

Generative Art via Grammatical Evolution

Erik Fredericks, Abigail Diller, and Jared Moore
GI2023 → May 20, 2023

Motivation



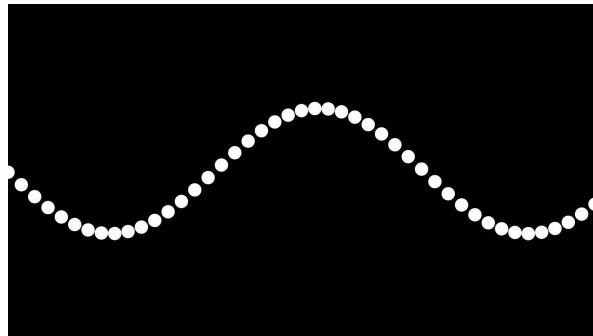
Daniel Shiffman - TheCodingTrain



Background - Generative Art

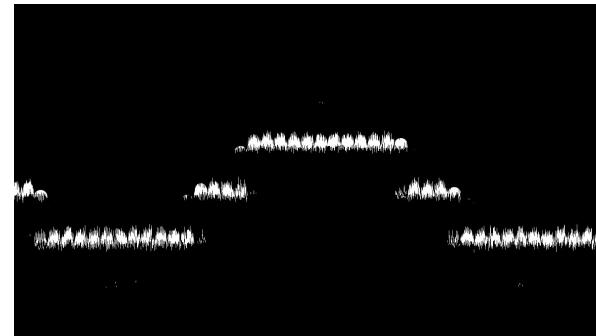
Visualization of algorithms and/or mathematical functions [1-5]

- Creative coding [11-15]
- Real-world displays [16]



$$y = 75.0 * \sin(x)$$

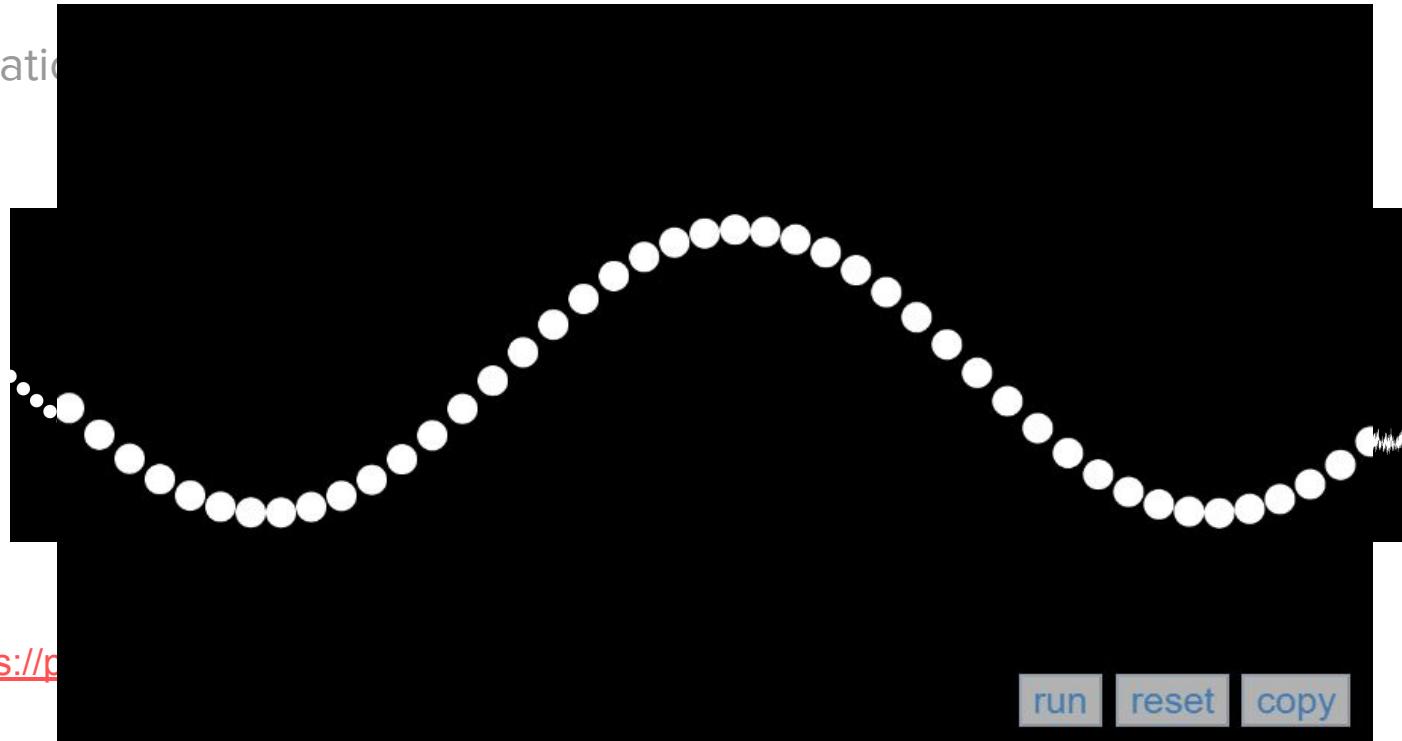
<https://p5js.org/examples/math-sine-wave.html>



