Rapport de TP : PNG to SVG

M1 IA - Université de Rennes (ISTIC)

Axel OZANGE
Zakary SADOK

Janvier 2026

Table des matières

1 Introduction

Ce projet s'inscrit dans le cadre du module de Recherche Opérationnelle. L'objectif principal est de développer une application capable de reproduire une image matricielle (PNG) en utilisant exclusivement un ensemble limité de formes géométriques (ellipses, rectangles, triangles) pour générer un rendu vectoriel au format SVG.

Ce travail de "recréation géométrique" constitue un problème d'optimisation complexe. Le défi consiste à déterminer la configuration optimale (le phénotype) — position, couleur, taille et rotation des formes — afin de minimiser la différence visuelle avec l'image originale.

Pour résoudre ce problème, nous avons implémenté et comparé plusieurs approches métaheuristiques. Nous avons débuté par une recherche aléatoire pour établir une base de comparaison, avant d'explorer des méthodes plus avancées telles que les Algorithmes Génétiques et une approche Gloutonne (Greedy) couplée à une recherche locale (Hill Climbing), dans le but d'identifier la stratégie offrant le meilleur compromis entre temps de calcul et qualité esthétique.

2 Manuel d'utilisation

L'application a été conçue pour être exécutée en ligne de commande, ou sinon via le notebook.

2.1 Prérequis et Installation

Le projet nécessite **Python 3.8** ou supérieur. Les dépendances graphiques et mathématiques sont listées dans le fichier `requirements.txt`.

```
# Installation des biblioth ques (Pillow, Numpy, etc.)
pip install -r requirements.txt
```

Listing 1 – Installation des dépendances

2.2 Syntaxe de la commande

Le script principal `png2svg.py` accepte plusieurs arguments pour contrôler le comportement de l'algorithme d'optimisation.

```
python3 png2svg.py [OPTIONS] --input <IMAGE_SOURCE>
```

Détail des arguments

`-input` : Chemin vers l'image source (formats supportés : .jpg, .png). L'image sera automatiquement redimensionnée pour le calcul de la fitness.

`-output` : Chemin du fichier de sortie. L'extension détermine le format (.svg pour vectoriel, .png pour raster). Par défaut : `output.svg`.

`-algo` : Choix de la métaheuristique.
- `greedy` (recommandé) : Approche constructive rapide et précise.
- `ga` : Algorithme génétique standard (convergence plus lente).

`-n` : Budget de formes (nombre entier). Définit la complexité maximale de l'image générée (ex : 150 formes).

`-shape` : Type de primitive géométrique à utiliser.
- `rectangle` : Idéal pour les structures urbaines ou abstraites.
- `circle` : Donne un effet de "pointillisme".
- `mixed` : Utilise un mélange aléatoire de formes.

`-time` : Contrainte de temps en secondes (Time Budget). L'algorithme s'arrêtera strictement après ce délai et sauvegardera la meilleure solution trouvée.

2.3 Exemples de scénarios d'exécution

Exemple de commande

```
python3 png2svg.py \
  --input images/portrait.jpg \
  --output resultats/portrait_final.svg \
  --n 500 \
  --time 300 \
  --algo greedy \
  --shape rectangle
```

3 Méthodologie et Évolution

3.1 Essais préliminaires et abandon du Random Search

Dans un premier temps, nous avons expérimenté une méthode de **Recherche Aléatoire (Random Search)** pour établir une ligne de base (baseline). Cette approche consiste à générer des formes avec des positions, couleurs et tailles entièrement aléatoires, sans aucune stratégie d'apprentissage. Nous avons rapidement abandonné cette méthode car les résultats se sont révélés inexploitables. En l'absence de mécanisme de guidage ou de mémorisation des essais précédents, l'algorithme ne converge pas et produit des images s'apparentant à du bruit visuel. La probabilité de reconstruire une structure cohérente par pur hasard dans un espace de recherche aussi vaste est statistiquement nulle.

3.2 Implémentation de l'Algorithme Génétique (GA)

Pour pallier le manque de convergence du Random Search, nous nous sommes tournés vers un **Algorithme Génétique**. Cette méthode d'optimisation globale gère une population de solutions complètes qui évoluent au fil des générations. L'algorithme évalue chaque individu (image) et applique des principes de sélection naturelle et de mutation aléatoire pour espérer améliorer le rendu global. Bien que supérieure au hasard pur, cette méthode tente d'optimiser simultanément l'ensemble des formes (par exemple 50 ou 100 formes à la fois), ce qui rend la convergence lente et laborieuse, l'algorithme ayant du mal à isoler l'impact d'une seule forme parmi toutes les autres.

3.3 Développement de l'approche Gloutonne (Greedy + Hill Climbing)

Afin d'améliorer la précision et la vitesse de calcul, nous avons développé une approche hybride nommée Greedy. Celle-ci décompose le problème en deux phases distinctes :

- **Construction (Greedy)** : Ajout itératif basé sur une *error map*.
- **Raffinement (Hill Climbing)** : Recherche locale (Hill Climbing) pour ajuster les paramètres.

