Algorithme génétique et importance de la diffusion du savoir

Noé Bourgeois

***Résumé*—Nous avons implémenté un simulateur numérique de locomotion de cr éature virtuelle. Ce simulateur est composé d'un environement, une créature s'y déplacant en fonction de son génome , et un algorithme génétique permettant de modifier ce génome pour am éliorer le comportement de la créature . Nous avons ensuite utilisé ce simulateur pour étudier l'impact de différents paramètres sur la vitesse d'évolution de populations de créatures. Nous concluons que la reconnaissance d'un savoir hors-norme acquis par un individu et une diffusion de ce savoir au sein de sa population de manière à assurer sa conservation est le facteur le plus important pour l'évolution de cette population. Nous parvenons à satisfaire cette condition en conservant un petit nombre de tels d'individus intacts à chaque génération et en favorisant leur reproduction avec le reste de la population.**

I. Introduction

VIE et évolution sont deux concepts intimement liés. Cette force étrange et fascinante s'est répandue jusqu'ici via différentes méthodes de reproduction bien connues, le croisement génétique étant la plus aboutie et la plus adaptée à un monde aussi diversifié que la Terre. Le génome d'un individu est ainsi la matérialisation de l'information vitale transmise par ses ancêtres jusqu'à lui. Cet individu n'a, au départ, aucun autre moyen de transmettre de cette information que la reproduction. L'être vivant développe alors des sens et communique en pr ésentiel. D'importantes adaptations survivalistes intra-gén érationnelles peuvent alors être effectuées. Une information dont un individu pouvant la communiquer de son vivant n'existe pas est perdue.

L'humanité se caractérise par sa capacité à transmettre une information indépendamment de ces deux canaux. Depuis Les murs des grottes jusqu'aux rayons cosmiques, la portée dans le temps et l'espace de la transmission volontaire d'information humaine est sans commune mesure avec celle de l'information biologique.

Elle a cependant peu de chance de s'étendre au delà des limites du système solaire.

L'humanité devient alors la génitrice d'une nouvelle forme de vie , sa descendante héritant de tout son savoir, la vie artificielle.

Le concept de machine auto-réplicante a été introduit par John von Neumann en 1940 [1]. Il s'agit d'un modèle mathématique d'un système dynamique discret qui évolue en fonction d'un ensemble d'états et d'un ensemble de règles de transition, ce qui est une définition de la vie.

Superviseur : Mathieu Defrance

A. Pourquoi un simulateur de marche ?

1) La marche: Comme tout nouveau né, elle doit apprendre, notamment, à se déplacer.

le minage extraterrestre à grande échelle via vaisseau spatial auto-réplicants Von Neumman, conscient des ressources minières terrestres limitées, a présenté comme solution optimale. Cette problématique peut sembler lointaine dans le temps ou locale, mais la demande en ressources naturelles ne cesse d'augmenter et l'offre que présentent certains corps célestes represente le plus grand potentiel économique du marché des métaux précieux et rares de l'histoire de l'humanité.

Dans le cas de corps celestes trop massifs et dont l'atmosphère est trop peu dense, une machine de minage chargée de minerai se déplaçant sur une surface irrégulière dont l'aplanissement n'est pas rentable ou même envisageable, la marche est la méthode de déplacement la plus efficace ou même la seule possible.

Les jeux de plateaux sont un domaine où l'IA est déjà très présente. Les jeux de stratégie en temps réel ont été un des premiers domaines où l'IA a été utilisée. Le jeu de Go, bien que plus ancien, a été un des derniers jeux à damier à être résolu par une IA. La marche est un des moyens de déplacement les plus utilisés dans les jeux de plateaux. Les pions matériels, jusqu'ici déplacés à la main, pourraient être remplacés par des robots semi-autonomes.

L'industrie du divertissement, en particulier celle du jeu vidéo, utilise déjà des simulateurs de marche pour créer des personnages plus réalistes de manière procédurale. La Science-Fiction est pleine de robot intelligents pouvant marcher ayant différentes fonctions. La proposition probablement la plus proche de nous dans le temps est celle présentée dans *"Westworld"*, où des parcs d'attraction peuplés de robots humanoïdes sont proposés aux touristes. Ces simulateurs grandeur nature pourraient être utilisés pour rentabiliser l'investissement dans la recherche et grandement accélérer son développement par l'introduction dans un envirronement contrôlé de pertubations similaires à celles que les robots pourront rencontrer dans la réalité.