Pixel sorted

Background - Generative Art

Visualization

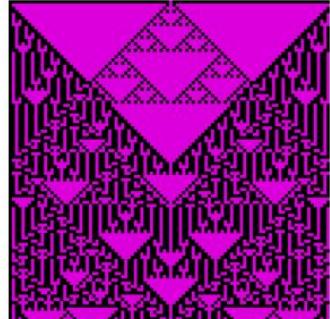


Included techniques

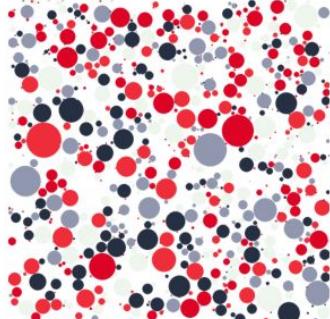
Stippling
Cellular automata
Circle packing
Flow fields (two implementations)
Drunkard's walk
Dithering
Pixel sorting
– any technique is viable!



(a) Stippled



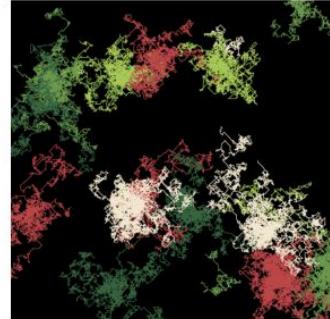
(b) Cellular Automata



(c) Circle Packing



(d) Flow Field



(e) Drunkard's Walk

Background - Grammatical Evolution

Subset of evolutionary computation (generally genetic programming-based) [24]

- Grammar-based genome [6-7]
- Uses very similar genetic operators (e.g., mutation, crossover, etc.)

Effective at constraining the *solution space*

- Target for genetic improvement! [7]

```
rules = {
    'ordered_pattern': ['#techniques#'],
    'techniques': ['#technique#','#techniques#,#technique#'],
    'technique': ['stippledBG','flowField'],
    ...
}
```

Tracery [9] grammar

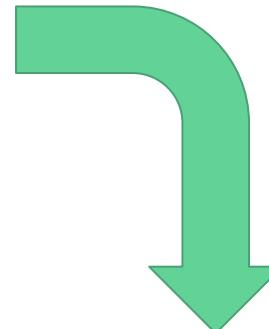
Background - Grammars

Top-level rule is "flattened" and production rules expanded

- Tracery enables randomness (among other things)

```
rules = {  
    'ordered_pattern': ['#techniques#'],  
    'techniques': ['#technique#','#techniques#,#technique#'],  
    'technique': ['stippledBG','flowField'],  
    ...  
}
```

Tracery [9] grammar



```
stippledBG(paramsa),stippledBG(paramsb),flowField(paramsc),stippledBG(paramsd)
```

Expanded output

Tracery [9]

<http://www.crystalcodepalace.com/traceryTut.html>

The screenshot shows a web browser displaying a tutorial for the Tracery library. The page has a pink and purple geometric background.

Tutorial: Tracery

Tracery is a JavaScript library, by GalaxyKate, that uses grammars to generate surprising new text. It's already been used to generate text in several released games and projects, but we're still discovering what it's for!

Tracery is still a work in progress, and we hope to soon have a code-free, hosted online version, but this tutorial will help you get Tracery working with an existing Javascript project.

This is an interactive tutorial that will teach you the basics of Tracery syntax. For a real project, you'd import the library, create a grammar, and then create new text with mygrammar.flatten(someRule), with detailed instructions at the [GitHub repo](#).

this tutorial is also a work in progress! send feedback to kate {{galaxykate at gmail}}

1: Generating text!

Tracery uses grammars to generate text. A replacement grammar takes a starting symbol, and replaces it with one of several rules

On the right (or below, depending on your screen size), you can see the JSON object that represents this very simple grammar. Farther right is the current start symbol (animal) and a list of possible texts that it can generate.

