Rendu des Travaux Pratiques sur les Algorithmes Génétiques

Raphaël Diana, Antoine Gaget

20 février 2017

Université Claude Bernard Lyon 1

Table des matières

1 Introduction

L'objectif de ce projet est d'utiliser un algorithme génétique pour simuler l'évolution de cellules photo-sensibles et vérifier si un nombre restreint de contraintes suffit à mener à la formation d'un œil comme il est décrit dans [Nilsson and Pelger, 1994].
Pour ce faire nous avons développer en C++ (Annexe 5.1) les étapes de sélection, croisement et mutation d'un algorithme génétique (présenté dans 2), inspiré par les travaux de Nilsson and Pelger [1994], et dans un second temps nous avons conçu une visualisation synthétisant nos résultats.

2 Description de l'algorithme

Notre algorithme génétique a pour but de faire évoluer une population d'individus codant les paramètres d'un œil (présentés dans 2.1) et vérifier s'ils convergent en une structure d'œil bien formé [Nilsson and Pelger, 1994].
Cependant, nous n'utilisons pas exactement les contraintes présentées dans Nilsson and Pelger [1994], à la place nous utilisons les contraintes simplifiées présentées dans Knibbe [2017].

2.1 Les paramètres à faire évoluer

Tout d'abord, il faut définir la largeur maximal de l'œil, qui est appelé w et qui est égale à 1.5 cm.
Les paramètres évolutifs de l'algorithme sont les suivants :

- ρ_c qui est rayon de courbure de l'œil, qui varie entre $w/2$ et 10000 cm.
- i représente la taille de l'iris. Il peut varier entre 0 et $w/2$ cm.
- ϕ_1 représente l'angle entre le centre de l'œil et le début de l'iris. Il varie entre 0 et $\pi/2$ radians.
- n_0 qui représente l'indice de réfraction au centre de la lentille. Il varie entre 1.35 et 1.55.

Ces paramètres constitue le génotype de chaque individu. Notre algorithme génétique va les faire muter tout au long des générations pour essayer d'atteindre les résultats de [Nilsson and Pelger, 1994].

2.2 L'algorithme

Nous allons maintenant parler précisément de l'algorithme et des méthodes utilisées.
D'abord, les variables ont été initialisées comme dans [Nilsson and Pelger, 1994] :

- $\rho_c = 10000$
- $i = 0$
- $\phi_1 = 0$
- $n_0 = 1.35$

Nous avons également un ensemble de variables globales constantes pour le fonctionnement de notre algorithme :

- NB_GEN représente le nombre de générations durant que l'algorithme va produire.
- POP_SIZE représente le nombre d'individus (ensemble de cellules photo-sensibles) par génération.
- $PRECISION$ représente la précision maximale pour les tests d'égalité. En effet, faire des égalités strictes n'allait pas avec la simplification des contraintes de Knibbe [2017], nous avons donc instauré un seuil de précision. La précision est égale à 10^{-4}.
- $CROSSOVER\_RATE$ représente le pourcentage de chance qu'un croisement arrive lors de l'étape de reproduction (voir 2.2.2).
- $MUTATION\_CHANCE$ représente la chance qu'une mutation ait lieu lors de la création d'un nouvel individu (voir 2.2.2 et 2.2.3).

- $MUTATION\_RATE$ représente le taux de mutation, i.e. la plage de pourcentage de changement de la valeur lors de la mutation (voir 2.2.3).

La fonction de *fitness* des individus est nommée $v()$ et tend à maximiser la fréquence spatiale détectable [Warrant and McIntyre, 1993].

2.2.1 Méthode de sélection

La méthode de sélection représente la manière de sélectionner les individus qui vont se reproduire ("*breed*") pour produire la génération suivante.
Dans notre algorithme, nous utilisons la méthode la sélection dite par roulette ("*Fitness proportionate selection*"), à la différence près que nous ne retirons pas l'individu tiré en premier du tirage (faute de technique). Elle consiste à l'utilisation d'une probabilité de sélection en fonction de la *fitness*[1] de l'individu.
Cette méthode permet à tout les individus d'avoir une chance, même ceux dont la *fitness* est la moins favorable.
Cependant, un cas peut poser problème : le cas où un individu a une *fitness* très élevé par rapport aux autres, par exemple les probabilités sont de 99.9% d'être tirer pour un individu et 0.1% réparti pour tout les autres. Pour palier à ce cas, nous arrêtons le tirage aléatoire après avoir tiré 5000 fois le même individu deux fois, et nous continuons l'algorithme avec deux parents identiques (i.e. l'individu le plus "fort" se reproduit avec lui-même). La méthode de sélection est présenté dans l'algorithme 1.

