a Cartesian genetic programming approach," in *Proc. 1st Annu. Conf. Genet. Evol. Comput.*, 1999, pp. 1135–1142.

[2] R. Miragaia, G. Reis, F. Fernandéz, T. Inácio, and C. Grilo, "CGP4Matlab–a Cartesian genetic programming MATLAB toolbox for audio and image processing," in *Proc. Int. Conf. Appl. Evol. Computation*, 2018, pp. 455–471, doi: 10.1007/978-3-319-77538-8_31.

[3] A. J. Turner and J. F. Miller, "Introducing a cross platform open source Cartesian genetic programming library," *Genet. Prog. Evolvable Mach.*, vol. 16, pp. 83–91, 2014, doi: 10.1007/s10710-014-9233-1.

[4] A. J. Turner and J. F. Miller, "Recurrent Cartesian genetic programming," in *Proc. 13th Int. Conf. Parallel Prob. Solv. Nature*, 2014, pp. 476–486, doi: 10.1007/978-3-319-10762-2_47.

[5] J. Miller, "Cartesian genetic programming: Its status and future," *Genet. Prog. Evol. Mach.*, vol. 21, no. 1–2, pp. 129–168, 2020, doi: 10.1007/s10710-019-09360-6.

[6] A. Horner, "Auto-programmable FM and wavetable synthesizers," *Contemporary Music Rev.*, vol. 22, no. 3, pp. 21–29, 2003, doi: 10.1080/0749446032000150852.

[7] M. Macret and P. Pasquier, "Automatic design of sound synthesizers as pure data patches using coevolutionary mixed-typed Cartesian genetic programming," in *Proc. Annu. Conf. Genet. Evol. Comput.*, 2014, pp. 309–316, doi: 10.1145/2576768.2598303.

[8] M. Puckette, "Pure data: Another integrated computer music environment," in *Proc. 2nd Intercollege Comp. Music Concerts*, 1996, pp. 37–41.

[9] E. Ly and J. Villegas, "Additive synthesis via recurrent Cartesian genetic programming in FAUST," in *Proc. 153rd Audio Eng. Soc. Conv.*, 2022, pp. 1–7.

[10] Y. Orlarey, D. Fober, and S. Letz, "FAUST: An efficient functional approach to DSP programming," in *Proc. New Comput. Paradigms Comp. Music*, 2009, pp. 65–96.

[11] P. Kaufmann and R. Kalkreuth, "Parametrizing Cartesian genetic programming: An empirical study," in *Proc. 40th Annu. German Conf. AI*, 2017, pp. 316–322, doi: 10.1007/978-3-319-67190-1_26.

[12] D. Jefferson et al., "Evolution as a theme in artificial life: The genesys/tracker system," in *Proc. Artif. Life II: Proc. 2nd Workshop Artif. Life*, 1990, pp. 549–578.

[13] SILSO World Data Center, "The international sunspot number the international sunspot number," in *Proc. Int. Sunspot Number Monthly Bull. Online Catalogue*, 1700–1987.

[14] R. Kalkreuth, "A comprehensive study on subgraph crossover in Cartesian genetic programming," in *Proc. 12th Int. Joint Conf. Comput. Intell.*, 2020, pp. 59–70, doi: 10.5220/0010110700590070.

[15] J. F. Miller and P. Thomson, "Aspects of digital evolution: Geometry and learning," in *Proc. 2nd Int. Conf. Evolvable Syst.: From Biol. Hardware*, 1998, pp. 25–35, doi: 10.1007/BFb0057604.

[16] M. A. Ardeh, Y. Mei, and M. Zhang, "Genetic programming hyper-heuristics with probabilistic prototype tree knowledge transfer for uncertain capacitated arc routing problems," in *Proc. IEEE Congr. Evol. Comput.*, 2020, pp. 1–8, DOI: 10.1109/CEC48606.2020.9185714.

[17] G. J. Sandell, "A library of orchestral instrument spectra," in *Proc. Int. Comp. Music Conf.*, 1991, pp. 98–101.

[18] H. Terasawa, M. Slaney, and J. Berger, "The thirteen colors of timbre," in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2005, pp. 323–326, doi: 10.1109/ASPAA.2005.1540234.

[19] O. Tange, "GNU Parallel: The command-line power tool," *;login: The USENIX Mag.*, vol. 36, no. 1, pp. 42–47, 2011.

[20] D. Bates, M. Mächler, B. Bolker, and S. Walker, "Fitting linear mixed-effects models using lme4," *J. Stat. Soft.*, vol. 67, no. 1, pp. 1–48, 2015, doi: 10.18637/jss.v067.i01.

[21] R Core Team, "R: A language and environment for statistical computing," R. Foundation for Statistical Computing, Vienna, Austria, 2023. [Online]. Available: https://www.R-project.org/

[22] X. A. Harrison et al., "A brief introduction to mixed effects modelling and multi-model inference in ecology," *Peer J.*, vol. 6, 2018, Art. no. e4794, doi: 10.7717/peerj.4794.

[23] R. V. Lenth, "Emmeans: Estimated marginal means, aka least-squares means," 2023, R package version 1.8.5. [Online]. Available: https://CRAN.R-project.org/package=emmeans

[24] B. W. Goldman and W. F. Punch, "Length bias and search limitations in Cartesian genetic programming," in *Proc. 15th Annu. Conf. Genet. Evol. Comput.*, 2013, pp. 933–940, doi: 10.1145/2463372.2463482.