Press the reroll button a few times to generate more. Not very interesting, yet, so let's get more complicated!

start symbol: animal

duck
kitten
zebra
owl
zebra
kitten
kitten

reroll

2: Rules within rules

Rules can call rules!

Take a look at the rule for sentence: The #color# #animal# of the #natureNoun# is called #name#. Each of the words between two hashtags is a symbol to be replaced.

Try adding your name to the list of names, surrounded by quotation marks like all the other options. It will now appear in some of the generated text!

Note: JSON is very very fussy. There must be a comma between every option, but none after the last option

start symbol: sentence

The grey scorpion of the cloud is called Azra
The black duck of the ocean is called Azra
The turquoise eagle of the sea is called Azra
The turquoise zebra of the forest is called Chiaki
The orange eagle of the tree is called Lina
The black zebra of the sky is called Azra
The white owl of the sea is called Darcy

reroll

3: Adding your own symbols

Try adding a new symbol and a set of replacement rules.

How should this story end? You could add a 'mood' symbol so that Darcv the blue raven is

start symbol: sentence

Darcy the indigo scorpion was ... something.
Darcy the purple lizard was ... something.

"sentence": ["#name# the #color# #animal# was ... something."],
"name": ["Arjun", "Yuuma", "Darcy", "Mia", "Chiaki", "Izzi", "Azra", "Lina"],
"color": ["orange", "blue", "white", "black", "grey", "purple", "indigo", "turquoise"]

Background - Lexicase Selection [10]

Many-objective selection operator

- Not pareto-based!

Evaluates individuals on an objective-by-objective basis

Each selection:

- Sample of population taken
- Comparison on first objective
 - If one individual better, selected
 - Else, advance to next objective
- If all objectives exhausted
 - Random selection

ϵ -Lexicase selection [30]

- Individuals tied if within ϵ
- Important for real-valued fitness objectives
(observable output may not change)