Data : tab : tableau de probabilités cumulées
Result : i : indice de l'élément tiré
$rand \leftarrow$ nombre réel aléatoire entre 0 et 1 ;
$i \leftarrow 0$;
while $rnd > tab[i]$ **do**
  $i \leftarrow i + 1$;
end
retourner i ;
Algorithme 1 : get_parent_index : tirage aléatoire d'un individu en fonction de sa fitness

2.2.2 Méthode de croisement (cross-over)

Lors de la reproduction, l' *"enfant"* reçoit les gènes de la part de ses *"parents"*. Si il n'y a pas croisement, soit une probabilité de $1 - CROSSOVER\_RATE$, l'enfant reçoit tout les gènes d'un des parents, parent choisit aléatoirement (50% de chance d'être choisit pour chacun des parents).
Si le croisement a lieu (probabilité de $CROSSOVER\_RATE$), alors chaque gène de l'enfant est choisit aléatoirement chez l'un des deux parents (probabilité de 0.5 pour chaque parents).
L'algorithme 2 présente la méthode de reproduction et de croisement.

2.2.3 Méthode de mutation

La mutation des gènes dans un algorithme génétique permet la diversité des individus.
Dans notre algorithme, nous faisons muter nos gènes selon une plage $MUTATION\_RATE$. Lors de la mutation, la valeur du gène va muter d'un pourcentage de sa valeur actuelle dans une plage de $[-MUTATION\_RATE/2, MUTATION\_RATE/2]$.
L'algorithme 3 décrit notre méthode de mutation.
Pour comprendre l'algorithme, il est important de noter que notre structure de données représentant un gène est composée de plusieurs valeurs :

- *value* qui représente la valeur du gène.
- *max* et *min* qui représente les bornes, respectivement maximum et minimum, que la valeur du gène peut prendre.

Aussi, le cas $g.value = 0$ est présent pour permettre aux gènes commençant avec une valeur de 0 de muter quand même (puisque sinon, les mutations se font sur un pourcentage de 0 et rien ne change).

[1] Valeur qui représente la "valeur" d'un individu, sa "force". Ici, elle est calculée selon la formule présente dans Knibbe [2017].

Data : p_1 : parent 1, p_2 : parent 2
Result : *child* : Résultat de la reproduction
child ← nouvel individu vide ;
//Cross-over
crossover ← nombre aléatoire entre 0 et 1 ;
if $crossover < CROSSOVER\_RATE$ **then**
  for *Tout les gènes g* **do**
    rand ← nombre aléatoire entre 0 et 1 ;
    if $rand < 0.5$ **then**
      $child.g \leftarrow p_1.g$;
    else
      $child.g \leftarrow p_2.g$;
    end
  end
else
  rand ← nombre aléatoire entre 0 et 1 ;
  if $rand < 0.5$ **then**
    $child.all\_genes \leftarrow p_1.all\_genes$;
  else
    $child.all\_genes \leftarrow p_2.all\_genes$;
  end
end
//Mutations
for *Tout les gènes g* **do**
  rand ← nombre aléatoire entre 0 et 1 ;
  if $rand < MUTATION\_CHANCE$ **then**
    $child.g.mutate()$;
  end
end

Algorithme 2 : *breed* : Méthode de reproduction

Data : g : un gène
Result : g : le même gène muté
if $g.value = 0$ **then**
  $change \leftarrow g.max * MUTATION\_RATE$;
  $g.value \leftarrow g.value + change$;
else
  rand ← nombre aléatoire entre 0 et 1 ;
  $change \leftarrow (MUTATION\_RATE/2 - (rand * MUTATION\_RATE))$;
  $g.value \leftarrow g.value * (1 + change)$;
end

Algorithme 3 : *mutate* : Méthode de mutation

2.2.4 Algorithme

Dans cette partie, nous allons présenter l'algorithme que nous avons utilisé pour générer les populations d'yeux. L'algorithme 4 est le pseudo-code simplifié de notre algorithme. Le code est présent en Annexe 5.1.
Lors de l'initialisation de la première génération, les paramètres sont initialisés ainsi :

