Sorbonne Université
Master ANDROIDE

Apprentissage social et imitation pour la robotique en essaim

UE de projet M1

Alessia Loi – Dorian LANCELIN

2022

Table des matières

Chapitre 1

Introduction

L'objectif de ce projet est de trouver des outils permettant à des robots avec une mémoire très limitée d'apprendre à réaliser une tâche par l'observation des mouvements d'autres robots ayant les mêmes spécifications. Dans cette optique, on considèrera que les robots ont des moyens de communication limités voire inexistants si possible. En utilisant des agents au comportement prédéfini, nommés ici les experts, capables de réaliser au mieux la tâche envisagée et des robots apprenants, on pourra toujours observer une convergeance vers le comportement de l'expert dans les cas où l'apprentissage se fait uniquement dans le sens expert-apprenant. Dans certains cas particuliers, des méthodes explorées permettent à l'apprentissage apprenant-apprenant d'accélerer cette convergeance.

Ce projet a été encadré par messieurs Nicolas Bredeche et Stéphane Doncieux.

L'ensemble des codes utilisés afin d'obtenir les résultats décrits dans le document suivant sont disponibles sur notre dépôt github.

Chapitre 2

État de l'art

Aujourd'hui, la plupart des algorithmes d'apprentissage utilisés en robotique en essaim sont soit déterministes, soit axés sur l'apprentissage par transfert de poids de neurones, à l'instar de l'algorithme HIT-EE [1] qui nous a servi de base pour le projet.

HIT-EE est un algorithme d'apprentissage en ligne qui suit un principe simple : Chaque agent subit une phase d'évaluation durant laquelle il ne peut transmettre son patrimoine mais reçoit les messages d'autres agents. A l'issue de cette phase, pour chaque message reçu, Si sa fitness est inférieure à celle de l'émetteur, il copie une partie des poids de l'émetteur. Une fois tous les messages lus, son patrimoine peut muter avec une probabilité donnée. Enfin si son patrimoine a changé durant au moins l'une des deux phases précedentes, il recommence sa phase d'évaluation.

1. Nicolas Bredeche and Nicolas Fontbonne. 2022. Social learning in swarm robotics. Philos. Trans. R. Soc. B Biol. Sci. 377, 1843 (January 2022), 20200309. DOI :https ://doi.org/10.1098/rstb.2020.0309

Chapitre 3

Contribution

3.1 Choix de l'expérimentation : La tâche de fourragement

Dans le but d'évaluer la capacité d'apprentissage par comportement de notre essaim de robots, on a reproduit l'expérience de fourragement et l'avons simulée à l'aide de Roborobo. L'évaluation en elle-même se fait sur la quantité d'objets capturés par un agent sur une fenêtre de temps donnée (sliding window). On considèrera qu'il n'y a pas de perte d'informations lors du transfert, c'est-à-dire que l'apprenant est capable de voir tout ce que voit l'expert.

On définit ainsi les paramètres de l'expérience :

- Taille de la population : 100 agents
- Nombre d'experts : 10
- Nombre d'apprenants : 90
- Nombre d'objets : 100
- Taille de l'arène : 1400x800 px
- Taille des robots : 5x5 px
- Distance des senseurs : 16 ou 24 (8 senseurs, capable de distinguer le type de l'objet et la distance à ce dernier)
- Distance des senseurs : 16
- Durée d'une expérience : 20,000 steps
- Nombre de répétions de l'expérience : 5

Les différents experts suivent un comportement donné. Les différents experts suivent un comportement donné. L'un sous forme de génome, c'est-à-dire 100 couple observations-actions. Et le second dont l'objectif est d'éviter les obstacles (qu'ils soient des murs ou d'autres robots) tout en se focalisant sur le ramassage des objets dès que possible, obtenu par simple succession de conditions.

3.2 Méthode par transmission de comportements

On considère que chaque individu possède une base de données de traces comportamentales, utile à stocker les comportements qu'un individu expert lui a envoyé lorsque les deux robots se sont croisés.

La base de données de chaque individu expert est initialisée au début de l'expérience : elle contient 100 traces pré-définies et n'évolue pas au cours du temps. Parmi toutes les traces possibles, les 100 traces que nous avons sélectionnées permettent aux individus experts d'être efficaces dans le contournement d'obstacles que ce soit des murs ou des robots pour prévenir les blocages et la collecte d'objets lorsqu'un objet se trouve dans le périmètre perceptible des senseurs.

Une trace comportementale est un couple $(s_t, a_t)^T$, avec :

- s_t : l'état courant du contexte, qui correspond à une liste de 24 valeurs des senseurs étendus. L'état courant est de la forme $(s_1, s_2, s_3, ..., s_n)$ avec n le nombre de senseurs du robot.
- a_t : l'action à appliquer en fonction du contexte sensoriel rélévé. L'action associée est de la forme $[t, r]$ avec t le degré de transition et r le dégré de rotation.
- T : le temps de croisement des deux trajectoires, exprimé en nombre de steps.

La base de données de chaque individu focal évolue au cours du temps. Au départ, elle contient une seule trace correspondant à un comportement par défaut à appliquer lorsqu'aucun des senseurs ne perçoit d'élement de l'environnement. Chaque fois que les trajectoires d'un individu focal et d'un individu expert se croisent, l'expert envoie une partie de ses traces à l'apprenant, qui enrichit ainsi sa collection de traces.

L'ensemble de traces comportementales d'un individu focal constitue la base d'exemples (senseurs) et étiquettes (actions) qui sera utilisée pour l'entraînement et la prédiction des algorithmes d'apprentissage.

Protocole expérimental spécifique à cette partie :

A chaque début d'évaluation, les conditions initiales sont réinitialisées, notamment :

- la position de tous les individus
- la base de données de comportements
- la performance de chaque individu (exprimée en nombre d'objets collectés)

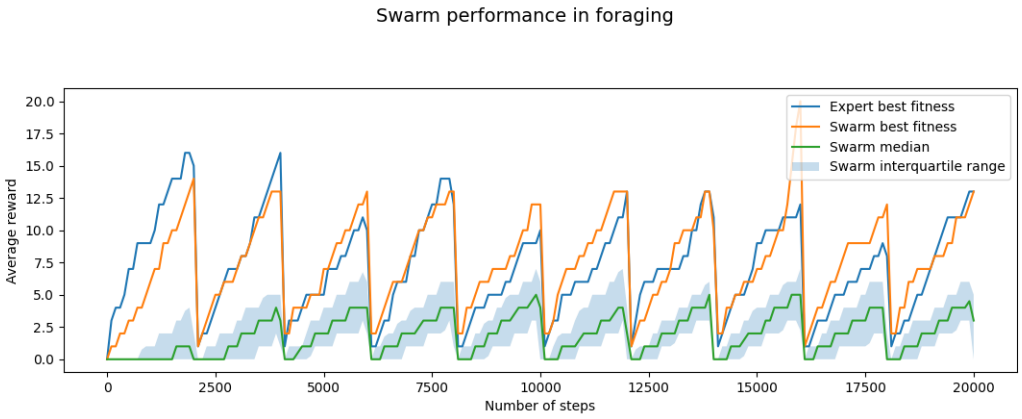
Remarque : Pour l'algorithme de rétropropagation du gradient, les poids du réseau de neurones ne seront pas réinitialisés d'une évaluation à l'autre, dans le but de raffiner les poids précédemment obtenus en les testant dans un nouveau contexte.

Les graphiques de distance sont obtenus ainsi :

- A chaque step, nous capturons l'univers aperçu par un individu expert, en enregistrant l'état qu'il détecte (liste de 24 senseurs) et l'action correspondante effectuée.
- A $t = 1000$ steps, nous comparons les individus focaux en faisant la somme de la distance euclidienne entre leur choix d'action pour chaque observation de l'expert, et l'action effectuée par l'expert pour l'ensemble des données enregistrées.
- Nous répétons la même technique à $t = 2500$ steps et à $t = 6000$ steps, ensuite nous comparons les résutats sur la tranche de $t = 0$ steps à $t = 2500$ steps.

