Émergence de comportements collectifs basée sur l'apprentissage progressif individuel

François Suro[1], Jacques Ferber[1], Tiberiu Stratulat[1], Fabien Michel[1]

[1] LIRMM, Université de Montpellier, CNRS, Montpellier, France

suro@lirmm.fr, ferber@lirmm.fr, stratulat@lirmm.fr, fmichel@lirmm.fr

Résumé

Dans la perspective d'un développement ouvert et continu il est crucial qu'un comportement final souhaité à un instant donné puisse être un élément pour la création future de comportements plus complexes, dont le but ne peut être anticipé. L'acquisition de nouveaux comportements est déclenchée par l'apparition de nouveaux besoins et problèmes résultants de l'interaction avec l'environnement, mais aussi à travers l'interaction avec d'autres agents.

Dans cet article nous soumettons l'architecture développementale MIND à un problème de coordination multi-agents. A partir de comportements établis individuellement, nous ferons apprendre à un groupe d'agents, au moyen d'un curriculum établi par un instructeur humain, un comportement collectif, ainsi que l'utilisation alternante de comportements individuels et collectifs afin d'accomplir un but commun.

Mots-clés

Robotique développementale, Apprentissage agent, Architecture modulaire, Architecture hiérarchique, Apprentissage par Curriculum, Coordination Multi-Agents.

Abstract

In the perspective of open ended and continuous development it is crucial that a final desired behaviour at a given moment can be the starting point for the future creation of more complex behaviours, the purpose of which cannot be anticipated. The acquisition of new behaviours is triggered by the rise of new needs and problems resulting from interaction with the environment, but also through interaction with other agents.

In this paper we subject the MIND developmental architecture to a problem of multi-agents coordination. Starting from individually established behaviors, we will teach a group of agents, through a curriculum established by a human instructor, a collective behavior, as well as the alternating use of individual and collective behaviors in order to accomplish a common goal.

Keywords

Developmental Robotics, Agent Learning, Modular Architecture, Hierarchical Architecture, Curriculum-Based Learning, Multi-Agents Coordination.

1 Introduction

Inspirée par la théorie de Jean Piaget [29][30] sur le développement cognitif chez l'homme, la recherche sur la robotique développementale [21] vise l'émergence d'une intelligence artificielle générale au moyen d'un processus progressif qui accumule des compétences d'une complexité croissante.

Ce processus dynamique de schémas de coordination, depuis les schémas sensorimoteurs jusqu'aux opérations de niveau abstrait, est depuis longtemps compris comme une structure hiérarchique [23]. L'organisation, l'émergence et la régulation d'une telle structure, avec la robustesse nécessaire pour un développement à long terme, reste une question fondamentale du domaine de la robotique développementale [27].

Dans nos travaux antérieurs nous avons proposé une solution à cette question d'organisation sous la forme de l'architecture MIND [35]. MIND (Modular Influence Network Design) est une architecture qui organise des modules de compétences (*skills*) en un réseau contrôlé par un mécanisme d'influence. MIND permet d'ajouter de nouveaux modules de compétences dont le rôle est de coordonner des modules plus simples pour obtenir un comportement complexe. Nous avons montré que grâce à cette architecture, un agent individuel est capable de transposer un curriculum établi par un instructeur en une hiérarchie de comportements correspondants.

Dans l'introduction au premier Workshop international sur la robotique épigénétique [38], un précurseur du domaine général de la robotique développementale, les auteurs soulignent l'importance des interactions sociales parmi les interactions menant au développement cognitif. Qu'il s'agisse du développement personnel, par instruction ou imitation, ou bien du développement d'une organisation sociale à travers les rôles individuels.

Dans la perspective de développer MIND comme architecture supportant le développement d'agents intelligents, nous souhaitons intégrer pleinement l'aspect social de l'apprentissage et des comportements résultants, que ce soit dans la coordination d'essaim de robots (swarm robotics) ou la communication entre agents artificiels (ou biologiques).

Dans cet article, nous expérimentons avec l'architecture

MIND dans un contexte de coordination multi-agents. En nous basant sur une hiérarchie apprise individuellement, nous apprenons à un groupe d'agents de nouveaux comportements collectifs se basant sur des comportements individuels. Nous montrons que le principe de conception modulaire de MIND fournit une grande flexibilité dans la coordination de comportements individuels et collectifs.

Après un bref état de l'art à la section 2 nous présenterons l'architecture MIND dans la section 3, puis nous décrirons dans la section 4 le contexte expérimental et dans la section 5 les résultats de la tâche de fourragement multi-agents que nous souhaitons apprendre. En conclusion nous discuterons des futurs travaux multi-agents que nous réaliserons à l'aide de MIND.

2 Contexte

Apprentissage et agents Contrairement à d'autres domaines axés sur l'apprentissage, la robotique développementale vise à créer des systèmes artificiels avec des compétences qui vont au-delà de l'apprentissage d'une seule tâche. *"La recherche d'un système d'apprentissage multi-tâche flexible, autonome et ouvert est, par essence, une réinstanciation particulière de la recherche de longue date sur l'IA à usage général (general purpose AI)"* [21]. La robotique développementale, inspirée par l'être humain, vise à l'émergence de l'IA par un apprentissage progressif et continu sous les contraintes des agents incarnés.

La robotique développementale est un domaine de recherche hautement interdisciplinaire, s'inspirant de domaines autres que l'informatique et la robotique, telles que les neurosciences et la psychologie. Mais au sein même de l'informatique, la robotique développementale s'appuie sur de nombreuses disciplines telles que les algorithmes d'apprentissage, structures de données, exploration, communication entre systèmes, traitement du signal, etc.

Apprentissage par Curriculum Bien qu'utilisé implicitement depuis les année 90, l'apprentissage par Curriculum [2] s'est vraiment établi grâce aux avancées dans les domaines du reinforcement learning(RL) et du transfer learning [36]. Cette méthode d'apprentissage progressive très étudiée à ce jour dans le domaine du machine learning viens compléter les techniques de RL traditionnelles et permet de résoudre et d'accélérer l'apprentissage de problèmes autrement impossibles. En subdivisant une tâche d'apprentissage en sous-tâches différentes mais complémentaires (Source tasks), les différents signaux de Feedback (ou récompense) sont séparés et chaque sous-tâche est beaucoup plus simple à apprendre. Cette technique et a été appliquée avec succès à des problèmes de robotique et de jeux vidéo [25]. D'autre part l'utilisation d'un comportement simple comme point de départ pour apprendre un comportement complexe aide également dans l'aspect exploratoire de l'apprentissage, en se concentrant sur la complexité supplémentaire du nouvel environnement associé à la tâche.

