Generative Art via Grammatical Evolution

Présentation d'article

Hugo Mouton

Polytechnique Montréal

12 mars 2025

Table of contents

Contexte

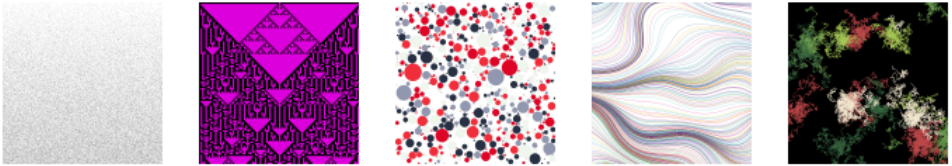
- *Generative Art via Grammatical Evolution*, 2023
- Présentation de *GenerativeGI*, un framework expérimental de génération d'art
- Utilisation de l'Évolution Grammaticale (GE), technique issue des algorithmes génétiques
- Pas vraiment d'algorithme mais plutôt une combinaison de techniques artistiques et une optimisation des paramètres

Techniques d'art génératif utilisées

Restriction à 8 techniques génératives :

- Stipple
- Cellular Automata
- Pixel Sorting
- Circle Packing
- Flow Field (2 implémentations différentes)
- Drunkard's Walk
- Dithering

Techniques d'art génératif utilisées



(a) Stippled (b) Cellular Automata (c) Circle Packing (d) Flow Field (e) Drunkard's Walk

Figure 1 – 5 des techniques utilisées

Qu'est ce que l'Évolution Grammaticale ?

- Issue des concepts de la programmation évolutive et des algorithmes génétiques
- Recherche d'une solution dans un espace contraint par des règles grammaticales établies
- Notion de *génome* = mot de la grammaire choisie

Tracery

- Framework de génération de texte par application de règles grammaticales
- Symboles : les différentes techniques et leurs paramètres
- Mots/Génomes : suites de techniques ainsi que leurs paramètres
- Nombre limite de transformations dans le cas où il y a peu de symboles terminaux

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

Figure 2 – Représentation Python de la structure de *Tracery*

Mutations de génomes

Dans *GenerativeGI*, 2 types de mutations sont autorisées :

1. Sélection d'une technique à un indice aléatoire du génome et remplacement par une autre technique (ou une suite récursive de techniques)
   `flowField('edgy', 0.1)` → `dither('halftone'),flowField('edgy', 0.1)`
2. Modification des paramètres associés à la technique
   `flowField('edgy', 0.1)` → `flowField('edgy', 0.025)`

Croisement de génomes

Une autre façon de créer de nouveaux individus est le croisement de génomes.
On choisit un symbole aléatoire dans chaque génome et on échange les 2 symboles.
Génome 1 : `dither('halftone'),flowField('edgy', 0.1)`
Génome 2 : `flowField('edgy', 0.025),flowField('edgy', 0.025)`
On échange les derniers symboles de chaque génome :
Nouveau génome 1 : `dither('halftone'),flowField('edgy', 0.025)`
Nouveau génome 2 : `flowField('edgy', 0.025),flowField('edgy', 0.1)`

Lexicase Selection

- Algorithme de sélection de parents dans les algorithmes génétiques
- Choix des parents à une certaine génération pour créer les individus enfants de la prochaine génération
- Algorithme de recherche de solutions dans un espace en utilisant un ensemble de fonctions objectif
- Domaines : programmation génétique, robotiques, géosciences, etc...
- Cadre de l'article : Utilisation de la variante ϵ-Lexicase Selection

Phase de sélection

Déroulement d'une phase de sélection :

1. On récupère un échantillon de la population
2. On mélange les fonctions objectifs et on évalue les individus avec la première
3. Si un individu est meilleur que tous les autres, on le conserve, sinon, on évalue les individus égaux avec la seconde fonction objectif
4. On répète jusqu'à n'avoir qu'un individu ou s'il n'y a plus de fonctions objectif
5. Si il reste des individus égaux selon toutes les fonctions objectifs, un individu est choisi aléatoirement

Le framework *GenerativeGI*

- Framework de création d'art génératif par évolution grammaticale
- Prend en entrée une suite de techniques paramétrées
- Génération d'images plus satisfaisantes
- Contrôle de la génération en fonction des préférences de l'artiste

