وزارة التعليم العالي و البحث العلمي

Université Badji Mokhtar –Annaba-

Badji Mokhtar –Annaba- University

جامعة باجي مختار- عنابة-

Année**: 2015-2016**

Faculté des sciences de l'ingéniorat
Département d'informatique

T H È S E

Pour obtenir le diplôme de
Doctorat en sciences

Résolution des problèmes de coordination d'agents réactifs dans un environnement incertain

Filière : Informatique
Spécialité : Intelligence Artificielle

Préparée par

Ouarda ZEDADRA

Président :	Labiba SOUCI-MESLATI	Pr.	Université Badji Mokhtar - Annaba
Directeur de thèse :	Hamid SERIDI	Pr.	Université 8 Mai 1945 - Guelma
Examinateurs :	Mohamed-Khireddine KHOLLADI	Pr.	Université Echahid Hamma Lakhdar- El oued
	Yamina MOHAMED BEN ALI	Pr.	Université Badji Mokhtar - Annaba
	Abdelkrim AMIRAT	Pr.	Université Med Chérif Messaidia -Souk Ahras

Remerciements

Merci du plus profond de mon coeur mon Dieu, de me faire tenir dans cet espoir et de me donner tout de même cette force pour mener à bien ce travail. Merci de m'avoir écoutée et de m'offrir des gens qui m'ont soutenue jusqu'à la fin.

Je remercie le Pr. Labiba SOUICI-MESLATI d'avoir accepté de présider le jury.

Je remercie vivement le Pr. Hamid SERIDI pour sa confiance et son soutien. Je tiens à exprimer toute ma reconnaissance à mon directeur de thèse, pour avoir m'orienter dans mes travaux de recherche tout en me laissant une grande liberté.

Je dois beaucoup au Dr. Nicolas JOUANDEAU qui m'a beaucoup aidé, encouragé et orienté depuis plusieurs années. Je le remercie pour ses nombreuses remarques et appréciations d'une extrême pertinence et je tiens en cette circonstance à lui exprimer ma profonde reconnaissance.

Je dois beaucoup au Pr. Giancarlo FORTINO qui, depuis quelques années m'a fait profiter de son immense expérience et m'a encouragé pour avancer dans mes travaux. Je le remercie infiniment de ses aides et je lui exprime ma profonde reconnaissance.

Je remercie le Pr. Yamina MOHAMED BEN ALI, le Pr. Mohamed-Khireddine KHOLLADI et le Pr. Abdelkrim AMIRAT de m'avoir fait l'honneur d'accepter d'être les examinateurs de mon travail, de participer au jury et de s'être intéressés à mes travaux.

Toute les remerciements aux membres de ma famille spécialement mes parents.

A celui qui ma franchir les frontières et ma ouvrir l'espoir à mon chèr ami Medhat SAAD. Je te remercie du fond de mon cœur pour tout ce que tu as fait pour me rendre à ce point, pour ta patience et tes conseils.

Enfin, je tiens à remercier sincèrement tous ce qui ont contribué de près ou de loin à l'achèvement de ce travail.

Que tous en soient remerciés.

Résumé

L'utilisation de plusieurs robots (agents) peut améliorer les performances du système, si le groupe est bien coordonné. Quand les buts des agents sont dépendants et que les actions des uns peuvent améliorer celles des autres et augmenter les performances du groupe, l'utilisation de plusieurs agents simples avec des capacités de réaction rapide et de marquage de leur environnement peut apporter des solutions à des problèmes donnés à travers des interactions simples entre eux et avec leur environnement, approche inspirée du vivant connue sous le nom de *résolution collective de problèmes*.

Dans le cadre de la coordination d'une communauté d'agents réactifs et pour répondre à des problèmes inconnus initialement, nous proposons et analysons un ensemble d'algorithmes de coordination réactive à base de stigmergie. Comme la proposition d'une technique de coordination générique reste encore un défis, nous étudions la coordination dans le problème de fourragement (*foraging problem*, comme ce dernier résume plusieurs problèmes du monde réel. En outre, il partage un ensemble d'aspects avec d'autres applications d'essaims de robotique ; comme par exemple, le besoin de coordonner de grands groupes d'individus et l'utilisation de robots avec des capacités de perception, d'actionnement et de calcul limitées. Nous considérons des interactions indirectes via des marquages passifs (par valeurs entières) et dynamiques (par phéromones digitales), et montrons comment la coordination réactive peut être efficace pour la résolution de problèmes. Les algorithmes proposés ont été validés par des simulations dans différentes configurations d'environnement, et les résultats de comparaison avec des travaux similaires sont jugés encourageants et très prometteurs.

Mots clés : Résolution collective de problèmes, systèmes multi-agents, agents réactifs, foraging multi-agents, coordination réactive, intelligence en essaim.

Abstract

Using multiple coordinated robots (agents) can improve system performance. When agent's goals are dependent and that their actions can improve the group's performance, the use of multiple simple agents with quick response and marking capabilities, can résolve problems through simple interactions between them and their environment. This kind of resolving problems inspired by animals is known as *collective problem solving*.

In the context of coordinating of group of reactive agents to resolve initially unknown problems, we propose and analyse a set of reactive coordination algorithms based on stigmergy. We study the problem of coordination in the *foraging*, as it summarizes several real-world problems. Further, it shares a number of aspects with other swarm robotics applications (the need to coordinate large groups of individuals and the use of robots with perception, actuating and limited computing capabilities). We consider indirect interactions via passive markings (using integer values) and dynamic markings (using digital pheromones), and show how reactive coordination can be effective in solving problems. The proposed algorithms were validated through simulations in different environmental settings, and comparaisons with similar works are considered very encouraging.

Keywords : collective problem solving, multi-agent systems, reactive gents, multi-agents foraging, reactive coordination, swarm intelligence.

مـلخـــص

يمكن تحسين أداء النظام باستخدام مجموعة من الروبوتات (الوكلاء) إذا كانت المجموعة منسقة تنسيقا جيدا. عندما تكون أهداف الوكلاء تتعلق ببعضها البعض وأن قراراتهم يمكن أن تحسن من اداء لآخرين و تزيد من فعالية المجموعة، فان استخدام مجموعة بسيطة من الوكلاء ذوو قدرات استجابة سريعة وقدرات تعليم بيئتهم، يمكن أن يقدموا حلولا لمشاكل معينة من خلال تفاعلات بسيطة فيما بينهم ومع بيئتهم، وهذه الاستراتجية مستوحاة من الأحياء المعروفة باسم الحل *الجماعي للمشاكل*.

في اطار تنسيق مجموعة من الوكلاء و للاجابة على مشاكل غير معروفة من البداية، نقترح و نحلل مجموعة من خوارزميات لتنسيق الوكلاء مستوحاة من نظام النمل الكلاسيكي. اخترنا دراسة التنسيق لتحسين اداء المجموعة في مشكلة معروفة و هي *البحث عن الطعام*. سبب اختيار هذه المشكلة عن غيرها هو انها مشكلة تلخص العديد من المشاكل في العالم الحقيقي. علاوة على ذلك، فانها تشاطر مجموعة من الجوانب مع تطبيقات او مشاكل آخرى من أسراب الروبوتات, على سبيل المثال: الحاجة إلى تنسيق مجموعات كبيرة من الوكلاء واستخدام الرؤية، الحركيات والقدرات الحسابية المحدودة. نهتم في هذا العمل علي التفاعلات غير المباشرة بين الوكلاء من خلال استعمال علامات ثابتة لا تتغير مع الوقت (اعداد صحيحة) و علامات ديناميكية (فيرومونات رقمية)، ونظهر من خلال النقطتين السابقتين مدى اهمية التنسيق التفاعلي في حل المشاكل. جميع الخوارزميات المقترحة في هذه الرسالة تم برمجتها و قد قمنا بمجموعة كبيرة من المحاكاة في إعدادات بيئية مختلفة. قد تم مقارنة النتائج المتحصل عليها لجميع الاقتراحات مع اعمال مماثلة وتعتبر النتائج المتحصل عليها جيدة و مشجعة للغاية.

كلمات مفتاحية: حل جماعي للمشاكل، نظام متعدد الوكلاء، وكيل تفاعلي، التنسيق التفاعلي، البحث عن الطعام متعدد الوكلاء، ذكاء الاسراب.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Introduction Générale

1 Positionnement scientifique

De l'Intelligence Artificielle à l'Intelligence Artificielle Distribuée

L'Intelligence Artificielle (IA) est définie comme l'élaboration des programmes informatiques imitant le comportement intelligent humain pour la résolution des problèmes donnés par intégration de capacités mentales de haut niveau tels que : l'apprentissage, l'organisation de la mémoire et le raisonnement [91]. Le but est de donner des notions de rationalité à travers des fonctions de décision autonomes et de perception pour commander un robot dans un milieu qui lui est inconnu. Elle trouve ses origines dans les travaux d'Alan Turing [136], où il propose une expérience connue sous le nom de *test de Turing* permettant de qualifier une machine de consciente.

A la différence de l'IA qui modélise le comportement intelligent d'un seul agent, l'intelligence artificielle distribuée (IAD) s'intéresse à des comportements intelligents qui résultent de l'activité coopérative de plusieurs entités (agents), elle partage la résolution du problème entre différents agents autonomes, dotés de capacités de perception et d'action sur l'environnement [47]. Le but de l'IAD est de concevoir des approches de développement de communautés d'agents intelligents qui puissent interagir de façon coopérative mais aussi conflictuelle ou concurrente afin de résoudre des problèmes complexes. Le partage de la résolution crée, par sa nature, plusieurs problèmes parmi lesquels les plus importants sont ceux du contrôle, de la coordination, de la communication et de l'organisation des agents.

Au même moment, une autre révolution a envahi l'IAD, où l'intelligence n'est pas uniquement individuelle mais également collective, provoquant l'établissement du domaine des Systèmes Multi-

Agents [125].

Les systèmes Multi-Agents

Un domaine de recherche relativement complexe, dérivé de l'IAD, est celui des systèmes multi-agents (SMA). La thématique SMA se focalise sur l'étude des comportements collectifs et sur la répartition de l'intelligence sur des agents plus au moins autonomes, capables de s'organiser et d'interagir pour résoudre des problèmes. Ferber [68], présente une étude détaillée sur les SMA dans son célèbre livre *"Les systèmes multi-agents. Vers une intelligence collective"*, où il classifie les différents types des SMA selon différents critères, ainsi que les enjeux de leur conception et de leur maitrise dans le cadre de l'informatique et de la robotique. Il définit deux types d'SMA *délibératifs (communicants)* ou *réactifs (collectifs)* selon que les agents utilisent un raisonnement symbolique et des capacités cognitives (anticipation, mémorisation...), ou des réactions simples de type stimulus-réponse [18]. Plusieurs méta-heuristiques inspirées des techniques collectives des insectes sociaux (éco-résolution [52], ACO [44]) ont montré leurs succès et provoquent ainsi l'intérêt pour ces approches dans la modélisation des systèmes complexes, ainsi que dans la robotique mobile. Cette approche a remis en cause l'utilisation des processus de prise de décision complexes basés sur des représentations symboliques (processus cognitif). Les interactions locales à travers des actions simples résultent en un comportement collectif globale complexe [135].

Nous nous sommes positionnés dans l'étude des SMA réactifs, où nous focalisons sur des questions centrales tels que : (1) comment coordonner à travers des règles simples, une suite d'agents réactifs, pour résoudre un problème donné ? et (2) comment introduire la coopération dans le système construit sans modifier l'aspect réactif des agents, pour améliorer encore les performances ?

2 Problématique de la recherche

Le problème abordé est celui de l'organisation d'une communauté d'agents réactifs de manière à produire une réponse collective capable de résoudre un problème inconnu initialement. Pour se faire, les agents doivent disposer de mécanismes de coordination leur permettant de s'organiser et de résoudre collectivement les problèmes sans aucune planification, ni anticipation. En effet, un système n'est performant que si les agents sont capables de s'organiser pour produire un tout

supérieur à la somme des performances individuelles. La coopération est parfois nécessaire à l'accomplissement de certaines tâches et peut être raison d'accroissement des performances. Plusieurs situations de coopération peuvent se présenter si l'on demande à plusieurs robots d'effectuer certaines tâches [68] : (1) les agents ont des buts indépendants et ils doivent s'éviter ou synchroniser leurs actions pour résoudre les conflits éventuels d'accès à des ressources communes, et (2) les agents ont des buts dépendants et les actions des uns peuvent améliorer celles des autres et donc augmenter les performances du groupe tout entier, Le système multi-robot pourrait alors accomplir une tâche dans *un temps plus rapide, moins cher et plus efficace* [4]. Le principe général consiste à utiliser les capacités des agents réactifs à réagir aux modifications de l'environnement et souvent à marquer cet environnement pour coordonner les actions des agents entre eux.

La résolution collective de problème, branche des SMA est une approche inspirée du vivant, repose sur les interactions d'un grand nombre d'agents simples entre eux et avec leur environnement, capable de faire émerger des structures ou des organisations constituant des solutions à des problèmes donnés. Le moyen de coordonner ces agents dans de telles conditions consiste à utiliser des techniques de coordination réactive. Ces dernières se basent presque toute à l'utilisation des techniques du champs de vect. Comme la proposition d'une technique de coordination réactive générique reste un travail ambitieux, une première ébauche consiste à se concentrer sur un problème dans lequel la notion d'interaction est fondamentale avant de proposer des approches plus génériques.

3 Objectifs de la recherche et contributions

L'objectif à long terme est de disposer de mécanismes permettant aux agents réactifs de s'organiser de manière à produire une réponse collective capable de résoudre un problème inconnu initialement. Notre travail se positionne dans cette dernière branche de coordination, connu sous le nom *coordination coopérative*. Proposer un mécanisme de coordination générique restent encore un défis, et nous focalisons sur une classe de problèmes dans laquelle la notion d'interaction (coordination coopérative) est fondamentale tels que : le problème de navigation en robotique mobile : *recherche des échantillons (Multi-Target Search), la couverture (coverage) et le foraging multi-agents (foraging)*. Nous pouvons finalement particulariser notre objectif de recherche à la résolution collective de problème par un ensemble d'agents réactifs, disposant de mécanismes de coordination leurs per-

mettant d'augmenter les performances du groupe. La résolution d'un problème est alors obtenue par un ensemble de coopérations locales (via phéromones ou marquage) produisant au niveau globale une auto-organisation ou émergence d'une structure organisationnelle cohérente et performante.

Pour atteindre ces objectifs, nous ajoutons les contributions suivantes :

1. Proposition d'un algorithme de coordination à base de stigmergie, applicable pour tout problème de navigation robotique, qui permet d'assurer une large dispersion des agents pour accélérer le processus d'exploration dans des environnement inconnus.
2. proposition d'un algorithme de foraging qui utilise un marquage passif par valeurs entières et qui introduit la forme de coopération intentionnelle en terme d'interaction simples et locales dans l'approche réactive pour garantir un accroissement des performances.
3. Adaptation de l'algorithme de foraging par marquage passif à un algorithme qui utilise la marquage dynamique.
4. Adaptation de l'algorithme de foraging pour qu'il prend en charge la gestion de l'énergie.
5. Généralisation des différents algorithmes à un Framework pour foraging

4 Plan de la thèse

Ce manuscrit s'organise ainsi autour de la question de coordination réactive d'un ensemble d'agents simples pour résoudre un problème donné. Nous étudions par ce manuscrit deux cases complémentaires pour répondre à cette question. Dans le premier cas, nous montrons le bénéfice de la coordination réactive pour la résolution de problèmes. Dans le deuxième cas, nous prouvons par les résultats quantitatifs l'effet positif de la coopération pour améliorer les performances de groupe, où nous ajoutons des règles de coopération simples à base de phéromones digitales (jugé intentionnelle par un observateur extérieur). Des structures auto-organisées émergent dans les deux cases.

Ainsi le manuscrit est organisé en deux grandes parties. La première présente un état de l'art et la deuxième partie est consacrée à la conception et l'implémentation de nos propositions.

Première partie : La première partie se trouve subdivisée en deux chapitres :

– Le premier chapitre, présente le domaine SMA autant que contexte de recherche pour notre problématique, nous rappelons les concepts marquants de ce paradigme de l'agent aux SMA, aux modèles de coordination et de coordination réactive, l'incertitude dans les systèmes multi-agents et les domaines d'application.

– Le deuxième chapitre, présente le contexte applicatif de nos propositions par présentation des deux domaines qui sont directement liés à notre travail à savoir l'intelligence en essaim et la robotique en essaim. Une deuxième partie du chapitre est consacrée à la présentation des problèmes d'exploration et de foraging multi-agents avec présentation des travaux reliés, une taxonomie de foraging est aussi proposée, selon laquelle nous comparons qualitativement les travaux reliés.

Deuxième partie : Cette partie se trouve subdivisée en deux chapitres aussi :

– Le troisième chapitre, aborde la conception en matière des algorithmes proposés. Nous présentons alors un algorithme de coordination réactive conçu pour tout problème de navigation de robotique, nous adaptons par la suite l'algorithme pour le problème de foraging avec marquage passif et dynamique, la où nous ajoutons de nouvelles règles de coordination. Nous définissons à la fin, un Framework pour le foraging sous forme de diagramme de conception UML.

– Le quatrième chapitre, décrit les différents scénarios de simulations que nous avons menées dans le cadre de ce travail, avec les résultats obtenus et des comparaisons quantitatives avec des travaux similaires.

Nous terminerons ce manuscrit par une conclusion où nous synthétisons ce qui a été proposé pour répondre à notre problématique, et quelques perspectives pour améliorer nos travaux de recherches.

Chapitre 1

Les Systèmes Multi-Agents

Sommaire

1 Introduction

Nous nous intéressons dans un premier temps à la notion de l'agent isolé qui occupe une place prépondérante en Intelligence Artificielle où nous présentons quelques définition du terme agents ainsi que les caractéristiques reliées. Dans un second temps, nous nous intéressons à la dimension sociale des agents qui constitue le propos de l'Intelligence Artificielle Distribuée où nous présentons certains concepts relatifs aux systèmes multi-agents (l'environnement, l'interaction, la coopération, la communication, ... etc.). Nous nous attardons par la suite sur les mécanismes de coordination, en particulier sur la coordination réactive. Finalement nous introduisons quelques propriétés des systèmes multi-agents qui sont jugées essentielles dans la suite de notre travail (l'incertitude dans les systèmes multi-agents) et les potentiels domaines d'applications de ce paradigme.

2 Agent : Concepts Fondamentaux

Nous présentons dans cette section la notion d'agent, commençant par présenter et discuter les différentes définitions d'un agent. Nous présentons par la suite les architectures des agents et nous essayons à la fin de cette partie de donner une définition à l'agent relative au cadre de notre travail.

2.1 La Diversité des Définitions

Le caractère interdisciplinaire des agents qui, d'un côté, sont sujets aux effets des différentes directions de recherches scientifiques et, de l'autre, reflètent les besoins d'applications pratiques a rendu la définition du terme *agent* n'est pas aisée. Toutefois, Il n'y a pas une définition acceptée en unanimité pour la notion d'agent et plusieurs chercheurs ont donné des essais de définition de ce terme dans le cadre de leurs travaux. Nous présentons et discutons dans ce qui suit une multitude de définitions du terme *agent*.

Definition 2.1 ***Agent selon Wooldridge et Jennings*** *[142]*

"An agent is a computer system that is situated in some environment and that is capable of autonomous action in this environment in order to meet its design objectives."

Un agent selon cette définition est vu comme une entité abstraite qui reflète la notion d'un système informatique simulé qui n'as pas d'existence réel et qui agit d'une façon autonome pour atteindre

ses objectifs internes.

Definition 2.2 ***Agent selon Russell et Norvig*** *[114]*

"An agent is just something that acts."

Un agent selon ces auteurs est une entité qui agisse sur un environnement dont elle fait partie.

Les définitions présentées précédemment considèrent l'agent comme une entité qui n'a de lien direct qu'avec le sens premier du mot : *"un agent est celui qui agit"* [1]. Ils considèrent les agents comme des entités simulés qui n'ont pas d'existence réelle et qui répondent à leurs objectifs par une séquence d'actions selon les informations dont ils capturent depuis leur environnement. La dimension sociale dans ces définitions a été exclue et un agent existe en isolation et interagit avec son environnement. Nous pouvons conclure trois concepts communs de la notion agent à partir des définitions précédentes. La Figure 1.1, montre une représentation d'un agent selon ces dernières :

– *Autonomie :* l'autonomie est une caractéristique majeure qui permet de différencier un agent d'un simple programme informatique. Un agent est dit autonome, si la prise de décision est basée sur son état interne et les observations dont ils reçoit sans l'intervention d'un tiers (humain ou système informatique). La capacité d'être autonome est considérée comme un trait d'intelligence [6].

– *Réactivité :* des capacités telles que la perception, et l'adaptation aux changements de l'environnement forment des points clés de l'autonomie d'un agent. Les agents sont situés dans un environnement (monde physique, internet, ...) et sont capables de percevoir cet environnement c'est-à-dire récolter et extraire l'information qui lui permet de réagir. De plus ils peuvent évoluer en fonction de changements qui interviennent dans cet environnement.

– *Pro-activité :* le comportement des agents n'est pas une simple réponse aux changements de leur environnement. En fait, ils poursuivent leur(s) propre(s) but(s).

Les définitions suivantes diffères de celles présentées au dessus par leur considération de la dimension sociale d'un agent et donc de sa capacité d'interaction avec d'autres agents et non seulement avec son environnement :

Definition 2.3 ***Agent selon Shoham*** *[120]*

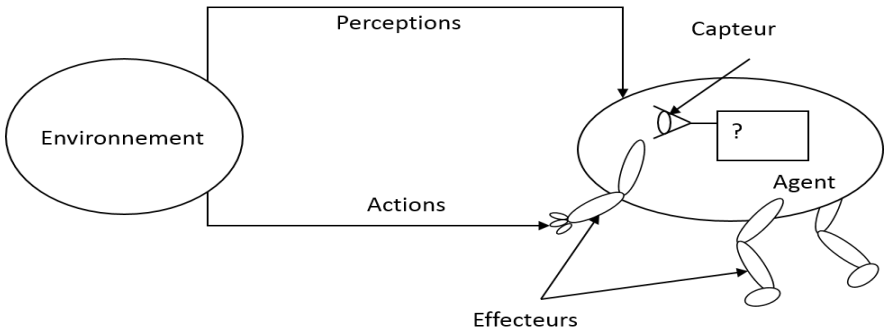


FIGURE 1.1 – Représentation d'un agent [114]

"Entity that functions continuously and autonomously in an environment in which other processes take place and other agents exist"

L'auteur définit un agent comme une entité autonome qui agit sur un environnement où d'autres agents existent. L'auteur déclare que la notion d'un agent reste croustillante et un agent doit souvent être de haut niveau, ainsi certaines mesures doivent êtres prises en compte pour les distinguer des autres composantes logicielles ou matérielles. Cette distinction chez Shoham [121] est assurée par l'ajout de composantes mentales tels que les croyances, les capacités, les choix et les engagements [1].

Definition 2.4 ***Agent selon Jaques Ferber*** *[68]*

"Un agent est une entité physique ou virtuelle qui :

- *Est capable d'agir dans un environnement ;*
- *Peut communiquer directement ou non avec les autres agents ;*
- *Peut être menée par un ensemble de tendances (sous la forme d'objectifs individuel sous de fonction de satisfaction/survie qu'elle essaie d'optimiser) ;*
- *Possède des ressources propres ;*
- *Capable de percevoir son environnement (mais de manière limitée) ;*
- *N'a qu'une représentation partielle de son environnement ;*
- *Possède des compétences et peut offrir des ressources ;*
- *Peut être capable de se reproduire ;*
- *Et dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences qu'il possède tout en dépendant de ses perceptions, de ses représentations et des*

1. "An agent is an entity whose state is viewed as consisting of mental components such as beliefs, capabilities, choices and commitments" [121]

communications qu'il reçoit."

Ferber donne une définition plus complète qui touche plusieurs aspects du terme agent. Cette définition constitue en faite une synthèse des différentes définitions présentées dans ce manuscrit. Elle ajoute plusieurs autres caractéristiques à savoir la capacité de communication, la capacité de perception limité d'un agent, un agent peut être aussi une entité concrète, un robot par exemple et la capacité de reproduire.

Definition 2.5 ***Agent selon Demazeau et Müller*** *[37] "Un agent est une entité réelle ou virtuelle, dont le comportement est autonome, évoluant dans un environnement, qu'il est capable de percevoir, sur lequel il est capable d'agir et d'interagir avec les autres agents"*

Les auteurs dans cette définition évoquent aussi qu'un agent peut être un programme ou encore un robot (voir un avion ou un aspirateur par exemple). Ils rajoutent encore la notion de sociabilité d'un agent et sa capacité d'interagir avec d'autres agents qui partagent le même environnement de travail.

En comparant les définitions ci-dessus on peut distinguer deux tendances différentes, la première considère que l'agent peut être défini indépendamment tandis que la deuxième voit l'agent comme entité agissant dans une société d'autre agents, donc le paradigme des Systèmes Multi-Agents (SMA). Depuis ces dernières définitions, plusieurs autres caractéristiques peuvent être ajoutés à la notion d'agent. Comme l'agent ne peut exister comme une entité pour lui seul, et doit rencontrer d'autres agents dans son environnement, il possède donc une *dimension sociale*. La distinction entre l'existence réel ou non d'un agent est fondamentale, puisque la simulation n'est qu'un modèle de la réalité, et qu'elle est sujette aux biais de programmation [123]. Toutefois, elle constitue une étape préliminaire pour la compréhension des systèmes et les problèmes qui ont découlent avant leur réalisation.

- *Habilité sociale :* les agents interagissent avec d'autres agents et peuvent parfois mettre en œuvre des processus de coopération ou négociation pour accomplir un but.
- *Réels (vivants ou robotisés)* : ont une existence physique et qui agissent dans un monde réel [68].
- *Simulés* : ont une existence virtuelle [68].

2.2 Classification

Selon l'axe *décisionnel*, différents modèles d'agents peuvent être envisagés. La vision décisionnelle s'appuie sur le degré de couplage à l'environnement. Un agent est réactif [19] [129] [2], hybride [29] [60] ou cognitif [41] [33].

2.2.1 Réactifs

Les agents réactifs ne peuvent que réagir à des stimuli simples provenant de leur environnement, et leur comportement est alors simplement dicté par leur relation à leur entourage sans que ces agents ne disposent d'une représentation des autres agents ou de leur environnement (leur comportement est basé sur des réactions à des stimuli). Ils n'ont pas de mémoire individuelle. La force des agents réactifs vient de leur capacité à se mettre en groupe, c'est-à-dire de constituer des colonies capables de s'adapter à leur environnement. Ainsi, ce n'est pas au niveau de l'individu que les agents réactifs sont intéressants, mais au niveau globale, au niveau de la population qu'ils offrent des comportements qualifies d'intelligents et complexes. Ils peuvent faire face à des tâches complexes et ainsi rivaliser en termes de performances avec des agents plus sophistiqués mais moins nombreux [68]. Le temps de réponse octroyé à un agent pour décider quelle action réalisée est dépendant du système. Les exigences concernant ce délai peuvent donc différer d'une application à une autre. Ainsi, une forte réactivité par minimisation du temps de réaction est obtenue si un agent réalise un maximum de tâches en un minimum de temps. De manière générale, la simplification du niveau délibératif induit à une grande réactivité.

L'architecture de subsomption est l'une des architectures de contrôle réactives les plus populaires qui a été proposée en opposition à IA traditionnelle. Au lieu d'orienter les comportements par des représentations mentales symboliques du monde, l'architecture de subsomption réunies l'information sensorielle à la sélection de l'action d'une manière intime et bottom-up [6]. Elle décompose le comportement globale en sous-comportements. Ces sous-comportements sont organisés en une hiérarchie de couches. Chaque couche met en œuvre un niveau particulier de compétence comportementale, et des niveaux plus élevés sont capables de subsumer niveaux inférieurs afin de créer un comportement viable [20].

2.2.2 Agents Cognitifs

Les agents cognitifs, peuvent résoudre des problèmes compliqués de manière relativement individuelle, du fait qu'ils disposent de capacités cognitives importantes. Leur représentation interne et les mécanismes d'inférence dont ils disposent leur permettent de fonctionner indépendamment des autres agents et leur offrent une grande souplesse dans l'expression de leur comportement [68]. Leur aptitude délibérative s'appuie sur la représentation symbolique de l'univers, des raisonnements à l'aide de systèmes formels et des communications de haut niveau. Ils offrent des temps d'exécution lentes, complexité croissante, fragilité et coût élevé par rapport aux agents réactifs [123].

L'architecture cognitive repose sur le traitement d'informations symboliques et suppose donc l'existence d'un langage sur lequel pourront raisonner les agents [57]. L'une des architectures cognitives les plus connues est l'architecture BDI (Beliefs Desires-Intentions) [111] qui permet aux agents de raisonner sur une base de croyances (Beliefs) pour répondre au but qu'ils se sont fixés (Desires). Ils produisent ensuite des intentions, c'est à dire des actions qu'ils souhaitent effectuer et qui pourront ou non se réaliser.

2.2.3 Agents Hybrides

Les systèmes réactifs peuvent bien convenir pour certains types de problèmes et moins bien pour d'autres. De même, pour la plupart des problèmes, les solutions cognitives basée uniquement sur la représentation symbolique et la planification, ne conviennent pas non plus. Une architecture hybride vise à combiner les deux approches pour répondre à des problèmes qui nécessite réactivité et planification pour être résolus [28]. Dans ce cas, un agent est composé de plusieurs couches, arrangées selon une hiérarchie. Ainsi, la couche réactive se positionne au plus bas niveau de l'architecture, qui prend ses décisions en se basant seulement sur des données de perception. La couche intermédiaire fait abstraction des donnés brutes et travaille plutôt avec une vision qui se situe au niveau des connaissances de l'environnement. Finalement, la couche supérieure se charge des aspects sociaux de l'environnement [28].

De façon générale, les agents dans une architecture hybrides ont des capacités réactives qui sont

subordonnées à des couches cognitives. D'autres part, des techniques d'apprentissages permettant d'augmenter les capacités limitées des agents réactifs purs [123].

