Intégration de rôles, de flocking et de *Language Games* dans un essaim de robots pour une tâche d'extinction de feux

PROJET TUTORÉ
Master 1 Sciences Cognitives

RAPPORT DE RÉALISATION
Comportements collectifs, langages et culture incarnés pour les robots coopératifs et collaboratifs

PRÉSENTÉ PAR
Erin Bernardoni - Manuella Comesse - Laïla Moatassim

ENCADRÉ PAR
Laurent Ciarletta - Christine Bourjot

ORGANISME D'ACCUEIL
Équipe SIMBIOT - LORIA

Institut des Sciences du Digital, Management et Cognition
Université de Lorraine

Année universitaire 2021-2022

Table des matières

Introduction **2**

Le choix d'un modèle couplant rôles, flocking et jeux de langage **4**
Les pistes non retenues 4
Les rôles 5
Le flocking 7

Une construction formelle du modèle **8**
Un automate fini non déterministe pour les rôles 8
Des algorithmes pour les déplacements 9
Un POMDP et de l'apprentissage par renforcement pour les jeux de langage 10

Une tentative d'application réaliste du modèle pour l'extinction de feux **11**
Présentation de la tâche 11
Expériences 13
Présentation des résultats 13
Efficacité du modèle selon la valeur *min_ agents* 14
Efficacité du modèle selon le type de déplacement 15
Évolution des mots dans le temps 17
Récapitulatif et interprétation des résultats 18

Conclusion **20**

Bibliographie **24**

Introduction

Ce projet tutoré est dirigé par le responsable de l'équipe de recherche SIMBIOT du LORIA, Laurent Ciarletta, ainsi que par une membre de cette même équipe, Christine Bourjot. Il fait suite à un premier projet tutoré dans les domaines de la robotique coopérative et collaborative et des systèmes multi-agents, effectué l'an dernier par Lionel Aquilanti, Rafaelle Belorgey et Erwan Plantec [1].

Leur travail s'inspire de l'évolution culturelle avec l'utilisation des Language Games, permettant aux agents spatialement proches de communiquer entre eux en s'échangeant des mots codant leurs comportements, dans le but d'apprendre les comportements à adopter selon leurs observations de l'environnement. La formalisation de leur approche est définie au moyen des processus décisionnels de Markov partiellement observables (POMDP). Le modèle alors formulé a été analysé en simulation dans le cadre d'une tâche de recherche de cibles dans un monde torique, où une cible est détruite quand un agent passe dessus, ce qui déclenche aussi l'apparition d'une nouvelle cible à une localisation aléatoire. Des dangers ralentissant les agents ont été introduits dans certaines expériences. Les résultats obtenus ont permis au groupe de l'an passé de conclure que leur modèle était prometteur pour l'adaptation de comportements. Malgré cela, certaines limites ont été soulevées, notamment une instabilité des comportements lorsqu'il y a peu d'agents et une adaptation jugée non-optimale face aux dangers.

Notre projet cette année vise alors à poursuivre leur étude en restant dans l'approche des jeux de langage, afin qu'un système multi-agents ou qu'un essaim de robots puisse encore mieux s'adapter de manière autonome et continue à des environnements inconnus. Nous avons rassemblé dans notre travail bibliographique plusieurs inspirations biologiques, psychologiques et linguistiques à partir desquelles un modèle d'essaim de robots peut être réalisé. Nous désirions que le modèle puisse s'appliquer à un cas concret d'utilisation, à savoir une tâche de recherche et d'extinction de feux. Afin d'être plus réalistes, les feux requièrent plusieurs agents pour s'éteindre. Nous avons amélioré le modèle de l'an passé en y intégrant des rôles et du flocking, couplés à des jeux de langage, pour cette tâche précise.

Plusieurs expériences ont été réalisées afin d'évaluer l'efficacité du modèle, notamment selon les différents types de flocking mais aussi selon le nombre minimal d'agents requis pour éteindre un feu. Cette efficacité est évaluée selon le nombre d'itérations nécessaires aux agents pour éteindre tous les feux de la simulation. Les Language Games sont repris dans notre modèle avec l'introduction de deux nouveaux concepts : un premier concept pour l'apprentissage du nombre d'agents requis afin d'éteindre un feu et un second

pour détecter et suivre le signal d'un agent recruteur. L'évolution de ces mots sera aussi étudiée dans les résultats expérimentaux.

La première partie du rapport consiste en une présentation du contexte ainsi que des choix qui ont été opérés à partir du travail de l'an passé et de notre état de l'art, afin de construire le modèle présenté dans le rapport. La seconde partie propose une formalisation pour répondre au problème des essaims de robots à partir de laquelle un modèle est élaboré. Dans une troisième partie est présentée l'application du modèle en simulation dans Netlogo, ainsi que les résultats obtenus pour différentes valeurs des paramètres. Enfin, une discussion des résultats est réalisée dans une dernière partie où des pistes d'améliorations sont suggérées pour le modèle.

Le choix d'un modèle couplant rôles, flocking et jeux de langage

Nous avions conclu notre rapport bibliographique en évoquant plusieurs pistes sur lesquelles nous avions décidé de nous concentrer au second semestre. Nous allons revenir ici sur les réflexions et les recherches supplémentaires que nous avons consacrées à chacune d'elle lors de ce semestre de réalisation, afin de mieux comprendre l'abandon de certains axes et nos choix finaux de solutions. Par ailleurs, il nous paraissait essentiel d'expliquer dans ce rapport des pistes qui n'ont pas été retenues, puisqu'elles pourraient tout de même servir aux potentiels futurs étudiants qui reprendraient notre travail.

Avant toute chose, nous avons commencé par définir la tâche applicative de notre modèle. Nous souhaitions rester dans une tâche de recherche de cibles afin d'éventuellement comparer nos résultats à ceux obtenus l'an passé. Toutefois, nous tenions à donner un contexte à la tâche, dans le but de pouvoir prendre en compte certains aspects réalistes et de réfléchir à des solutions plus précises. C'est pourquoi nous avons décidé de travailler sur un modèle d'extinction de feux dans une forêt par un essaim de robots et/ou de drones. Nous expliquerons en détail l'application dans la partie qui lui est dédiée.

Les pistes non retenues

Une piste que nous avons rapidement laissée de côté est celle du tableau noir. Pour rappel, un tableau noir ou tableau d'affichage est un dispositif situé dans l'environnement sur lequel les agents peuvent déposer un message ou lire ceux déposés par d'autres agents. De par l'ancienneté du concept, l'impossibilité de prévoir les agents qui liront les messages, et le mécanisme centralisé gênant lorsque le tableau dysfonctionne, il n'est guère plus utilisé aujourd'hui.

