DOCTORAT BRETAGNE LOIRE / MATHSTIC

Université de Bretagne Occidentale

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE BRETAGNE OCCIDENTALE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Thomas ALVES

Visualisation et Interactions avec une Colonie d'Abeilles Virtuelle

Simulation, complexité et pédagogie

Thèse présentée et soutenue à Brest, le « date »
Unité de recherche : Lab-STICC CNRS UMR 6285
Thèse N° : « si pertinent »

Rapporteurs avant soutenance :

Prénom NOM Fonction et établissement d'exercice
Prénom NOM Fonction et établissement d'exercice
Prénom NOM Fonction et établissement d'exercice

Composition du Jury :

Attention, en cas d'absence d'un des membres du Jury le jour de la soutenance, la composition du jury doit être revue pour s'assurer qu'elle est conforme et devra être répercutée sur la couverture de thèse

Président :	Prénom NOM	Fonction et établissement d'exercice *(à préciser après la soutenance)*
Examinateurs :	Prénom NOM	Fonction et établissement d'exercice
	Prénom NOM	Fonction et établissement d'exercice
	Prénom NOM	Fonction et établissement d'exercice
	Prénom NOM	Fonction et établissement d'exercice
Dir. de thèse :	Vincent RODIN	Professeur d'Université, Université de Bretagne Occidentale, Brest
Co-dir. de thèse :	Thierry DUVAL	Professeur, IMT Atlantique, Plouzané
Encadrant :	Jérémy RIVIERE	Maître de Conférences, Université de Bretagne Occidentale, Brest

Invité(s) :

Prénom NOM Fonction et établissement d'exercice

Remerciements

Je tiens à remercier
I would like to thank. my parents..
J'adresse également toute ma reconnaissance à ....
....

Table des Matières

TABLE DES FIGURES

INTRODUCTION

Orga Abeille - Visu Interagir avec - Simuler - Modeliser - (pk existant pas bon)proposition Modele motivation (orga++) - Eval Ccl (motivation mieux) - Enfin intéragir et visu

— Colonie d'abeille en tant que système complexe. Beaucoup de recherche sur les butineuses mais moins sur l'intérieur de la ruche.
— Multi agent car concentration sur les individus et leurs interactions
— Comprendre l'auto organisation interne par la modélisation SMA vs Équation différentielles - de l'importance des contacts individuels.
— Transmettre et faciliter l'apprentissage avec l'environnement immersif.

Contexte : Colonie d'abeilles et Complexité, Biologie et Systèmes Complexe

s

Les insectes sociaux sont depuis longtemps étudiés pour leurs capacités complexes à se répartir dynamiquement le travail, sans l'aide d'un contrôle central. Ce chapitre présente tout d'abord les principales notions de complexité, et donne plusieurs exemples de phénomènes complexes que l'on peut retrouver dans la vie d'une colonie d'abeilles. Nous verrons dans le chapitre suivant quelques modèles Multi-Agents présents dans la littérature servant à modéliser ces systèmes.

Systèmes complexes

Il n'existe à ce jour pas de définition précise de ce qu'est un système complexe [19], du fait de l'étendue vertigineuse des domaines et cas d'applications touchés. Nous pouvons cependant parler de systèmes complexes pour ceux qui suivent l'adage "Le tout vaut plus que la somme des parties" [15]. En effet, nous retrouvons souvent dans la littérature les notions de chaos, d'interactions locales et d'imprévisibilité. Au centre de tout ceci se trouvent les boucles de rétroactions. Les boucles de rétroaction se présentent sous deux formes, les rétroactions positives et négatives. Les boucles de rétroactions positives concernent moins les systèmes complexes, car elles n'ont pas de rôle de régulation, mais plutôt d'accélération. Une bombe nucléaire fonctionne avec ce principe, une fission va en déclencher plusieurs autres, qui vont à leurs tours en déclencher bien d'autres, sous une forme d'escalade de la violence exponentielle. Sans autre mécanisme pour les réguler, ce sont des phénomènes très transitoires.

Nous nous intéressons ici surtout aux boucles de rétroactions négatives. Ces boucles consistent en mécanismes régulateurs, stabilisant un système vers un équilibre. Nous pouvons en trouver de toutes sortes, sous bien des formes. Le soleil par exemple, voit sa forme maintenue par ce qui est appelé un équilibre hydrostatique : lorsque son cœur réalise la

fusion nucléaire, il crée une immense force en son centre, le faisant s'étendre. Le soleil gonfle alors et fait ainsi baisser la pression en son centre, le nombre de fusion est alors légèrement réduit. La force poussant le soleil à s'étendre est alors contrée par la gravité, le poussant de toute part vers son centre. Le soleil s'effondre sur lui même ! Mais du même temps, la pression en son centre remonte, les fusions reprennent de plus belle et il gonfle à nouveau : le soleil danse légèrement autour de son point d'équilibre, ce qui peut être vu comme une boucle de rétroaction. Autre exemple bien différent, un régulateur de vitesse de voiture : lorsque la vitesse mesurée est trop basse, le régulateur déclenche une accélération, qui augmente alors la vitesse du véhicule. Le régulateur arrête d'accélérer lorsque la vitesse est satisfaisante. Le véhicule est alors ralenti par les différents frottements sur ses composants, jusqu'à ce que la vitesse soit à nouveau assez basse pour déclencher une nouvelle accélération.

De cette manière, nous pouvons imaginer un système peuplé de nombreuses entités aux caractéristiques bien différentes, mais qui interagissent entre-elles via diverses boucles de rétroactions. C'est ainsi que l'ordre semble émerger du chaos, nous sommes alors face à un système complexe ! Un bel exemple de tels systèmes auxquels nous sommes soumis tous les jours est le climat. De faibles interactions qui peuvent paraitre insignifiantes ont de grosses répercussions sur le système dans son ensemble. Nous augmentons d'un demi degré la température des océans et ce sont les courants marins, les vents et tempêtes, les chaines alimentaires et bien d'autres qui se dérèglent. Augmenter de quelques points le pourcentage de gaz à effet de serre dans l'atmosphère et le circuit s'emballe et réchauffe le tout, augmentant la fréquence d'événements violents etc [2]. Une grande quantité d'entités, jusqu'à l'échelle moléculaire, vont interagir entre elles, faisant varier leurs températures, leurs pressions, leurs absorptions et réflexions de la lumière, altérant par la même occasion le climat sur une échelle planétaire, affectant jusqu'à notre mode de vie.

Nous allons maintenant constater en quoi une colonie d'abeilles est un bon représentant de la grande famille des systèmes complexes.

Abeilles et Systèmes complexes

Une colonie classique d'abeilles domestiques *Apis Melifera* héberge en moyenne quelques dizaines de milliers d'individus, 50 000 est un nombre qui revient souvent. Tous ces individus ont des besoins en nourritures et en eau, mais l'attention au couvain, regroupant tous les stades de vie de l'abeille avant sa phase adulte, demande un tout autre niveau d'attention. Nourriture spéciale, température précise, des cellules de cires, et nettoyées en

plus ! Tous ces besoins créent une chaine logistique impressionnante, que des apiculteurs et chercheurs observent depuis des centaines d'années. En effet, prouesse remarquable, les abeilles (et bien d'autres insectes sociaux) arrivent à survivre et même à s'épanouir sans aucun contrôle central, aucun individus spéciaux chargés du management ou de la surveillance des stocks. Des rôles ont été créés pour classer les différents maillons de cette chaîne, dont nous allons décrire les principaux [34, 35, 31].

Emblématique de la colonie d'abeilles et présentant des différences physiques notables, la Reine, plus grande que les autres abeilles, est chargée de pondre à une vitesse ahurissante : près d'un œuf par minute en moyenne sur toute sa vie. Elle est en permanence entourée d'une "cour royale", d'autres abeilles qui viennent la lécher de tout côté, attirées par ses phéromones puissantes. Elles vont ensuite au fil de leur vie, peut être même sans s'en rendre compte, répandre les phéromones royales dans toutes la colonie, assurant ainsi la suprématie de la reine. Souvent ces abeilles sont des nourrices, elles sont chargées de nourrir le couvain. De vraies cuisinières, elles parcourent la ruche à la recherche de miel et de pollen afin de concocter un liquide calorifique dont les larves se nourrissent. Parfois, une recette alternative encore plus riche, la célèbre gelée royale, est donnée à des larves spécialement sélectionnées pour devenir de nouvelles reines.

Les larves ne sont que d'énorme systèmes digestifs autonomes. Elles ingurgitent des quantités gigantesque de nourriture comparé à leur poids, pour assurer leur croissance rapide, et leur permettre de devenir par la suite de solides ouvrières. La reine pond des œufs, qu'elle dépose au fond de cellules. Ces cellules doivent être bâties par des ouvrières cirières, et nettoyées, la reine ne pond jamais dans une cellule sale. Une fois pondu, un œuf va mettre trois jours pour évoluer en larve, puis consommer son mélange de miel et de pollen pendant les six jours que représentent l'état de larve, puis va devenir une nymphe, étape importante où la larve va se métamorphoser progressivement en sa forme finale, une abeille adulte, 21 jours après la ponte. Dès qu'une larve passe au stade de nymphe, elle est repérée par une ouvrière cirière qui va méthodiquement operculer la cellule : créer une sorte de toit, celant la nymphe qui devra ouvrir sa cellule elle même, dans ses premières heures d'adulte, à l'aide de ses nouvelles mandibules.

Pour que toute cette croissance se passe bien, la température du couvain doit être exactement de 35°C. Un écart de l'ordre du demi degré peut engendrer des pertes colossales, et mettre la colonie en péril. Lorsqu'il fait légèrement trop chaud nous trouvons des abeilles thermo-régulatrice : elles battent des ailes sur le couvain et/ou au niveau de la sortie de la ruche pour créer un courant d'air et rafraichir la ruche. Certaines vaporisent

même de l'eau dans la ruche pour aider à faire chuter la température. Lorsqu'il fait froid, les abeilles se resserrent progressivement au dessus du couvain, afin de créer ce qui est appelé la "grappe" et tenir le couvain au chaud, grâce à leurs corps. Si leur simple présence ne suffit plus, elles peuvent actionner certains de leurs muscles afin de générer un peu plus de chaleur, ce qui arrive pendant l'hiver, phase critique pour la colonie. Si la grappe est trop petite pour recouvrir l'ensemble de couvain, alors une bonne partie de ce-dernier, celui en périphérie qui n'aura pas été protégé par les adultes, va mourir, on l'appelle alors froidement le "couvain refroidi".

Les abeilles sont surtout célèbres pour leur butinage, rôle tenu par les "butineuses". Celles-ci sélectionnent les meilleurs sites de ressources sur quelques kilomètres à la ronde et ramènent nectar et pollen à la colonie. Certaines abeilles plus téméraires vont spontanément quitter la ruche sans destination, dans le seul but de trouver une nouvelle source de nectar. Ces butineuses sont appelées des "scouts", ou des "éclaireurs". Elles repèrent les fleurs grâce à leurs immenses yeux sensibles aux ultra-violet, que les fleurs ont appris à bien réfléchir dans une co-évolution avec les pollinisateurs. Une fois sur une source de nectar, il est temps d'en juger la qualité. L'abeille va récolter le nectar, et estimer sa teneur en sucre. Si cette teneur lui convient, elle va rentrer à la colonie et commencer la troisième étape, le point clé, le recrutement. Mais avant ceci, une fois rentrée, la butineuse cherche une "receveuse" : une fois trouvée, elle lui donne une partie de sa charge de nectar. Les butineuses donnent ainsi leur récolte à trois ou quatre receveuses. Ensuite, elles viennent déposer sans trop d'attention leurs ballots de pollen dans des cellules proches du couvain, puis repartent. Avant de décoller, certaines dansent ! L'abeille recruteuse va se mettre à danser la désormais célèbre *"Waggle Dance"*, en forme de 8. Cette danse a un objectif double. Le premier : communiquer la position de la source de nectar par rapport au soleil, pour permettre aux autres de la retrouver. Le second est plus indirect : plus la source est sucrée et proche, et donc profitable, plus l'abeille va danser longtemps, et même répéter la danse dans plusieurs endroits de la ruche. Une danse plus longue offre plus de temps à d'autres abeilles de venir la suivre et apprendre la position de cette nouvelle source. Ainsi, plus la source est profitable, plus la recruteuse va communiquer la position à de nombreuses butineuses.

Dans le même temps, les receveuses vont déposer le nectar reçu dans des cellules, toujours très hautes sur le cadre. Certaines iront aussi tasser les ballots de pollen dans leurs cellules, afin de ne pas perdre de place. Le nectar est alors prêt à subir ses transformations pour devenir du miel. Certaines abeilles vont alors venir "cracher" sur le nectar, afin d'y

déposer leurs enzymes qui feront le travail de transformation. Elles y déposent du même coup une substance antiseptique, s'assurant ainsi que le miel ne sera pas contaminé par de mauvaises bactéries ou virus (c'est en partie ce qui lui confère ses propriétés médicales). Ensuite, pendant ce travail d'enzymes, des "ventileuses" viendront se placer au dessus du miel et battre frénétiquement des ailes. Elles ajustent ainsi l'hygrométrie du miel, l'amenant très précisément a 15% d'humidité, parfait pour la conservation.

Toutes ces ouvrières sont sœurs ou demi-sœurs. Elles ont la plupart du temps toutes la même mère, mais pas forcément le même père. Au moment de la ponte, la reine "choisi" de féconder ou non son œuf. Les cellules où sont pondus les mâles sont plus grandes que celles des femelles, l'hypothèse la plus répandue est que les cellules plus petites des femelles viennent comprimer l'abdomen de la reine, provoquant ainsi la fécondation, sans décision au moment de la ponte. Par la magie de la biologie, un œuf non fécondé donnera un mâle, un œuf fécondé donnera une femelle. Les mâles ont une vie très particulière : ils sont incapables de quoi que ce soit, ni de se laver, ni de se nourrir seuls, et ne font rien dans la colonie. Dès que la situation se gâte et que les ressources se font rares, ce sont les premiers à être chassés de la colonie. Reconnaissable à leurs immenses yeux, ils ne sont bon qu'à un chose, s'envoler au bon moment vers un point de rencontre défini (on ne sait pas trop comment), localiser une reine et la féconder. Ils meurent instantanément après l'acte (globalement leur intérieur explose). Une reine est ainsi fécondée par une dizaine de mâles et va garder leurs semences dans sa spermathèque et s'en servir tout au long de sa vie.

Nous allons désormais nous intéresser plus en détails à ces nombreux mécanismes de contrôles comme celui-ci. Comment les abeilles parviennent à savoir quoi faire, sans personne pour diriger le tout.

Auto-Organisation de la Colonie

Afin d'assurer le bon fonctionnement et l'épanouissement de la colonie, chaque individu la composant doit effectuer un certain nombre de tâches lorsqu'elles sont nécessaires, et sans aucun contrôle central. Par exemple, personne pour surveiller la température et assigner une équipe à la régulation de la température pendant un certain temps. Nous parlons alors d'auto-organisation. Chaque individu utilise ses perceptions locales pour savoir quel travail réaliser, ainsi nous parlons aussi d'allocation du travail. Les quelques mécanismes complexes d'auto-organisation (et leurs boucles de rétroactions) que nous allons aborder sont les suivants (il en existe bien d'autres, et surement une majorité dont

nous ignorons pour l'instant jusqu'à l'existence) :

- La thermorégulation : la température est maintenue précisément à 35°C au niveau du couvain.
- La sélection des meilleures sources de nourritures par les butineuses par le biais de leur mécanisme de recrutement.
- La régulation de l'âge du premier butinage en fonction des demandes du couvain, les variations de vitesses de vieillissement chez les ouvrières.

Thermorégulation

Un des points clés de cette capacité est que chaque individu a des "tolérances" légèrement différentes. Une diversité qui provient notamment de la présence des ces fameuses "demi-sœurs", dont nous avons parlé juste avant. J. Jones et.al. [20] ont montré que a capacité des abeilles à réguler la température est liée à cette diversité génétique. En effet, pour que la colonie régule correctement, il est important d'éviter des mouvements de foules à l'échelle de la colonie, afin de ne pas osciller entre trop chaud et trop froid en permanence. La diversité génétique modifie légèrement les seuils des tolérances des abeilles, qui vont alors progressivement réguler la température. Lorsque la température sera légèrement trop haute, seules certaines abeilles ventileront, les autres, de part leurs tolérances plus élevées, ne vont pas participer. Ainsi, si l'action des premières est suffisante, la température revient doucement au bon niveau. Si la température continue de monter, alors de plus en plus d'abeilles la verront atteindre leur seuil de tolérance, et se joindront à l'effort.

De cette manière, chaque individu mesure et juge la température actuelle de la colonie, et prends alors la décision de réchauffer, refroidir, ou réaliser d'autres devoirs. Sa position dans la ruche et sa sensibilité personnelles vont entrer en compte dans cette décision. Ainsi, par l'action autonome de chacun des individus, la température du couvain ne s'écarte jamais plus d'un degré de 35°C. La colonie voit a toujours un nombre optimal d'individu régulant la température, permettant de répondre parfaitement au besoin sans délaisser les autres tâches.

Sélection des meilleures sources de nectar

Lors de la collecte du nectar, les butineuses sont très sélectives et préfèrent les nectars aux hautes teneurs en sucres sur des fleurs placés le plus proche possible de la ruche. La colonie utilise donc un système de recrutement, presque de recommandation, afin d'allouer

ses effectifs de butineuses de manière optimale et dynamique, préférant les sources proches et nutritives. Une fois recrutées, les nouvelles butineuses vont se rendre à la source et répéter le processus : collecter, juger, rentrer et parfois recruter. Là encore, la diversité est clé : des abeilles moins portées sur la communication vont passer moins de temps à recruter, afin de maximiser le temps de butinage. En effet, il y a tout de même des dizaines de milliers de bouches à nourrir ! De plus, certaines butineuses sont moins difficiles que d'autres, elles vont donc communiquer des sources de faible qualité et y maintenir un faible contingent. Ce contingent alors moins utile dans l'immédiat sert de surveillance, car les teneurs en sucres des différents nectars peuvent fortement varier selon les saisons mais aussi pendant les périodes de la journée. Une source de faible qualité peut alors devenir une source extrêmement intéressante en quelques heures. Le groupe alors déjà présent peut observer ce changement et déjà danser sur une zone bien définie, proche de l'entrée, appelée la "piste de dance", pour avertir les autres, gagnant ainsi un temps précieux de re-découverte de la source mais aussi le temps d'amorcer une réponse conséquente. Gagner du temps sur une réaction exponentielle est toujours extrêmement précieux. C'est en effet une boucle de rétroactions positive, plus il y a d'abeilles à apprécier la source, plus il y aura de recruteuses, augmentant ainsi encore le nombre de recruteuses par la suite.

la diversité génétique fait ici effet de régulation, et permet d'ajuster le butinage en un savant équilibre entre exploration et exploitation. Il est certes important de ramener d'énorme quantité de nourriture à la colonie, mais avoir la totalité de ses effectifs sur une même source présente des risques. Les abeilles "scouts", ainsi que les "moins difficiles" citées plus tôt vont alors permettre de maintenir l'exploration et la découverte de nouvelle ressources à un niveau sécurisant, multipliant les opportunités.

Phéromones et Physiologie

L'auto-organisation de la colonie ne se joue pas que dans les perceptions, la colonie s'appuie sur des mécanismes indirects, long-termes, physiologiques, qui prennent place grâce à différentes hormones et phéromones. Nous étudions ici en détail l'importance physiologique des glandes hypopharyngiennes (GH) et de la Corpora Allata. C'est ce mécanisme qui nous intéresse tout particulièrement dans ce manuscrit. La Corpora Allata permet aux adultes de sécréter une hormone appelée Hormone Juvénile(HJ). Cette hormone est retrouvée en grande quantité chez les butineuses, et en faible quantité chez les nourrices. Les GH permettent aux abeilles s'occupant du couvain de transformer le pollen et le nectar en une substance riche destinée aux larves. Elles permettent aussi aux buti-

neuses de traiter chimiquement le nectar, le rendant utilisable pour les nourrices, transformable en miel et même consommable directement par les autres adultes. Or, ces deux comportements sont incompatibles, les GH subissent une modification physiologique pour effectuer l'une ou l'autre de ces fonctions. De plus, les abeilles (ainsi que d'autres insectes sociaux) emploient différentes phéromones pour parvenir à s'épanouir. Ces phéromones peuvent être séparées en deux catégories : les phéromones modificatrices, et les phéromones incitatrices. Les phéromones modificatrices, comme leur nom l'indique, viendront modifier la physiologie des individus : ce sont elles qui sont notamment responsable de la modification des propriétés des GH que nous venons d'aborder. Les phéromones incitatrices quant à elles viendront déclencher des comportements, sans altération physique sur les agents. Ainsi, nous pouvons voir les phéromones incitatrices comme ayant un effet très court terme, et les phéromones modificatrices un effet très long terme. Par exemple, lors d'une attaque, les premières abeilles témoins viendront émettre une phéromone d'alarme très volatile, qui aura pour effet de faire sortir une grande quantité d'abeilles de la ruche pour servir de renfort. Cette phéromone incitatrice augmente aussi fortement l'agressivité des abeilles. Fait étonnant, d'après certains apiculteurs, cette phéromone aurait parfois une odeur proche de la banane !