2.3 La notion d'agent dans le cadre de notre travail

D'après la multitude de définitions que nous avons présentées, nous avons pu remarquer que la définition d'un agent diffère d'un auteur à l'autre selon le problème traité et les capacités nécessaires pour répondre à ce problème. Il parait donc essentiel de donner une définition d'un agent dans le cadre de notre travail, ainsi que les capacités assurés par ces agents. Nos donnons une définition proche de celle de Ferber [68].

Definition 2.6 ***Un agent dans le cadre de notre travail***

"Un agent est une entité situé dans un environnement capable de : (1) percevoir localement cet environnement, (2) agir de manière autonome et réactive sur cet environnement, (3) communiquer indirectement et localement à travers des phéromones avec d'autres agents afin de répondre à un problème inconnu initialement"

Les agents utilisés dans le cadre de ce travail sont des agents réactifs dotés de capacités de perception, de mémorisation et de calcul limités. Ils peuvent communiqué leur information avec les autres agents à travers l'environnement considéré comme leur mémoire partagée. Les agents décident par eux mêmes sans l'intervention d'un tiers et leurs décisions sont de type stimulus-réponse. Aucune intelligence ni associée à ces agents au niveau individuel mais leurs interactions locales provoquent un comportement collective jugé d'intelligent et complexe (intelligence en essaim). Les agents ne sachent pas leurs objectifs globaux mais contribuent à travers leurs interactions a répondre à un problème initialement inconnu.

Étant donnée une mission a remplir, à priori distribuée dans l'espace (ramasser des objets, chercher des matériaux...). Utiliser une colonie de robots présentera certains avantages par rapport à un robot solitaire. Un premier avantage concerne le coût de production minimale, du fait que le groupe sera simple dans sa conception par rapport à un agent isolé. Un deuxième avantage est que la répartition d'une colonie peut s'ajuster dynamiquement et automatiquement à la distribution dans l'espace et le temps du problème considéré. Un troisième avantage attendu est qu'un ensemble de robots est plus robust qu'un seul robot utilisé pour la même tâche. Donc l'utilisation d'un système multi-agents

(voir multi-robots) au lieu d'un agent en isolation portera plusieurs avantages non négligeables. Cependant plusieurs effets indésirables ou difficultés de conception peuvent apparaître dans un système multi-agents : en particulier pour qu'une coopération soit mise en place, il faut souvent que les agents puissent interagir, communiquer et s'organiser. Ces notions et bien d'autres font l'objet de la section suivante.

3 Des agents aux Systèmes Multi-Agents (SMA)

Depuis les années 80, les systèmes multi-agents ne cessent de se développer et d'étendre leur domaine d'application. Malgré toutes ces années d'existence, aucune définition, que ce soit aussi bien pour le terme d'agent que de système multi-agent, n'a encore fait l'unanimité au sein de la communauté. Le thème des systèmes multi-agents (SMA), s'il n'est pas récent, est actuellement un champ de recherche très actif. Cette discipline est à la connexion de plusieurs domaines en particulier de l'intelligence artificielle, des systèmes informatiques distribués. C'est une discipline qui s'intéresse aux comportements collectifs produits par les interactions de plusieurs entités autonomes et flexibles appelées agents, que ces interactions tournent autour de la coopération, de la concurrence ou de la coexistence entre ces agents [28].

Dans la partie précédente nous avons présenté le concept d'agent isolé comme une entité située dans un environnement qu'elle perçoit et sur lequel elle peut agir. Cette partie se concentre sur des systèmes composés de plusieurs agents. Elle met l'accent sur le concept d'interaction, élément central pour ces systèmes puisqu'il représente ce qui permet de construire une réponse collective à partir de réponses individuelles [135].

3.1 Définitions

Un SMA peut être ouvert ou fermé, homogène ou hétérogène. Au niveau de sa conception, la localité et la décentralisation sont imposées dans un SMA. Une vision locale, parce que chaque agent est responsable de ses connaissances et de ses actions, et également de l'organisation qu'il met en place avec d'autres agents, un agent ne possédant toutefois pas une vue globale du SMA. Une vision décentralisée, parce que les tâches et les compétences sont distribuées sur les agents. Les

agents sont dotés de capacités de représentation et de contrôle des interactions afin qu'ils participent aux processus sociaux dans lesquels ils sont impliqués. Nous donnons dans les sections suivantes deux définitions basiques d'un système multi-agents.

Definition 3.1 ***SMA selon Ferber*** *[68]*

"Un Système multi-agents est un système mettant en œuvre des programmes informatiques capables d'avoir un comportement propre (les agents) au sein d'un environnement global commun."

En 1991 Demazeau et Müller proposent une autre définition (plus large) des SMA qu'ils on appelé « modèle voyelle AEIO », et cette définition est comme suit :

Definition 3.2 ***SMA selon Demazeau et Müller***

- *Un ensemble B d'entités plongées dans un environnement E (E est caractérisé par l'ensemble des états de l'environnement S) ;*
- *Un ensemble A d'agents avec $A \subseteq B$;*
- *Un système d'action (opérations) permettant à des agents d'agir dans E (une opération est une fonction de $S \Rightarrow S$;*
- *Un système de communication entre Agents (envoi de messages, diffusion de signaux,... (I comme interaction) ;*
- *Une organisation O structurant l'ensemble des agents et définissant les fonctions remplies par les agents (notion de rôle et éventuellement de groupes).*

Dans cette définition Demazeau a mis l'accent sur quatre axes caractérisant les SMA : Agent, Environnement, Interactions et Organisations. Il découpe les SMA en quatre axes, que nous développerons tout au long de ce chapitre.

SMA = Agent + Environnement + Interaction + Organisation (AEIO).

Nous détaillons dans ce qui suit les trois axes : environnement, interactions et organisations.

3.2 Environnement

Correspond à l'ensemble d'éléments décrivant le monde dans lequel évoluent les agents, ça sert d'intermédiaire pour les interactions entre les agents ainsi que pour l'accès aux ressources [94]. Il

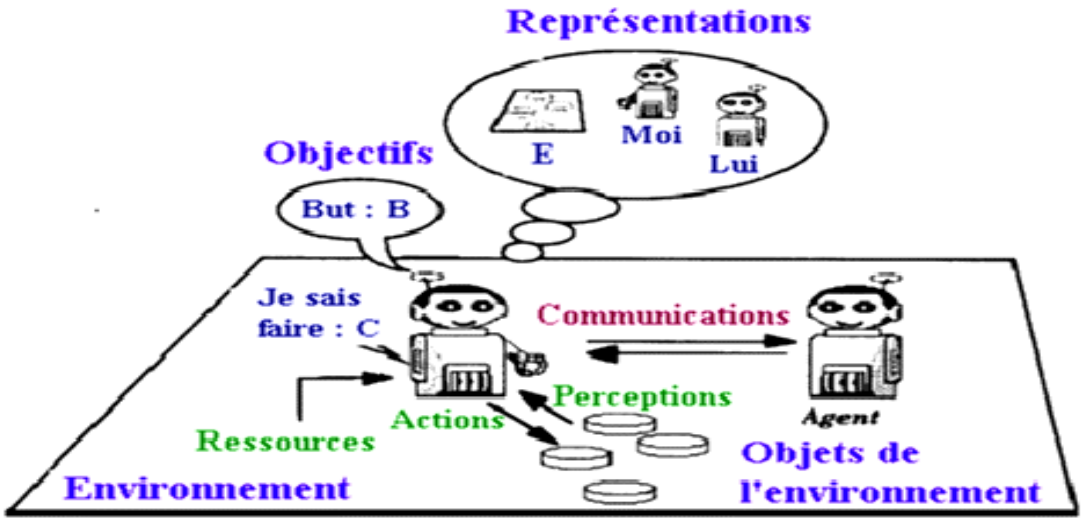


FIGURE 1.2 – Classification des agents [68]

existe plusieurs types d'environnements [22], parmi lesquels on peut citer :

– *Environnement totalement observable ou partiellement observable :* Un environnement est dit totalement observable si l'agent perçoit toutes les informations nécessaires à la décision. Il est dit partiellement observable lorsque l'agent ne perçoit qu'une partie de l'environnement.

– *Environnement déterministe ou non déterministe :* Si l'état suivant de l'environnement est déterminé d'une manière unique par l'état courant et l'action de l'agent, alors l'environnement est déterministe. Si le résultat est incertain, notamment si, par suite d'une action de l'agent, l'environnement peut évoluer de différentes manières, alors on est dans le cas non déterministe ou encore stochastique.

– *Environnement statique ou dynamique :* Si l'environnement ne peut pas changer d'état sans l'intervention de l'agent, on est dans le cas statique. L'environnement est dynamique si son état peut se modifier sans l'action de l'agent dans l'intervalle de temps entre deux perceptions de l'agent.

– *Environnement discret ou continu :* Si tout passage d'un état de l'environnement à un autre nécessite le passage par une séquence d'états intermédiaires, alors on a un environnement continu ; autrement, l'environnement est discret.

– *Environnement épisodique ou séquentiel :* dans un environnement épisodique, l'expérience de l'agent peut être divisée en épisodes, chaque épisode consiste en une phase de perception, sui-

vie de l'exécution d'une action, chaque épisode est indépendante des autres et le résultat de l'exécution d'action ne dépend que de l'épisode courante. Dans un environnement séquentiel, l'exécution d'une action peut influencer toutes les actions à venir.

3.3 Interaction

On peut dire qu'il existe une interaction lorsque la dynamique propre d'un agent est perturbée par les influences des autres. Chaque agent est ainsi lié à un ensemble d'autres agents que l'on appelle ses accointances. Jaques Ferber [68] définit ce concept comme suit : *"Une interaction est une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques"*. Les interactions entre agents peuvent variées selon les situations dont se trouvent ces agents (coexistence, compétition ou coopération). Une situation d'interaction est vue comme *"un ensemble de comportements résultant du regroupement d'agents qui doivent agir pour satisfaire leurs objectifs en tenant compte des contraintes provenant des ressources plus ou moins limitées dont ils disposent"* [68].

Selon la nature des buts, les relations aux ressources et les compétences des agents, Ferber [68] propose les trois catégories d'interaction suivantes :

- Coexister : chaque agent ne considère les autres agents que comme des composantes de l'environnement. Il n'y a aucune communication directe entre les agents.
- Compétition : le but de chaque agent est de maximiser sa propre satisfaction. La compétition entre agents peut avoir plusieurs sources : les buts des agents peuvent être incompatibles ou les ressources peuvent être insuffisantes.
- Coopération : le but des agents n'est plus seulement de maximiser leurs propres satisfactions mais aussi de contribuer à la réussite du groupe. Les agents travaillent ensemble à la résolution d'un problème commun. Deux méthodes fondamentales permettant la mise en œuvre de la coopération sont la collaboration et la coordination. La collaboration consiste à travailler à plusieurs sur un même projet. Elle désigne les techniques permettant aux agents de se répartir les tâches, les informations et les ressources. La coordination est l'ensemble des activités supplémentaires que les agents doivent exécuter et qui permettent d'améliorer leur comportement [13].

Les agents peuvent interagir en communiquant directement entre eux ou par l'intermédiaire d'un autre agent ou même en agissant sur leur environnement. Dans les SMA deux stratégies principales ont été utilisées pour supporter la communication entre agents : les agents peuvent échanger des messages directement ou ils peuvent accéder à une base de données partagées (appelée tableau noir ou "blackboard") dans laquelle les informations sont postées. Les communications sont à la base des interactions et de l'organisation sociale d'un SMA. La communication directe consiste à envoyer directement un message à un ou plusieurs agents. Elle est dite point à point lorsque le message est envoyé à un seul agent, alors que si l'information est envoyée à tous les autres agents elle est dite diffusion(ou broadcasting). Dans ce type de communication, le canal de transmission peut être un réseau informatique, un câble téléphonique, des ondes hertziennes, ... La communication indirecte est réalisée par modification de l'environnement. Les agents peuvent laisser des traces dans l'environnement afin d'indiquer aux autres quelle action a été exécutée [13].

3.4 Organisation

Les interactions entre agents sont à l'origine des multiples problématiques liées aux systèmes multi-agents. Or, l'état de l'organisation d'une société est la conséquence des interactions entre les agents. Réciproquement, le comportement des agents est contraint par l'ensemble des structures organisatrices de la société. Une organisation est un schéma de prise de décision et de communication appliqué à un ensemble d'acteurs qui réalisent un ensemble de tâches afin de satisfaire des buts tout en assurant un état global cohérent. Ce schéma peut être défini par le concepteur ou dynamiquement par les acteurs [84]. Dans un système multi-agents, l'organisation est une notion à dimension multiple, elle est implicite ou explicite, statique ou dynamique, prédéfinie ou émergente, ascendante ou descendante [86]. Une organisation peut s'instancier suivant plusieurs modèles : groupe, hiérarchie, de marché, etc [126] [142] [36] [79].

3.5 SMA dans le cadre de notre travail

Du fait que nous nous intéressons dans ce travail à la coordination réactive dans le cadre de la coopération et de la résolution collective de problèmes, nous focalisons sur certains classes des SMAs dont les caractéristiques sont les suivantes :

1. Des systèmes Multi-Agents simulés ;
2. Des systèmes réactifs composés d'un ensemble d'agents réactifs ;
3. Les systèmes réalisés sont homogènes ;
4. Les agents coopèrent pour la réalisation d'un objectif globale ;
5. Les agents peuvent communiquer indirectement entre eux (localement), par des phéromones digitales.

Dans la section précédente, nous avons présenté un ensemble de concepts liés aux systèmes multi-agents que nous jugeons utile de les connaître afin de rendre la lecture de ce manuscrit plus simple et lisible. Tandis que, dans la section suivante 4, nous focalisons sur les mécanismes de coordination dans des systèmes multi-agents et particulièrement la coordination réactive.

4 Mécanismes de coordination multi-agents

4.1 Définitions

La coordination d'actions a ainsi été décrite par Thomas W. Malone [85] comme *"l'ensemble des activités supplémentaires qu'il est nécessaire d'accomplir dans un environnement multi-agents et qu'un seul agent poursuivant les même buts n'accomplirait pas"*. Le choix du mécanisme de coordination à mettre en œuvre varie selon le domaine d'utilisation du système, du nombre d'agents qui le composent et leurs tâches à réaliser, de l'environnement et ses caractéristiques (stabilité, prédictibilité,...etc). Ce mécanisme a pour but de gérer les dépendances qui existent entre les activités des agents qui sont souvent incomplètement spécifiées.

4.2 Pourquoi coordonner ?

La coordination d'actions est nécessaire pour quatre raisons principales [68] :

1. *Information incomplète :* les agents ont besoin d'informations et de résultats que seuls d'autres agents peuvent fournir ;
2. *Les ressources sont limitées :* les ressources sont réduites et plusieurs agents se retrouvent à

les utiliser (temps, espace, énergie, argent ou outils), Il faut donc partager ces ressources de manière à optimiser les actions à effectuer, tout en essayant d'éviter les conflits éventuels ;

3. *Optimiser les coûts :* coordonner des actions permet aussi de diminuer les coûts en éliminant les actions inutiles et en évitant les redondances d'action ;

4. *Se franchir de la dépendance des objectifs :* permettre à des agents ayant des objectifs distincts mais dépendants les uns des autres de satisfaire ces objectifs et d'accomplir leur travail en tirant éventuellement parti de cette dépendance.

4.3 Mise en œuvre des mécanismes de coordination

Pour coordonner les activités des agents dans un SMA, il faut déterminer l'existence de situations d'interaction, les identifier et les gérer, mais le problème est comment mettre en œuvre une coordination au niveau du système. On doit opter pour une solution parmi les plusieurs possibilités d'implémentation des mécanismes de coordination, qui peuvent être réalisés au sein de l'agent, entre les agents, au travers de l'environnement,...etc. Toutes ces possibilités de mise en œuvre de mécanismes de coordination sont envisageables et peuvent être combinées pour donner naissance à d'autres. Les mécanismes multi-agents les plus courants et occupant une place privilégiée dans la littérature sont présentés par la Figure 1.3.

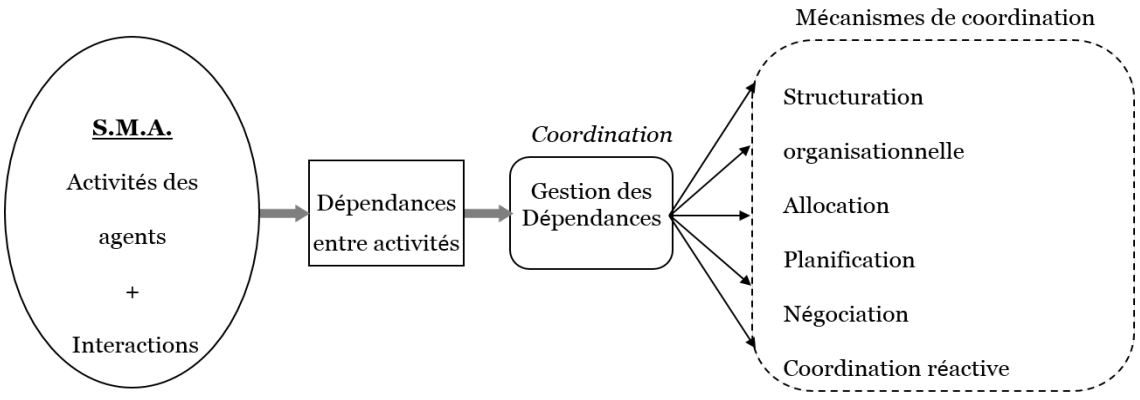


FIGURE 1.3 – Principaux mécanismes de coordination multi-agents [30]

4.3.1 Structuration Organisationnelle

Les organisations d'agent peuvent être classifiées selon plusieurs critères, ce qui implique l'existence de nombreuses classes d'organisation. Tandis que pour leurs structuration on trouve que trois configurations :

– *Organisation hiérarchique :* cette structure est inspirée des entreprises humaines imposant un certain ordre hiérarchique, dans ce type d'organisation chaque tâche est contrôlée par un agent gestionnaire qui coordonne son processus de réalisation [48].

– *Organisation latérale :* à l'opposé, dans une organisation latérale, il n'existe pas de gestionnaire unique et chaque tâche est réalisée de manière coopérative.

– *Organisation informelle :* la configuration intermédiaire et la plus courante d'organisation est la structure informelle dans laquelle cohabitent une certaine hiérarchie induite par des relations d'autorité et des contacts horizontaux directs entre les agents (Figure 1.4).

Bien entendu, il est généralement admis qu'aucune structure organisationnelle n'est appropriée à toutes les circonstances [35]. Ainsi, lorsqu'une situation évolue, l'organisation devrait se remettre en cause pour déterminer s'il est bénéfique pour elle de se restructurer [29] [27] [79].

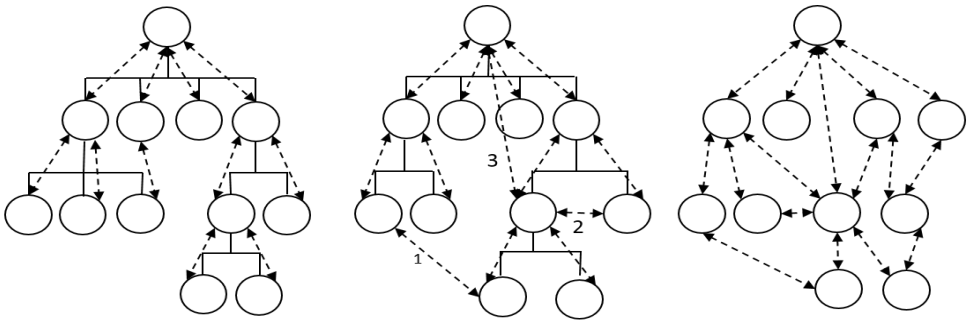


FIGURE 1.4 – Exemples de structures organisationnelles [99]

4.3.2 Allocation

La résolution de problèmes s'effectue généralement en quatre phases répondant à la question suivante : qui doit faire quoi et comment, en fonction des buts, des compétences des agents et des contraintes contextuelles ? Concernant l'allocation de tâches, celle-ci peut s'effectuer de deux manières [30] : centralisée où un seul agent décompose le problème en sous-problèmes et les réparti entre les autres agents, ou distribuée où chaque agent est capable de décomposer son problème en sous-problèmes et de répartir les tâches associées (Figure 1.5).

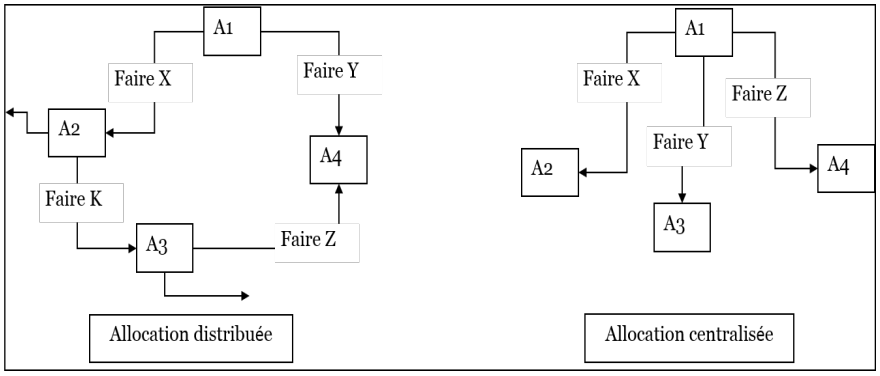


FIGURE 1.5 – Les deux manières d'allocation [30]

4.3.3 Planification

La planification multi-agents tente de coordonner les actions de plusieurs agents de manière à ce qu'un but commun soit atteint [63]. La planification, lorsqu'elle est appliquée à la coordination, consiste à développer un plan explicite qui prend en compte les actions et les interactions des agents dans la réalisation de buts. Elle définit notamment la synchronisation des actions et la gestion des accès concurrents aux ressources. La planification peut être centralisée ou distribuée et peut se décomposer en trois étapes : la construction de plans, la coordination de plans et l'exécution de plans. Un système de planification est alors composé d'un ensemble d'agents pouvant planifier, synchroniser ou exécuter des plans, un même agent pouvant accomplir une seule ou plusieurs de ces tâches (Figure 1.6).

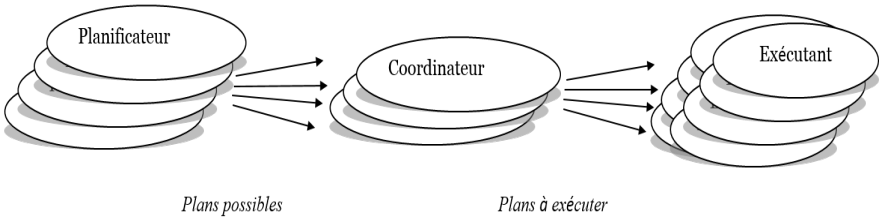


FIGURE 1.6 – Exemple d'un système de planification multi-agents [30].

4.3.4 Négociation

La négociation permet de résoudre les conflits qui peuvent altérer la coopération des agents. Elle a donc une influence sur l'indice de coopération se rapportant à la non-persistance des conflits et a ainsi un impact sur la coopération générale d'un système. Par contre, l'acte de négociation ne semble pas constituer un acte de coopération ; il ne répond pas au critère énoncé par J.R. Galliers [55] dans sa définition de la coopération sur le libre choix de l'agent. La négociation peut, en outre, entraîner la non opérationnalité d'un agent (la négociation d'une résiliation) et donc conduire à une forme bien étrange de coopération ! (Figure 1.7)

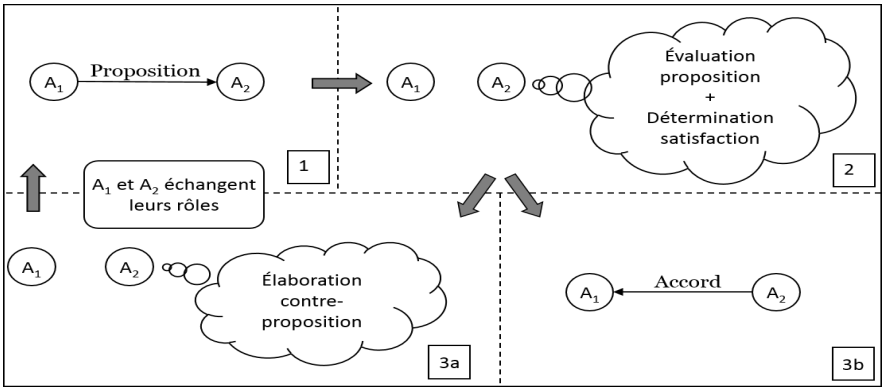


FIGURE 1.7 – Exemple de protocole de négociation entre deux agents [96].

4.3.5 Coordination réactive

Les agents réactifs sont des agents situés qui opèrent dans *'l'ici et le maintenant'*, c'est-à-dire qu'ils ne mémorisent pas les événements passés et ne peuvent pas anticiper sur le futur [68], nécessitent des mécanismes de coordination répondant à leurs besoins en respectant le but global du système. La coordination réactive étant un mécanisme de coordination qui s'intéresse à la coordination des

agents réactifs.

La coordination réactive a débuté en IA vers les années 80 à la suite des travaux fondateurs de Reynolds sur le *flocking* [112] et de Khatib sur la *navigation par champs de potentiels* [75] et de Minsky sur la simulation du comportement du cerveau humain [91]. Elle se présente comme une solution alternative aux approches de planification avec la particularité d'utiliser l'environnement comme élément central de communication. Le comportement des agents dans cette approche dépend seulement de la perception courante de l'agent dans un rayon limité dans son voisinage. Les agents utilisent leur environnement comme mémoire partagée pour stocker ou modifier des informations. Ce paradigme est connu sous le nom *calcul via l'environnement* [125]. C'est une approche inspirée de l'étude des processus de la stigmergie [2] introduit initialement par Grassé [58] sur le comportement de reconstruction collective du nid observé chez les termites. Les chemins de phéromones chez les fourmis représente aussi un exemple classique de la stigmergie [131], où les fourmis déposent de phéromone à leur retour à la fourmilière quand une nourriture est trouvée. Cet phéromone attire et recrute les autres fourmis dans les chemins créés, qui a leurs tours renforcent ces chemins par des dépôts de phéromones, cela résulte en l'apparition d'un réseaux de chemins efficace qui inclu des chemins optimaux vers toutes les localisations de nourriture [64]. Avec le développement du paradigme agent dans la simulation informatique, les insectes sociaux ont été conceptualisés comme des essaims d'agents simples, qui peuvent collectivement réaliser des tâches complexes à travers la stigmergie connu sous le nom intelligence en essaim [40] [15] [72].

Depuis les années 1990, le terme stigmergie a vu une grande diffusion dans différents domaines. Elle a été appliquée non seulement au agents logiciels mais à leurs analogues aussi : les robots autonomes dans les domaines de la vie artificielle, de l'intelligence artificielle et la robotique mobile. Du fait de ses capacités à résoudre des tâches complexes sans aucune planification, aucun contrôle et aucune interaction directe [64], la stigmergie a suscité un grand intérêt en informatique. Dans les années 2000, l'étude du modèle par marquage de phéromones prend différents formes. En particulier, par la généralisation du processus de diffusion/évaporation de phéromones sur des environnements grilles, nommés phéromones digitales [101] [119] [118]. Certains travaux visent à ne

2. La stigmergie est un concept biologique énoncé par Grassé [58] qui réfère à l'utilisation de l'environnement, par un groupe d'agents, comme moyen de communication et de coordination à travers l'utilisation de marqueurs chimiques, les phéromones.

pas limiter les marques à des phéromones digitales, ni nécessairement à exploiter un marquage actif [122], ils traitent plutôt le marquage de l'environnement par des valeurs entières qui permettent la création des champs de potentiels numériques (marquage passif). Les travaux présentés dans ce manuscrit s'attachent à viser ce sorte de techniques de coordination réactive à travers un marquage passif (champs de potentiel) et un marquage dynamique (phéromones digitales) dans le cadre de la résolution collective de problème (recherche des échantillons, couverture et foraging).

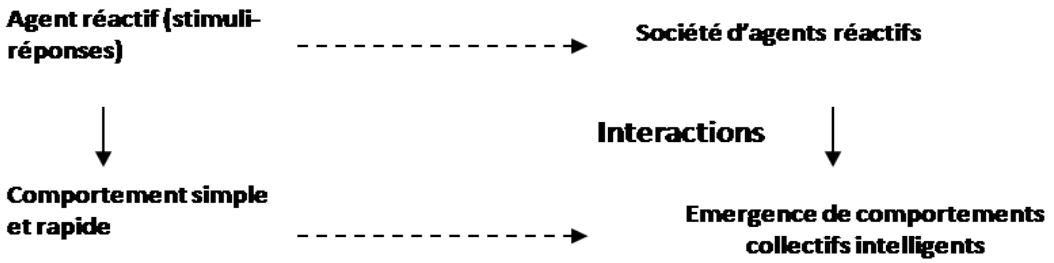


FIGURE 1.8 – Le comportement intelligent d'une société d'agents réactifs résulte de l'interaction de nombreux agents simples [123].

Plusieurs méthodes de coordination réactive existent. Elles diffèrent par leur usage de champs ou de routines (réflexes), par leur usage ou non de représentations limitées des autres et par l'emploi ou non de contraintes et de dépendances. Les méthodes de coordination réactives se résument presque toutes à l'utilisation des techniques suivantes [68] :

– Utilisation des champs de potentiels ou plus généralement de champs de vecteurs pour la détermination du déplacement des agents mobiles. Si le comportement de meute peut s'expliquer par une simple combinaison vectorielle en est-il de même lorsque plusieurs agents mobiles, et alors possible de modéliser le déplacement d'un agent à partir des forces attractives et répulsives résultant de la combinaison de potentiels correspondant à des buts ou à des obstacles. Des exemples de cette approche contient : le comportement de meute [112] et le comportement de bancs de poissons [128]).

– Utilisation de marques pour coordonner l'action de plusieurs agents. Une marque est un signe matériel ou une empreinte qui sert à indicer artificiellement l'environnement afin de coordonner les actions des agents. Les marques peuvent être déposées, lues et retirées par les agents, mais leur effet peut aussi disparaître lentement, par *évaporation naturelle* dans l'environnement. Ces

marques permettent d'exploiter l'environnement comme un système de communication souple, robuste et simple. Les marques servent soit à la synchronisation des actions des agents ou à l'amélioration des performances du groupe. des exemples connus dans ce cadre comporte : la récupération de minerais par des robots explorateurs [39] [129], la résolution de problèmes par création de *chemins d'odeur*, métaphore faisant référence aux fourmis et aux phéromones qu'elles laissent sur leur passage [45].