Architecture de *GenerativeGI*

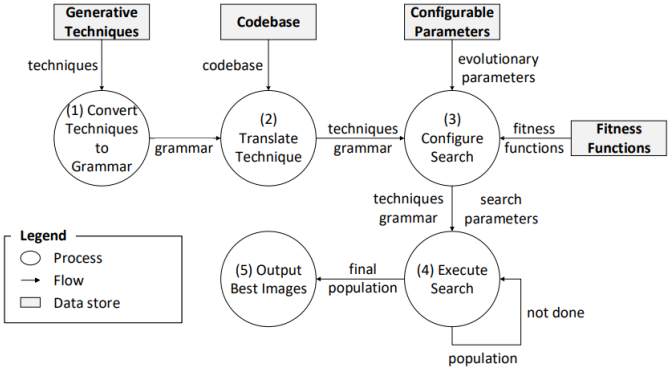


Figure 3 – Diagramme du fonctionnement de *GenerativeGI*

(1) et (2) Utilisation de Tracery

```
rules = {
  ...
  'technique'   : ['flow-field', ...],
  '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)],
  ...
}
```

Figure 4 – Exemple de conversion grammaticale dans le cas du `flow-field`

Architecture de *GenerativeGI*

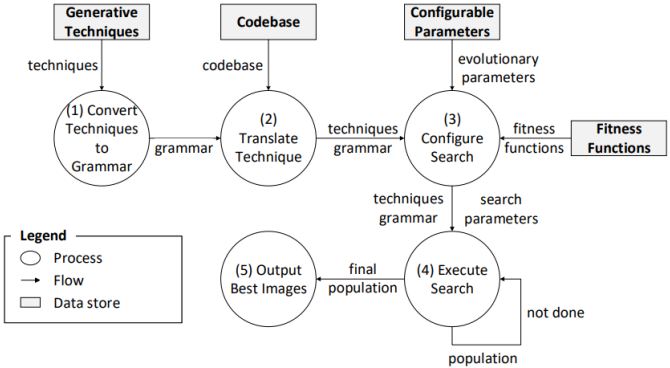


Figure 5 – Diagramme du fonctionnement de *GenerativeGI*

(3) Configuration de l'espace de recherche

- Choix du type de recherche (Lexicase Selection | Sélection aléatoire | Utilisation d'une unique fonction objectif)
- Paramétrisation de la population → #croisements, #mutations, #individus, etc...
- Fonctions objectifs choisies :
  - $ff_{\min(genome)}$ → Minimiser les doublons de génome entre les individus
  - $ff_{\max(techniques)}$ → Maximiser le nombre de techniques utilisées
  - $ff_{\max(RMS)}$ et $ff_{\max(Chebyshev)}$ → Maximiser les différences entre les images générées

Architecture de *GenerativeGI*

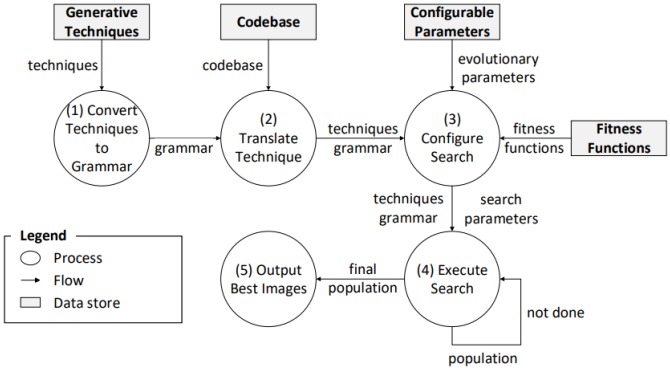


Figure 6 – Diagramme du fonctionnement de *GenerativeGI*

Configuration des expérimentations

Paramètre	**Valeur**
Nombre d'expérimentations	10
Taille de l'image	1000×1000
Nombre de techniques	8
Nombre de générations	100
Taille de la population	100
Nombre de cycles dans le cas aléatoire	50
Taux de croisement	0.5
Taux de mutation	0.4
Nombre de fonctions objectifs dans Lexicase	4
ϵ	0.85