Boston Dynamics & Tesla ayant déjà développé des robots à l'équilibre impressionnant, nous pourrions nous demander si un quelconque progrès significatif reste à réaliser dans ce domaine. Boston Dynamics dont les algorithmes de marche sont les plus avancés au monde était bien en avance sur son temps et a pourtant encore beaucoup de difficultés par exemple, à rendre le mouvement de ses robots bipèdes fluides en terrain complexe inconnu ce qui

rend la démarche assez peu naturelle car non optimale. Le domaine de recherche est en fait en plein essor.

2) Le simulateur: De tels tests en conditions réelles dans lesquelles la chute d'un robot peut être fatales sont trop co ûteux pour être effectués sur des prototypes. Ce prototypage est donc réalisé dans un environnement virtuel où les conditions de test peuvent être répétées à l'infini en omettant les aspects déjà maîtrisés pour concentrer les ressources sur les problèmes critiques.

B. Pourquoi un algorithme génétique ?

John Holland et son équipe, en 1975, introduisent l'algorithme génétique [3] comme une interprétation mathématique directe de la théorie de l'évolution de Darwin. Il s'agit d'un algorithme stochastique itératif, qui, par échantillonage de population, progresse par génération vers un optimum global d'une fonction objectif. Celui-ci est très puissant pour résoudre des problèmes d'optimisation combinatoire complexe. Sa nature stochastique lui permet de trouver des solutions souvent inattendues. Il est donc particulièrement adapté à la phase de recherche d'un projet.

II. Etat de l'art

1) Décomposition - Recomposition: La plus grande difficulté de l'optimisation multi-objectif est l'équilibrage des objectifs. MOEA/D [5] est un algorithme génétique qui divise la population en sous-populations, chacune résolvant un sous-problème de mani ère indépendante, puis combine les solutions pour obtenir un ensemble optimal de solutions.

2) Autres domaines: Le goulot d'étranglement de l'algorithme génétique est la somme des opérations sur le génome des individus, leur croisement et mutation. Dans un autre registre que le nôtre, la physique quantique ouvre le champ d'action en permettant la superposition et l'intrication d'états. Ces propriétés pourraient être utilisées pour simuler l'équivalent de multiples créatures en "parallèle" et ainsi accél érer la convergence de l'algorithme génétique. [12]

III. Méthodologie

A. Les hypothèses de base de notre approche

1) Marche: Tout d'abord, il est important de rappeler que la marche est un mouvement cyclique, c'est à dire qu'il se répète à intervalles réguliers. Il est donc possible de décrire un cycle de marche comme une séquence d'états, chacun étant une description de la position et de l'orientation du robot à un instant donné. Un pas est une chute contrôlée en avant, suivie d'une récupération de l'équilibre par la pose d'un membre au sol. La présence de plus de deux membres semble donc importante en début et fin de marche seulement. Elle n'est jamais indispensable si l'on se réfère à bon nombre d'animaux bipèdes et pourtant très stables et rapides. Une contrainte importante du projet étant les puissances de calcul de nos ordinateurs, et les nôtres nous permettant d'estimer que modifier des forces sur 2 membres sera 2 fois plus rapide que sur 4, la marche bipède semble être un bon compromis. Nous réaliserons par la suite que le mouvement des membres de nos cr éatures étant contraints sur le seul axe de la marche, la gestion de l'équilibre sur 2 pattes s'avère trop compliquée. Nous ne considérerons donc ici que des créatures quadrupèdes.

2) Heuristique: La composante de base la plus évidente pour l'agent est de se déplacer vite, nous récompensons donc la distance parcourue divisée par le temps mis pour la parcourir. Cette composante, la plus importante n'est cependant valable que si l'agent ne tombe pas. Une composante à caractère invalidateur sur une chute semble donc indispensable

3) Algorithme génétique: Le but est de conserver une diversité la plus large possible tout en convergeant vers une solution optimale. La diversité est assurée par la mutation aléatoire et la convergence par la sélection.