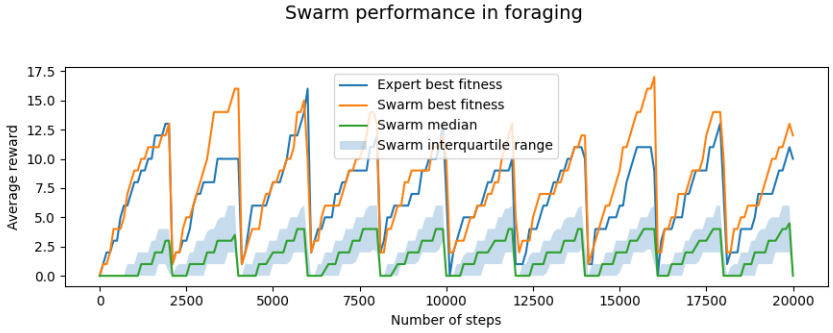
3.2.1 Apprentissage par rétropropagation du gradient

L'apprentissage par rétropropagation du gradient permet d'observer une évolution fluide du comportement des individus focaux : les trajectoires dessinent des ellipses qui deviennent progressivement plus larges et ouvertes au cours du temps. En presence d'obstacles, les individus répondent rapidement pour le contournement des murs et nous observons qu'il ne se crée pas d'interblocage avec les autres robots. De manière moins importante par rapport aux experts, nous remarquons aussi la reconnaissance d'objets.



EVALUATION DETAILS :

nbRobots : 100 nbExpertsRobots : 10 nbNotExpertsRobots : 90
nbFoodObjects : 100 maxSizeDictMyBehaviors : 100
hit_ee_transferRate : 0.1 hit_ee_maturationDelay : 0 hit_ee_learningOnlyFromExperts : True
swarmLearningMode : neuralNetworkBackpropagation
nb_hiddenLayers : 1 nb_neuronsPerHidden : 16 nb_neuronsPerOutputs : 2
defaultBehavior : [1,0.5] learningRate : 0.4 distanceEpsilon : 0.25 nbEpoch : 200
nbSteps : 20000 evaluationTime : 100 resetEvaluation : True resetEvaluationTime : 2000

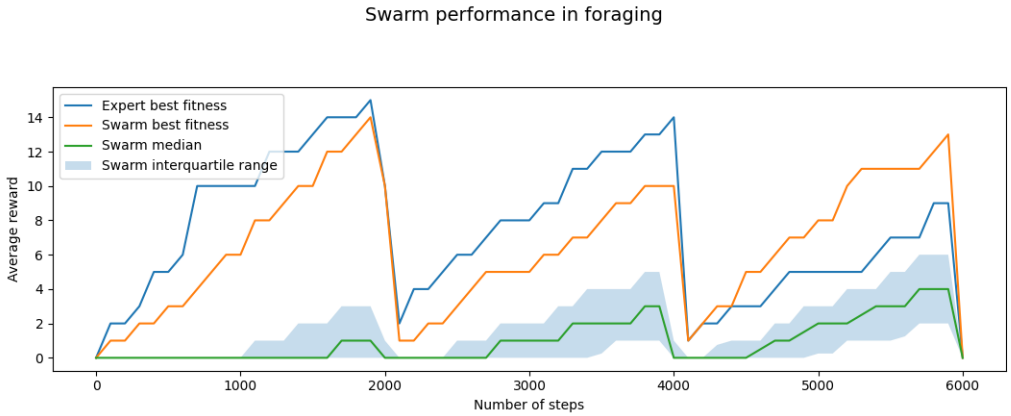


EVALUATION DETAILS :

nbRobots : 100 nbExpertsRobots : 10 nbNotExpertsRobots : 90
nbFoodObjects : 100 maxSizeDictMyBehaviors : 100
hit_ee_transferRate : 0.1 hit_ee_maturationDelay : 0 hit_ee_learningOnlyFromExperts : False
swarmLearningMode : neuralNetworkBackpropagation
nb_hiddenLayers : 1 nb_neuronsPerHidden : 16 nb_neuronsPerOutputs : 2
defaultBehavior : [1,0.5] learningRate : 0.4 distanceEpsilon : 0.25 nbEpoch : 200
nbSteps : 20000 evaluationTime : 100 resetEvaluation : True resetEvaluationTime : 2000

Les deux graphiques ci-dessus montrent respectivement l'apprentissage de l'essaim à partir des experts et l'apprentissage de l'essaim à partir de tous les individus avec une valeur de

fitness meilleure. Nous ne remarquons pas des differences importantes sur les performances de l'essaim.

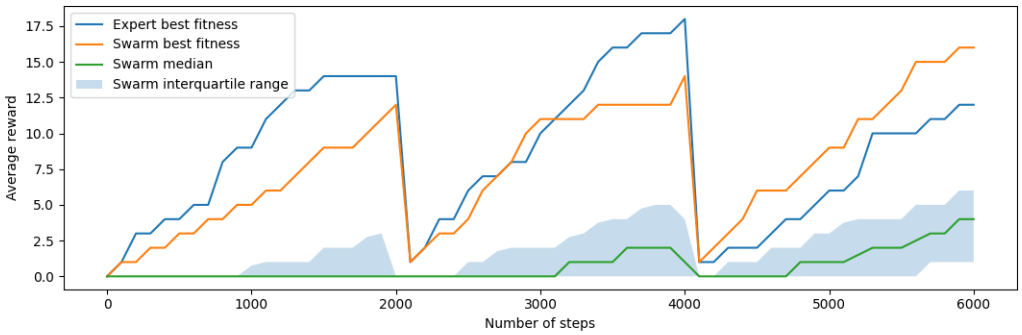


EVALUATION DETAILS :

nbRobots : 100 nbExpertsRobots : 10 nbNotExpertsRobots : 90
nbFoodObjects : 100 maxSizeDictMyBehaviors : 100
hit_ee transferRate : 0.1 hit_ee maturationDelay : 0 hit_ee learningOnlyFromExperts : True
swarmLearningMode : neuralNetworkBackpropagation
nb_hiddenLayers : 1 nb_neuronsPerHidden : 16 nb_neuronsPerOutputs : 2
defaultBehavior : [1, 0.5] learningRate : 0.4 distanceEpsilon : 0.25 nbEpoch : 200
evaluationTime : 100 resetEvaluation : True resetEvaluationTime : 2000

L'apprentissage par rétropropagation du gradient se révèle efficace et confère à chaque individu un mouvement continu qui n'imite pas parfaitement le comportement de l'expert mais assure la poursuite des mêmes objectifs. Cette propriété permet parfois d'avoir des résultats meilleurs en termes de performance par rapport à ceux des experts. Dans le graphique, on observe qu'à la troisième évaluation le meilleur individu non-expert de l'essaim obtient une meilleure performance par rapport à celle de l'expert observé. On remarque aussi la progressive amélioration de la performance des individus non experts ce qui entraîne une diminution des scores de l'expert de part la raréfaction des ressources.

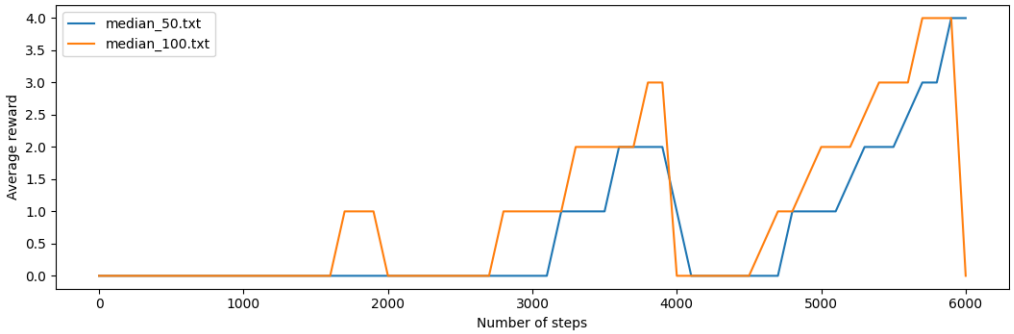
Swarm performance in foraging



EVALUATION DETAILS :