La Figure 1 schématise les interactions entre phéromones modificatrices, hormones et glandes, ainsi qu'entre différents individus de la colonie, que nous allons décrire dans cette partie. Nous avons pu construire ce modèle simplifié des phéromones modificatrices avec des collègues biologistes de l'INRAE, à Avignon *Comment citer Yves et Cedric ici ?*. Cette Figure présente les trois principales phéromones modificatrices connues à ce jour. À gauche, les phéromones E-β-Ocimene (que nous appellerons désormais Ocimene pour plus de clarté) sont majoritairement émisent par de très jeunes larves (moins de 3 jours). Ces phéromones réduisent le développement ovarien des ouvrières (représenté par un lien rouge sur la figure) et déclenche une forte hausse de butinage de pollen de la part des butineuses, qui se concentrent habituellement sur le nectar [25]. Ensuite, le 9-ODA, un des composants des puissantes phéromones de reine, ralenti lui aussi le développement ovarien des butineuses mais réduit aussi les concentrations en Hormone Juvénile (HJ) chez les ouvrières. Une hypothèse répandue est de considérer que l'HJ sécrétée par la Corpora Allata permet d'altérer le fonctionnement des GH, dictant ainsi leur utilité pour les nourrices ou les butineuses. Typiquement, la transition de nourrice à butineuse se fait en une vingtaine de jours. La colonie suit ce qu'on appelle le Polyéthisme d'Âge : les adultes ayant le même âge réalisent les mêmes activités. Or, il a été montré que ce polyéthisme

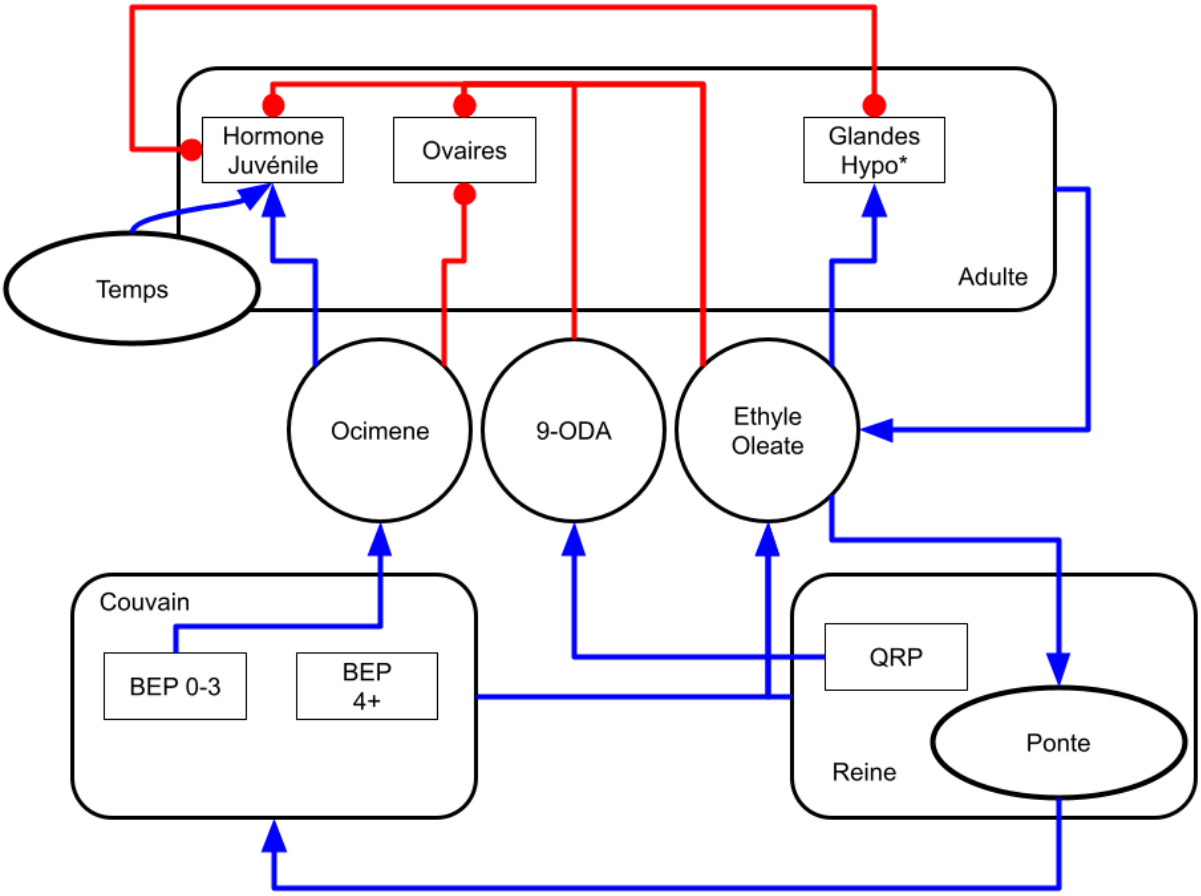


FIGURE 1 – Synthèse des connaissances simplifiées et schématisées des phéromones modificatrices au sein de la colonie.

est souple, et que dans les bonnes conditions, une abeille peut aller butiner dès ses 5 jours, au lieu de la vingtaine habituelle. Ce mécanisme est lié à une phéromone, émise par toute la colonie, l'Ethyle Oléate (EO). La réduction d'HJ provoquée par l'Ocimene et le 9-ODA permet donc de maintenir ces ouvrières à une physiologie de nourrices.

L'Ethyle Oleate est retrouvée majoritairement sur le couvain et la reine, elle est aussi retrouvée chez les butineuses. Lorsque cette phéromone est injectée en grande quantité à des abeilles adultes, il a été montré que celles-ci arrêtent le butinage et voient leur taux d'HJ diminuer. L'Ethyle Oléate n'est pas une phéromone volatile, elle est majoritairement transmise par contact, principalement lors d'échange de nourriture et de nettoyage mutuel : lorsqu'une abeille en nettoie une autre, ou lorsqu'une nourrice nettoie une larve. Elle serait aussi transmissible sur de courtes distances par évaporation. On observe sur la Figure 1 que l'EO favorise le développement des GH, qui elles même réduisent et son altérées par l'HJ. Nous y observons aussi que l'EO ralenti le développement ovarien des ouvrières. Dans le cas classique du polyéthisme d'âge, les jeunes abeilles effectuent un travail de nourrices, et les plus âgées butinent. Mais, comme nous venons de le voir, une nourrice peut accélérer sont vieillissement, et une butineuse peut même l'inverser, afin de s'adapter aux besoins changeant de la colonie. C'est pour ceci que nous parlerons ici d'Age Physiologique, opposé à l'âge réel. Une abeille avec un faible âge physiologique possède les GH nécessaires aux nourrices, et les plus âgées physiologiques possèdent les GH et les muscles nécessaires au vol et au butinage.

Conclusion

Dans ce chapitre nous avons présenté le concept de système complexe, en quoi une colonie d'abeilles en est un représentant tout en réalisant un rapide état de l'art sur les connaissances biologiques de ces insectes sociaux. Régulation de la température, adaptation de la physiologie en fonction des besoins, allocation de travailleurs sur différentes sources de nourritures en fonction de leur profitabilité sont les points que nous avons abordés, mais la colonie en contient bien d'autres, et beaucoup restent encore à être découvert. Toute cette auto-organisation, sans aucun contrôle central, représente un grand intérêt de recherche, pour comprendre leurs méthodes, les reproduire et essayer d'en extraire les principes pour des applications scientifiques et technologiques. Le chapitre suivant aborde donc les différents modèles informatiques utilisés pour simuler de tels systèmes, et présente quelques exemples d'application de ces-derniers pour des simulation.

Chapitre 1

État de l'Art : Colonies d'Abeilles et Simulation Multi-Agents

Afin de comprendre et reproduire les capacités complexes des insectes sociaux, ils sont étudiés depuis une cinquantaine d'années dans le domaine des Simulations Multi-Agents (SMA). Plusieurs travaux ont ainsi pu reproduire et approcher leurs phénoménales capacités d'auto-organisation dynamique par la simulation, notamment Swarm Intelligence [4] et bien d'autres [13, 28, 11]. Ce chapitre présente quelques modèles théoriques présents dans la littérature servant à modéliser ces systèmes, ainsi que quelques applications de ceux-ci, dans le cadre de la simulation d'insectes sociaux, et d'autres.

1.1 Modèles existants de répartition des tâches

La division du travail se produit lorsque les agents doivent décider quelle tâche exécuter dans un environnement partagé. Les sociétés d'individus (ou d'agents) doivent trouver des moyens de répartir efficacement leur main-d'œuvre entre les tâches nécessaires pour survivre et s'étendre. En informatique, le contrôle décentralisé inspiré par les insectes sociaux a été étudié pendant des années et s'est avéré efficace dans de nombreuses applications. Dans cette section, nous allons passer en revue ce qui a été fait dans le domaine des modèles de répartition des tâches.

1.1.1 Foraging For Work [17]

Dans ce modèle, les différentes tâches que les agents doivent accomplir sont spatialement dispersées en zones. Les agents, en recherche active de travail, tentent d'exécuter la tâche associée à leur zone ou se déplacent de manière aléatoire. Ainsi, les zones surpeuplées "poussent" les agents vers les zones voisines offrant du travail, ce qui entraîne une division du travail. Lorsque de nouveaux agents apparaissent dans une zone spécifique et

que les agents les plus âgés meurent à un certain âge, ce modèle assez simple recrée le polyéthisme d'âges : les agents du même âge effectuent globalement les mêmes tâches. En effet, la zone voyant des agents naitre va contenir plus d'agents qu'elle n'offre de travail, là ou du même temps une zone voyant des agents mourir sera dans le cas inverse. Ainsi, nous obtenons une zone de naissance qui aura tendance à repousser les agents, et la zone de "mort" va les attirer, car elle a besoin de main d'œuvre. Nous obtenons une répartition spatiale liée à l'âge des individus, nous retrouvons donc bien une forme de polyéthisme car les zones sont associées a des tâches. Ce modèle nous intéresse particulièrement pour cette capacité, en effet une des hypothèses sur la migration des nourrices vers le rôle de butineuse est qu'elle est provoquée par l'émergence de nouvelles nourrices au centre de zones de couvain, repoussant ainsi les nourrices plus âgées vers d'autres activités, ce que le *Foraging For Work* est tout a fait à même de recréer.

Ce modèle repose sur deux hypothèses fortes :

1. Les agents doivent avoir la capacité d'évaluer les besoins de chaque tâche, leur priorité en quelque sorte.

2. Les tâches sont dispersées en zones géographiques définies.

1.1.2 Modèles à Seuils

FTM : "Fixed Threshold Model" [4]

Avec ce modèle, chaque tâche a un score, représentant sa priorité. Un agent s'engage puis exécute la tâche ayant la priorité la plus élevée. Le FTM est basé sur quelques hypothèses fondatrices, dont voici la première : chaque tâche est associée à un stimulus. Le score de chaque tâche est calculé à partir de l'intensité du stimulus associé perçu par l'agent, généralement à l'aide d'une fonction sigmoïde. Soit T la tâche évaluée par l'agent, $F(T)$ le score de la tâche T et x_T le stimulus associé (simple ou complexe) perçu par l'agent, ces fonctions prennent alors la forme :

$$F(T) = \frac{x_T^n}{x_T^n + \Theta_T^n} \tag{1.1}$$

avec n un entier pour la non-linéarité de la fonction (généralement $n = 2$ [28]) et Θ_T le seuil de la tâche, aussi appelé biais, de la sigmoïde. La Figure 1.1 présente différentes sigmoïdes mettant en valeur l'impact du paramètre Θ_T. Le seuil sert en quelque sorte de point d'ancrage : lorsque le seuil est strictement équivalent au stimulus d'entrée, alors la valeur du résultat est exactement 0,5. Ce biais est utilisé pour modifier la perception

des agents : avec un biais très faible, les agents sont très sensibles au stimulus associé et s'engagent dans la tâche plus tôt que les agents avec un biais plus élevé [12].

Largement utilisés pour modéliser et piloter des simulations d'insectes sociaux, les modèles à seuils reposent fortement sur l'association entre tâches et stimulus. Voici les autres hypothèses fondatrices : ils supposent également que l'exécution d'une tâche diminue le stimulus qui lui est associé, et que ne pas exécuter une tâche augmente son stimulus associé. Dans le cas contraire, les agents exécuteraient constamment cette tâche, ou du moins même jusqu'à ce qu'elle ne soit plus prioritaire. Le stimulus doit être une représentation de la priorité de la tâche qui lui est associée.

Par exemple, Bonabeau et al. [5] utilisèrent un FTM pour modéliser la répartition du travail au sein d'une espèce de fourmi contenant deux types d'individu aux caractéristiques physiques très différentes [33]. Appelés "Majors" et "Minors" dans leurs travaux, ces castes correspondent respectivement a ce que nous pourrions voir comme de grands soldats et de petites ouvrières. Dans la nature, il a été observé que les ouvrières travaillent en permanence, alors que les soldats travaillent seulement lorsque la demande est trop forte par rapport au nombre d'ouvrières présentes. Ces deux castes ont alors été modélisées, chacune avec un seuil différent pour une même tâche abstraite. Ainsi, les ouvrières ont reçu un seuil très faible, elles s'engagent donc dans la tâche même lorsque le stimulus déclencheur est relativement faible. À l'inverse, les soldats ont un seuil élevé, elles nécessitent donc un stimulus déclencheur très intense pour engager la tâche. Ainsi, lorsque le nombre d'ouvrières est suffisant pour maintenir le stimulus à un niveau faible (elles sont suffisamment nombreuse par rapport à la demande), les soldats ne travaillent pas car le stimulus n'atteindra jamais une valeur suffisamment élevée. En revanche, lorsque des ouvrières sont retirées de la simulation (ou de la colonie), le peu qui reste ne parvient plus a maintenir le stimulus bas : la demande dépasse l'offre. Le stimulus grimpe donc régulièrement, jusqu'à atteindre la seuil déclencheur pour les soldats, qui se mettent alors au travail. Ce qui est intéressant c'est que lors de la réintroduction des ouvrières, les soldats arrêterons rapidement de travailler, le stimulus déclencheur redevenant trop faible. On obtient donc un magnifique exemple d'auto-organisation sans aucun contrôle central, sur une seule tâche et avec deux populations aux seuils fixes, mais différents.

Chaque tâche a également une probabilité d'interruption aléatoire évaluée à chaque pas de temps. Par exemple, un agent peut avoir 0.5% de chances d'interrompre sa tâche en cours, à chaque pas de temps [29]. Lorsque c'est le cas, l'agent recherche une nouvelle tâche en utilisant les scores de chaque tâche et choisit celle au score le plus élevé. Cette

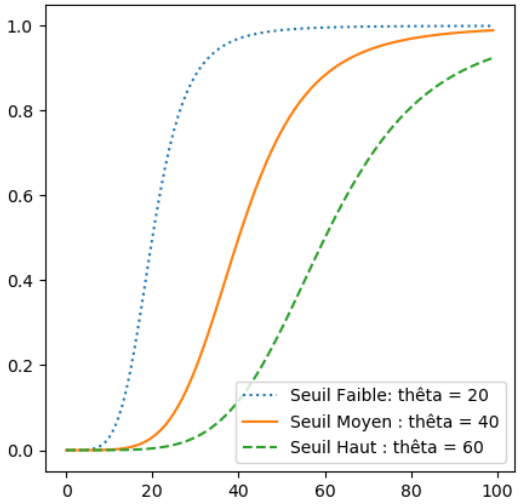


FIGURE 1.1 – L'influence du paramètre thêta (Θ) sur la forme des sigmoides.

interruption totalement aléatoire se repose sur les hypothèses fondatrice des modèles à seuils.

RTM : "Response Threshold Model", Renforcement du biais

Sur la base du FTM et de l'équation 1.1, différents travaux des années 90 [32, 14] ont proposé de mettre en place des mécanismes de renforcement de la valeur Θ, en modifiant la sensibilité des agents pendant l'exécution, formant ainsi efficacement des spécialistes. Cette mise à niveau du FTM est plus généralement appelée "Response Threshold Model" (RTM).

Par exemple, Cicirello et Smith [8] utilisèrent un modèle à seuils pour résoudre un problème d'allocation de ressource : une ligne d'assemblage de *General Motors* qui doit peindre des camions tout juste assemblés, de différentes couleurs. Chaque compartiment de peinture est alors vu comme un agent qui possède une tâche par couleur de camion. Ainsi, une tâche consiste à peindre un camion d'une couleur, et changer de couleur signifie changer de tâche. Chaque tâche possède un seuil variable, permettant d'exprimer à la fois la spécialisation d'un compartiment pour une couleur, mais aussi indirectement d'exprimer le coût du changement de couleur. En effet, lors d'un changement de couleur, beaucoup de temps est perdu car il faut purger tout le système du compartiment, gâchant du même coup une bonne quantité de peinture. L'idée est donc de minimiser les coûts en peinture

ainsi que le temps pour peindre une grande série de camions de couleurs différentes et inconnues *a priori*. Une file de camions à peindre arrive en entrée et les compartiments doivent en accepter certains pour les peindre. Un compartiment ajuste les seuils de ses tâches à chaque pas de temps. Ainsi, il diminue la tâche correspondant à sa couleur actuelle, augmentant ses chances d'accepter de peindre un camion de cette couleur, et à l'inverse augmente les seuils de toutes ses autres tâches. Lorsqu'un compartiment n'a aucun camion à peindre, il diminue alors tous ses seuils, de plus en plus vite avec le temps qui passe.

De cette manière, Cicirello et Smith arrivent à grandement limiter le nombre de changement de couleurs nécessaires, tout en conservant un rendement proche des méthodes traditionnelles comme l'approche basée sur les lois du marché, que nous allons désormais aborder rapidement. PEUT ETRE

1.2 Modèles de Répartitions des Tâches et Insectes Sociaux

Nous nous intéressons ici aux applications pratiques de modèles théoriques, principalement ceux que nous venons de décrire.

Conclusion

Le modèle "Foraging For Work" nous intéresse car il permet de simuler un cas intéressant de la colonie, la migration des nourrices vers le butinage. En revanche, il ne nous permettra pas de simuler l'ensemble de la colonie. C'est pour ceci que nous nous tournons vers les modèles à seuils : les "RTM". Ils présentent trois hypothèses fondatrices pour fonctionner : chaque tâche doit être associée à un stimulus déclencheur, l'exécution de la tâche vient réduire l'intensité de son stimulus associé, qui augmente lorsque la tâche n'est pas (ou pas assez) exécutée par les agents. D'après nos connaissances sur les abeilles, nous trouvons des tâches qu'elles réalisent qui ne respectent pas ces deux conditions : pas de stimulus directs pour pousser au butinage (sauf quelques cas précis) ou à l'alimentation du couvain. Nous nous intéressons donc ici aux situations dans lesquelles ces hypothèses ne sont pas vraies. Aussi, si certaines tâches, notament celles liées au couvain, peut être séparée en zone définie dans la ruche, ce n'est pas le cas de la majorité : le modèle FFW

ne pourra donc pas nous aider à modéliser l'ensemble de la colonie.

Nous décrivons dans le chapitre suivant notre modèle fondé sur un RTM et agrémenté d'un mécanisme supplémentaire basé sur la motivation interne pour gérer ces tâches ne respectant pas les hypothèses. Nous allons aussi devoir nous passer de l'interruption aléatoire proposée par ces modèles, qui ne fait sens que lorsque les deux hypothèses sont valides.

Chapitre 2

Proposition : Prise de Décision et Interruption

Dans ce chapitre et à l'aide de ce que nous venons d'apprendre au sujet de la répartition des tâches, nous allons pouvoir construire notre propre mécanisme générique, que nous utiliserons ensuite pour notre cas d'application : la colonie d'abeilles (Chapitre 3). Nous allons voir comment modéliser nos tâches et comment les faire exécuter par nos agents. Nous verrons ensuite les mécanismes de sélection puis d'interruption que nous avons mis en place afin que nos agents effectuent toujours une tâche qui a du sens par rapport à l'état actuel de l'environnement et à l'activité des autres agents. Nous terminerons ce chapitre par un exemple possible d'application de ce modèle dans le cadre de la robotique en essaim, où une population de robots devra se partager deux tâches : collecte de ressources et patrouille.

2.1 Modélisation des tâches : Exécution du comportement

2.1.1 Actions, Activités et Tâches

Afin de modéliser nos tâches, nous allons utiliser 3 concepts : Actions, Activités et Tâches. Plusieurs définitions différentes existent dans la littérature. Un point qui revient souvent est que le très souvent cité "rôle" est un concept abstrait [16, 36, 7], raison pour laquelle il n'apparait pas à ce niveau de notre modèle, malgré son omniprésence dans le domaine Multi-Agents. Ici et dans la littérature, un rôle peut être utilisé pour qualifier des agents qui réalisent certaines tâches particulières. Un rôle peut ainsi englober plusieurs tâche, une seule, ou même ne concerner qu'une sous-partie d'une tâche. Le rôle n'a pas d'intérêt pour le système, en revanche, il est utile à l'observateur pour mieux analyser, communiquer et décrire ce qu'il voit. Tâches et Activités sont souvent utilisées de manière

très différentes, et parfois interchangeable, mais nous nous rapprochons de la vision de [21], disant qu'une Tâche correspond à ce qui doit être fait, là où une Activité définit plutôt ce qui est en train d'être fait. Ainsi nos Tâches représentent un travail à réaliser dans sa globalité, là où l'Activité est plus proche de ce que va réaliser un agent, plus concret. Nous ajoutons à ceci le concept d'Action, qui se glisse dans cette définition décrivant le travail élémentaire réalisé par un agent à un instant t. Voici donc une définition précise de ce que nous entendons dans ce manuscrit par ces trois termes, Action, Activité et Tâches.

Une Action est définie comme une interaction avec l'environnement extérieur, non interruptible et d'une durée déterminée et courte (pas plus de quelques pas de temps de simulation, pour ne pas bloquer l'agent). Elle n'est donc pas forcément élémentaire, mais doit s'en approcher. Chaque Action possède une condition d'activation.

Ensuite, une Activité est un ensemble d'Actions et/ou d'autres Activités. Une Activité possède aussi sa propre condition d'activation. Indirectement, tout ce qu'elle contient partage alors sa condition d'activation, ce qui nous permet de factoriser cette condition et d'alléger notre écriture, et ainsi de modéliser des comportements élaborés.

Pour finir, une Tâche contient l'ensemble des Activités et Actions concernant un même comportement général. On peut donc voir une Tâche comme l'Activité racine, un peu à la manière d'un système de fichiers : les Activités sont des dossiers et contiennent d'autres dossiers et/ou des fichiers, que sont les Actions. Une Tâche est alors le dossier racine, le *root*, de cet ensemble de fichier.

2.1.2 Subsomption Hiérarchique et Exécution

Une architecture de subsomption permet de hiérarchiser différents comportements entre eux, afin d'obtenir un comportement général cohérent[6]. Dans un ordre défini, la subsomption interroge tour à tour la condition d'activation de chacun de ses différents blocs comportements, et exécute le premier dont la condition est valide. Par exemple, modéliser le comportement d'un mouton peut se faire en trois blocs. Un premier bloc "Chercher à manger", toujours valide. Au dessus de celui-ci, donc avec une priorité plus importante, nous plaçons un autre bloc : "Brouter". Ce-dernier s'active lorsque le mouton a trouvé de quoi manger. Enfin, nous plaçons au sommet le bloc "Fuir", s'activant dès que le mouton perçoit un prédateur, et reste activé le temps de le semer. Ainsi, tant qu'aucun grand méchant loup n'est en vue, le mouton va brouter paisiblement. Dès qu'il en verra un, alors il pourra fuir.

Afin de respecter l'aspect quasi-élémentaire des comportements, le bloc "Fuir" sera

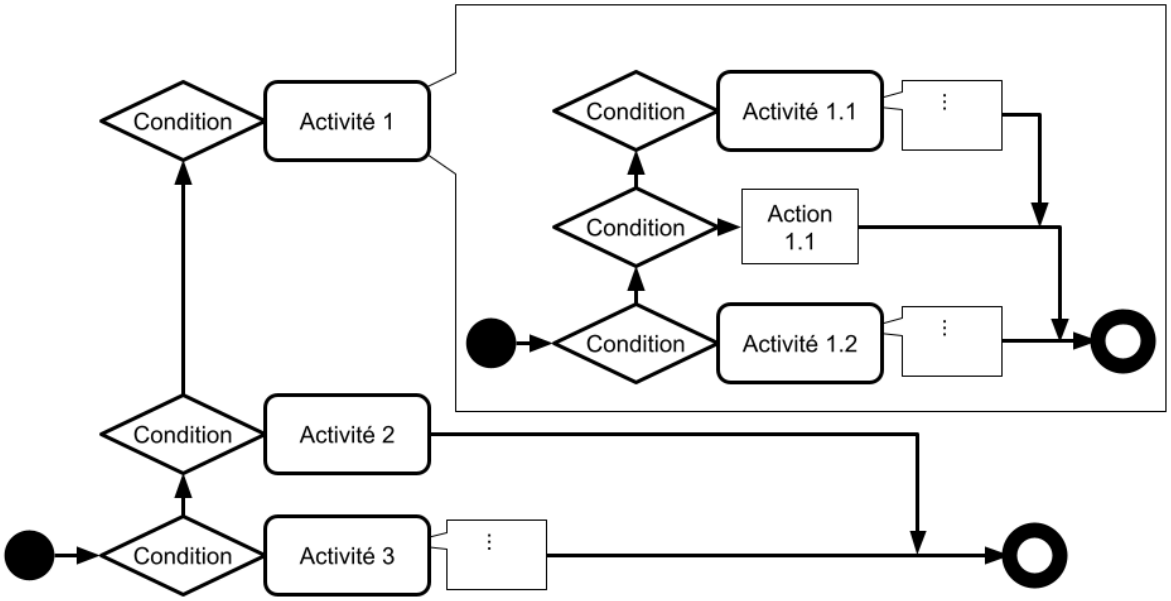


FIGURE 2.1 – Modélisation d'une tâche à l'aide d'une subsomption hiérarchique.

réalisé une multitude de fois. Ainsi, une seule exécution de Fuir ne fera faire au mouton qu'un pas l'éloignant du prédateur. Il va y faire appel plusieurs fois avant de considérer avoir semé le loup, tant que la condition du bloc "Fuir" sera valide.