La figure 1 montre un comportement complexe final

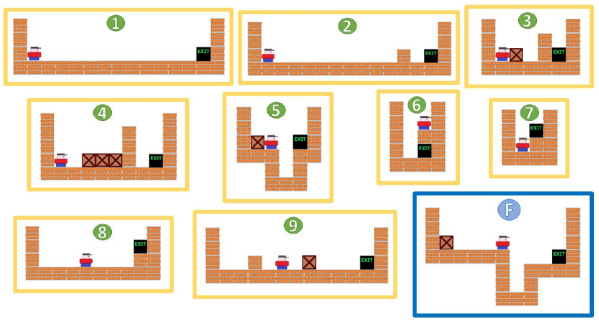

FIGURE 1 – La compétence finale F (encadré en bleu) est apprise en transférant toutes les compétences apprises précédemment, telles qu'atteindre la sortie(1), sauter sur un bloc(2), pousser un bloc(3) ...(extrait de [14])

mémorisé par un approximateur de fonction unique grâce à l'apprentissage par transfert des différentes tâches sources [28].

Bien que ces méthodes permettent d'apprendre par des moyens progressifs, leur portée est limitée à la tâche spécifique à accomplir. Ces travaux ne couvrent pas l'apprentissage continu, ouvert et cumulatif d'une variété de tâches, un défi au coeur de la robotique développementale qui se caractérise par l'acquisition progressive de compétences et de connaissances [27].

Coordination de compétences Afin de faciliter l'apprentissage de compétences diverses, se basant potentiellement sur des compétences de base communes, il serait intéressant de pouvoir assimiler le Curriculum sous forme de modules de compétences indépendants. La principale difficulté réside dans le fait de devoir coordonner des modules dont les sorties sont concurrentes (par exemple : plusieurs compétences affectant le déplacement d'un agent).

Des travaux antérieurs dans le domaine des systèmes de commande de robots tels que l'architecture de subsomption [3] offrent des solutions pour alterner l'utilisation de différentes compétences selon le contexte. Cette méthode de composition séquentielle et exclusive de compétences est la plus répandue, depuis le "world of blocks" de Minsky [23] jusqu'aux systèmes de contrôles avancés des architectures cognitives (SOAR [19], CLAIRON, ICARUS [20]), en passant par des méthodes telles que le Skill Chaining [18] qui utilisent une sequence d'actions très simple dont la transition est déclenchée par l'apparition d'un contexte particulier.

Une autre possibilité de composition est la composition simultanée, ou somme vectorielle. Les travaux sur les Boids [32] constituent un des premiers exemples de cette méthode de coordination entre des comportements concurrents. Les Boids sont des oiseaux virtuels conçus pour expérimenter le comportement de troupeau (*flocking*). Ils doivent ainsi satisfaire trois contraintes : la cohésion (rester proche du groupe), la séparation (garder une distance minimale avec les autres individus) et l'alignement (se diriger dans la même direction que le groupe). Trois compé-

tences motrices indépendantes sont en charge de satisfaire chacune de ces contraintes, et la compétence globale devient, comme le dit Reynolds, un problème "d'arbitration de comportements indépendants".

"Chaque comportement dit : "si J'étais aux commandes, J'accélérerai dans cette direction." [...] C'est au module de navigation du cerveau du Boid de collecter toutes les demandes d'accélération pertinentes et de déterminer ensuite une seule accélération désirable du point de vue du comportement. Il doit combiner, prioriser et arbitrer entre des impulsions potentiellement conflictuelles."[32]

Ce mécanisme est employé dans des architectures comme AuRA [1] ou Sat-Alt [33], ou encore récemment sur des problèmes de locomotion complexe pour un robot humanoïde à 54 dimensions [16]. On retrouve également ce principe dans le "Social Force Model" décrivant le comportement de piétons [17].

Organisation hiérarchique Un autre aspect important de l'accumulation de compétences indépendantes sous forme de modules est l'établissement d'une hiérarchie de contrôle entre ces modules.

L'architecture ICARUS [6] est une architecture teleoréactive [24, 26], utilisant une organisation hiérarchique de deux manières différentes. L'évaluation de l'état actuel se fait en évaluant une hiérarchie de percepts/concepts de bas en haut, réunissant des percepts en concepts de bas niveau, puis ces concepts de bas niveaux en concepts de haut niveau. L'exécution se fait en parcourant une hiérarchie de buts de haut en bas, le but de plus haut niveau est activé par les concepts correspondants et cherche à réaliser ses sous buts. Les sous buts possèdent eux mêmes des sous buts qui sont choisis en fonction de l'état de la hiérarchie de percepts/concepts. Une des particularités d'ICARUS est d'évaluer la prochaine action non seulement en fonction de l'état de la hiérarchie de percepts/concepts, mais aussi de sa position courante dans la hiérarchie de buts/sous-buts. Cette particularité donne à ICARUS une approche équilibrée entre un système purement réactif et un engagement fort dans les plans de haut niveau (*commitment*).

Une série de travaux qui réunissent une grande partie des éléments de la robotique développementale présentés jusqu'ici sont les travaux sur les techniques de Robot Shaping de Dorigo et Colombetti [9, 10]. Ici, les comportements s'apprennent par Curriculum et sont représentés comme des compétences indépendantes. Ces compétences sont ensuite combinées pour obtenir des comportements de plus haut niveau. L'accent est mis sur le processus d'apprentissage en relation avec la structure représentant la connaissance. Plusieurs architectures sont discutées, des hiérarchies monolithiques aux hiérarchies multi-niveaux. Des méthodes d'apprentissage et de récompense adaptées aux agents artificiels sont également proposées.

Afin de répondre aux besoins spécifiques de la robotique développementale nous avons proposé l'architecture MIND [35]. MIND (Modular Influence Network Design) est une architecture de contrôle pour agent artificiel conçue pour apprendre des comportements à partir d'un Curriculum établi par un instructeur humain. MIND encapsule les comportements dans des modules et les combine en une hiérarchie reflétant la nature modulaire et hiérarchique des tâches complexes, et permet la préservation et la réutilisation des compétences acquises. Les résultats présentés dans l'article [35], sur une tâche de collecte d'objets avec évitement d'obstacles, ont montré que l'architecture MIND a su transformer le Curriculum en une hiérarchie de skills avec à la fois coordination et exclusion mutuelle de skills. MIND a été en mesure de traiter les tâches sensori-motrices de bas niveau ainsi que les opérations complexes de coordination de skills, sans fournir aucune information a priori sur la nature de la tâche. Nous avons montré les avantages de cette conception modulaire, par exemple pour l'identification et le réentrainement spécifique d'un aspect du comportement sous-optimal. Enfin nous avons montré que cette modularité est essentielle dans le cadre d'un développement continu où les futures compétences à apprendre ne peuvent pas être anticipées, et bénéficieront d'une large bibliothèque de compétences de base disponibles pour être combinées.