Un autre axe de recherche était l'implémentation d'une hiérarchie dans l'essaim robotique. L'auto-organisation et la décentralisation sont des principes chers aux essaims de robots. Toutefois, Dorigo et al. [2] pensent que les essaims seraient parfois plus simples à contrôler si ces concepts étaient couplés à un contrôle sous forme hiérarchique, comme chez certaines espèces de guêpes par exemple. Nous nous sommes penché.es en particulier sur la technique de l'apprentissage par renforcement féodal défini par Dayan et Hinton [3], qui offre un contrôle hiérarchique strict parallélisant l'apprentissage par renforcement dans le temps et dans l'espace. A chaque niveau de la hiérarchie, les individus sont des "managers" pour les individus "sous-managers" du niveau directement inférieur. Autrement dit, les supérieurs apprennent à donner des

tâches aux inférieurs, et les inférieurs apprennent à satisfaire leur supérieur. Les managers de niveaux différents ont des tâches différentes à accomplir, de la plus abstraite ou complète en haut de la hiérarchie aux plus concrètes ou partielles en bas. La hiérarchie que les auteurs présentent est dite féodale en raison de la totale liberté laissée aux managers de choisir les tâches, de sélectionner les sous-managers qui doivent les effectuer, et de les récompenser ou les punir selon s'ils ont obéi ou non aux ordres du manager. Les individus sont donc à la fois les managers des individus du niveau inférieur, et les sous-managers des individus du niveau supérieur. Cela entraîne un apprentissage par renforcement récursif jusqu'à ce que l'objectif du manager de plus haut niveau soit atteint. Concrètement, un exemple d'objectif global serait de trouver une cible dans un environnement. Le manager ou leader supérieur pourrait ordonner à ses inférieurs de quadriller chacun une zone de l'environnement, qui pourraient ordonner à leur tour à leurs inférieurs de quadriller une sous-partie de leur zone, et ainsi de suite. Un frein important nous a dissuadé à mettre en place une telle hiérarchie de robots ou de drones : elle est très vulnérable face aux dysfonctionnements. En effet, il suffit d'un seul individu défectueux pour potentiellement corrompre le fonctionnement normal de l'essaim tout entier, et rendre la tâche irréalisable [2]. Une solution que nous avons envisagée est de faire émerger un nouveau leader, du même rang ou de rang supérieur à celui dysfonctionnel et qui effectuerait la tâche à sa place pour compenser. N'ayant pas trouvé de manière simple par laquelle un leader pourrait émerger dans un essaim de robots hiérarchisés de la sorte, nous avons définitivement abandonné cette piste.

Les rôles

Finalement, nous nous sommes concentré.es sur l'idée des rôles, qui nous paraissait déjà la plus envisageable et la plus intéressante lors du premier semestre. Plusieurs notions sont liées à celles de rôles, telles que l'homogénéité ou l'hétérogénéité, la division du travail, le travail d'équipe, les groupes.

La première question que nous nous sommes posée a été de savoir s'il valait mieux avoir des robots homogènes qui pourraient apprendre pendant les expériences à adopter un rôle spécifique ; ou bien avoir des robots hétérogènes à qui nous assignerions des rôles différents selon leurs capacités avant le lancement des expériences. Brièvement, les robots homogènes sont identiques et exécutent le même programme de contrôle, ce qui les rend remplaçables. Leur comportement peut se spécialiser, mais cela sera seulement dû aux interactions individuelles avec l'environnement. En revanche, les robots hétérogènes sont différents physiquement et/ou comportementalement [2]. Dans un article datant de 2005 [4], Sahin liste les caractéristiques des essaims de robots qui sont essentielles selon lui : ceux-ci doivent être composés de peu de groupes de robots, être en nombre important dans chaque groupe

et être homogènes. Les robots hétérogènes ne peuvent alors pas être considérés comme des essaims de robots. De nos jours, Dorigo et al. [2] trouvent que l'hétérogénéité est une piste de recherche intéressante qui existe chez certains essaims d'insectes, mais qui est encore peu transposée aux essaims robotiques à cause de sa complexité. Elle permettrait pourtant une meilleure flexibilité des comportements, une meilleure adaptation à la nouveauté, et une résilience aux perturbations. Les robots peuvent être considérés hétérogènes de multiples manières : par exemple, ils peuvent être physiquement différents et essayer de se coordonner, ou bien être physiquement identiques mais apprendre à se spécialiser pour accomplir des tâches différentes (cela revient donc à des robots homogènes qui se spécialisent).

Dans un premier temps, nous étions davantage inspiré.es par l'hétérogénéité par rapport à notre tâche d'extinction de feux. En effet, nous avons imaginé qu'une partie des drones pourrait seulement utiliser leur caméra pour s'occuper du traitement d'image, tandis qu'une autre partie utiliserait uniquement leur système de géolocalisation pour le positionnement. Une autre possibilité d'hétérogénéité est le déploiement conjoint de drones pour le repérage aérien des feux et de robots mobiles au sol se rendant aux endroits enflammés communiqués par les drones afin de les éteindre. Cependant, dans un objectif à plus long terme de ce projet tutoré, nous avons réalisé que l'hétérogénéité demande des codes différents à produire pour chaque type de dispositif, ce qui peut faire émerger des problèmes de compatibilité. Bien que la piste serait donc à creuser, nous avons préféré conserver des robots ou des drones homogènes, pour lesquels il existe davantage d'exemples de solutions dans la littérature et la mise en œuvre est plus sûre grâce à l'interchangeabilité.

Les questions suivantes posées portaient sur le choix des rôles et la manière dont ils allaient être définis : de quels rôles avons-nous besoin ? Allons-nous fixer un rôle précis à chaque robot ? Sinon, comment faire émerger ou faire apprendre un rôle dans un essaim de robots ? Les robots pourront-ils changer de rôle en cours d'expérience ?

Quelques réponses instinctives nous sont venues à l'esprit, comme définir des groupes chacun spécialisé pour la recherche d'un type particulier de cibles ou ayant chacun un comportement différent (explorer ou exploiter l'environnement, ...), mais nos recherches nous ont aiguillés vers l'étude "Teamwork in Self-Organized Robot Colonies" de Nouyan et al. de 2009 [5]. Dans l'expérience menée, des robots homogènes ont pour but de ramener une proie à leur nid en réalisant des tâches en équipe, c'est-à-dire en réalisant différentes sous-tâches en même temps, ce qui implique une division du travail. Les sous-tâches sont l'exploration de l'environnement, le recrutement de robots près du nid vers la proie, la formation d'un chemin entre la proie et le nid, l'auto-assemblage en structures pour tirer la proie, et le transport groupé de

la proie jusqu'au nid. Les robots changent de sous-tâche (ou de rôle) selon le contexte dans lequel ils se trouvent. Par exemple, ils sont par défaut dans le rôle d'exploration, mais passent au rôle de recrutement dès qu'ils perçoivent la proie. L'assignement de rôles suit donc des règles statiques (déterministes et stochastiques), et par conséquent le système n'est pas adaptatif. Malgré cela, une division complexe et efficace du travail d'équipe a pu être observée. Nous nous sommes ainsi basés sur certaines caractéristiques du modèle de cette expérience, notamment les transitions déterministes et probabilistes entre rôles, pour notre propre modèle que nous détaillerons dans la partie de formalisation.

Le flocking

En parallèle de ces recherches, nous avons davantage réfléchi à notre tâche d'extinction de feux pour tenter de la rendre plus réaliste comparée à la tâche de recherche de cibles de l'an passé. Déjà, nous voulions que la présence de plusieurs agents soit nécessaire autour des feux pour parvenir à les éteindre, et que l'extinction prenne du temps (contrairement à l'an passé où il suffisait qu'un seul agent trouve une cible pour la détruire immédiatement). Cette nouvelle contrainte nous a poussé.es à rechercher comment des drones pourraient faire du flocking. En effet, ce type de déplacement groupé et coordonné, comme des nuées d'oiseaux ou des bancs de poissons, faciliterait grandement le recrutement d'agents, car il permettrait d'avoir directement un nombre suffisant d'agents proches spatialement pour éteindre un feu lorsque celui-ci serait découvert par un membre du groupe de flocking. Le premier modèle de flocking a été introduit par Reynolds en 1987 dans son article "Flocks, Herds, and Schools : A Distributed Behavioral Model" [6]. Pour qu'un flocking apparaisse, chaque agent doit suivre trois règles simples de cohésion, de séparation et d'alignement. Respectivement, il doit rester proche des autres agents, tout en évitant les collisions avec les autres, et en essayant de garder la même vitesse et direction que les autres.