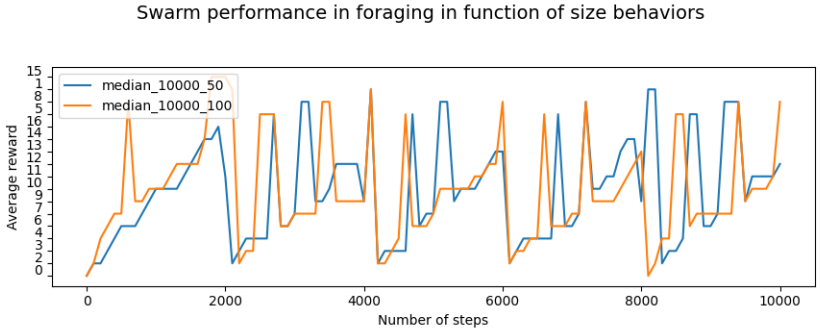
nbRobots : 100 nbExpertsRobots : 10 nbNotExpertsRobots : 90
nbFoodObjects : 100 maxSizeDictMyBehaviors : 50
hit_ee transferRate : 0.1 hit_ee maturationDelay : 0 hit_ee learningOnlyFromExperts : True
swarmLearningMode : neuralNetworkBackpropagation
nb_hiddenLayers : 1 nb_neuronsPerHidden : 16 nb_neuronsPerOutputs : 2
defaultBehavior : [1, 0.5] learningRate : 0.4 distanceEpsilon : 0.25 nbEpoch : 200
evaluationTime : 100 resetEvaluation : True resetEvaluationTime : 2000

Swarm performance in foraging in function of size behaviors



EVALUATION DETAILS :

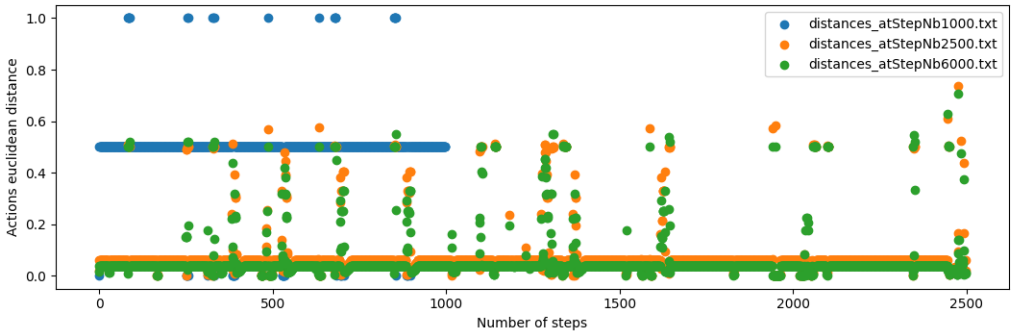
nbRobots : 100 nbExpertsRobots : 10 nbNotExpertsRobots : 90
nbFoodObjects : 100 hit_ee transferRate : 0.1 hit_ee maturationDelay : 0 hit_ee learningOnlyFromExperts : True
swarmLearningMode : neuralNetworkBackpropagation
nb_hiddenLayers : 1 nb_neuronsPerHidden : 16 nb_neuronsPerOutputs : 2
defaultBehavior : [1, 0.5] learningRate : 0.4 distanceEpsilon : 0.25 nbEpoch : 200
evaluationTime : 100 resetEvaluation : True resetEvaluationTime : 2000



EVALUATION DETAILS :

nbRobots : 100 nbExpertsRobots : 10 nbNotExpertsRobots : 90
nbFoodObjects : 100 maxSizeDictMyBehaviors : 100
hit_ee_transferRate : 0.1 hit_ee_maturationDelay : 0 hit_ee_learningOnlyFromExperts : True
swarmLearningMode : neuralNetworkBackpropagation
nb_hiddenLayers : 1 nb_neuronsPerHidden : 16 nb_neuronsPerOutputs : 2
defaultBehavior : [1,0.5] learningRate : 0.4 distanceEpsilon : 0.25 nbEpoch : 200
nbSteps : 10000 evaluationTime : 100 resetEvaluation : True resetEvaluationTime : 2000

La limitation de la base de données des individus focaux à 50 éléments (sur 100 traces totales possiblement acquérable depuis les experts) réduit légèrement la capacité de l'essaim à apprendre de l'expert. Sur une plage de 10000 steps, il semblerait que les différences s'atténuent.



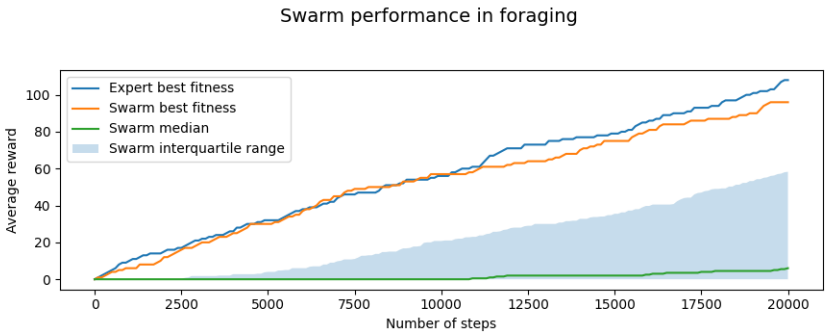
EVALUATION DETAILS :

nbRobots : 100 nbExpertsRobots : 10 nbNotExpertsRobots : 90
nbFoodObjects : 100 hit_ee transferRate : 0.1 hit_ee maturationDelay : 0 hit_ee learningOnlyFromExperts : True
swarmLearningMode : neuralNetworkBackpropagation
nb_hiddenLayers : 1 nb_neuronsPerHidden : 16 nb_neuronsPerOutputs : 2
defaultBehavior : [1, 0.5] learningRate : 0.4 distanceEpsilon : 0.25 nbEpoch : 200
evaluationTime : 100 resetEvaluation : True resetEvaluationTime : 2000

Les choix effectués par le meilleur individu focal en termes d'actions s'améliorent au cours du temps. La bande verte réprésente une distance minimale entre l'action de l'individu focal et l'action de l'expert au moment $t = 6000$ step. Nous remarquons que le meilleur individu focal a déjà trouvé des poids efficaces pour le réseau de neurones au moment $t = 2500$ step.

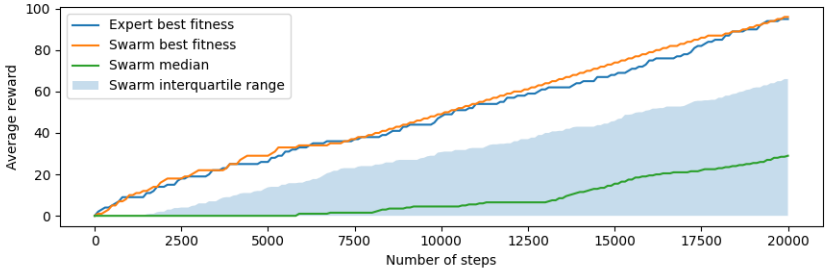
3.2.2 Apprentissage par k Plus Proches Voisins

L'apprentissage par k Plus Proches Voisins permet d'observer une modification nette des comportements chez les individus focaux : les trajectoires changent rapidement suite à la phase d'entraînement, et ressemblent au comportement d'un individu expert. Cela est dû au fait que chaque action, sélectionnée par l'apprentissage de l'individu focal, correspond exactement à au moins une action de la base de données de l'expert, sans perte d'information.



EVALUATION DETAILS :

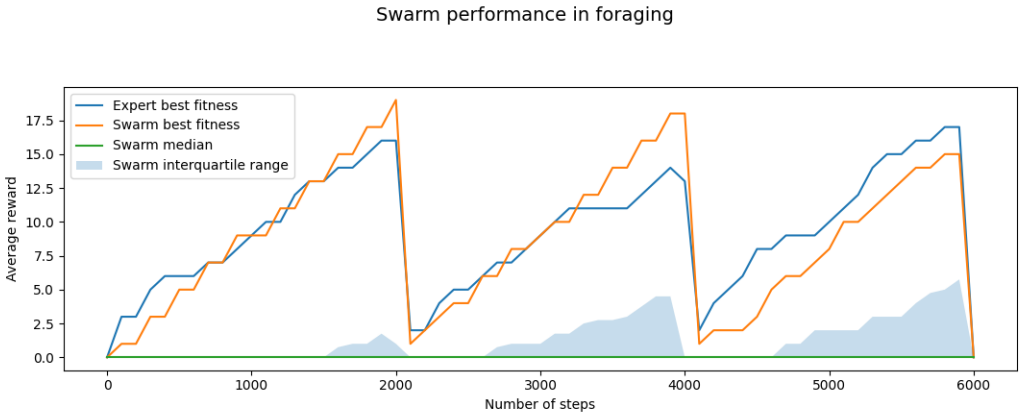
nbRobots : 100 nbExpertsRobots : 10 nbNotExpertsRobots : 90
nbFoodObjects : 100 maxSizeDictMyBehaviors : 100
hit_ee_transferRate : 0.1 hit_ee_maturationDelay : 0 hit_ee_learningOnlyFromExperts : True
swarmLearningMode : kNearestNeighbors
k : 7
nbSteps : 20000 evaluationTime : 100 resetEvaluation : False resetEvaluationTime : 2000