Une subsomption hiérarchique ajoute à cette structure simple, le fait que chaque bloc comportement puisse être une autre architecture de subsomption[18]. Cette légère modification apporte une grande modularité dans la conception de ces architectures, et permet de modéliser des comportements plus élaborés sans la lourdeur des subsomption classiques.

Ce que nous avons appelé "bloc comportement" des subsomptions correspond à nos Actions et Activités. Les blocs qui contiennent une autre subsomption sont appelés Activités, et ceux qui contiennent du comportement sont des Actions. Ensuite, la subsomption en elle même est alors une Tâche. La Figure 2.1 présente la notion de subsomption hiérarchique contenant nos concepts définis plus tôt. Nous y observons une tâche représentée par un rectangle au trait épais, et nommée "Vie de Mouton". Elle contient une Activité "Fuir", en haut donc la plus prioritaire et représentée par un rectangle au trait moyennement épais. La Tâche contient aussi deux Actions, rectangles aux traits fins, par ordre de priorité "Brouter" et "Chercher à Manger". L'Action la moins prioritaire ne possède pas de condition, qui sont ailleurs représentées par des losanges, c'est une Action par défaut : si

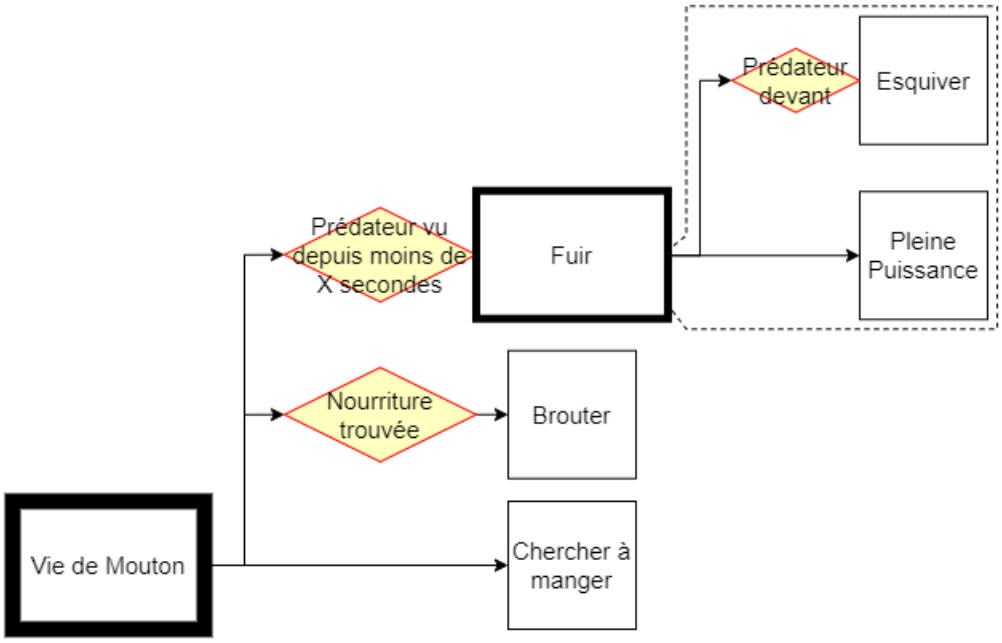


FIGURE 2.2 – Modélisation de la tâche "Vie de Mouton" contenant tout le comportement du mouton de notre exemple.

aucun autre bloc plus prioritaire a sa condition validée, cette dernière Action est toujours valide. Ensuite, l'Activité "Fuir" contient elle même deux actions, "Esquiver" et "Pleine puissance", avec et à l'instar de "Brouter" et "Chercher à manger", "Pleine Puissance" en Action par défaut ne contenant aucune condition, alors qu'"Esquiver", prioritaire par rapport à "Pleine Puissance", en possède bien une. Nous observons donc dans cette Figure l'architecture en subsomption hiérarchique, grâce à l'Activité "Fuir" qui contient une nouvelle subsomption.

Ainsi, pour qu'un agent puisse exécuter une tâche, il interroge l'Activité racine puis va récursivement interroger ses composants. Chaque Activité ou Action interrogée va ainsi vérifier sa condition d'activation. Une Activité dont l'activation est valide va alors continuer d'interroger ses composantes. On a donc une recherche en profondeur, par ordre de priorité, qui s'arrête à la première Action interrogée avec une condition d'activation valide. Cette Action est alors remontée à l'agent, qui pourra l'exécuter pendant toute sa durée. Ensuite, une fois l'Action terminée, tout ce processus recommence afin de pouvoir déterminer une nouvelle Action à exécuter.

Pour reprendre l'exemple du mouton, nous pouvons complexifier son comportement en transformant son Action "Fuir" en une Activité "Fuir" contenant deux Actions. L'Action prioritaire "Esquiver" a pour condition le fait de voir le loup droit devant, et consiste à fuir mais en tournant, afin d'éviter le loup. Ensuite, la deuxième Action, "Pleine Puissance" est l'Action par défaut, sans condition, qui consiste à courir tout droit tant que ça ne fait pas suffisamment de temps que le loup n'a pas été vu. La condition d'activation de l'Activité "Fuir" définie plus tôt, qui est validée lorsqu'un prédateur a été vu dans les dernières secondes/minutes, est alors nécessaire à l'activation des deux Actions "Esquiver" et "Pleine Puissance" sans que nous ayons à les réécrire explicitement. La Figure 2.2 reprend la structure de la tâche du mouton que nous décrivons dans cette section.

2.2 Sélection : Modèle à Seuils

Maintenant que nous avons modélisé le fonctionnement interne de nos tâches, nous allons pouvoir construire le mécanisme permettant à nos agents de sélectionner la plus prioritaire.

2.2.1 Sélection avec un RTM Classique

Pour cette sélection nous allons utiliser un modèle à seuils, que nous allons légèrement adapter en lui ajoutant un mécanisme d'interruption décrit dans la Section 2.2.3. Dans un modèle à seuils, chaque tâche possède une fonction lui permettant de calculer son score. Un agent peut ainsi sélectionner la tâche qui possède le plus élevé. Lié à un stimulus déclencheur, le score de la tâche est calculé à l'aide d'une fonction sigmoïde, paramétrée par un seuil, qui prend en entrée le stimulus (ou une combinaison linéaire de plusieurs stimulus), et nous donne en résultat le score de la tâche, comme nous avons pu le constater dans l'état de l'art, sous-section 1.1.2.

Même si plusieurs agents sont en mesure d'effectuer la même tâche, les seuils de ses tâches leurs sont propres. Chaque agent possède une instance différente des tâches, et chaque instance de tâche possède son propre seuil.

Afin de toujours réaliser une tâche utile à la communauté, nos agents doivent très régulièrement interroger leurs perceptions, afin de se "mettre à jour". C'est pour ceci que nous avons introduit la notion d'évaluation systématique : un agent réévalue l'ensemble de ses tâches à chaque fois qu'il termine une Action. C'est également pour cette raison

que les Actions doivent avoir une durée courte, de préférence atomique, pour permettre la mise à jour. Ce rafraichissement des perceptions est essentiel pour que le système puisse réagir face à une urgence. Même si notre mouton est paisiblement en train de brouter, il est intuitif de l'autoriser à fuir à la vue d'un loup avant d'avoir parfaitement terminé de brouter. Il doit aussi pouvoir faire une pause lors de la résolution d'un casse tête (nous en reparlerons) afin de se nourrir, pour ne pas mourir de faim.

En modélisant nos tâches, il arrive que certaines n'aient pas de stimulus déclencheur. C'est le cas pour les abeilles butineuses : le butinage de nectar semble être leur comportement par défaut, aucun stimulus global n'a encore été identifié. Pour continuer à construire notre modèle, nous avons besoin d'un mécanisme nous permettant d'améliorer les modèles à seuils dont nous avons discuté dans l'état de l'art. En effet, ceux-ci ne permettent pas de modéliser des tâches n'ayant pas de stimulus déclencheur : par exemple, la faim est un stimulus déclencheur de l'action de manger. En revanche, le stimulus déclencheur de "résoudre un casse tête", "ranger sa chambre" ou encore "rédiger un état de l'art", est beaucoup plus complexe. Nous proposons donc une solution simple, permettant de prendre en compte ces tâches sans stimulus tout en utilisant un modèle à seuils : en utilisant la motivation interne de l'agent. Avant d'aller plus loin dans la description de notre utilisation de la motivation, voici un rapide état de l'art sur son usage dans la littérature, et notre positionnement par rapport à celui-ci.

2.2.2 Motivation : Point sur la Littérature

Pour les psychologues, la motivation est la source de l'action et guide son exécution. Deux types de motivations existent : extrinsèque (ou externe), lorsqu'une récompense est offerte par le monde extérieur, et intrinsèque (ou interne), qui n'a à voir qu'avec les croyances ou besoins de l'agent, comme l'amusement ou la curiosité. La théorie du Flow[10], de son côté, dit que la motivation interne d'un individu est maximale lorsque la difficulté rencontrée lors de la réalisation d'une tâche est suffisante pour susciter l'intérêt (qu'elle n'est pas ennuyeuse) mais suffisamment faible pour ne pas être décourageante (qu'elle n'est pas impossible à réaliser pour l'agent).

On note dès lors que la motivation interne présente dans la littérature peut se diviser en deux catégories : d'un côté la motivation source, comme la faim, qui va provoquer un comportement, et de l'autre côté la motivation guide, motivation au sens d'implication, d'intérêt, qui elle sera plutôt un guide de cette action, comme la curiosité. On retrouve ces notions en éthologie, par exemple chez Lorenz[23], pour qui la motivation interne (guide),

couplée à un stimulus (source), va déclencher et entretenir un comportement.

En intelligence artificielle, la motivation intrinsèque *source* est particulièrement utilisée pour les systèmes d'apprentissage [30], *e.g.* pour aider ou guider des agents apprenants [3], notamment en apportant la notion de curiosité à des agents informatiques. Certains travaux [14, 24] s'attachent à sa définition proche de l'éthologie, dans laquelle la motivation intrinsèque peut venir de nombreux stimulus internes différents et plus primitifs, tels que la faim ou la peur.

D'autres travaux se basent sur la théorie du Flow : un agent qui ne parvient pas à réaliser sa tâche ressent de l'anxiété et cherche une tâche moins difficile [9]. De la même manière, un agent qui accomplit une tâche facile s'ennuie et passe à des tâches plus difficiles. L'idée de compétence apportée par Roohi et al.[27] rejoint ces notions : la compétence est, pour un agent, le sentiment d'être en contrôle et capable d'accomplir sa tâche actuelle. Ainsi, un agent ayant un niveau de compétence qu'il juge trop faible pour la tâche actuelle cherchera une tâche plus facile, plus adaptée.

Nous trouvons notamment [1] qui utilise une mesure locale de l'efficacité de robots pour distribuer des tâches.

Nous distinguons donc bien deux catégories dans la motivation interne. La première, que nous allons continuer à appeler la Motivation Source, proche de l'éthologie, décrit une motivation comme un stimulus interne servant à déclencher des comportements. Pour reprendre l'exemple de notre mouton, si voir un loup ne déclenche pas de réaction physique directe (au final ce ne sont que des pixels plus sombres sur le fond de sa rétine), son cerveau va pourtant reconnaitre le loup et faire augmenter la soudaine motivation de prendre ses jambes à son cou. On peut le voir comme si c'était la peur qui déclenchait la fuite, la peur est donc une Motivation Source.

La deuxième, la Motivation Guide, proche de l'idée du *Flow*, permet à l'agent de se situer par rapport à la difficulté de la tâche qu'il exécute, afin de maximiser son apprentissage, et donc d'optimiser l'usage de son temps. L'exemple de notre mouton sera ici quelque peu improbable, mais nous nous y tiendrons : nous souhaitons apprendre à ce mouton comment résoudre un *Rubik's Cube*. Si le célèbre casse-tête lui est donné sans introduction, la tâche est insurmontable, décourageante. Le mouton n'apprend rien, il perd son temps et va donc naturellement se déconcentrer et essayer autre chose (peut être essayera-t-il d'apprendre à jongler avec plusieurs cubes ?). En revanche, si la courbe

de difficulté est adaptée, en exercices simples et incrémentaux, le mouton apprendra bien mieux et restera concentré : la difficulté sera alors toujours adaptée à ses compétences.

Avec ces deux notions en tête, Motivation Source et Guide, nous pouvons passer à la suite de la description de notre modèle, dont la sélection et l'interruption des tâches se feront dorénavant grâce à ces motivations.

2.2.3 Action Démotivante et Tâche Motivée

Stimulus Artificiel

Dans le cas de Tâche dont aucun stimulus déclencheur ne peut être trouvé, nous appliquons la Motivation Source et proposons de créer un stimulus déclencheur artificiel, d'une valeur donnée, qui peut être fixe mais que ne pourrions faire varier au besoin. Ce stimulus artificiel viendra agir exactement comme si l'agent percevait ce stimulus, et sera alors traité par la fonction de calcul de score de la tâche comme un stimulus classique. Ainsi, toutes nos tâches, ayant recours à un stimulus artificiel ou non, sont capables de produire un score cohérent et sont comparables entre elles. Nous pouvons alors dire d'une Tâche qu'elle est "Motivée" lorsqu'elle utilise un stimulus déclencheur artificiel.

Nous avons désormais à notre disposition deux mécanismes influant sur le score d'une tâche : nous pouvons jouer sur sa motivation source et modifier son seuil. Afin de garder une approche cohérente, nous proposons de modifier le seuil pour représenter l'état interne de l'agent (caractéristiques physiques, physiologiques, etc.). De la même manière, modifier la motivation source peut permettre de représenter un changement de perception, ou une décision de la part d'un agent, si besoin. La motivation source est conservée constante dans la totalité de nos implémentations.

En revanche, le score d'une Tâche Motivée ne représente alors plus sa priorité. Afin de résoudre ce débordement des hypothèses fondatrices des modèles à seuils, nous proposons un mécanisme d'interruption basé sur la motivation guide.

Motivation comme Mécanisme d'Interruption

La réévaluation systématique nous autorise à ne pas avoir de mécanisme d'interruption : dans le cas général, la fluctuation des stimulus internes et externes suffit à l'agent pour effectuer la bonne tâche. En revanche, la question se pose lorsqu'une tâche possède un stimulus déclencheur artificiel. Pour décider quand arrêter une telle tâche, nous avons

besoin de ce mécanisme d'interruption. Dans ce cas, nos agents vont essayer d'estimer leur apport à la communauté dans leur tâche actuelle. Ainsi, lorsqu'un agent verra qu'il n'arrive pas à réaliser sa tâche, il sera dans l'état de malaise décrit pas le *Flow* et cherchera de plus en plus à changer de tâche. Nous cherchons alors à mesurer l'efficacité de chaque agent dans sa tâche, afin de pouvoir détecter lorsqu'il arrive à la réaliser, mais surtout lorsqu'il n'y arrive pas. Il suffit alors de déterminer dans le contenu de la tâche quelles sont les Actions effectuées représentatives de son échec. Par exemple, un robot ayant pour tâche de récolter des ressources aura dans cet algorithme une séquence de déplacement aléatoire lorsqu'aucune ressource n'est en vue. Répéter en boucle ce déplacement aléatoire, cette Action, est un signe que ce robot n'arrive pas à correctement réaliser sa tâche, il devrait donc essayer de faire autre chose.

Nous proposons d'ajouter aux Actions définies plus tôt le fait de pouvoir être "Démotivantes" (notées **M-** dans nos schémas). Lors de l'exécution d'une Action Démotivante, un agent va baisser sa motivation interne (représentante de la Motivation Guide de l'agent) d'un montant défini. Le but est d'augmenter les chances qu'il abandonne cette tâche au profit d'une autre, lors du processus de sélection. Rien d'aléatoire, mais plus une tâche a un score élevé, moins il y a de chances qu'une autre tâche soit sélectionnée à sa place. En effet, lors du calcul du score d'une tâche, celui d'une Tâche Motivée est remplacé par la motivation interne de l'agent, seulement si c'est cette tâche que l'agent exécutait au pas de temps précédent. Ainsi, une Tâche Motivée est sélectionnée grâce à son score provenant du stimulus artificiel, mais est interrompue par une valeur de motivation interne plus basse, qui aura tendance à favoriser d'autres tâches. Lorsqu'une nouvelle Tâche Motivée est sélectionnée (et qu'elle n'était pas la tâche sélectionnée au pas de temps précédent), la motivation interne de l'agent est remontée à sa valeur maximale.

L'Action de déplacement aléatoire du robot que nous venons de citer est un bon exemple d'Action Démotivante. Une Action Démotivante doit forcément être dans une Tâche Motivée : si la motivation interne ne joue aucun rôle dans la sélection de la tâche, il n'y a aucun intérêt à la diminuer. Nous avons aussi fait le choix de ne pas intégrer de Tâches Motivantes dans le modèle, qui augmenterai la motivation interne de l'agent. Ce choix tient plus de positionnement, afin de garder un modèle simple, nous désirions en effet voir si remonter la motivation interne aux changements de tâches suffirait. La Section 4.4 aborde notamment nos discussions et perspectives sur ce point.

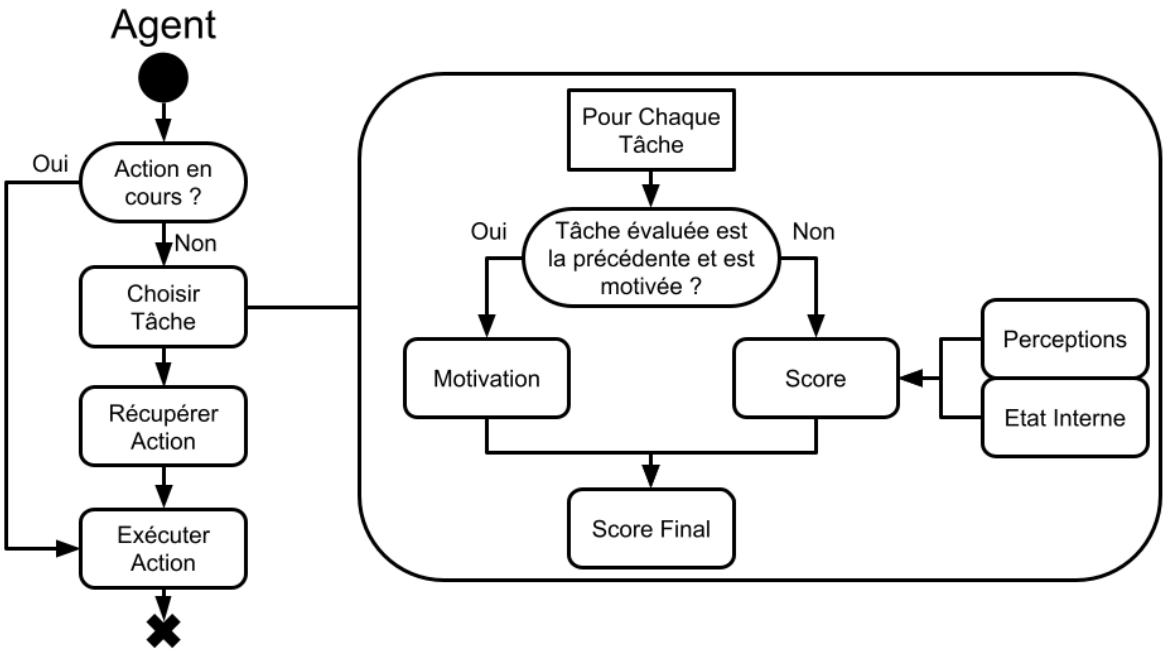


FIGURE 2.3 – Sélection et exécution des tâches par chaque Agent, à chaque pas de temps. Si l'Action en cours est terminée, l'agent va sélectionner une nouvelle tâche, en extraire l'Action à réaliser puis l'exécuter.

Dans la littérature nous trouvons les travaux de W. Agassounon [1], qui sont assez proches de ce que nous proposons ici. Ils proposent de mesurer le temps depuis lequel un agent n'a pas réussi à être productif. Il mesure l'efficacité de chaque robot dans la tâche de collecte de ressource en regardant depuis combien de temps chaque agent n'a pas attrapé ou déposé une ressource. Lorsque ce temps dépasse un seuil, l'agent arrête alors ce qu'il fait et déclenche sa tâche de repos. Lorsque le temps de repos passe à son tour au delà d'un certain seuil, l'agent reprends alors sa tâche de collecte de ressource. Ces transitions sont fixes et fonctionnent en couple, sa tâche de repos n'est sélectionnée que lorsque des agents ne parviennent pas à correctement réaliser leur tâche de collecte. Ainsi, notre modèle fonctionne sur le même principe mais étends la sélection et l'interruption aux modèles à seuils, permettant à de nombreuses tâches de fonctionner ensemble, sans liens explicites.

2.3 Définir un Agent

À l'aide de ces définitions nous pouvons désormais décrire nos agents. Un agent est situé dans l'environnement, possède une série de senseurs internes et externes (comme la faim et l'odorat), et contient aussi une liste de tâches qu'il sera peut-être amené à réaliser au court de sa vie. Lors d'une sélection de tâche, l'agent pourra confronter ses perceptions courantes aux différents seuils et conditions de l'algorithme de sélection de tâche, donnés par son état interne ainsi que son environnement direct, afin de savoir quelle Action effectuer. La Figure 2.3 reprend ces notions et décrit le comportement et la prise de décision d'un agent à chaque pas de temps : si l'agent n'a pas d'Action en cours, il sélectionne une nouvelle Tâche en fonction de son état interne, de sa motivation interne et de ses perceptions, récupère l'Action à réaliser de sa Tâche nouvellement sélectionnée grâce à son architecture de subsomption hiérarchique, puis l'exécute.

Parmi les perceptions internes de l'agent, nous trouvons une variable de motivation interne, servant aussi à la sélection de tâche. Associer une valeur de motivation à chaque tâche aurait aussi été possible et nous pouvons en trouver des équivalences dans d'autres travaux cités précédemment, nous avons donc décidé de tenter l'expérience avec une valeur transversale à toutes les tâches, liée à l'agent lui même, permettant de limiter le nombre de paramètres. La Section 4.4 aborde nos discussions sur ce point.

Un agent peut aussi présenter des variations individuelles, un léger *offset* que nous pouvons utiliser pour ajuster légèrement les seuils de ses différentes tâches, en plus des conditions internes et externes, afin de créer une population d'agents plus ou moins homogène. Via ses tâches, un agent possède des seuils variables qui lui sont propres : deux agents avec les mêmes tâches présentent le plus souvent des seuils différents.

2.4 Application possible à la Robotique en Essaim

Les systèmes multi-agents sont souvents utilisés dans la mise en place d'essaims de robots, ce qu'on appelle le domaine des "Swarm Robotics". Ces essaims consistent en une grande quantité (dizaines, voire centaines) de robots simples, amenés à exécuter des tâches complexes en collectivité. L'image de la capacité de fourragement des fourmis est souvent utilisée comme cas d'application, nous avons donc construit notre exemple dans ce contexte. Un ensemble de robots va devoir ramener des ressources à leur base commune.

Les ressources sont éparpillées dans l'environnement et doivent être traitées par un robot avant d'être déplacées jusqu'à la base. En plus de cette activité de collecte, les robots vont devoir assurer une surveillance de la base, en patrouillant autour. Ces robots possèdent une mémoire limitée : ils connaissent leur position ainsi que celle de la base, et peuvent se rappeler de la position de gisements de minerai qu'ils ont directement observés. Ils ne peuvent pas communiquer entre eux. Ils possèdent en revanche des capteurs leurs permettant de se voir entre eux.