Une autre modification que nous souhaitions apporter est le type d'environnement. Le monde torique employé lors des simulations du projet précédent faisait que les agents passant au-delà du côté gauche réapparaissaient du côté droit, ce qui est impossible dans la réalité et introduit alors de possibles biais. L'environnement dans lequel seront placés nos agents est donc une "boîte" fermée avec des bords infranchissables. Une gestion des demi-tours lorsque les agents rencontrent ces bords est nécessaire afin qu'ils ne restent pas bloqués, d'autant plus en étant en flocking pour éviter les collisions. Une piste de solution très intéressante à ce sujet s'inspire des bancs de poissons qui font des demi-tours collectifs rapidement sans pour autant se heurter [7]. Nous ne l'avons pas retenue à cause de la complexité du modèle proposé qui comporte beaucoup de paramètres. Une autre suggestion consiste à donner

une force repoussante aux bords qui soit proportionnelle à la distance entre l'agent et le bord, ou bien à donner une force attractive au centre de l'environnement pour que les agents ne s'en éloignent pas trop. Cette dernière solution définirait une zone circulaire de forêt attractive autour du centre, ce qui serait plus logique qu'une zone carrée. Au final, nous avons opté pour un environnement de forêt circulaire afin que les agents ne soient pas bloqués dans des coins, avec une solution de demi-tours plus simple où les agents se retournent de 180° quand ils heurtent un bord du cercle pour repartir dans le sens opposé. Cela reste envisageable en réalité avec des drones multirotors capables d'effectuer des changements de direction importants et brusques, même s'il resterait à gérer l'évitement de collisions entre les drones dans ces situations.

Enfin, il nous restait à trouver comment incorporer les Language Games de l'an passé dans notre modèle. Une idée s'est présentée afin d'obtenir un déplacement en flocking plus optimal et adaptatif dans la tâche d'extinction des feux. Bien qu'un nombre minimal d'agents doit être près d'un feu pour l'éteindre, il n'est pas judicieux d'avoir trop d'agents présents. En effet, certains agents y sont inutiles, alors qu'ils pourraient passer ce temps à explorer l'environnement pour chercher à éteindre d'autres feux en parallèle de celui-ci. Il faudrait donc que la population d'agents puisse former plusieurs groupes (ou clusters) de flocking de taille suffisante sans être trop grands. C'est là que les jeux de langage interviennent : les agents pourraient apprendre ce nombre minimal d'agents pour éteindre un feu et pour se déplacer en flocking, grâce à un "mot" codant ce nombre et que les agents échangeraient au fil des parties de jeux de langage. Là encore, ce mécanisme sera détaillé dans les parties ultérieures.

Une construction formelle du modèle

Notre modèle est un assemblage de plusieurs modèles correspondant aux notions retenues précédemment. Il se place dans un monde discret constitué de cases. Nous en proposons une formalisation, sans perdre de vue sa transposition en simulation informatique dans la partie suivante.

Un automate fini non déterministe pour les rôles

Pour le modèle des rôles et de leurs transitions, nous nous sommes inspiré.es du modèle de Nouyan et al. [5]. Dans notre tâche, les agents peuvent prendre trois rôles :

— Par défaut, ils *explorent* l'environnement en s'y déplaçant ;
— Quand ils perçoivent un feu, ils s'y arrêtent et tentent de le *détruire* ;
— S'ils n'ont pas réussi à éteindre le feu, ils peuvent envoyer des signaux de *recrutement* aux agents alentour.

Les transitions entre les rôles peuvent être représentées à l'aide d'un automate fini non déterministe (cf. figure 1). Lorsqu'un agent suit un rôle à un moment donné, il est dans un certain état. Le passage d'un état vers un autre état (donc le changement de rôle) peut survenir à cause d'un événement particulier, ou selon une probabilité de transition.

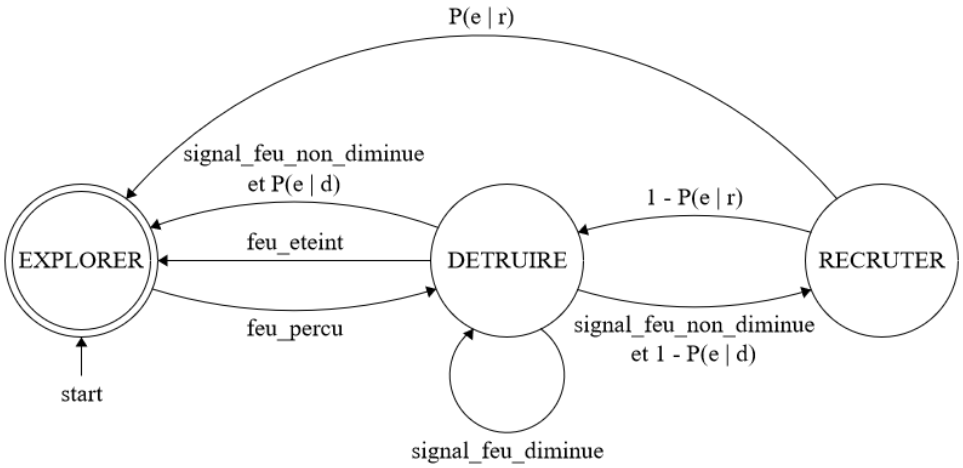

FIGURE 1 – Diagramme d'automate fini non déterministe représentant les transitions entre rôles

Au départ du modèle, tous les agents explorent. Quand un agent perçoit un feu, il passe à l'état de destruction du feu pendant un temps fixé.
Dès que le feu est éteint, l'agent passe du mode destruction au mode exploration, même si le temps de destruction fixé n'est pas encore complètement écoulé.
Si l'agent arrive au terme du temps de destruction et que le signal émis par le feu a diminué comparé à avant qu'il essaie de détruire le feu, cela signifie qu'il y avait assez d'agents autour de ce feu qui cherchaient à l'éteindre ; l'agent reste alors à nouveau en mode destruction pendant le temps fixé. Au contraire, si le signal du feu n'a pas diminué, cela signifie qu'il n'y avait pas assez d'agents près du feu cherchant à l'éteindre. Avec une certaine probabilité P(e | d), l'agent repasse dans l'état d'exploration, sinon il adopte le rôle de recrutement pendant un temps fixé.
A la fin de ce temps de recrutement, l'agent passe dans l'état d'exploration selon une probabilité P(e | r), sinon il repasse dans l'état de destruction.

Des algorithmes pour les déplacements

L'algorithme de décision de l'an dernier, qui s'inspire du modèle d'agent de Jones [8], a été repris. Selon le modèle de Jones, chaque agent possède trois capteurs mesurant l'intensité des signaux devant lui à sa gauche, devant lui en face et devant lui à sa droite, ce qui compose la perception de l'agent.

L'algorithme implémentant ce modèle permet alors à l'agent de choisir la direction dans laquelle avancer (à gauche, tout droit, ou à droite), selon les signaux émis par les feux et par les agents qui recrutent, et selon les mots target et agent qui codent l'attirance ou le repoussement pour les signaux de feux et de recrutement d'agents.

Quant au flocking, nous nous sommes contenté.es du modèle de Flocking [9] de Netlogo [10] qui implémente le modèle de Reynolds [6] mentionné dans la première partie. Si l'agent perçoit au moins un autre agent dans son rayon de vision, il regarde d'abord si la distance qui le sépare de son voisin le plus proche est insuffisante. Si c'est le cas, il change de direction dans le but de s'éloigner de lui. Ensuite, l'agent s'aligne avec les voisins compris dans son rayon de vision ; c'est-à-dire qu'il prend la direction moyenne de ses voisins. Enfin, l'agent ajuste sa direction pour se rapprocher de ses voisins et assurer une bonne cohésion.