– Utilisation de l'éco-résolution : c'est une méthode de résolution par coordination qui consiste à reformuler un problème en un ensemble d'agents en interaction -les éco-agents- qui tentent de satisfaire individuellement leur propre but. Chaque éco-agent tente de réaliser son but, et s'il perçoit un gêneur il l'agresse. Un agent agressé tente de fuir en tenant compte des contraintes de son agresseur. Ce schéma se produit et fait évoluer le système jusqu'à ce qu'il aboutisse à un état stable (satisfaction de tous les agents), la solution du problème.

Dans le cadre de ce travail, nous nous intéressons à la coordination d'un ensemble d'agents réactifs. La coordination dans le cadre de coopération (*coordination coopérative*) sera le souci centrale de toute les contributions dans ce manuscrit. Donc nous cherchons à coordonné un ensemble d'agents réactifs dans le but d'améliorer les performances du groupe pour la résolution des problèmes.

5 Incertitude dans les Systèmes multi-agents

Des conditions parfaites sont définies lorsque le système est parfaitement modélisé : les capteurs et les effecteurs des agents soit d'une précision irréprochable et que la portée soit illimitée et que l'agent connaisse exactement les états et les comportements des autres agents et qu'il peut communiquer avec eux à tout moment. Ces conditions ne sont souvent pas respectées dans un environnement réel ce qui conduit à un certain nombre d'incertitudes lors de décision d'un agent (voir robot mobile). Ces incertitudes dans un système multi-agents ou même dans un agent sont dues à certaines raisons qui dépendent des systèmes mis en place et de l'environnement dans lequel évolue les agents :

– Imprécision et limitation des capteurs– la perception à travers des capteurs permet à un agent de maintenir une connaissance de sa position ainsi qu'une connaissance sur les éléments composant

le système. Les capteurs permet donc de détecter l'espace navigable sous la forme d'un nuage de points délimitant ce qui est obstacle de ce qui ne l'est pas avec une certaine précision. L'imprécision des capteurs causée par leurs capacités limitées ou par le bruit entraine des incertitudes sur l'information perçue. En plus, quelque soit le type de capteur, il ne permet généralement pas de percevoir tout l'environnement mais un champs de vision de celui ci. Tout ce qui hors champs de vision du capteur est considéré comme non-observable et donc incertain.

– Imprécision des effecteurs et manque de connaissance sur l'environnement– des incertitudes sur les résultats des actions des agents surviennent à cause de la modélisation incomplète de l'environnement. La modélisation incomplète de l'environnement est due principalement au concepteur pour lequel il été impossible de dégager toutes les cas ou les paramètres intervenant dans l'évolution du système. Cette incomplétude entrainera une perte de précision des données modélisées.

– Interaction entre agents et le non contrôle d'événements extérieurs aux agents– L'agent n'a généralement qu'une observabilité partielle des états des autres agents et leur comportement. Le fait que les agent entrent en interaction avec d'autres agents, leurs actions conduisent donc à des effets non envisagés. Le fait que l'agent n'a pas de connaissance des actions des autres agents, cela ajoute un certain degré d'incertitude sur ses actions.

6 Application des Systèmes Multi-Agents

Les domaines d'application des systèmes multi-agents sont particulièrement riches. Dans la présente section, nous citons seulement les principales directions faisant référence à notre travail. Wooldridge [141], Ferber [51] et Jennings [69] détaillent un certain nombre d'applications. Parmi ces applications, nous pouvons citer la fabrication, le contrôle de processus, les systèmes de télécommunication, le contrôle de la circulation aérienne, le trafic et la gestion du transport, le filtrage et la collecte d'informations, le commerce électronique, la gestion des processus d'affaires, le divertissement et les soins médicaux [69]. Une présentation plus complète des applications susmentionnées fait l'objet de l'article de Jennings [69]. Ferber [51] les classifie en trois grandes catégories d'applications : *la conception de systèmes complexes ouverts ou génie logiciel multi-agent*, *la simulation multi-agent* et *la robotique distribuée*. Nos travaux de recherche font référence aux deux

derniers domaines d'applications (*la simulation multi-agent* et *la robotique distribuée*).

La simulation est utilisée par des domaines tels que la physique, la chimie, la biologie, la géologie et les sciences sociales pour modéliser des phénomènes, des systèmes ou même des entités du monde réel en vue de les comprendre, les expliquées ou les prévoir. Les SMA ajoutent une nouvelle dimension à la modélisation classique du fait qu'ils modélisent les individus, leurs comportements et leurs interactions sous la forme d'entités informatiques autonomes (agents), présentant ainsi l'avantage de briser la barrière des niveaux (macro et micro) dans les modélisations classiques. En effet, des structures cohérentes émergent au niveau macro à partir seulement des interactions locales modélisées au niveau micro [51].

A la différence du domaine précédent, la robotique utilise des agents concrets qui se déplacent dans un monde physique réel. La robotique distribuée qui porte sur l'utilisation de plusieurs robots peut appartenir à l'un des trois sous domaines : *la robotique cellulaire* qui s'intéresse à la construction modulaire de robots, *la production distribuée* qui consiste à prévoir la meilleure manière d'arranger les différentes unités de production, et de définir les programmes de chacune de ces unités de façon qu'elles puissent travailler ensemble et donc coordonnent leur travail, *la robotique mobile* qui fait intervenir plusieurs robots qui doivent coordonner pour se déplacer et résoudre collectivement des problèmes.

Dans ce manuscrit, nous nous intéressons à la coordination d'un ensemble d'agents réactifs, une branche de recherche très active dans la robotique mobile qui s'appelle *la robotique en essaim ou swarm robotics*, toute en s'appuyant sur la simulation pour la validation de nos propositions dans ce cadre.

7 Conclusion

La première partie de ce chapitre s'est intéressée au concept d'agent. Une multitude de définitions de cette notion, les types des agents, ainsi qu'une définition dans le cadre de notre travail ont été présentés. La deuxième partie de ce chapitre s'est focalisée sur les systèmes multi-agents. En plus des définitions, nous avons présenté les concepts d'interaction et d'organisation fondamentaux dans

ces systèmes puisqu'il s'agit du concept qui assure le lien entre les comportements individuels des agents et le comportement global qui en résultent, nous avons terminé cette partie par citer les caractéristiques attendues des systèmes multi-agents dans le cadre de ce travail. La troisième partie s'est concentrée sur les techniques de coordination dans les systèmes multi-agents plus spécifiquement celle abordée dans notre travail : *la coordination réactive*. La quatrième partie s'est attardée sur quelques propriétés des systèmes multi-agents qui sont en relation directe avec le travail abordé dans ce manuscrit à savoir l'incertitude.

Chapitre 2

Contexte applicatif et travaux reliés

Sommaire

1 Introduction

La résolution collective de problème est une branche des systèmes multi-agents qui s'intéresse à la résolution de problème par un ensemble d'individus simple qui se coordonne à travers des interactions locales. Devant la diversité des problèmes existantes, il nous semble primordial de définir clairement les problèmes que nous traiterons dans cette thèse. A cet effet, nous décrivons dans ce chapitre le contexte de recherche du travail présenté et discutons des études connexes.

Nous nous intéressons au développement de colonies de robots autonomes. C'est pourquoi les applications envisagées dans ce manuscrit se rattachent au domaine de la robotique en essaim. Deux domaines qui font la base de nos travaux sont l'intelligence en essaim et la robotique en essaim. Nous illustrons dans la première section de ce chapitre (section 2) les principes et propriétés de base des deux études. Nous nous focalisons par la suite dans la section 3 sur le problème d'exploration multi-agents où nous donnerons quelques définitions et caractéristiques intéressantes du problème avec une présentation des approches existantes. De même dans la section 4, nous focalisons sur le problème de foraging multi-agents où nous définissons le problème, ses caractéristiques de base et les approches proposées pour sa résolution. Nous proposons par la suite une nouvelle taxonomie de foraging à travers laquelle nous présentons une comparaison qualitative de ces travaux dans un tableau de synthèse.

2 Intelligence en essaim et robotique en essaim

L'intelligence en essaim dans les systèmes multi-robots est devenue un domaine de recherche important avec la robotique collective. Les chercheurs s'inspirent des systèmes biologiques pour proposer une variété d'applications robotique industrielles, commerciales et militaires. Afin de combler le fossé entre la théorie et la l'application, un accent fort est requis sur l'implémentation robotique de l'intelligence en essaim. A ce jour, la recherche théorique et les simulations informatiques dans le domaine ont dominé, avec peu de succès des démonstrations des systèmes robotiques intelligents en essaim. La plus part des travaux se trouvant encore dans des laboratoires de recherche en phase de validation. La section 2.1 est dédiée à l'intelligence en essaim et l'auto-organisation. Avec l'introduction de quelques exemples, nous illustrons ce que l'intelligence en essaim est et décrivons ses

principes et caractéristiques de base. Dans la section 2.2, nous introduisons la robotique en essaim. Nous nous concentrons aussi sur les propriétés qui rendent attrayant l'implémentation et le contrôle des systèmes multi-robots et nous discutons de leurs applications potentiels.

2.1 Intelligence en essaim et auto-organisation

L'intelligence en essaim (intelligence collective ou swarm intelligence en anglais) a été introduit pour la première fois par Gerardo Beni et Jing Wang en 1989 [11]. Le mot "essaim" évoque l'image d'un grand nombre de petits insectes où chaque individu effectue une tâche simple, mais dont l'action produit un comportement complexe dans son ensemble [74]. Les essaims sont définis comme des collections de nombreuses individus simples qui interagissent avec les autres individus et l'environnement en utilisant des formes de contrôle décentralisé et auto-organisée pour atteindre leurs objectifs [90]. La combinaison de leurs comportements simples ou microscopiques provoque des actions beaucoup plus complexes et macroscopiques, qui permettent à l'ensemble du système d'obtenir des résultats remarquables dans son ensemble. Par conséquent, l'intelligence en essaim fournit un nouveau cadre de travail pour la conception et l'implémentation de systèmes comprenant de nombreux agents qui sont capables de coopérer afin de résoudre des problèmes très complexes.

Les agents disposent de capacités très limitées. Ils ne peuvent percevoir et agir que sur leur environnement immédiat, ne sont capables de communiquer que de manière indirecte en déposant de l'information dans l'environnement, ne possèdent généralement aucune mémoire et ne réalisent leurs décisions qu'en réagissant à leur perceptions.

Les avantages d'une telle approche sont multiples :

– robustesse : la défaillance d'éléments individuels ne dégrade pas significativement les performances de l'ensemble du système.

– simplicité : le comportement individuel est simple, mais encore il permet de réduire la complexité des individus.

– évolutivité : les mécanismes de contrôle utilisés ne dépendent pas du nombre des agents au sein d'un essaim.

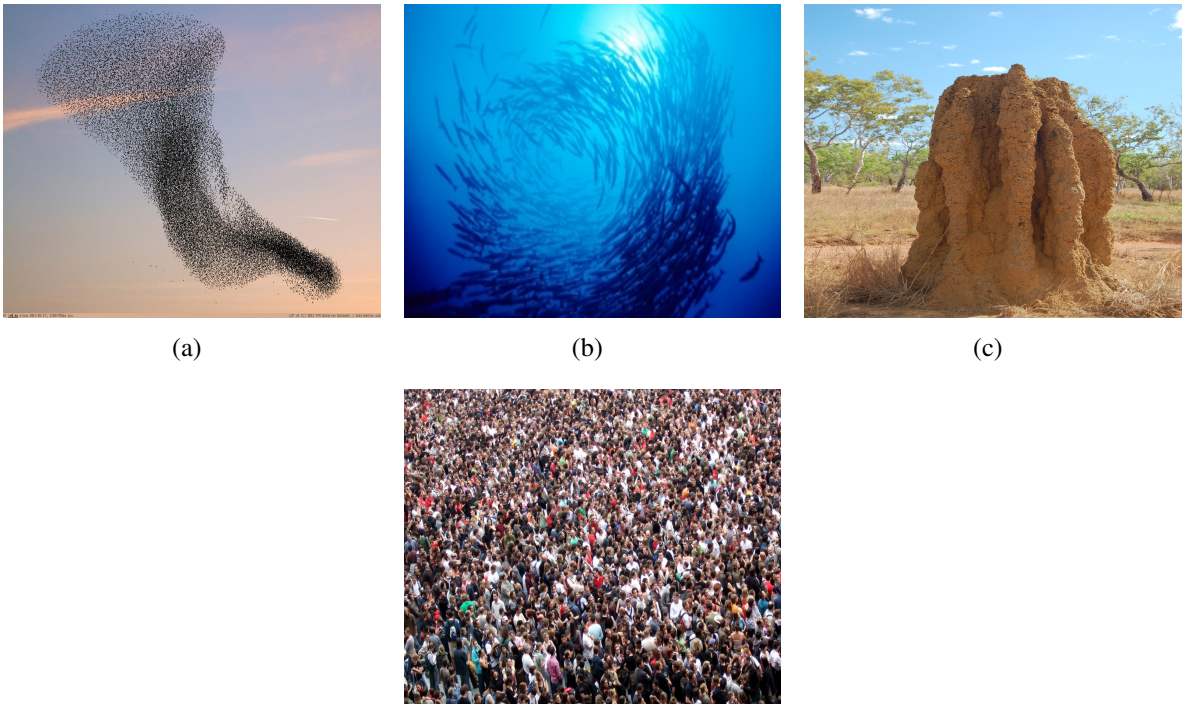
Dans la nature, l'intelligence en essaim peut être observée dans les variétés de systèmes (voir Figure 2.1). Le mouvement coordonné de troupeaux d'oiseaux ou des groupes de poissons sont des exemples emblématiques du domaine. Chaque individus oiseau ou poisson actes de sa façon et surtout n'as aucun contrôle sur les autres, ni est conscient de ce que le groupe dans son ensemble est en train de faire. En résultat, le mouvement du groupe et sa cohésion paraissent comme une seule entité contrôlée par un cerveau unique. Les insectes sociaux, tels que les fourmis, les termites, les abeilles et les guêpes forment les exemples les plus étonnants de l'intelligence en essaim. Les foules peuvent aussi présenter des comportements d'intelligent en essaim. Des exemples peuvent êtres observés dans les mouvements des piétons. Normalement, chaque individus se déplace librement vers sa destination, mais dans des conditions de haute densité, les mouvements sont limités et des modèles émergent souvent [49].

Les actions des individus qui interagissent localement conduit à des comportements collectifs qui semblent être guidé par une volonté unique et contrôlé par une seule entité. Cette forme de comportements collectifs est appelée comportements auto-organisés. Bounabeau [15] définie l'auto-organisation comme l'ensemble de mécanismes dynamiques à partir desquels les structures apparaissent au niveau globale du système via des interactions entre ses composants de niveau locale [1]. Ces interactions sont exécutés sur la base des informations purement locales, sans référence au modèle globale, qui est une propriété émergente du système plutôt qu'une propriété imposée sur le système.

Typiquement, les approches relevant de l'intelligence en essaim s'inspirent principalement de deux disciplines, la physique et la biologie. De la physique s'est inspiré les champs de potentiels qui reposent sur l'utilisation de champs vectoriels. Ainsi chaque point de l'espace est associé à un vecteur [5] [17] [75]. Ce type d'approches s'est utilisé pour des problèmes de navigation [68], de foraging [123] et d'optimisation pour le positionnement d'usines [93]. Un principal problème de cette approche est les maxima et les minima locaux qui peuvent piéger les agents. De la biologie s'est inspiré les approches à base de phéromones, connus aussi sous le nom stigmergie, un mode de communication indirecte entre agents [2]. En effet, les insectes sociaux (termites, fourmis, araignées

1. "set of dynamical mechanisms whereby structures appear at the global level of a system from interactions among its lower-level components" [15]

2. Le terme agent se réfère dans ce contexte à toute entité (physique ou virtuel) qui peut effectuer des actions dans



(a) (b) (c)

(d)

FIGURE 2.1 – Exemples des systèmes d'intelligence en essaim : (a) Troupeaux d'oiseaux (b) Groupes de poissons (c) Nid de termites (d) Foule

sociales...) utilisent l'environnement comme moyen de communication indirecte via le dépôt des phéromones. Ces dernières ont été la source d'inspiration des phéromones digitales, l'équivalent artificiel des phéromones naturels que les agents inscrivent sur une matrice modélisant l'environnement. Ainsi plusieurs algorithmes basés sur le dépôt de phéromones ont été développés : algorithmes de foraging [15], ACO [43] [3], coordination entre drones [103]. Les phéromones digitales font l'avantage d'adaptabilité au changement de l'environnement grâce aux propriétés de diffusion et d'évaporation.

2.2 Robotique en essaim

Avec l'augmentation du nombre de robots, la complexité du système augmente aussi, l'utilisation de contrôle centralisé et de communication globale devient infaisable et il faut recourir à d'autres approches. Une approche proposée été d'implémenter le contrôle du système robotique en s'ins-

un environnement

3. Ant Colony Optimization

pirant des systèmes naturels d'intelligence en essaim. L'application des principes de l'intelligence en essaim à la robotique collective donne naissance à la robotique en essaim (swarm robotics en anglais).

2.2.1 Qu'est ce que la Robotique en essaim ?

La Swarm Robotics selon Dorigo, Birattari et Brambilla [42] est l'étude de la manière de créer des groupes de robots qui s'exécutent sans dépendre d'aucune infrastructure extérieure ou aucune forme de contrôle centralisée. Dans un essaim de robots, le comportement collectif des robots résulte de l'interaction locale entre les robots et l'environnement dans lequel ils interagissent. La création de ces essaims est guidée par les principes de la Swarm Intelligence. Ces principes encouragent la réalisation de systèmes tolérants aux erreurs, extensible et flexible [4].

La robotique en essaim peut être définie comme l'étude de comment un nombre d'agents physique relativement simple peuvent êtres conçus de telle sorte qu'un comportement collectif complexe émerge des interactions locales entre les agents et entre les agents et leur environnement. C'est une approche prometteuse lorsque différentes activités doivent être effectuées simultanément, lorsque la redondance élevée et l'absence d'un point de défaillance unique sont souhaitées, et quand il est techniquement impossible de mettre en place l'infrastructure nécessaire pour contrôler les robots de façon centralisée.

2.2.2 Propriétés souhaitables des systèmes de robotique en essaim

Les caractéristiques précitées de la robotique en essaim sont réputés pour promouvoir la réalisation de systèmes qui sont tolérants aux pannes, évolutive et flexible.

– parallélisme : généralement un problème complexe est subdivisé en plusieurs sous-tâches et chaque agent (soit disant unité) accomplit une tâche donnée plus rapidement qu'un robot en isolation.

4. "Swarm robotics is the study of how to design groups of robots that operate without relying on any external infrastructure or on any form of centralized control. In a robot swarm, the collective behavior of the robots results from local interactions between the robots and between the robots and the environment in which they act. The design of robot swarms is guided by swarm intelligence principles. These principles promote the realization of systems that are fault tolerant, scalable and flexible" [42]

– Robustesse : le système doit assurer une grande tolérance aux fautes. Dans la pratique, si certain robot ne termine pas l'exécution de sa tâche, le système va s'engager en une nouvelle configuration qui ré-établie le bon fonctionnement de système.

– Extensibilité (evolutivité) : l'incrémentation de nombre d'agents ne dégrade pas les performances de l'ensemble du système.

– Hétérogénéité : chaque unité peut être caractérisée avec des propriétés spécifiques qui seront effectivement exploitées, pour accomplir des tâches appropriées ;

– Flexibilité : un système doit être ré-configurable afin d'accomplir les différentes tâches et d'exécuter différentes applications.

– Tâches complexes : généralement, un seul agent n'est pas en mesure d'accomplir une tâche complexe, mais un essaim est capable de le faire, en raison des capacités conjointes des dispositifs simples.

– Coût moins cher : les robots sont simples, faciles à construire et moins cher qu'un seul robot puissant.

Malgré son potentiel de promouvoir la robustesse, évolutivité et la flexibilité, la robotique en essaim doit encore être adoptées pour résoudre les problèmes du monde réel. Divers facteurs limitant le déploiement des systèmes de robotique en essaim dans le monde réel. Des recherches complémentaires sont nécessaires sur le matériel robotique pour remédier aux lacunes matérielles qui limitent la fonctionnalité des systèmes robotiques actuels. A ce jour, aucune application de la robotique en essaim n'as été déployée dans le monde réel, toutes les applications se trouvant dans des laboratoires de recherche. Cependant, avec l'évolution dans le domaine de la robotique autonome et le progrès technologique, les robots envahissent nos vies. Par exemple, les robots nettoyeurs [113] ou encore le système robotique d'entrepôt Kiva [5]. En raison de leurs propriétés les systèmes de robotique en essaim sont destinés aux applications qui nécessitent de la redondance ou l'évolutivité, qui entrainent un danger et se déroulent dans des environnements non-structurés et dynamiques ou dans des grands espaces [16]. Lorsque la technologie évolue, la robotique en essaim sera éventuellement appliquée à des domaines avec ces caractéristiques comme : l'exploration, la surveillance,

5. http ://www.kivasystems.com

le déminage, la recherche et le sauvetage, ainsi que les applications militaires [16].

3 Exploration multi-agents

La recherche des échantillon et le foraging sont des applications du problème d'exploration [83]. La recherche des échantillons est une issue très importante pour l'industrie [147], comme la recherche des échantillons perdus dans des grands environnements inconnus, et la détection de matières radioactives dans certaines zones. En robotique, le foraging se compose de deux étapes complémentaires : la recherche des échantillons (exploration) et leur transport vers un store (homing).

3.1 Définition

le problème de l'exploration est l'un des problèmes fondamentaux de la robotique. Afin de construire le modèle de l'environnement, les robots mobiles ont besoin d'explorer l'espace de travail sous le modèle conçu. L'objectif final d'une tâche d'exploration consiste à détecter l'espace de travail pour un coût minimal qui est généralement représenté par le temps d'exploration. Par conséquent, il est important que les robots tentent d'éviter les zones explorées de façon répétée.

Les chercheurs montrent un grand intérêt pour l'application des compétences des robots isolés à la coopération multi-robots. Cependant, par rapport à l'exploration robotique isolé, l'application de multi-robots mis en place une nouvelle exigence de la coordination et des communications efficaces possibles parmi les membres du groupe.

3.2 Approches existantes

Dans la plupart des travaux d'exploration basés sur la marche aléatoire, l'agent à tendance à retourner à la même cellule plusieurs fois, avant d'aller plus loin, puisque l'agent ne dispose pas d'information historique sur les cellules (régions) déjà visitées. Mais quand le temps et la consommation d'énergie sont déterminants, il serait efficace de guider l'agent vers des cellules non encore visitées et le ré-pulser de celles déjà visitées. Inverse Ant System-Based Surveillance Systems (IAS-SS), est une technique de coordination distribuée et coopérative, appliquée à la surveillance des environnements inconnus [109]. C'est une extension du système classique de la fourmis, où

la phéromone déposée permet de répuler les agents des cellules déjà visitées. La marche aléatoire et le guidage stigmergic ont été combinés pour produire une stratégie d'exploration probabiliste guidée pour les environnements inconnus [132]. Un algorithme basé sur l'intelligence en essaim, pour l'exploration distribuée et le nettoyage [34], divise la carte en différents sous-espaces dont chacun est structuré en grille. Chaque agent décide individuellement, depuis ses informations locales de la cellule suivante à choisir. Une communication directe par WIFI est utilisée entre les robots dans le voisinage. Une synthèse des stratégies d'exploration dans des environnements 2D a été présenté dans [56]. L'algorithme S-MASA [150], est notre algorithme d'exploration, il utilise les phéromones pour guider la recherche, où nous utilisons des agents réactifs sans mémoire. Il peut localiser très rapidement les objets, à un pourcentage proportionnel au nombre des agents. Il fonctionne comme certains animaux, qui recherche leur nourriture autour de leur nid, connu sous le nom *central place foraging theory* [98].

Pour le processus d'exploration, la planification de chemin est éminemment nécessaire pour un robot mobile, puisque par définition, un robot accomplit des tâches en se déplaçant dans le monde réel [76], où le robot est recensé trouver un chemin optimal (ou presque optimal) sans collision (que se soit avec des obstacles ou avec d'autres robots). La planification de chemin est un problème complexe, reliée au problème d'optimisation. Dans des environnements complètement inconnus, une solution optimale est presque impossible à atteindre [70] et des heuristiques ou méta-heuristiques sont utilisés pour améliorer les résultat [104]. Les principales approches de planification sont des extensions de celles proposées dans le cadre mono-robot. Dans [12], une approche basée sur l'algorithme de planification [82] a été proposée, c'est une méthode aléatoire qui redéfinit itérativement l'organisation des robots pour trouver une séquence correspondante au plan minimisant les longueurs de l'ensemble des chemins à parcourir pour l'ensemble des robots. Une approche basée sur l'algorithme de planification de chemin [130]est ainsi proposée [61] où chaque robot planifie son propre chemin indépendamment, puis un diagramme de coordination est construit sur la base des situations de collision possibles. Dans la littérature, certaines méthodes de planification marquantes se regroupent sous les catégories : *la feuille de route probabiliste (Probabilistic Road Map ou PRM)* [71], qui a été largement utilisée pour les bras manipulateurs [115] [116]. Une autre méthode probabiliste est l'exploration rapide par arbres aléatoires (Rapidly-exploring Random Tree

ou RRT) [78] [21]. Le diagramme de voronoi, est employé en robotique dans la résolution des problèmes d'exploration. Wurm et al [143], présentent une stratégie de coordination d'une équipe de robots explorateurs, utilisant le diagramme de voronoi pour segmenter la carte de l'environnement en une squelette permettant de définir des chemins. D'autres travaux à base de la feuille de route probabiliste inclut [23] [24] [144] [146] [82]. Dans le cadre de *la décomposition de l'espace des configurations (cell decomposition)*, deux grandes familles d'approches sont connues sous le nom de décomposition en cellules et de squelettisation. Dans ces méthodes, l'espace initial est divisé en un ensemble de cellules individuelles, puis la relation d'adjacence entre les cellules est calculée. Elles placent les problèmes de planification dans des espaces géométriques discrets. Par association d'une valeur de surface à chaque cellule, la méthode de décomposition en cellules permet un calcul de couverture. L'algorithme ORMSTC (Online Robust Multi-Robot Spanning Tree Coverage) [62], algorithme s'exécutant enligne, basé sur une décomposition approximative en cellules, est l'un des algorithmes les plus connus dans ce cadre. Une approche nommée *champ de potentiel artificiel (Artificial Potential Field ou APF)* est largement utilisée pour le contrôle de formation des systèmes Multi-Robots, lié au problème de planification. khatib [75], été le premier a appliqué les champs de potentiel artificiel à la planification de mouvement d'un robot. Tanner et Kumar [134] ont présenté une stratégie pour des ensembles de robots dans des formations choisies.

Certaines des méthodes présentées posent des problèmes qui leur rendent inefficaces en réalité : APF (minima locaux), Silhoute (complexité inévitable dans les environnements de grande taille). De nouvelles méthodes sont ainsi présentées pour surmonter de telles problèmes tels que : PRM et RRT (pour les espaces de grande taille), des heuristiques et méta-heuristiques telles que : réseau neuronal artificiel [148] [50], les algorithmes génétiques [102] [14], les essaims de particules [73] [105], recuit simulé [26] [89], optimisation de colonie de fourmis [38] [156], la théorie des ondelettes [3] [133], la logique flou [137] [32] et recherche tabu [95] [87] (pour le problème des minima locaux). En réalité, les heuristiques n'atteint pas nécessairement la solution optimale, mais il rapproche une solution candidate plus rapidement que les méthodes déterministes. D'autres alternatives ont été proposés dans le cadre d'évitement des minimas locaux dans les approches basé APF [9] [122] [124]. Les propositions dans cette thèse bénéficient des propositions dans le cadre des APF et les améliorent pour résoudre les problèmes de recherche des échantillons et de foraging

multi-agents.

4 Foraging Multi-Agents

Dans la robotique, le foraging est un problème de référence, particulièrement pour les systèmes multi-robots. C'est un problème très important pour différents raisons : (1) le comportement de foraging sophistiqué observé chez les insectes sociaux, devient récemment plus compréhensible, fournit des inspirations et des modèles de systèmes pour les systèmes artificiels, (2) le foraging est une tâche complexe, qui inclut la coordination de plusieurs - chacune est aussi difficile – tâches, comme l'exploration (recherche) d'un objet ou d'une nourriture, la collection physique des objets qui requièrent certains manipulations physiques, transport des objets, la navigation ou le retour au dépôt, et le dépôt de l'objet dans la base avant de revenir au foraging, et (3) le foraging requière de la coopération entre les agents, soit par communication de l'endroit de la source aux autres agents (chemins de phéromones, ou direction offerte) et/ou le transport coopératif des objets qui sont beaucoup plus difficiles à transporter par un seul agent [139].

À nos jours, il y a un petit nombre de robots fourrageurs employés avec succès dans des applications du monde réel. La plupart des robots fourrageurs sont dans des laboratoires de recherches ou, qui sont conçus pour des applications du monde réel mais ils sont encore dans des étapes d'évaluation. La raison pour cela, c'est que le foraging est une tâche complexe qui requière un nombre de compétences qui doivent être intégrés au sein du robot physique, aussi les principes des robots fourrageurs sont maintenant très difficile. Particulièrement, la perception (capteurs), la gestion d'énergie, la navigation saine dans des environnements inconnus sont tous des problèmes difficiles dans la robotique.

4.1 Définitions

Plusieurs définitions du foarging ont été proposées dans la littérature, nous montrons dans ce qui suit quelques unes.

Definition 4.1 ***Foraging selon Cao et al*** *[25]*

"In foraging, a group of robots must pick up objects scattered in the environment "

Cette définition présente le foraging comme une tâche de collecte d'objets par un ensemble de robots dans un environnement de travail. Il définit trois axes principaux importants dans le problème de foraging qui sont : *le groupe de robots, les objets et l'espace de recherche*. Cette définition reste vague et montre des aspects limités du foraging.