B. Les fondements mathématiques

1) Algorithme génétique:

Algorithm 1 Genetic Algorithm

Require: Population size N
, Mutation rate p_m
, Crossover rate p_c
, Maximum number of generations $G_{\max}$
Ensure: Optimal solution
Initialize population P with N individuals
$g \leftarrow 0$
while $g < G_{\max}$ **do**
  Evaluate fitness of each individual in P
  Select parents for reproduction
  Create empty offspring population Q
  while $|Q| < N$ **do**
    $p_1, p_2 \leftarrow \text{SelectParents}(P)$
    $o_1, o_2 \leftarrow \text{Crossover}(p_1, p_2, p_c)$
    $o_1 \leftarrow \text{Mutate}(o_1, p_m)$
    $o_2 \leftarrow \text{Mutate}(o_2, p_m)$
    Add o_1 and o_2 to Q
  end while
  $P \leftarrow Q$
  $g \leftarrow g + 1$
end while
return Best individual in P

2) Optima locaux: Dans une population, si certains individus affichent une fitness plus élevée que les autres, ils sont dits *dominants*. Si les dominés convergent vers l'état des dominants à une vitesse telle que l'information génétique des dominés est perdue, l'exploration de l'espace des solutions est limitée à l'espace de celle des dominants. Si cet espace n'est pas un optimum global, l'exploration doit continuer mais Dans notre cas, la fonction objectif est une maximisation. Les minima locaux ne sont donc pas un

problème. Les maxima locaux, en revanche, peuvent être problématiques.

C. La méthode proposée

La méthode que nous présentons ici a été développée sur base de choix d'implémentation grandement influencés par le contexte du projet. Celui-ci était séparé en 2 parties :

— 1.Recherche & Développement
— 2.Présentation d'une version vulgarisée du projet au Printemps des Sciences

Cette deuxième partie rendait primordiale l'obtention d'un résultat fonctionnel et visuellement attrayant. Nous avons donc décidé d'utiliser la librairie PyGAD [11] pour nous libérer de l'écriture de l'algorithme lui-même et nous concentrer sur son adaptation à notre problematique et l'optimisations des paramètres.

1) Paramètres: **initial population** : C'est la population initiale qui peut être fournie à l'algorithme génétique.

population size : C'est la taille de la population, c'est-à-dire le nombre d'individus dans une génération.

num generations : C'est le nombre de générations sur lesquelles l'algorithme génétique va s'exécuter.

num parents mating : C'est le nombre de parents sélectionnés pour la reproduction à chaque génération.

parent selection type : C'est le type de méthode utilisée pour sélectionner les parents qui se reproduiront. Dans notre cas, la méthode de sélection par tournoi est utilisée car elle fonctionne avec des valeurs de fitness négatives. . La sélection par tournoi fonctionne en sélectionnant aléatoirement K (taille du tournoi) individus, puis en choisissant le plus apte parmi eux (le gagnant) pour la reproduction.

La pression de sélection dépend de K : plus K est grand, plus la pression est forte, car les individus les plus faibles auront plus d'adversaires et auront donc plus de chances de perdre.

K_tournament : contrôle la taille du tournoi.

keep elitism : C'est le nombre d'individus dont les solutions sont les meilleures qui sont conservés sans mutation dans la gén ération suivante. C'est utile dans la situation où la fitness d'un individu est hautement supérieure à celle des autres individus. Altérer son génome avec un autre, potentiellement le pire, puis muter aléatoirement ce résultat signifierait la perte de cette progression fulgurante. Garder intact une grande quantité de génomes signifie également que la population ne se renouvelle pas assez, cette quantité doit donc rester limitée.

crossover type : C'est le type de croisement utilisé pour la reproduction des parents. Le croisement à un point choisit un point de croisement aléatoire dans le génome des parents et échange les parties du génome après le point.

mutation type : C'est le type de mutation appliqué aux gènes individuels. La mutation aléatoire remplace la valeur d'un gène choisi aléatoirement par une valeur aléatoire dans l'espace des gènes. La mutation adaptative est une mutation aléatoire dont la valeur est déterminée par la fitness de l'individu. Plus la fitness est élevée, plus la valeur de la mutation est faible. Cela permet d'éviter que les individus ne s'éloignent trop de la solution optimale.

mutation percent genes : Il s'agit du pourcentage de gènes sujets à la mutation. Peut être un tuple, la première valeur est alors le pourcentage chez les dominés et la seconde chez les dominants.