- $\rho_c = 10000$
- $i = 0$
- $\phi_1 = 0$
- $n_0 = 1.35$

Data : Aucune
Result : Population d'yeux évolués après NB_GEN générations
pop : tableau d'individus de taille POP_SIZE
for *i allant de* 0 *à* $POP\_SIZE - 1$ **do**
  pop[*i*].*init*()
end
for *g allant de* 0 *à* $NB\_GEN - 1$ **do**
  $fitness$: tableau de réels de taille POP_SIZE
  sum : réel
  for fi *allant de* 0 *à* $POP\_SIZE - 1$ **do**
    $fitness[fi] \leftarrow pop[fi].v()$;
  end
  Ecriture du meilleur individu de pop *dans le fichier* timestamp_best_results.tsv
  Ecriture du pire individu de pop *dans le fichier* timestamp_worse_results.tsv
  Ecriture de la moyenne des individus de pop *dans le fichier* timestamp_avg_results.tsv
  proba : tableau de réels de taille POP_SIZE
  for fi *allant de* 0 *à* $POP\_SIZE - 1$ **do**
    $proba[fi] \leftarrow fitness[fi]/sum$;
  end
  repartition : tableau de réels de taille POP_SIZE
  $repartition[0] \leftarrow proba[0]$;
  for fi *allant de* 1 *à* $POP\_SIZE - 1$ **do**
    $repartition[fi] \leftarrow repartition[fi-1] + proba[fi]$;
  end
  new_pop : tableau d'individus vide
  while $new\_pop.size() < POP\_SIZE$ **do**
    $ip1 \leftarrow get\_parent\_index(repartition)$;
    $ip2$: entier ;
    $timeout \leftarrow 0$;
    do
      $ip2 \leftarrow get\_parent\_index(repartition)$;
      $timeout \leftarrow timeout + 1$;
    while $ip1 = ip2$ *OR* $timeout < 5000$;
    $p1 \leftarrow pop[ip1]$;
    $p2 \leftarrow pop[ip2]$;
    $child \leftarrow breed(p1, p2)$;
    if *NOT* $child.isDead()$ **then**
      $new\_pop.append(child)$;
    end
  end
  $pop \leftarrow new\_pop$;
end

Algorithme 4 : *EyeSea* : Algorithme Génétique

2.3 Visualisation

Nous avons utiliser la bibliothèque D3.js[2] pour construire une visualisation interactive permettant de voir l'évolution du phénotype du meilleur individu de chaque génération d'œil et de certains de ses paramètres. Le lien de notre visualisation est disponible dans l'annexe 5.2.

3 Analyse des résultats

Dans cette section, nous allons analyser les résultats de notre algorithme, et les comparer avec les résultats de Nilsson and Pelger [1994].

3.1 Génération des données

Tout d'abord, voici les valeurs des paramètres avec lesquelles nous avons fait tourner notre algorithme :

- $NB\_GEN = 25000$: d'après nos expérience l'algorithme converge bien avant ce nombre de générations.
- $POP\_SIZE = 300$: compromis entre diversité minimum et temps d'exécution raisonnable.
- $PRECISION = 10^{-4}$: valeur arbitraire.
- $CROSSOVER\_RATE = 0.20$: valeur assez haute pour permettre une diversité importante.
- $MUTATION\_CHANCE = 0.20$: compromis entre temps d'exécution raisonnable et oscillation limitée des paramètres (observations personnelles).
- $MUTATION\_RATE = 0.02$: valeur recommandée par Nilsson and Pelger [1994]. En effet, monter à 0.05 (mutation entre $[-0.025, 0.025]$ de la valeur du paramètre) entraîne des mutations trop grande et peu précise et empêche les paramètres d'atteindre leurs limites (observations personnelles).

3.2 Analyse brute

Dans cette partie, nous présentons les évolutions de l'ouverture de l'iris dans la figure 1, de la profondeur de l'œil dans la figure 2, du rapport entre le rayon de la lentille et la distance focale dans la figure 3 et de l'angle de vu dans la figure 4.
Pour chacun de ces paramètres, on peut voir grâce aux graphiques qu'ils atteignent tous un palier au bout d'un certain temps d'évolution. Pour ces quatre paramètres, on voit également que ce palier est atteint très rapidement, relativement au nombre de générations.
Une fois le palier atteint pour chaque paramètres, l'œil reste stable pendant le reste de l'évolution.