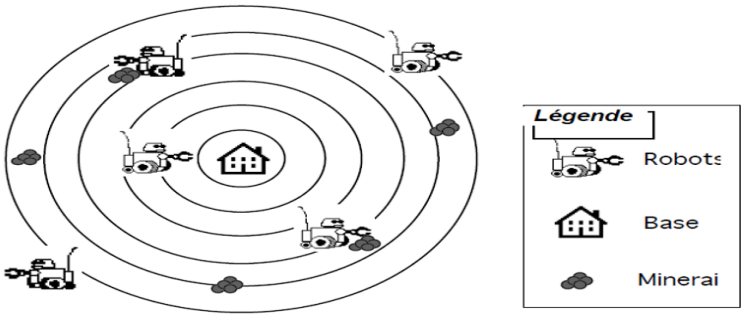


FIGURE 2.2 – Le problème de foraging multi-robots (robots explorateurs) [123].

Definition 4.2 ***Foraging selon Ostergaard et al*** *[100]*

"a two step repetitive process in which (1) robots search a designated region of space for certain objects, and (2) once found these objects are brought to a goal region using some form of navigation"

Cette définition valorise dans le problème de foraging deux tâche : l'exploration jusqu'à localiser les objets et le retour au dépôt, négligeant par contre d'autres aspects aussi important dans le foraging comme par exemple la collecte et le dépôt des objets qui nécessitent des manipulations mécaniques de la part du robot qui peuvent influencer encore la qualité du foraging.

Dans ce qui suit, nous présentons une définition formelle du problème de foraging multi-agents dans une espace de recherche 2D [155].

Definition 4.3 ***Notre définition formelle du problème de foraging multi-agents***

"L'environnement E est représenté par une grille 2D de cellules, de taille NxN. E_{libre} dénote le sous-ensemble de E contenant les cellules sans obstacles et $E_{obstacles}$ dénote le sous-ensemble de E contenant des obstacles. Un obstacle (Obs_i) est un sous-ensemble de $E_{obstacles}$ ($Obs_i \subseteq E_{obstacles}$). Il est défini comme un ensemble fini de cellules obstacles regroupées sous forme rectangulaire ou carré dans des positions fixes de l'environnement. Chaque cellule $c = (x,y) \in E$ a un maximum de

quatre voisins $(x-1,y),(x+1,y),(x,y-1),(x,y+1)$. *Soit* c_0 *la cellule de démarrage pour tout les agents, positionnée dans le centre de l'environnement(coordonnées (0,0)). Soit* E_{visit} *l'ensemble des cellules déjà visitées et* E_{Nvisit} *l'ensemble des cellules non encore visitées, contenant au démarrage toutes les cellules de* E_{libre} *sauf* c_0. *A est l'ensemble d'agents identiques* $a_0...a_n$, *où n est le nombre totale des agents. Chaque* a_i *a une orientation initiale (*0°*=haut,* 90°*=droite,* 180°*=bas or* 270°*=gauche) et peut percevoir les quatre cellules dans les orientations haut, bas, gauche et droite. N locations de nourriture (dénommée dans le reste du manuscrit densité de nourriture), chacune avec M item de nourriture (dénommée dans le reste du manuscrit concentration) sont dispersés sur un ensemble de cellules inclu dans la liste* C_{but} *(liste des cellules contenant de la nourriture). Les agents ne connaissent pas la position de nourriture (cellules dans* C_{but}*). L'objectif est donc de rechercher et transporter toute la nourriture par réduction du temps* $T_{foraging}$ *(i.e. le temps totale nécessaire pour localiser et transporter toute la nourriture dans l'environnement)".*

4.2 Modèle de référence

Winfield [140] définit les robots fourrageurs comme des robots mobiles capables de chercher des objets, et lorsque ils les trouvent, les transportent à un ou plusieurs points de stockages (dépôts) (voir Figure 2.2). Ils peuvent êtres des robots isolés opérants individuellement, ou plusieurs robots opérants collectivement. Les premiers peuvent être contrôlés à distance par des humains, ou semi-autonomes ; les derniers sont beaucoup plus des systèmes complètement autonomes. Il propose un modèle de référence pour un robot fourrageur dont la machine d'état finis est donnée par la Figure 2.3.

Un robot est toujours dans un des quatre états : *Exploration, Extraction, Retour au nid* et *Déposer*. Les quatre états sont définis comme suit :

– *Exploration (recherche)* : dans cet état l'agent se déplace dans l'environnement en utilisant ses capteurs pour localiser et reconnaitre les objets. Cette recherche peut être aléatoire ou en appliquant une certaine méthode de recherche. Lorsqu'un robot trouvera un objet, il changera d'état à *Extraction*, sinon il restera à l'infini dans l'état exploration. L'état rechercher est l'état par défaut à partir duquel tout les agents commencent la foraging.

– *Extraction (collecte)* : dans cet état le robot collecte les objets pour les transporter à la base. Lorsque les objets sont collectés et le robot est prêt pour retourner à la base il change son état à l'état *Retour au nid*.

– *Retour au nid* : dans cet état le robot doit se déplacer avec les objets collectés à la base. Cet état contient un ensemble d'étapes : premièrement, détermination de la direction de la base, deuxièmement, orientation de l'agent à cette position, et troisièmement, navigation à la base. Lorsque l'agent atteint la base, il change son état à l'état *Déposer*.

– *Déposer* : dans cet état le robot dépose les objets dans la base, et changera d'état immédiatement à l'*Exploration*.

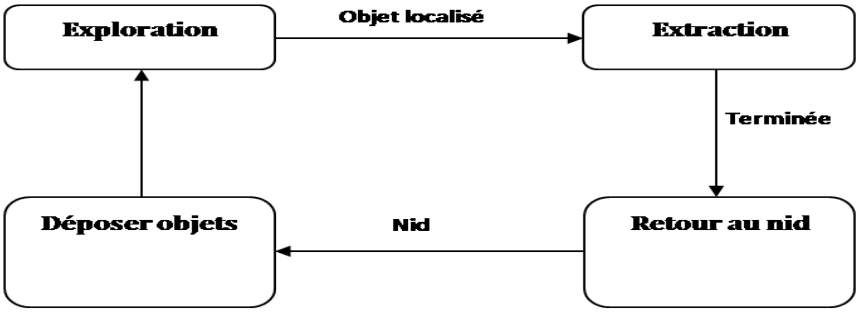


FIGURE 2.3 – Machine d'état finis d'un robot forageur [139].

4.3 Approches existantes

Nous résumons dans ce qui suit les travaux de littérature du foraging pertinents. Nous classifions certains de travaux selon la taxonomie de foraging proposée dans le Tableau 2.1.

Les auteurs dans [66], proposent deux algorithmes de foraging décentralisés inspirés du comportement de fourmis naturels. Les robots disposent de capacités de perception et de calcul limités et ils peuvent communiqués entre eux dans le voisinage par Infra Rouge (IR). Au lieu d'utiliser des phéromones chimique pour communiquer (comme ce fait par les fourmis), les robots mobile se transforment en robots stationnaires (beacons) pour stocker l'information ou bien la phéromone virtuelle. Dans l'algorithme Virtual Pheromone (VP), les phéromones sont représentées par des valeurs flottante et sont stockés dans les robots stationnaires et transmissent par ces derniers aux robots mobiles dans le voisinage, tandis que dans l'algorithme de cardinalité les phéromones sont

des valeurs entières et représentent le nombre des robots qui se trouvent entre le robot stationnaire courant et la position de nourriture ou la position de dépôt. Les auteurs dans [88], proposent deux modèles de foraging multi-agents (Direct Ant Colony Foraging (DACF2) et Direct Ant Colony Foraging (DACF3)) qui adaptent les modèles de Panait [101] et Wilinskey [138] par l'introduction de l'interaction directe (communication directe) en plus de l'interaction indirecte (via la phéromone). Cependant, les agents utilisent peu d'espace pour enregistrer les dernières traces et la valeur de deux variables (le temps écoulé de dépôt ou de la nourriture et la valeur optimal des agents dans le voisinage). Les modèles proposés améliorent largement la possibilité de créer des chemins optimaux. Les auteurs dans [65], proposent trois algorithme de foraging (gradient, sweeper, adaptive). Dans l'algorithme *gradient*, les agents diffusent des informations sur le gradient de nid ou de la nourriture, ils changent des robots stationnaires vers des robots mobiles selon certains critères. Tandis que, dans l'algorithme *sweeper* les agents utilisent des forces virtuelles pour former une ligne dépensant du nid à la nourriture qui déferle l'environnement de recherche. Lorsque la nourriture est localisée, certains robots devient stationnaires tandis que d'autres restent mobiles (workers). L'algorithme *adaptive* permet de basculer entre les deux algorithmes mentionnés ci-dessus selon la position de la nourriture. Lorsque la nourriture est distante de dépôt, les agents utilisent l'algorithme de *gradient*, dans le cas contraire ils basculent vers l'algorithme *sweeper* et lorsque le nombre des agents est insuffisant pour couvrir tout l'espace de recherche ils utilisent la marche aléatoire pour explorer. La préférence des tâches existe dans de nombreuses espèces de fourmis et les auteurs dans [92] prouvent par des simulations que l'introduction de préférences envers certains tâches outre que le stimulus a un certain impact sur les performances de la colonie. Les auteurs utilisent un groupe d'agents hétérogènes et pour chaque groupe ils proposent un ensemble de règles de communications locales avec les agents voisins. Les agents basculent entre trois tâches : *soin de couvains, foraging* et *repos* sur la base de trois seuils correspondant à ces tâches. La mise à jour de ces seuils se fait selon des règles inspirées du comportement de plusieurs espèces de fourmis [59]. Les auteurs dans [107] comparent des systèmes mono-robot avec des systèmes multi-robots, ils montrent que le recrutement d'agents pour l'exploitation d'une nourriture est utile lorsqu'il est difficile de localiser la nourriture. Les agents signalent aux autres la position de nourriture lors de retour au dépôt et ils utilisent l'odométrie pour se localiser. Dans [81] et [80], les auteurs proposent un modèle de foraging inspiré des abeilles dont le but est d'améliorer l'efficacité d'énergie par division de

l'environnement et des tâches. Ils utilisent des dépôts temporaires dans des positions prédéfinies, des agents hétérogènes qui communiquent entre eux la position de nourriture et utilisent du GPS pour retourner à domicile. Deux algorithmes de foraging imitant le comportement d'essaim avec un nombre fini d'agents ont été proposé dans [31]. Dans les deux algorithmes, les agents sont équipés de cameras, capteurs infrarouges et un dispositif de communication radiofréquence. Chaque robot possède un système de positionnement installé afin de déterminer sa position actuelle. La marche aléatoire est adoptée par les deux algorithmes aux premières étapes de l'exploration et les robots transmettent à chaque étape leurs positions actuelles entre eux. Cependant, dans l'algorithme Virtual Pheromone Field (VPF), les agents construisent lors de l'exploration des chemins virtuels de phéromones qu'ils peuvent utilisés pour retourner au dépôt lorsque une nourriture est découverte. Dans l'algorithme Sampling-Graph based foraging (SGF), utilisent (en plus de la marche aléatoire) l'algorithme A^* pour déterminer son chemin (lors de l'exploration et de homing). Dans [122], les auteurs proposent un algorithme de foraging multi-agents. Il représente une version distribuée et asynchrone de l'algorithme classique de la vague [9]. Les agents quand ils explorent construisent simultanément des chemins entre le dépôt et la nourriture ce qui entraîne le calcul d'un front d'onde à partir de la destination de l'agent qui comprend la construction d'un champ de potentiel artificiel ascendant (Artificial Potential Field–APF) de manière incrémentielle. Les agents ont besoin de visiter la même cellule à plusieurs reprises avant que l'APF atteint sa valeur optimale. Les valeurs de l'APF sont utilisées pour retourner à la maison quand une nourriture est découverte.

Nous avons étudié le problème de foraging à travers plusieurs travaux de recherche où nous utilisons des agents avec des capacités de perception et de calcul limités. Dans [53], [149] et [97], nos agents utilisent la marche pseudo-aléatoire pour explorer leur environnement, le marquer avec des valeurs entières pour construire des chemins de retour et coopèrent pour transporter la nourriture découverte par dépôt de phéromone. Dans [152], les agents utilisent notre algorithme d'exploration Stigmergic Multi-Ant Search Area (S-MASA) [150], qui permet une exploration de l'environnement rapide par la large dispersion d'agents et des chemins optimaux. Deux types de phéromones ont été utilisés, une pour marquer le chemin de retour et l'autre pour alerter et attirer les autres agents vers la position de nourriture. Dans [151], nous adoptons l'algorithme précédent par utilisant des phéromones digitales au lieu des valeurs APF. Ce dernier algorithme a été étendue pour

faire face a la limitation des stores d'énergie d'un agent par l'agent d'une suite de comportements permettant aux agents de revenir au dépôt quant leurs énergie courante est inférieur à un seuil prédéfinie [153] [154].

4.4 Nouvelle taxonomie de foraging

Les taxonomies offrent un double avantage pour les chercheurs : (1) résumer et décrire de manière simple les travaux de la littérature, (2) leur offrir des orientations et perspectives lorsqu'ils sont engagés dans des travaux similaires et même les aider à parcourir et comparer leurs travaux avec ceux qui existent déjà. Nous synthétisons des taxonomies de foraging existantes [25] [100] [8] [46] [139] et nous proposons à la base de ces dernières une nouvelle taxonomie de foraging où nous définissons un ensemble de balises descriptives qui identifient les principales caractéristiques de *l'environnement, du collective, de la stratégie et des simulations effectuées*. Suivant un style similaire à celui de la taxonomie proposée dans [139], nous ajoutons d'autres fonctionnalités que nous croyons qu'elles sont pertinentes dans la classification des travaux. La définition de la taxonomie a été détaillée en vue de rapprocher à un modèle qui permet la distinction entre les travaux de littérature sans nécessité de revenir sur la lecture des travaux réels. La taxonomie proposée est résumée par la Figure 2.4.

Elle est représentée hiérarchiquement par trois niveau dont le premier niveau représente les axes principaux de la taxonomie qui sont les composantes du système (*environnement, collective, stratégie* et *simulations*), alors que le deuxième niveau représente les composantes dans chaque axe principal (soit disant axes minimaux) et le troisième niveau explique les axes minimaux par un ensemble de caractéristiques, ces dernières peuvent avoir de différentes valeurs durant l'évolution du système et nous les mettrons entre parenthèses dans la Figure 2.4. Nous donnerons dans ce qui suit une explication générale de la taxonomie, du fait que la plus part des terminologies utilisés sont auto-explicative.

4.5 Comparaison qualitative des approches de Foraging

Nous donnons dans le Tableau 2.1, une comparaison qualitative entre les travaux de foraging présentés dans la section 4.3 selon la taxonomie de foraging présentée dans la section 4.4.

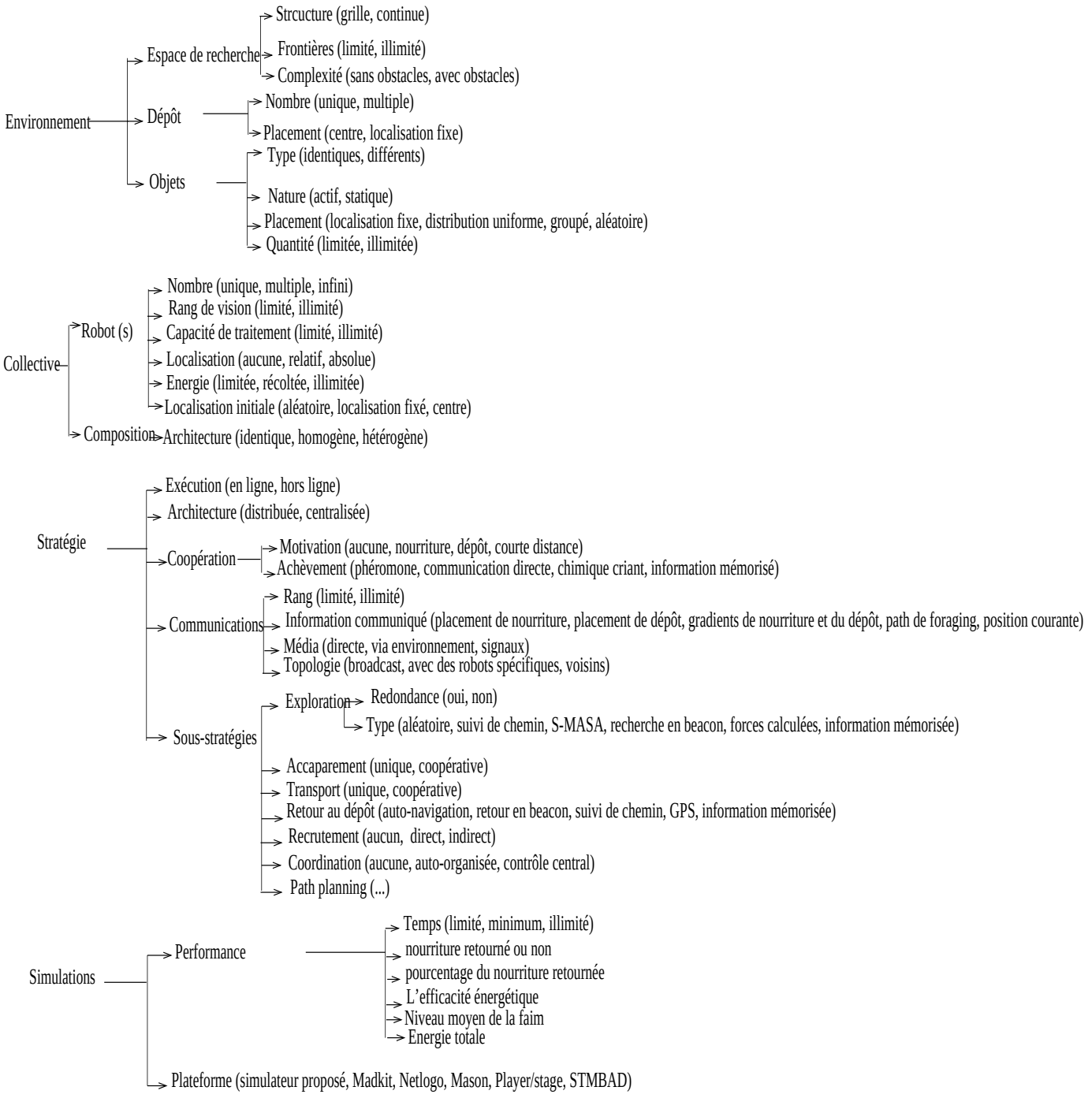


FIGURE 2.4 – La taxonomie de foraging proposée

	axe 2	axe 3	axe 4	[66]	[88]	[65]	[92]	[81]	[31]	[107]	[122]	[152]	[151]	[153]
Environnement	espace de recherche	structure	grille		X		X		X		X	X	X	X
			continue	X		X		X		X				
		Frontières	limité	X	X		X	X	X	X	X		X	X
			illimité			X						X		
		complexité	sans obstacles	X	X	X	X	X	X		X	X	X	X
			avec obstacles		X				X	X	X	X	X	X
	Dépôt	nombre	unique	X	X	X	X	X	X	X	X	X	X	X
			multiple											X
		placement	centre			X		X		X	X	X	X	X
			localisation fixe	X	X		X		X					X
	objets	nombre	unique	X		X		X						
			multiple		X		X		X	X	X	X	X	X
		type	identiques	X	X	X	X	X	X	X	X	X	X	X
			différents											
		quantité	limitée		X			X	X	X	X	X	X	X
			illimitée	X		X	X							
		placement	localisation fixe	X			X		X					
			aléatoire			X		X		X	X	X	X	X
			groupé		X		X			X				
			distribution uniforme							X				
		nature	statique						X			X	X	X
			active					X						
Collective	Robot (s)	nombre	unique											
			multiple	X	X	X	X	X	X	X	X	X	X	X
			infini											
		rang de vision	limité	X	X	X	X	X		X	X	X	X	X
			illimité						X					
		capacité de traitement	limitée	X	X		X	X	X	X	X	X	X	X
			illimité			X								
		Localisation	aucune		X		X			X	X	X	X	X
			relative	X		X		X	X	X				
			absolue											
		Énergie	limitée					X	X	X				X
			récoltée											
			illimitée	X	X	X	X				X	X	X	
		localisation initiale	aléatoire											
			fixe	X	X		X		X					
			centre			X				X	X	X	X	X
		mémoire	limitée		X	X	X	X	X	X	X	X	X	X
			illimitée											
	Composition	architecture	identique		X	X			X	X	X	X	X	X
			homogène											
			hétérogène	X			X	X						
Stratégie	Exécution	exécution	online	X	X	X	X	X	X	X	X	X	X	X
			offline											
	architecture	architecture	distribuée	X	X	X	X	X	X	X	X	X	X	X
			centralisée											
	coopération	motivation	aucune	X		X								
			nourriture		X		X	X	X		X	X	X	X
			dépôt											
			courte distance							X				
		achèvement	phéromone		X		X		X		X	X	X	X
			communication directe	X	X	X		X	X	X				
			chimique criant				X							
			information mémorisé						X	X				
	communication	aucune	aucune											
		information communiqué	position de nourriture				X	X	X	X	X	X	X	X
			position de dépôt											
			gradients de nourriture et du dépôt	X		X								
			chemin de foraging		X									
			position courante						X					
		rang	limité	X	X	X	X		X	X	X	X	X	X
			illimité					X						
		média	directe	X	X	X	X	X	X	X				
			via l'environnent		X		X		X		X	X	X	X
			signaux											
		topologie	broadcast	X		X			X					
			avec des robots spécifiques					X						
			voisins		X		X			X	X	X	X	X
	sous-stratégies	exploration	sans redondance									X	X	X
			redondance	X		X		X	X		X			
			aléatoire	X	X	X	X	X	X	X	X			
			suivi de chemin		X		X		X		X			
			S-MASA									X	X	X
			recherche en beacon	X		X								
			forces calculées	X										
			information mémorisée											
			A∗						X					
		accaparement	unique	X	X	X	X	X	X	X	X	X	X	X
			coopératif											
		transport	unique	X	X	X	X	X	X	X	X	X	X	X
			coopératif											
		retour au dépôt	auto-navigation		X									
			retour en beacon	X		X				X				
			suivi de chemin		X		X		X		X	X	X	X
			GPS					X						
			A∗						X					
			information mémorisée											

		recrutement	none											
			direct	X		X		X		X				
			indirect		X		X		X		X	X	X	X
		coordination	aucune											
			auto-organisée	X	X	X	X		X		X	X	X	X
			contrôle central					X						
Simulations	performance	temps	limité									X	X	
			minimum		X	X					X	X	X	
			illimité											
			nourriture retourné ou non ?			X		X						
			pourcentage du nourriture retournée	X		X			X	X		X	X	X
			niveau moyen de la faim				X							
			l'efficacité énergétique											X
			énergie totale	X					X					X
	plateforme	plateforme	simulateur proposé			X								
			Madkit								X			
			Netlogo				X					X	X	X
			Mason		X									
			Player/stage					X						
			SIMBAD						X					

TABLE 2.1 – Comparaison de certains travaux de foraging selon la taxonomie proposée

5 Conclusion

Dans ce chapitre, nous avons présenté le contexte applicatif de notre travail de recherche, à savoir les domaines de l'intelligence en essaim et la robotique en essaim. Puis nous avons focalisé sur les problèmes d'exploration et de foraging multi-agent. Une première ébauche sur l'exploration, ses caractéristiques et les approches existantes a été faite en premier. Puis une deuxième a été faite sur le foraging. A travers laquelle, nous avons montré que le foraging est une application dans laquelle les notions de coordination et de coopération sont centrales. Nous avons présenté encore une nouvelle taxonomie de foraging qui permet de décharger les chercheurs d'une lecture lente des travaux reliés. Nous avons détaillé ainsi les propriétés du système pour rendre la comparaison plus aisée où nous avons pris en considération plusieurs taxonomies de foraging existantes.

Comme la proposition d'une technique de coordination générique reste encore un défis dans les systèmes multi-agents, nous avons choisi le problème de foraging comme notre champ d'application. Dans le chapitre suivant, nous proposons une série d'algorithmes permettant la coordination de la suite des agents pour résoudre un tel problème.

Chapitre 3

Algorithmes proposés pour les problèmes d'exploration et de foraging multi-agents

Sommaire

1 Introduction

Nous avons vu dans les chapitres précédents les concepts théoriques du domaine qui seront utilisés le long de cette thèse. Notre objectif dans ce travail est de coordonner un ensemble d'agents réactifs qui coopèrent en vue de réaliser un objectif commun. La coordination réactive à base de stigmergie est l'une des techniques les plus connues dans ce cadre qui fournit des solutions adaptatives et robustes. Proposer une technique de coordination générique reste encore un défis, pour

cela nous avons choisi un problème dans lequel la notion d'interaction est fondamentale. Ce chapitre focalise sur les mécanismes de coordination réactive mis en jeu pour la résolution collective des problèmes d'exploration, de couverture et de foraging multi-agents (souvent étudiés en simulation). Le chapitre se trouve partagé en cinq parties correspondantes aux quatre contributions faites dans ce travail de recherche. Nous présentons dans un premier lieu un algorithme de coordination appliqué aux problèmes de navigation multi-robots, nommé Stigmergic Multi-Ant Search Area (S-MASA) [150], à base de phéromones digitales dynamiques. Conçu initialement pour les problèmes de recherche des échantillons et de couverture multi-agents, il montre comment une organisation dynamique peut émerger sous l'effet du processus d'évaporation de phéromones. Nous adaptons par la suite, l'algorithme S-MASA à un algorithme de foraging multi-agents nommé Cooperative-Color Marking Foraging Agents (C-CMFA) [152] utilisant un marquage passif par valeurs entières. C'est une version distribué et synchrone de l'algorithme de la vague [9]. L'algorithme a été comparé à la version asynchrone dans [122]. Par la suite, nous adaptons ce dernier algorithme à un nouveau algorithme reposant non plus sur des marques passives, mais actives (Cooperative Switching Algorithm for Foraging - C-SAF [151] [155]). Une extension de l'algorithme C-SAF pour des agents avec énergie limitée est faite (Energie-aware Cooperative Switching Algorithm for Foraging -EC-SAF) [153] [154]. Une généralisation de ces différents algorithmes à un Framework pour le problème de foraging est ensuite présentée.

2 Contributions majeures de notre recherche

Dans cette section nous présentons nos différentes contributions pour coordonner un ensemble d'agents réactifs coopérant pour la résolution collective de problèmes. Particulièrement le problème de recherche des échantillons, de couverture et de foraging multi-agents. Nous associons l'algorithme S-MASA aux deux premiers problémes dans la section 2.1, les algorithmes C-CMFA et C-SAF pour le problème de foraging avec un marquage passif et un marquage dynamique dans la section 2.2 et la section 2.3 respectivement. Nous étendons l'algorithme C-CMFA pour permettre aux agents de faire face aux limitations d'énergie dans la section 2.4. Un Framework pour le foraging multi-agents est alors généralisé dans la section 2.5.

2.1 Stigmergic Multi-Ant Search Area (S-MASA)

La stigmergie est un mécanisme de coordination indirecte entre les agents. Le principe est que la trace laissée dans l'environnement par l'action initiale stimule une action suivante, par le même agent ou un agent différent. De cette façon, les actions successives ont tendance à se renforcer et ainsi conduisant à l'émergence spontanée d'activité cohérente. Les fourmis sont un exemple des insectes sociaux qui communiquent par stigmergie. Elles communiquent en déposant des phéromones derrière elles, pour que d'autres fourmis puissent suivre la piste jusqu'à la nourriture (voir la colonie), ce qui constitue un système stigmergique.

Dans cette partie, nous proposons un algorithme de coordination d'un ensemble d'agents réactifs coopératifs, à base de stigmergie (S-MASA, voir Algorithme 1). L'algorithme est inspiré du système classique de fourmis et des tourbillons d'eaux. Conçu initialement pour les problèmes de recherche des échantillons [108] [67] et de couverture [54] [117] dans des environnements inconnus.

– *Système de fourmis classique :* Le premier algorithme de colonies de fourmis proposé est appelé système fourmi (Ant system). Proposé par [44] pour résoudre le problème du voyageur de commerce. La communication entre les fourmis se fait au moyen d'une substance chimique volatile nommée phéromones. Les fourmis sentent avec leurs antennes, leur permettant d'identifier aussi bien la direction que l'intensité des odeurs. Ce système d'orientation olfactif est combiné avec des composantes visuelles, capacité à mesurer la distance parcourue. L'utilisation principale des phéromones réside dans la définition et le repérage des pistes olfactives destinées à guider les fourmis vers des sources de nourriture. Un autre type de phéromone est produit quand la fourmi est écrasée ou attaquée connue sous le nom phéromone d'alerte. La concentration élevée de cette dernière provoque une frénésie agressive chez les fourmis à proximité. Les phéromones peuvent aussi être utilisées pour tromper les ennemis, ou même pour influencer le développement des individus. Certaines fourmis émettent des sons, on parle alors de stridulations. Ces sons permettent alors d'attirer d'autres ouvrières pour, par exemple, porter une proie trop lourde pour un individu isolé. Nos agents suivent le comportement agressif des fourmis permettant d'alerter d'autres fourmis du fait que les agents déposent des phérmones dans les cellules pour alerter ou ré-pulser les autres agents de ces cellules déjà visitées pour garantir une large dispersion et une

rapide exploration de l'environnement.

– *Tourbillons d'eaux :* Un tourbillon est, en dynamique des fluides, une région d'un fluide dans laquelle l'écoulement est principalement un mouvement de rotation autour d'un axe, rectiligne ou incurvé. Ce type de mouvement s'appelle écoulement tourbillonnaire. On en observe toutes les échelles, depuis le tourbillon de vidange d'une baignoire jusqu'à celui des atmosphères des planètes, en passant par les sillages observés au voisinage d'un obstacle situé dans un écoulement liquide ou gazeux [10]. Une fois formés, les tourbillons peuvent se déplacer, s'étirer, se tordre et interagir de manière complexe. En imitant le comportement des tourbillon d'eaux, nous proposons une règle simple pour coordonner les mouvements des agents de telle sorte qu'ils tournent autour d'un point central (la base) et autour d'eux même en utilisant les phéromones digitales dynamiques pour marquer les cellules comme déjà visitées.

Les constituants de notre SMA ont été modélisés comme suit :

– *Modèle de l'environnement :* une grille de NxN cellules. Une cellule contient un ou plusieurs agents, un objet (item), une base ou des obstacles.

– *Modèle de l'agent :* des agents réactifs simples, avec des capacités limitées de calcul, sans mémoire, avec un rang de vision limité au quatre cellules voisines (dans les directions 0°, 90°, 180° et 270°) et ne disposent pas de la position des objets à rechercher. Ils utilisent des phérmones digitales dynamiques pour communiquer.