De plus, nous ajoutons la notion d'outil : un robot devra posséder le bon outil pour exécuter une tâche, par exemple une pioche pour collecter des ressources, et des jumelles pour patrouiller. Les ressources brutes devront être traitées sur place avant de pouvoir être collectées puis amenées à la base. L'outil porté par un agent est visible des autre agents l'observant.

Nous pouvons dès lors commencer à construire nos tâches :

- **Patrouiller**, Tâche Motivée : le robot effectue des cercles larges autour de la base, observant les alentours. Il se démotive légèrement au fil du temps (noté M- sur la Figure 2.4), et un peu plus lorsqu'il croise un autre robot (noté M– sur la Figure 2.4). Ceci permet aux robots d'éviter de patrouiller si un grand nombre de robots le fait déjà. Le seuil est élevé, sauf si l'agent est équipé des jumelles.
- **Collecter**, Tâche Motivée : le robot parcourt l'environnement aléatoirement à la recherche d'un gisement. Une fois trouvé, il traite le gisement pour collecter des ressources, puis les ramène à la base. Il se démotive légèrement à chaque pas de temps où il exécute un déplacement aléatoire. Le seuil est élevé, sauf si l'agent est équipé d'une pioche.
- **Recharger** : le robot se connecte à la base pour recharger ses batteries.
- **Mémoriser** : lorsque le robot voit un gisement qu'il ne connait pas, il l'ajoute à sa mémoire. A l'inverse, lorsqu'il ne voit pas de gisement (ou qu'il voit un gisement épuisé) là où il en avait retenu un, il l'oublie. Ainsi, un robot qui patrouille peut découvrir de nouveaux gisements, tout autant qu'un robot qui en cherche activement un. Chaque robot pourra ensuite, dans sa tâche de collecte, utiliser cette mémoire pour se rendre directement au gisement.

Pour *Patrouiller* et *Collecter*, si l'agent active la tâche sans posséder le bon outil, la tâche commencera par le faire retourner à la base pour équiper le bon. Cet outil va alors changer les seuils de ces deux tâches, priorisant celle dont l'outil est le bon. Ainsi, un agent

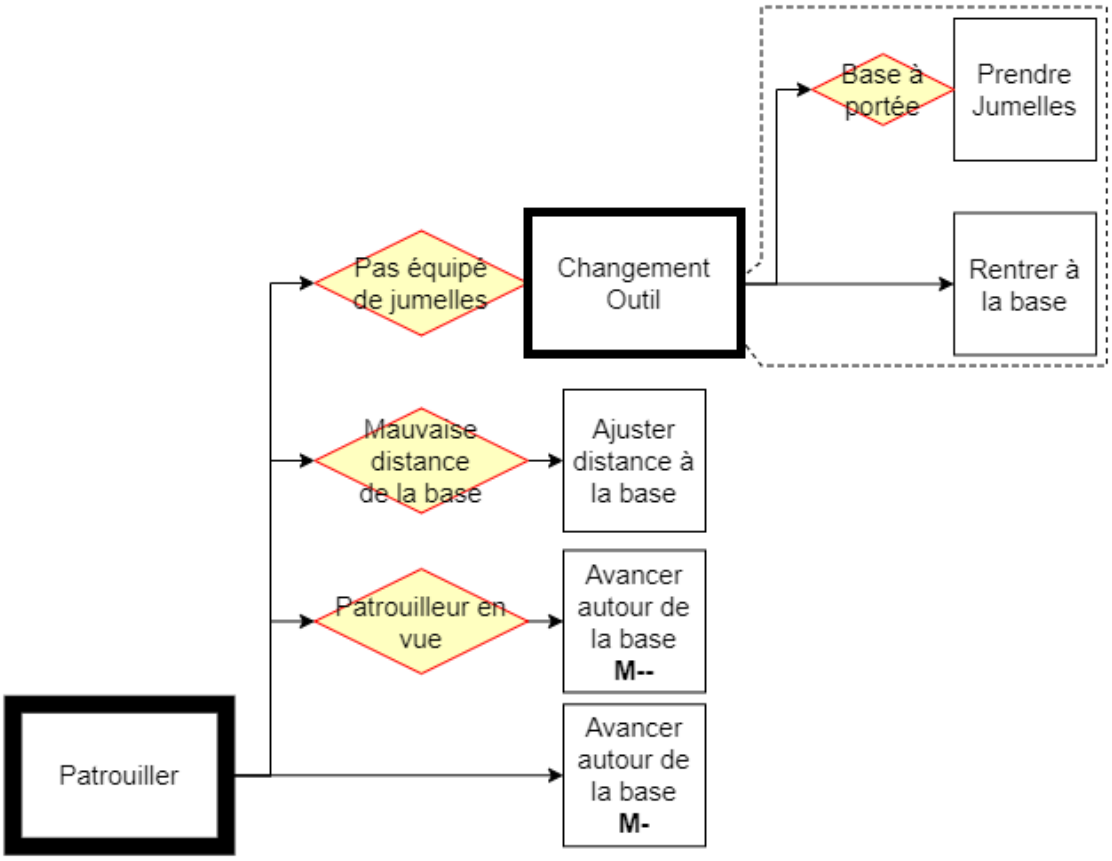


FIGURE 2.4 – Robotique en essaim : modélisation de la tâche de patrouille.

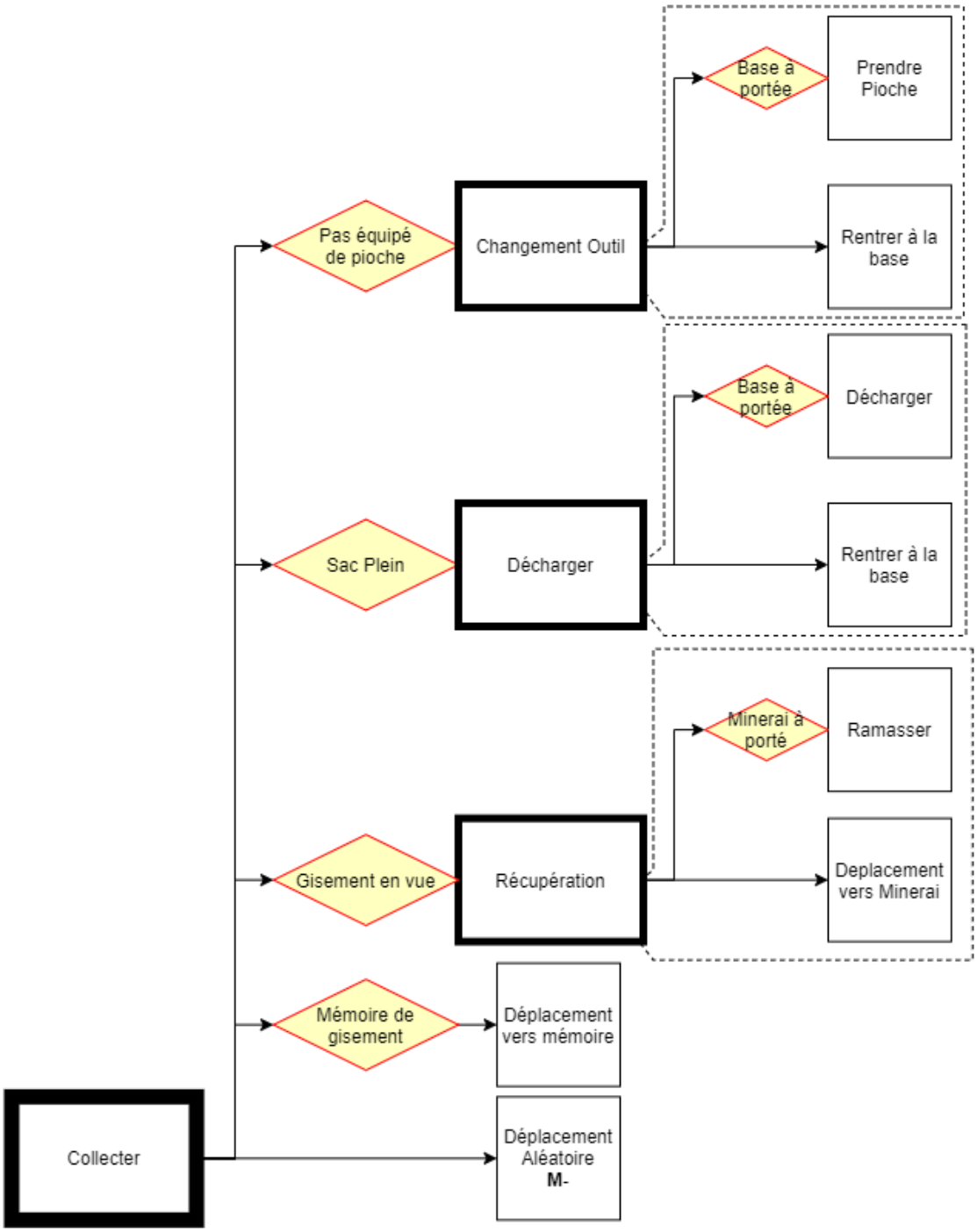


FIGURE 2.5 – Robotique en essaim : Modélisation de la tâche de collecte de ressources.

ne changera d'outil que lorsque cela sera nécessaire, les seuils ajoutent ici une notion de coût du changement d'outil. Ensuite, si nous le souhaitons, en ajustant les valeurs des seuils bas (prioritaires) et haut (changement d'outil nécessaire) et/ou la force répulsive des Actions Démotivantes, nous pouvons ajuster ce coût empiriquement. Les Figures 2.4 et 2.5 présentent respectivement nos modélisations en subsomption hiérarchique des tâches de patrouille et de collecte de ressources.

Ensuite, si *Recharger* a un stimulus déclencheur évident, le niveau courant de batterie, ce n'est pas le cas pour *Patrouiller* et *Collecter*. Nous leur appliquons donc un stimulus artificiel. Nous pouvons ensuite modifier légèrement ces stimulus artificiels avec des perceptions de l'agent : nous pouvons réduire légèrement cette stimulation pour la tâche *Patrouiller* lorsqu'un robot avec des jumelles est dans le champs de vision. De même, nous pouvons réduire la stimulation de *Collecter* lorsqu'un robot avec une pioche est en vue, et l'augmenter lorsqu'un gisement de minerai est en vue.

Cet exemple d'implémentation du modèle de répartition des tâches à fait l'objet d'un projet étudiant de Master 1.

Conclusion

Dans ce chapitre nous avons décrit notre modélisation des Tâches en Activités et Actions, ainsi que la modification des modèles à seuils pour notre sélection de tâches. Cette sélection se fait via un système à seuils adapté afin de prendre en compte les Motivations Source (nous venant plutôt de l'éthologie) et Guide (ou interne, nous venant de la théorie du *Flow*), afin d'inclure des tâches ne présentant pas de stimulus déclencheur défini. Un score est calculé par tâche selon les perceptions et l'état interne de l'agent, et ce dernier sélectionne la tâche qui possède le plus élevé. Lorsque la tâche précédemment réalisée est une Tâche Motivée, son score est remplacé par la Motivation Guide, interne à l'agent et transversale à toutes ses tâches. Une fois la tâche sélectionnée, l'agent utilise notre architecture de tâches en subsomption hiérarchique afin d'obtenir le comportement, l'Action, qu'il doit réaliser. Certaines Actions peuvent être Démotivantes : un agent qui exécute une Action Démotivante va réduire sa Motivation Guide, augmentant ainsi ses chances de changer de tâche au prochain pas de temps. On parle alors de mécanisme d'interruption. Nous avons décrit un exemple possible d'application à un essaim de robots, et nous allons désormais pouvoir aborder l'implémentation de l'application principale de ces travaux, la colonie d'abeilles et son simulateur.

Chapitre 3

Modélisation et Implémentation pour la Colonie d'Abeilles

Nous allons désormais aborder dans ce chapitre l'implémentation du modèle que nous venons de décrire, appliqué au cas qui nous intéresse ici : la colonie d'abeilles. Pour une première itération, nos abeilles auront pour objectif de correctement se répartir le travail entre deux tâches principales : Nourrir le couvain et Butiner. Nous allons commencer par discuter de notre simulateur et de son architecture logicielle, puis du modèle physiologique de l'abeille adulte et du couvain, tiré de la biologie mais fortement simplifié. Nous verrons ensuite comment nous avons intégré le modèle de répartition des tâches décrit dans le chapitre précédent. Nous finirons par discuter de la calibration de ce système complexe, au plus proche de la biologie, tout en prenant en compte les différentes hypothèses et simplifications de notre modèle.

3.1 Description du Simulateur

3.1.1 Architecture Logicielle

Pour réaliser ce simulateur, nous avions en tête quelques points clés : l'idée est de produire un simulateur propre, qui serait simple d'entretien et facile à améliorer et complexifier par la suite, au delà des travaux de thèse présentés ici. Un compromis entre confort d'écriture et performances qu'offre Java, ainsi que notre bonne maitrise du langage nous a poussé l'utiliser. Pour nous aider à mettre en place ces notions de propreté du code, nous avons utilisé des Interfaces Java, permettant de découpler des grande parties du programme, facilitant l'implémentation de modifications. Nous avons, le plus possible, pensé l'architecture en composants indépendants échangeant le moins d'informations possibles. Nous allons désormais décrire les principaux composants de cette architecture, la Figure 3.2 résume le tout graphiquement.

Le simulateur est pensé en couches d'abstraction successives. Au plus haut niveau, nous trouvons le lanceur, la classe *BeekeeperLauncher*, le célèbre "*main*", qui se charge des simulations. Il présente deux modes :

— Interactions : Ce mode permet de lancer une simulation et prends en compte les interactions de l'utilisateur (via le réseau) pour altérer le comportement de la simulation, et renvoie une importante quantité de donnée par ce même réseau.
— Barrage de Simulations : mode permettant de lancer un grand nombre de simulations à la suite, sans interactions possibles. Ce mode sert principalement pour calibrer certains paramètres.

Dans le mode "Interactions", le lanceur prépare le composant réseau *NetworkManager* qui servira à envoyer et recevoir les informations de l'application interactive, composant qui n'est pas instancié dans le mode "Barrage de Simulations". Le lanceur instancie ensuite le *MainController*, le contrôleur principal, et lui partage le composant réseau si besoin. Le *MainController* se charge de gérer une simulation. Désormais nous trouvons deux comportements différents pour les modes : en "Barrage de Simulations", les simulations s'enchainent à pleine vitesse, souvent en variant quelques paramètres. En mode "Interactions", lorsque la simulation est arrêtée, soit le programme s'arrête soit une autre est relancée avec les mêmes paramètres, selon ce qui a été décidé par l'utilisateur.

Le *MainController* gère une simulation, une colonie d'abeilles. Il se charge d'instancier correctement les différents composants d'une simulation et du bon déroulement de l'ensemble. Il compte les pas de temps, ce qui lui permet d'envoyer un signal aux agents, via un autre composant, leur ordonnant de vivre un pas de temps. Via une interface Java, il offre un grand nombre de services haut niveau à différents composants, notamment au *CombManager*, le manager des cadres, qu'il instancie avec les bons paramètres. Le *CombManager* se charge d'instancier le bon nombre de faces de cadres, qui seront associées par deux pour former les cadres. Il instancie ensuite les agents initiaux et les réparti sur ces cadres, via un autre composant, l'*AgentFactory*, qui simplifie et centralise la création d'agents. La Figure 3.1 illustre les différentes interactions entre composants logiciels lors de l'initialisation d'une simulation.

Chaque face de cadre possède une liste de cellules, des *Cell*, qui elles même possèdent quelques attributs : leur numéro et leur position x et y sur le cadre ($numéro = y * largeur + x$), ainsi qu'une place "visite" pour un agent qui serait au dessus de la cellule, et une place "contenu", pour un agent à l'intérieur de la cellule ou de la nourriture, ou même rien, pour une cellule vide. Chaque face de cadre possède une liste contenant tous

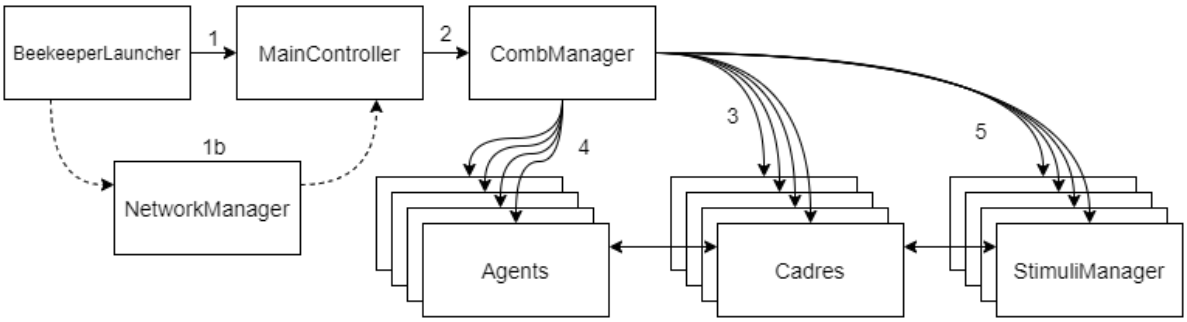


FIGURE 3.1 – Sequencement de l'initialisation d'une simulation.
1- Le lanceur *BeekeeperLauncher* instancie et initialise le *MainController* qui se chargera d'une simulation. 1b- Dans le cas où le lanceur est dans le mode Interactions, il instancie un *NetworkManager* et en donne la référence au MainController (Lors d'un changement de simulation, le *MainController* est remplacé mais le *NetworkManager* reste le même). 2- Le *MainController* se charge désormais d'instancier un *CombManager* avec les bons paramètres. 3- Le *CombManager* commence par initialiser les *Cadres*, il instancie en réalité des faces de cadres qu'il va associer par deux pour former les cadres de la simulation. 4- Une fois créés, les *Cadres* sont alors peuplés par le *CombManager* avec des *Agents*. 5- Ensuite, des *StimuliManager* sont instanciés par le *CombManager* et associés à des faces de cadres.

les agents que contiennent ces cellules. Cette liste agit comme un raccourci, et permet de ne pas avoir à interroger toutes les cellules à chaque fois que nous voulons accéder aux agents. Le *CombManager* possède aussi une liste d'agents, ceux qui n'appartiennent à aucun cadre : la liste de tous les agents à l'extérieur de la ruche, les butineuses.

Le *CombManager* est aussi responsable de la création, mise à jour, et bonne association des *StimuliManager*, les managers de propagation des différents stimulus dans l'environnement. Les cadres sont placés les uns à côtés dans autres, les faces se faisant face partagent le même *StimuliManager*.

Un *StimuliManager* est une grille 2D de la même taille qu'un cadre, dont chaque case possède plusieurs flottants, représentant les intensités des différents stimulus présents sur le cadre. Sa mise à jour utilise une méthode de "*double buffering*" : la grille est lue et une deuxième est remplie avec les nouvelles valeurs. Une fois la lecture terminée, la deuxième grille prend la place de la première. La propagation se fait à chaque pas de temps pour chaque stimulus et, afin de gagner du temps, l'évaporation se fait en même temps que le calcul de la propagation. Pour propager les stimulus, nous utilisons une fonction proche d'un flou en traitement d'image. Chaque pixel devient une moyenne pondérée de lui même et de tous ses voisins. Pour nous, chaque cellule voit sont intensité devenir la moyenne

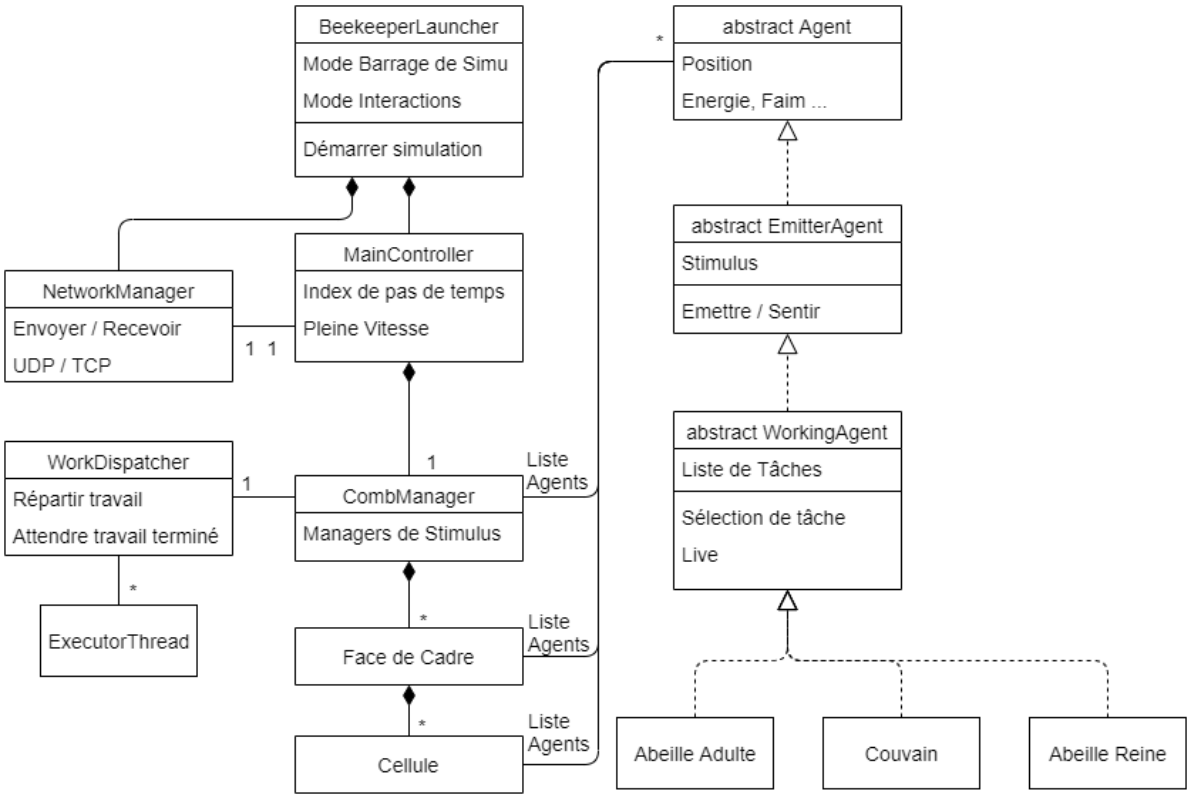


FIGURE 3.2 – Diagramme de classe esquissant l'architecture logicielle du simulateur.

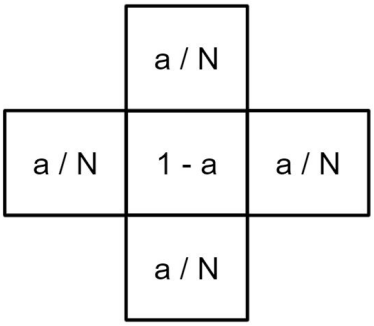


FIGURE 3.3 – Filtre utilisé pour connaitre la nouvelle valeur de l'intensité du stimulus évalué. Avec a la volatilité du stimulus évalué, et N le nombre de voisin, ici $N = 4$. Chaque pixel prend alors comme valeur la combinaison linéaire de sa valeur et de celles de ses voisins avec ce filtre pour coefficient.

pondérée de sa propre intensité et de celles de ses voisines, en suivant cette équation :

$$V0_{t+1} = p * ((1 - a) * V0_t + \sum_{n=1}^{N} Vn_t * \frac{a}{N}) \tag{3.1}$$

avec $V0_t$ la cellule évaluée à l'instant t, Vn ses voisines et N le nombre total de voisines. Nous utilisons aussi a et p deux paramètres de stimulus : la propagation dans l'espace a (volatilité) et dans le temps p (évaporation/amortissement). Ces paramètres nous permettent d'obtenir des comportements variés partageant les mêmes mécanismes. La figure 3.3 illustre cette équation sous la forme d'un filtre. Les valeurs d'évaporations sont alors utilisées sous la forme "par pas de temps". Pour mieux les contrôler, nous utilisons l'équation suivante pour les exprimer en demie-vie (durée après laquelle le stimulus aura perdu la moitié de son intensité), notion plus courante en biologie, puis les convertir en valeurs "par pas de temps" utilisables par l'algorithme :

$$k = \exp(\frac{-ln(2) * C}{\lambda}) \tag{3.2}$$

avec λ la demie vie en seconde, C le coefficient appliqué pour convertir des secondes en pas de temps, et k le coefficient à appliquer à chaque pas de temps pour obtenir une demie-vie de λ.