Un POMDP et de l'apprentissage par renforcement pour les jeux de langage

Le début de l'utilisation de Language Games ou jeux de langage dans les systèmes multi-agents remonte à 1995 dans l'article "A Self-Organizing Spatial Vocabulary" de Steels [11]. Dans notre modèle, nous reprenons le même jeu de langage particulier utilisé l'an passé qu'est le jeu de dénomination ou Naming Game, où les agents s'échangent des mots représentant des concepts dans l'environnement. Ces mots évoluent au gré des perceptions des agents et des parties de jeux, qui sont pour certaines biaisées selon le succès des agents ou de la nouveauté des mots.

Ainsi, les comportements individuels des agents évoluent selon des paramètres dont les valeurs sont définies par des mots. Chaque agent possède ainsi ses propres valeurs de mots qui codent les concepts suivants : son attirance pour les signaux des feux, son attirance pour les signaux de recrutement, et son estimation du nombre minimal d'agents pour éteindre un feu.

La quasi totalité de la formalisation mathématique des jeux de langage de l'an passé avec des processus décisionnels de Markov partiellement observables [1] convient toujours pour notre propre modèle. Les agents sont à nouveau en observabilité partielle, puisqu'ils ne connaissent pas entièrement les états du monde : les positions des autres agents, les positions des feux, le nombre minimal d'agents pour éteindre un feu, ... leur sont inconnus. Nous n'allons donc expliquer que le changement opéré vis-à-vis du critère de performance des agents. Celui-ci est maintenant la somme de deux sous-critères : la performance pour trouver et éteindre des feux, et la performance pour se déplacer en groupe de flocking de taille égale au mot estimant le nombre minimal d'agents pour éteindre un feu. Ces sous-critères sont chacun la somme

des récompenses obtenues au cours du temps lors de la bonne réalisation des actions correspondantes. Étant donné que l'objectif final des agents est d'éteindre les feux et non de constituer des groupes de flocking de taille suffisante, bien que cette capacité ait peut-être une influence positive sur la performance pour atteindre l'objectif final, les récompenses données pour le flocking sont plutôt considérées comme des *pseudo-récompenses* pour indiquer à l'agent qu'il progresse dans sa tâche. Selon Russell et Norvig [12], ces récompenses intermédiaires aident à résoudre le problème de l'assignement de crédit (*credit assignment problem*) dans l'apprentissage par renforcement, où un agent éprouve des difficultés pour savoir laquelle ou lesquelles de ses actions passées lui ont permis d'atteindre son objectif final.

Un mécanisme d'apprentissage par renforcement est en effet mis en place dans le modèle. Les agents agissent dans le monde et tentent de renforcer (ou non) leurs comportements de manière positive (avec une récompense) ou négative (avec une punition). Les agents qui ont un biais de succès n'échangent qu'avec des agents dont la performance globale est meilleure que la leur, et ce dans l'optique de maximiser la somme des récompenses reçues dans le temps. Cela a potentiellement des répercussions bénéfiques sur la performance des autres agents non biaisés ou possédant un biais de nouveauté. Comme les agents avec un biais de succès auront tendance à avoir de meilleurs mots, ils pourront les partager avec les autres agents durant les parties de jeux de langage, qui à leur tour verront leur performance augmenter.

Une tentative d'application réaliste du modèle pour l'extinction de feux

Une fois le modèle formalisé, nous l'avons instancié sur NetLogo. La tâche choisie pour éprouver notre modèle est donc la recherche de cibles, qui est classique dans le domaine de la robotique en essaim et a été utilisée par le groupe de l'année dernière. Nous avons créé la tâche spécifique et l'environnement de NetLogo en nous inspirant des feux de forêts.

Présentation de la tâche

La tâche choisie se déroule dans un environnement fermé de forme circulaire dans lequel circulent *n_agents* agents et où sont disposées *n_targets* cibles de feux. L'objectif (implicite) des agents est d'éteindre tous les feux dans l'environnement. Au début de chaque simulation, les agents sont placés de manière aléatoire dans une zone rectangulaire en bas de l'environnement circulaire, afin qu'ils se mettent rapidement en un seul groupe de flocking. En outre, cette disposition est plus réaliste que de les éparpiller dans la forêt, étant donné que la mise en place d'un essaim de robots dans la réalité se ferait à partir d'un point de départ similaire à tous les agents. En reprenant

les Language Games de l'équipe précédente, les concepts ici définis sont les cibles, les agents en rôle de recrutement, et le nombre minimal d'agents pour éteindre un feu (c = {cible, agent, min_agents}). Les cibles ainsi que les agents en rôle de recrutement émettent un signal qui se diffuse à une vitesse *diff* et s'évapore à une vitesse *evap*. Les agents ont trois capteurs positionnés devant eux pour la détection des intensités des signaux. En outre, toutes les cibles ont une même quantité de points de vie *lp*. L'intensité du signal des cibles reflète leur niveau de vie : plus un feu a de points de vie, plus il émettra un signal fort. Au cours du temps, les feux gagnent en intensité de signal et donc en points de vie pour simuler leur propagation. A un pas de temps donné, quand il y a suffisamment d'agents en rôle de destruction près d'un feu, celui-ci perd autant de points de vie qu'il y a d'agents en train de le détruire. Un feu s'éteint et disparaît de l'environnement quand ses points de vie tombent à 0.

Les feux ayant un nombre requis d'agents (*min_ agents*) pour s'éteindre, nous avons émis l'hypothèse que des groupes de flocking pourraient accélérer la résolution de la tâche. Ainsi, les agents en flocking reçoivent une récompense de 1 lorsque leur nombre de voisins est environ égale (avec un écart-type de 1) à leur mot *min_ agents* car ils pensent être suffisamment d'agents pour éteindre un feu ; au contraire, ils reçoivent une punition de -1 lorsqu'ils estiment être trop ou pas assez dans le groupe. Afin de comparer l'utilité du flocking dans la résolution de la tâche, trois types de déplacements exploratoires ont été distingués : sans flocking, avec un flocking "basique" (modèle Flocking de base dans Netlogo) et avec un flocking "biaisé" (apprentissage du mot *min_ agents*).

Lorsqu'un agent trouve un feu, il gagne une récompense de 1. S'il parvient à éteindre un feu, il reçoit une récompense de 100.

Enfin, des rôles ont été ajoutés afin d'organiser l'essaim et de faciliter le recrutement pour éteindre les feux. Pour rappel, trois rôles ont été définis :

- Exploration : l'agent parcourt l'environnement à la recherche de feux, en se déplaçant ou non en flocking ;
- Recrutement : lorsqu'il a trouvé un feu mais qu'il n'y a pas assez d'agents pour l'éteindre, l'agent émet un signal de recrutement pendant 20 pas de temps afin d'attirer d'autres agents ;
- Destruction : lorsque l'agent perçoit une cible, il essaie de l'éteindre pendant 20 pas de temps.

Les rôles ne sont pas fixes, donc chaque agent peut prendre différents rôles consécutivement au fur et à mesure de la simulation. Les probabilités du diagramme des transitions entre états (cf. figure 1) sont les suivantes : P(e | d) = 0.32 et P(e | r) = 0.1 . Il y a donc environ une chance sur

trois de passer du rôle de destruction au rôle d'exploration, et une chance sur dix de passer du rôle de recrutement au rôle d'exploration.
Les agents ne reçoivent pas de récompense spécifique lorsqu'ils entrent dans un rôle ou un autre, mais ils apprennent à percevoir et à suivre le signal des recruteurs.