– *Modèle de la phéromone et sa gestion :* les phéromones sont utilisées pour marquer les cellules comme visitées, et de ré-pulser les agents de ces cellules là. Ce sont des phérmones ($\mathbb{P}$) digitales dynamiques qui ne se diffusent pas mais qui s'évaporent avec un coefficient d'évaporation entre le temps *t* et *(t+1)*. Cette évaporation est représentée par l'équation 3.1 [127].

$$\Gamma_i(t+1) = \Gamma_i(t) - p * \Gamma_i(t) \tag{3.1}$$

où : p est le coefficient d'évaporation.

Tous les agents ont une direction (orientation) initiale (0°, 90°, 180° ou 270°). A chaque itération, l'agent teste si la cellule dans sa droite est visitée (contient une phéromone $\mathbb{P}$), il se déplace à la

cellule voisine dans la même direction, sinon il ajuste sa direction selon les règles de coordination dans la Figure 3.1 et l'algorithme 1.

(a) (b) (c) (d)

FIGURE 3.1 – Règles de coordination de S-MASA représentant les changements de directions possibles : (a) De 180° a 270° (b) De 270° a 0° (c) De 0° a 90° (d) De 90° a 180°, ou les croix blanches représente les cellules déjà visitées

l'algorithme S-MASA a été appliqué pour la résolution de problèmes : recherche des échantillons (voir algorithme 1) et couverture multi-agents, avec modification du critère d'arrêt (quand les agents atteignent les frontières) dans des environnements avec et sans obstacles.

Algorithm 1 S-MASA

Entrées : Direction (0°, 90°, 180° et 270°)
Sortie : Nombre des itérations, Direction

1: **while** (nombre des échantillons et les frontières de l'environnement n'ont pas été atteintes) **do**
2: Ajuster_Direction() 2
3: Déposer $\mathbb{P}$()
4: Mettre_A_jour $\mathbb{P}$()
5: Déplacer ()
6: **end while**

Algorithm 2 Ajuster_Direction.

if *(*$\nexists$ $\mathbb{P}$ *à droite)* **then**
  if *(direction* = 270°*)* **then**
    direction := 0°
  else
    direction := direction + 90°

2.2 C-CMFA : Algorithme de foraging Multi-Agents

L'utilisation des phéromones digitales dans la résolution collective des problèmes (foraging par exemple) a subit certains critiques tels que : (1) le temps nécessaire à l'émergence des chemins optimaux est long en comparaison avec les approches de planification classique, (2) Les minima locaux, générées du dépôt et de la diffusion des phéromones et (3) le coût computationnel considérable de la diffusion/évaporation d'une phéromone. Des problèmes multi-agents tels que les problèmes de navigation robotique, peuvent êtres traités par construction de champs de potentiels artificiels (Artificial Potential Field ou APF). Nous adaptons l'algorithme S-MASA pour le problème de foraging avec marquage passif artificiel, nous ré-examinons donc dans cette partie, la construction d'un champ de potentiel de façon distribuée et synchrone par le processus de marquage passif d'un ensemble d'agents réactifs partant de l'algorithme classique de la vague [9]. Ce dernier a été réexaminé pour une première fois par Simonin [122] afin de l'exploiter pour des tâches de *navigation* et de *foraging* en environnement inconnu. Des propriétés comme la construction, la validité, la convergence et l'exploitation du champ de potentiel pour la résolution du problème de *foraging* sont alors abordées dans cette partie. Dans ce qui suit, nous présentons l'algorithme de la vague et les deux révisions faites sur ce dernier [124] [152]

2.2.1 Algorithme de la vague

C'est un algorithme classique de planification de chemin qui fournit le plus court chemin vers une cellule but depuis n'importe quelle cellule dans un environnement discret (grille) [9]. Il calcule un champ de potentiel artificiel optimal où les minima locaux sont évités par l'expansion de front de vague calculé depuis la position de démarrage de l'agent, qui consiste à construire un APF ascendant incrémentiel. Il s'agit d'une adaptation de l'algorithme du parcours Breath First Search (algorithme de parcours en largeur) [1] d'un graphe au cas d'un environnement grille. Le principe est de construire le champ depuis la cellule but, de valeur 0, en incrémentant la distance à chaque itération. Le chemin ainsi crée est le plus court chemin vers le but on suivant la descente du gradient. La descente de gradient consiste simplement à suivre la direction indiquée par la valeur minimale

1. permet le parcours d'un graphe on commence par explorer un noeud source, puis ses successeurs, puis les successeurs non explorés des successeurs, etc. Il permet de calculer les distances de tous les nœuds depuis un nœud source dans un graphe non pondéré

dans les quatre cellules voisines, et à avancer dans cette direction d'un pas de longueur 1 (passage d'une cellule à l'autre).

2.2.2 Ré-examen de l'algorithme de la vague

L'algorithme de la vague a été ré-examiné par Simonin et al [124] à travers le comportement d'un ensemble d'agents réactifs pour son utilisation dans des problèmes de navigation robotique comme le *foraging*. Ces agents ont la capacité de marquer leurs environnement par des valeurs numérique (des entiers positifs) pour la construction du APF, et de perception limitée à un voisinage de 1 selon la *distance de Manhattan*. Ils font référence à des techniques populaires basées sur un champ potentiel artificiel [6] [75]. Toutefois, il est bien connu que le champ de potentiel peut contenir des minima locaux où les robots peuvent être piégés [157]. La construction des APF proposée par Barraquand et al. [9] est prometteuse à cet égard, car elle empêche la formation des minima locaux. L'environnement est passif et donc toute modification des valeurs APF est faite par le marquage des agents. Les agents inscrivent dans l'environnement (leur mémoire partagée), de façon distribuée et asynchrone des parties de la vague, qui avec le temps converge vers un champ complet optimal. La marche pseudo-aléatoire par la préférence des cellules non encore visitées engendre l'inscription des valeurs pas nécessairement optimales, et plusieurs visites peuvent êtres nécessaires pour la convergence de ces valeurs aux valeurs optimales. *c-marking agents* pour "color marking agents" [122], sont des agents fourrageurs qui ont la capacité de marquer leur environnement par des valeurs entières et de colorer le chemin de retour à la base (par couleur spécifique), par suivi de la descente du gradient crée lors de la phase d'exploration. L'extension de l'algorithme de la vague, et son application au problème de foraging a permis l'établissement d'un modèle multi-agents plus efficace que le modèle de la fourmis classique (d'après les résultats de comparaison [122]).

L'algorithme *c-marking agents*, présente certains inconvénients : (1) La convergence du champ à sa valeur optimal prend beaucoup de temps et les cellules doivent êtres visitées plusieurs fois avant d'atteindre la valeur optimale, (2) la marche pseudo-aléatoire conduit à des chemins de grande longueur, ce qui augmente significativement le temps de foraging.

2.2.3 Ré-examen de l'algorithme de la vague et de l'algorithme c-marking

L'algorithme c-marking agents est une version distribuée et asynchrone de l'algorithme de la vague. La marche aléatoire étant la raison principale de la lente convergence du champ de potentiel. L'objectif principale dans cette partie est de revoir l'algorithme c-marking [122] et celui de la vague [9] pour réécrire une version distribuée et synchrone de ce dernier et de l'utiliser pour la résolution du problème de foraging. Une façon de le faire sera de proposer ou d'utiliser une stratégie d'exploration qui couvre les problèmes induits par la marche aléatoire. La réécriture de l'algorithme de la vague se limite donc à un comportement réactif de deux étapes : (1) la mise à jour de la valeur APF dans la cellule courante par l'opération dans l'équation 3.2 [122], et (2) se déplacer à une cellule voisine selon les règles de coordination de notre S-MASA algorithme 3, dont la valeur initiale vaut $+\infty$, sauf celle de départ qui vaut 0.

$$val = min(val, 1 + min(valeursAPFdes4 - voisins)) \tag{3.2}$$

Algorithm 3 Règle de coordination de S-MASA sur des valeurs APF

Entrées : Valeur APF
Sortie : Direction

```
1: if (cellule de droite marquée) then
2:     direction := direction
3: else if (direction = 270) then
4:     direction := 0
5: else
6:     direction := direction + 90
7: end if
```

Avec la règle de coordination dans l'algorithme 3, les agents inscrivent de façon synchrone et simultanément quand ils explorent, un champ de potentiel complet et optimal. Toutefois, une valeur inscrite dans une cellule est la valeur optimale (minimale) depuis la première visite et les agents n'ont pas besoin de revisitées les cellules.

2.2.4 Algorithme C-CMFA

La règle de coordination dans l'algorithme 3, a été étendue pour être utilisée avec le problème de foraging. Un agent fourrageur peut être *explorateur* ou *transporteur*. Il est *transporteur*, s'il porte une nourriture, sinon, il est *explorateur*. Nos agents suivent une architecture de subsumption [17] à quatre couches, où chaque couche implémente un comportement particulier : *Exploration de l'environnement* est la couche de priorité la plus basse qui consiste à explorer l'environnement. Elle est représentée par l'état composite *Exploration de l'environnement* dans la Figure 3.3. L'agent dans cet état est qualifié d'explorateur, il se déplace dans son environnement dans les quatre directions (haut, bas, gauche et droite), selon les règles de coordination de l'algorithme 3. *Exploitation de nourritures*, consiste en l'exploitation de nourritures quand elles sont localisées, elle enveloppe les états simple composant l'état composite *Exploitation de nourritures* représenté par la Figure 3.5. L'agent dans cet état est un transporteur, il transporte une quantité de nourriture limitée et retourne à la base pour la stockée, il propage l'information à ces voisins et colore le chemin de retour pour garder la trace de la position de nourriture (position à partir de laquelle il résume son exploration après la terminaison de l'exploitation). *Évitement d'obstacles* est la couche la plus prioritaire, elle implémente le comportement d'évitement d'obstacles. Les couches les plus prioritaires sont capables de subsumer les niveaux inférieurs afin de créer un comportement viable (voir Figure 3.2 pour une illustration de l'architecture de subsumption). Un agent fourrageur est toujours dans l'un des deux états composites : *Exploration de l'environnement* et *Exploitation de nourritures*. Chacun des deux états englobe un ensemble d'états élémentaires. Les agents basculent entre les états en fonction des événements entourant. Les diagrammes d'états-transitions dans les Figure 3.4 et la Figure 3.5, représentent les deux états composites respectivement. Le comportement des agents qui décrit le passage entre les deux états composites *Exploration de l'environnement* et *Exploitation de nourritures*, est représenté dans la Figure 3.3.

Dans l'état *Exploration de l'environnement*, l'agent explore son environnement tout en recherchant la nourriture. Il contient les deux états élémentaires :

– *Rechercher-Nourriture :* L'agent vérifie si la cellule courante est une nourriture, si oui il change d'état à *Prendre-Nourriture* dans l'état composite *Exploitation de nourriture* (voir Figure 3.4), sinon il change d'état à *Choisir-Prochaine-Cellule*.

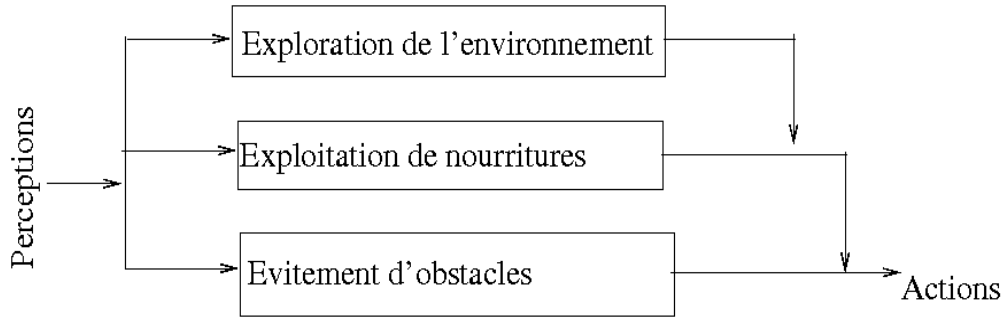


FIGURE 3.2 – Architecture de subsumption utilisée par nos agents fourrageurs

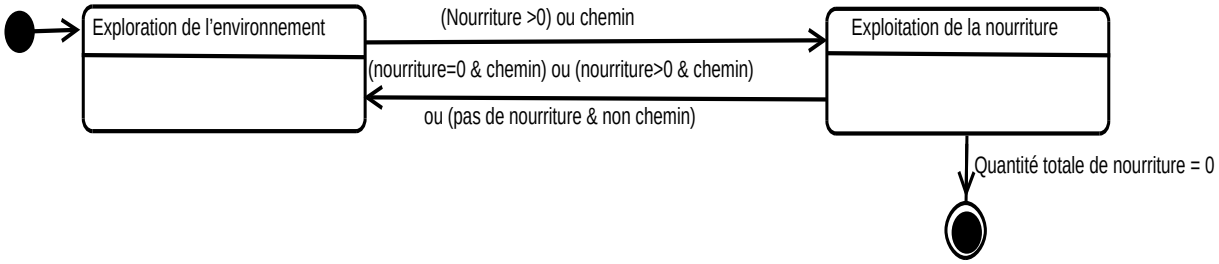


FIGURE 3.3 – Diagramme d'état-transition représentant le comportement de nos agents fourrageurs

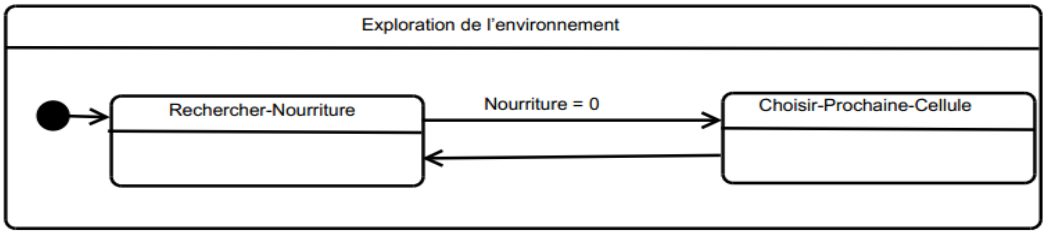


FIGURE 3.4 – Diagramme d'état-transition représentant l'état composite *Exploration de l'environnement*

– *Choisir-Prochaine-Cellule :* Si l'agent rencontre une phéromone de recrutement ($\mathbb{PT}$), il propage l'information à sa cellule gauche, pour recruter autant que possible d'agents pour une exploitation collective de cette source de nourriture et suivre le chemin de recrutement jusqu'à la nourriture, sinon il mit à jour la valeur APF de la cellule courante selon la règle dans l'équation 3.2, ajuste sa direction selon la règle de coordination dans l'algorithme 3 et se déplace à la cellule résultante. Il change automatiquement à l'état *Rechercher-Nourriture*.

Dans l'état composite *Exploitation de nourritures* (voir Figure 3.5), les agents exploitent les nourritures localisées, ils les transportent graduellement et collectivement vers la base, et simultanément créant des chemins reliant la base et la nourriture. Les agents passent par les différents états expli-

Algorithm 4 C-CMFA [152]

if *A-Base* **then**
  Déposer nourriture
  if *(∃ chemin) et (nourriture > 0)* **then goto** *MONTER*
  else if *(∃ chemin) et (nourriture = 0)* **then goto** *Effacer-Chemin*
  else goto *Rechercher-Nourriture*

if *Rechercher-Nourriture* **then**
  if *(nourriture > 0)* **then goto** *Prendre-Nourriture*
  else goto *Choisir-Prochaine-Cellule*

if *Choisir-Prochaine-Cellule* **then**
  if *(∃ obstacle)* **then** Éviter_Obstacle()
  else
    Mettre_A_Jour_APF()(équation 3.2)
    if *(∃* $\mathbb{PT}$ *ici) et (∃* $\mathbb{PT}$ *à droite))* **then** Dépose $\mathbb{PT}$ à gauche
    Monter le chemin de recrutement au nourriture par suivi des $\mathbb{PT}$
    else if *(∃* $\mathbb{PT}$ *ici) and (∄* $\mathbb{PT}$ *à droite)* **then** Effacer le chemin de recrutement
    else Ajustement_Direction ()(Algorithme 3) ; Mettre_A_Jour_APF() 3.2 ; Se_Déplacer()

if *Prendre-Nourriture* **then**
  Prendre une quantité donnée de nourriture
  Dépose $\mathbb{PT}$ à gauche
  if *(∃ chemin)* **then goto** *Revenir-Base*
  else goto *Revenir-et-Colorer*

if *Effacer-Chemin* **then**
  repeat
    Déplacer à un des quatre voisins colorées avec valeur APF > à celle courante
    Modifier sa couleur à la couleur par défaut(noir)
  until *∄ chemin*;
  goto *Rechercher-Nourriture*

if *Monter* **then**
  repeat
    Déplacer à un des quatre voisins colorées avec valeur APF > à celle courante
  until *∄ chemin*;
  goto *Rechercher-Nourriture*

if *Revenir-Base* **then**
  repeat
    Déplacer à un des quatre voisins colorées avec valeur APF < à celle courante
  until *Base atteinte*;
  goto *A-Base*

if *Revenir-et-Colorer* **then**
  repeat
    Déplacer à un des quatre voisins avec la valeur APF minimale
    Colorer la cellule à une couleur spécifique (chemin de retour, déposer une $\mathbb{PR}$)
  until *Base atteinte*;
  goto *A-Base*

qués ci-dessous :

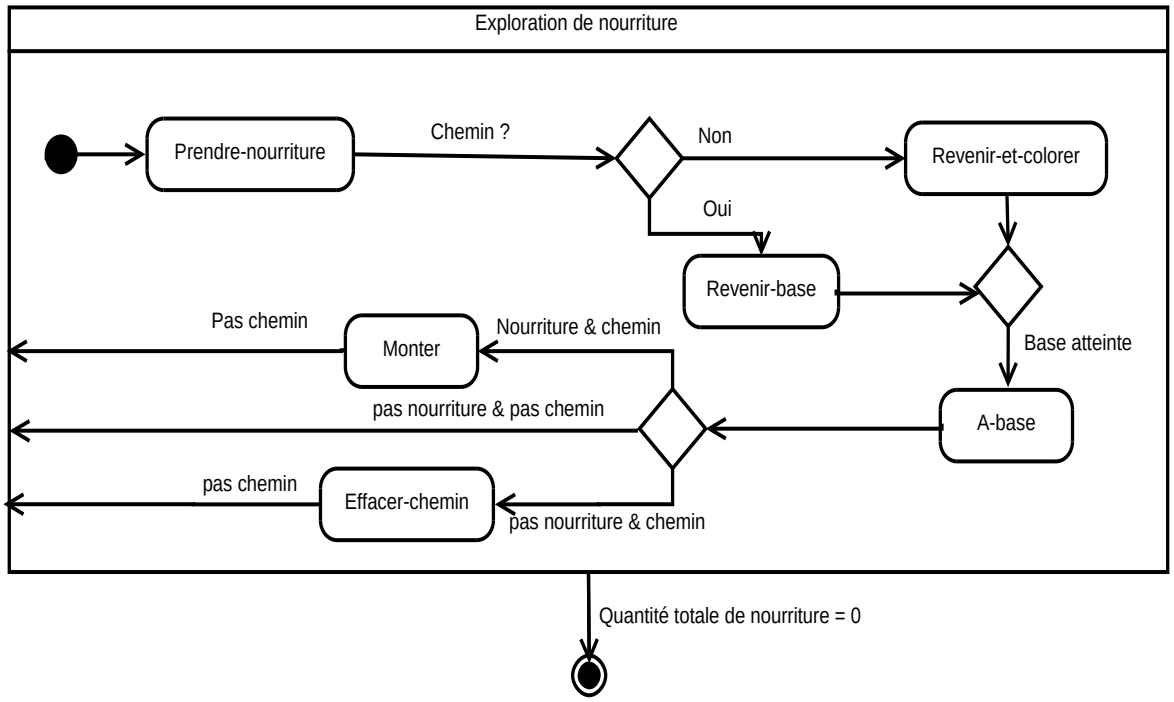


FIGURE 3.5 – Diagramme d'état-transition représentant l'état composite *Exploitation de nourritures*

– *Prendre-Nourriture :* l'agent ramasse une quantité limitée de nourriture et propage l'information à son voisin gauche (dépôt $\mathbb{PT}$). S'il existe un chemin de retour, il change d'état vers *Revenir-Base*, sinon il change d'état vers *Revenir-et-Colorer*.

– *Revenir-Base :* l'agent se déplace à l'une des quatre cellules voisines colorées avec une valeur APF minimale à celle de la cellule courante. Il reste dans cet état jusqu'à ce que la base est atteinte, il change alors à l'état *A-Base*.

– *Revenir-et-Colorer :* l'agent se déplace à l'une des quatre cellules voisines avec la valeur APF minimale et change sa couleur à une couleur de chemin spécifique (chemin de retour, autrement dit il dépose une $\mathbb{PR}$ ou phéromone de retour). Il reste dans cet état jusqu'à ce que la base est atteinte, il change alors à l'état *A-Base*.

– *A-Base :* l'agent dépose la nourriture à la base. S'il existe encore de nourriture et il existe un chemin l'agent exécute l'état *Monter*, sinon si la nourriture est épuisée et il existe un chemin l'agent exécute l'état *Effacer-Chemin*, sinon il change d'état à *Choisir-Prochaine-Cellule* pour

continuer sa recherche.

– *Monter :* l'agent se déplace à l'une des quatre cellules voisines colorées avec une valeur APF supérieur à celle de la cellule courante. Il reste dans cet état jusqu'à ce qu'aucun chemin n'est détecté, il change alors à l'état *Rechercher-Nourriture*.

– *Effacer-Chemin :* l'agent se déplace à l'une des quatre cellules voisines colorées avec une valeur APF supérieur à celle de la cellule courante et change sa couleur à la couleur par défaut (noir). Il reste dans cet état jusqu'à ce que aucun chemin n'est détecté, il change alors à l'état *Rechercher-Nourriture*.

2.3 Algorithme C-SAF

Le marquage numérique et moins adaptatif que le marquage volatile par phéromone digitale [125]. Comme prolongement de l'algorithme précédent (C-CMFA), nous examinons maintenant des algorithmes reposant sur des marques actives (phéromones digitales). La stratégie d'exploration utilisée (algorithme S-MASA) et le processus d'évaporation d'une phéromone digitale montrent comment une organisation dynamique peut émerger et traiter efficacement le problème de foraging. Ils montrent la création d'un gradient de concentration des phéromones, similaire à celui crée par des valeurs APF de l'algorithme C-CMFA. Nous proposons ainsi une extension de l'algorithme précédent qui adapte un marquage passif en un algorithme qui adapte un marquage dynamique (algorithme C-SAF [151] [155]). C'est un algorithme distribué de foraging multi-agents qui forme une adaptation de l'algorithme C-CMFA par utilisation des phéromones digitales au lieu des valeurs APF pour la coordination des agents et pour la création d'un champ optimal. Les agents créent simultanément et de façon synchrone une vague de front par la concentration des phéromones entre la nourriture et la base. Il augmente la coopération entre les agents pour transporter la nourriture localisée par diffusion de l'information aux voisins quand une nourriture est localisée.

Les composantes du système sont modélisées comme suit :

– *Environnement :* une grille de taille NxN cellules. Une cellule contient un ou plusieurs agents, une nourriture (item ou objet), une base ou des obstacles.

– *Agent :* des agents réactifs simples, avec des capacités limités de calcul, mémoire très petite,

perception limitée aux quatre cellules voisines (0°, 90°, 180° et 270°) et ne disposent pas de la position de la nourriture à rechercher. Ils utilisent des phérmones digitales dynamiques pour communiquer.

– *Phéromone et sa gestion :* Trois types de phérmones sont utilisées. La première est utilisée pour marquer les cellules comme visitées, et pour les ré-pulser. Ce sont des Phérmones ($\mathbb{P}$) digitales dynamiques qui ne se diffusent pas mais qui s'évaporent avec le temps. Cette évaporation est représentée par l'équation 3.1. Les deux autres phéromones sont de type passif (n'ont pas de propriétés de diffusion ni d'évaporation), qui sont les Phérmones de recrutemenT ($\mathbb{PT}$) et les Phérmones de Retour $\mathbb{PR}$. On utilise des couleurs différentes pour différencier les deux types.

Les agents suivent la même architecture de subsumption représentée par la Figure 3.2 et le comportement des agents reste le même que celui de l'algorithme C-CMFA dans les Figures 3.3, 3.4, 3.5, sauf que les agents utilisent des phéromones digitales au lieu des valeurs APF. La séquence des états avec une définition différente sont expliquées comme suit :

– *Choisir-Prochaine-Cellule :* Si l'agent rencontre une phéromone de recrutement ($\mathbb{PT}$), il propage l'information à sa gauche, pour recruter autant que possible d'agents pour une exploitation collective de cette source de nourriture et suivre le chemin de recrutement jusqu'à la nourriture, sinon il dépose une phéromone ($\mathbb{P}$), ajuste sa direction selon la règle de coordination dans l'algorithme 3 et se déplace à la cellule résultante. Il change automatiquement à l'état *Rechercher-Nourriture*.

– *Revenir-Base :* l'agent se déplace à l'une des quatre cellules voisines colorées avec la concentration $\mathbb{P}$ inférieur à celle de la cellule courante. Il reste dans cet état jusqu'à ce que la base est atteinte, il change alors à l'état *A-Base*.

– *Revenir-et-Colorer :* l'agent se déplace à l'une des quatre cellules voisines avec la concentration $\mathbb{P}$ inférieur et change sa couleur à une couleur de chemin de retour (dépose $\mathbb{PR}$). Il reste dans cet état jusqu'à ce que la base est atteinte, il change alors à l'état *A-Base*.

– *Monter :* l'agent se déplace à l'une des quatre cellules voisines colorées dont la concentration de $\mathbb{P}$ est supérieur à celle de la cellule courante. Il reste dans cet état jusqu'à ce que aucun chemin n'est détecté, il change alors à l'état *Rechercher-Nourriture*.

Algorithm 5 C-SAF [151] [155]

if *A-Base* **then**
  Déposer nourriture
  if *(∃ chemin* **et** *nourriture > 0)* **then goto** *Monter* **else if** *(∃ chemin* **et** *nourriture = 0)* **then goto** *Effacer-Chemin* **else goto** *Rechercher-Nourriture*

if *Rechercher-Nourriture* **then**
  if *(nourriture > 0)* **then goto** *Prendre-Nourriture* **else goto** *Choisir-Prochaine-Cellule*

if *Choisir-Prochaine-Cellule* **then**
  if *(∃ obstacle)* **then** Éviter_Obstacle() **else**
    if *(∃* $\mathbb{PT}$ *ici)* **et** *(∃* $\mathbb{PT}$ *à droite)* **then**
      Diffuser($\mathbb{PT}$)
      Monter le chemin de recrutement au nourriture par suivi des $\mathbb{PT}$
    else
      if *(∃* $\mathbb{PT}$ *ici) et (∄* $\mathbb{PT}$ *à droite)* **then**
        Effacer le chemin de recrutement
      else
        Déposer($\mathbb{P}$) Ajustement_Direction ($\mathbb{P}$)(Algorithme 2)
        Évaporation($\mathbb{P}$)(équation 3.1)
        Avancer()

if *Prendre-Nourriture* **then**
  Prendre une quantité donnée de nourriture
  Diffuser($\mathbb{P}$) **if** *(∃ chemin)* **then goto** *Revenir-Base* **else goto** *Revenir-et-Colorer*

if *Effacer-Chemin* **then**
  while *∃ chemin* **do**
    Déplacer à un des quatre voisins colorées avec concentration $\mathbb{P} >$ à celle courante
    Modifier sa couleur à la couleur par défaut(noir)
  goto *Rechercher-Nourriture*

if *Monter* **then**
  while *∃ chemin* **do**
    Déplacer à un des quatre voisins colorées avec concentration $\mathbb{P} >$ à celle courante
  goto *Rechercher-Nourriture*

if *Revenir-Base* **then**
  repeat
    Déplacer à un des quatre voisins colorées avec concentration $\mathbb{P} <$ à celle courante
  until *base atteinte*;
  goto *A-Base*

if *Revenir-et-Colorer* **then**
  repeat
    Déplacer à un des quatre voisins colorées avec la concentration minimale de $\mathbb{P}$
    Colorer la cellule à une couleur spécifique(jaune)
  until *base atteinte*;
  goto *A-Base*

– *Effacer-Chemin :* l'agent se déplace à l'une des quatre cellules voisines colorées dont la concentration de $\mathbb{P}$ est supérieur à celle de la cellule courante et change sa couleur vers la couleur par défaut (noir). Il reste dans cet état jusqu'à ce que aucun chemin n'est détecté, il change alors à l'état *Rechercher-Nourriture*.
– *Rechercher-Nourriture*, *Prendre-Nourriture*, *A-Base* : ont la même définition que les états dans l'algorithme C-CMFA (section 2.2.4).

2.4 EC-SAF : algorithme de foraging multi-agents avec gestion d'énergie

Cet algorithm fait une extension de l'algorithme C-SAF. Il rajoute d'autres comportements permettant à l'agent de revenir au dépôt pour recharger son énergie quand elle tombe au dessus d'un seuil prédéfinie ([154]). Nous adressons et analysons alors deux questions : (1) quelle est l'impact de l'exploitation collective de nourritures (par recrutement) sur la consommation d'énergie ? (2) est ce que la division de l'environnement de recherche par l'utilisation de plusieurs bases diminue la consommation d'énergie ?

Les agents utilisent l'architecture de subsumption dans la Figure 3.6. Cette dernière diffère de celle utilisée auparavant par l'ajout de la couche *rechargement d'énergie*, permet à un agent de retourner à la base pour recharger quand son énergie courante (E_c) tombe en dessous d'un seuil (énergie minimal ou E_{min}). Elle inclut les états : *Revenir-Base*, *Revenir-et-Colorer*, *Effacer-Chemin* et *A-Base*. Les couches *exploration de l'environnement* et *exploitation de nourritures*, incluent respectivement les états : *Choisir-Prochaine-Cellule*, *Rechercher-Nourriture* et *Prendre-Nourriture*, *Revenir-Base*, *Revenir-et-Colorer*, *A-Base*, *Monter*, *Effacer-Chemin*. La couche avec la priorité supérieur est celle d'*évitement d'obstacle*. Elle permet de trouver un chemin libre d'obstacles lors de l'exploration ou lors du retour à la base. Le comportement des agents EC-SAF étend celui des agents C-SAF pour faire face à la limitation d'énergie. On ajoute donc un ensemble de transitions permettant à l'agent le retour à la base quand il y a une chute dans son niveau énergétique, on utilise les mêmes états.