3.1.2 Architecture des Agents

Les mêmes objectifs en tête, lisibilité et facilité d'ajouts, nos agents ont été pensés en niveaux d'abstraction successifs. Au plus haut niveau nous trouvons la classe abs-

traite *Agent*, qui regroupe l'âge, la position, l'énergie, la faim, une fonction abstraite *live* (vivre), ainsi que des fonctions de déplacement simples, comme le mouvement aléatoire. Nous définissons ensuite la classe *EmitterAgent*, ou Agent Émetteur, qui hérite d'*Agent* et ajoute les interactions avec un *StimuliManager*, permettant d'émettre, ressentir des stimulus dans l'environnement. Enfin, héritant d'*EmitterAgent*, nous trouvons la classe *WorkingAgent*, ou Agent Travailleur, qui implémente enfin *live*, avec l'algorithme de sélection de tâches présenté Chapitre 2 et détaillé section 3.1.4. La Figure 3.2 présente aussi l'architecture des agents. La fonction *live* se déroule en quelques étapes. D'abord on vérifie si l'agent est toujours en vie, ensuite nous mettons à jours ses perceptions. L'algorithme de sélection et exécution des tâches est alors appliqué, puis une fonction abstraite est appelée, fonction permettant de faire avancer le métabolisme de l'agent, et qui est implémentée différemment par nos différentes implémentations de *WorkingAgent*. Nous avons pour l'instant 3 classes implémentant *WorkingAgent*, les classes d'abeille adulte *AdultBee*, de couvain *BroodBee*, et de reine *QueenBee*. Chacune de ces implémentations ajoute différentes tâches à sa liste de tâches (que nous allons décrire Section 3.2), et implémente la fonction d'avancée du métabolisme.

Ainsi, dans les plus hautes couches d'abstraction, les différents composants ont des références *Agent*, ce qui améliore la modularité du programme, facilitant l'ajout de nouvelles implémentations de *WorkingAgent* à l'avenir.

3.1.3 Pas de temps et Multi-Thread

Afin d'accélérer nos simulations pour d'itérer plus confortablement dessus, nous avons mis en place un système de parallélisation, aussi appelé *multi-thread*. Chaque duo de faces de cadres se faisant face, ainsi que le groupe des butineuses, vivent en concurrence, car ces différents ensembles n'interagissent pas entre eux. Pour ceci nous avons créé une classe *WorkDispatcher*, qui s'occupe de gérer un groupe d' *ExecutorThread*, de leur répartir le travail, d'en créer de nouveaux si nécessaire et de surveiller le moment où tous ont terminé leur travail. À chaque pas de temps, sur signal du *MainController*, le *CombManager* récupère et combine si nécessaire les différents groupes d'agents à faire vivre ensemble. Il envoie alors ces ensembles au *WorkDispatcher* qui redirige la liste sur un *ExecutorThread* disponible. Chaque *ExecutorThread* va alors appeler la fonction *live* de chacun des agents de sa liste. La Figure 3.4 illustre ces échanges.

Pour le cas particulier des butineuses rentrant dans la ruche, ou en sortant, deux listes synchronisées spéciales ont été ajoutées au niveau du *CombManager*. Une liste contenant

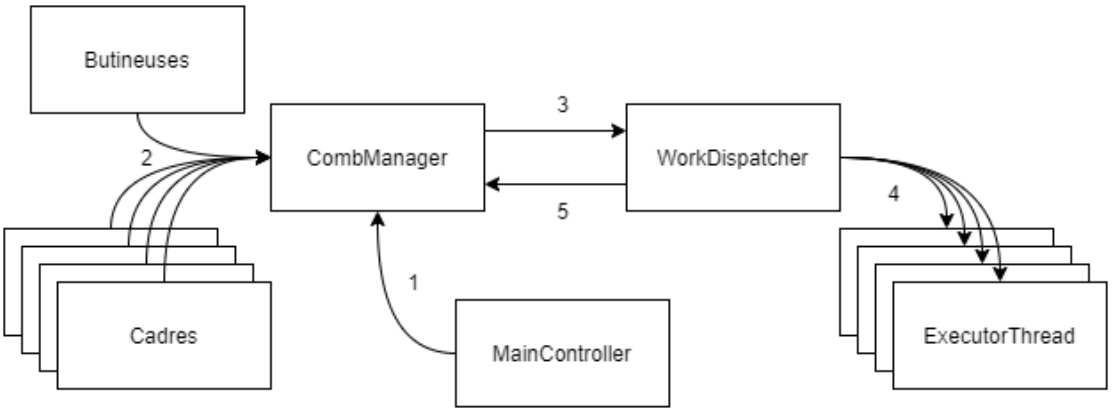


FIGURE 3.4 – Sequencement de la partie d'un pas de temps concernant les agents. 1- Le *MainController* envoie le signal au *CombManager* de faire vivre les agents d'un pas de temps. 2- Celui-ci va alors récupérer les listes d'agents présents sur les cadres ainsi que celle des butineuses. 3- Il réarrange ensuite ces listes en rassemblant les agents ne pouvant pas vivre en concurrence, puis envoie ces listes aux *WorkDispatcher*. 4- Ce-dernier va ensuite envoyer ces listes à des *ExecutorThread* et en instancier de nouveaux s'il le faut. Ils vont alors, chacun en concurrence, parcourir leur liste d'agents et les faire vivre chacun leur tour. 5- Enfin, lorsque le *WorkDispatcher* détecte que tous les threads ont terminé de faire vivre leurs agents, il le signal au *CombManager* qui rend alors la main au *MainController*.

toutes les abeilles qui sont sorties de la ruche sur ce pas de temps, et une liste contenant celles qui sont rentrées. Les butineuses et les abeilles des cadres ne vivant pas sur les même *threads*, une abeille rentrante ne peux pas être ajoutée directement au cadre. Ce n'est qu'après avoir fait vivre tout le monde, que le *CombManager* se charge de faire les transferts, déplaçant les abeilles sortantes vers la liste des butineuses, et les abeilles entrante vers un cadre libre aléatoire.

Ensuite, lorsque tous les agents ont vécus, et que les déplacements entre cadres ont été réalisés, le *CombManager* rend la main au *MainController* qui se charge alors de terminer le pas de temps. Dans le mode "Barrage de Simulations", le pas de temps suivant est exécuté juste après, de la même manière que le précédent. En revanche, en mode "Interactions", le *MainController* va attendre une seconde (dans le cas où un pas de temps correspond à une seconde, ce qui est notre cas), en en déduisant le temps de calcul du pas de temps courant. Ainsi, nous respectons notre simulation en temps réel où un pas de temps représente une seconde réelle, et l'assurant précisément tant que la durée de calcul d'un pas de temps ne dépasse pas la seconde. Dans ce cas théorique le *MainController* enchainerait directement avec le pas de temps suivant, sa rapprochant ainsi autant que possible de notre temps réel voulu. Ce temps réel est nécessaire pour permettre à l'utilisateur d'interagir avec les agents sans créer de biais dans la simulation. Lorsque l'utilisateur décide d'accélérer le temps pour observer l'état futur de la colonie, le *MainController* cesse de faire ces pauses entre les pas de temps, comme dans le mode "Barrage de Simulations", ce qui permet de grandement accélérer les calculs, mais interdisant toutes interactions (ainsi, simuler 80 jours, au lieu de prendre quelques mois, ne prends que quelques minutes).

3.1.4 Architecture des Tâches

Nos tâches sont décrites en subsomption hiérarchique dans le modèle, nous les avons donc implémenté de la sorte dans le simulateur. Au sommet de la hiérarchie nous trouvons la classe *Tâche*, associant un seuil, un nom de tâche, différents paramètres et une activité racine. Une *Tâche* est une classe abstraite : toute classe que nous voudrons instancier devra implémenter la fonction qui permet de calculer son score, et peupler l'Activité racine d'Actions et/ou d'Activités.

La classe *TaskNode*, ou Nœud de Tâche, est une interface très simple contenant deux fonctions : *search* (recherche) et *check* (vérifier). Elles nous permettent de mettre en place le fonctionnement de subsomption. La fonction *check* représente la condition de chacun des

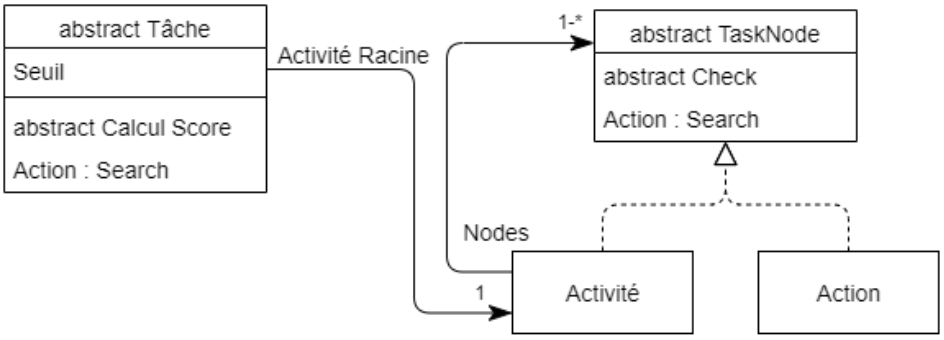


FIGURE 3.5 – Diagramme de classe de l'architecture logicielle de Tâche.

blocs de notre subsomption, condition booléenne que chaque Nœud devra implémenter. Ensuite, la fonction *search*, la principale, diffère selon les nœuds. La classe *TaskNode* est implémentée par deux classes, *Activité* et *Action*, respectant notre modèle vu plus tôt. Vous trouverez Figure 3.5 un diagramme de classe de cette architecture. La fonction *search* renvoi toujours une *Action*, et permet d'interroger récursivement toutes la subsomption hiérarchique. La fonction *search* de la classe Action renvoi l'instance d'*Action* sur laquelle elle est appelée. En revanche, une *Activité* possède une liste de *TaskNode*, permettant la mise en place de la subsomption hiérarchique. Dans la fonction *search*, l'*Activité* va interroger une par une (dans l'ordre de priorité de la subsomption) la fonction *check* de tous ses nœuds. Si l'un d'eux valide son *check*, l'*Activité* appelle alors le *search* du nœud validé, continuant la recherche en profondeur si c'est appelé sur une Activité, ou mettant ainsi fin à la recherche lorsqu'appelé sur une Action.

Ainsi, pour qu'un agent puisse récupérer l'*Action* à effectuer de sa *Tâche* sélectionnée, il lui suffit d'appeler *search* sur l'*Activité* racine de la *Tâche*. Récursivement, la recherche va descendre dans la subsomption dans les blocs aux conditions validées. Dès qu'une *Action* est trouvée, sa fonction *search* la renvoie et elle est remontée dans toutes les fonctions *search* appelées précédemment, jusqu'à ressortir au niveau du tout premier *search* que nous avons appelé sur l'*Activité* racine de la *Tâche*. L'*Action* est prête à être exécutée par l'agent.

Pour conclure, lors d'un pas de temps, le contrôleur principal demande au manager des cadres de faire vivre tous les agents. Ce-dernier parcourt alors toutes ses faces de cadres et récupère tous leurs agents, afin de les envoyer aux gestionnaire de *thread* pour une exécution parallèle. Une fois tous les agents mis à jour, les agents entrant et sortant de la

ruche sont transférés. Si le contrôleur principal a demandé d'enregistrer le tour, alors une nouvelle mécanique s'enclenche. Le manageur des cadres récupère à nouveau la liste de tous les agents, et leur demande de décrire en une ligne leur état : numéro d'identifiant, tâche, énergie, HJ et EO. À cette ligne est ajoutée au début le numéro du pas de temps transmis par le contrôleur principal. Toutes ces lignes sont alors envoyées au *Logger*, qui les transfère sur un autre *thread* afin d'être écrites dans un fichier, permettant une trace de la simulation, pour utilisations ultérieures en analyse de résultat.

3.2 Modélisation de la Colonie d'Abeilles

3.2.1 Modélisation des Agents "Abeille Adulte"

Dans notre colonie virtuelle, les adultes (qui sont désormais des agents) tendent toutes à aller butiner rapidement, grâce à l'Hormone Juvénile (HJ) qu'elles sécrètent, mais sont retenues "nourrices" par l'Ethyle Oléate(EO) émise par le couvain et les butineuses, qui "détruit" une part de leur HJ. Les détails de la biologie de l'abeille, des phéromones en jeu et leurs effets sur l'auto-organisation sont décrits dans la section **??**sectionBio), au début de ce manuscrit. Cette rétroaction permet de réguler le nombre de nourrices et de butineuses au sein de la colonie : lorsqu'il y a beaucoup de couvain, très peu d'adultes vont partir butiner, car le couvain va injecter de grande quantité d'EO dans les agents de la colonie, et certaines butineuses peuvent même redevenir des nourrices. Lorsque les butineuses meurent de vieillesse, elles freinent moins le développement des nourrices, et certaines pourront partir butiner. Ces différentes interactions ont été synthétisées Figure 3.6. Dans les colonies réelles, les butineuses échangent très régulièrement des phéromones avec les receveuses, qui sont chargées de délester les butineuses de leur cargaison pour aller les stocker dans un endroit adapté dans la ruche. Nous posons une hypothèse ici, qui est que les receveuses sont un vecteur majeur de l'effet rajeunissant des butineuses sur les nourrices. Or, comme nous ne simulons pas de tâches de receveuses dans cette itération du modèle, nous nous attendons à ne presque pas observer ce mécanisme.

L'Hormone Juvénile (HJ) représente directement la physiologie de nos agents abeilles adultes, ce que nous appelons l'âge physiologique en référence au polyéthisme d'âge que nous observons dans les colonies réelles. Un agent avec très peu d'HJ ($HJ < 0.5$), sera capable de nourrir le couvain mais incapable de butiner. À l'inverse, un agent avec un fort taux d'HJ ($HJ > 0.5$), sera capable de butiner mais incapable de nourrir le couvain (ces

mécanismes sont décrits section). On parle donc de nourrices lorsque nous considérons les agents physiologiquement jeunes, et de butineuses pour les agents physiologiquement âgés. Nous avons ajouté un paramètre à nos agents, leur développement ovarien qui est toujours nul sauf la reine pour qui il vaut 1. Cette variable n'est pas utilisée dans cette itération de l'implémentation, mais permettra de simuler les ouvrières pondeuses que nous retrouvons dans la réalité, lorsque les phéromones de la reine ne parviennent plus à certaines ouvrières en périphérie de la colonie et que ces dernières commencent à pondre.

Dans nos lectures, nous avons pu apprendre que les abeilles d'hiver pouvaient vivre jusqu'à une année entière, mais que les butineuses meurent en une trentaine de jours, avec en moyenne les dix derniers jours passés à butiner. Les biologistes pensent que le vol est une activité très épuisante, et qu'il est possible qu'il réduise fortement l'espérance de vie des butineuses. Nous avons donc implémenté ce mécanisme pour les morts de vieillesses de nos agents. Ils meurent en une année, mais subissent une pénalité lorsqu'ils butinent. Ainsi, un agent qui butine voit son âge effectif (celui qui est utilisé pour déterminer la mort de vieillesse) augmenter trente fois plus vite qu'un agent à l'intérieur de la colonie.

Nos agents sont placés sur le cadre, savent que la sortie s'effectue par le bas (et savent aller vers le bas). Le travail de butinage est laissé très simple, le fourragement ne faisant pas partie de nos priorités ici. En effet, ce mécanisme passionnant fait déjà l'objet de grandes quantités de recherches, et nous nous concentrons ici sur l'intérieur de la ruche. À l'avenir, une simulation plus poussée des mécanismes de fourragement, sélection des sources de nectar, recrutement etc. pourraient être ajoutés au modèle (nous avons travaillé sur un modèle de butinage [26] en parallèle de ces travaux, qui pourra tout à fait être intégré par la suite). Pour l'instant, les butineuses sortent de la ruche, attendent simplement pendant un nombre donné de pas de temps puis rentrent à nouveau dans la ruche, sur un cadre aléatoire possédant une case disponible au niveau de sa ligne la plus basse. La durée de ce butinage est pour l'instant la même systématiquement, pour tous les agents, et correspond à une valeur moyenne des temps de butinages observés pour des butineuses réelles.

3.2.2 Modélisation du Couvain

Nous avons modélisé les trois étapes majeures de la vie du couvain : œuf, larve et nymphe. Seule la larve requiert de la nourriture, les deux autres étapes n'ont pas besoin de nourrice pour se dérouler correctement. La nymphe n'émet aucune phéromone, la cellule étant fermée et l'EO que nous avons modélisée étant transmise par contact. Un agent larve émet de l'EO en permanence, nous avons aussi fait le choix de faire émettre

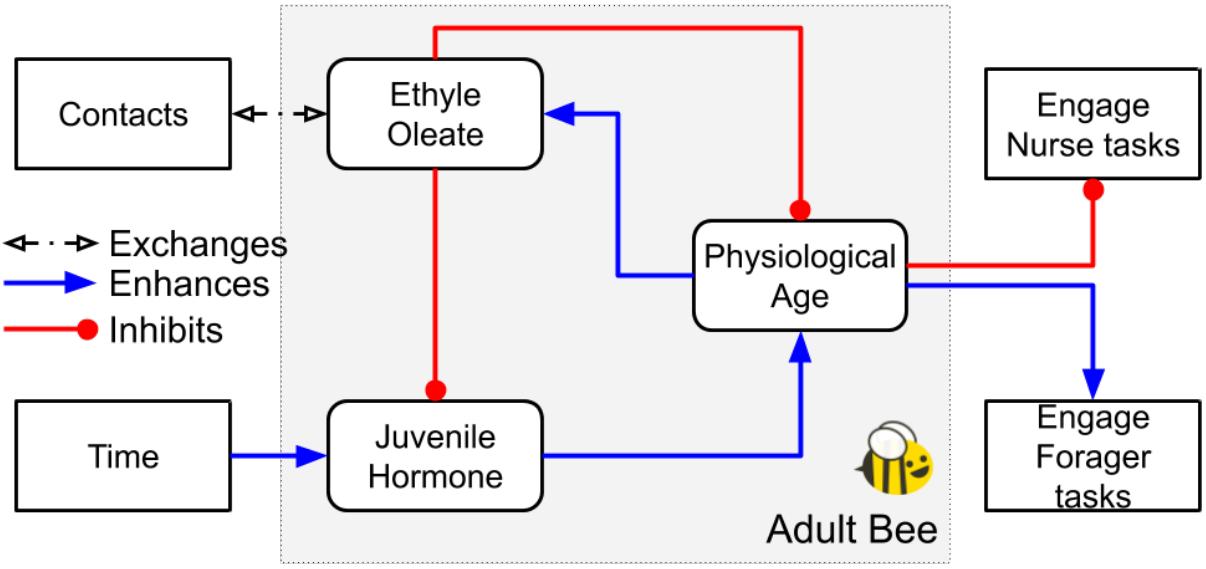


FIGURE 3.6 – Modélisation simplifiée des effets physiologiques responsables de l'auto-organisation.

ces phéromones par les agents œufs. Plusieurs aspects ont motivé ce choix, que nous n'avons pas pu confirmer ou infirmer en biologie : l'œuf est pondu par la reine qui émet des phéromones très puissantes, elle en transmet donc surement à l'œuf pendant la ponte. De plus, n'ayant pas simulé les receveuses, le mécanisme de rajeunissement des nourrices par les butineuses est minoré, l'œuf permet donc de compenser légèrement ce biais. L'œuf ayant une durée de vie très courte, seulement 3 jours sur les 21 totaux avant l'émergence, il est tout à fait possible que cette émission n'ait quasiment aucun effet.

Les phéromones émisent par les agents du couvain le long de leur vie vont altérer la physiologie des agents adultes. Elles sont échangées lorsqu'un agent adulte passe sur la case d'un agent couvain, et en plus grande quantité lorsqu'un agent effectuant la tâche de nourrisse vient déposer de la nourriture à un agent larve, du fait du contact prolongé avec celle-ci.

3.2.3 Tâches et Auto-Organisation

Le but de notre simulation est de retrouver un équilibre dans le partage des tâches entre ces deux tâches clés : Butiner et Nourrir le couvain. Ces deux tâches ne présentent pas de stimulus déclencheur direct comme nous avons pu en discuter dans le chapitre précédent. Nous avons donc recours à une motivation source pour ces deux tâches, que

nous plaçons arbitrairement à 0.5. Cette motivation source est abaissée à 0 lorsque l'abeille n'est pas physiologiquement apte a réaliser la tâche, ou, pour le butinage, si l'agent n'a pas suffisamment d'énergie pour survivre le vol aller-retour. Ensuite, nous utilisons l'HJ de chaque agent pour ajuster le seuil de chacune de ces deux tâches. Moins une abeille a d'HJ, plus elle aura de chance de sélectionner la tâche de nourrice, et plus une abeille aura d'HJ, plus elle aura de chance de sélectionner la tâche de butineuse (La description des modèles à seuils se trouve Section 1.1.2). Le débattement des seuils a été ajusté pour empêcher le score de la tâche de dépasser 0.8 : l'intervalle $[0.8; 1]$ est réservé aux tâches prioritaires, que nous allons maintenant décrire. En effet, si nos agents doivent se répartir deux tâches principales, ce ne sont pas les seules, nous avons ajouté des tâches d'entretiens : se reposer, donner à manger, demander à manger. Ces tâches ne sont pas motivées, elles possèdent des stimulus déclencheurs : la faim et l'énergie. La tâche de repos est prioritaire, son score est donc autorisé à dépasser 0.8. En effet, si l'énergie d'un agent devient négative, il meurt [1]. Le couvain possède aussi trois tâches très simple, une pour chacune de ses étapes de développement. Ces différentes tâches altèrent seulement leurs émissions de phéromones (en effet, le seul rôle du couvain est de se nourrir). La reine possède une tâche lui permettant de pondre. La Table 3.1 présente un récapitulatif de toutes les tâches implémentées.

3.3 Calibration des Phéromones

La répartition des tâches au sein de la colonie, en particulier le nourrissage et le butinage, dépendent majoritairement de mécanisme d'hormones et de phéromones. Comme nous l'avons vu plus tôt, nous nous intéressons ici au bras de fer entre l'Hormone Juvénile (HJ) et l'Ethyle Oléate (EO), comme décrit Figure 3.6.

Paramétrer cette dynamique a été un processus complexe : notre grande simplification du modèle, et surtout des différentes phéromones nous détache, pour cette partie, de la réalité biologique. Nous avons donc émis des hypothèses, fixé des paramètres arbitrairement, dans le but de retrouver d'autres "macro-paramètres" émergents. Les deux points clés de cette paramétrisation sont 1. la quantité relative d'EO émise par rapport à l'HJ, et 2. la

1. Dans cette version du modèle nous avons fait le choix de ne représenter la nourriture qu'au plus simple. Ainsi, la mort par sous alimentation ne fait pas partie de cette itération. L'importance des butineuses en est donc fortement réduite : même si 100% de la colonie s'occupe du couvain, il y aura toujours suffisamment de nourriture. Ajouter la nourriture, détailler ses mécanismes de distribution et introduire son mécanisme de collecte est une des perspectives privilégiées pour la suite.