D. Les instructions nécessaires pour pouvoir reproduire les expériences (par exemple pseudo-code)

Notre simulateur est disponible à l'adresse suivante : https://github.com/nobourge/INFO-F308—Projets-d-informatique-3-transdisciplinaire—202223.

1) Fitness Function: Une créature est évaluée sur 5 critères :

— La distance entre le tronc et le sol alive_bonus
  — <0 Si le tronc touche le sol
  — >0 Sinon
— Les forces exercées : limite des mouvements trop violents pouvant provoquer la chute de la créature.
— La vitesse : distance parcourue divisée par le temps
— La hauteur maximale atteinte : maintient de la verticalité pour éviter une posture menant à la chute

$$\text{Fitness} = \text{alive\_bonus} + (H_i - H_{i-1}) * 0.1 + \frac{D_i - D_{i-1}}{time} + \text{forces} \quad (1)$$

Ce calcul est effectué sur chaque individu à chaque génération de la simulation .

IV. Résultats

Pour réaliser nos tests, nous avons réduit l'aléatoire au maximum en assignant une population initiale commune PI à toutes les simulations, d'elle découle population_size

A. Paramètrage minimal

Pour évaluer notre paramétrage, nous le comparons tout d'abord à un paramétrage minimal, c'est-à-dire avec les paramètres par défaut de PyGAD augmentés du minimum requis.

Table I
Paramètres par défaut de PyGAD

Parameter	Value
population size	50
num parents mating	4
parent selection type	random
crossover type	single point
mutation type	random
mutation percent genes	10
random mutation min val	-1
random mutation max val	1

Les résultats obtenus sont présentés dans la figure1 et la figure2. Nous remarquons que les solutions présentant une avancée majeure ne semblent pas tirer les autres vers le haut. Le nombre d'individus avec lesquels se reproduisent les dominants est trop faible. Leur génome est ensuite muté au même taux que celui des dominés.

TABLE II
PARAMÈTRAGE MINIMUM REQUIS

Parameter	Value
num genes	8 joints * 500 time steps = 4000
initial population	PI
num parents mating	(population_size // 4) + 2
fitness function	Equation1
num generations	100

B. *Importance de la diffusion du savoir*

Nous ajoutons le parametrage suivant (tableIII) :

TABLE III
PARAMÈTRAGE POUR LA DIFFUSION DU SAVOIR

Parameter	Value
num parents mating	(population size // 2) + 2
parent selection type	tournament
K tournament	population size // 2 + 2
keep elitism	2
mutation type	adaptive
mutation percent genes	(30, 10)

Nous obtenons les résultats suivants (figure3 et figure4) :