Expériences

Plusieurs combinaisons de paramètres, présentées dans le tableau 1, ont été testées 20 fois chacune afin d'évaluer notre modèle. Les résultats ont ensuite été moyennés sur ces 20 simulations pour tirer une tendance fiable de chaque combinaison. Le modèle a enfin été évalué selon le temps mis pour éteindre tous les feux, l'évolution des mots dans le temps, la différence entre le mot *min_ agents* estimé final et la vraie valeur du paramètre *min_ agents* de l'interface. Le nombre d'itérations maximal est fixé à 5000 afin de limiter le temps mis par les simulations (qui se compte déjà en plusieurs heures) et la taille des fichiers. Ayant des machines limitées en capacités, il fallait aussi prendre en compte la mémoire requise par Python pour traiter toutes les données accumulées (6 Go de fichiers csv). L'interface Netlogo que nous avons réalisé pour la tâche est visible en figure 2.

Paramètre	**Valeurs testées**	**Description**
n-agents	{25, 50, 75, 100}	Nombre d'agents
n-targets	{1, 5, 10, 20}	Nombre de feux
forest-distance	35	Taille de la forêt
fires-disposition	{disp 1, disp 4, disp 7}	Disposition des feux
with-language-games	{true, false}	Avec ou sans jeux de langage
game-frequency	0.4	Fréquence de déclenchement de parties de jeux de langage
f-success	0.5	Fréquence du biais de succès
f-novelty	0.4	Fréquence du biais de nouveauté
mutation-rate	0.1	Probabilité de mutation d'un mot
min-agents	{1, 5, 10, 15}	Nombre minimal d'agents pour éteindre un feu
do-flocking	{deactivate, basic, biased}	Type de déplacement
roles-probabilities	fixed	Type de probabilités de transitions entre les rôles

TABLE 1 – Tableau résumant les paramètres utilisés lors des expériences.

Concernant les valeurs fixées de *game-frequency*, *f-success*, *f-novelty* et mutation-rate, nous avons gardé les valeurs utilisées par le groupe de l'année dernière afin de pouvoir éventuellement comparer nos résultats aux leurs.

Présentation des résultats

Les effets de quatre variables principales ont été étudiés pour évaluer le modèle : le type de déplacement (sans flocking, flocking basique et flocking biaisé), le nombre de *min_ agents*, avec ou sans Language Games et un score correspondant à la densité. En effet, afin d'évaluer d'un seul coup le nombre d'agents par rapport au nombre de cibles, nous avons utilisé le résultat de *n-agents / n-targets*. Ce score nous donne le nombre d'agents qui peuvent

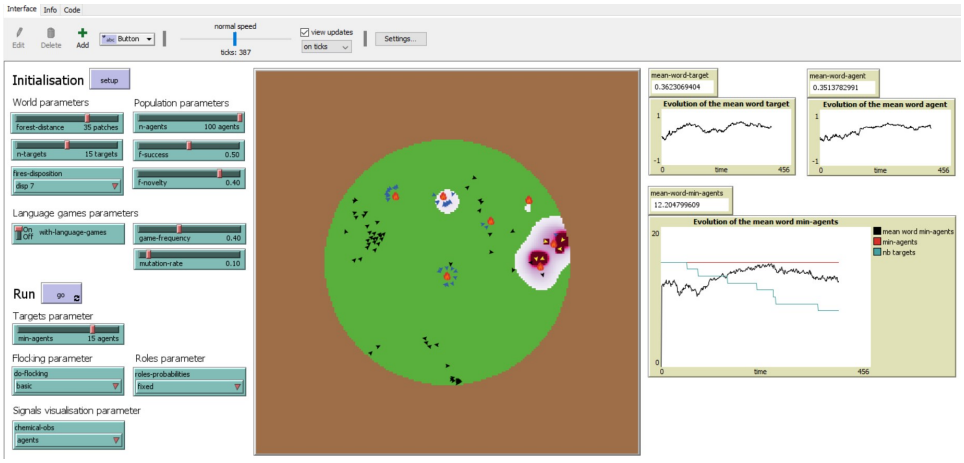

FIGURE 2 – Interface de la tâche simulée sur Netlogo. Les agents noirs sont en rôle d'exploration, les agent bleus sont en rôle de destruction, et les agents jaunes sont en rôle de recrutement. Les tâches dégradées en violet représentent l'intensité des signaux de recrutement.

s'occuper de la destruction d'une cible quand les agents sont répartis de manière égale sur l'ensemble des cibles. Avec nos expériences, ce score va de 1,25 agents pour un feu (dans le cas de 25 agents et 20 cibles) à 100 agents pour un feu (dans le cas de 100 agents et une cible).

Efficacité du modèle selon la valeur *min_ agents*

Etant donné que les expériences sont stoppées à maximum 5000 itérations, les résultats ayant 5000 itérations représentent les cas où les agents n'ont pas réussi à éteindre tous les feux et donc potentiellement les combinaisons les moins efficaces. Dans un premier temps, il est donc intéressant de visualiser quels sont ces cas et aussi quelles sont les combinaisons de paramètres les plus efficaces. La représentation graphique ci-dessous (cf. figure 3) représente donc le temps mis pour éteindre toutes les cibles en fonction de la densité du nombre d'agents par rapport au nombre de cibles en abscisse et ce pour les différentes valeurs de *min_ agents*.

Sur ces premier résultats, les expériences avec le paramètre *min_ agents* à 1 ou à 5 sont celles où le temps pour trouver toutes les cibles est le plus faible, et ce quelque soit la densité. De plus, les densités les plus faibles (jusqu'à environ 5 agents/cibles) sont celles où le temps mis est le plus élevé pour toutes les valeurs de *min_ agents*. Nous pouvons observer des pics, notamment pour une densité de 25 agents/cible ou 5 agents/cible. Les pics à une densité de 5 s'expliquent par le fait que cette densité regroupe les combinaisons 100 agents

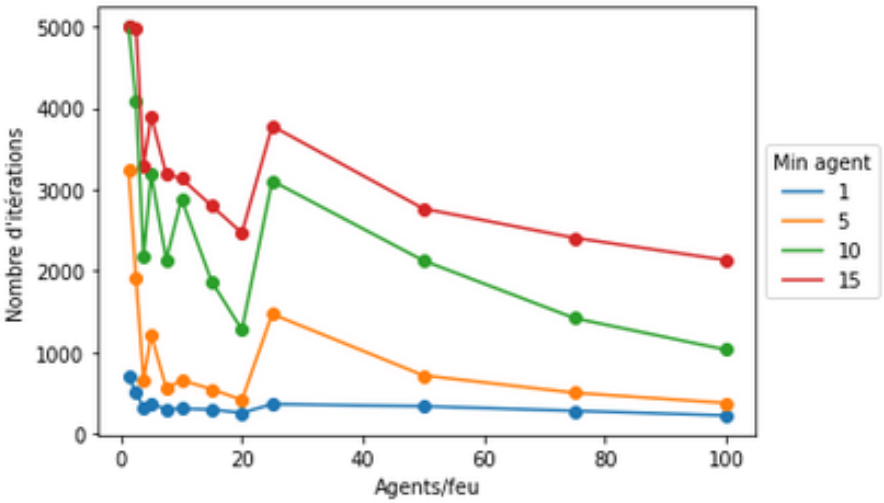

FIGURE 3 – Graphe du temps mis pour éteindre tous les feux en fonction de la densité du nombre d'agents rapport au nombre de cibles et de la valeur de *min_ agents*

et 20 cibles, 50 agents et 10 cibles, 25 agents et 5 cibles, et il est possible que la combinaison avec 20 cibles ait pris beaucoup de temps, ce qui "tire" la courbe vers le haut. Les pics à une densité de 25 signifient qu'il y avait 25 agents et une cible. Il semblerait donc qu'une population de 25 agents soit assez lente pour performer la tâche, peu importe les autres paramètres.