Développement et interactions Un point sur lequel s'appuie la robotique développementale est l'idée d'incarnation, l'affirmation selon laquelle le fait d'avoir un corps qui sert de médiateur entre la perception et le comportement joue un rôle intégral dans l'émergence de la cognition humaine [22]. Cette incarnation soumet l'entité apprenante aux contraintes physiques élémentaires de l'environnement (sa réalité) et pousse à l'élaboration de méthodes de résolution fondées sur une approche locale. La condition de cette entité est donc celle d'un agent [11].

Dans les premiers travaux se revendiquant de la robotique développementale, encore appelée robotique épigénétique, l'importance des interactions sociales est mise en avant comme faisant partie des interactions menant au développement cognitif [38]. L'étude des Systèmes Multi-Agents a montré les bénéfices d'une vision orientée agent et des interactions sociales dans un système. Les travaux sur la modélisation de sociétés d'insectes [7, 31] ont montré que la coordination d'agents réactifs très simples peut aboutir à des comportements complexes, capables de résoudre des problèmes bien au-delà de la simple somme des capacités individuelles de chaque agent. En complexifiant l'échange de simple signaux en un échange de messages il est possible de passer d'une simple coordination réactive vers une collaboration mettant en oeuvre des rôles différents au sein d'un groupe, la mise en oeuvre de tels systèmes est étudiée dans le cadre de la collaboration dans une flottille d'AUV [5].

L'intérêt de se pencher sur la question de l'interaction sociale pour un agent développemental va au-delà du gain de productivité en rapport à une tâche à accomplir. Pour qu'un agent en développement puisse sortir de la phase expérimentale et entrer dans le monde réel, il devra être capable d'interagir socialement, à la fois avec d'autres agents du même type et avec une population hétérogène

d'agents, y compris biologiques.

L'interaction sociale est étroitement liée avec les représentations internes, dans une notion de soi par exemple, mais aussi dans la communication des intentions de résoudre les problèmes collectifs qui pourraient nécessiter l'acquisition de symboles de manière émergente afin de former un proto-langage [21, 37]. Les interactions sociales jouent également un rôle important dans le processus d'apprentissage, avec des algorithmes exploitant des aspects sociaux tels que l'imitation, ou l'interaction avec un instructeur humain par une forme de communication beaucoup plus naturelle que des algorithmes de programmation.

3 Modular Influence Network Design

MIND (Modular Influence Network Design) est une architecture hiérarchique de modules capables de coordonner des comportements (*Skills*) distincts pour accomplir une tâche complexe. Cette architecture permet d'accumuler continuellement de nouvelles compétences et d'utiliser la coordination des compétences antérieures pour maîtriser rapidement de nouvelles tâches d'une complexité croissante. Dans cette section nous reprenons les principes de base de l'architecture MIND [35].

3.1 Base skill, complex skill, et influence

Considérons un agent dont les informations sensorielles et les commandes motrices sont représentées comme un vecteur de réels, normalisés entre 0 et 1. On peut créer un module qui encapsule une fonction $f(x)$ qui prend en entrée le vecteur $V_I = [I_1, I_2, ..., I_k]$ et fournit en sortie le vecteur $V_O = [O_1, O_2, ..., O_2, ..., O_j]$. Cette fonction peut être implémentée comme une procédure de programmation, ou peut être un approximateur de fonction, un réseau neuronal, ou tout autre type de fonction qui associe deux vecteurs de nombres réels. Nous appellerons un tel module un *skill*, et le module dont le vecteur de sortie est utilisé directement comme commande moteur un *base skill*.

$$V_O = f(V_I) \tag{1}$$

Eq.1 : le vecteur d'entrée V_I est fournit à la fonction interne $f()$ du skill pour produire le vecteur de sortie V_O.

Il est possible de créer un unique base skill qui utilise toutes les entrées sensorielles et toutes les sorties motrices de l'agent pour apprendre à accomplir une tâche, même relativement complexe, chaque leçon du Curriculum étant mémorisée dans la même structure unique. Ce skill unique agrège toutes les différentes expériences dans sa propre fonction interne, sans qu'il soit possible de différencier ce qui a été enseigné.

Au lieu d'accomplir une tâche complexe au moyen d'un seul skill, nous accomplirons de nombreuses sous-tâches, potentiellement concurrentes, avec des base skills distincts. Chaque base skill n'associera que les entrées et sorties nécessaires à l'exécution de la tâche qui lui est associée.

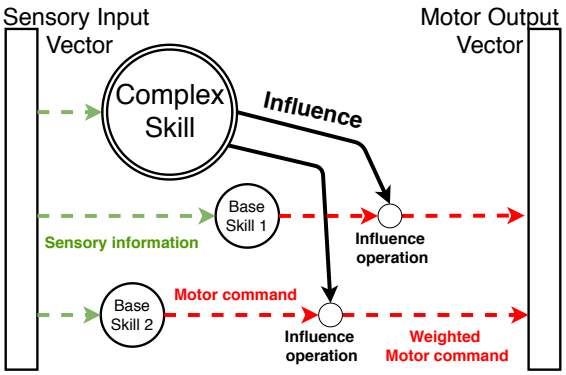

FIGURE 2 – Un *complex skill* influençant deux *base skills*

Pour accomplir la tâche complexe, nous allons ensuite créer un *complex skill* qui va coordonner plusieurs base skills, que nous appelons ses *subskills* (Fig.2). Un *complex skill* coordonne ses *subskills* en envoyant un signal appelé *influence* qui détermine le poids qu'un subskill aura sur l'action résultante, ce qui revient à déléguer à une ou plusieurs sous-capacités la résolution de la tâche courante.

Un Complex skill, comme n'importe quel skill, encapsule une fonction qui prend en entrée un vecteur de valeurs issues des capteurs V_I et sort un vecteur de nombres réels V_O, dont la sortie est dirigée vers ses subskills. Ce vecteur de sortie est appelé le vecteur d'influence V_{Infl} = $[Infl_1, Infl_2, ..., Infl_i]$, et ses éléments $Infl_x$ sont appelés *influences*.