EVALUATION DETAILS :

nbRobots : 100 nbExpertsRobots : 10 nbNotExpertsRobots : 90
nbFoodObjects : 100 maxSizeDictMyBehaviors : 100
hit_ee_transferRate : 0.1 hit_ee_maturationDelay : 0 hit_ee_learningOnlyFromExperts : False
swarmLearningMode : kNearestNeighbors
k : 7
nbSteps : 20000 evaluationTime : 100 resetEvaluation : False resetEvaluationTime : 2000

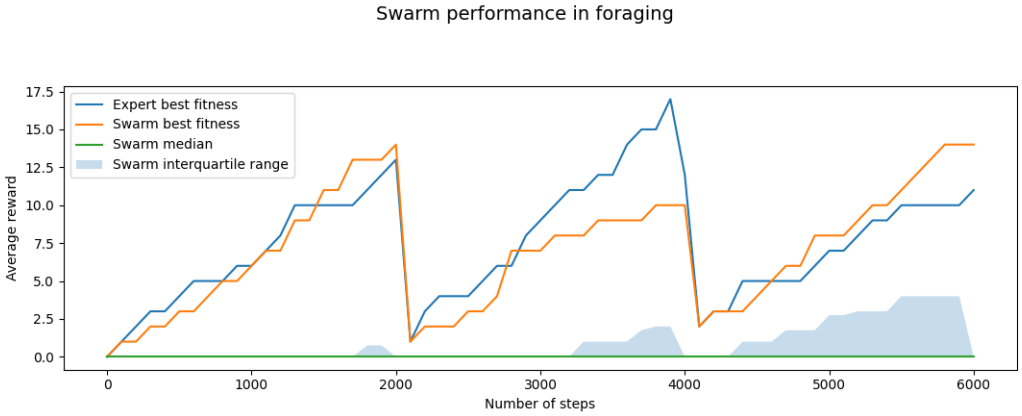
Les deux graphiques ci-dessus montrent respectivement l'apprentissage de l'essaim que à partir des experts et l'apprentissage de l'essaim à partir de tous les individus avec une valeur de fitness meilleure. Nous remarquons une forte ressemblance entre la performance du meilleur individu focal et l'expert dans les deux cas, et une nette progression de l'essaim dans le cas de diffusion non limitée à l'expert mais réalisable par tout robot dans l'essaim.



EVALUATION DETAILS :

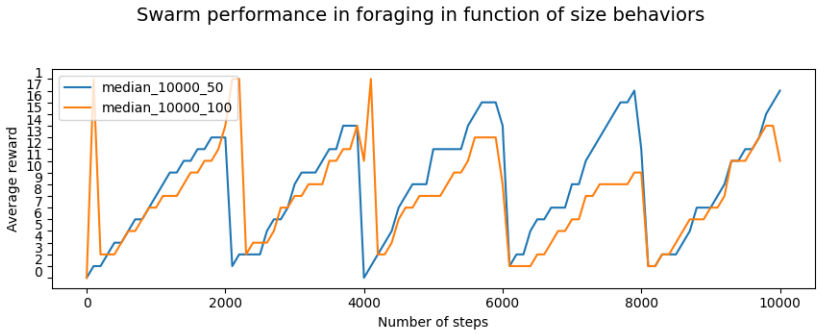
nbRobots : 100 nbExpertsRobots : 10 nbNotExpertsRobots : 90
nbFoodObjects : 100 maxSizeDictMyBehaviors : 100
hit_ee transferRate : 0.1 hit_ee maturationDelay : 0 hit_ee learningOnlyFromExperts : True
swarmLearningMode : kNearestNeighbors
k : 7
evaluationTime : 100 resetEvaluation : True resetEvaluationTime : 2000

L'apprentissage par k Plus Proches Voisins est efficace et tend à imiter parfaitement le comportement de l'expert. Dans le graphique, nous pouvons voir que les performances du meilleur apprenant et de l'expert sont presque superposées. La connaissance générale du groupe évolue en fonction du temps mais progresse moins vite par rapport à l'apprentissage par rétropropagation du gradient.



EVALUATION DETAILS :

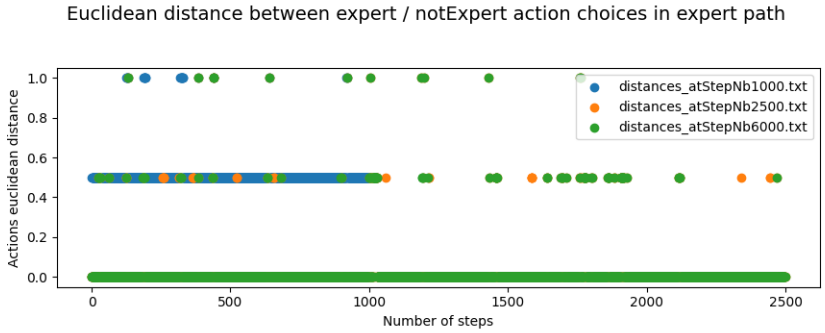
nbRobots : 100 nbExpertsRobots : 10 nbNotExpertsRobots : 90
nbFoodObjects : 100 maxSizeDictMyBehaviors : 50
hit_ee transferRate : 0.1 hit_ee maturationDelay : 0 hit_ee learningOnlyFromExperts : True
swarmLearningMode : kNearestNeighbors
k : 7
evaluationTime : 100 resetEvaluation : True resetEvaluationTime : 2000



EVALUATION DETAILS :

nbRobots : 100 nbExpertsRobots : 10 nbNotExpertsRobots : 90
nbFoodObjects : 100 maxSizeDictMyBehaviors : 100
hit_ee_transferRate : 0.1 hit_ee_maturationDelay : 0 hit_ee_learningOnlyFromExperts : True
swarmLearningMode : kNearestNeighbors
k : 7
nbSteps : 10000 evaluationTime : 100 resetEvaluation : True resetEvaluationTime : 2000

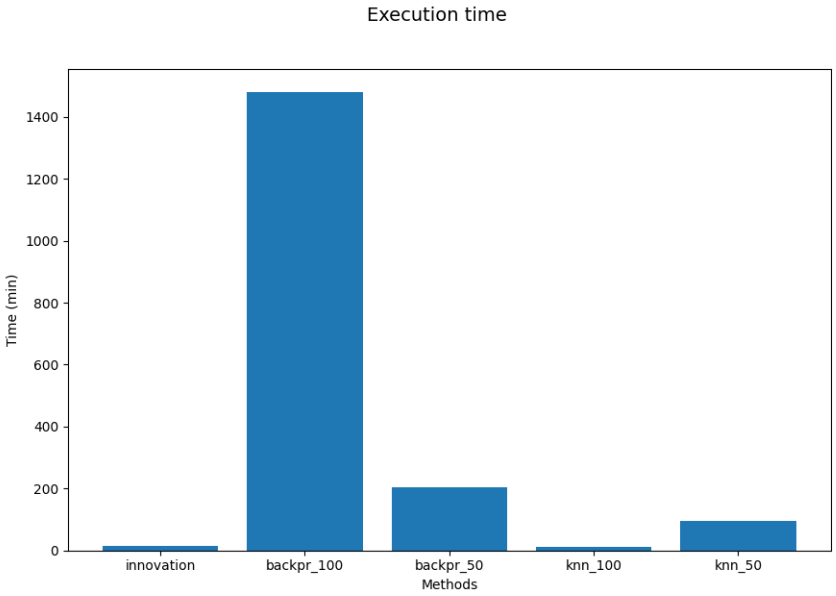
La limitation de la base de données des individus focaux à 50 éléments (sur 100 traces totales qui peuvent être apprise des experts) réduit la capacité de l'essaim à apprendre de l'expert dans un premier temps, mais elle semblerait offrir à chaque individu une diversité capable d'accroître le niveau de performance de l'essaim au fur et au mesure que le nombre de steps augmente.