$$\epsilon = 0.85$$

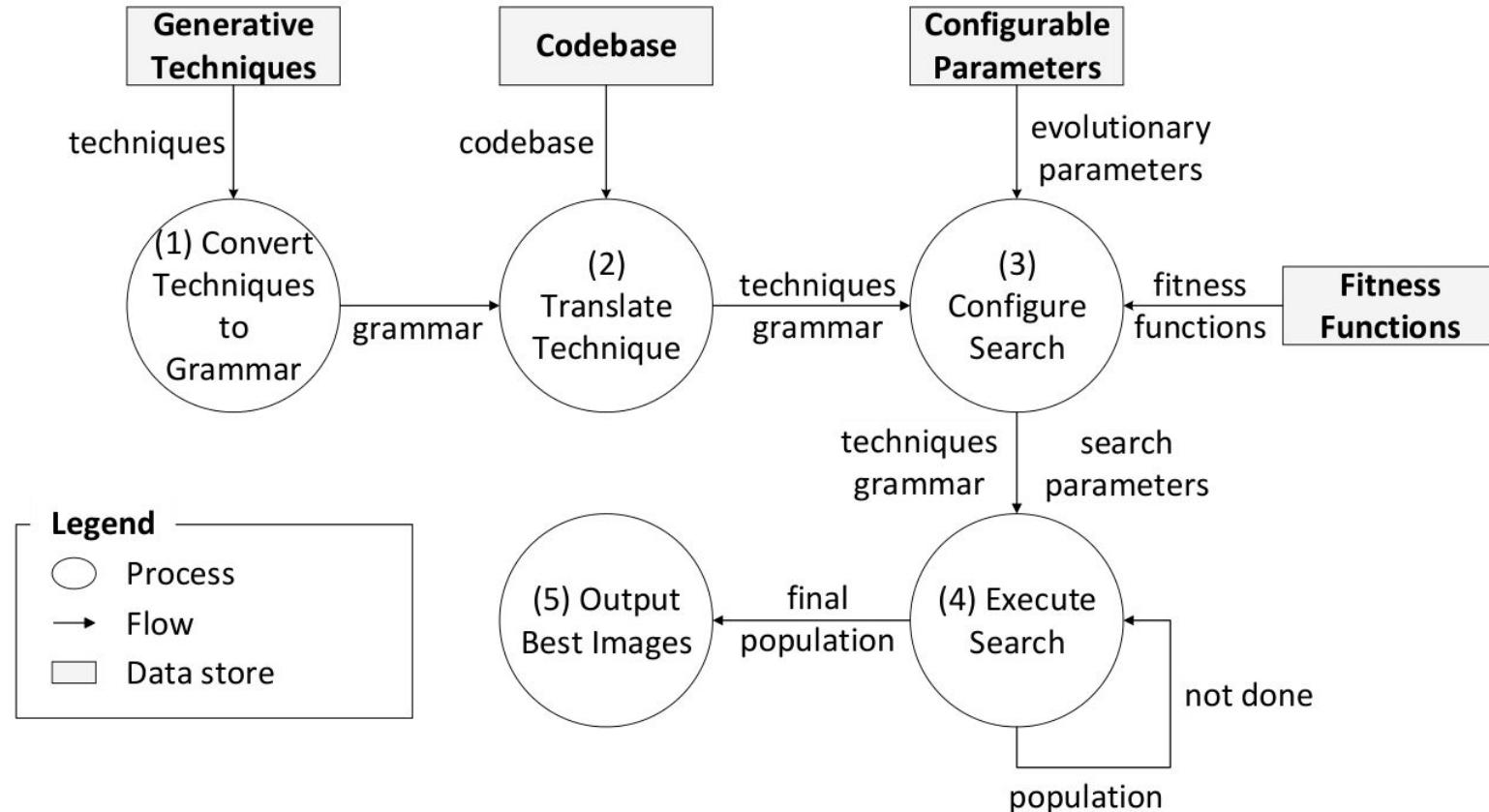
Project

Apply GI techniques to **optimize the grammar** defining the order and parameters of a set of generative art techniques

- I like glitch art, which makes for a lovely (and naive) proof of concept
 - (Pixel sorting below → <https://github.com/satyarth/pixelsort>)



Approach



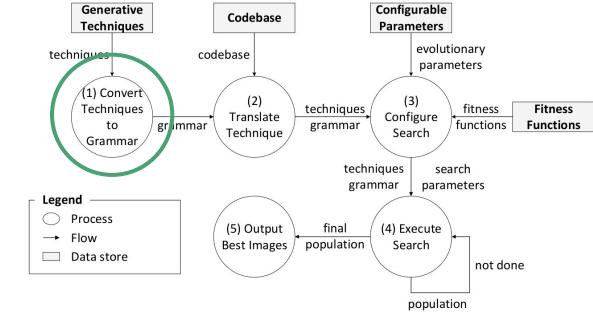
(1) Convert Techniques to Grammar

Suite of generative art techniques required as input

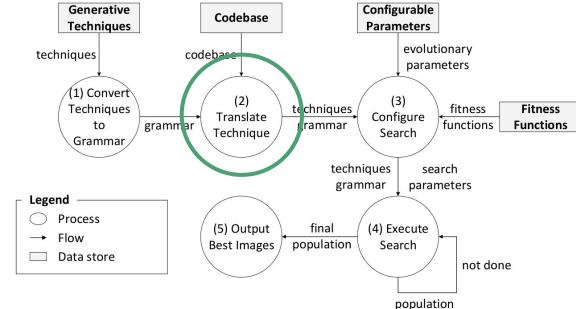
```
def flowField(type, palette, zoom):
    # type: ['edgy', 'curves']
    # palette: list of colors
    # zoom: float between 0.001 and 0.500
```



```
'flow-field' : '#flow-field-type#:#palette#:#flow-field-zoom#',
'flow-field-type' : ['edgy', 'curves'],
'flow-field-zoom': [str(x) for x in np.arange(0.001, 0.5, 0.001)],
...
```



(2) Translate Technique



Each generative art technique must also be translated to framework requirements

E.g., flow field must (minimally) accept a Pillow Image object

```
def flowField(image, type, palette, zoom)
    # image: PIL image object
    # type: ['edgy', 'curves']
    # palette: list of colors
    # zoom: float between 0.001 and 0.500
```

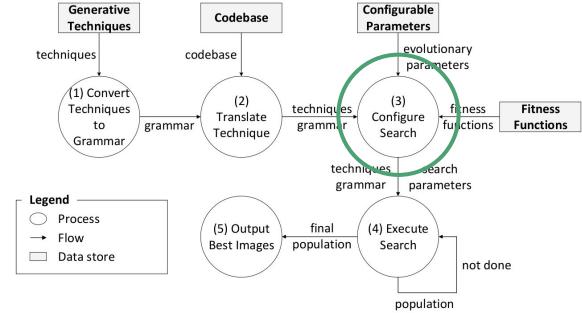
(3) Configure Search

Selection operator:

- Many-objective (Lexicase selection)
- Single-objective (Tournament selection)
- Random (No selection)

Standard configurable parameters

- E.g., population size, mutation rate, etc.