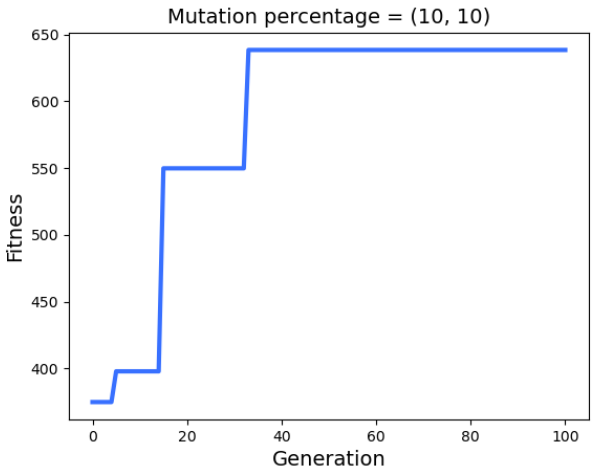

FIGURE 1. fitness par génération pour un paramétrage minimal

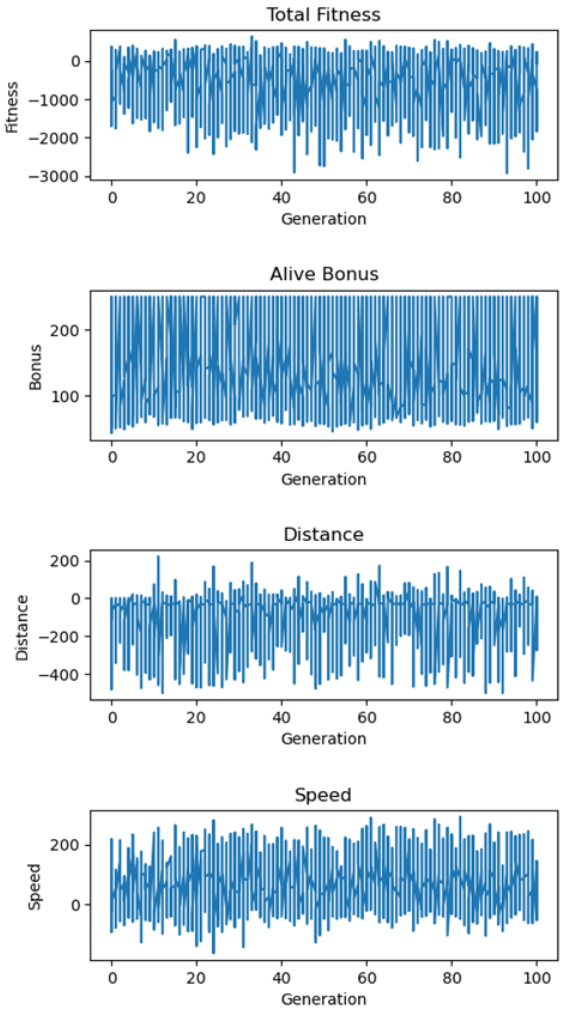

FIGURE 2. plages de valeurs par génération pour un paramétrage minimal

V. DISCUSSION

Nous pouvons observer une nette amélioration de la qualité des solutions grâce à notre paramétrage. La progression faite à la génération 40 est immédiatemment répandue à l'ensemble de la population. Cependant, la borne inférieure de la fitness générale correspondante n'évolue pas en conséquence. Cela traduit une sous-estimation de la distance parcourue par rapport à la vitesse. Les bonnes solutions étant indiscernables des mauvaises par leur fitness, la progression s'annule à la génération 70 et s'inverse par la suite.

Nous avons également pu constater qu'il est très difficile de trouver les bons paramètres pour que l'algorithme converge vers une solution optimale. Même sur une population initiale commune et un paramétrage identique , la qualité des solutions trouvées est très variable. Nous pouvons cependant affirmer certains faits :

La mutation doit opérer prioritairement sur les dominés mais ne doit pas épargner les dominants ou un maximum local est atteint. Des valeurs de mutation aléatoires trop grandes reviennent à ignorer les parents et par là même, l'évolution par transmission des gènes. Celles-ci doivent donc être comprises dans un intervalle autour de zéro très restreint.

Si l'élitisme est trop important ou la compétition est trop sélective, les solutions ne sont pas assez diversifiées. Si la mutation est trop forte, les solutions convergent trop rapidement. Lorsqu'une ou plusieurs de ces conditions sont réunies, l'évolution atteint un maximum local et la progression est fortement ralentie. Ce phénomène se traduit visuellement par les plateaux observés.