EVALUATION DETAILS :

nbRobots : 100 nbExpertsRobots : 10 nbNotExpertsRobots : 90
nbFoodObjects : 100 hit_ee transferRate : 0.1 hit_ee maturationDelay : 0 hit_ee learningOnlyFroı
swarmLearningMode : kNearestNeighbors
k : 7
evaluationTime : 100 resetEvaluation : False

Ce graphique des distances entre actions est obtenu à partir de l'expérimentation de 6000 steps sans réinitialisation des conditions initiales représentée ci-dessus. Nous remarquons qu'en fin d'évaluation, l'essaim a un comportement beaucoup plus proche de celui de l'expert - en termes d'actions choisies - par rapport au début de l'évaluation.



Le temps d'exécution requis par les algorithmes d'apprentissage employés varie en fonction de la taille du database des robots pour chacune des deux méthodes. kNearestNeighbors avec taille database 100 est le seul aussi rapide que l'algorithme d'apprentissage par genome auquel on s'inspire dans l'état de l'art. La méthode de retropropagation de gradient est très lente suite à l'entrainement du réseau de neurones pour chaque robot : la limitation du database dans ce cas réduit considerablement les temps de calcul.

Nous tenons à remercier nos encadrants car il a été très enrichissant de travailler avec des experts qui ont su nous guider en valorisant nos attitudes et nos points de force. Nous avons pu compter sur leur disponibilité à tout moment du projet, et progresser vers un resultat que nous a poussé à nous améliorer constamment, en nous donnant encore plus envie de poursuivre sur cette voie.

3.3 Méthode par mimétisme

On décide ensuite de généraliser l'apprentissage en ajoutant de nouvelles contraintes : pour qu'un individu apprenne un comportement, ce dernier doit être observé. On change ainsi les informations transmises lors de la rencontre entre deux individus en un couple (observation, action) envoyé par l'expert. Pour simplifier, on ne considère pas la perte d'information due au changement de point de vue.

3.3.1 Apprentissage ad-hoc

Dans un premier temps, on se propose d'évaluer l'apprentissage par mimétisme en donnant aux apprenants une mémoire extensible et en déterminant le comportement des agents non-experts de la manière suivante :

Entrées *une observation o*;
Résultat : l'action à effectuer a
On cherche l'observation o' présente dans la mémoire telle que la distance o-o' soit minimale;
L'observation o' étant stockée avec une action a' associée : $a \leftarrow a'$;
return a;

La première remarque que l'on puisse faire sur cette méthode d'apprentissage est qu'en plus d'être coûteuse en mémoire, la simulation dure très longtemps. Ceci s'explique par le fait que les observations appartiennent à des segments continus compris entre 0 et 1 ce qui fait que la quasi totalité des comportements observés sont stockés et ensuite mesuré lors du calcul du plus proche voisin.

De plus, lorsqu'un apprenant intègre un comportement, celui-ci est toujours dans le rayon de perception de l'expert, ce qui entraîne une perte de généralisation. Par exemple, les situations où un agent se retrouve en face d'un objet sans qu'aucun autre agent ne soit détecté par ses senseurs, ne peuvent être apprises. Pour palier à ce problème pour cette algorithme, on a fait le choix de ne transmettre que les informations des capteurs frontaux. Lorsque l'on ajoute l'apprentissage inter-apprenants, on observe rapidement un problème : un agent non-expert peut apprendre un mauvais comportement. Pour l'instant, l'on n'a aucun moyen de filtrer ou d'éliminer ces comportements sub-optimaux.

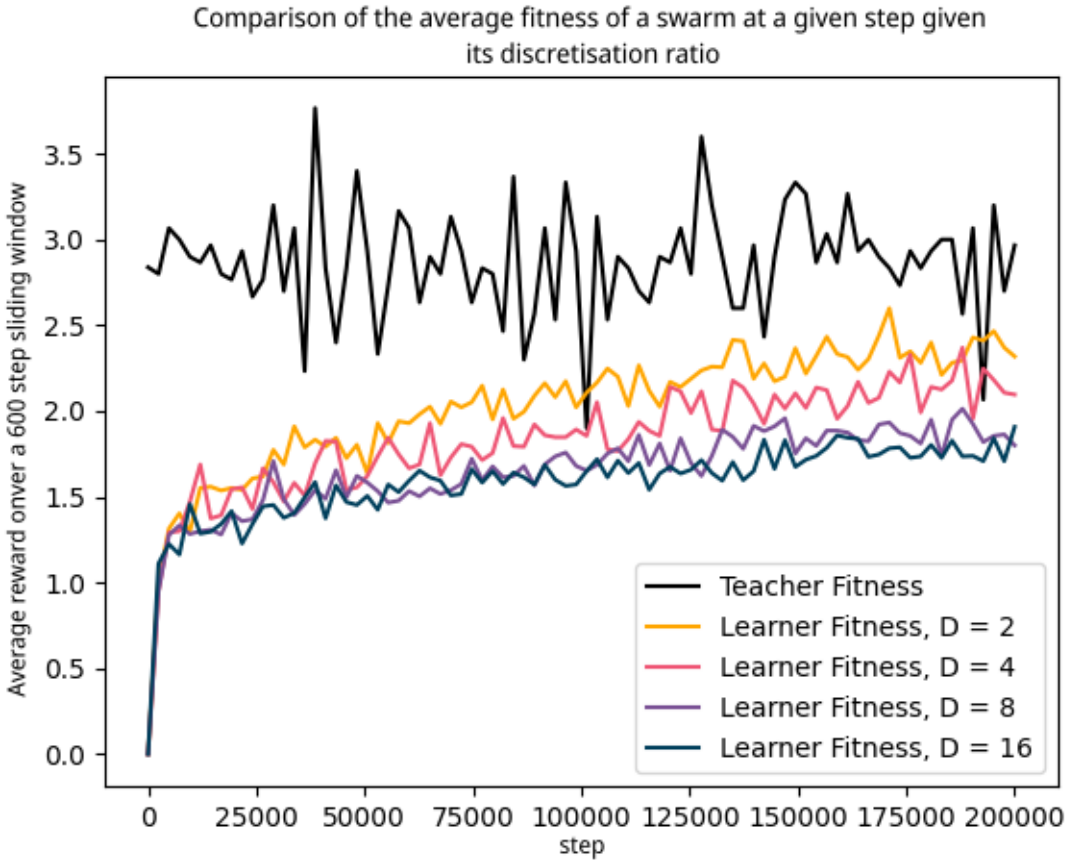
3.3.2 Apprentissage ad-hoc avec discrétisation des entrées

Afin d'adapter la méthode ad-hoc au domaine de la robotique en essaim, et donc de limiter la mémoire nécessaire, on a essayé de discrétiser les entrées. Une telle méthode a plusieurs avantages : elle réduit le domaine des observations

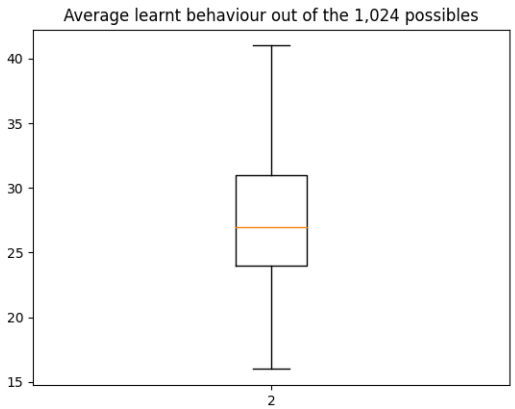
$$nombred'intervalles^{nombredesenseurs} * 2^{nombredesenseursdetypage}$$

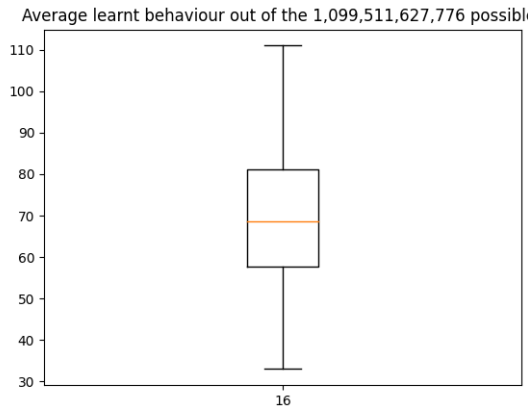
et permet de hasher ces dernières pour accéder plus efficacement aux comportements. Pour cette version, le comportement des apprenants suit les règles suivantes :

Entrées *une observation o*;
Résultat : l'action à effectuer a
$o' \leftarrow$ Discrétiser(o);
if *o' est une clé du dictionnaire mmoire* **then**
  return mémoire[o']
end
return mouvement par défaut;



Par ces analyses, on remarque que lorsque le nombre d'intervalles de discrétisation augmente, la convergeance est de plus en plus lente. Ceci s'explique par un plus grand nombre de comportements à apprendre. Cette supposition est confirmée par la quantité de mouvements appris en fonction du nombre d'intervalles de discrétisation.



