Algorithme génétique et importance des données d'entrée

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 cr éature. Une population de départ générée aléatoirement ayant évolué au long d'une simulation, nous conservons ses individus et les replaçons comme population de départ dans la simulation suivante.**

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].

La nécessité ultime de la vie artificielle ayant été établie, il

replicants champs des possibles limité par ce que nous savons possible ne pas reinventer la roue nature pas de roue genetique amelioree

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.

La médecine pourrait bénéficier chaise roulante prothèse

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

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 pattes sera 2 fois plus rapide que sur 4, la marche bipède semble être un bon compromis.

2) Paramètres:

3) 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

4) 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 sélection des parents et la convergence par la sélection des enfants. La sélection des parents est réalisée par un tournoi entre individus choisis aléatoirement dans la population.

Dans un algorithme génétique classique, l'exécution est divisée en générations - à chaque génération, le résultat du processus de sélection et de reproduction remplace l'ensemble (ou du moins la plupart) de la population existante ; seuls les enfants survivent. Cependant, dans un algorithme génétique à état stable (steady state), seuls quelques individus sont remplacés à chaque fois, ce qui signifie que la plupart des individus sont conservés pour la génération suivante ; il n'y a pas de notion de génération à proprement parler.

B. Les fondements mathématiques

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

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 3 parties :

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

Cette troisième partie rendait primordiale l'obtention d'un résultat fonctionnel et visuellement attrayant. Nous avons donc décidé d'utiliser la librairie PyGAD [9] 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: **gene space** : Il s'agit de l'espace des gènes où les valeurs des gènes individuels de la population doivent se situer. Dans ce cas, l'espace des gènes va de -1 à 1 avec un pas de 0.1.

init range low : C'est la valeur minimale autorisée pour l'initialisation des gènes individuels dans la population.

init range high : C'est la valeur maximale autorisée pour l'initialisation des gènes individuels dans la population.

random mutation min val : C'est la valeur minimale autorisée pour la mutation aléatoire d'un gène individuel.

random mutation max val : C'est la valeur maximale autorisée pour la mutation aléatoire d'un gène individuel.

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.

parallel processing : Indique si le traitement parallèle doit être utilisé.

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.

La taille du tournoi est contrôlée par le paramètre K_tournament.

keep elitism : C'est le nombre d'individus élites qui sont conservés sans mutation dans la gén ération suivante. C'est utile dans la situation où la fitness d'un individu est largement 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. Dans notre cas, 2 individus élites sont conservés afin de laisser la possibilité qu'ils se reproduisent entre eux.

crossover type : C'est le type de croisement utilisé pour la reproduction des parents. Dans ce cas, le croisement uniforme est utilisé.

mutation type : C'est le type de mutation appliqué aux gènes individuels. Dans ce cas, la mutation adaptative est utilisée.

mutation percent genes : Il s'agit du pourcentage de gènes sujets à la mutation. Dans ce cas, 60

2) 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

$$\text{Fitness} = \text{alive\_bonus} + (\text{H}_{i-1} \text{ - } H_i) * 0 .1 + \frac{D_i - D_{i-1}}{time} + \text{ forces } (1)$$

Ce calcul est effectué sur chaque individu de chaque gén ération de la simulation .