Le comportement de nos agents est représenté par le diagramme d'état-transitions dans la Figure 3.7, l'algorithme EC-SAF est donné par l'algorithme 6, et les états sont expliqués dans ce qui suit :

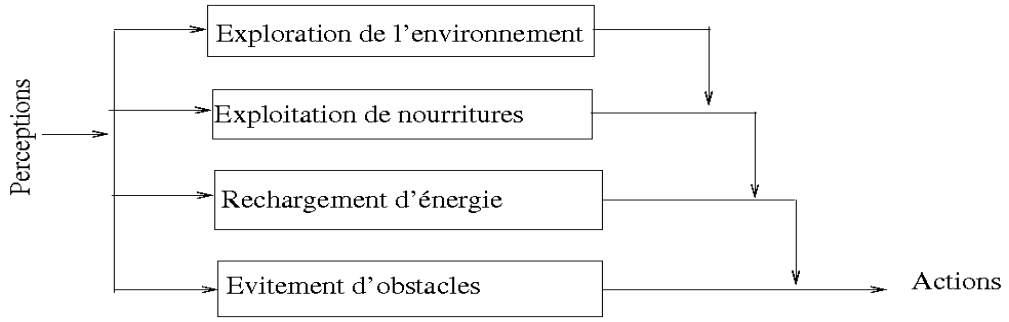


FIGURE 3.6 – Architecture de subsumption utilisée par nos agents EC-SAF

– *Rechercher-Nourriture :* si $E_c > E_{min}$ et une nourriture est localisée ici, l'agent exécute l'état *Prendre-Nourriture*, ou s'il n'existe pas de nourriture il change à l'état *Choisir-Prochaine-Cellule*. Si son $E_c <= E_{min}$, il exécute *Revenir-Base*, s'il existe un chemin de retour, sinon *Revenir-et-Colorer* s'il n'existe pas de chemin.

– *Prendre-Nourriture* : si $E_c > E_{min}$, l'agent capture une quantité limitée de nourriture et propage l'information à son voisin gauche par un dépôt d'une $\mathbb{PT}$. Cependant, si $E_c <= E_{min}$, il revient à la base pour recharger en exécutant *Revenir-Base* (s'il existe un chemin de retour), *Revenir-et-Colorer* (s'il n'existe pas de chemin) sans capturer de nourriture.

– *Choisir-Prochaine-Cellule :* si l'agent détecte un obstacle, il l'évite par exécution de la fonction *éviter_obstacle()*, sinon si l'agent rencontre une ($\mathbb{PT}$), il propage l'information à sa cellule gauche, pour recruter autant que possible d'agents pour une exploitation collective de cette source de nourriture et suivre le chemin de recrutement jusqu'à la nourriture, sinon il dépose une phéromone ($\mathbb{P}$), ajuste sa direction selon la règle de coordination dans l'algorithme 5 et se déplace à la cellule résultante. Il change automatiquement après un pas de mouvement à l'état *Rechercher-Nourriture*.

– *Revenir-Base :* l'agent se déplace à l'une des quatre cellules voisines colorées avec une concentration de $\mathbb{P}$ inférieur à celle de la cellule courante. Il reste dans cet état jusqu'à ce que la base est atteinte, il change alors à l'état *A-Base*.

– *Revenir-et-Colorer :* l'agent se déplace à l'une des quatre cellules voisines avec la concentration de $\mathbb{P}$ minimale et change sa couleur à une couleur de chemin de retour (dépose $\mathbb{PR}$). Il reste dans cet état jusqu'à ce que la base est atteinte, il change alors à l'état *A-Base*.

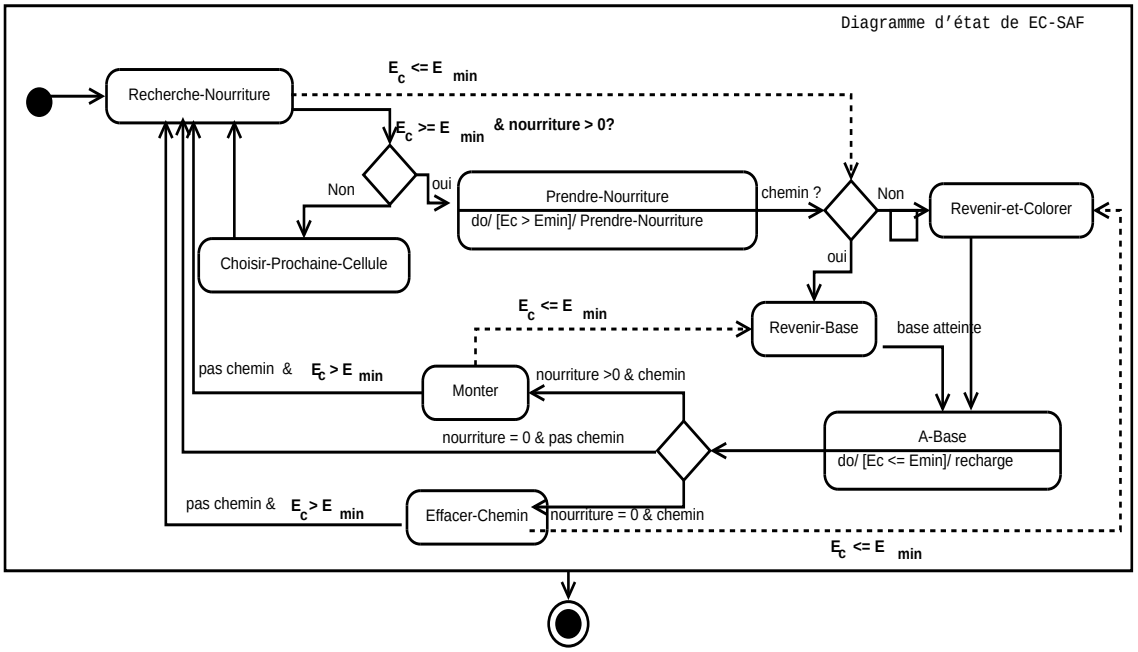


FIGURE 3.7 – Diagramme d'état-transition représentant le comprtement des agents EC-SAF

– *A-Base :* l'agent dépose la nourriture s'il possède. Si son $E_c <= E_{min}$, il recharge son énergie à une valeur maximale (E_{max}). Il exécute alors l'état *Monter* s'il existe un chemin de retour et la nourriture est non encore épuisée, ou exécute l'état *Effacer-Chemin* s'il existe un chemin de retour et la nourriture est épuisée, sinon il exécute *Rechercher-Nourriture*.

– *Monter :* l'agent se déplace à l'une des quatre cellules voisines colorées avec la concentration $\mathbb{P}$ supérieur à celle de la cellule courante. Il reste dans cet état jusqu'à ce que aucun chemin n'est détecté et son $E_c > E_{min}$, il change alors à l'état *Rechercher-Nourriture*. Si $E_c <= E_{min}$ il exécute l'état *Revenir-Base*, pour recharger son énergie.

– *Effacer-Chemin :* l'agent se déplace à l'une des quatre cellules voisines colorées avec une concentration de $\mathbb{P}$ supérieur à celle de la cellule courante et change sa couleur à la couleur par défaut (noir). Il reste dans cet état jusqu'à ce que aucun chemin n'est détecté et son $E_c > E_{min}$, il change alors à l'état *Rechercher-Nourriture*. Si $E_c <= E_{min}$ il exécute l'état *Revenir-et-Colorer*, puisque le chemin existant a été déjà effacé en le montant, l'agent doit revenir recharger et en même temps colorer le chemin pour l'effacer à sa prochaine monté.

Les composantes du système sont modélisés comme suit :

– *Environnement :* une grille de taille NxN. Une cellule contient un ou plusieurs agents, une nour-

Algorithm 6 EC-SAF [154]

if *Rechercher-Nourriture* **then**
  if *($E_c <= E_{min}$)* **then**
    if *($\exists$ chemin)* **then goto** *Revenir-Base* **else goto** *Revenir-et-Colorer*
  else
    if *(nourriture > 0)* **then** Diffuse($\mathbb{P}$) ; **goto** *Prendre-Nourriture* **else goto** *Choisir-Prochaine-Cellule*

if *Choisir-Prochaine-Cellule* **then**
  if *($\exists$ obstacle)* **then** Éviter_Obstacle() **else**
    if *($\exists$ $\mathbb{PT}$ ici)* **et** *($\exists$ $\mathbb{PT}$ à droite)* **then**
      Diffuse($\mathbb{PT}$)
      Monter le chemin de recrutement au nourriture par suivi des $\mathbb{PT}$
    else
      if *($\exists$ $\mathbb{PT}$ ici) et ($\exists$ $\mathbb{PT}$ à droite)* **then**
        Effacer le chemin de recrutement
      else
        Déposer($\mathbb{P}$) Ajustement_Direction ($\mathbb{P}$)(Algorithme 2)
        Évaporation($\mathbb{P}$) 3.1
        Avancer()
  goto *Rechercher-Nourriture*

if *Prendre-Nourriture* **then**
  if *($E_c > E_{min}$)* **then** Prendre une quantité donnée de nourriture
  Diffuser($\mathbb{P}$) **if** *($\exists$ chemin)* **then goto** *Revenir-Base* **else goto** *Revenir-et-Colorer* ;

if *Rrevenir-Base* **then**
  while *base non atteinte* **do**
    Déplacer à un des quatre voisins colorées avec $\mathbb{P} <$ à celle courante
  goto *A-Base* ;

if *Revenir-et-Colorer* **then**
  while *base non atteinte* **do**
    Déplacer à un des quatre voisins colorées avec $\mathbb{P} <$ à celle courante
    Colorer la cellule à une couleur spécifique(jaune)
  goto *A-Base*

if *A-Base* **then**
  Déposer nourriture **if** *($E_c <= E_{min}$)* **then** recharge E_c à E_{max} **if** *($\exists$ chemin* **et** *nourriture > 0)* **then goto** *Monter* **else if** *($\exists$ chemin* **et** *nourriture $= 0$)* **then goto** *Effacer-Chemin* **else goto** *Rechercher-Nourriture*

if *Monter* **then**
  while *$\exists$ chemin* **do**
    if *($E_c <= E_{min}$)* **then goto** *Revenir-Base* **else** Déplacer à un des quatre voisins colorées avec valeur $\mathbb{P} >$ à celle courante
  goto *Rechercher-Nourriture* ;

if *Effacer-Chemin* **then**
  while *$\exists$ chemin* **do**
    if *($E_c <= E_{min}$)* **then goto** *Revenir-et-Colorer* **else** Déplacer à un des quatre voisins colorées avec valeur $\mathbb{P} >$ à celle courante
    Modifier sa couleur à la couleur par défaut(noir)
  goto *Rechercher-Nourriture*

riture (item ou objet), un dépôt ou des obstacles. Il est divisé en carrés égaux dans un système de coordonnées cartésiennes. Les cartes quadrillées (grid maps) sont classés comme une métrique efficace pour la navigation à grande échelle. En robotique, elles sont souvent utilisé pour résoudre des tâches comme la planification de chemin, la localisation, la recherche et la surveillance [7] [145] [77]. Il est efficace que la grille complète est carré et toutes les cellules sont de la même taille d'un agent. N nourriture avec M éléments (nous nous référons par densité et concentration respectivement) sont situés aléatoirement sur la grille. Des obstacles avec des formes rectangulaires ou carrées ont lieu dans des positions fixe sur la grille, le dépôt est au centre (si l'environnement est multi-dépôt, les dépôt prennent des positions prédéfinies pour garantir la complétude des algorithmes).

- *Agent :* Tous les agents de la colonie sont homogènes avec des capacités de perception et de calcul limitées. Ils commencent tous à partir d'une position prédéfinie (le nid) et ont des orientations prédéfinies (0°, 90°, 180° et 270°). Un agent peut effectuer la séquence d'actions suivante :
  - *Détecte si une nourriture existe sur la cellule courante et si il existe une phéromone dans les quatre cellules voisines, ou s'il est au dépôt, et si il peux connaitre s'il charge de nourriture ou non.*
  - *Dépose une quantité de phéromone limitée dans la cellule courante.*
  - *Il peut charger une quantité donnée de nourriture quand il est devant une nourriture ou la décharger quand il est au dépôt.*
  - *Choisir et exécuter des actions selon le percept reçu. Les actions sont définies par le comportement globale d'un agent.*
- *Phéromone et sa gestion :* Trois types de phérmones sont utilisées. La première est utilisée pour marquer les cellules comme visitées et pour ré-pulser les agents de ces cellules la. Ce sont des phérmones ($\mathbb{P}$) digitales dynamiques qui ne diffusent pas mais qui s'évaporent avec un coefficient d'évaporation entre le temps t et $(t+1)$. Cette évaporation est représentée par l'équation 3.1. Les deux autres phéromones sont de type passif (n'ont pas de propreites de diffusion ni d'évaporation), qui sont les phérmones de recrutemenT ($\mathbb{PT}$) et les phérmones de Retour $\mathbb{PR}$. On utilise des couleur différents pour différencier les deux types.

2.4.1 Algorithme Ec-marking

L'algorithme Ec-marking est une version étendue de l'algorithme c-marking permettant la gestion d'énergie. Elle étend la version c-marking par l'ajout de comportements et transitions permettant à l'agent de revenir à la base pour se charger quand son énergie est inférieur ou égale à un seuil prédéfinie. Le comportement des agents Ec-marking est donnée par la Figure 3.8 et la suite des états est expliquées en dessous. L'algorithme Ec-marking est donné par l'algorithme 7.

- *Rechercher-et-Monter :* si nourriture ici > 0 et $E_c > E_{min}$, l'agent exécute l'état *Prendre-Nourriture*. Si $E_c <= E_{min}$, Il vérifie s'il existe un chemin de retour, si il existe un, il exécute *Revenir-Base*, sinon s'il n'existe aucun chemin il exécute *Revenir-et-Colorer*, sinon il se déplace à la nourriture si elle est découverte, sinon il se déplace à un des quatre voisins pas encore marquée.
- *Prendre-Nourriture :* l'agent prend une quantité limitée de nourriture, et si la nourriture est épuisée et il existe un chemin de retour, il exécute *Revenir-et-Effacer-Chemin*, si la nourriture n'est pas encore épuisée et il n'existe pas de chemin l'agent exécute *Revenir-et-Colorer*, sinon il exécute *Revenir-Base*.
- *Revenir-et-Colorer :* l'agent se déplace à l'un des quatre voisins avec la valeur APF minimale et change sa couleur à une couleur de chemin spécifique, il reste dans cet état jusqu'à ce que la base est atteinte, il change alors à l'état *Déposer-Nourriture*.
- *Effacer-Chemin :* l'agent se déplace à l'un des quatre voisins avec la valeur APF minimale et change sa couleur à la couleur par défaut (noir), il reste dans cet état jusqu'à ce que la base est atteinte, il change alors à l'état *Rechercher-et-Monter*.
- *Revenir-Base :* l'agent se déplace à l'un des quatre voisins colorées avec une valeur APF inférieur à celle de la cellule courante, il reste dans cet état jusqu'à ce que la base est atteinte, il change alors à l'état *Rechercher-et-Monter*.
- *Déposer-Nourriture :* l'agent dépose la nourriture à la base. Si $E_c <= E_{min}$, il recharge son énergie à E_{max} et si son statut s'est *recharge* il exécute l'état *Effacer-Recharge-Chemin*, sinon il exécute *Rechercher-et-Monter*.
- *Effacer-Recharge-Chemin :* l'agent se déplace à une cellule voisine colorée avec valeur APF supérieur à celle de la cellule courante et change sa couleur à la couleur par défaut (noir), jusqu'à

ce que aucune cellule colorée n'est détectée, il change alors à l'état *Rechercher-et-Monter*.

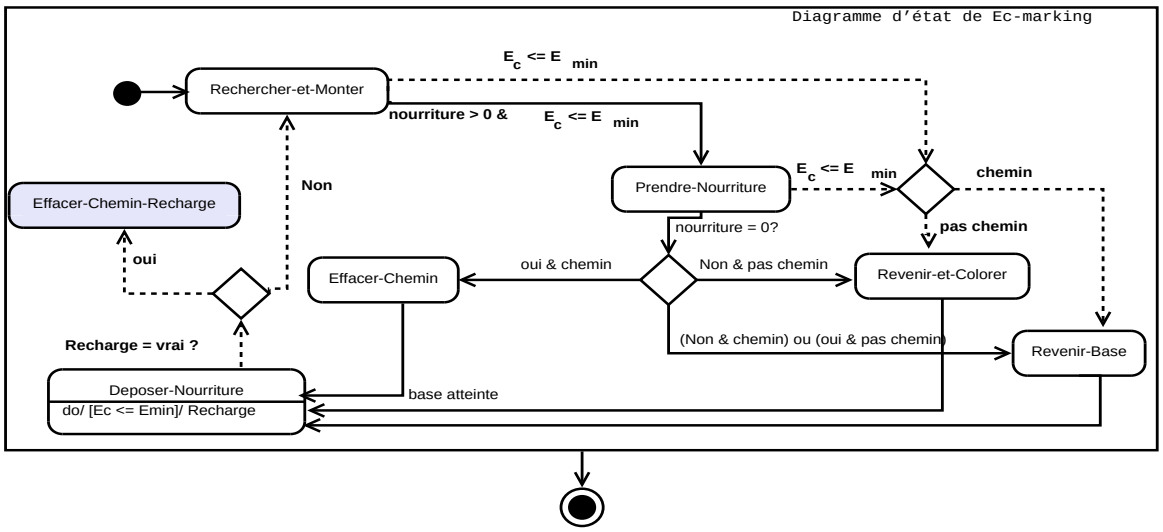


FIGURE 3.8 – Diagramme d'état-transition représentant les agents Ec-marking agents

2.5 Framework pour le foraging Multi-Agents

Le fait de représenter un problème (foraging par example) comme une collection d'individus indépendants qui interagissent ensemble pour atteindre un but, constituent de sérieux progrès vers la résolution des problèmes par algorithmes. Nous voulons que ce Framework soit un guide pour les chercheurs intéressés à la modélisation et la compréhension du comportement de certains systèmes de foraging par simulation. En fournissant un cadre commun, nous facilitons les comparaisons de différents systèmes de foraging. Le langage de modélisation UML a été utilisé pour la représentation des composantes de notre système de foraging et les relations entre eux. Il est extensible étant donné que la définition de chaque objet est indépendante des autres, et il peut être réutilisé dans d'autres algorithmes de foraging. Il est également flexible, puisque d'autres algorithmes de foraging peuvent ajouter des classes importantes, supprimer des classes qui ne sont pas importants dans leurs systèmes, ou de modifier la définition de certaines méthodes (méthode de déplacement d'un agent par exemple), ou même à ajouter d'autres états (comme attente, recharge ..), qui peuvent être définis directement dans la classe comportement. Dans la Figure 3.9, le framework est représenté par un diagramme de classe UML, soulignant les classes fondamentales et les associations entre elles.

Algorithm 7 Ec-marking Algorithm [154]

if *Rechercher-et-Monter(Répéter)* **then**
- **if** *($E_c <= E_{min}$)* **then**
  - **if** *(∃ chemin)* **then goto** *Revenir-Base*
  - **else goto** *Revenir-et-Colorer*
- **else**
  - **if** *(∃ nourriture dans voisinage)* **then** déplacer à cette cellule
  - **goto** *Prendre-Nourriture* **else if** *(∃ chemin)* **then**
  - *déplacer à la cellule colorée avec la valeur APF la plus élevée*
  - **else goto** *Exploration-et-Construction-APF*

if *Prendre-Nourriture* **then**
- **if** *($E_c <= E_{min}$)* **then**
  - **if** *(∃ chemin)* **then goto** *Revenir-Base*
  - **else goto** *Revenir-et-Colorer*
- **else**
  - Prendre une quantité Q_{max} de nourriture **if** *(nourriture non encore épuisé)* **then if** *(∃ chemin)* **then goto** *Revenir-Base*
  - **else goto** *Revenir-et-Colorer*
  - **else goto** *Effacer-Chemin*

if *Revenir-et-Colorer* **then**
- Colorer la cellule avec une couleur spécifique *MAJ_valeur APF* **if** *(base atteinte)* **then goto** *Déposer-Nourriture*
- **else** déplacer à la cellule colorée avec la valeur APF la plus petite

if *Effacer-Chemin* **then**
- **if** *(cellule avec la couleur de chemin)* **then** effacer cette couleur and *MAJ_valeur APF* **if** *(base est atteinte)* **then goto** *Déposer-Nourriture* **else if** *(∃ cellule avec couleur de chemin)* **then** déplacer à une cellule de chemin
- **else** déplacer à la cellule avec la valeur APF la plus petite

if *Déposer-Nourriture* **then**
- Déposer la nourriture **if** *($E_c <= E_{min}$)* **then** Recharger une quantité limitée d'énergie **if** *(recharge = vrai)* **then goto** *Effacer-Recharge-Chemin*
- **else goto** *Rechercher-et-Monter*

if *Effacer-Recharge-Chemin* **then**
- **while** *∃ chemin* **do**
  - **if** *($E_c <= E_{min}$)* **then goto** *Revenir-et-Colorer*
  - **else** déplacer à la cellule colorée avec la valeur APF la plus grande
- **goto** *Rechercher-et-Monter*

Les principales classes de notre Framework sont :

- *Environnement :* est composé de tout les acteurs de la simulation.
- *Cellule :* est le lieu physique où se trouvent les nourritures, les agents, les obstacles et nid (base). Elle a deux variables : *phéromone* et *couleur* qui peuvent être modifiées par les fourrageurs. La valeur de phéromone peut être modifiée au fil du temps par évaporation.
- *Nourriture :* qui peut être exploitée par un ensemble de fourrageurs. C'est un agent stationnaire qui a une position aléatoire et une quantité limité d'unités, elle diminue sa quantité à chaque fois un fourrageur capture une unité de nourriture (voir plusieurs unités). Elle disparait de la simulation quand elle s'épuise ;
- *Obstacle* et *Base* prend place dans certains cellules (aléatoirement ou au centre de l'environnement respectivement).
- *Fourrageur* est un agent, qui peut se déplacer d'une cellule à l'autre en recherchant de la nourriture. Il utilise l'algorithme S-MASA comme stratégie d'exploration, et il dépose une quantité de phéromone à chaque étape pour marquer une cellule comme visitée. Comme le Framwork est modélisé pour être flexible, de nouveaux algorithmes ou même des comportements peuvent être définies ou utilisés, donc d'autres stratégies que S-MASA peuvent êtres utilisées. Les comportements individuels des fourrageurs sont complexes et nous les représentons par une classe séparée (classe comportement). La classe *Fourrageur* donne naissance à deux sous-classes *Transporteur* et *explorateur*, à travers une association de type héritage :

  Transporteur et *explorateur* ce sont des sous-classes de la classe *Fourrageur*. Ils héritent les mêmes attributs et méthodes de cette classe et rajoutent encore des attributs ou des méthodes qui leurs sont spécifiques ou encore redéfinirent certaines méthodes de la classe mère. Par exemple la méthode *se déplacer ()* est redéfinie dans les deux sous-classes *Transporteur* et *explorateur*, parce que la façon de se déplacer chez un *Transporteur* (crée un chemin de retour en suivant la descente du gradient de concentration des phéromones) est différente de celle d'un *Explorateur* (exécuter l'algorithme S-MASA pour une exploration de l'environnement).
- *Comportement :* Inclut la suite des comportements qui peuvent être exécutés par un agent fourrageur, représenté par une suite des états où l'agent change d'un état à l'autre selon le percept reçu de l'environnement (voir Algorithme 6). Les diagrammes d'état-transition dans les Figures 3.3,

3.4 et 3.5, montrent cette suite d'états.

Les classes de notre foraging Framework sont liés par les associations suivantes :

– une composition entre *environnement* et *Fourrageur*, *Nourriture*, *cellule*, *Obstacle*, *Nid*.

– Association 1 à 1 entre *Cellule*, *Nourriture* (la même entre *Cellule* et *Nid*, *Obstacle*. Puisque toute nourriture est située sur une seule cellule.

– Association 1 à * entre *Fourrageur* et *Cellule*, puisque un ou plusieurs agents peuvent prendre place sur la même *Cellule*.

– Auto-association 1 à 4 caractérise la classe *Cellule*, signifie que chaque *Cellule* est associée à ses quatre voisins.

– Une dépendance entre *Fourrageur* et *Nourriture* (aussi entre *Fourrageur* et *Nid*). Durant une simulation, un *Fourrageur* n'est pas en relation permanente avec une *Nourriture* (ou avec le *Nid*), il change son comportement selon la quantité de nourriture restante (voir la quantité dans le *Nid*).

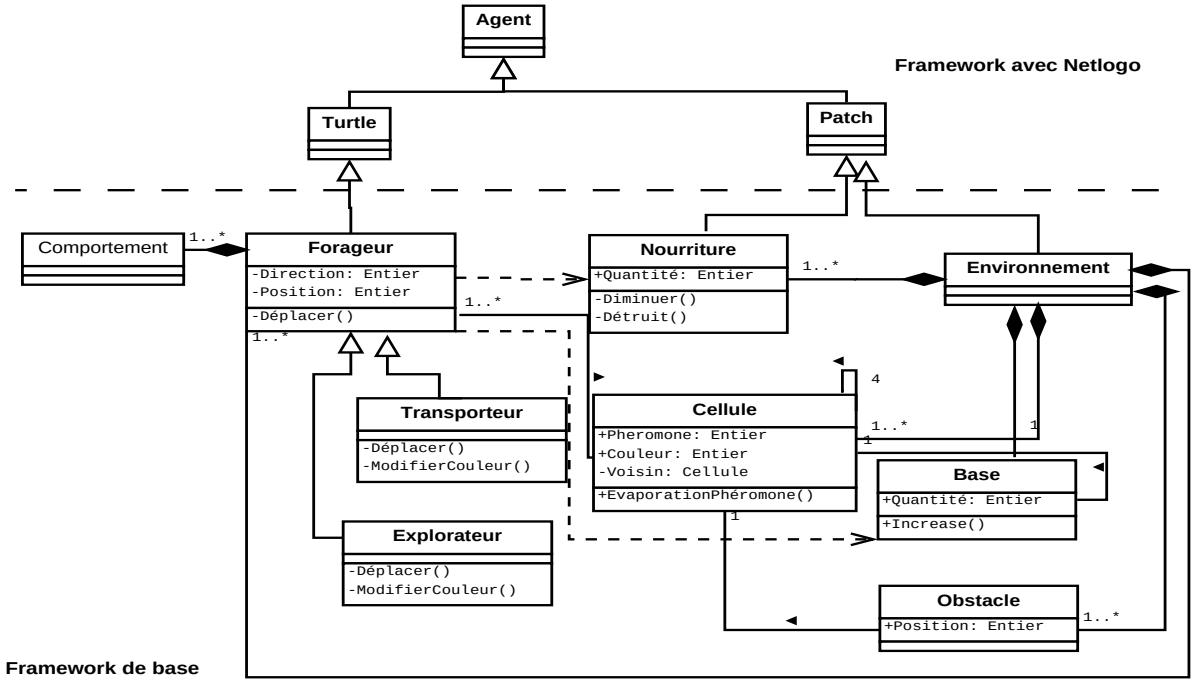


FIGURE 3.9 – Diagramme de classe représentant notre Foraging Framework

3 Conclusion

L'objectif de ce travail été de proposer une approche pour la coordination d'un groupe d'agents réactifs en vue de la résolution coopérative d'un problème inconnu initialement. L'intelligence en essaim est maintenant reconnue comme un paradigme robuste et flexible pour faire face au problèmes distribués dans des environnements complexes. Parmi ces approches, celles basées sur la stigmergie sont les plus populaires pour adresser le problème de foraging. Nous nous intéressons aux groupes d'agents/robots avec des capacités de calcul et de perception limitées avec peu ou pas de mémoire et qui utilisent leur environnement comme mémoire partagée.

Dans ce chapitre nous avons examiné la coordination d'un ensemble d'agents réactifs dans les problèmes de recherche des échantillons, couverture à travers l'algorithme S-MASA qui utilise un marquage dynamique par des phéromones digitales. L'algorithme a été ensuite étendue et adapté pour répondre au problème de foraging multi-agents à travers l'algorithme C-CMFA qui permet de construire des champs de potentiels artificiels à travers des marquages passifs et l'algorithme C-SAF pour le foraging avec un marquage dynamique à travers des phéromones (ou lieu des valeurs entières). Nous avons par la suite analysé et étudié la consommation d'énergie dans ce dernier algorithme où nous avons ajouté des comportements permettant aux agents la gestion de l'énergie le long du foraging. Ces différents algorithmes ont été enfin généralisés en un Framework de foraging modélisé avec UML.

En vue de valider la consistance des algorithmes proposés dans le cadre de coordination dans les problèmes d'exploration et de foraging, nous avons implémenté puis testé et comparé ces différents algorithmes avec des algorithmes connus dans ce domaine. La description des outils développés, les simulations réalisées, les résultats obtenus ainsi les comparaisons feront l'objet du chapitre suivant.

Chapitre 4

Mise en Œuvre et Expérimentations

Sommaire

1 Introduction

Pour mettre en œuvre les différentes propositions présentées dans le chapitre précédent, nous avons développé une plateforme de foraging correspondante à nos algorithmes. Afin de valider nos propositions et les différents choix adoptés nous avons conduit une large série de simulations pour les problèmes de recherche des échantillons, de couverture et de foraging multi-agents. Dans ce chapitre, nous présentons et comparons les résultats obtenus avec les différents algorithmes présentés ultérieurement. Nous commençons par les résultats obtenus avec l'algorithme d'exploration S-MASA pour les deux problèmes recherche des échantillons et de couverture multi-agents et nous le compare avec les algorithmes R-Walk (Random Walk ou marche aléatoire) et S-R-Walk (Stigmergic Random Walk ou marche aléatoire basé stigmergy). Ensuite, nous présentons les résultats

obtenus par notre algorithme de foraging C-CMFA et sa comparaison avec l'algorithme c-marking. Nous montrons aussi les résultats obtenus par C-SAF pour le problème de foraging et sa comparaison avec les algorithmes : la version non-coopérative de C-SAF (NC-SAF), c-marking et la version non-coopérative de ce dernier (NC-c-marking). Après, nous présentons les résultats obtenus par l'algorithme EC-SAF et sa comparaison avec l'algorithme Ec-marking dans des environnements avec et sans obstacles.