Efficacité du modèle selon le type de déplacement

Après avoir évalué l'efficacité avec différentes valeurs de min-agents indépendamment, nous nous sommes intéressé.es aux différents types de déplacement, avec et sans language games. Au calcul de la densité, nous avons soustrait *min_ agents*. Plus le résultat de ce nouveau score est haut, plus il y a suffisamment d'agents pour éteindre les feux et donc les agents ont plus de chances de réussir. Au contraire, lorsque ce score est faible, il y aura peu d'agents en réserve pour aider les nombreux autres à éteindre les feux ; autrement dit, il y aura besoin de beaucoup d'agents pour éteindre les feux par rapport au nombre total qu'ils sont dans l'essaim. Lorsque le score est négatif, cela signifie qu'il y a trop peu d'agents pour réussir à éteindre tous les feux en même temps. Ici, les scores prennent donc en compte le nombre de *min_ agents*. Cependant, avec les premiers résultats, nous avons vu que le modèle n'est pas efficace avec des valeurs de *min_ agents* élevées. Nous avons donc fait deux graphiques, un premier en laissant toutes les valeurs de *min_ agents* (cf. figure 4) et un second en laissant uniquement les *min_ agents* à 1 et 5 (cf. figure 5). Le cas du flocking biaisé sans language games n'est

pas représenté étant donné que le flocking biaisé a besoin de language games pour fonctionner.

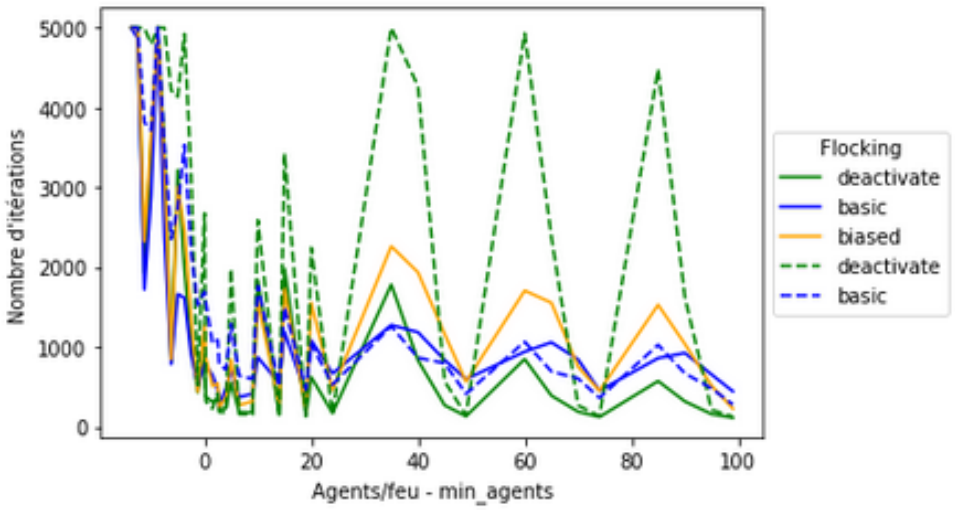

FIGURE 4 – Graphe du temps mis pour éteindre tous les feux en fonction du score (*n-agents*/*n-targets*) - *min-agents* (avec toutes les valeurs de *min-agents*), du type de déplacement (*deactivate* pour sans flocking, *basic* pour flocking basique et *biased* pour flocking biaisé) et avec (traits pleins) ou sans (tirets) *language games*.

Sur la première figure, nous observons de nombreux pics, notamment pour les valeurs en-dessous de 20, puis à 35, 60 et 85. Lorsque l'on enlève les *min_ agents* 10 et 15, ces pics ne sont plus présents et les courbes sont plus basses dans l'ensemble. Nous en concluons que peu importe le type de déplacement et la présence ou non de Language Games, des valeurs élevées de *min_ agents* comme 10 et 15 entraînent directement un temps élevé d'extinction totale des feux. De plus, pour des valeurs du score négatives, les temps moyens pour les différents types de flocking avec ou sans language games sont tous assez voire beaucoup élevés. De manière générale, les cas où le flocking est désactivé, peu importe s'il y a ou non des language games, sont les cas où les cibles sont détruites le plus rapidement. L'exception est lorsqu'il y a 10 ou 15 *min_ agents*, où le déplacement sans flocking et sans language games rend la tâche très longue, alors que le déplacement sans flocking mais avec language games impacte moins le temps mis. Les agents en flocking basique semblent aussi bien voire même un peu mieux se débrouiller que ceux en flocking biaisé.

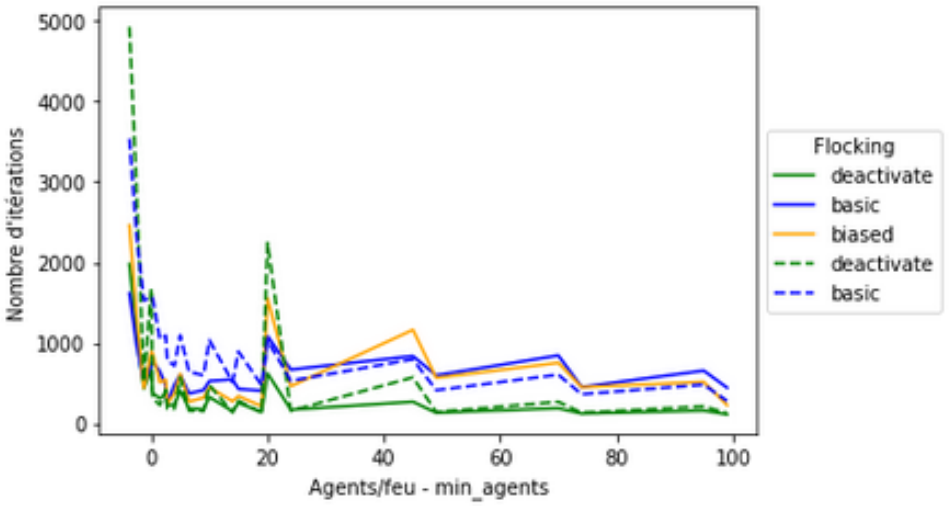

FIGURE 5 – Graphe du temps mis pour éteindre tous les feux en fonction du score (*n-agents*/*n-targets*) - *min-agents* (avec seulement les valeurs 1 et 5 de *min-agents*), du type de déplacement (*deactivate* pour sans flocking, *basic* pour flocking basique et *biased* pour flocking biaisé) et avec (traits pleins) ou sans (tirets) *language games*.

Évolution des mots dans le temps

Nous nous intéressons maintenant à l'évolution du mot *min_ agents* au cours du temps. Etant donné que les expériences n'ont pas le même nombre d'itérations et que l'étendue du nombre d'itérations entre les expériences peut être grande, nous avons séparé les résultats en fonction du nombre de *min_ agents* et du type de déplacement.
Sur ces résultats (cf. figure 6), sans flocking ou avec un flocking basique, l'évolution du mot *min_ agents* est plutôt similaire pour les valeurs réelles 10 et 15 du nombre d'agents minimal nécessaire pour éteindre un feu : le mot est initialisé à environ 10 au départ, puis ne change pas au cours du temps. Seule pour une valeur réelle de *min_ agents* égale à 1, la valeur du mot tend à augmenter au cours du temps, peu importe le type de déplacement. Ce résultat est étonnant étant donné que pour *min_ agents* = 1, les agents devraient apprendre qu'il est utile d'avoir un nombre réduit d'agents dans des groupes de flock pour éteindre les feux et non pas un nombre élevé. Toutefois, ce mauvais apprentissage peut s'expliquer par le fait que la tâche se finit rapidement lorsque *min_ agents* vaut 1, et que les agents n'ont donc peut-être pas assez de temps pour apprendre qu'il n'ont pas besoin d'être beaucoup. Avec le flocking biaisé, le mot *min_ agents* tend à baisser puis à ré-augmenter pour des vraies valeurs entre 5 et 15.