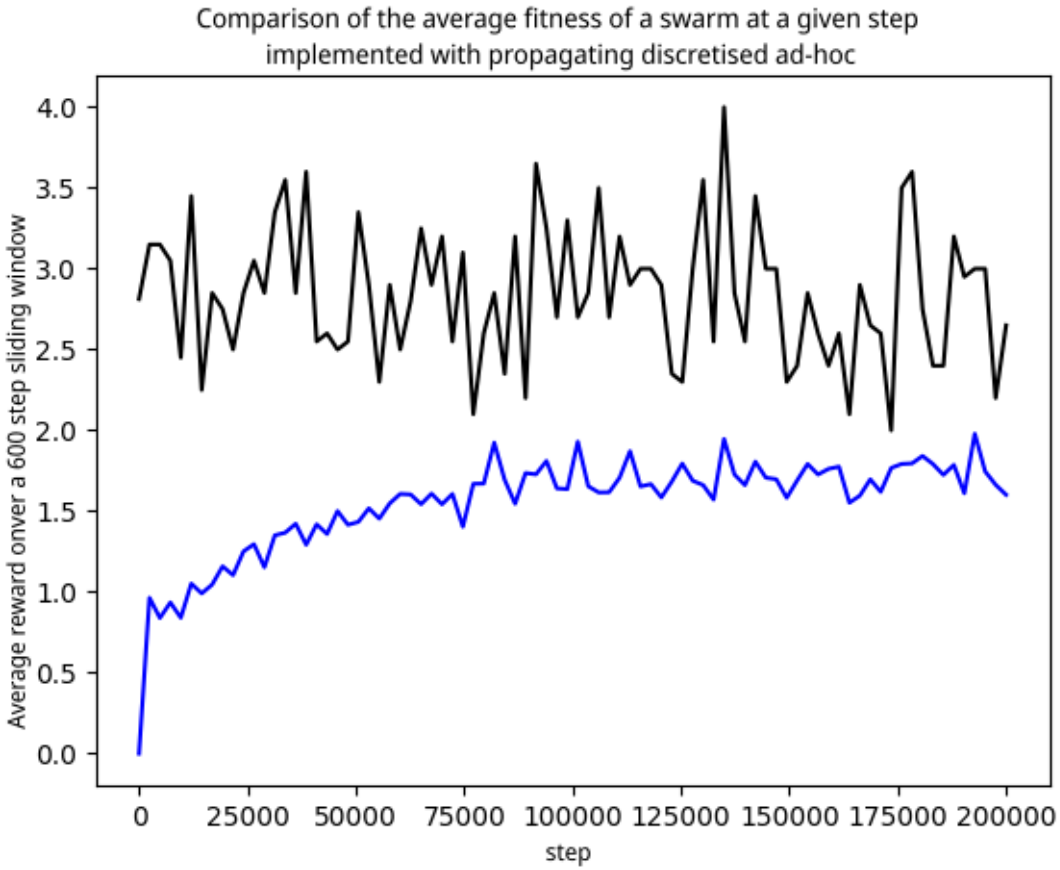


En revanche, si le nombre d'intervalles est insuffisant, une partie du comportement de l'expert peut être perdue. Par exemple, si un expert adopte trois comportements différents en fonction de la valeur du senseur 1, une discrétisation en deux intervalles (respectivement [0,0.5],]0.5,1]) ne permettrait pas l'apprentissage exact de ce comportement.

Pour implémenter la propagation, il suffit de modifier la règle d'apprentissage de la manière suivante :

Lors de la réception d'un message m;
o,$a \leftarrow m$ **if** $a \neq$ *action par défaut* **then**
  $o' \leftarrow$discrétiser(o) **if** o' *n'est pas une clé de du dictionnaire mmoire* **then**
    $mmoire[o'] \leftarrow a$
  end
end

N.B. Pour les tests, on a fait le choix de définir le mouvement par défaut comme (0.5, 0.05), c'est-à-dire de continuer tout droit, mais avec un léger déplacement vers la droite pour éviter aux agents de se coincer.



A première vue, cette expérience semble donner des résultats moins bons que sans propagation, mais elle n'est pas représentative, car nous avons dû limiter le nombre de répétitions de l'expérience par manque de temps. Il faudrait donc refaire cette expérience en la comparant à sa version sans propagation, et ce sur un nombre représentatif d'expériences.

3.3.3 Apprentissage ad-hoc avec plus proche voisin

Une autre option envisagée pour réduire la demande en mémoire de notre algorithme est de toujours exécuter l'action de la mémoire dont l'observation est la plus proche de l'observation actuelle. De plus, afin de limiter le nombre de points insérés dans la mémoire, on a implémenté l'algorithme suivant :

Lors de la réception d'un message m;
o,a, $w \leftarrow m$ On cherche le couple (o', a') tel que la distance $o - o'$ soit minimale;
if $a \neq a'$ **then**
| On ajoute le tuple (o, a, 1) à la mémoire
end
On modifie o de la manière suivante;
$o' \leftarrow (\frac{o' * w + o}{w+1}$

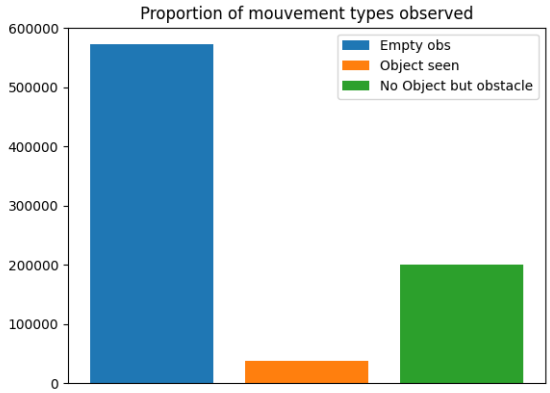
Les principales différences observées lors de ces tests sont dues au choix de l'heuristique de calcul de distance. En effet, une heuristique de simple calcul de distance euclidienne a comme défaut de ne pas prendre en compte le type des objets rencontrés (ceux ci étant encodés sous forme de booléens 0,1). Une solution est d'utiliser une heuristique maximin, mais les résultats restent insuffisants. Au final, nous avons fait le choix de projeter les distances aux obstacles sur différents hyperplans correspondants aux différentes permutations du type d'objet rencontré, puis effectué le calcul de la distance euclidienne.

L'inconvénient majeur de cette méthode est la difficulté à l'adapter à la propagation entre apprenants. Simplement permettre cette dernière provoquerait l'apprentissage de "mauvais" comportements qui limiteraient à long terme la convergence de l'efficacité de la population.