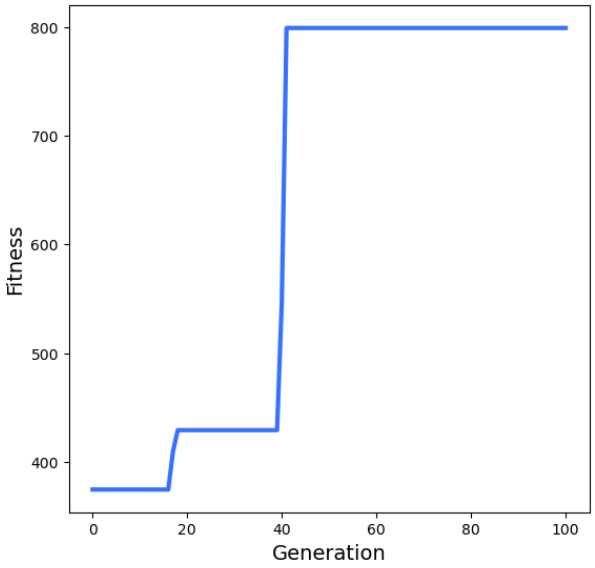

FIGURE 3. fitness par génération pour notre paramétrage

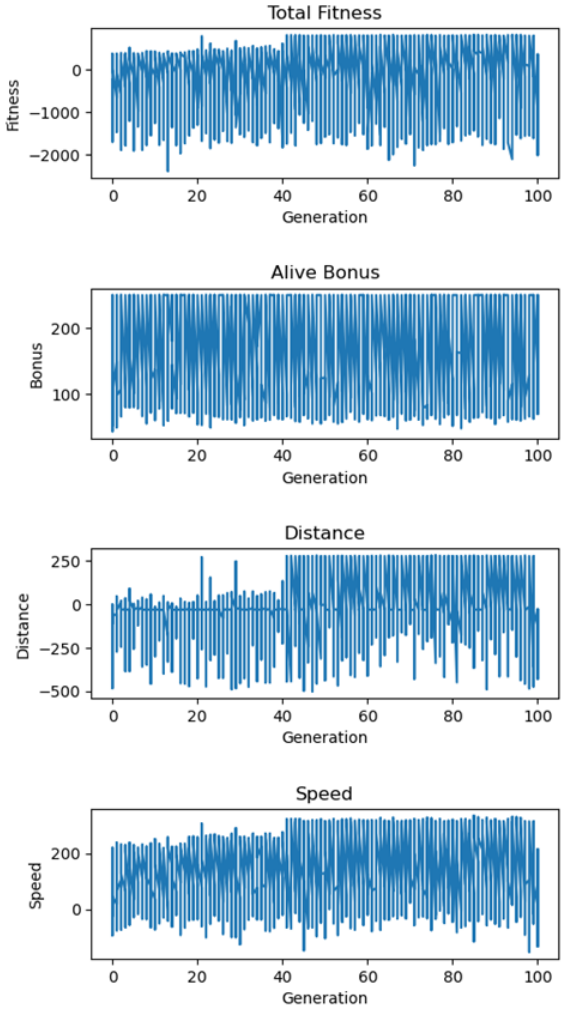

FIGURE 4. plages de valeurs par génération pour notre paramétrage

VI. CONCLUSION

Notre simulateur permet de trouver en temps raisonnable des solutions permettant à des créatures de se maintenir debout sans chuter et se déplacer le temps de la simulation.

De tout système, ce que nous attribuons au hasard est sa partie incomprise. Dans notre simulateur, un système quasi-totalement compris, l'aspect stochastique de l'algorithme génétique est donc fondamentalement sous-optimal. Cependant, nous le contrôlons très peu, et même si nous en avions la complète maîtrise, la taille de l'algorithme déterministe optimal serait (beaucoup) plus grande que celle de l'algorithme génétique.

L'algorithme génétique nous semble donc être le parfait compromis entre les objectifs suivants :

- algorithme compact
- adaptabilité au type d'information disponible
- robustesse
- exhaustivité des solutions (fonction du temps d'execution)

Les valeurs d'entrée ont un impact minime sur la qualité des solutions trouvées.

A. Limites

Les limites de notre simulateur sont les suivantes :

- Les paramètres par défaut de PyGAD ont à peine été modifiés. La vision globale du traitement de la problématique offerte par l'espace de notre parametrisation est très étroite.
- La majeure partie du temps d'execution du programme est consacrée à la simulation physique dont nous ne valorisons que très peu les informations que nous en retirons.
- Il est difficile de déterminer si la fonction de fitness est trop sévère par rapport au principe stochastique de l'algorithme ou trop laxiste vis-à-vis de la simulation physique.

Les forces à appliquer sur les articulations étant inscrites dans le génome, et la créature étant dépourvue de sens, aucune adaptation intra-générationelle n'est possible. La moindre irrégularité introduite dans l'envirronement pourrait faire chuter la créature.