Le second mot a avoir été étudié dans nos expériences est le mot

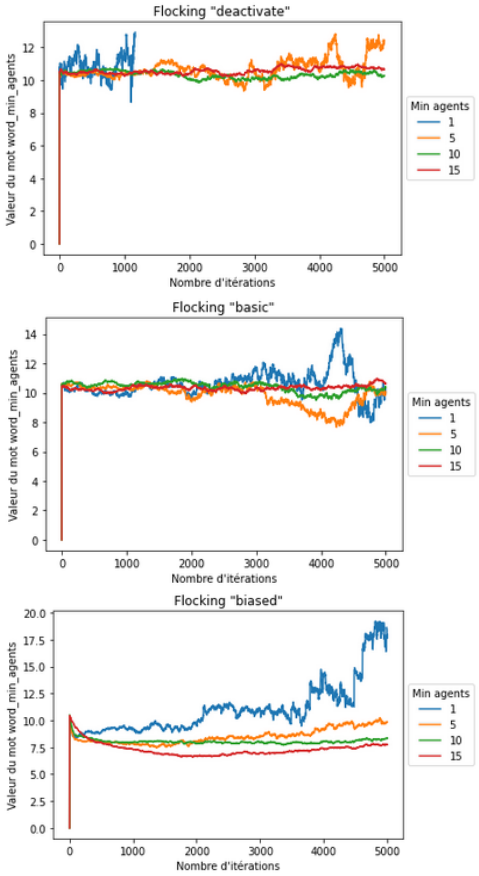

FIGURE 6 – Graphes de l'évolution dans le temps du mot *min_ agents* en fonction des valeurs fixées du paramètre *min_ agents* et du type de déplacement

correspondant au signal d'un agent recruteur (*word_ agents*).
Les graphiques (cf. figure 7) montrent que les agents ont du mal à apprendre et à être attirés aux signaux de recrutement. Dans toutes les combinaisons de paramètres, l'évolution de leur mot *word_ agents* est généralement soit instable, soit constante autour de 0. La même observation d'instabilité dans tous les cas a été faite pour le mot *target*.

Récapitulatif et interprétation des résultats

Les expériences réalisées ont permis d'évaluer l'efficacité du modèle intégrant les rôles et le flocking dans la tâche de recherche et destruction de cibles. Plusieurs paramètres ont été utilisés, notamment le nombre d'agents nécessaires à la destruction d'une cible et le type de déplacement. Une comparaison avec/sans language games a aussi été réalisée.

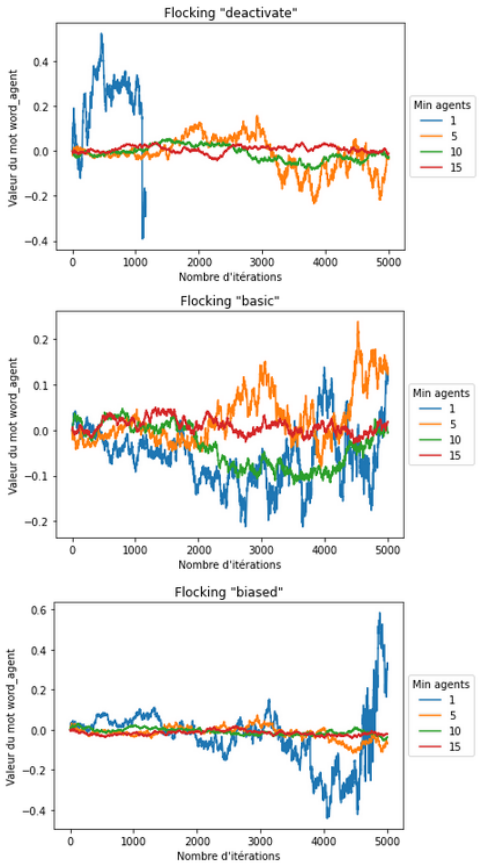

FIGURE 7 – Graphes de l'évolution dans le temps du mot *word_ agent* en fonction des valeurs fixées du paramètre *min_ agent* et du type de déplacement

D'après les résultats obtenus, peu importe la densité d'agents par feu, le nombre d'agents requis *min_ agents* pour éteindre un feu joue beaucoup. En effet, les résultats tendent à montrer que lorsqu'un feu a besoin d'un grand nombre d'agents pour s'éteindre, les agents mettent plus de temps en moyenne à accomplir la tâche. Cependant, quel que soit le nombre d'agents requis pour éteindre un feu, la valeur du mot *min_ agents* au sein de l'essaim ne change finalement pas beaucoup et ne semble pas tendre vers sa valeur théorique. Les agents ne semblent donc pas apprendre le nombre optimal qu'ils doivent être pour éteindre les feux.

Quant aux mots *min_ agents* et *target*, l'apprentissage correct d'attraction ne s'effectue pas dans notre tâche car il reste instable, alors qu'un apprentissage tendant vers des valeurs positives attractives s'établissait dans la tâche de l'an dernier. Leur apprentissage est peut-être affecté négativement par notre distribution de récompenses et de punitions.

Le type de déplacement a aussi été évalué avec ou sans language games. Le cas le plus efficace est lorsqu'il n'y a pas de flocking. Le flocking tel que nous l'avons implémenté, qu'il soit biaisé ou non avec des language games, ne se montre donc pas utile dans notre tâche de recherche de cibles. Deux raisons peuvent expliquer ce résultat : soit le flocking est effectivement inefficace et il vaut mieux laisser les agents explorer librement l'environnement, soit nous avons mal implémenté cette caractéristique. Lorsque les agents explorent librement l'environnement sans faire de flocking, il se peut qu'ils couvrent finalement une surface plus grande et qu'ils aient donc plus de chances de tomber sur une cible. Ils entrent alors dans le rôle de recruteur pour émettre un signal qui permet aux autres agents de les rejoindre sur la cible, et ce mécanisme pourrait donc être plus efficace que le flocking.

Conclusion

Notre projet, qui implémente des rôles, du flocking et des jeux de langage dans une tâche d'extinction de feux simulée sur Netlogo, s'est montré pertinent sous plusieurs angles. Déjà, l'exécution des simulations sur le modèle Netlogo a été grandement accélérée grâce à la ré-écriture en langage Netlogo du code python des language games écrit l'an passé. Notre modèle codé pourra donc servir de base à d'autres modèles qui nécessiteraient des jeux de langage.
Même si nous aurions espéré le contraire, les résultats observés nous incitent à dire que notre modèle est peu efficace avec l'ajout de déplacements en flocking biaisé ou non par les language games. Des modifications dans l'algorithme de flocking ou des paramètres des language games amélioreraient peut-être le modèle.

D'autres pistes pourraient être creusées. Bien que pertinente, la distinction des deux critères de performances (recherche et extinction des feux, et constitution d'un groupe de flocking de taille suffisante) n'a finalement pas été exploitée dans notre simulation. Il faudrait tester si des traitements différents de ces critères améliorerait l'apprentissage des mots. En outre, plusieurs répartitions ou distributions des récompenses et des pseudo-récompenses pourraient être évaluées afin de sélectionner celle qui donne les meilleurs résultats.
Pour notre application, nous avons choisi un recrutement statique des agents, mais il pourrait être mobile, avec des agents qui s'éloignent un peu de leur cible pour atteindre davantage d'agents avec leur signal de recrutement.

Abordons désormais quelques points regrettables de notre projet. Même si cela n'est guère contrôlable, il est dommage dans notre cas que les cours d'Agents intelligents et collectifs ainsi que de Phénomènes collectifs en biologie ne soient programmés qu'à partir du second semestre de M1 Sciences Cognitives. Effectivement, nous aurions pu acquérir plus rapidement et plus

en amont de la partie réalisation des connaissances utiles pour la compréhension d'articles et la formalisation dans le domaine des systèmes multi-agents.