[2] `https://d3js.org`

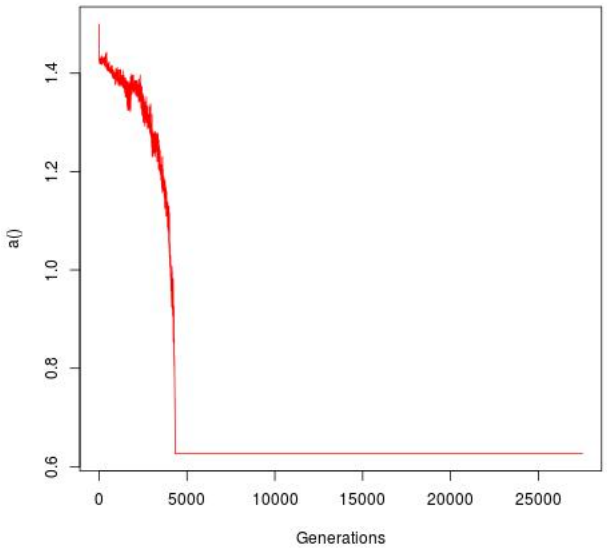

FIGURE 1 : Évolution de a

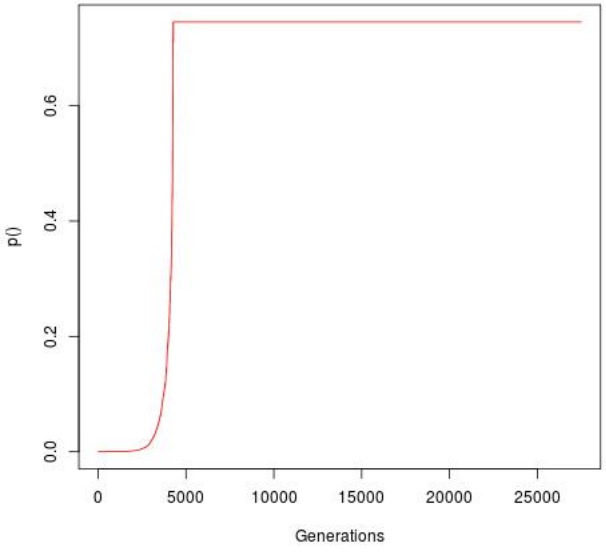

FIGURE 2 : Évolution de p

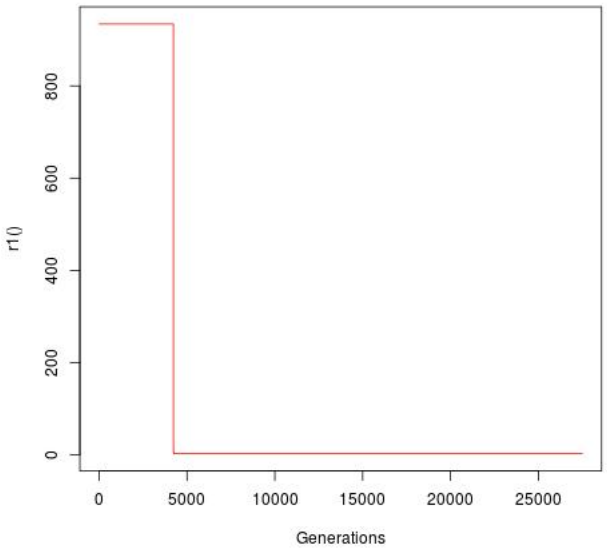

FIGURE 3 : Évolution de r1

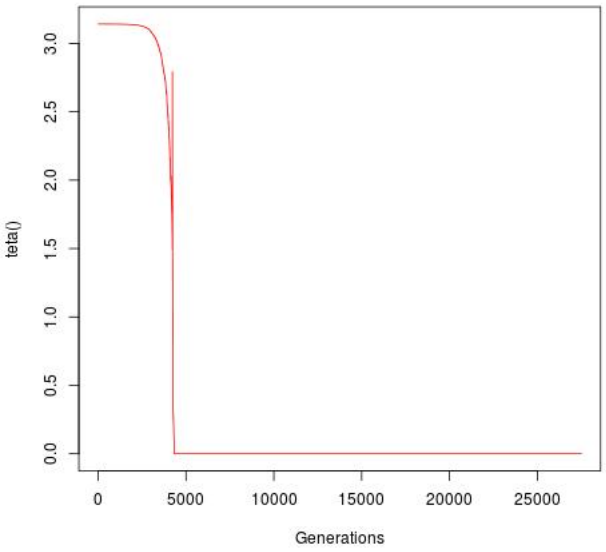

FIGURE 4 : Évolution de θ

3.3 Comparaison avec les résultats de Nilsson and Pelger [1994]

Dans Nilsson and Pelger [1994] l'évolution du modèle d'œil est guidée dans la mesure où les paramètres sont choisit judicieusement dans un ordre précis pour permettre à l'œil d'évoluer vers une configuration voulue. Notre approche est différente, tous les paramètres peuvent varier tout au long de l'évolution, seule la fonction de *fitness* dicte le chemin de l'évolution.

Notre visualisation nous permet de comparer nos résultats avec la séquence d'évolution de l'œil présenté dans Nilsson and Pelger [1994] et en particulier la dernière étape (figure 5). Visuellement nos résultats convergent vers un œil qui a une configuration similaire à l'étape finale de Nilsson and Pelger [1994].

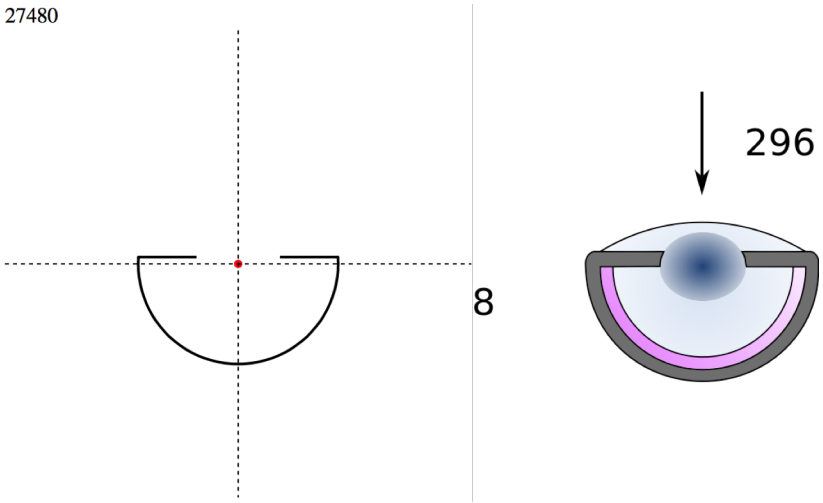

FIGURE 5 : Comparaison des résultats : à gauche notre modèle à droite celui de Nilsson and Pelger [1994]

4 Conclusion