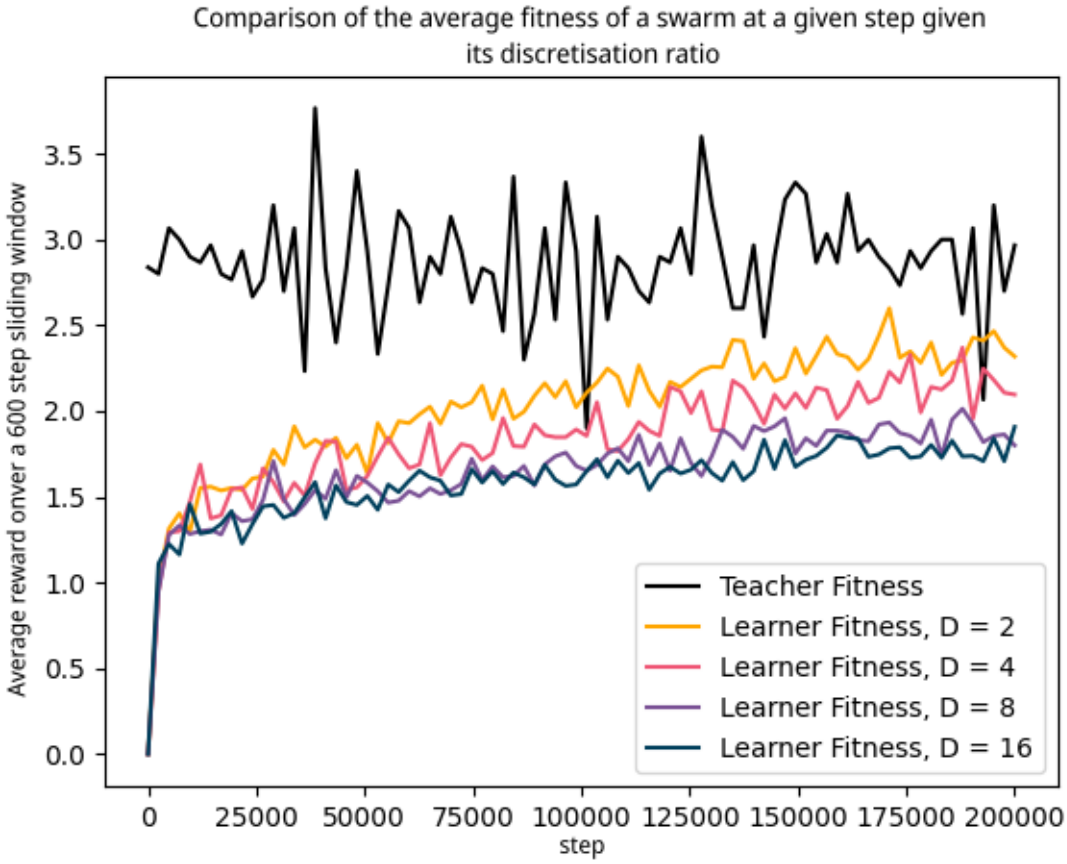
Une solution envisagée à été de calculer pour chaque observation faite la vraisemblance du mouvement associé à cette observation. Mais cette méthode, très calculatoire, demandait une grande mémoire pour avoir une base de comparaison.

3.3.4 Apprentissage sur un réseau de neurones par Backpropagation

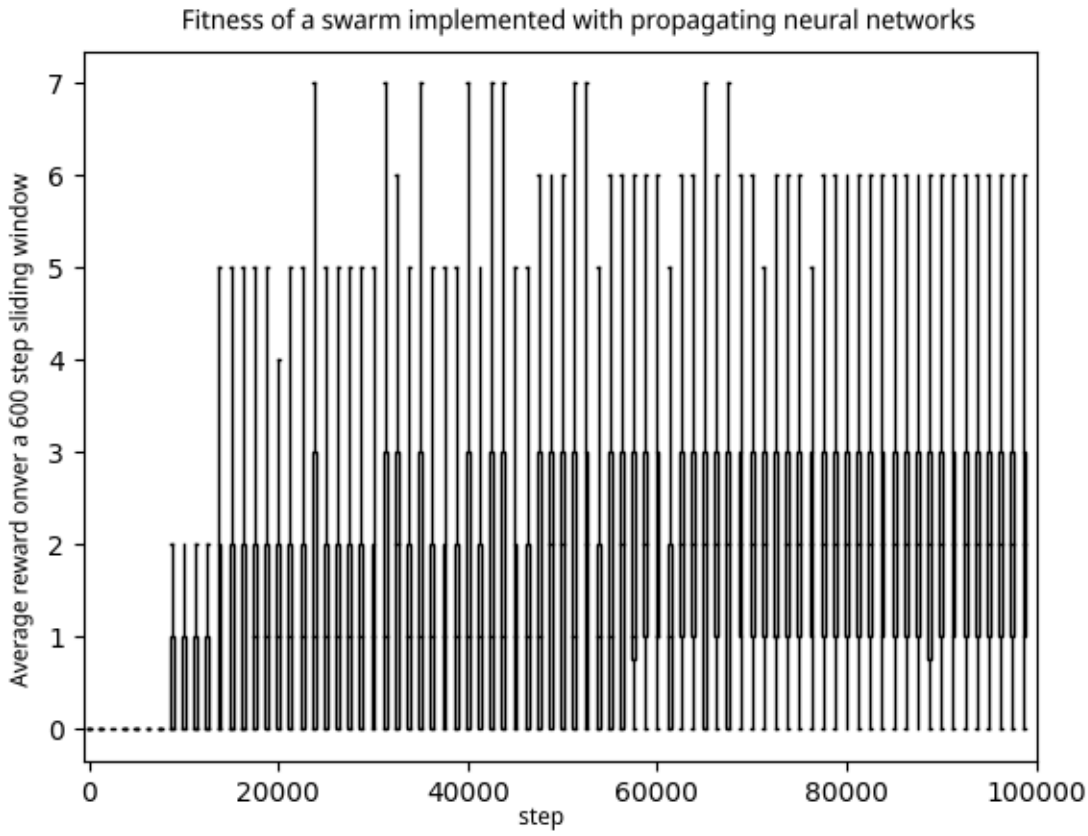
Pour remédier aux différents problèmes remarqués dans les méthodes ad-hoc, on a essayé d'appliquer l'apprentissage par mimétisme sur un réseau neuronal. Ainsi, la présence d'un agent sur un senseur donné devrait disparaître lors de la généralisation. Un problème s'est néanmoins présenté : certaines observations sont beaucoup plus représentées que d'autres.



Pour palier à cela, on a donc renforcé l'apprentissage en pondérant la rétropropagation lors de l'apprentissage en fonction de la distance à l'objectif.



Cette méthode est, comme attendu, plus lente que la version ad-hoc. En revanche elle semble suivre une croissance continue. Il serait envisageable de poursuivre l'expérience jusqu'à la stabilisation de la population.



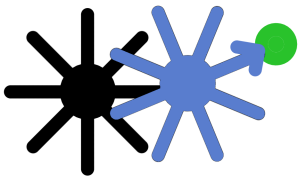
En permettant en plus à l'apprentissage d'avoir lieu entre apprenants, on observe que la moyenne ne semble pas s'améliorer plus vite que dans l'expérience précédente. Néanmoins, l'ensemble de la population évolue de manière plus homogène.

Chapitre 4

Conclusion

A première vue, l'apprentissage par comportement au sein d'un essaim de robots semble être une méthode prometteuse pour l'apprentissage à moindre coût.

Néanmoins, avant de pouvoir vraiment la comparer à d'autres méthodes existantes et la tester dans le monde réel, il faudrait déjà la tester avec la perte de connaissances qu'induit un changement de perspective.



Dans l'exemple ci-dessus, l'apprenant(en noir), ne voit pas l'objet(en vert) vers lequel se dirige l'expert(en bleu), pour lui l'expert ne détecte aucun objet à part l'apprenant.

De plus, certaines de nos expériences n'ont pu avoir de résultats représentatifs par manque de temps. Ainsi, pour la poursuite de notre projet, nous devrions optimiser les différents codes pour permettre des expériences plus représentatives, pour ensuite pouvoir comparer nos résultats avec les algorithmes "state of the art"

Annexe A

Bibliographie

1. Nicolas Bredeche and Nicolas Fontbonne. 2022. Social learning in swarm robotics. Philos. Trans. R. Soc. B Biol. Sci. 377, 1843 (January 2022), 20200309. DOI :https ://-doi.org/10.1098/rstb.2020.0309
2. Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. 2016. Quality Diversity : A New Frontier for Evolutionary Computation. Front. Robot. AI 3, (2016). Retrieved February 26, 2022 from https ://www.frontiersin.org/article/10.3389/frobt.2016.00040
3. Jean-Baptiste Mouret and Stéphane Doncieux. 2012. Encouraging Behavioral Diversity in Evolutionary Robotics : an Empirical Study. Evol. Comput. 20, 1 (2012), 91–133. DOI :https ://doi.org/10.1162/EVCO_a_00048

Annexe B

Cahier des charges

Contexte

Dans le cadre de l'UE de projet du deuxième semestre de Master 1, nous avons été chargés du projet *Apprentissage social et imitation pour la robotique en essaim*.
La robotique en essaim est un sous-domaine de la robotique collective qui désigne un ensemble de robots miniatures aux faibles capacités de calcul individuelles qui collaborent pour accomplir des tâches complexes.
Ce domaine bio-inspiré se révèle très efficace pour la résolution de problèmes et la prise de décision dans des systèmes multi-agents. Mis à part son efficacité évidente, il manque aujourd'hui une méthode théorique permettant de déterminer les meilleurs descripteurs comportementaux et comment les transmettre aux autres agents. L'objectif du projet est d'implémenter en Python ces descripteurs comportementaux et ensuite de définir la stratégie de sélection des individus experts à imiter. Cela permet d'observer les modalités de diffusion de compétences au sein de la population. L'implémentation d'un tel algorithme devrait favoriser la diversité au sein d'une population, et pourrait être plus efficace que d'autres algorithmes existants pour résoudre certains problèmes.