Résultats empiriques

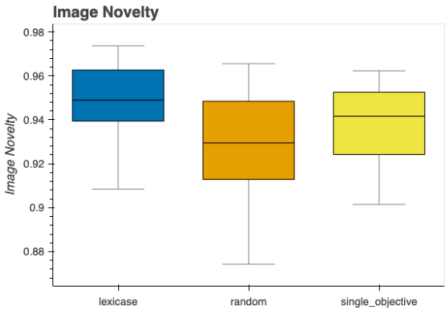


Figure 7 – Score de nouveauté de l'individu le plus "nouveau" pour chaque technique

- Calcul des scores de nouveauté de chaque méthode de sélection
- Test de Wilcoxon avec correction de Bonferroni pour comparer
- $p < 0.01$ pour Lexicase | Random
- $p < 0.03$ pour Single | Random

Résultats empiriques

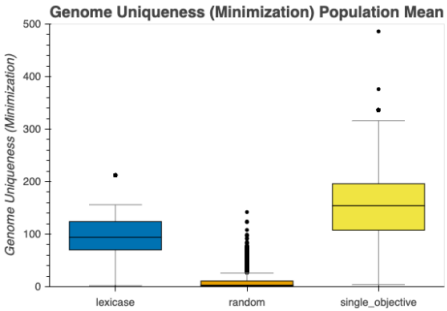
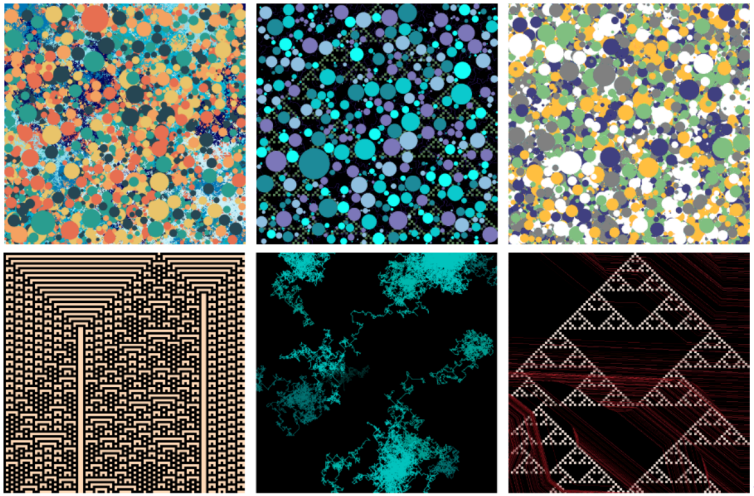


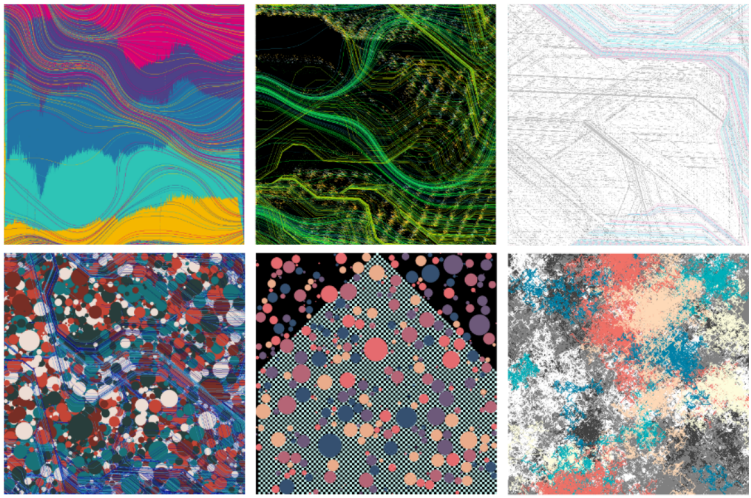
Figure 8 – Valeur moyenne de $ff_{\min(genome)}$ pour chaque technique

- Plus la valeur est proche de 0 et plus la population est diverse
- $p < 0.001$ pour Random | Lexicase et Single
- $p < 0.001$ pour Lexicase | Single
- Diversification plus importante avec Lexicase, grande variété d'images

Images générées



Images générées



Conclusion

- *GenerativeGI* se présente comme une alternative aux techniques d'IA actuelles (DALL-E, Midjourney, Stable Diffusion, etc...)
- Ne nécessite aucune données préalables
- Certaines limitations :
  - Peu de fonctions objectif utilisées $\rightarrow$ Dirige énormèment la génération
  - Faible résolution
  - Peu de techniques de génération

Merci

Merci de votre attention !