Un complex skill peut avoir d'autres complex skills comme subskills, créant des hiérarchies de skills. Au sommet de la hiérarchie se trouve le *Master skill* dont la seule particularité est de recevoir une influence constante de 1.0, une impulsion permettant de lancer le mécanisme de calcul.

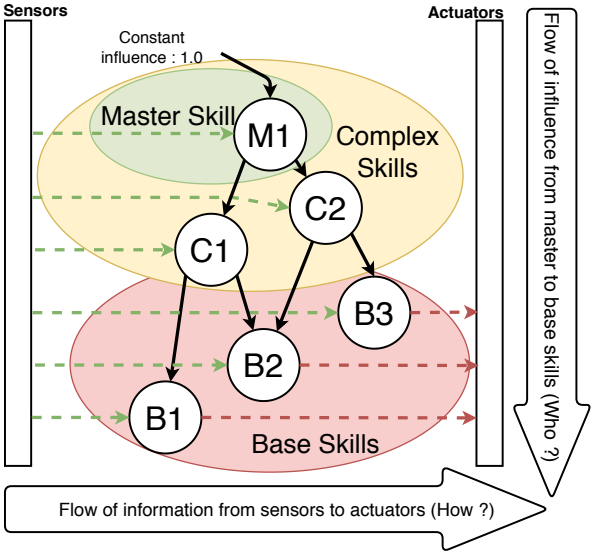

FIGURE 3 – Une hiérarchie de skills, le *master skill* influence des *complex skills* qui à leur tour influencent des *base skills*

Cette hiérarchie de skills forme un graphe orienté acyclique (Fig.3). L'influence s'exerce le long d'un axe vertical, du master skill jusqu'aux base skills, et détermine qui est responsable de l'action résultante. L'information des capteurs est disponible pour tous les skills de la hiérarchie et les commandes des moteurs sont envoyées par les base skills vers les actionneurs formant un flux d'information horizontal. Ce flux détermine comment l'action résultante va être exécutée.

3.2 Utilisation de l'influence pour déterminer les commandes moteurs

A partir du master skill, chaque complex skill calcule son vecteur de sortie V_O et multiplie chaque élément par la somme des influences reçues, formant le vecteur d'influence V_{Infl}. Le skill envoie alors chaque élément $Infl_x$ de V_{Infl} au subskill correspondant (Fig.4).

$$V_{Infl} = V_O * \sum_{x=1}^{i} Infl_x \tag{2}$$

Eq.2 : V_{Infl} le vecteur d'influence vers les subskills, $V_O = f(V_I)$ le vecteur de sortie de la fonction interne du skill, et $\sum_{x=1}^{i} Infl_x$ la somme de toutes les influences reçues par le skill (également noté $\Sigma Infl$).

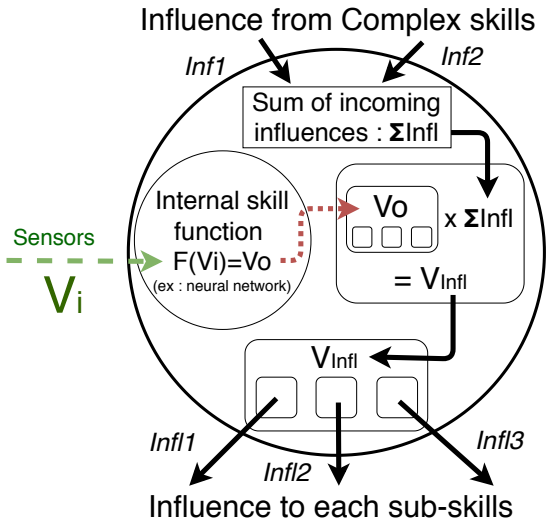

FIGURE 4 – Architecture interne d'un skill

Un base skill, comme un complex skill, calcule son vecteur de sortie et multiplie chaque élément par la somme des influences qu'il a reçues (Eq.2), formant le vecteur de commandes moteur $V_{Com} = [Com_1, Com_2, ..., Com_j]$. Le base skill envoie alors chaque élément Com_x du vecteur de commandes moteur au module moteur correspondant, ainsi que la somme des influences reçues par lui même $\Sigma Infl$.

Chaque module moteur calcule ensuite la commande correspondant à son actionneur à partir des commandes reçues par les skills concurrents au moyen d'une somme pondérée normalisée (Eq.3).

$$M = \frac{\sum_{x=1}^{l} Com_x}{\sum_{x=1}^{i} \Sigma Infl_x} \tag{3}$$

Eq.3 : M la commande de moteur résultante, x l'index d'un des l base skills qui envoie une commande de moteur, Com_x la commande de moteur pondérée pour ce module moteur depuis le base skill x, $\Sigma Infl_x$ la somme des influences du base skill x.

4 Éléments de l'expérimentation

L'environnement, les éléments de l'expérimentation et les techniques d'apprentissage utilisés sont les mêmes que ceux présentés dans l'article précédent [35]. Nous rappelons ici brièvement leurs caractéristiques essentielles et présentons les quelques fonctionnalités ajoutées.

Le robot utilisé dans notre simulation est composé de deux moteurs, un pour chaque roue, et d'une pince pour saisir l'objet cible. Il dispose également de 18 capteurs, détecteurs d'obstacles et d'orientation des objets à collecter. Il peut également générer une onde sinusoïdale qui peut être utilisée à la manière d'un capteur (par exemple, pour résoudre des situations de deadlock réactif).

Le robot est également équipé d'un émetteur de signal et d'un récepteur correspondant capable d'indiquer la présence, l'orientation et la distance d'un signal émis par un autre agent. Enfin le robot est capable de connaître la distance et l'orientation du robot le plus proche.

Les fonctions internes des skills sont principalement des perceptrons multicouche, dont la couche d'entrée correspond au vecteur d'entrée du skill et la couche de sortie à son vecteur de sortie. Selon le skill le nombre de couches cachées varie entre 2 et 3. Les fonctions de transfert des couches du réseau neuronal peuvent être sigmoïde ou ReLU. La couche de sortie utilise une fonction de transfert linéaire fixée entre 0 et 1 (clamp). Certains skills peuvent utiliser une topologie différente, par exemple un réseau neural convolutionel pour traiter un grand nombre d'entrées, ce qui réduit le temps de convergence de l'algorithme d'apprentissage.