2 Résultats de simulation

La simulation a été un outil important pour valider les algorithmes proposés. Nous avons utilisé la plateforme Multi-Agents Netlogo [138] pour la réalisation des simulations.

2.1 Résultats obtenus par S-MASA

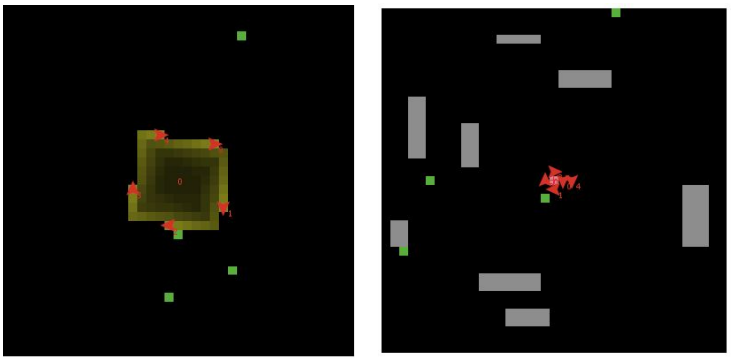
2.1.1 Pour le problème de recherche des échantillons

Scénarios de simulation

Nous avons utilisé deux types de scénarios, pour tester les performances de notre algorithme S-MASA. Dans le premier, nous avons varié le nombre d'agents de *5* à *30* dans des environnements sans et avec obstacles. Dans le deuxième, nous avons varié la taille de l'environnement de *20 X 20* à *100 X 100* (voir Tableau 4.1 pour la description des deux scénarios). Les obstacles dans les deux scénarios ont été définis de deux manières : (1) nous donnons un pourcentage spécifique d'obstacles, les cellules ont été désignées au hasard comme des obstacles, et (2) les obstacles ont été spécifiquement conçus à la main. Pour évaluer les résultats, nous prenons la moyenne de 20 simulations dans la même configuration de l'environnement. Un seul critère de performance a été utilisé : *Temps moyen de recherche*, définie comme le temps nécessaire pour la recherche et pour la localisation de tous les objets dans l'environnement. Les deux configurations d'environnement utilisées pour les simulations sont représentées par la Figure 4.1.

Résultats obtenus et comparaisons

L'algorithme S-MASA a été comparé avec deux autres stratégies d'exploration : *Marche aléatoire*



(a) (b)

FIGURE 4.1 – Les deux configurations d'environnement utilisées pour les simulations de l'algorithme S-MASA. (1) sans obstacles et (2) avec obstacles.

TABLE 4.1 – Paramètres de simulation du scénario 1 et scénario 2

Paramètre	Valeur
Scénario 1	
Nombre des agents	5 – 30
Taille de l'environnement	40 X 40 cellules
Nombre des échantillons	4
Scénario 2	
Taille de l'environnement	20 X 20 – 100 X 100 cellules
Nombre des agents	20
Nombre des échantillons	4

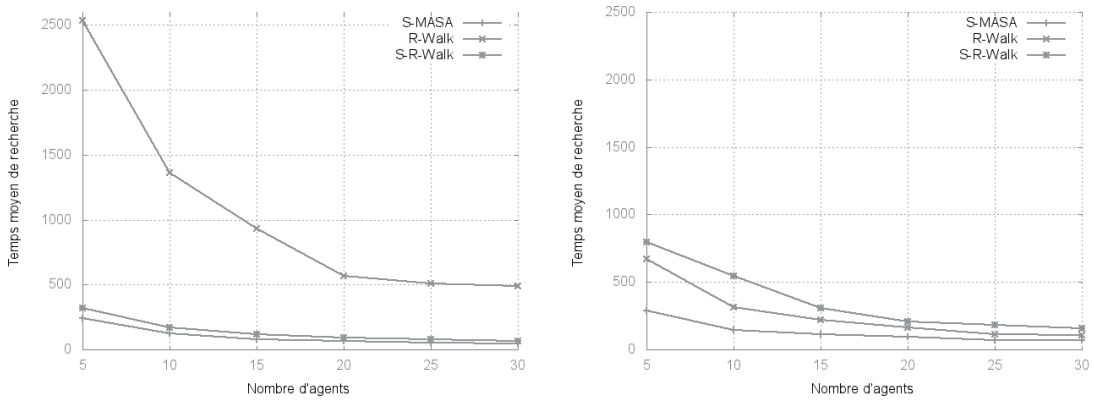
ou Random walk (R-Walk) et *Marche aléatoire basé stigmergie ou Stigmergic Random Walk (S-R-Walk)*. Dans la première, l'agent n'est pas guidé et il choisit la cellule suivante de façon aléatoire même si elle est déjà marquée (l'une des quatre cellules voisines). Tandis que dans la deuxième, les agents sont guidés par des phéromones et donc ils choisissent une cellule des quatre voisins qui n'a pas été encore visitée, sinon ils choisissent une au hasard, les agents déposent des phéromones dynamiques lors de l'exploration pour marquer les cellules comme déjà visitées. Le Tableau 4.2, montre les résultats obtenus par les trois stratégies S-MASA, R-Walk et S-R-Walk, et la Figure 4.2, montre la représentation graphique et la comparaison entre ces différents résultats lorsque on varie le nombre d'agents de 5 à 30, dans des environnements avec et sans obstacles.

Le Tableau 4.2 et la Figure 4.2 montrent que le *Temps moyen de recherche* devient plus rapide avec l'augmentation dans le nombre des agents. Notez que les agents utilisent un type de communication

indirect par phéromones digitales dynamiques. S-MASA donne des meilleurs résultats en comparaison avec les deux autres stratégies. Il est plus rapide et donne les mêmes résultats obtenus par 25 agents dans *S-R-Walk* avec seulement 15 agents dans un environnement sans obstacles.

TABLE 4.2 – Effet de la variation du nombre d'agents sur les performances

	5	**10**	**15**	**20**	**25**	**30**
Environnement sans obstacles						
S-MASA	242,8	122,2	78,8	63,5	54,8	43,9
R-Walk	2536,3	1365,6	932,6	567	508,7	487,3
S-R-Walk	320	171	120	89	80	68
Environnement avec obstacles						
S-MASA	289,8	143,3	114,1	93,5	71,8	69,5
R-Walk	673,5	313,6	218,3	164,7	111,4	105,3
S-R-Walk	798,7	543,1	306,7	205,7	182	157,2



(a) (b)

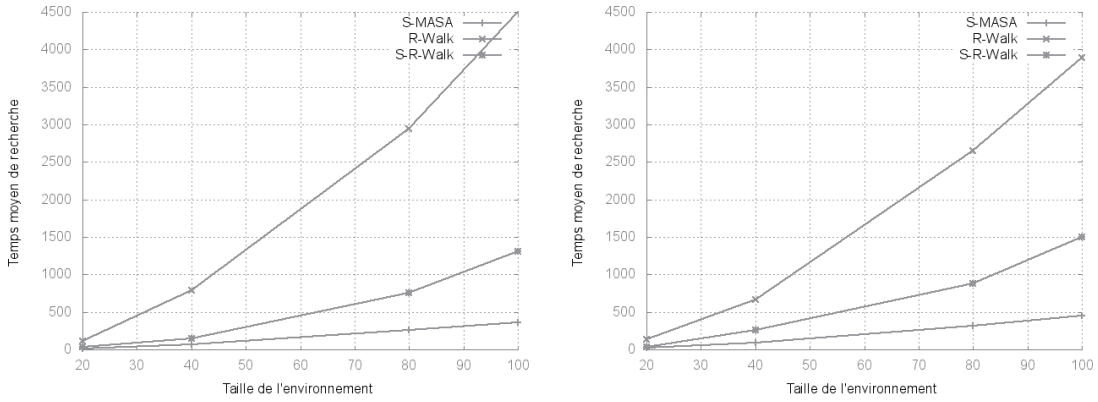
FIGURE 4.2 – Résultats de simulation quand nous varions le nombre d'agents (scénario 1) dans des environnements : (a) sans obstacles, (b) avec obstacles

Le Tableau 4.3 et la Figure 4.3 montrent que le *Temps moyen de recherche* augmente avec l'augmentation de la taille de l'environnement dans les environnements avec et sans obstacles. S-MASA donne des meilleurs résultats par rapport aux deux autres stratégies avec ce scénario aussi, il est de deux fois à trois fois plus rapide que les deux autres stratégies. La marche aléatoire devient inutile et elle augmente de façon considérable quand la taille de l'environnement augmente. Dans les environnements avec obstacles, les résultats de S-MASA augmentent en comparaison avec ceux dans

les environnements sans obstacles, cela revient au nombre des itération en plus nécessaire pour l'évitement d'obstacles dans la direction déjà couverte de l'environnement.

TABLE 4.3 – Effet de variation de la taille de l'environnement sur les performances

	20X20	**40X40**	**80X80**	**100X100**
Environnement sans obstacles				
S-MASA	16,2	63,4	254,8	366,1
R-Walk	108,2	789,6	2946,8	4501,8
S-R-Walk	37,05	149,8	754	1312,2
Environnement avec obstacles				
S-MASA	24,5	92,2	315,3	449
R-Walk	137,8	665,4	2651,5	3889,9
S-R-Walk	39,6	261,5	882,4	1502,7



(a) (b)

FIGURE 4.3 – Résultats de simulation quand nous varions la taille de l'environnent (scénario 2) dans des environnements : (a) sans obstacles, (b) avec obstacles

2.1.2 Pour le problème de couverture Multi-Agents

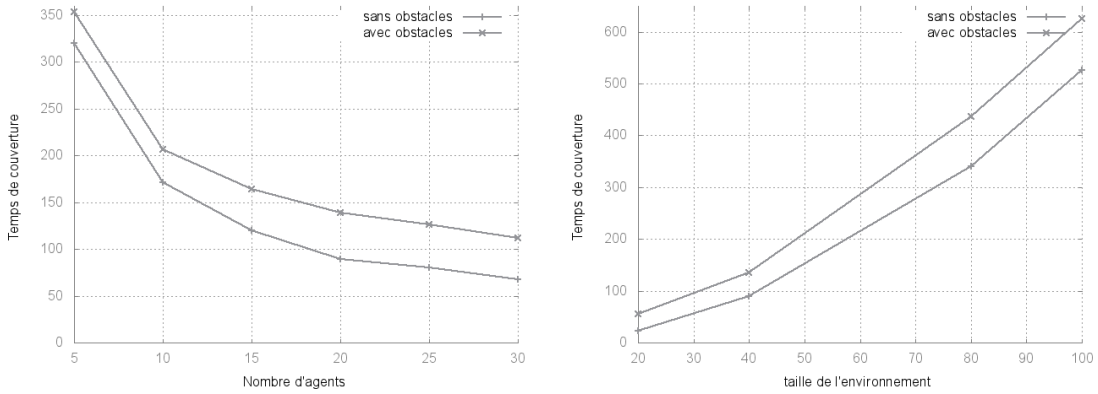
L'algorithme S-MASA donne des résultats de recherche très efficaces à travers des règles de coordination relativement trés simples. Il peut être appliqué à tout problème de navigation robotique (recherche des échantillon, foraging...). Il a été appliqué au problème de couverture multi-robots où un ensemble d'agents visitent leur environnement sans retour sur les zones déjà visitées. La condition d'arrêt dans l'algorithme S-MASA pour le problème de recherche des échantillons dans

le chapitre précédent (Algorithme 1), à été modifiée pour assurer une couverture dans des environnements connus (limités). Nous utilisons les deux types de configuration d'environnements dans la Figure 4.1 (sans échantillons). Deux types de scénarios sont utilisés, nous varions le nombre des agents de 5 à 30 dans le scénario 1 et nous varions la taille de l'environnement de 20x20 cellules à 100x100 cellules.

Les résultats de simulation sont représentés par le Tableau 4.4, Tableau 4.5 et la Figure 4.4. Les résultats montrent que S-MASA donne des temps de couverture très rapide, plus rapide dans les environnement sans obstacles que dans ceux avec obstacles, vue la revisite des cellules déjà visitées pour l'évitement des obstacles. Ce temps rapide revient à la large dispersion garantie par S-MASA, en favorisant les cellules visitées (sauf dans le cas d'obstacles), où les agents déposent des phéromones pour faire pousser les autres agents à d'autres cellules.

TABLE 4.4 – Effet du nombre d'agents sur les performances

	5	**10**	**15**	**20**	**25**	**30**
Environnement sans obstacles	320	171	120	89	80	68
Environnement avec obstacles	354,2	206,7	164,3	138,6	126,2	111,5



(a) (b)

FIGURE 4.4 – Résultats de simulation pour la couverture avec S-MASA dans des environnements avec et sans obstacles, quand nous varions : (a) le nombre d'agents, (b) la taille de l'environnement

TABLE 4.5 – Effet de la taille de l'environnement sur les performances

	20X20	**40X40**	**80X80**	**100X100**
Environnement sans obstacles	23	89	341	527
Environnement avec obstacles	55,4	135,7	435,9	625,1

2.2 Résultats obtenus par C-CMFA

Nous discutons dans cette section les performances de l'algorithme C-CMFA, et nous le comparons avec l'algorithme c-marking dans des environnements inconnus avec et sans obstacles. Nous présentons dans ce qui suit, les indices de performances, les paramètres et les scénarios de simulation utilisés pour évaluer l'algorithme. Nous présentons par la suite les résultats et nous les comparons avec l'algorithme c-marking.

2.2.1 Indices de performances, paramètres et scénarios de simulation

Trois indices de performances ont été utilisés pour tester les performances de notre algorithme :

– *Temps de recherche (exploration) :* c'est le temps totale nécessaire à la découverte d'une nourriture. Nombre d'itérations entre le début de la mission de foraging jusqu'à l'arrivée à une nourriture.
– *Temps de foraging :* c'est le temps nécessaire à l'exploitation (transport) d'une nourriture à la base. Il se mesure en nombre d'itérations.
– *Nourriture totale retournée :* représente la quantité totale de nourriture retournée dans un temps particulier. Elle se mesure en unités.

Les simulations ont été réalisées dans des environnement illimités, avec et sans obstacles. La base et les obstacles prennent des positions fixes et la nourriture est distribuée de façon aléatoire dans une région limitée de l'environnement. Les paramètres dans les trois scénarios utilisés sont présentés dans le Tableau 4.6, où *Densité de nourriture* représente le nombre de localisations de nourriture dans l'environnement, chacune contenant une quantité limitée d'unités de nourriture (*Concentration*), et chaque agent peut transporter une quantité d'unités limitée à chaque fois (*Capacité de l'agent*).

Trois scénarios sont alors définies. Par le scénario 1, nous testons l'effet de la stratégie d'exploration des deux algorithmes (C-CMFA et c-marking) pour mesurer le *Temps de recherche*, où nous varions le nombre des agents de 5 à 30 et nous fixons tout les autres paramètres de la simulation. Par le scénario 2, nous examinons l'effet de la coopération pour l'exploitation des nourritures par un ensemble d'agents coordonnés, nous varions donc le nombre des agents de 5 à 30 et nous enregistrons le *Temps de foraging* à chaque simulation. Par le scénario 3, nous testons les performances de notre algorithme pendant un certain temps écoulé, nous mesurons donc la quantité (en unités) de nourriture retournée et nous varions le nombre des itérations a chaque simulation. Chaque simulation est répétée 20 fois, le résultat moyen est alors calculé de ces 20 simulations pour les trois scénarios.

TABLE 4.6 – Paramètres du scénario 1, scénario 2 et scénario 3

Paramètres	**Valeur**
Scénario 1	*Analyse du temps d'exploration*
Nombre des agents	5–30
Densité de nourriture	1 localisation
Concentration de nourriture	30 unités
Capacité de l'agent	5 unités
Scénario 2	*Analyse du temps d'exploitation*
Nombre des agents	5–30
Densité de nourriture	1 localisation
Concentration de nourriture	60 unités
Capacité de l'agent	5 unités
Scénario 3	*Analyse de nourriture totale retournée*
Nombre des itérations	600–4000
Nombre des agents	10
Densité de nourriture	2 sites
Concentration de nourriture	30 unités
Capacité de l'agent	5 unités

2.2.2 Résultats de simulations et comparaisons

Dans cette section, nous présentons, comparons et discutons les résultats obtenus par les deux algorithmes dans les trois scénarios représentés dans la Tableau 4.6.

Les résultats du scénario 1, prouvent l'efficacité de la stratégie d'exploration utilisée (Algorithme S-MASA) (voir Tableau 4.7 et Figures 4.5(a) 4.5(b)). Avec l'augmentation du nombre des agents, le *temps d'exploration* diminue dans les deux algorithmes. Dans l'algorithme C-CMFA, même avec un petit nombre d'agents nous avons eu des *temps d'exploration* très rapides (346 itérations avec 5 agents), alors que dans l'algorithme c-marking le *temps d'exploration* est beaucoup plus long (9832 itérations avec 5 agents). L'algorithme C-CMFA est cinq fois plus rapide que le c-marking quand le nombre des agents est petit (5–10) et deux à trois fois plus rapide quand le nombre d'agents est grand (15–30). Nous avons obtenu les mêmes résultats de c-marking avec 25 agents, avec seulement 10 agents dans C-CMFA dans les mêmes configurations d'environnement.

TABLE 4.7 – Résultats du scénario 1 : Temps d'exploration

	5	**10**	**15**	**20**	**25**	**30**
sans obstacles						
C-CMFA	*346.4*	*285.6*	*187.4*	*157.7*	*118.7*	*89.4*
c-marking	9832.7	1551.2	541.7	412.3	277.8	202.4
avec obstacles						
C-CMFA	*515.7*	*414.1*	*287.3*	*178.5*	*149.6*	*114.8*
c-marking	11081.2	2310.5	661.2	526.4	303.5	261.8

Le scénario 2, a été fait pour tester l'efficacité de *l'exploitation de nourriture*, qui dépend de la longueur du chemins de retour (reliant une nourriture à la base) et le nombre des agents coopérants à l'exploitation. Nous obtiendrons de bons résultats avec notre algorithme C-CMFA dans des environnements illimités sans et avec obstacles. Le *temps d'exploration* diminue avec toute augmentation dans le nombre des agents dans les deux algorithmes. C-CMFA est deux à quatre fois plus rapide que l'algorithme c-marking. Nous obtiendrons les mêmes résultats que celui du c-marking avec 25 agents, avec seulement 10 agents dans C-CMFA. Ces résultats quantitatives prouvent le bénéfique de la coopération dans le foraging multi-agents. Le Tableau 4.8 et les Figures 4.5(c) 4.5(d) montrent les résultats obtenus dans des environnements illimités avec et sans obstacles.

Dans le scénario 3, nous testons les performances des deux algorithmes (C-CMFA et c-marking), en calculant la quantité en unités de nourriture retournée durant un certain temps écoulé. La quantité de nourriture totale déposée dans l'environnement égale à 60 unités dans deux localisations

TABLE 4.8 – Résultats du scénario 2 : Temps d'exploitation

	5	**10**	**15**	**20**	**25**	**30**
sans obstacles						
C-CMFA	*291.7*	*243*	*172.5*	*140.4*	*125.8*	*92.2*
c-marking	1231.5	894.1	656.7	471.6	284.1	213.7
avec obstacles						
C-CMFA	*335.2*	*271.8*	*206.6*	*192.3*	*157.8*	*132.6*
c-marking	1404.1	1055	847.8	529.6	390.6	294.4

de nourritures. Avec l'augmentation du nombre des itérations, la quantité de nourriture retournée augmente dans les deux algorithmes. C-CMFA atteint la quantité de nourriture totale dans 1000 itérations dans des environnement sans obstacles (1400 itérations dans des environnements avec obstacles). Tandis que, l'algorithme c-marking n'atteint pas la quantité de nourriture totale, parce que l'environnement est illimité et le nombre des agents est insuffisant pour soutenir les cellules correspondantes au front de vague, et la marche pseudo-aléatoire éloigne les agents des localisations de nourritures. Cependant, lorsque on augmente le nombre des agents de 10 à 40 dans c-marking, il atteint la quantité totale dans 1500 itérations dans des environnements sans obstacles (voir 1800 itérations dans des environnements avec obstacles).

TABLE 4.9 – Résultats du scénario 3 : Nourriture totale retournée

	600	**900**	**1000**	**1400**	**2100**	**2500**	**3000**	**3500**	**4000**
sans obstacles									
C-CMFA	*43.7*	*50.7*	*60*	*60*	*60*	*60*	*60*	*60*	*60*
c-marking	19.2	29.5	33.5	40	49.7	57	50.7	52.5	43.5
avec obstacles									
C-CMFA	*31.2*	*40.7*	*47*	*60*	*60*	*60*	*60*	*60*	*60*
c-marking	15.7	22.3	25.2	34	43.2	47.5	45.7	36.2	40.7

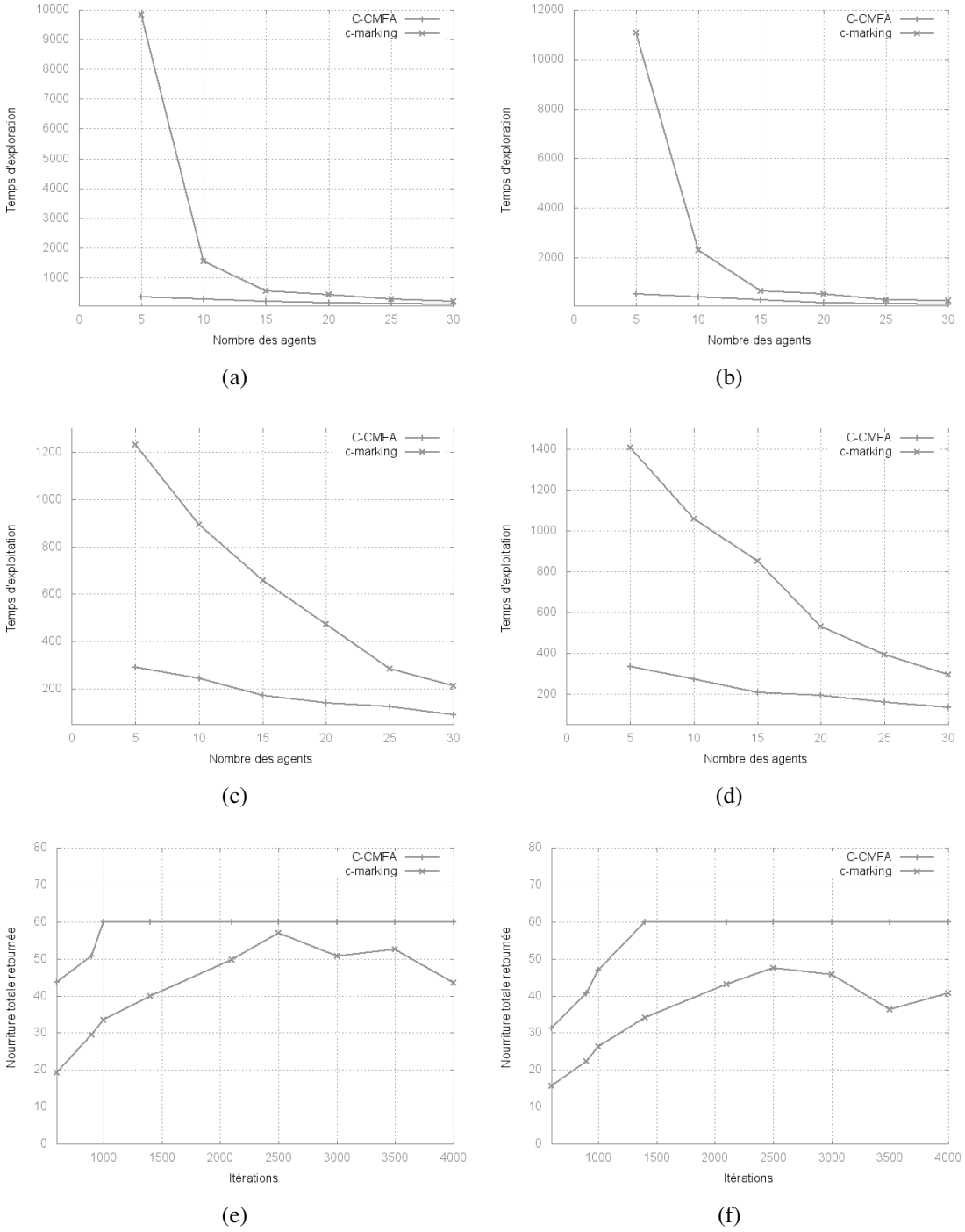


FIGURE 4.5 – Résultats de simulations dans des environnements illimités : (a), (b) Résultats du scénario 1. (c), (d) résultats du scénario 2. (e), (f) Résultats du scénario 3.

2.3 Résultats obtenus par C-SAF

Dans cette section, nous discutons les performances des quatre algorithmes C-SAF, NC-SAF, c-marking et NC-c-marking dans des environnements sans et avec obstacles [155]. Nous présentons dans la section 2.4.2 les paramètres de simulation et les indices de performances utilisées pour évaluer les algorithmes, nous décrivons ensuite trois scénarios fondamentaux et nous rapportons l'importance et les avantages de ces trois scénarios pour la compréhension et la comparaison des quatre algorithmes. Dans la dernière section 2.3.2, nous présentons, comparons et discutons les différents résultats obtenus par les différents algorithmes dans les trois scénarios.

2.3.1 Indices de performance et scénarios de simulation

Les deux configurations de l'environnement utilisées dans les simulations y compris les positions de la base, les nourritures, les agents et les obstacles sont indiquées par la Figure 4.6. Il y a plusieurs paramètres qui doivent être choisis comme : la taille de l'environnement, la densité de nourritures, la concentration de nourriture, la capacité de l'agent et le nombre des agents.

Pour évaluer la performance des algorithmes trois indices de performances ont été utilisés :

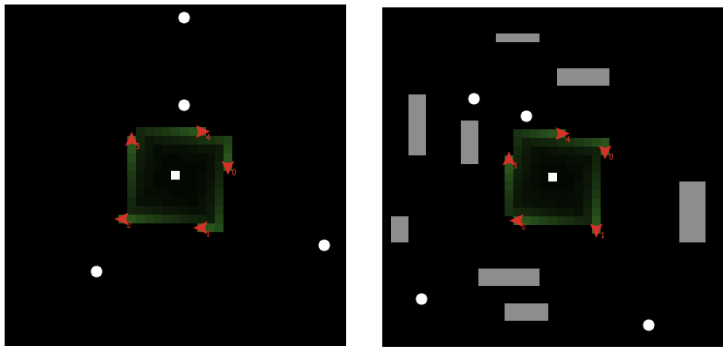
– *Temps de foraging :* le temps de terminaison de la mission. C'est quand toutes les localisations de nourriture ont été découvertes et épuisées. Se mesure en itérations.
– *Nourriture totale retournée :* la quantité totale de nourriture retournée à la base après un temps donné, mesuré en itérations.
– *Longueur du chemin :* représente la longueur du chemin reliant la base et la nourriture. C'est le nombre de cellules composant le chemin.

Trois scénarios fondamentaux ont été utilisés pour les simulations (voir Tableau 4.10). Dans chaque scénario, un des paramètres principaux de la simulation (nombre des agents, capacité de l'agent, taille de l'environnement, densité de nourriture et concentration de nourriture) est varié, pour tester l'effet de chaque paramètre sur la performance des algorithmes en se basant sur les indices de performance définis ultérieurement. Chaque simulation est répétée 20 fois et les résultats moyen de ces 20 simulations sont alors enregistrées. Ces trois scénarios forment les scénarios les plus importants qui nous permettent de : (1) comprendre quel paramètre affecte les performances, (2)

TABLE 4.10 – Paramètres du scénario 1, scénario 2 et scénario 3

Paramètres		Valeur
Scénario 1	*Temps de foraging*	
Taille de l'environnement		40 X 40 – 100 X 100 cellules
Nombre des agents		1 – 30
Densité de nourriture		2 – 10 localisations
Concentration de nourriture		40 – 200 unités
Capacité de nourriture		10 – 500 unités
Scénario 2	*Nourriture totale retournée*	
Nombre des itérations		200–4000
Taille de l'environnement		40 X 40 cellules
Nombre des agents		20
Densité de nourriture		1 localisation
Concentration de nourriture		60 unités
Capacité de nourriture		1 unité
Scénario 3	*Longueur du chemin*	
Taille de l'environnement		40 X 40 cellules
Nombre des agents		1–80
Densité de nourriture		1 localisation
Concentration de nourriture		30 unités
Capacité de nourriture		1 unité

comprendre la dynamique et l'organisation émergente de la modification des paramètres, et (3) faire ressortir les avantages et les inconvénients de chaque algorithme.



(a) (b)

FIGURE 4.6 – Configurations d'environnement utilisé (a) sans obstacles (b) avec obstacles, où les flèches représentent des agents, la cellule blanche au centre c'est la base, les cercles blancs sont les nourritures et les clusters gris sont les obstacles.

2.3.2 Résultats de simulation

Les résultats de simulation dans le sous-scénario 1, quand nous varions le nombre des agents de 1 à 80, montrent que l'algorithme C-SAF surpasse les trois autres algorithmes dans les environnements sans et avec obstacles (voir Tableau 4.11 et les Figures 4.7(a) et 4.7(b)). Ça coûte moins de temps pour trouver la nourriture (*temps d'exploration*), puisque les agents utilisent l'algorithme S-MASA comme stratégie d'exploration, ce qui leur permet une large dispersion par évitement des cellules déjà visitées. Il leur permet aussi de créer des chemins optimaux simultanément quand ils explorent. Le *temps d'exploitation* est aussi minimale du fait que les agents alertent chacun l'autre à la position de nourriture pour une exploitation collective. L'algorithme C-SAF fournit une amélioration importante des performances en comparaison avec l'algorithme c-marking, où 5 agents dans C-SAF terminent la mission de foraging au même temps que 30 agents dans c-marking. A chaque augmentation dans le nombre des agents, le *temps de foraging* diminue.