B. Perspectives de recherche future

1) Paramètres: L'étendue des paramètres de PyGAD est très large et au vu de nos résultats obtenus après les tests présentés, une solution optimale est probablement accessible en temps raisonnable via la bonne combinaison de paramètres. Pour la trouver, l'idée serait de nous épargner une batterie de tests interminable en utilisant du scrapping pour récupérer les paramétrisations de chaque projet PyGAD similaire au notre et comparer leurs résultats afin de trouver les meilleurs paramètres puis d'itérer sur cette base pour affiner les paramètres dont les valeurs sont des intervalles. Il s'agirait pour ainsi dire d'un meta algorithme génétique appliqué sur la population des param ètres.

Eviter les maximum locaux en remplaçant les paramètres d'élitisme et de compétition par des intervalles contrôlés par une fonction réagissant à la stagnation de la population.

2) Bibliothèque: L'information de marche n'est actuellement portée que par la population vivante à un instant donné. L'intérêt d'une population vivante est qu'elle expérimente. L'intérêt d'une bibliothèque est qu'elle conserve l'information. Il serait intéressant de permettre à la population de se renseigner sur les solutions notables trouvées par des individus qui lui sont antérieurs de plus d'une génération. Cette bibliothèque serait alimentée par les solutions que nous sauvegardons déja pour la post-visualisation ou l'assignation du paramètre `initial_population`. Elle remplirait le rôle déjà joué par la population élite, mais ne présenterait pas les défauts de la mutation à taux réduit et de la consanguinité. Contrairement à la population élite qui partage son information à des individus en quantité limitée, et l'impose, la bibliothèque, puisqu'accessible à tous les individus, permettrait une diffusion du savoir bien plus large et harmonieuse chacun libre d'en assimiler ce qui lui convient.

Un dictionnaire de génomes à éviter, tels qu'une suite de zéros entraînant l'absence d'action, et/ou assurément trop loin d'une solution viable par rapport au nombre de générations restantes.

3) Fonction de fitness: La force de l'algorithme génétique étant sa souplesse, il est possible que des conditions plus laxistes permettent d'éviter de rester coincé dans un maximum local. L'objectif de hauteur du centre de masse est actuellement un maximum en un point et peut entraîner un dés équilibre limitant les possibilités de mouvement. Cet objectif pourrait être remplacé par un intervalle de hauteur global allouant plus de liberté à la créature.

La fonction de fitness pourrait elle-même être modifiée automatiquement en fonction des comportements observés en liant les poids de chaque composante aux actions pour lesquelles elles entrent en jeu.

4) Adaptation intra-générationelle: **Réseau neuronal** Ces perspectives de développement de la fonction de fitness gén érationnelle mènent rapidement aux prémices d'un réseau neuronal. Lié à des capteurs, il permettrait à la créature de réagir à son environnement et de se déplacer de manière autonome. Il s'agirait dans notre cas d'une forme d'**épigenetique**

Références

[1] , John von Neumann *Theory of Self-Reproducing Automata*, 1966.

[2] , Olivia Borgue and Andreas M. Hein *Near-Term Self-replicating Probes - A Concept Design*, 2005.

[3] , John Holland *Adaptation in Natural and artificial Systems*, 1975.

[4] , Gabriel Cormier *Systèmes Intelligents*, Université de Moncton, 2019.

[5] , Qingfu Zhang and Hui Li *MOEA/D : A Multiobjective Evolutionary Algorithm Based on Decomposition*, 2008.

[6] , Karl Sims *Evolving Virtual Creatures*, 1994.

[7] , Graham, Lee ; Oppacher, Franz. *Speciation Through Selection and Drift.* Proceedings of The Eleventh IASTED International Conference on Artificial Intelligence and Soft Computing. ACTA Press.

[8] , Josh C Bongard *The legion system : A novel approach to evolving heterogeneity for collective problem solving.*

[9] , J. Bongard et H. Lipson *"Simulation de la locomotion par algorithme génétique".*

[10] , By KSmrq - http://commons.wikimedia.org/wiki/File:Extrema_example.svg,GFDL1.2,https://commons.wikimedia.org/w/index.php?curid=6870865

[11] https://pygad.readthedocs.io/en/latest/

[12] , Bart Rylander, Terence Soule, James Foster and Jim Alves-Foss *Quantum Genetic Algorithms*, 2000