(3) Configure Search

Fitness functions

$ff_{min(genome)}$

: minimize duplicate genes

$ff_{max(techniques)}$

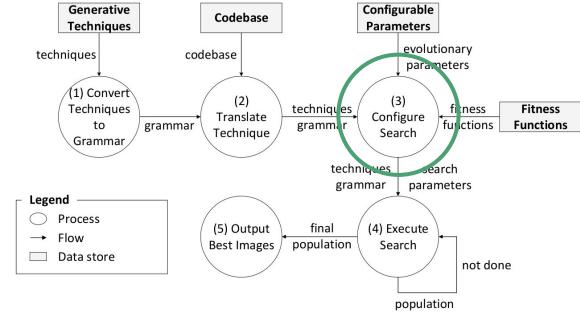
: maximizing diversity of included techniques

$ff_{max(RMS|Chebyshev)}$

: maximize pixel differences between images

Many-objective search uses all four fitness functions

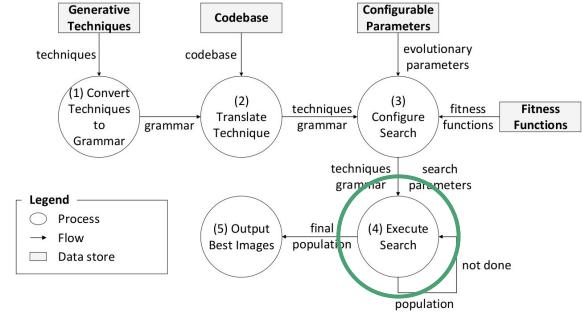
Single-objective search only uses $ff_{max(RMS)}$



(4) Execute Search

Search executed according to (3)

- Each genome evaluated on flattened grammar
- *For this work, image object **not** cleared*
 - Subject of future work

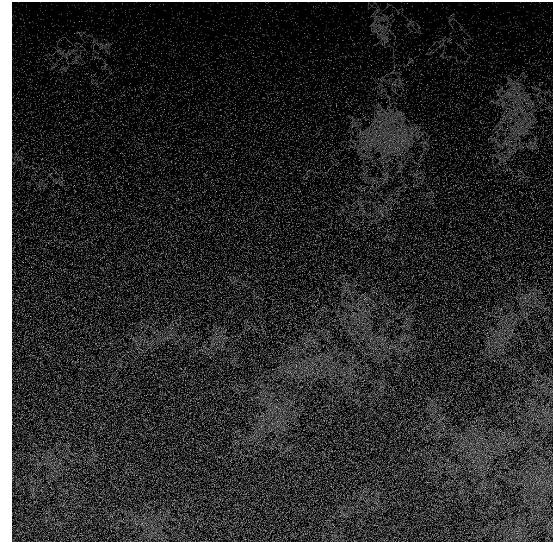
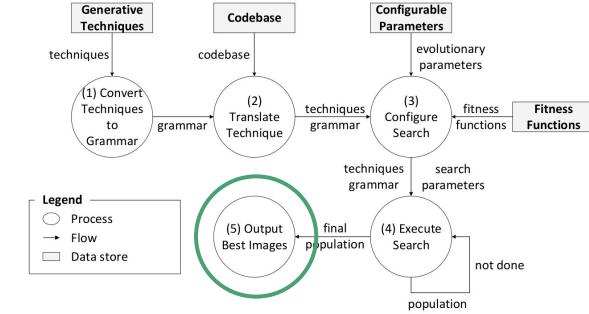
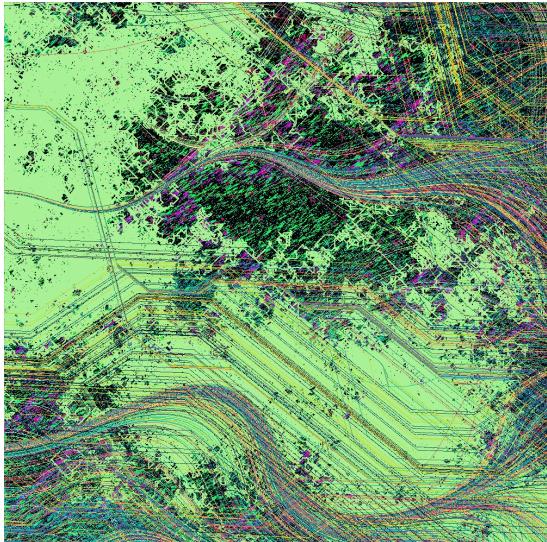


Depending on configuration

- Many-objective: Lexicase
- Single-objective: Tournament selection
- Random: No selection

(5) Output Best Images

Final population of image objects stored to disk upon completion



Experiment Configuration

Parameter	Value
Experimental replicates	10
Image size (pixels)	1000 x 1000
Number of generative techniques	8
Generations	100
Population size	100
Crossover rate	0.5
Mutation rate	0.4
Number of Lexicase objectives	4
ϵ (Lexicase - many-objective)	0.85

TABLE I: Evolutionary parameter configuration.