TABLE 3.1 – Les différentes tâches que nos agents peuvent exécuter.

Nom de tâche	Calcul du score
Tâches d'entretien	
Se reposer	1-Energie [0 ;1]
Demander à manger	Faim [0 ;0.8]
Donner à manger	Stimulus de demande détecté [0 ;0.8]
Déplacement aléatoire	0.2 (tâche par défaut)
Tâches Principales (et motivées)	
Butiner	0 si $HJ < 0.5$, sinon Sigmoïde seuil : 1-JH déplacé dans [0.3 ;1]
Nourrir le couvain	0 si $HJ > 0.5$, sinon Sigmoïde seuil : JH déplacé dans [0.3 ;1]
Tâches du couvain et de la reine	
Tâche d'oeuf	1 si âge d'oeuf, 0 sinon (le couvain n'a pas besoin de repos)
Tâche de larve	1 si âge de larve, 0 sinon
Tâche de nymphe	1 si âge de nymphe, 0 sinon
Pondre	0.8 si développement ovarien, 0 sinon

force des effets de l'HJ et de l'EO. De plus, la paramétrisation s'articule autour de deux points d'équilibre. Le premier, l'équilibre en EO (EO_{Eq}), qui représente le moment où un agent a suffisamment d'EO pour parfaitement compenser son vieillissement, et donc son émission d'HJ. L'autre point d'équilibre, cette fois en HJ (HJ_{Eq}), représente le moment où un agent sécrète la bonne quantité d'HJ pour parfaitement compenser l'évaporation d'EO, et donc maintenir le niveau de cette-dernière.

3.3.1 Hypothèses et décisions arbitraires

Pour l'instant, les quantités absolues des différentes substances ne sont en rien liées à la réalité, nous avons donc pu choisir arbitrairement les quantités et points d'équilibres. Il sera peut être intéressant par la suite de correspondre aux quantités relevées sur les abeilles réelles, mais cette approche ne présentait pas d'intérêt, par rapports à la difficulté apportée, pour cette première version. Nous avons donc décidé de fixer la quantité d'HJ dans $[0; 1]$, et $HJ_{Eq} = 0.8$. De son côté, la quantité d'EO est simplement comprise dans $[0; \infty[$, et avec $EO_{Eq} = 1$.

Ainsi, un agent abeille adulte avec un taux d'HJ supérieur à HJ_{Eq} va émettre plus d'EO qu'il ne s'en évapore sur lui, permettant à l'EO de s'accumuler, le faisant potentiellement rajeunir. Un agent avec un taux d'EO inférieur à EO_{Eq} va éliminer moins d'HJ qu'il n'en émet, et va donc vieillir.

Nous faisons ici l'hypothèse que la réduction d'HJ par l'EO se fait seulement avec une fonction prenant en compte la quantité d'EO. De même pour l'émission d'EO en fonction de la quantité d'HJ qui ne se fait que via une fonction dépendante de la quantité d'HJ de l'agent concerné. Nous posons aussi que ces fonctions ont la forme x^n, avec n un paramètre fixé expérimentalement (que nous décrirons plus tard) et x la différence entre le taux courant et la valeur d'équilibre donnée plus tôt.

Les deux effets énoncés précédemment sont imbriqués dans une boucle de rétroaction. Les quantités absolues de ces éléments ne sont donc pas pertinentes, en revanche, les écarts entre ces deux produits provoquent la dynamique que nous recherchons. Nous avons ainsi décidé, pour simplifier le paramétrage, d'en fixer un et de chercher le deuxième. Nous avons donc fixé l'émission d'EO en fonction de l'HJ avec la fonction linéaire :

$$EO_{em} = (HJ - HJ_{Eq}) * EO_{Evap} \tag{3.3}$$

avec EO_{Evap} la quantité d'EO qui s'évapore lorsque l'on considère une quantité d'EO égale à EO_{Eq}. Nous retrouvons donc la fonction sous la forme $s = x^n$ décrite plus tôt, avec $n = 1$ et x l'écart d'HJ à l'équilibre facteur de EO_{Evap}.

3.3.2 Intensités des effets

Après avoir fixé ces paramètres, nous devons nous atteler à trouver la bonne combinaison d'intensité des effets des substances, ainsi que la quantité à laquelle on veut les émettre. Pour l'HJ, il suffit de prendre le point de transition, nous avons choisi 0.5, et de le diviser par la quantité de pas de temps minimum qu'il faut pour l'atteindre. Nous pourrions décider de la majorer légèrement afin de prendre en compte l'effet de l'EO, ralentissant très légèrement le vieillissement en faible quantité. Ce ralentissement est cependant très faible pour un agent isolé, ainsi nous n'avons pas gardé l'option de la majoration.

Nous avons ainsi la quantité d'HJ émise par chaque agent à chaque pas de temps, que nous appellerons désormais HJ_{Incr}. L'EO présente sur chaque agent viendra éliminer une partie de son HJ, combattant ainsi indirectement cet HJ_{Incr}, dont nous pouvons désormais nous servir comme référence. Nous voulons qu'à EO_{Eq}, la réduction d'HJ HJ_{red} soit égale à HJ_{Incr}, pour compenser parfaitement le vieillissement de l'agent. Nous pouvons donc écrire :

$$HJ_{red} = (EO - EO_{Eq})^n * HJ_{Incr} \tag{3.4}$$

avec n un coefficient dont nous allons nous servir pour modeler l'intensité de l'effet de réduction d'HJ. Avec $n = 1$ (Figure 3.7a), notre fonction est linéaire, nous nous servirons de cette fonction comme base de comparaison. Lorsque $n > 1$, on obtient une fonction quadratique (Figure 3.7b). L'effet rajeunissant de l'EO est diminué sous EO_{Eq} et amplifié au delà. Nous obtenons donc un vieillissement ainsi qu'un rajeunissement plus rapide. L'effet de l'EO est amplifié lorsque n augmente. Enfin, avec $n < 1$, on obtient une fonction racine (Figure 3.7c). De la même manière, avec une racine, l'intensité des effets de l'EO est réduite.

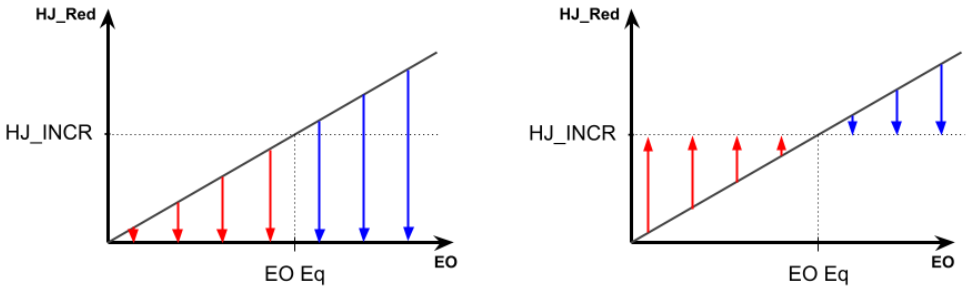
Pour illustrer, nous pouvons aussi imaginer que l'abeille vieillit naturellement rapidement. Mais, cet état d'équilibre est altéré par l'EO, nous pouvons donc visualiser ceci comme l'abeille étant attachée par un ressort à ce point d'équilibre. La rigidité de ce ressort peut alors être interprétée comme la puissance de l'EO, proportionnelle à n. Plus n est grand, plus l'abeille sera tirée rapidement vers ce point d'équilibre.

Nous n'avons pas trouvé de moyen de fixer mathématiquement n, ni dans la littérature, ni même après mûres réflexions : il a donc été trouvé expérimentalement, nous y viendrons dans la partie suivante, concernant la calibration.

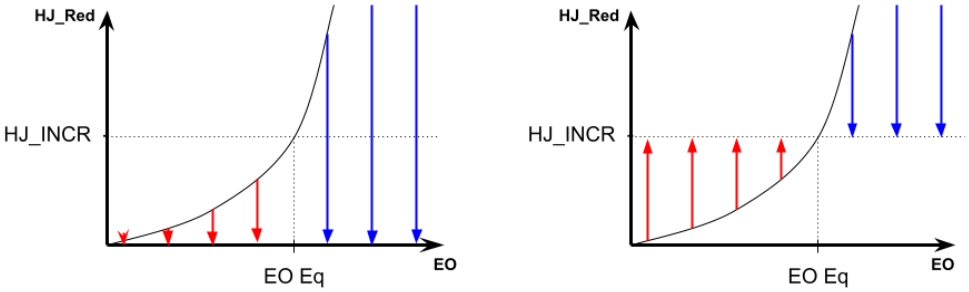
3.3.3 Quantités Émises par les Agents Larves

Un autre point clé de ce modèle est la capacité des agents larves à influer directement sur la physionomie des adultes, en contraignant certain à rester physiologiquement jeunes. Il faut donc que ces larves en émettent la bonne quantité : trop peu et il n'y aura pas assez de nourrices pour s'occuper du couvain, trop et il n'y aura plus assez de butineuses à la récolte de nourriture.

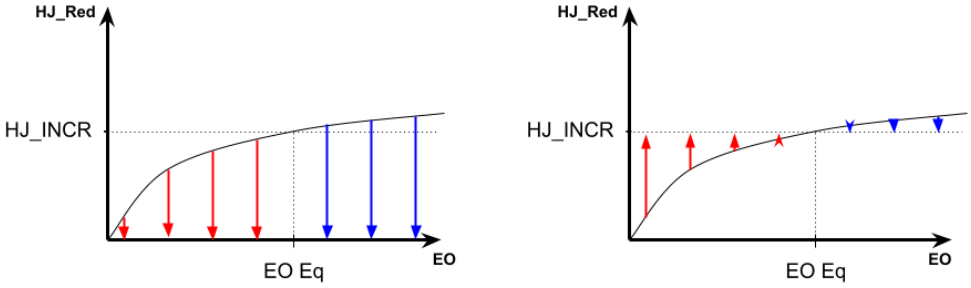
La quantité d'EO émise par chaque larve est donc importante, car elle est censée permettre à l'abeille de rester jeune, et même de rajeunir. Nous avons donc commencé par placer cette émission à EO_{Evap} (Voir eq. 3.3). De plus, les contacts étant assez bref, l'émission d'EO des larves doit permettre à une nourrice sollicitée de rester jeune malgré quelques temps de trajets. Ce dernier point porte une incertitude liée à l'émergence de ce comportement : la répartition des larves sur le cadre, les temps de trajets des nourrices entre les larves qui accepteront de la nourriture, l'aléatoire dans le trajet même que va choisir la nourrice etc. Nous avons donc décidé de placer l'émission des larves à $k * EO_{Evap}$,



(a) Réduction d'HJ **linéaire** (exposant = 1) par l'abeille en fonction de la quantité d'EO qu'elle possède. Lorsque $EO = EO_{Eq}$, la réduction compense parfaitement le vieillissement naturel de l'agent : $HJ_{Red} = HJ_{Incr}$.



(b) Lorsque la fonction de réduction est **quadratique** (exposant > 1) plutôt que linéaire (Fig 3.7a), on observe à gauche que l'EO a moins de puissance sous EO_{Eq}, et beaucoup plus au dessus. A droite, on voit donc que le vieillissement (lorsque $EO < EO_{Eq}$) ainsi que le rajeunissement ($EO > EO_{Eq}$) sont plus intenses.



(c) Lorsque la fonction de réduction est **une racine** (exposant < 1) plutôt que linéaire (Fig 3.7a), on observe à gauche que l'EO a plus de puissance sous EO_{Eq}, et beaucoup moins au delà. A droite, on voit donc que le vieillissement ($EO < EO_{Eq}$) ainsi que le rajeunissement ($EO > EO_{Eq}$) sont plus doux.

FIGURE 3.7 – Différents degrés de fonctions pour ajuster l'intensité des effets de l'Ethyle Oléate sur les abeilles adultes. Pour toutes les figures, à gauche les flèches indiquent la force de réduction de l'EO, à droite elles indiquent la vitesse de vieillissement (en rouge lorsque l'agent vieillit, en bleu lorsqu'il rajeunit).

avec k un coefficient que nous allons trouver expérimentalement, qui sera forcément supérieur à 1. En effet, nous savons qu'en dessous les larves n'auront pas le pouvoir d'empêcher les nourrices de vieillir et de partir butiner par la suite. La question est donc de savoir à quel point il doit être supérieur à 1.

3.3.4 Objectifs de calibration

Nous saurons que le coefficient k de l'émission d'EO des larves est juste lorsque, peu importe la quantité de larves, la proportion de nourrices par rapport aux butineuses sera globalement constante. Avec une émission trop importante, augmenter le nombre de larves va drastiquement augmenter le ratio de nourrices. Par exemple, lors d'un essai nous obtenions 50% de nourrice dans la population d'adultes pour 500 larves, mais presque 100% de nourrice pour 1000 larves. À l'inverse, lorsque trop peu de nourrices sont captées par le couvain, ce coefficient doit être augmenté.

Ensuite, il arrive qu'il faille diminuer la quantité d'EO émise par les larves alors qu'elle est déjà trop faible pour un petit nombre de larves (ou vice versa). C'est ici le signal que l'intensité des effets de l'EO sur les adultes est à ajuster en variant l'exposant de l'équation donnant HJ_{Red} (Eq 3.4). Ainsi, lorsque 500 larves maintiennent un ratio nourrices/butineuses correct mais faible, mais que 1000 retiennent trop de nourrices, c'est que l'EO a trop d'effet, il faut alors abaisser l'exposant de HJ_{Red}.

Nous avons gardé un exposant entier lorsqu'il est supérieur à 1, et de la forme $1/x$ (avec x entier) pour un exposant inférieur à 1.

Ce mécanisme par échange de phéromones est à la base de ce que nous cherchons à démontrer ici, et est profondément émergent, car dépendant des interactions entre tous nos agents, ce qui explique notre recours ici à un paramétrage expérimental. La méthodologie et les résultats de la calibration seront présentés et discutés dans le chapitre suivant.

Conclusion

Dans ce chapitre nous avons discuté de l'architecture logicielle du simulateur qui fait tourner notre première itération du modèle. Nos agents à l'intérieur de la colonie doivent se répartir deux tâches automatiquement, "Nourrir les larves" et "Butiner". Plusieurs couches d'abstraction nous offrent une grande modularité dans le développement, et l'utilisation de plusieurs *thread* d'exécution permet d'accélérer la simulation, nous permettant d'itérer

plus confortablement sur le modèle. Nous avons ensuite décrit l'implémentation concrète du modèle de sélection et d'interruption de tâche, à partir de ce que nous avons pu construire au chapitre précédent : les seuils pour la physiologie de l'agent, modéliser les tâches en Activité ainsi qu'en Action, parfois démotivante etc. Nous avons aussi décrit l'implémentation du modèle physiologique simplifié de l'abeille adultes, ses glandes, hormones, phéromones et leurs influences mutuelles et sur le comportement de nos agents. Après calibration de certains paramètres émergents difficile à pré-calculer ou à estimer, nos agents devraient être capables de se répartir le travail sans contrôle central, et de manière dynamique, en s'adaptant aux changements d'environnement et de populations. Dans le chapitre suivant nous allons pouvoir nous intéresser à cette auto-organisation : comment la mesurer, est-elle satisfaisante ? Nous parlerons aussi de calibration expérimentale, amenée par le caractère émergent de beaucoup de paramètres à retrouver nous venant d'observations de colonies réelles.

Chapitre 4

Evaluation de l'Implémentation de la Simulation de Colonie d'Abeilles

Après avoir modélisé puis construit notre modèle multi-agents et son simulateur, il est désormais temps d'en observer, interpréter puis valider (ou non) les résultats qu'il nous donne. Nous allons donc définir ce que nous attendons de ce modèle et de sa calibration. Nous discuterons ensuite des résultats obtenus puis les interpréterons, afin d'en sortir notamment des perspectives d'améliorations.

4.1 Hypothèses et Calibration

4.1.1 Hypothèses de Validation

Nous avons mis en place deux grandes familles de vérifications, afin de cerner le problème sous plusieurs approches. La première approche vise a vérifier les capacités d'auto-organisation de notre modèle, elle consiste donc en plusieurs simulations aux paramètres identiques mais dont nous avons fait varier la situation initiale. La seconde approche vise à vérifier si notre modèle de la colonie arrive à reproduire le comportement de colonies réelles. Afin de mieux préciser cette deuxième vérification, pour l'instant très large, nous nous sommes concentrés sur un cas particulier de la vie d'une colonie : l'adaptation de la colonie suite à une division. une division est une opération apicole qui consiste à séparer une colonie dont la population a dépassé un certain seuil, afin de créer deux colonies. L'apiculteur doit alors faire de son mieux pour bien répartir les différentes populations sur les deux colonies. Les deux nouvelles colonies ainsi crées doivent alors faire face à un changement rapide d'environnement, et donc s'auto-organiser, afin de survivre cette épreuve.

Nous avons donc deux hypothèses principales :

— **H1** : Notre modèle de colonie d'abeilles virtuelle est capable d'auto-organisation.

— **H2** : Notre modèle est capable d'approcher les dynamiques de populations observées dans les colonies d'abeilles réelles.

Afin de valider **H1** nous avons simplifié le modèle de la colonie, afin d'obtenir un environnement plus stable, mettant mieux en valeur l'adaptation de nos agents. Ainsi, le cycle de vie n'était pas simulé : aucune naissance (et donc, pas de reine pour pondre), aucune émergence (larves devenant adultes), aucune mort de vieillesse. En revanche, les larves manquant de nourritures meurent de faim. Nous pouvions ainsi conserver un nombre d'ouvrières et de larves constant, faisant de ce fait mieux ressortir les différents ratio de population. Dans cette validation, la proximité avec la biologie de l'abeille n'est pas nécessaire, nous avons ainsi largement accéléré les phénomènes physiologiques de nos agents, ainsi que réduit le nombre d'agents pendant les simulations. Ceci nous a permis d'itérer plus confortablement. Valider **H1** consiste alors à observer un point d'équilibre dans les populations d'ouvrières, entre le nombre de butineuses et de nourrices. Il faut aussi que ce point d'équilibre dépende du nombre de larves présentes dans la colonie. Nous parlons alors de "ratio d'ouvrières par larve". Nous avons donc mis en place 5 scénario :

— **Scénario 1.1** : Répartition initiale aléatoire des âges avec 150 abeilles adultes et 150 larves.

— **Scénario 1.2** : 150 adultes et 150 larves mais toutes les abeilles adultes commencent très jeunes.

— **Scénario 1.3** : 150 adultes et 150 larves, mais toutes les abeilles adultes commencent âgées.

— **Scénario 1.4** : Répartition initiale aléatoire des âges avec 150 adultes et 50 larves.

— **Scénario 1.5** : Répartition initiale aléatoire des âge avec 150 adultes et 300 larves.

Afin de valider **H2**, nous avons mis en place une expérimentation dont nous parlerons bien plus tard dans la Section 6.4, dont une partie est dédiée à cette hypothèse. Nous avons montré des graphiques de populations de nos simulations à des apiculteurs de différents niveaux. Ils ont ensuite été invités à noter les évolutions de populations de 1 à 5, si elles étaient ou non, selon eux, cohérentes avec ce qu'ils auraient attendu d'une colonie réelle. Notre modèle est alors dans sa dernière version, avec cycle de vie et calibration au plus proche de la biologie. Comme énoncé plus tôt, les scénario présentés aux apiculteurs correspondaient à des colonies sortant tout juste de division, sans reine pondeuse [1] mais

1. En réalité la reine est présente dans la simulation, mais n'est autorisée à pondre qu'après avoir

avec réserve de couvain. Nous avons ensuite pu faire varier la répartition des âges de la population de la nouvelle colonie. Nous en avons ainsi fait 4 scénario :

- **Scénario 2.1** : Répartition initiale avec seulement des abeilles adultes très jeunes.
- **Scénario 2.2** : Répartition initiale avec 50% d'abeilles adultes jeunes, et 50% d'abeilles adulte âgées.
- **Scénario 2.3** : Répartition initiale avec 100% d'abeilles adultes âgées.
- **Scénario 2.4** : Répartition initiale avec 50% d'abeilles adultes jeunes, et 50% d'abeilles adulte âgées, mais la reine ne commence pas à pondre en 30 jours, au lieu de 21.

Tous ces scénario démarrent avec 500 adultes et 500 agents couvains, répartis uniformément en œuf, larves, et nymphes. Le délai avant la ponte de la nouvelle reine a été choisi pour être dans la tranche basse des âges observés en réalité, afin de rester réaliste tout en proposant un cas favorable, pour aider la survie de la colonie. Pour des raisons de temps de passage seulement les deux premiers ont été intégrés dans l'expérimentation. En effet, ces deux scénario présentent le plus d'intérêts : le troisième est très court et conduit à la perte de la colonie, et le quatrième n'est qu'une légère modification du second mais entrainant lui aussi la perte de la colonie. L'auto-organisation est donc beaucoup moins visible.

De plus, il est classique dans la réalité d'obtenir une colonie avec une immense majorité de très jeunes adultes. En effet, si la nouvelle colonie est déposée proche ($<$ 1km) de son ancien placement, alors les butineuses retournerons à leur emplacement initial. Nous nous retrouvons ainsi avec une colonie uniquement composée de nourrices, n'ayant pas encore réalisé leur vol d'orientation. Si la nouvelle ruche est placée loin de sa position initiale avant division, alors les butineuses resterons dans ce nouvel emplacement.

4.1.2 Calibration

La calibration du modèle pour les expériences visant à valider **H1** a été plus rapide et plus simple que la calibration du modèle complet, car nous ne cherchions pas à obtenir de résultat proche de la biologie, mais seulement retrouver l'auto-organisation. La méthode a tout de même été plus ou moins la même. En bon système complexe, nous avons à notre disposition une grande quantité de paramètres s'influant mutuellement de manières émergentes. Les deux principaux points de la calibration concernait les phéromones émisent

attendu un certain temps, correspondant au temps avant ponte d'une reine réelle après une division.

par nos agents, ainsi que leurs effets sur ces mêmes agents. Notre objectif de calibration, comme détaillé plus tôt section 3.3.4, est de retrouver un équilibre des proportions entre les populations de couvain et de nourrices, avec le reste de la population se plaçant en butineuses. On remarque alors que l'objectif de la calibration est déjà, au final, de valider **H1**, le modèle est alors calibré pour répondre à l'hypothèse initiale. La question est en réalité légèrement différente : nous essayons de voir si l'espace offert par l'ensemble des paramètres du modèle nous permet d'atteindre une calibration pour laquelle le modèle réponds à l'hypothèse. Ce qui pose d'ailleurs d'autres soucis d'ordres pratiques : pendant la calibration, si nous n'arrivons pas a obtenir les résultats que nous souhaitons, est-ce parce que le modèle ne permet pas de les atteindre ou n'est-il seulement pas correctement calibré ?

Après une bonne quantité d'itérations sur le modèle complet, nous avons finis par positionner le coefficient n de l'intensité des phéromones à $n = \frac{1}{3}$ (equation 3.4), et le coefficient k de la quantité de phéromones émisent par les agents du couvain à $k = 2$ (section 3.3.3). Ces paramètres ont été ajustés pour une simulation démarrant avec 500 larves, 500 abeilles et une reine, sur deux faces de cadres. Nous avons réalisés quelques essais nous faisant penser que la calibration reste valide en changeant le nombre d'agents en présence ainsi que leur répartition spatiale, mais la question est vaste et pas si facilement répondue, plus de travaux devront donc être réalisés dans ce sens.