Enfin, certains skills triviaux utilisent une procédure programmée comme fonction interne, la complexité de la tâche ne justifiant pas un apprentissage.

L'algorithme d'apprentissage est un algorithme génétique simple. Les poids des connexions du réseau neuronal sont ordonnés dans un vecteur, qui correspond au génome de l'individu du point de vue de l'algorithme génétique. Le génome sera croisé et muté en fonction du résultat de la fonction de fitness (d'adaptation) que l'environnement fournit en retour. L'algorithme génétique se prête bien à ce genre de tâches, l'évaluation ayant lieu à la fin du cycle de vie de l'individu simplifie les problèmes de récompense différée et ne nécessite pas de set d'entraînement comme dans le cas d'un algorithme de rétropropagation. L'inconvénient de l'algorithme génétique est son coût élevé en ressources dans l'évaluation de chaque génome. Toutefois,

l'évaluation est totalement indépendante et parallélisable et permet donc l'utilisation de solutions de calcul haute performance.

Notre implantation de l'algorithme génétique permet d'initialiser la population soit de manière aléatoire, soit en fournissant des individus déjà entraînés. Ceci permet de reprendre l'entraînement à partir d'un point donné pour optimiser un comportement, et complémente la technique d'optimisation dans le contexte final présenté dans l'article précédent (Scenario 2 [35]).

5 Fourragement multi-agents

Le problème des robots fourrageurs est un problème bien connu dans le domaine de l'intelligence artificielle et des systèmes multi-agents, que ce soit du point de vue de la modélisation de sociétés d'insectes [8] ou du développement de robots autonomes pour l'exploration spatiale [4]. Ce problème met en jeu des questions de résolution distribuée de problèmes et de communication inter-agents. Nous souhaitons apprendre une tâche de fourragement multi-agents en nous basant sur une hiérarchie MIND préexistante qui réalise la collecte d'un objet dans un environnent contenant des obstacles [35]. Nous souhaitons montrer qu'il est possible d'étendre les capacités d'un agent MIND, dans le sens développemental du terme, à de nouvelles tâches nécessitant un comportement réactif de coordination multi-agents. Ce comportement de fourragement multi-agents proche d'un comportement d'insecte reste dans un domaine purement réactif et ne requiert que l'échange de signaux simples.

5.1 Protocole

Un groupe de quatre agents est placé dans l'environnement. Chaque agent possède un capteur donnant l'orientation de l'objet, avec une portée de perception limitée, un émetteur de signaux et un récepteur donnant l'orientation du signal le plus proche. Les agents ont également un capteur qui donne l'orientation de l'agent le plus proche.

L'environnement contient des obstacles à éviter et une zone de dépôt pour les objets à collecter. Les objets sont placés proches les uns des autres dans l'environnement, en groupes de 8.

Les agents ne peuvent transporter qu'un seul objet à la fois et leur émetteur de signaux a une portée bien supérieure à celle de leur capteur d'orientation d'objet.

Étant donné que les agents ne possèdent aucune information sur leur environnement, le comportement optimal attendu consiste à chercher les objets à collecter là où aucun autre agent n'est présent afin d'obtenir la plus grande zone observée. Quand un agent trouve le groupe d'objets, il envoie un signal pour demander aux autres agents de venir.

Tous les agents de la simulation utilisent la même hiérarchie MIND, chacun possédant une instance traitant les valeurs de ses propres capteurs. La réponse globale durant l'évaluation du groupe d'agents utilisant le même

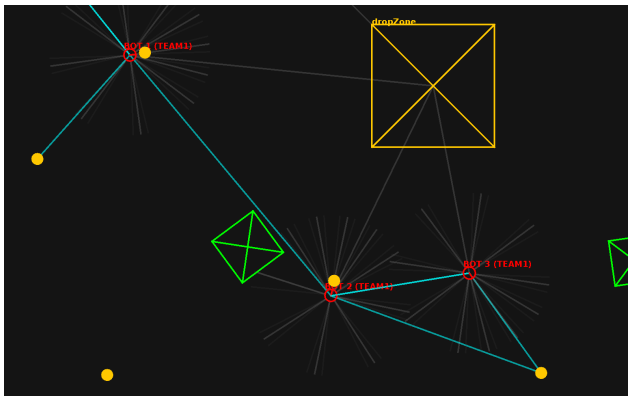

FIGURE 5 – l'environnement de simulation : en jaune la zone de dépôt et les objets à collecter, en vert les obstacles et en rouge les agents. Les différentes lignes représentent les informations des capteurs et des signaux.

génome pour la fonction interne à entraîner constitue le score de ce génome.

5.2 Hiérarchie

La hiérarchie finale que nous avons établie est montrée dans la figure 6. Les flèches vertes représentent les entrées, les rouges les sorties et les noires les relations d'influence. Les skills en pointillé utilisent une fonction interne programmée et les skills en bleu sont les skills qui nécessitent un apprentissage collectif.

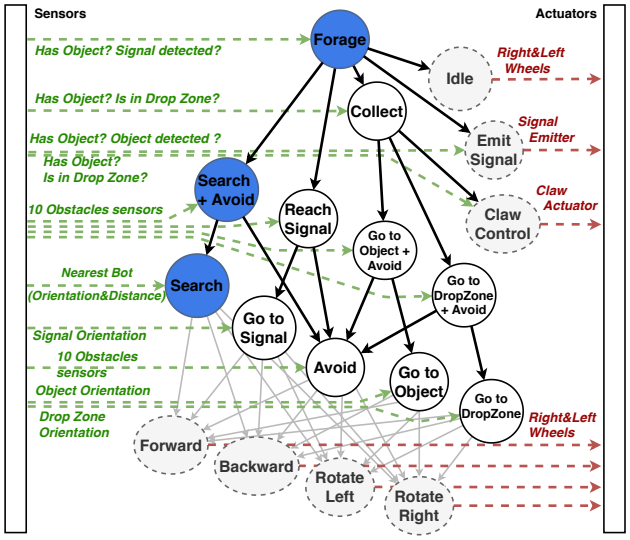

FIGURE 6 – L'architecture MIND pour la tâche de fourragement

Hiérarchie initiale Pour accomplir la tâche de fourragement, nous nous basons sur la hiérarchie *Collect* préexistante [35]. Cette sous hiérarchie a été apprise individuellement, la tâche de collecte étant accomplie par un seul agent.