Comme nous l'avons montrez notre approche à base d'un algorithme génétique et le modèle d'œil avec un nombre restreint de paramètres permet de converger vers la formation d'un œil avec lentille sans contrôler l'évolution comme c'est le cas dans Nilsson and Pelger [1994].

Bibliographie

Carole Knibbe. TP sur les algorithmes génétiques, 2017. URL `http://liris.cnrs.fr/carole.knibbe/ibi/tp/tp-IBI-AG-evolution-oeil.pdf`.

Dan-E Nilsson and Susanne Pelger. A pessimistic estimate of the time required for an eye to evolve. *Proceedings of the Royal Society of London B : Biological Sciences*, 256(1345) :53–58, 1994.

Eric J. Warrant and Peter D. McIntyre. Arthropod eye design and the physical limits to spatial resolving power. *Progress in Neurobiology*, 40(4) :413 – 461, 1993. ISSN 0301-0082. doi : http://dx.doi.org/10.1016/0301-0082(93)90017-M.

5 Annexes

5.1 Lien vers le code

Voici un lien vers le dépôt GitHub du projet :
`https://github.com/sheepolata/algogenibi`

5.2 Lien vers la visualisation

Voici un lien vers la visualisation en ligne de l'évolution de l'œil :
`https://sheepolata.github.io/algogenibi/`

5.3 Liste des algorithmes

Liste des Algorithmes

5.4 Liste des figures

Table des figures