Nous allons pouvoir désormais aborder nos résultats, sous la forme de courbes générées à partir de fichiers eux même générés pendant chaque simulation.

4.2 Résultats

4.2.1 Modèle à environnement constant

Le modèle à environnement constant, réalisé principalement pour vérifier la validité d'**H1**, comporte cinq scénario détaillés plus tôt. Les résultats de ces scénario sont décrits Figure 4.1, chacun ayant été joué cinq fois, ainsi les écarts-types sont aussi présent sur la figure, on y observe :

— **Scénario 1.1 (S1)** : Répartition initiale uniforme des physiologies et ratio adultes/larves de 1. Au lancement de la simulation, 50% des adultes de la colonie sont des nourrices, du fait de notre initialisation. Nous observons ici une convergence en quelques milliers de pas de temps vers 60% d'agents effectuant un travail de nourrice, et donc

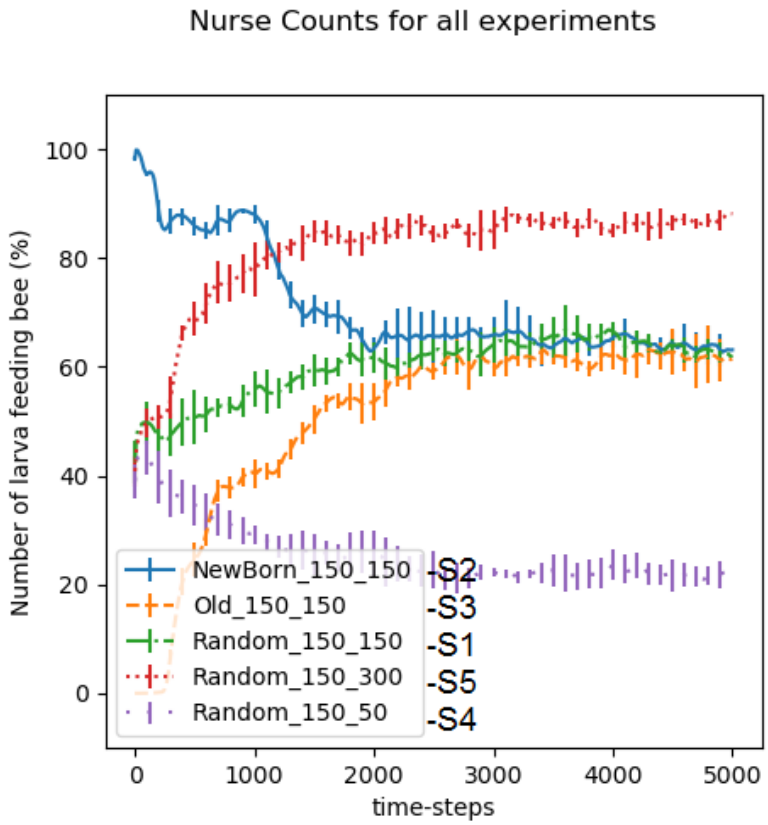


FIGURE 4.1 – Proportions de nourrices pour les 5 scénario visant à valider **H1**. Chaque scénario a été simulé 5 fois, les écarts-types sont visibles en barres verticales sur chacune des courbes.

Scénario	Population Initiale	Ratio Adultes par Larve	Équilibre
1.1	uniforme	1 pour 1	60%
1.2	jeune	1 pour 1	60%
1.3	âgée	1 pour 1	60%
1.4	uniforme	1 pour 3	20%
1.5	uniforme	2 pour 1	85%

TABLE 4.1 – Récapitulatif des différents scénario, leurs ratio adultes par larves ainsi que leur valeur finale d'équilibre, en proportion de nourrices dans la population d'adultes de la colonie virtuelle.

environ 40% de butineuses.

— **Scénario 1.2 (S2)** : Répartition initiale des physiologies avec uniquement de jeunes adultes et ratio adultes/larves de 1. Au lancement de la simulation, 100% des adultes de la colonie sont des nourrices, du fait de notre initialisation. Nous observons ici la même convergence en quelques milliers de pas de temps vers ce même équilibre, 60% de nourrices.

— **Scénario 1.3 (S3)** : Répartition initiale uniforme des physiologies et ratio adultes/larves de 1. Au lancement de la simulation, 0% des adultes de la colonie sont des nourrices, toutes sont butineuses, du fait de notre initialisation. Nous observons à nouveau un équilibre à 60% de nourrices en un peu plus de 2000 pas de temps.

— **Scénario 1.4 (S4)** : Répartition initiale uniforme des physiologies et ratio adultes/larves de 1 pour 3. Au lancement de la simulation, 50% des adultes de la colonie sont des nourrices, du fait de notre initialisation. Nous observons cette fois ci un équilibre en un peu plus de 2000 pas de temps, mais à environ 20% de nourrices.

— **Scénario 1.5 (S5)** : Répartition initiale uniforme des physiologies et ratio adultes/larves de 2 pour 1. Au lancement de la simulation, 50% des adultes de la colonie sont des nourrices, du fait de notre initialisation. L'équilibre est atteint un peu avant 2000 pas de temps, mais se situe cette fois aux alentours de 85% de nourrices.

Le Tableau 4.1 reprend ces différents résultats de manière plus concise. Vous y trouverez, pour chaque scénario, le ratio adultes/larves ainsi que l'équilibre de répartition de travail atteint, donné en proportion de nourrices dans la population d'adultes de la colonie virtuelle.

Avant d'interpréter ces résultats, nous allons consulter ceux du modèle complet vis à vis d'**H1**, mais aussi de **H2**.

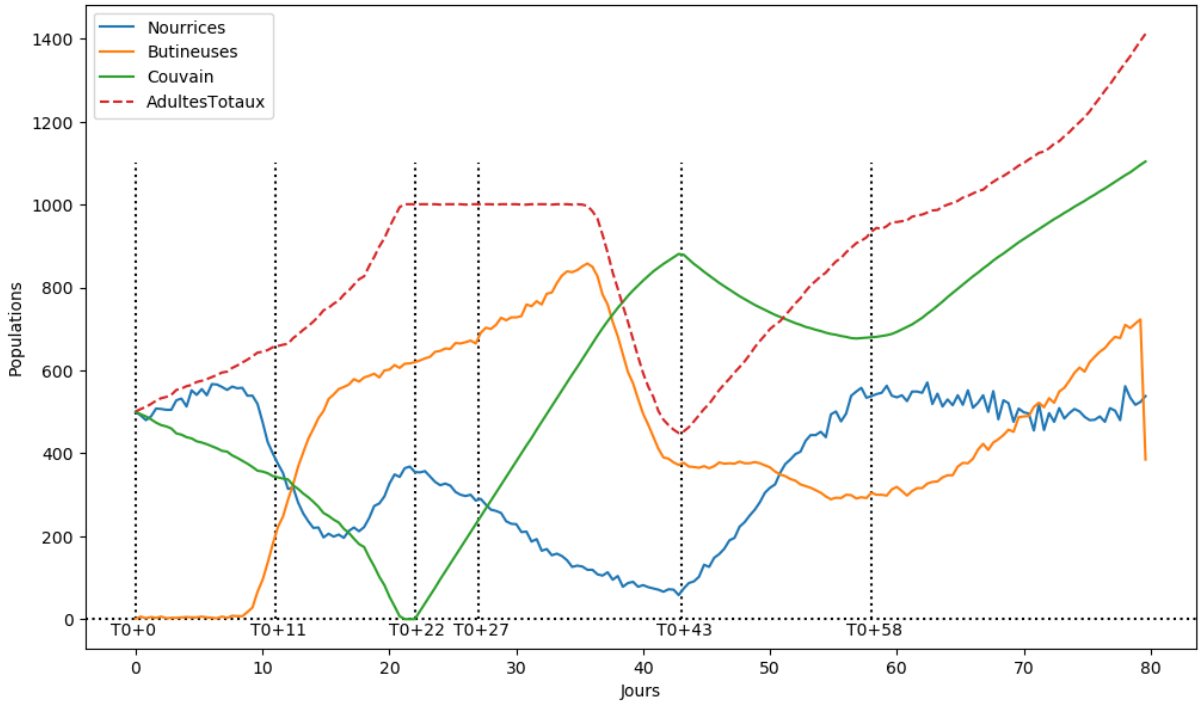


FIGURE 4.2 – Les différentes populations de la colonie après la division du Scénario 2.1.

4.2.2 Modèle Complet, Division et Cycle de vie

Les Figures 4.2 et 4.3 présentent graphiquement respectivement les résultats des scénario 2.1 et 2.2. Sous la forme de différentes courbes de populations au court du temps, ces figures nous renseignent sur la vie de nos colonies virtuelles juste après une division. Quelques indicateurs de temps ont été placés afin d'en faciliter la lecture : T0 indique le début de la simulation, le moment ou la division vient d'être réalisée. Par exemple, T0+11 indique le 11ème jour après la division. Nous y observons les populations de nourrices, de butineuses, mais aussi de couvain ainsi que la population totale des adultes de la colonie, soit la somme des populations de nourrices et de butineuses. Dans ces simulations, des agents naissent et meurent régulièrement, ce qui rend la validation de **H1** moins triviale : le fameux ratio adultes par larves n'est pas constant, loin de la. Voici nos résultats :

Scénario 2.1 (Figure 4.2) : Répartition initiale des physiologies avec uniquement de très jeunes abeilles adultes, départ avec 500 adultes et 500 agents de couvain (1/3 d'œufs, 1/3 de larves et 1/3 de nymphes). Par ordre chronologique, nous observons à T0+11 un important changement dans la répartition du travail, où le nombre de nourrices chute drastiquement au profit du nombre de butineuses. La quantité de couvain n'a de cesse

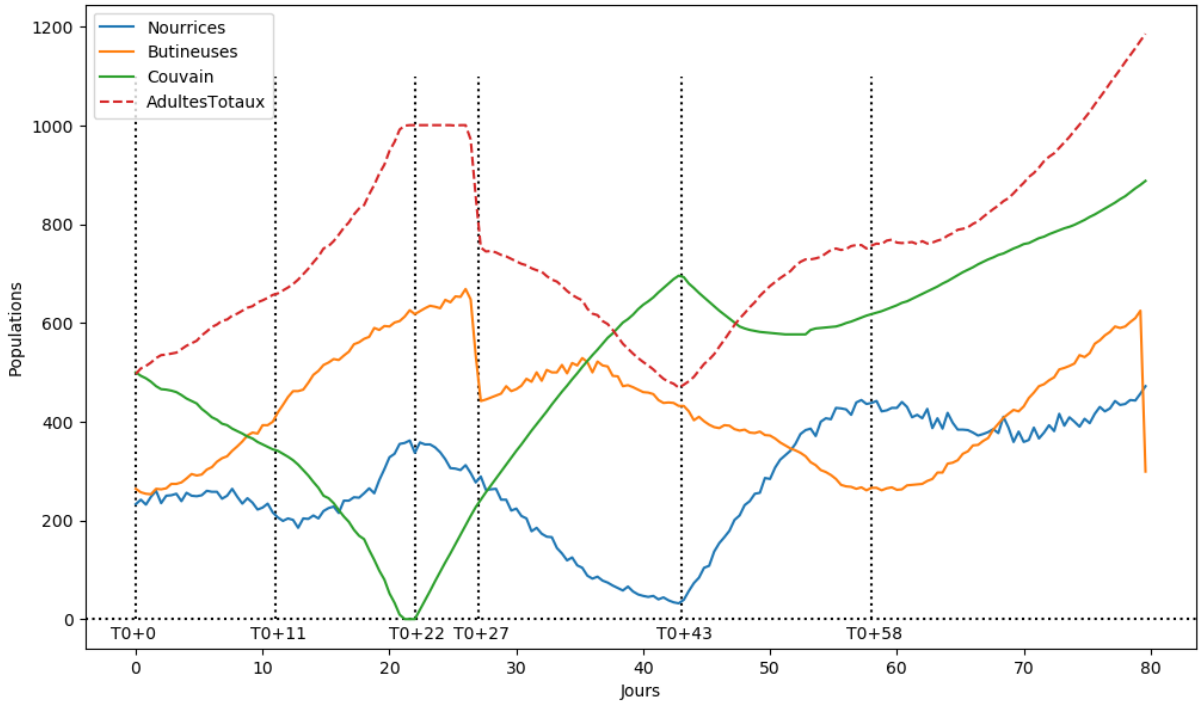


FIGURE 4.3 – Les différentes populations de la colonie après la division du Scénario 2.2.

de chuter jusqu'a T0+21 où il atteint zéro. La population totale cesse alors de croitre. À T0+22, la reine commence sa ponte, le couvain repart à la hausse, mais le nombre de nourrices continue de chuter. Nous observons ensuite une chute de la population totale : les butineuses commencent à mourrir de vieillesse. En effet, la reine vient seulement de recommencer à pondre. La population repart à la hausse à T0+43, lorsque les premières pontes de la reine émergent enfin, venant renforcer le faible contingent de nourrices. Enfin, à T0+58, nous observons à nouveau un transfert de répartition du travail : la population de nourrices qui croit alors très rapidement, va voir nombre de ses agents partir butiner. La population de nourrices se stabilise ensuite jusqu'à la fin de la simulation, 80 jours après la division, alors que le couvain augmente en permanence.

Scénario 2.2 (Figure 4.3) : Répartition initiale des physiologies uniforme, départ avec 500 adultes et 500 agents de couvain (1/3 d'œufs, 1/3 de larves et 1/3 de nymphes). Nous observons une population constante de nourrices, aidée par les émergences du couvain, et une population de butineuses croissante. La population de nourrices décrit alors un pic à T0+22, au moment où les dernières nymphes émergent et alors que la reine commence à pondre. À T0+27, les butineuses déjà présentent au début de la simulation meurent déjà de vieillesse. La progression suit ensuite de très près la chronologie du scénario précédent,

Scénario	Moyennes	Ecarts-types	Notes Individuelles (1-5)														
2.1	3.93	0.77	4	3	3	4	4	2	5	4	4	4	4	5	4	5	4
2.2	3.93	1.06	4	4	4	4	5	4	1	4	5	2	4	5	4	5	4

TABLE 4.2 – Résultats de l'expérimentation auprès de quelques apiculteurs concernant la cohérence des variations de populations de nos simulations par rapports aux colonies réelles.

avec une population globale légèrement plus faible.

Nous avons ensuite pu présenter ces deux courbes à des apiculteurs au cours d'une journée d'expérimentations organisée entre l'UBO, l'IMT-Atlantique et le GDSA29. Le Tableau 4.2 récapitule les réponses qu'ont donnés les participants à la question *" L'évolution de la population de la colonie vous semble-t-elle cohérente avec la façon dont la division a été réalisée ? "*, tour à tour pour les Figures 4.2 et 4.3. "La façon dont la division a été réalisée" concerne les quantités de populations présentes dans la colonie après la division, et est un exercice auxquels les apiculteurs sont très sensibilisés : il en va de la survie de la colonie. Les deux scénario obtiennent alors la même note moyenne de 3.93 ainsi que la même médiane à 4, mais les écarts-types diffèrent, respectivement 0.77 et 1.06.

4.3 Interprétations des Résultats

Nous allons désormais comparer nos résultats à nos hypothèses, évaluer les écarts ainsi que les biais, et parler d'éventuelles perspectives d'améliorations.

Nous nous intéressons ici à notre première hypothèse, sobrement appelée **H1**, que nous rappelons :

H1 : Notre modèle de colonie d'abeilles virtuelle est capable d'auto-organisation.

Une première version du modèle, appelée version "à environnement constant" (car elle conserve les niveaux de populations sur toute la durée de la simulation) nous sert principalement à vérifier cette hypothèse. Ses cinq scénario, manipulant les conditions initiales ainsi que les rapports de populations permettent de faire ressortir les capacités d'adaptation de notre modèle. Ainsi, comme récapitulé dans le Tableau 4.1, on note que la répartition initiale de la physiologie de nos agents n'influe pas sur l'équilibre de la répartition des tâches entre eux. Le ratio adultes / larves en revanche semble être le facteur directeur de l'équilibre. Plus il y a de larves par agents adultes, plus ces-derniers

seront nombreux à effectuer un travail de nourrice. Ainsi, nous pouvons dire que notre modèle valide **H1** : nos agents se répartissent correctement les tâches selon les besoins de l'environnement, ici plus ou moins de larves.

Nous pouvons nous intéresser à ce qu'apporte les scénario 2.1 et 2.2 en ce qui concerne cette première hypothèse. En effet, la calibration "temps réel", sans les accélérations physiologiques présentent dans le modèle environnement constant, ainsi que le cycle de vie apporte des nouveautés. La répartition du travail est toujours visible, mais le cycle de décision des agents est plus long. En effet, une abeille nourrice décidant de devenir butineuse va devoir attendre quelques jours que ses glandes soient en état (attendre que son niveau d'HJ soit assez élevé). Ce délai influe la réactivité des agents, mais représente plus fidèlement la réalité. Les scénario 2.1 et 2.2 dans leurs figures respectives 4.2 et 4.3, présentent tout deux à T0+22 des répartitions de populations identiques, avec environ 350 nourrices et 600 butineuses, alors que les conditions initiales sont différentes. Ceci est un pas vers la vérification d'**H1** sur le modèle complet. Les suites de ces deux scénario diffèrent car la reine ajuste sa vitesse de ponte en fonction de la population de la colonie, et que dans le scénario 2.2, les butineuses meurent beaucoup plus tôt, ralentissant plus rapidement la vitesse de ponte.

Il est toutefois étonnant de constater le temps que les nourrices continuent de devenir butineuses entre T0+22 et T0+43, alors que la population du couvain explose. Les premières larves ont besoin de soins aux alentours de T0+25 (après 3 jours en tant qu'œuf), il est donc très étonnant que la population de nourrices ne se stabilise pas. Une des raisons auxquelles nous pensons, et que nous avons déjà évoquée dans ce manuscrit, est l'absence des receveuses. Les receveuses agissent comme un intermédiaire systématique entre les butineuses et les nourrices, permettant ainsi aux butineuses, plus âgées, de mieux ralentir la production d'HJ de leurs jeunes compatriotes. Sans cet effet, la proportion de nourrice par rapport aux butineuses peut être très faible, alors qu'en réalité, il n'est jamais observé de colonie avec si peu de nourrices.

Nous allons désormais pouvoir nous intéresser à notre deuxième hypothèse, **H2**, qui énonce :

H2 : Notre modèle est capable d'approcher les dynamiques de populations observés dans les colonies d'abeilles réelles.

D'après les retours donnés par les apiculteurs, avec une médiane à 4 sur 5 lorsque nous leur avons demandé de juger de la cohérence des évolutions de populations comparés à

leur connaissances pratiques, nous pouvons dire qu'**H2** est validée. Pas parfaite, pour les quelques raisons que nous avons évoquées juste au dessus. Il nous a aussi été pointé que si les dynamiques semblent bonnes, les quantités ne sont pas du tout respectées. En effet, les colonies d'abeilles comptent régulièrement jusqu'à 50 000 individus, ce qui est bien loin de nos 1000 individus. Il sera donc intéressant, à l'avenir et maintenant que la dynamique est validée, d'essayer d'augmenter drastiquement le nombre d'individus, afin d'observer si la calibration ainsi que ses propriétés sont toujours valides ou si elles ne passent pas à l'échelle. La calibration ne fonctionne peut être qu'avec un nombre réduit d'agents. En effet, la répartition spatiale peut avoir un impact très important sur l'émergence des propriétés d'auto-organisation.

Lors de nos premiers essais préliminaires sur de très grand nombre d'agents, nous observions dans nos simulations un "embouteillage" d'abeilles, où les agents se bloquent tellement entre eux que leur navigation devient impossible. Il est peut être possible de régler ce soucis en gérant différemment les déplacements, mais il sera peut être nécessaire de changer d'approche quant à la spatialisation des agents dans la ruche et sur les cadres.

4.4 Perspectives d'Améliorations

4.4.1 Perspectives du Modèle de prise de décisions

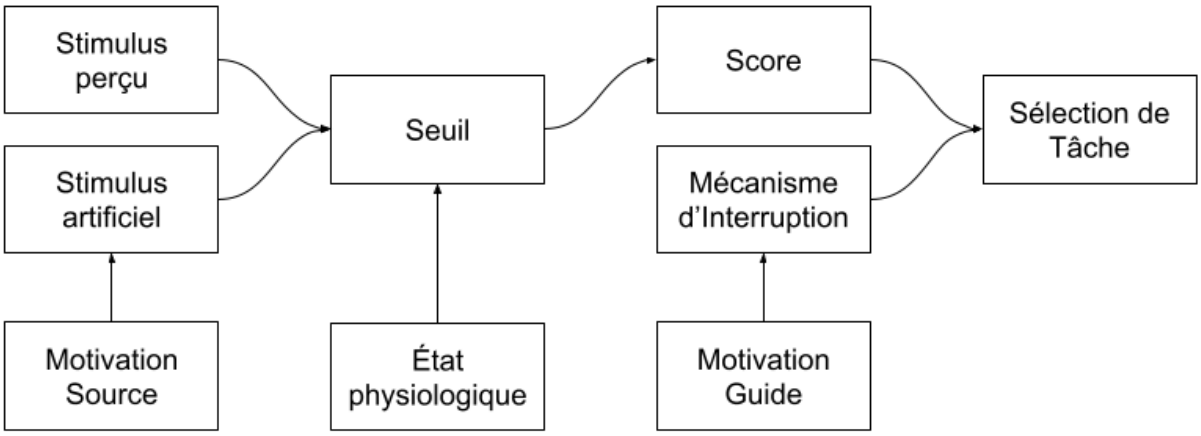
Après ces travaux sur le modèle de prise de décision présenté Chapitre 2, nous avons en tête quelques améliorations pour d'éventuelles nouvelles implémentations de celui-ci. La Figure 4.4a illustre le modèle de prise de décision tel qu'il est décrit dans ce manuscrit : un stimulus, qu'il soit perçu ou artificiel (représentant la Motivation Source), sert à calculer un score pour sa Tâche par le biais d'une fonction sigmoïde et de son seuil. Ce-dernier représente alors l'état physiologique/physique de l'agent. Enfin, la sélection se fait soit en prenant en compte le score de la tâche, soit, pour une Tâche Motivée précédemment exécutée, c'est le mécanisme d'interruption, représentant la Motivation Guide transversale à l'agent, qui est pris en compte. Ainsi la Tâche ayant le score final, score ou motivation, le plus élevé sera sélectionnée.