Collect se base sur 2 complex skills *GoToObject+Avoid* et *GoToDropZone+Avoid* qui, respectivement, cherche un objet en évitant les obstacles et retourne à la zone de dépôt

en évitant les obstacles. Ces deux skills sont eux même basés sur 3 skills : GoToObject, GoToDropZone et Avoid qui, respectivement, dirige l'agent vers un objet, dirige l'agent vers la zone de dépôt, et permet à l'agent d'éviter les obstacles.

Un autre skill présent est *Reach Signal* qui combine *Avoid* et *GoToSignal* et permet à l'agent d'atteindre un signal émis par un autre agent en évitant les obstacles. Bien que ce skill mette en oeuvre les signaux, il n'a pas été appris dans un contexte multi agent, l'environnent d'apprentissage utilisé est capable de simuler l'émission d'un signal.

Skills programmés Certains skills ne sont pas appris mais programmés, leur fonction étant triviale et ne justifiant pas la mise en place d'un curriculum. C'est le cas de *Emit Signal* qui émet un signal dès qu'un objet est présent et *Claw Control* qui saisit l'objet quand il est à portée et le lâche quand l'agent est présent dans la zone de dépôt. Certain skills sont aveugles, c'est à dire qu'ils ne prennent aucune entrée capteur et fournissent une sortie constante : *Forward*, *Backward*, *Rotate Left*, *Rotate Right* et *Idle*.

Comportement multi-agents *Search+Avoid* et son sub-skill *Search* sont les skills réellement dépendants d'un fonctionnement multi-agents, et ne peuvent êtres appris que dans un contexte multi-agents. Un agent seul ne dispose d'aucune information qui puisse l'aider à établir un parcours de recherche efficace, d'aucune mémoire des lieux déjà visités ou d'un système de phéromones/marqueurs à déposer dans l'environnement. Il ne connaît pas la forme ou les dimensions de l'environnement et ne peut pas établir, par exemple, un parcours de balayage. Toutefois, en tant que groupe, les agents peuvent tirer parti du fait qu'ils ont plusieurs fois le champ de perception d'un agent individuel et utiliser les informations sur la position de chacun pour se coordonner en une formation qui couvrira la plus grande zone possible.

Master skill Enfin *Forage* coordonne *Search+Avoid* pour trouver les objets, *Emit Signal* pour appeler les autres agents, *Reach Signal* pour rejoindre un agent qui aurait trouvé les objets avant lui, et *Collect* pour collecter les objets trouvés. Comme l'agent ne possède pas de mémoire et de système de coordonnées pour décrire la position du groupe d'objet, *Forage* a la possibilité d'utiliser le skill *Idle* qui immobilise l'agent et permet ainsi d'attendre que d'autres agents arrivent. L'agent agit ainsi comme une balise marquant la position [15].

5.3 Résultats

Cette section décrit le comportement d'un agent sous la forme d'une succession d'étapes.

Les figures 7 et 8 montrent l'état de la hiérarchie MIND pour l'agent aux étapes 3 et 5. En rouge les actionneurs (placés proches des skills correspondants) et en bleu les 4 capteurs fournis au skill **Forage** : *NearFriendlyActive* présence d'un autre agent à porté de détection, *ListenerAActive* présence d'un signal à porté de détection, *ACTOBJA* présence d'un objet cible est à porté de détection, *SENSOBJ* indique le transport d'un objet. La hiérarchie correspond à celle présentée dans la figure 3, les skills sont en gris, en dessous du nom de chaque skill apparaît la somme de l'influence reçue par le skill. Enfin, les liens d'influence sont représentés par un dégradé allant de noir pour la valeur 0 au vert pour la valeur 1, le trait central représente la commande du skill seul et le trait extérieur représente l'influence effectivement transmise (voir Fig.4 et Eq.2).

Etape 1 : L'agent ne perçoit pas d'objets, ne reçoit pas de signal. Le master skill *Forage* active *Search+Avoid* avec une intensité maximale (une très faible valeur relative et donnée à *ReachSignal* mais ceci n'a aucun impact sur le comportement).

Etape 2 : L'agent perçoit un objet, le signal est activé, le skill *Collect* est activé en majorité, *Search+Avoid* reste partiellement actif mais ne perturbe pas le comportement de *Collect*. Dans cette phase *Collect* active la branche *GoToObject* qui permet d'atteindre l'objet.

Etape 3 : L'agent a ramassé l'objet, le capteur correspondant est actif. Dans cette phase *Collect* active la branche *GoToDropZone* qui permet de rejoindre la zone de dépôt. *Forage* active complètement *Collect* et désactive *Search+Avoid*. À cette étape les autres agents sont attirés par le signal émis.

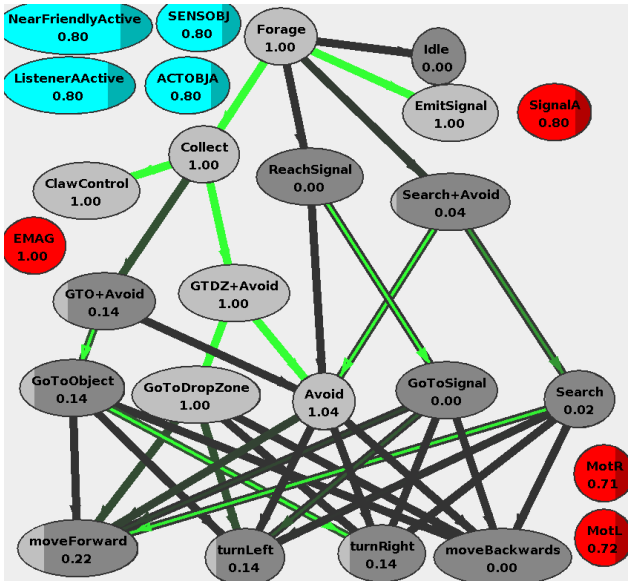

FIGURE 7 – Etape 3

Etape 4 : L'agent en route pour la zone de dépôt, quitte le champ de perception des objets, son signal s'éteint. Le changement des valeurs des capteurs influe sur la hiérarchie sans pour autant perturber le comportement de l'agent. Bien que notre agent ait cessé d'émettre le signal, un autre agent est maintenant à portée de perception des objets et émet le signal pour attirer les autres agents ne transportant pas encore d'objets.

Etape 5 : L'agent a déposé l'objet. *Collect* est désactivé, *Search+Avoid* et *ReachSignal* sont activés. Meme si *ReachSignal* est inférieur à *Search+Avoid*, on peut voir (fig.8) que l'influence transmise résulte dans l'activation

majoritaire du comportement requis par *ReachSignal* (i.e : *turnRight* : 0.33 contre *turnLeft* : 0.14).