Dans notre rapport bibliographique, nous avions défini comme l'un de nos objectifs du second semestre l'instanciation physique du modèle que nous aurions produit. Malheureusement, nos conditions de projet tutoré n'ont pas été favorables pour mener une conception de modèle, des essais en simulation, puis son instanciation robotique. Nous avons pris conscience ensemble avec nos encadrants que l'implémentation est d'une envergure importante pour des étudiants débutants dans ce domaine. La prise en main de robots ou de drones, de leurs logiciels et des langages de programmation employés, prendrait en effet beaucoup de temps. Cela nécessiterait sûrement aussi un travail de transposition du code entre celui de la simulation (qui pour nous est en langage Netlogo) et ceux des robots (tels que le langage Python). Il n'est pas non plus aisé de planifier suffisamment de moments où nous pourrions avoir accès aux dispositifs en étant encadrés. L'équipe de recherche SIMBIOT du LORIA ainsi que le TechLab de l'Ecole des Mines de Nancy possèdent des robots et des drones, mais les moments de disponibilité coïncident mal avec nos périodes dédiées au projet tutoré un mois sur deux. Dans le contexte de projet tutoré de notre Master, il conviendrait mieux que l'implémentation sur des robots ou des drones fasse l'objet d'un projet entier. Les étudiants partiraient d'un modèle déjà existant qu'ils ne modifieraient pas ou très peu, et se concentreraient uniquement sur la transposition physique de ce modèle. Ils pourraient éventuellement commencer la prise en main des robots et/ou des drones dès le premier semestre, en parallèle de recherches bibliographiques sur les besoins détaillés des essaims de robots et de drones, le fonctionnement des dispositifs spécifiques disponibles pour le projet, et des propositions préventives de solutions aux problèmes techniques éventuels.

Nous pouvons tout de même partager quelques recherches relatives à l'implémentation physique de notre tâche. Par exemple, il existe une méthode d'extinction acoustique de flammes, à l'aide d'un haut-parleur de très haute puissance envoyant des ondes d'une certaine forme. Les feux peuvent être détectés en utilisant des réseaux de neurones profonds [13]. Cela permettrait à des drones de pouvoir éteindre un feu à une distance suffisante pour ne pas être endommagés par la chaleur et les flammes.
Dans l'étude de Zhou et al. [14], un réseau de neurones de graphe est utilisé dans le cadre de tâches de perception visuelle avec plusieurs robots. Ce réseau a pour objectif d'augmenter la précision de la perception par inférence des robots individuels, ainsi que la résilience aux défaillances et aux perturbations des capteurs. Cela permet par exemple de mieux estimer la profondeur monoculaire, même avec des images corrompues (à cause de bruit, d'occlusion de caméra ou de panne de caméra).
Les forêts étant souvent des environnements non structurés, le temps pour

accomplir des tâches est long, mais il pourrait être réduit grâce à des essaims de robots hétérogènes qui auraient des types de déplacement différents, par exemple au sol, en l'air ou encore dans l'eau [15].
Le vol en flocking de drones resterait compliqué à implémenter, tout simplement parce que cela n'existe encore pas vraiment. Nous pourrions croire que les spectacles aériens rassemblant des centaines de drones montrent des drones appliquant un comportement de flocking, mais leurs déplacements sont en fait totalement chorégraphiés ; ils ne peuvent donc pas s'adapter à des imprévus. Même lorsque les déplacements de plusieurs drones ne sont pas prévus à l'avance, ils sont en tout cas soit pilotés à la main, soit plutôt seuls ou peu nombreux pour éviter les collisions.

Bibliographie

[1] Erwan Plantec, Lionel Aquilanti et Rafaelle Belorgey. *Vers l'adaptation de comportements par le biais de l'évolution culturelle pour des essaims de robots autonomes*. University works. LORIA, UMR 7503, Université de Lorraine, CNRS, Vandoeuvre-lès-Nancy, juill. 2021. URL : https://hal.inria.fr/hal-03280188.

[2] Marco Dorigo, Guy Theraulaz et Vito Trianni. « Swarm Robotics : Past, Present, and Future [Point of View] ». In : *Proceedings of the IEEE* 109.7 (2021), p. 1152-1165. DOI : 10.1109/JPROC.2021.3072740.

[3] Peter Dayan et Geoffrey E Hinton. « Feudal Reinforcement Learning ». In : *Advances in Neural Information Processing Systems*. Sous la dir. de S. Hanson, J. Cowan et C. Giles. T. 5. Morgan-Kaufmann, 1992. URL : https://proceedings.neurips.cc/paper/1992/file/d14220ee66aeec73c49038385428ec4c-Paper.pdf.

[4] Erol Sahin. « Swarm Robotics : From Sources of Inspiration to Domains of Application ». In : t. 3342. Jan. 2005, p. 10-20. ISBN : 978-3-540-24296-3. DOI : 10.1007/978-3-540-30552-1_2.

[5] Shervin Nouyan et al. « Teamwork in Self-Organized Robot Colonies ». In : *IEEE Transactions on Evolutionary Computation* 13.4 (2009), p. 695-711. DOI : 10.1109/TEVC.2008.2011746.

[6] Craig W. Reynolds. « Flocks, Herds and Schools : A Distributed Behavioral Model ». In : *SIGGRAPH Comput. Graph.* 21.4 (août 1987), p. 25-34. ISSN : 0097-8930. DOI : 10.1145/37402.37406.

[7] Valentin Lecheval et al. « Social conformity and propagation of information in collective U-turns of fish schools ». In : *Proceedings of the Royal Society B : Biological Sciences* 285.1877 (2018), p. 20180251. DOI : 10.1098/rspb.2018.0251.

[8] Jeff Jones. « Characteristics of Pattern Formation and Evolution in Approximations of Physarum Transport Networks ». In : *Artificial Life* 16.2 (avr. 2010), p. 127-153. ISSN : 1064-5462. DOI : 10.1162/artl.2010.16.2.16202.

[9] Uri Wilensky. *NetLogo Flocking model*. Netlogo Model. Center for Connected Learning et Computer-Based Modeling. Northwestern University, Evanston, IL, 1997. URL : http://ccl.northwestern.edu/netlogo/models/Flocking.

[10] Uri Wilensky. *NetLogo*. Computer Software. Center for Connected Learning et Computer-Based Modeling. Northwestern University, Evanston, IL, 1999. URL : http://ccl.northwestern.edu/netlogo/.

[11] Luc Steels. « A Self-Organizing Spatial Vocabulary ». In : *Artificial Life* 2.3 (avr. 1995), p. 319-332. ISSN : 1064-5462. DOI : 10.1162/artl.1995.2.3.319.

[12] Peter Norvig Stuart Russell. « Reinforcement Learning ». In : *Artificial Intelligence : A Modern Approach, 4th ed.* Sous la dir. de Pearson. T. 1. Morgan-Kaufmann, 2021. Chap. 23.4.4, p. 858.

[13] Jacek Wilk-Jakubowski et al. « Control of Acoustic Extinguisher with Deep Neural Networks for Fire Detection ». In : *Elektronika ir Elektrotechnika* 28 (fév. 2022), p. 52-59. DOI : 10.5755/j02.eie.24744.

[14] Yang Zhou et al. « Multi-Robot Collaborative Perception With Graph Neural Networks ». In : *IEEE Robotics and Automation Letters* 7.2 (avr. 2022), p. 2289-2296. DOI : 10.1109/lra.2022.3141661.

[15] Luiz F. P. Oliveira, António P. Moreira et Manuel F. Silva. « Advances in Forest Robotics : A State-of-the-Art Survey ». In : *Robotics* 10.2 (2021). DOI : 10.3390/robotics10020053.