Spécificités générales

Le projet a deux objectifs principaux :

1. Implémenter des descripteurs comportementaux permettant aux robots d'enseigner ou d'apprendre comment réaliser une tâche.
2. Déterminer la stratégie de sélection des individus à imiter (Par exemple : imiter le comportement majoritaire ? imiter le comportement de l' individu le plus âgé ?)

Tout au long du projet, un rendez-vous hebdomadaire en présentiel ou en distanciel avec au moins l'un des encadrants est prévu. Ces rendez-vous ont pour objectif d'informer les encadrants du travail effectué, des éventuelles difficultés rencontrées et de déterminer dans quelle direction orienter la recherche la semaine suivante.

Objectifs spécifiques

- ☐ Rendre le projet et ses résultats disponibles sur la plateforme collaborative Git.
- ☐ Tester ce projet sur le simulateur *Roborobo!* et permettre la reproduction de ces résultats par exécution d'un programme disponible sur Git.
- ☐ Coder en Python un algorithme d'apprentissage par imitation basé sur l'algorithme HIT-EE [1].
  Ce dernier doit être modulable :
  - Choix du taux d'apprentissage.
  - Choix du taux de mutation.
  - Choix du nombre de robots dont robots experts.
  - Choix de la tâche à effectuer (par modification de l'environnement, des données accessibles aux robots et de leur fonction de fitness).
  - Et les différentes caractéristiques modifiables de manière inhérente par *Roborobo!* (taille des robots et des objets, nombre de senseurs, distance des senseurs, etc).

  L'apprentissage se fait par observation des comportements des autres agents :
  - Qu'ils soient définis comme agents-experts
  - Ou qu'ils aient un indice de fitness supérieur
- ☐ Effectuer et analyser une étude statistique des résultats de l'algorithme pour une expérience donnée : le foraging.
- ☐ Identifier les descripteurs comportementaux permettant de maximiser les objectifs de l'essaim et analyser leurs performances sur la population.
- ☐ Rédiger un rapport détaillé sur les expériences réalisées, leurs résultats et les ouvertures possibles.
- ☐ Présenter ces résultats lors de la soutenance.

Ressources

Ressources disponibles:

- Deux personnes travaillent sur le projet
- Un rendez-vous régulier avec les encadrants, spécialisés dans le domaine de la robotique en essaim
- Des publications scientifiques de référence :
  - Nicolas Bredeche and Nicolas Fontbonne. 2022. Social learning in swarm robotics. *Philos. Trans. R. Soc. B Biol. Sci.* 377, 1843 (January 2022), 20200309. DOI:https://doi.org/10.1098/rstb.2020.0309
  - Nicolas Bredeche. 2019. HIT-EE: a novel embodied evolutionary algorithm for low cost swarm robotics. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ACM, Prague Czech Republic, 109–110. DOI:https://doi.org/10.1145/3319619.3321928
  - Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. 2016. Quality Diversity: A New Frontier for Evolutionary Computation. *Front. Robot. AI* 3, (2016). Retrieved February 26, 2022 from https://www.frontiersin.org/article/10.3389/frobt.2016.00040
  - Jean-Baptiste Mouret and Stéphane Doncieux. 2012. Encouraging Behavioral Diversity in Evolutionary Robotics: an Empirical Study. *Evol. Comput.* 20, 1 (2012), 91–133. DOI:https://doi.org/10.1162/EVCO_a_00048
- Simulateur *Roborobo!*

Contraintes :

- Temps disponible limité pour travailler ensemble en présentiel (un jour/semaine)
- Pas de serveurs dédiés pour faire fonctionner les expériences sur de très longues durées

Annexe C

Manuel utilisateur

C.0.1 Méthode par transmission de genome et méthode par transmission de comportements

— Se positionner sur le répertoire *part1Innovation_part2Diffusion\ASIRE_project*

— Se positionner dans le fichier :

  — *part1_main_innovation.py* pour exécuter l'apprentissage par genome ;

  — *part2_main_diffusion.py* pour exécuter l'apprentissage par comportements.

— Dans le fichier *main* choisi, en début de page, fixer les paramètres à utiliser :

  — section ***configuration file parameters***. Fixer le nombre de individus experts *nbExpertsRobots*, le nombre d'individus focaux *nbNotExpertsRobots* et le nombre d'objets *nbFoodObjects* pour l'expérience. Remarque : il ne faut pas modifier le fichier de configuration qui se trouve dans le répertoie *config*.

  — section ***learning mode configuration***. Choisir la mèthode d'apprentissage pour l'expérience courante : commenter la ligne de la mèthode à ne pas utiliser.

  — section ***parameters***. Fixer le nombre total de steps *nbSteps* de l'expérience et la taille de la base de données des individus *maxSizeDictMyBehaviors*. Affecter *True* à la variable *learningOnlyFromExperts* pour limiter la possibilité de diffuser ses propres caractéristiques (genome ou traces comportamentales) aux seuls individus experts ; affecter *False* pour permettre à tous les individu de diffuser.

  — section ***debug and plot parameters***. Il est possible de visualiser les détails d'execution des algorithmes en affectant la valeur *True* aux différentes variables *debug_ *.

  Remarque : les fichiers *part1_bestParameters.txt* et *part2_bestParameters.txt* conservent une copie des meilleurs paramètres testés lors des expériences.

— Exécuter le fichier *main* choisi, *part1_main_innovation.py* ou *part2_main_diffusion.py*.

C.0.2 Méthode par mimétisme

Comment recréer les expériences citées :

Installer Roborobo et lancer une instance d'Anaconda.

Se déplacer dans le dossier $mimetic_Learning$:

Choisir les paramètres dans le fichier Cons.py :

- **Global**
  - VERBOSE = True / False
  - $DATA\_SAVE$ = True / False
  - $SAVE\_FILE$ = "filepath"
  - $OVERWRITE\_FILE$ = True / False
- **ExtendedAgents** *Options pour l'apprentissage par réseau de neurones*
  - $NB\_HIDDENS$ = int
  - $EVALUATION\_TIME$ = 600 *Taille de la sliding window*
  - $MEMORY\_RANGE$ = 20
  - $LEARNING\_STEPS$ = 30 *Nombre de répétitions de la phase d'apprentissage*
  - $LEARNING\_RATE$ = 0.8
  - $MUTATION\_RATE$ = 0.
- **Foraging Task**
  - $LEARNING\_ALGORITHM$ = "adhoc" *"adhoc" or "neural"*
  - NB_ITEMS = 100 *Nombre d'objets instanciés*
  - $REGROWTH\_TIME$ = 10 *Durée avant réapparition des objets*
  - $CHANGE\_POSITION$ = True *Si les objets réapparaissent à d'autres endroits*
  - NB_ITER = 200000 *Durée de l'expérimentation*
- **NeuralLearner**
  - $PROPAGATION$ = False *Apprentissage apprenant - apprenant*
  - $DECAY\_FUNCTION$ = constantLearningRate *Choix de la fonction de réduction du taux* d'apprentissage
  - $DECAY\_RATIO$ = 0.01
- **MemoryAgents**
  - $EXPERT\_SPEED$ = 1
  - $LEARNING\_GAP$ = 60
  - $MEMORY\_SIZE$ = 100 *Nombre de messages stockés simultanément*
  - $NB\_LEARNER$ = 90 *Nombre de robots apprenants*
  - $DISCRETISE\_RATIO$ = 2 *Si -1 : plus proche voisin*
  - $LEARNT\_BEHAVIOUR\_PROPAGATION$ = False *Apprentissage apprenant - apprenant*

Exécuter le runner : python $Foraging\_Task$