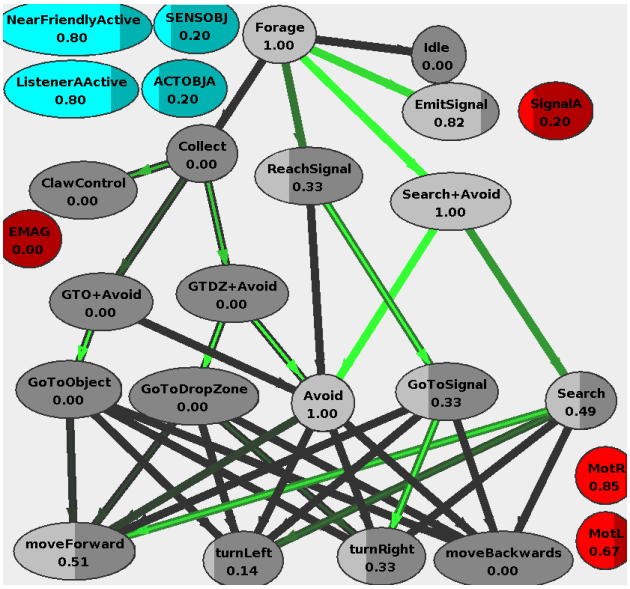

FIGURE 8 – Etape 5

Etape 6 et 7 : Le virage en direction de la source du signal est effectué, l'agent se déplace en ligne droite vers celui-ci. Une fois que l'agent est à nouveau à portée de perception d'un objet cible, le processus de collecte est répété.

L'élaboration d'une hiérarchie MIND pour un fonctionnement multi-agents en coordination ne présente ni contraintes ni difficultés particulières, autres que de respecter le point de vue et la démarche agent. La coordination se fait localement, mettant en oeuvre les solutions les plus simples en se coordonnant avec les agents les plus proches. En outre, l'algorithme génétique est extrêmement flexible vis à vis de la composition du skill fourni. Par exemple, au cours d'une expérience différente sur la tâche *Collect*, une erreur dans les paramètres a rendu l'agent incapable de saisir l'objet. Exploitant le moteur physique de la simulation, l'agent s'est contenté de pousser l'objet dans la zone de dépôt. On peut voir dans notre cas le comportement de l'agent à l'étape 5 ou encore le fait que le skill *Idle* permettant à l'agent d'attendre que les autres agents arrivent n'a pas été utilisé.

La grande difficulté, comme l'ont remarqué Dorigo et Colombetti [9][10], réside dans l'élaboration de la fonction de fitness. Régler l'équilibre entre plusieurs sources de récompenses, la valeur à attribuer à chacune est un processus qui nécessite souvent plusieurs essais pour observer (et interpréter) l'impact de chaque réglage. Bien que dans notre cas la fonction de fitness elle même ne soit qu'une simple récompense pour chaque objet collecté, ce travail s'est trouvé reporté sur l'équilibrage des paramètres de l'environnent et la compréhension de leurs interactions avec la fonction de fitness. Pour reprendre l'exemple du skill *Idle*, après avoir observé le comportement des agents, nous nous sommes aperçus que l'agent qui trouve le groupe d'objets en premier a rarement le temps de quitter la zone de perception avant que les autres agents arrivent, et donc il n'est pas nécessaire de les attendre (c'est même un gain de temps et donc une chance de collecter plus d'objets dans le temps imparti). Ici la fonction de fitness n'est pas directement en cause mais simplement la taille de l'environnent ne donne pas l'occasion aux agents d'être assez éloignés pour avoir besoin du skill Idle. D'autre part, la taille de l'environnement, la dispersion du groupe d'objets et même la densité des obstacles dans l'environnement impactent l'apprentissage et l'utilisation d'une recherche d'objet coordonnée au lieu d'un simple parcours aléatoire de l'environnent.

6 Conclusion

Nous avons vu que l'architecture MIND est capable d'accomplir une tâche de coordination pour un groupe homogène d'agents réactifs. Il a été possible de combiner des comportements acquis de manière individuelle avec de nouveaux comportements strictement collectifs, et ce sans changement majeur dans les techniques d'apprentissage employées. Le seul impact sur le coût en ressources de calcul vient de la gestion des signaux dans l'environnement simulé. L'aspect modulaire de MIND a encore un fois montré son avantage puisqu'il a été possible d'ajouter non seulement de nouveaux comportement, mais aussi de nouveaux capteurs et actionneurs sans aucune modification à la hiérarchie d'origine. Pris individuellement, le comportement *Collect* est toujours capable de fonctionner de manière autonome.

Comme nous avons traité de l'apprentissage d'un groupe homogène d'agents se coordonnant pour atteindre un but commun, tous les agents du groupe utilisent le même génome durant l'évaluation d'une population. L'intérêt réside dans le fait que d'une part, à chaque instant le même génome est évalué autant de fois que le nombre d'agents présents, et d'autre part le même génome est évalué de part et d'autre d'une interaction. Ce dernier aspect simplifie l'évaluation car il ne s'agit plus de savoir si chaque agent a bien effectué son rôle dans l'interaction, ou même quels sont les rôles et la forme de cette interaction, mais simplement de savoir si un résultat globalement positif a eu lieu. Dans le cas d'une évaluation d'une interaction entre des génomes différents, un cas d'échec amène soit à punir les deux, soit devoir déterminer lequel des deux a causé l'échec. Il est possible d'imaginer une tâche compétitive ou une évaluation concurrente est au contraire plus simple à établir, par exemple un combat avec élimination.

Cette question de l'hétérogénéité, plus particulièrement dans les capacités et les motivations des agents est un point important pour notre projet d'agent développemental. Serait-il possible d'obtenir une coordination dans un groupe hétérogène d'agents réactifs ? Dans le cadre, par exemple, d'une évaluation concurrente de génomes de différentes populations dans un contexte proie-prédateur. Ou encore conduire à l'émergence d'une relation symbiotique dans un contexte de coopération où l'hétérogénéité induit un biais initial vers le développement d'un rôle spécifique.

Cette première approche de coordination multi agent s'est basée sur des agents uniquement réactifs. Nos récents travaux sur l'architecture MIND nous ont permis d'intégrer un système de mémoire simple, les variables, que nous utilisons pour stocker des représentations internes. Ces représentations sont apprises de manière émergente, en utilisant le même système d'apprentissage par curriculum que pour les skills. En utilisant ce système nous sommes parvenus à apprendre à notre agent à compter et à retenir le nombre d'étapes accomplies dans une séquence d'actions afin de choisir le bon comportement à exécuter.