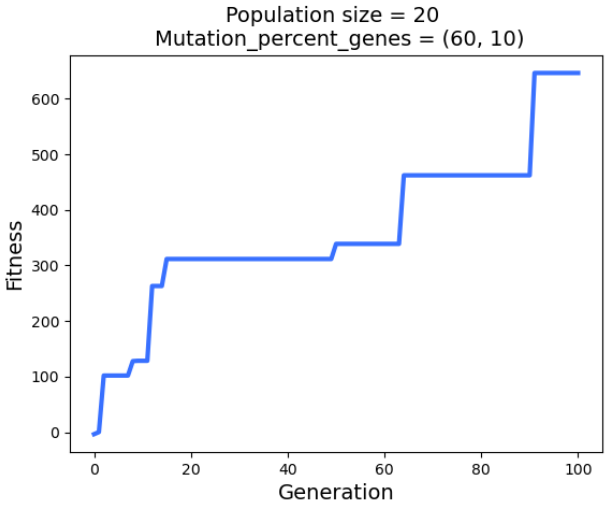

Figure 1. Example Image

D. Les jeux de données utilisés

E. 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

IV. Résultats

Table I
Paramètre par défaut des simulations

Parameter	Value
gene space	low : -1, high : 1, step : 0.1
init range low	-1
init range high	1
random mutation min val	-1
random mutation max val	1
initial population	None
population size	100
num generations	100
num parents mating	4
parallel processing	None
parent selection type	tournament
keep elitism	5
crossover type	uniform
mutation type	adaptive
mutation percent genes	(60, 10)

Si la population est trop petite, le nombre de solutions différentes est faible. Si l'élitisme est trop important ou la compétition est trop sélective, les solutions ne sont pas assez diversifiées. Si la mutation est trop faible, 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

V. Conclusion

Notre simulateur permet de trouver en temps raisonnable des solutions permettant à des créatures de se

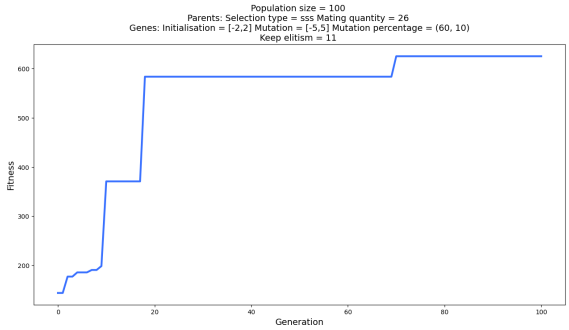

FIGURE 2. Example Image

maintenir debout sans chuter et se déplacer le temps de la simulation. Cependant, la qualité des solutions trouvées est très variable et la posture des créatures est toujours sous-optimale. Nous avons pu constater qu'il est très difficile de trouver les bons paramètres pour que l'algorithme converge vers une solution optimale. De plus, il est très difficile de trouver une fonction de fitness qui permette de bien évaluer les solutions.

L'algorithme génétique nous semble donc adapté à la pose de larges fondations.

Des valeurs de mutation aléatoires trop grandes reviennent à ignorer les parents et à générer une population aléatoire. Celles-ci doivent donc être comprises dans un intervalle très restreint.

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 très difficile de définir si la fonction de fitness est trop sévère par rapport au principe stochastique de l'algorithme génétique 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. L'envirronement physique est le plus simple possible : un sol plat. La moindre irrégularité introduite pourrait faire chuter la cr éature.

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.

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

2) Fonction de fitness: **correction** d'un mauvais comportement, en plus de l'attribution de score, nous pourrions implémenter une fonction de correction pour identifier la région du génome commandant une action non désirée et la remplacer par des valeurs plus adaptées. 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.

3) 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 **épigenetique**

4) Environnement: Dès lors que la créature est capable de réagir à son environnement, celui-ci peut être modifié pour augmenter la difficulté. Parmi les modifications les plus intéressantes : **Générales** :

- Des pentes
- Des obstacles
- Des projectiles
- Des créatures adverses

A finalité réelle : Se concentrer sur les envirronements encore mal maîtrisés par les robots actuels :

- Des surfaces à solidité variable (boue, etc.)
- Des surfaces glissantes (verglas, etc.)
- Des surfaces instables (sable, etc.)

A finalité virtuelle :

- Environnement de jeux vidéo déjà existant
- Environnement de jeux vidéo généré aléatoirement

5) Parallélisation:

6) 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. [10]

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

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

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

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

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

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

Annexe A
Consignes

Document

Le rapport doit être rédigé en LaTeX en utilisant ce template. La longueur du rapport ne devra pas, en tout cas, dépasser les 6 pages. Ce rapport doit être *self-contained*, c-à-d il doit pouvoir être lu et compris sans avoir besoin de se documenter ailleurs.