3.4 Comparaison et justification du choix final

Après comparaison des résultats, nous avons choisi de retenir l'approche Greedy au détriment de l'Algorithme Génétique pour trois raisons techniques majeures :

- **Réduction de la complexité (Divide and Conquer)** Le GA tente de résoudre un problème unique et complexe (placer 100 formes en même temps). Le Greedy résout 100 problèmes simples séquentiellement (placer 1 forme, 100 fois de suite), ce qui est mathématiquement beaucoup plus efficace.
- **Guidage spatial** : Contrairement au GA qui mute à l'aveugle, le Greedy utilise une "carte d'erreur" (error map) pour cibler intelligemment les zones de l'image qui nécessitent une correction.
- **Qualité de convergence** : L'approche Greedy produit des images plus nettes et plus fidèles à la cible en un temps de calcul considérablement réduit, là où le GA tend à stagner sur des optimums locaux de mauvaise qualité.

4 Choix Techniques

4.1 Représentation

Le génotype est une liste d'objets `Shape` possédant des attributs normalisés : (x, y, w, h, r, g, b, a).

4.2 Fonction de Fitness

Nous utilisons une perte L_1 (Mean Absolute Error) :

$$L_1 = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Cette métrique est privilégiée car elle est moins sensible aux valeurs aberrantes que la MSE.

5 Analyse des Résultats

5.1 Exemple 1 : Random Search

— **Observations :** On reconnait difficilement l'image d'origine, l'approche a donc vite été abandonnée puisque la méthode Random Search a été jugé trop peu efficace.

FIGURE 1 – Algorithme Random Search

5.2 Exemple 2 : Comparaison de la stratégie d'optimisation

Dans cette première expérience, nous fixons les contraintes de budget ($N = 150$) et de temps ($T = 60s$) pour isoler l'impact de l'algorithme choisi.

FIGURE 2 – Algorithme Génétique

FIGURE 3 – Algorithme Glouton

Commentaire : On observe une différence nette de convergence. L'algorithme Génétique (GA) en 60 secondes n'a pas pu stabiliser une structure cohérente, il y a encore beaucoup de flou. À l'inverse, l'Algorithme Glouton est beaucoup plus efficace, il a placé précisément les 150 rectangles les plus impactants, offrant une reconstruction immédiatement reconnaissable.

TABLE 1 – Comparaison des performances avec un budget ($N = 150$) et ($T = 60s$)

Critère	**Algorithme Génétique**	**Algorithme Glouton**
Vitesse de convergence	Faible	Très élevée
Qualité visuelle	Moyenne (flou)	Bonne (contours nets)
Stabilité	Faible	Forte

5.3 Exemple 3 : Du temps et des formes !

Les contraintes utilisées : de budget ($N = 180$) et de temps ($T = 300s$) avec l'algorithme Glouton et des formes rectangulaires.

— **Observations :** On remarque bien que l'algorithme arrive bien à converger vers une solution "correcte", on arrive à bien reconnaitre Mona Lisa. Le résultats pour 180 formes et 5 minutes de temps est très satisfaisant.

FIGURE 4 – Algorithme Glouton

5.4 Exemple 4 : Glouton avec différentes formes

Les contraintes utilisées : de budget ($N = 150$) et de temps ($T = 60s$) avec l'algorithme Glouton et différentes formes.

FIGURE 5 – Rectangles

FIGURE 6 – Cercles

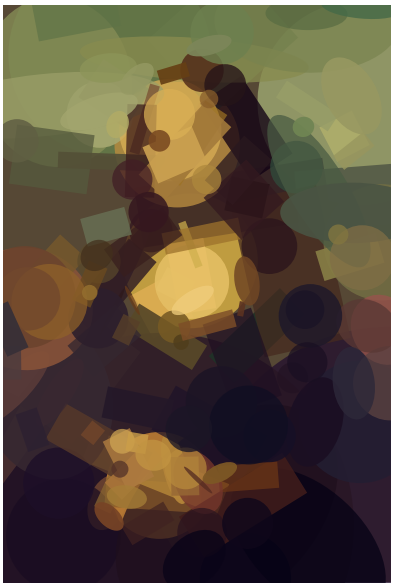

FIGURE 7 – Mixte

5.5 Exemple 5 : Images de fins avec différentes formes

Les contraintes utilisées : de budget ($N = 180$) et de temps ($T = 180s$) avec l'algorithme Glouton et des formes mixtes.

FIGURE 8 – Algorithme Glouton

Les contraintes utilisées : de budget ($N = 108$) et de temps ($T = 60s$) avec l'algorithme Glouton et des formes mixtes.

FIGURE 9 – Algorithme Glouton

6 Conclusion

Ce projet nous a permis d'explorer les défis liés à l'approximation d'images par des primitives géométriques, un problème d'optimisation complexe où l'espace de recherche est immense.

L'analyse comparative de nos trois approches a été déterminante pour identifier la solution la plus viable. L'échec initial du Random Search a rapidement démontré l'inutilité d'une recherche purement stochastique sans mécanisme de guidage. Si l'Algorithme Génétique a permis d'obtenir des résultats cohérents, il a révélé ses limites en termes de temps de calcul et de précision, peinant à optimiser simultanément un grand nombre de formes.

L'approche Gloutonne (Greedy) associée au Hill Climbing s'est finalement imposée comme la méthode optimale. En décomposant le problème global en une suite de sous-problèmes locaux (construire l'image forme après forme) et en ciblant les zones d'erreur, elle offre le meilleur compromis entre vitesse de convergence et fidélité visuelle.

Au-delà du résultat technique, ce projet illustre qu'une stratégie constructive et guidée est souvent bien supérieure à une méthode évolutionnaire globale pour des problèmes de reconstruction séquentielle. Le programme final permet ainsi de générer des reproductions artistiques et légères d'images complexes avec une efficacité redoutable.