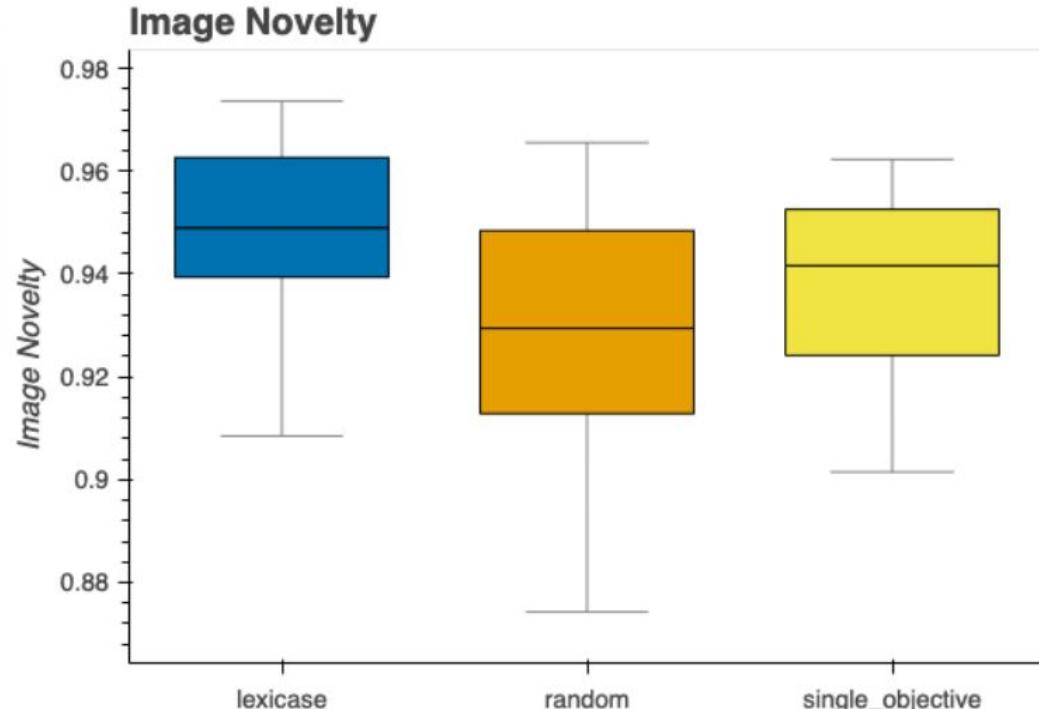
Results - Novelty

Wilcoxon rank-sum test with Bonferroni correction

Lexicase vs. random ($p < 0.01$)

Single-objective vs. random ($p < 0.03$)

Lexicase vs. single-objective ($p > 0.03$)