En utilisant ce système de mémoire nous envisageons d'apprendre un comportement multi agent de coopération, au moyen d'une variable représentant le rôle de l'agent, comme une première approche d'un problème d'organisation AGR [12]. La relation entre la variable représentant le rôle et les skills de la hiérarchie MIND serait apprise au moyen d'un curriculum. La valeur de cette variable évoluerait ensuite au cours d'une simulation pour obtenir l'émergence d'une organisation par rôle au sein d'un groupe d'agents homogènes, d'une manière similaire aux travaux des simulations MetaCiv [13, 34], où une spécialisation sociale est apprise par renforcement.

Références

[1] R. C. Arkin and T. Balch. Aura : Principles and practice in review. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2-3) :175–189, 1997.

[2] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48. ACM, 2009.

[3] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1) :14–23, 1986.

[4] R. A. Brooks, P. Maes, M. J. Mataric, and G. More. Lunar base construction robots. In *EEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*, pages 389–392. IEEE, 1990.

[5] N. Carlési. *Coopération entre véhicules sous-marins autonomes : une approche organisationnelle réactive multi-agent.* Theses, Université de Montpellier 2, 2013.

[6] D. Choi and P. Langley. Evolution of the icarus cognitive architecture. *Cognitive Systems Research*, 48 :25–38, 2018.

[7] J.-L. Deneubourg and S. Goss. Collective patterns and decision-making. *Ethology Ecology & Evolution*, 1(4) :295–311, 1989.

[8] J.-L. Deneubourg, S. Goss, and al. The dynamics of collective sorting robot-like ants and ant-like robots. In *1st international conference on simulation of adaptive behavior on From animals to animats*, pages 356–363, 1991.

[9] M. Dorigo and M. Colombetti. Robot shaping : Developing autonomous agents through learning. *Artificial intelligence*, 71(2) :321–370, 1994.

[10] M. Dorigo and M. Colombetti. *Robot shaping : an experiment in behavior engineering*. MIT press, 1998.

[11] J. Ferber. Les systèmes multi-agents : vers une intelligence collective. *InterEditions, Paris*, 322, 1995.

[12] J. Ferber, O. Gutknecht, F. Michel, et al. Agent/-group/roles : Simulating with organizations. In *ABS'03 : Agent Based Simulation*, 2003.

[13] J. Ferber, J. Nigon, G. Maille, T. Seguin, S. Holtz, and T. Stratulat. De masq à metaciv : un cadre générique pour modéliser des sociétés humaines dans une approche transdisciplinaire, 2014.

[14] F. Foglino, C. C. Christakou, and M. Leonetti. An optimization framework for task sequencing in curriculum learning. In *Joint IEEE 9th International Conference ICDL-EpiRob*, pages 207–214. IEEE, 2019.

[15] S. Goss and J.-L. Deneubourg. Harvesting by a group of robots. In *Proceedings of the First European Conference on Artificial Life*, pages 195–204, 1992.

[16] N. Heess, G. Wayne, Y. Tassa, T. P. Lillicrap, M. A. Riedmiller, and D. Silver. Learning and transfer of modulated locomotor controllers. *CoRR*, abs/1610.05182, 2016.

[17] D. Helbing and A. Johansson. Pedestrian, crowd, and evacuation dynamics. *arXiv preprint arXiv :1309.1609*, 2013.

[18] G. Konidaris and A. G. Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in neural information processing systems*, pages 1015–1023, 2009.

[19] J. E. Laird. Extending the soar cognitive architecture. *Frontiers in Artificial Intelligence and Applications*, 171 :224, 2008.

[20] P. Langley and D. Choi. A unified cognitive architecture for physical agents. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 1469. Menlo Park, CA ; Cambridge, MA ; London ; AAAI Press ; MIT Press ; 1999, 2006.

[21] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini. Developmental robotics : a survey. *Connection science*, 15(4) :151–190, 2003.

[22] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini. Developmental robotics : a survey. *Connection Science*, 15(4) :151–190, 2003.

[23] M. Minsky. *Society of mind*. Simon and Schuster, 1988.

[24] J. L. Morales, P. Sánchez, and D. Alonso. A systematic literature review of the teleo-reactive paradigm. *Artificial Intelligence Review*, 42(4) :945–964, 2014.

[25] S. Narvekar, J. Sinapov, M. Leonetti, and P. Stone. Source task creation for curriculum learning. In *International Conference on Autonomous Agents & Multiagent Systems*, pages 566–574. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[26] N. Nilsson. Teleo-reactive programs for agent control. *Journal of artificial intelligence research*, 1 :139–158, 1993.

[27] P.-Y. Oudeyer. Developmental robotics. In *Encyclopedia of the Sciences of Learning*, pages 969–972. Springer, 2012.

[28] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10) :1345–1359, 2010.

[29] J. Piaget. *The construction of reality in the child.* Basic Books, New York, 1954.

[30] J. Piaget and E. Duckworth. Genetic epistemology. *American Behavioral Scientist*, 13(3), 1970.

[31] M. Resnick. Learning about life. *Artificial life*, 1(1_2) :229–241, 1993.

[32] C. W. Reynolds. Flocks, herds and schools : A distributed behavioral model. In *ACM SIGGRAPH computer graphics*, volume 21, pages 25–34. ACM, 1987.

[33] O. Simonin and J. Ferber. Modeling self satisfaction and altruism to handle action selection and reactive cooperation. In *Proceedings of the 6th International Conference on the Simulation of Adaptive Behavior*, volume 2, pages 314–323, 2000.

[34] F. Suro, J. Ferber, and T. Stratulat. Coglogo : une implémentation de metaciv pour netlogo. In *JFSMA*, pages 166–167. PFIA, 2019.

[35] F. Suro, J. Ferber, T. Stratulat, and F. Michel. Une représentation hiérarchique de comportements agents pour l'apprentissage progressif et continu. In *JFSMA*, pages 67–76. PFIA, 2019.

[36] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains : A survey. *Journal of Machine Learning Research*, 10(Jul), 2009.

[37] P. Varshavskaya. Behavior-based early language development on a humanoid robot. Technical report, MIT AI Lab, 2002.

[38] J. Zlatev and C. Balkenius. Introduction : Why epigenetic robotics ? 2001.