Ce que nous proposons pour une version améliorée est de se passer de ce mécanisme d'interruption tel qu'il est présenté, en fusionnant les deux types de motivations au niveau du stimulus artificiel. La Figure 4.4b décrit ce nouveau fonctionnement : le calcul du score se fait soit via le stimulus perçu, soit, dans le cas de Tâche Motivée, via le

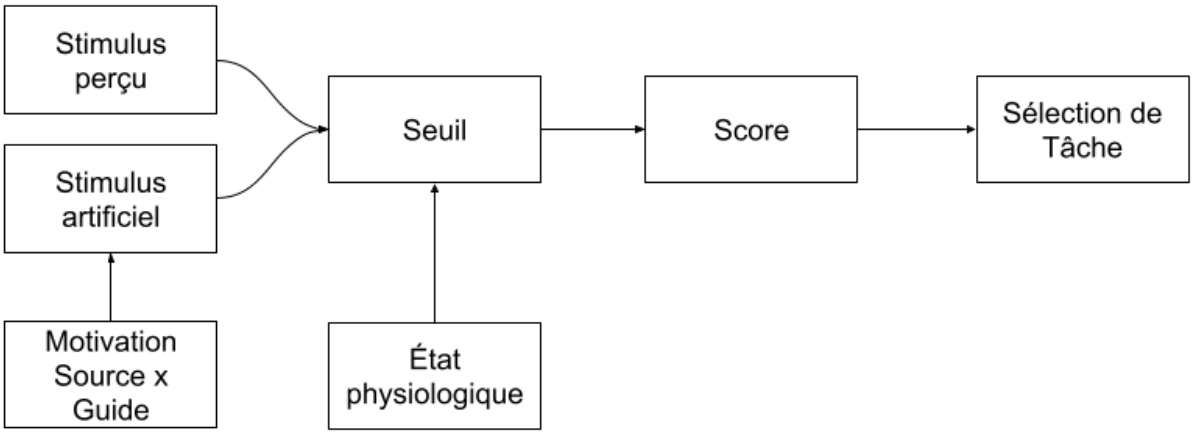
produit des Motivations Source et Guide de la Tâche. L'agent perd alors sa motivation interne transversale à toutes les tâches, et voit ses Motivations Guides être liées à chaque Tâche Motivée. Ainsi, le score d'une tâche représente les deux motivations, et contient déjà l'information que notre mécanisme d'interruption apportait, car le stimulus est alors une image des deux motivations que nous avons décrites. Il est alors nécessaire d'intégrer le concept d'Actions Motivantes, qui, à l'inverse d'Actions Démotivantes, remontent la Motivation Guide d'une tâche à chaque exécution. Agissant un peu à la manière d'une récompense, les Actions Motivantes seront sélectionnées afin de concerner les Actions qu'un agent réalise lorsqu'il réussi sa Tâche. La Motivation Guide devra aussi remonter lentement au fil du temps même si la Tâche n'est pas sélectionnée, propriété que nous retrouvons classiquement dans les modèles à seuils, à propos de seuils s'ajustant au fil de la simulation. Il sera alors intéressant d'observer l'importance des paramètres de baisses et de hausses de la motivation dans la répartition du travail, et d'en modifier les dynamiques en utilisant par exemple des incréments constants ou pourquoi pas une fonction exponentielle accélérant la baisse ou la hausse selon la fréquence d'appel de ces changements. Nous pensons qu'avec la Motivation Guide liée à chaque tâche, chaque agent aura un comportement plus cohérent, en évitant d'osciller entre deux tâches, ce qui devrait par la même occasion augmenter son efficacité. Cette nouvelle itération du modèle permet une intégration plus naturelle des concepts de motivations dans les modèles à seuils : en ne jouant que sur un stimulus artificiel nous n'avons pas besoin de modifier le fonctionnement du modèle à seuils, là où la version actuelle nous demande d'y intégrer un nouveau mécanisme.

4.4.2 Perspectives de la Simulation Multi-Agents

Il est possible d'enrichir le modèle de la colonie d'abeilles de beaucoup de manières différentes, afin de rapprocher nos résultats d'observations faites sur de réelles colonies. L'amélioration la plus prioritaire selon nous serait d'enrichir la simulation de la nourriture dans le modèle. La nourriture est pour l'instant ramenée par les butineuses, mais est aussi présente à l'intérieur de la colonie, dans quelques cellules de cadres agissant comme des sources infinies. Ainsi, nous avons pu nous intéresser pour l'instant à l'importance de sa distribution, plutôt qu'à sa collecte. Pour ceci, il sera nécessaire d'échelonner plus en détail les quantités de nourritures consommées par les larves et les adultes, mais aussi les quantités rapportées par les butineuses. Il sera aussi nécessaire d'ajouter les tâches de "receveuses" que nous avons décrit dans le Chapitre de Contexte, au début de ce



(a) Notre modèle de prise de décision tel qu'implémenté. La Motivation Source nous permet de créer un stimulus artificiel pour les Tâches n'ayant pas de stimulus déclencheurs évidents. La Motivation Guide permet d'interrompre la réalisation de cette tâche lorsque l'agent n'arrive pas à la réaliser correctement, via les Actions Démotivantes.



(b) Proposition pour une amélioration du modèle de prise de décision. Le mécanisme d'interruption disparait, et le stimulus artificiel pour les Tâches Motivées devient le produit des Motivations Source et Guide. Ainsi, le score devient l'image de la combinaison de ces deux motivations sans avoir recours à un mécanisme d'interruption tout en produisant les mêmes effets.

FIGURE 4.4 – Schéma du modèle de prise de décision tel que décrit dans ce manuscrit (a), sous lequel nous présentons une version améliorée (b).

manuscrit. De plus, ces tâches de receveuses recoupe une hypothèse que nous avons déjà énoncé dans ce manuscrit, qui est qu'elles sont les principales responsables de l'impact rajeunissant des abeilles plus âgées sur les jeunes adultes. Elles jouent en effet un rôle très important dans la propagation de phéromones dans la colonie, en faisant le pont entre les âges. De la même manière il serait intéressant d'intégrer un module gérant le butinage de manière plus poussée que la version minimaliste de cette itération. Nous pourrions alors observer les impacts qu'ont les changements de l'environnement extérieur de la ruche sur la répartition du travail à l'intérieur de celle-ci.

Il serait aussi possible d'intégrer la gestion de la température, dont nous avons aussi parlé dans le Chapitre de Contexte, qui est un point clé de la colonie. De plus, les abeilles d'hiver présentent des caractéristiques très intéressantes qu'il serait intéressant de retrouver par la simulation. Nous pourrions alors intégrer la récolte d'eau par les butineuses, qui sert à la régulation de la température.

Autre point intéressant qu'il serait intéressant d'aborder, est l'ajout du parasite Varroa Destructor dans la simulation. Varroa est un acarien parasite qui s'attaque aux larves et nymphes et provoquent des malformations lorsque ces abeilles deviennent adultes, mettant ainsi en danger toute la colonie [22]. Il serait donc particulièrement intéressant d'utiliser la simulation pour essayer de reproduire les attaques Varroa, pour éventuellement essayer de détecter des points clés dans les procédés, et en sortir des propositions pour aider les apiculteurs et biologistes dans leur lutte contre ce parasite particulièrement vorace.

Couplé à un butinage plus poussé, nous pourrions de la même manière et pour les mêmes raisons ajouter dans l'environnement extérieur des facteurs de stress : variations brutales de température limitant le butinage, monocultures limitant la disponibilité de nectar et pollen, pesticides et bien d'autres.

Dans tous les cas, étoffer la simulation pourra nous permettre de simuler plus de scénario différents, nous permettant de réitérer l'expérimentation où nous demandons l'avis d'apiculteurs sur des scénario plus variés, amenant donc à des résultats plus précis. Nous pourrions de la même manière proposer ces interprétations de résultats de simulation à des biologistes de l'abeille, dont l'attention se portera naturellement sur d'autres aspects de la colonie, apportant un regard complémentaire à celui des apiculteurs.

Conclusion

Dans ce chapitre nous avons abordé les enjeux et les manières de la calibration des paramètres du modèle multi-agents. Nous avons ensuite observé les résultats qu'il a pu produire dans différentes conditions, et différentes itérations du modèle. Nous avons alors pu valider nos deux hypothèses : notre modèle multi-agent est capable d'auto-organisation et il produit des résultats cohérents avec les observations de colonies d'abeilles réelles. Forts de ces résultats, nous avons pu clore en énonçant quelques perspectives pour la suite de ces travaux, tant au niveau du modèle multi-agents qu'au niveau du modèle de prise de décision. Voici qui conclut la première partie de ce manuscrit, orienté vers la simulation multi-agents, nous allons désormais aborder la partie concernant la visualisation, les interactions et les environnements immersifs. De cette manière le chapitre suivant réalise un état de l'art de ces domaines, mais toujours à travers le prisme des systèmes complexes.

Chapitre 5

Etat de l'art : Simulation Multi-Agents et Environnements Immersifs

5.1 SMA : Recréer et comprendre des systemes complexes existants par l'interaction

Comprendre les mécanismes, créer un modèle puis évaluer l'impact des différents paramètres sur l'évolution du comportement du système.

5.2 Manipuler et observer ces systèmes complexes

Systèmes en général non immersif, on s'arrête à la RA, et [de ce que j'ai vu], sans utilisation d'interacteurs tangibles. Comment mieux comprendre un système complexe : Environnement immersif et interacteurs tangibles. Ruche, cadre et connaissances apicoles.

5.3 DataViz

Comment visualiser de grande quantité de données, les données micro.

5.4 Interaction / visu immersive pour comprendre

Environnement immersif

5.5 Interaction tangible

Interacteurs tangibles

Conclusion

Chapitre 6

Proposition visu interaction

Notre proposition

6.1 Interaction Immersive avec Manettes

Manipulation des cadres avec les manettes

6.2 Interaction Immersice ET tangible

L'apport et les contraintes du tangible (qu'il faut encore définir) pour la manipulation des cadres

6.3 Visualisation : Graph3D sur l'état interne de la colonie

Le graph3D et les information qu'il apporte

6.4 Résultats/Évaluation Visualisation Interactive proposée

Conclusion

Conclusion

Comment l'ensemble se comporte et avis critique sur la totalité. Notamment le modèle multi agent simplifié qui apporte une quantité gigantesque de biais.

Discussions

Perspectives

Tout est possible, pousser le modèle de l'abeille de la simulation, pousser le domaine tangible avec peut être un cadre manette (un cadre avec un ou deux boutons ?)

Conclusion

C'était cool

Rideau

clapclapclapclapclapclapclapclapclap

BIBLIOGRAPHIE

[1] W. Agassounon, A. Martinoli et R. Goodman. « A scalable, distributed algorithm for allocating workers in embedded systems ». In : *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat.No.01CH37236)*. IEEE International Conference on Systems, Man & Cybernetics. T. 5. Tucson, AZ, USA : IEEE, 2001, p. 3367-3373. ISBN : 978-0-7803-7087-6. DOI : 10.1109/ICSMC.2001.972039. URL : http://ieeexplore.ieee.org/document/972039/ (visité le 02/04/2021).

[2] M.R. Allen et al. *2018 : Framing and Context. In : Global Warming of 1.5°C. An IPCC Special Report on the impacts of global warming of 1.5°C above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty*. 2018. URL : https://www.ipcc.ch/site/assets/uploads/sites/2/2019/05/SR15_Chapter1_Low_Res.pdf (visité le 22/07/2021).

[3] Gianluca Baldassarre et Marco Mirolli. *Intrinsically Motivated Learning in Natural and Artificial Systems*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2013. ISBN : 978-3-642-32374-4 978-3-642-32375-1. DOI : 10.1007/978-3-642-32375-1. URL : http://link.springer.com/10.1007/978-3-642-32375-1 (visité le 07/11/2019).

[4] Eric Bonabeau, Marco Dorigo et Guy Theraulaz. *From Natural to Artificial Swarm Intelligence*. New York, NY, USA : Oxford University Press, Inc., 1999. ISBN : 978-0-19-513158-1.

[5] Eric Bonabeau, Guy Theraulaz et Jean-Louis Deneubourg. « Quantitative Study of the Fixed Threshold Model for the Regulation of Division of Labour in Insect Societies ». In : *Proceedings : Biological Sciences* 263.1376 (1996), p. 1565-1569. URL : http://www.jstor.org/stable/50403.

[6] R. Brooks. « A robust layered control system for a mobile robot ». In : *IEEE Journal on Robotics and Automation* 2.1 (1986), p. 14-23. ISSN : 0882-4967. DOI :

10.1109/JRA.1986.1087032. URL : http://ieeexplore.ieee.org/document/1087032/ (visité le 18/01/2020).

[7] Adam Campbell et Annie S. Wu. « Multi-agent role allocation : issues, approaches, and multiple perspectives ». In : *Autonomous Agents and Multi-Agent Systems* 22.2 (2011), p. 317-355. ISSN : 1387-2532, 1573-7454. DOI : 10.1007/s10458-010-9127-4. URL : http://link.springer.com/10.1007/s10458-010-9127-4 (visité le 18/09/2019).

[8] Vincent A Cicirello et Stephen F Smith. « Wasp-like Agents for Distributed Factory Coordination ». In : (2004), p. 30.

[9] Miquel Cornudella, Paul Van Eecke et Remi van Trijp. « How Intrinsic Motivation can Speed Up Language Emergence ». In : European Conference on Artificial Life 2015. The MIT Press, 20 juill. 2015, p. 571-578. ISBN : 978-0-262-33027-5. DOI : 10.7551/978-0-262-33027-5-ch100. URL : https://www.mitpressjournals.org/doi/abs/10.1162/978-0-262-33027-5-ch100 (visité le 12/11/2019).

[10] Mihaly Csikszentmihalyi. *Finding flow : The psychology of engagement with everyday life*. Basic Books, 1997.

[11] Anna Dornhaus et al. « Task Selection in Honeybees - Experiments Using Multi-Agent Simulation ». In : (1998), p. 13.

[12] Anna Dornhaus et al. « Task Selection in Honeybees - Experiments Using Multi-Agent Simulation ». In : (), p. 13.

[13] Alexis Drogoul. « De la simulation multi-agents a la resolution collective de problemes : une etude de l'emergence de structures d'organisation dans les systemes multi-agents ». thesis. Paris 6, 1er jan. 1993. URL : http://www.theses.fr/1993PA066544 (visité le 29/05/2019).

[14] Alexis Drogoul et Jacques Ferber. « Multi-agent simulation as a tool for modeling societies : Application to social differentiation in ant colonies ». In : *Artificial Social Systems*. Sous la dir. de Cristiano Castelfranchi et Eric Werner. Réd. par J. G. Carbonell et al. T. 830. Berlin, Heidelberg : Springer Berlin Heidelberg, 1994, p. 2-23. ISBN : 978-3-540-58266-3 978-3-540-48589-6. DOI : 10.1007/3-540-58266-5_1. URL : http://link.springer.com/10.1007/3-540-58266-5_1 (visité le 14/01/2020).

[15] Bruce Edmonds. « What is Complexity ? - The philosophy of complexity per se with application to some examples in evolution. » In : *Centre of Policy Modelling, Manchester Metropolitan University* (1999).

[16] Jacques Ferber et Olivier Gutknecht. « eta-model for the analysis and design of Organizations in multi-agent systems ». In : (1998), p. 8.

[17] Nigel R Franks et Chris Tofts. « Foraging for work : how tasks allocate workers ». In : *Animal Behaviour* 48.2 (1994), p. 470-472.

[18] Frederick W. P. Heckel, G. Michael Youngblood et Nikhil S. Ketkar. « Representational complexity of reactive agents ». In : *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*. 2010 IEEE Symposium on Computational Intelligence and Games (CIG). Copenhagen, Denmark : IEEE, août 2010, p. 257-264. isbn : 978-1-4244-6295-7. doi : 10.1109/ITW.2010.5593345. url : http://ieeexplore.ieee.org/document/5593345/ (visité le 18/01/2020).

[19] Francis Heylighen. « Complexity and Self-organization ». In : *Free University of Brussels, Krijgskundestr* (2008), p. 20.

[20] J. C. Jones. « Honey Bee Nest Thermoregulation : Diversity Promotes Stability ». In : *Science* 305.5682 (2004), p. 402-404. issn : 0036-8075, 1095-9203. doi : 10.1126/science.1096340. url : https://www.sciencemag.org/lookup/doi/10.1126/science.1096340 (visité le 16/07/2021).

[21] Michael J.B. Krieger et Jean-Bernard Billeter. « The call of duty : Self-organised task allocation in a population of up to twelve mobile robots ». In : *Robotics and Autonomous Systems* 30.1 (2000), p. 65-84. issn : 09218890. doi : 10.1016/S0921-8890(99)00065-2. url : https://linkinghub.elsevier.com/retrieve/pii/S0921889099000652 (visité le 21/07/2021).

[22] Yves Le Conte, Marion Ellis et Wolfgang Ritter. « Varroa mites and honey bee health : can Varroa explain part of the colony losses ? » In : *Apidologie* 41.3 (mai 2010), p. 353-363. issn : 0044-8435, 1297-9678. doi : 10.1051/apido/2010017. url : http://link.springer.com/10.1051/apido/2010017 (visité le 23/07/2021).

[23] Konrad Lorenz et Jeanne Etoré. *Les fondements de l'éthologie*. Flammarion Paris, 1984.

[24] Pattie Maes. « The agent network architecture (ANA) ». In : *ACM SIGART Bulletin* 2.4 (1er juill. 1991), p. 115-120. ISSN : 01635719. DOI : 10.1145/122344.122367. URL : http://portal.acm.org/citation.cfm?doid=122344.122367 (visité le 12/11/2019).

[25] Alban Maisonnasse et al. « E-B-Ocimene, a Volatile Brood Pheromone Involved in Social Regulation in the Honey Bee Colony (Apis mellifera) ». In : *PLoS ONE* 5.10 (21 oct. 2010). Sous la dir. de Martin Giurfa, e13531. ISSN : 1932-6203. DOI : 10.1371/journal.pone.0013531. URL : https://dx.plos.org/10.1371/journal.pone.0013531 (visité le 05/06/2019).

[26] Jérémy Rivière et al. « Modèle multi-agent d'auto-organisation pour le butinage au sein d'une colonie d'abeilles ». In : (2021), p. 27.

[27] Shaghayegh Roohi et al. « Review of Intrinsic Motivation in Simulation-based Game Testing ». In : *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. the 2018 CHI Conference. Montreal QC, Canada : ACM Press, 2018, p. 1-13. ISBN : 978-1-4503-5620-6. DOI : 10.1145/3173574.3173921. URL : http://dl.acm.org/citation.cfm?doid=3173574.3173921 (visité le 12/11/2019).

[28] T Schmickl et K Crailsheim. « TaskSelSim : a model of the self-organization of the division of labour in honeybees ». In : *Mathematical and Computer Modelling of Dynamical Systems* 14.2 (2008), p. 101-125.

[29] Thomas Schmickl et Karl Crailsheim. « Analysing honeybees' division of labour in broodcare by a multi-agent model ». In : (2008), p. 9.

[30] Jürgen Schmidhuber. « Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990–2010) ». In : *IEEE Transactions on Autonomous Mental Development* 2.3 (sept. 2010), p. 230-247. ISSN : 1943-0604, 1943-0612. DOI : 10.1109/TAMD.2010.2056368. URL : http://ieeexplore.ieee.org/document/5508364/ (visité le 15/11/2019).

[31] Thomas D. Seeley et Steven A. Kolmes. « Age Polyethism for Hive Duties in Honey Bees - Illusion or Reality ? » In : *Ethology* 87.3 (1991), p. 284-297. ISSN : 01791613, 14390310. DOI : 10.1111/j.1439-0310.1991.tb00253.x. URL : http://doi.wiley.com/10.1111/j.1439-0310.1991.tb00253.x (visité le 30/01/2020).

[32] G. Theraulaz, E. Bonabeau et J.-L. Deneubourg. « Response threshold reinforcements and division of labour in insect societies ». In : *Proceedings of the Royal Society B : Biological Sciences* 265.1393 (22 fév. 1998), p. 327-332. ISSN : 0962-8452. DOI : 10.1098/rspb.1998.0299. URL : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1688885/ (visité le 02/10/2019).

[33] Edward O. Wilson. « The relation between caste ratios and division of labor in the ant genus Pheidole (Hymenoptera : Formicidae) ». In : *Behavioral Ecology and Sociobiology* 16.1 (nov. 1984), p. 89-98. ISSN : 0340-5443, 1432-0762. DOI : 10.1007/BF00293108. URL : http://link.springer.com/10.1007/BF00293108 (visité le 07/07/2021).

[34] Mark L. Winston. *The Biology of the Honey Bee*. Harvard University Press, 1991. 300 p. ISBN : 978-0-674-07409-5.

[35] Mark L. Winston et al. « The role of queen mandibular pheromone and colony congestion in honey bee (Apis mellifera L.) reproductive swarming (Hymenoptera : Apidae) ». In : *Journal of Insect Behavior* 4.5 (sept. 1991), p. 649-660. ISSN : 0892-7553, 1572-8889. DOI : 10.1007/BF01048076. URL : http://link.springer.com/10.1007/BF01048076 (visité le 28/05/2019).

[36] Michael Wooldridge, Nicholas R. Jennings et David Kinny. « A methodology for agent-oriented analysis and design ». In : *Proceedings of the third annual conference on Autonomous Agents - AGENTS '99*. the third annual conference. Seattle, Washington, United States : ACM Press, 1999, p. 69-76. ISBN : 978-1-58113-066-9. DOI : 10.1145/301136.301165. URL : http://portal.acm.org/citation.cfm?doid=301136.301165 (visité le 21/07/2021).

DOCTORAT BRETAGNE LOIRE / MATHSTIC

Université de Bretagne Occidentale

Titre : Visualisation et Interactions avec une colonie d'abeilles virtuelle : simulation, complexité et pédagogie.

Mot clés : de 3 à 6 mots clefs

Résumé : Eius populus ab incunabulis primis ad usque pueritiae tempus extremum, quod annis circumcluditur fere trecentis, circummurana pertulit bella, deinde aetatem ingressus adultam post multiplices bellorum aerumnas Alpes transcendit et fretum, in iuvenem erectus et virum ex omni plaga quam orbis ambit inmensus, reportavit laureas et triumphos, iamque vergens in senium et nomine solo aliquotiens vincens ad tranquilliora vitae discessit. Hoc inmaturo interitu ipse quoque sui pertaesus excessit e vita aetatis nono anno atque vicensimo cum quadriennio imperasset. natus apud Tuscos in Massa Veternensi, patre Constantio Constantini fratre imperatoris, matreque Galla. Thalassius vero ea tempestate praefectus praetorio praesens ipse quoque adrogantis ingenii, considerans incitationem eius ad multorum augeri discrimina, non maturitate vel consiliis mitigabat, ut aliquotiens celsae potestates iras principum molliverunt, sed adversando iurgandoque cum parum congrueret, eum ad rabiem potius evibrabat, Augustum actus eius exaggerando creberrime docens, idque, incertum qua mente, ne lateret adfectans. quibus mox Caesar acrius efferatus, velut contumaciae quoddam vexillum altius erigens, sine respectu salutis alienae vel suae ad vertenda opposita instar rapidi fluminis irrevocabili impetu ferebatur. Hae duae provinciae bello quondam piratico catervis mixtae praedonum.

Title: Visualisation and Interactions with a virtual honey bee colony : simulation, complexity and pedagogy

Keywords: de 3 à 6 mots clefs

Abstract: Eius populus ab incunabulis primis ad usque pueritiae tempus extremum, quod annis circumcluditur fere trecentis, circummurana pertulit bella, deinde aetatem ingressus adultam post multiplices bellorum aerumnas Alpes transcendit et fretum, in iuvenem erectus et virum ex omni plaga quam orbis ambit inmensus, reportavit laureas et triumphos, iamque vergens in senium et nomine solo aliquotiens vincens ad tranquilliora vitae discessit. Hoc inmaturo interitu ipse quoque sui pertaesus excessit e vita aetatis nono anno atque vicensimo cum quadriennio imperasset. natus apud Tuscos in Massa Veternensi, patre Constantio Constantini fratre imperatoris, matreque Galla. Thalassius vero ea tempestate praefectus praetorio praesens ipse quoque adrogantis ingenii, considerans incitationem eius ad multorum augeri discrimina, non maturitate vel consiliis mitigabat, ut aliquotiens celsae potestates iras principum molliverunt, sed adversando iurgandoque cum parum congrueret, eum ad rabiem potius evibrabat, Augustum actus eius exaggerando creberrime docens, idque, incertum qua mente, ne lateret adfectans. quibus mox Caesar acrius efferatus, velut contumaciae quoddam vexillum altius erigens, sine re-

spectu salutis alienae vel suae ad vertenda opposita instar rapidi fluminis irrevocabili impetu ferebatur. Hae duae provinciae bello quondam piratico catervis mixtae praedonum.