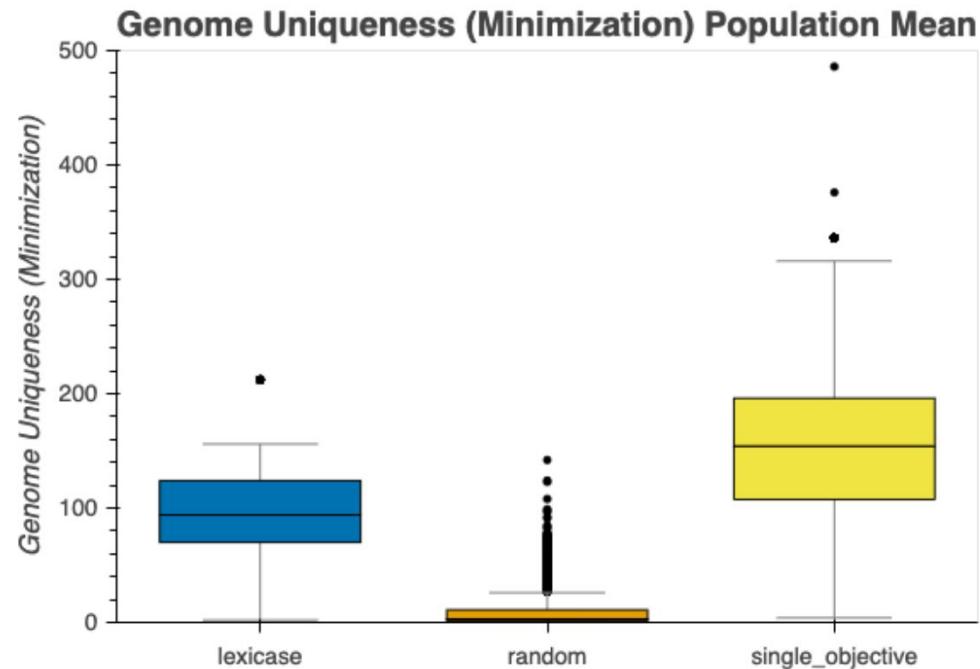
Results - Gene Uniqueness

Wilcoxon rank-sum test with Bonferroni correction

Lexicase vs. random ($p < 0.001$)

Single-objective vs. random ($p < 0.001$)

Lexicase vs. single-objective ($p > 0.001$)



Discussion / Sample Outputs

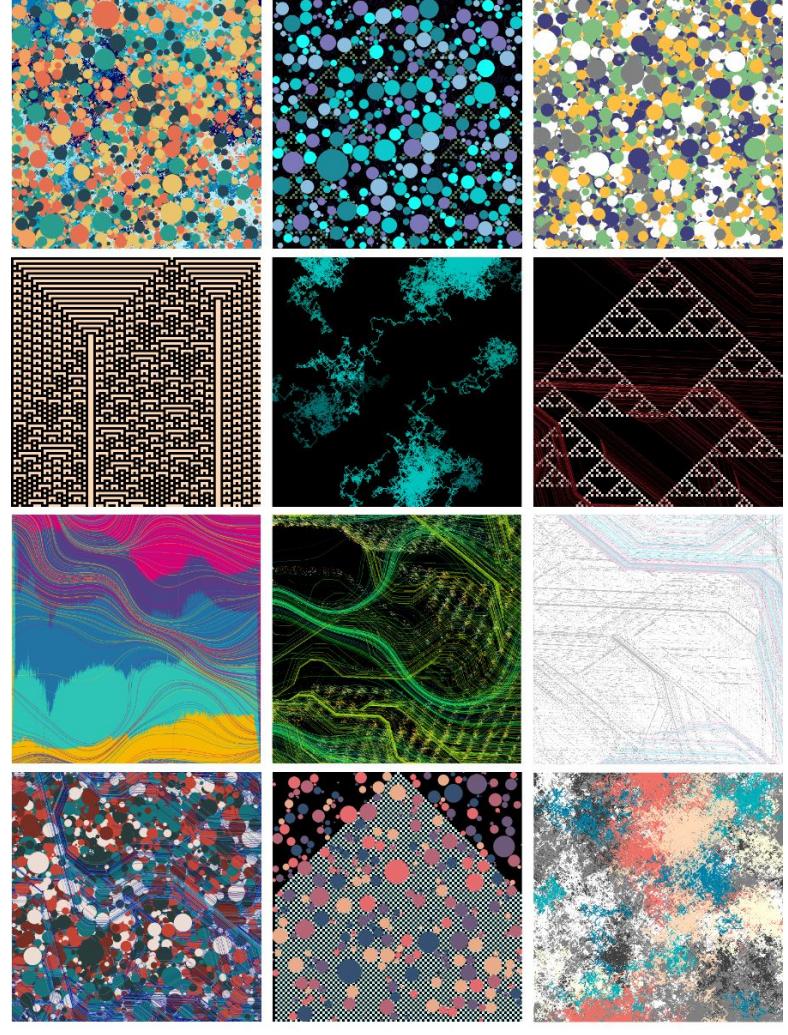
Single- and many-objective both tended to converge towards both multiple and common suites of techniques

- On a 'per run' basis
- Resulting from maximizing pixel differences and maximizing the number of techniques

Random search tended towards 'blank space'

- Fewer techniques in genome

Lexicase tended to converge to a smaller set of techniques with common outputs



Lexicase Selection

Random Generation

Single Objective

Related Work

Generative art via artificial intelligence

- Extremely popular right now thanks to large language models!
 - DALL-E, Midjourney, Stable Diffusion, VQGAN+CLIP, etc [15,35,36].
 - All require a massive dataset and massive amount of computing power
- *GenerativeGI* only requires a suite of techniques and computing necessary for evolutionary search

Everyone: AI art will make designers obsolete

AI accepting the job:



<https://www.reddit.com/r/StableDiffusion/comments/yxtdrh>

Related Work

Generative art via search heuristics

- Often used in visualization and creative coding domains
- Visualize 3D models of mathematical formulae [4]
- Creating environments within game worlds [5]
 - *GenerativeGI* focuses on fine-tuned control over artistic techniques
- Taxonomy of fitness metrics for evolutionary art/music [8]
 - Metrics can be non-trivial to evaluate (e.g., human preference)
 - Target for future work of this paper

Future Work

Additional fitness functions

- Guide towards specific outputs

Human in the loop

- What constitutes a "good" output?
- How do you measure aesthetic preference?

Merging artistic techniques

- How can distinct techniques provide a seamless output?
 - E.g., a flow field into an automata



Thanks to..



Award 80NSSC20M0124

Q&A



References

- [1] M. A. Boden and E. A. Edmonds, “What is generative art?” *Digital Creativity*, vol. 20, no. 1-2, pp. 21–46, 2009.
- [2] A. G. Forbes, T. H'ollerer, and G. Legrady, “Generative fluid profiles for interactive media arts projects,” in *Proceedings of the Symposium on Computational Aesthetics*, 2013, pp. 37–43.
- [3] B. Cabral and L. C. Leedom, “Imaging vector fields using line integral convolution,” in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, 1993, pp. 263–270.
- [4] H. Liu and X. Liu, “Generative art images by complex functions based genetic algorithm,” *Trends in Computer Aided Innovation*, pp. 125–134, 2007.
- [5] T. D. Smedt, L. Lechat, and W. Daelemans, “Generative art inspired by nature, using nodebox,” in *European Conference on the Applications of Evolutionary Computation*. Springer, 2011, pp. 264–272.
- [6] C. Ryan, J. Collins, and M. O. Neill, “Grammatical evolution: Evolving programs for an arbitrary language,” in *Genetic Programming*. Springer, 1998, pp. 83–96.
- [7] W. B. Langdon, “Genetic improvement of genetic programming,” in *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2020, pp. 1–8.
- [8] C. G. Johnson, “Fitness in evolutionary art and music: a taxonomy and future prospects,” *Fitness in evolutionary art and music: a taxonomy and future prospects*, vol. 9, no. 1, pp. 4–25, 2016.
- [9] K. Compton, B. Kybartas, and M. Mateas, “Tracery: an author-focused generative text tool,” in *International Conference on Interactive Digital Storytelling*. Springer, 2015, pp. 154–161
- [10] L. Spector, “Assessment of problem modality by differential performance of Lexicase selection in genetic programming: A preliminary report,” in *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*. Philadelphia, Pennsylvania, USA: ACM, 2012, pp. 401–408.

References

- [11] I. Greenberg, *Processing: creative coding and computational art*. Apress, 2007.
- [12] I. Bergstrom and R. B. Lotto, “Code bending: A new creative coding practice,” *Leonardo*, vol. 48, no. 1, pp. 25–31, 2015.
- [13] K. Peppler and Y. Kafai, “Creative coding: Programming for personal expression,” vol. 30, no. 2008, p. 314, 2005.
- [14] D. Shiffman, S. Fry, and Z. Marsh, *The nature of code*. D. Shiffman, 2012.
- [15] N. Dehouche and K. Dehouche, “What is in a text-to-image prompt: The potential of stable diffusion in visual arts education,” arXiv preprint arXiv:2301.01902, 2023.
- [16] L. S. Vestergaard, J. Fernandes, and M. Presser, “Creative coding within the internet of things,” in *2017 Global Internet of Things Summit (GloTS)*. IEEE, 2017, pp. 1–6.
- [24] J. R. Koza, “Genetic programming as a means for programming computers by natural selection,” *Statistics and Computing*, vol. 4, no. 2, pp. 87–112, 1994.
- [30] W. La Cava, L. Spector, and K. Danai, “Epsilon-lexicase selection for regression,” in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. Denver, Colorado, USA: ACM, 2016, pp. 741–748.
- [35] K. Crowson, S. Biderman, D. Kornis, D. Stander, E. Hallahan, L. Castricato, and E. Raff, “Vqgan-clip: Open domain image generation and editing with natural language guidance,” in *European Conference on Computer Vision*. Springer, 2022, pp. 88–105.
- [36] J. Ploennigs and M. Berger, “Ai art in architecture,” arXiv preprint arXiv:2212.09399, 2022.