TABLE 4.11 – Résultats du scénario 1 : en variant le nombre des agents

Nombre des agents	**1**	**5**	**10**	**20**	**25**	**30**	**50**
sans obstacles							
NC-SAF	2236	1294	1111	822	752	714	658
C-SAF	*2295*	*509*	*414*	*293*	*251*	*234*	*218*
NC-c-marking	6900	2236	1777	1393	1228	1096	1786
c-marking	6777	1566	1210	779	665	427	340
avec obstacles							
NC-SAF	3125	1403	1208	833	771	720	718
C-SAF	*3179*	*527*	*429*	*307*	*259*	*237*	*231*
NC-c-marking	9093	2100	1490	1165	1107	985	1918
c-marking	10735	1603	1266	814	641	455	372

Dans le sous-scénario 2, nous varions la taille de l'environnement de 40x40 cellules à 100x100 cellules. L'algorithme C-SAF donne aussi dans ce scénario les meilleurs résultats par rapport aux autres algorithmes dans des environnements sans et avec obstacles (voir Tableau 4.12 et Figures 4.7(c) et 4.7(d)). NC-SAF donne des résultats similaires à celles du c-marking lorsque la taille de l'environnement est inférieure à 60 X 60 cellules, mais il donne des résultats meilleurs lorsque la taille est supérieure à 60 X 60 cellules. Cependant, NC-c-marking donne des mauvaises résultats dans les deux configurations.

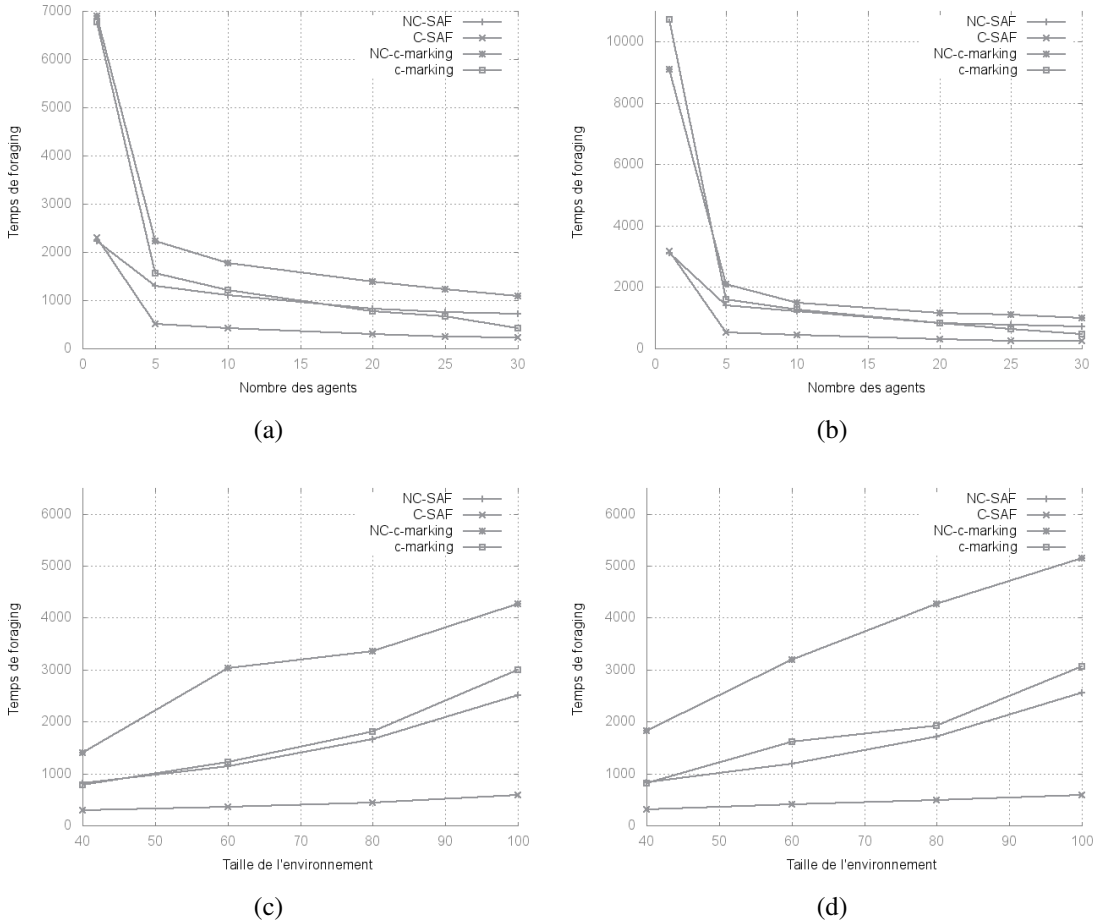


FIGURE 4.7 – Résultats de simulations dans des environnements sans et avec obstacles : (a), (b) quand nous varions le nombre des agents (scénario 1),(c) et (d) quand nous varions la taille de l'environnement (sous-scénario 2).

TABLE 4.12 – Résultats du scénario 1 : en variant la taille de l'environnement

Taille de environnement	**40X40**	**60X60**	**80X80**	**100X100**
sans obstacles				
NC-SAF	822	1144	1666	2515
C-SAF	*293*	*360*	*444*	*594*
NC-c-marking	1393	3033	3355	4270
c-marking	779	1227	1813	3004
avec obstacles				
NC-SAF	833	1194	1716	2565
C-SAF	*307*	*410*	*494*	*594*
NC-c-marking	1824	3198	4263	5155
c-marking	814	1613	1921	3057

Tableau 4.13 et Figures 4.8(a) et 4.8(b) montrent les résultats de simulations obtenus dans le sous-scénario 3. Le *temps de foraging* dans les quatre algorithmes augmente avec l'augmentation de la concentration de nourriture. Cependant, C-SAF fonctionne beaucoup mieux que les trois autres algorithmes. Il est trois à quatre fois plus rapide que le c-marking. Il peut effectuer la même tâche de foraging quand la concentration est 200 unités dans un temps équivalent (134 itérations) à celui du c-marking pour seulement une concentration de 40 unités. Aussi, NC-SAF est meilleur que c-marking dans des environnements sans et avec obstacles.

TABLE 4.13 – Résultats du scénario 1 : en variant la concentration de nourriture

Concentration de nourriture	**40**	**80**	**100**	**140**	**180**	**200**
sans obstacles						
NC-SAF	211.3	279.4	332.5	426.8	645.4	727.4
C-SAF	*111.6*	*138.5*	*169.4*	*180.1*	*195.1*	*245.6*
NC-c-marking	439.4	643.7	748.9	1068.2	1189.4	1261.9
c-marking	379.7	424.3	543.3	588.9	757.9	928.7
avec obstacle						
NC-SAF	242.3	321.9	357.2	464.8	701.5	805.9
C-SAF	*171.5*	*193*	*217*	*233.3*	*253.1*	*276.4*
NC-c-marking	513.7	696.3	813.5	1218.4	1316.3	1440.8
c-marking	400.6	550.2	613.9	648.7	702.8	804.9

Dans le sous-scénario 4 (Tableau 4.14 et Figures 4.8(c) et 4.8(d)), le *temps de foraging* diminue avec l'augmentation de la capacité de l'agent dans les environnements avec et sans obstacles. Dans ce scénario aussi, l'algorithme C-SAF donne des meilleurs résultats , où les agents peuvent terminer la mission de foraging très rapidement (77 et 50 itération) quand nous considérons des grandes capacités de l'agent (200 et 500 unités respectivement). Même si nous considérons de petites capacités d'agent (10 et 20 unités), nous aurons des temps de foraging minimales avec seulement 10 agents, cependant la même performance est obtenue avec des grandes capacités d'agents (100 et 200 unités) avec c-marking (de même avec 10 agents). C-SAF est trois à quatre fois plus rapide que c-marking et des deux autres algorithmes.

Dans le sous-scénario 5 (Tableau 4.15 et Figures 4.8(e) et 4.8(f)), le *temps de foraging* augmente avec l'augmentation dans la densité de nourriture. le temps minimale est toujours donné par l'algorithme C-SAF. Il est deux fois plus rapide que NC-SAF, et environ trois à quatre fois plus rapide que c-marking et beaucoup plus rapide que NC-c-marking dans les deux configurations d'environ-

TABLE 4.14 – Résultats du scénario 1 : en variant la capacité de l'agent

Capacité de l'agent	**10**	**20**	**50**	**100**	**200**	**500**
sans obstacles						
NC-SAF	2241.2	1353.9	621.6	384.6	195.9	121.7
C-SAF	*358.4*	*201.8*	*151.6*	*113.3*	*77.8*	*50.3*
NC-c-marking	4903.4	2430.4	1198.5	623.1	409.6	246.9
c-marking	1362.7	870.3	592.1	362.3	311.6	197.6
avec obstacles						
NC-SAF	3450.6	1736.4	742.7	405.2	217.8	148.7
C-SAF	*787.8*	*337.4*	*175.5*	*142.5*	*95.8*	*78.8*
NC-c-marking	5290.8	2677.7	1310.9	650.9	456.9	303
c-marking	1518.6	896.9	606.9	435.7	361.5	243.5

nements.

TABLE 4.15 – Résultats du scénario 1 : en variant la densité de nourriture

Densité de nourriture	**2**	**3**	**4**	**6**	**8**	**10**
sans obstacles						
NC-SAF	634.3	776.2	929.9	1050.3	1188.4	1404.6
C-SAF	*312.5*	*523.3*	*667.9*	*828.5*	*1038*	*1215*
NC-c-marking	2080.7	2543.3	3000.6	3294.3	4286.7	6558.8
c-marking	1445.2	2255.7	2515.9	3087.3	3830.8	5294.9
avec obstacles						
NC-SAF	702	785.6	997.9	1091.7	1225.7	1464.6
C-SAF	*368.2*	*604.4*	*725.8*	*947.3*	*1173.1*	*1296.6*
NC-c-marking	2980.7	3543.3	4300.6	4394.3	5186.7	7608.8
c-marking	2945.2	3455.7	3515.9	4187.3	5030.8	6094.9

Les algorithmes C-SAF et NC-SAF fournissent des meilleurs résultats par rapport aux autres algorithmes, en fonction de la quantité de nourriture retournée dans 4000 itérations (voir Tableau 4.16 et Figures 4.9(a) 4.9(b)). C-SAF atteint et retourne la quantité de nourriture totale dans seulement 300 itérations, qui est moins de la moitie du temps nécessaire pour atteindre et retourner toute la quantité dans l'algorithme c-marking. Cependant, NC-SAF prend plus de temps pour retourner toute la quantité, il prend 2300 et 2500 itération dans des environnements sans et avec obstacles respectivement, mais ça reste minimale que celui du NC-c-marking qui prend 3950 et 4000 itérations dans des environnements sans et avec obstacles respectivement. Ce scénario est relié au scénario 1, et alors la performance sera la même que scénario 1 quand nous varions les différents paramètres. Si nous varions le nombre des agents par exemple on aura une augmentation dans la quantité de nourriture retournée et C-SAF donne les meilleurs résultats.

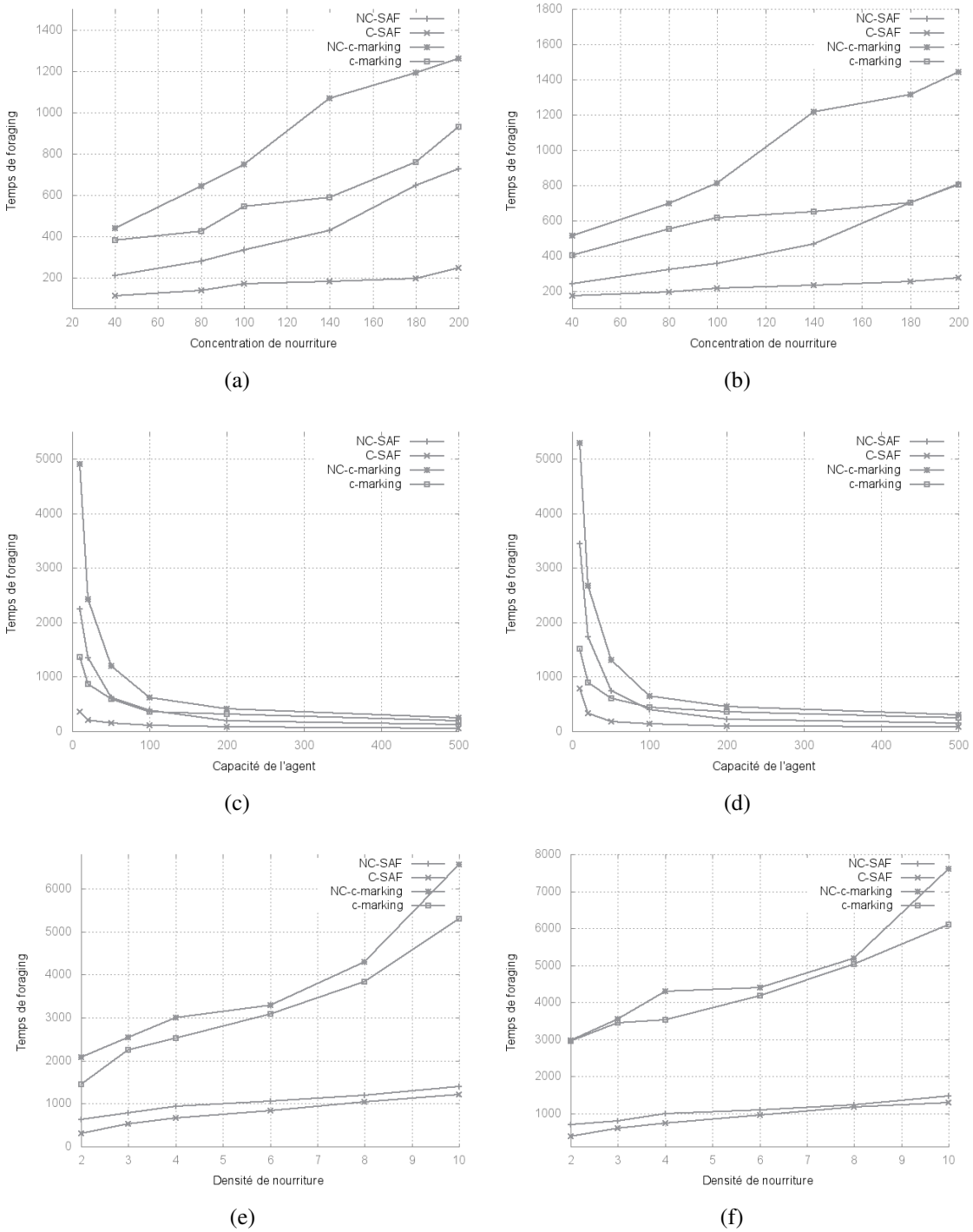
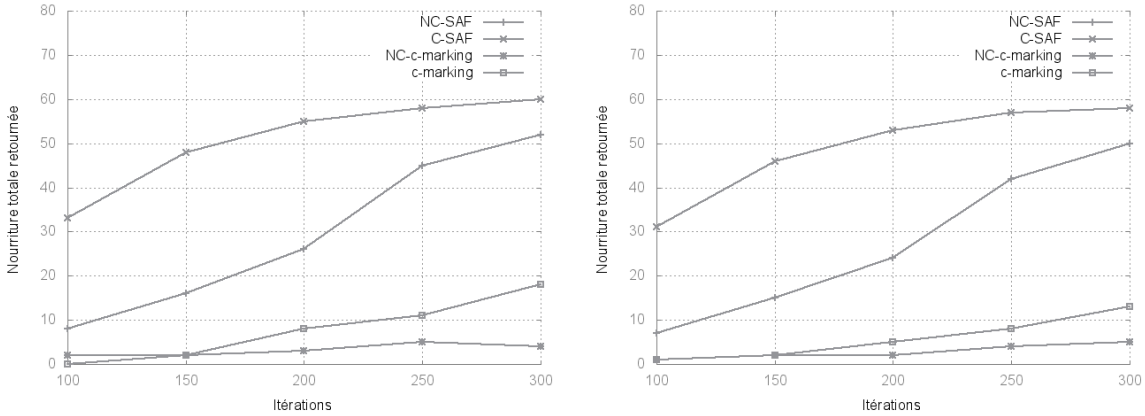


FIGURE 4.8 – résultats de simulations du scénario 1 dans des environnements sans et avec obstacles : (a), (b) Résultats quand nous varions la concentration (sous-scénario 3). (c), (d) Résultats quand nous varions la capacité d'agent (sous-scénario 4). (e), (f) Résultats quand nous varions la densité (sous-scénario 5).

TABLE 4.16 – Résultats du scénario 2 : nourriture retournée dans 4000 itérations

Itérations	200	300	350	850	1100	1300	2300	2500	3950	4000
sans obstacles										
NC-SAF	11	15	22	39	42	49	60	60	60	60
C-SAF	*55*	*60*	*60*	*60*	*60*	*60*	*60*	*60*	*60*	*60*
NC-c-marking	3	4	9	30	43	48	49	51	60	60
c-marking	8	18	26	60	60	60	60	60	60	60
avec obstacles										
NC-SAF	9	11	19	27	33	45	57	60	60	60
C-SAF	*53*	*58*	*60*	*60*	*60*	*60*	*60*	*60*	*60*	*60*
NC-c-marking	2	5	6	16	26	32	42	49	59	60
c-marking	5	13	25	60	60	60	60	60	60	60



(a) (b)

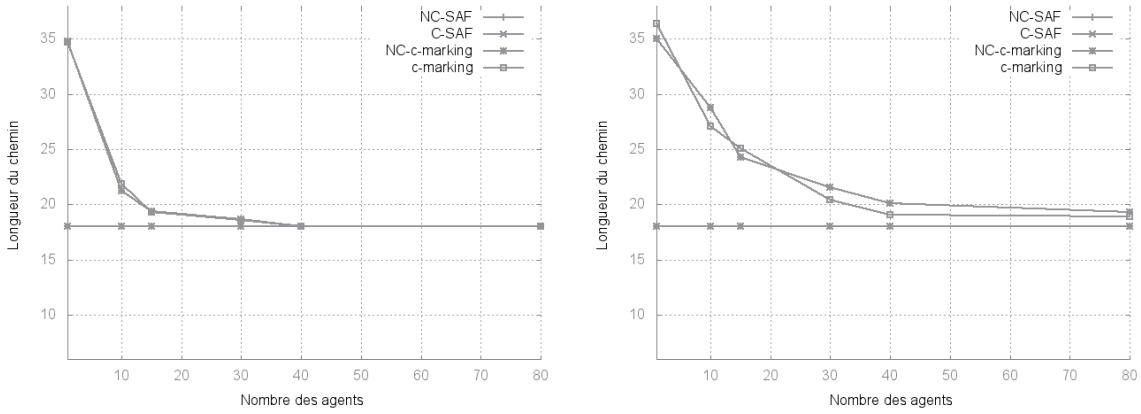
FIGURE 4.9 – Résultats de comparaisons de C-SAF, NC-SAF, c-marking et NC-c-marking, en variant le nombre des itérations (a) sans obstacles (b) avec obstacles

La position de nourriture dans le scénario 2 a été fixée pour exclure l'effet de la position sur les performances. La longueur du chemin reliant la nourriture et la base est calculée en avance. Avec un ou plusieurs agents les algorithmes C-SAF et NC-SAF fournissent des chemins optimaux lors de l'exploration du fait qu'ils utilisent l'algorithme S-MASA. Cependant, avec les algorithmes c-marking, les chemins ne sont pas optimaux et la création est asynchrone, ce qui nécessite plusieurs visites à la même cellule pour atteindre ça valeur optimale. La longueur du chemin diminue avec l'augmentation dans le nombre des agents dans c-marking, elle atteint la valeur optimale avec 40 agents et plus dans des environnements sans obstacles, mais elle n'est pas atteinte dans des environnements avec obstacles et elle augmente quand le nombre des agents est plus de 80 agents (voir Tableau 4.17 et Figures 4.10(a) 4.10(b) pour les résultats). Nous avons varié un seul paramètre dans

ce scénario qui est la *capacité de l'agent*. Nous aurons les mêmes résultats lorsque nous varions la taille de l'environnement, la densité d'obstacles ou même la concentration de nourriture. Avec notre algorithme, le seule paramètre qui affect la longueur du chemin est la position de nourriture, si elle est proche à la base la longueur diminue et si elle est distante la longueur augmente et donc le nombre des itérations augmente.

TABLE 4.17 – Résultats du scénario 3 : longueur du chemin

Nombre des agents	**1**	**10**	**15**	**30**	**40**	**80**
sans obstacles						
NC-SAF	18	18	18	18	18	18
C-SAF	*18*	*18*	*18*	*18*	*18*	*18*
NC-c-marking	34.8	21.2	19.4	18.7	18	18
c-marking	34.75	21.9	19.3	18.6	18	18
avec obstacles						
NC-SAF	18	18	18	18	18	18
C-SAF	*18*	*18*	*18*	*18*	*18*	*18*
NC-c-marking	35	28.8	24.25	21.55	20.1	19.3
c-marking	36.4	27.1	25.1	20.4	19.1	18.9



(a) (b)

FIGURE 4.10 – Résultats de comparaison C-SAF, NC-SAF, c-marking et NC-c-marking selon la longueur du chemin (a) sans obstacles (b) avec obstacles

2.4 Résultats Obtenus par EC-SAF

Dans cette section, nous présentons les résultats de simulation obtenus par les deux algorithmes EC-SAF et Ec-marking (voir chapitre précédent). Pour vérifier qu'un mécanisme de gestion d'énergie

peut améliorer les performances, nous avons comparé en premier lieu les deux versions de chacun des algorithmes avec et sans gestion d'énergie en fonction de la *Nourriture totale retournée*. Après, nous comparons les deux algorithmes EC-SAF et Ec-marking (avec gestion d'énergie) en fonction des indices de performance suivantes : *Nourriture totale retournée*, *Efficacité d'énergie*, *Énergie totale consommée* et *Temps totale de foraging*.

2.4.1 Indices de performance et paramètres de simulation

Pour toute les simulations il y a plusieurs paramètres à fixer :

- *Paramètres de l'agent* : le nombre des agents participant au foraging (*nombre des agents*) et la capacité par unité que chacun peut transporter à chaque fois (*capacité de l'agent*).
- *Paramètres de l'environnement* : inclut la dimension de l'environnement (*la taille*). La *complexité* qui désigne la présence ou non des obstacles et le *nombre de dépôts* qui sont les nids la où les agents doivent stocké la nourriture collectée.
- *Paramètres de nourriture* : le nombre des localisation de nourriture (*densité de nourritures*). Chaque localisation peut prendre plusieurs unités de nourritures définie par la *concentration*.
- E_{max} : représente la quantité maximale dont un agent peut recharger quand son énergie est inférieur à un seuil prédéfinie E_{min}. Sa valeur est fixée après un ensemble de simulation. Une analyse sur l'effet de la valeur de E_{max} sur le temps totale d'exploration et l'énergie totale consommée est donnée par le Tableau 4.18. Quand E_{max} prend une valeur minimale (1000 et 3000 unités), l'énergie des agents tombe souvent au dessus du seuil et ils reviennent plusieurs fois au dépôt (18 et 5 fois respectivement), ce qui augmente le temps d'exploration et l'énergie consommée et donc l'exploration devient inefficace. Alors qu'avec 5000 unités, les retours sont minimales (2 fois) et donc la consommation d'énergie est petite et le temps d'exploration est minimale.
- E_c : représente l'énergie courante d'un agent à un temps t. Elle est initialisé à E_{max} (5000 unités). Quand $E_c <= E_{min}$ l'agent revient au dépôt pour recharger.
- E_{min} : c'est le seuil qui permet d'activer les comportements de *retour au dépôt* et de *recharge*. Elle est fixée par expérimentation à une valeur qui permet au agents de retourner sans l'épuisement de leurs énergie (Tableau 4.19). Avec 100 unités, l'agent ne peut pas atteindre le dépôt parce le

chemin est plus long et nécessite plus d'énergie. Nous avons augmenté la quantité à 300 unités et elle a été suffisante pour atteindre le dépôt pour deux fois (la nourriture a été découverte au même temps). Alors que dans l'algorithme Ec-marking il était difficile de choisir une valeur E_{min} adéquate à cause de la marche pseudo-aléatoire, pour cela nous avons fixé la valeur E_{min} dans les deux algorithmes à 500 unités.

TABLE 4.18 – Analyse de l'E_{max}

	1000	3000	5000	7000	9000
Temps d'exploration	7761,4	5771,3	2530,1	2231,4	2030,1
Énergie totale consommée	12475	10550	10125	9935	6095
Nombre de retour	18	5	2	1	0

TABLE 4.19 – Analyse de l'E_{min}

	100	200	300	400	500
Nombre de retour	0	1	2	2	2
Succès à atteindre le dépôt ?	Non	Non	Oui	Oui	Oui

Quatre indices de performances sont définies comme suit :

– *Nourriture totale retournée :* c'est la quantité de nourriture (en unités) totale retournée après un temps bien déterminé.

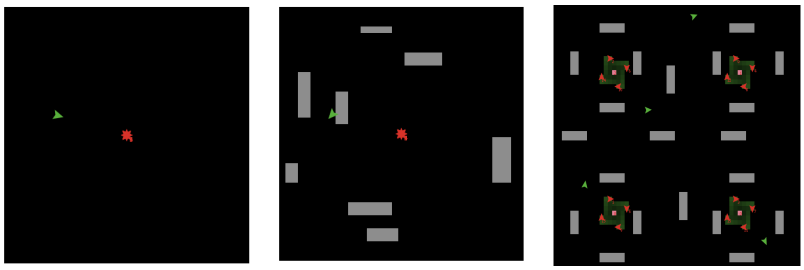
– *Énergie totale consommée :* représente l'énergie totale consommée par tous les agents de l'exploration à la localisation et l'épuisement des sources de nourriture.

– *Efficacité d'énergie :* c'est l'énergie consommée pour l'exploitation d'une source de nourriture. Elle est calculée selon l'équation 4.1. Si on a $E_{eff1} > E_{eff2}$, cela désigne que E_{eff2} est meilleur.

$$E_{eff} = \frac{EnergieTotaleConsomme}{QuantitdeNourritureRetourne} \tag{4.1}$$

où : *Énergie Totale Consommée* représente la somme d'énergie consommée par l'ensemble des agents en exploitation d'une nourriture de sa découverte jusqu'à son épuisement. *Quantité de Nourriture Retournée*, est la quantité de nourriture en unités retournée au dépôt.

– *Temps Total de Foraging ($T_{foraging}$)* représente le temps totale en secondes nécessaire pour découvrir et transporter toute les nourritures dans l'environnement.



(a) (b) (c)

FIGURE 4.11 – Les configurations de l'environnement utilisé pour les simulations (a) sans obstacles (b) avec obstacles (c) environnement avec 4 bases, où les flèches aux centre sont les agents, les flèches dans l'environnement sont les nourritures, les blocs gris sont les obstacles

La consommation d'énergie d'un agent à chaque état a été définie en fonction de l'énergie réelle consommée par l'équipement (capteur, processeur...) pour achever cet état. Elle a été inspirée du modèle B-Swarm [107]. L'agent consomme une unité d'énergie par changement de simulation pour les états qui ne nécessitent pas un travail lourd (comme *Monter*, *Revenir-Base*), alors que dans les états qui nécessitent une manipulation lourde comme : *Prendre-Nourriture*, *Choisir-Cellule-Prochaine*, *A-Base*, *Revenir-et-Colorer*, *Effacer-Chemin*, l'agent consomme 5 unités. Pour l'évitement d'obstacles, l'agent change seulement sa direction et consomme donc 3 unités d'énergie. Pour l'algorithme Ec-marking, l'énergie consommée est 5 unités pour les états : *Rechercher-et-Monter* (quand l'agent exécute Rechercher), *Prendre-Nourriture*, *Revenir-et-Colorer*, *Effacer-Chemin*, *Déposer-Nourriture* et *Effacer-Recharge-Chemin*. Une seule unité pour les états *Revenir-Base* et *Rechercher-et-Monter* (quand l'agent exécute Monter). Pour l'évitement d'obstacles, il consomme 3 unités. Le Tableau 4.20 résume les quantités d'énergie consommés dans chaque état pour les deux algorithmes EC-SAF et Ec-marking.

Comme première étape, nous avons proposé le scénario 1 (voir Tableau 4.21) à partir du quel nous voulons savoir l'utilité de rajouter des comportements de gestion d'énergie à nos agents. Du fait, nous comparons les résultats obtenus par 4 algorithmes : la version avec et sans gestion d'énergie de EC-SAF et la version avec et sans gestion d'énergie de Ec-marking. Dans le scénario 1, nous avons enregistré la quantité totale de nourriture retournée avec les quartes algorithmes jusqu'au temps

TABLE 4.20 – Énergie consommé dans chaque état

états	**Énergie consommée (unité/MAJ simulation)**
états EC-SAF	
A-Base, Choisir-Cellule-Prochaine, Prendre-Nourriture, Effacer-Chemin, Revenir-et-Colorer	5
Monter, Revenir-Base	1
Éviter-Obstacles	3
états Ec-marking	
Rechercher, Prendre-Nourriture, Déposer-Nourriture, Effacer-Chemin Revenir-et-Colorer Effacer-Recharge-Chemin	5
Monter, Revenir-Base	1
Éviter-Obstacles	3

de terminaison de foraging le plus grand (donné par les algorithmes Ec-marking). Les résultats obtenus avec les quatre algorithmes sont représentés par la Figure 4.12. Nous observons d'après cette Figure, que la quantité de nourriture retournée augmente dans les algorithmes sans gestion d'énergie (EC-SAF et Ec-marking) avec le temps. Nous observons que la quantité de nourriture évolue dans les quarte algorithmes avec le temps jusqu'au temps où l'énergie des agents est épuisée dans les deux versions sans gestion d'énergie, et jusqu'à l'épuisement de nourriture dans les deux autres versions avec gestion d'énergie puisque les agents revient au dépôt pour recharger et terminer leurs foraging. Notre algorithme EC-SAF termine le foraging à 1600 seconds (2100 seconds dans des environnements avec obstacles) alors que l'algorithme Ec-marking termine la même tâche dans 2300 seconds (3300 seconds dans des environnements avec obstacles) (voir Figure 4.12). Les agents consomment plus de temps dans l'exploration et donc plus d'énergie particulièrement dans l'algorithme Ec-marking, alors que dans EC-SAF les agents utilisent l'algorithme d'exploration S-MASA qui leur permet des chemins optimaux participant à réduire le temps de rechargement et donc l'énergie consommée.

2.4.2 Scénarios de simulation

Plusieurs simulations ont été réalisées pour tester et comparer les performances des deux algorithmes (EC-SAF et Ec-marking). Les paramètres et les valeurs attribués que nous avons utilisés

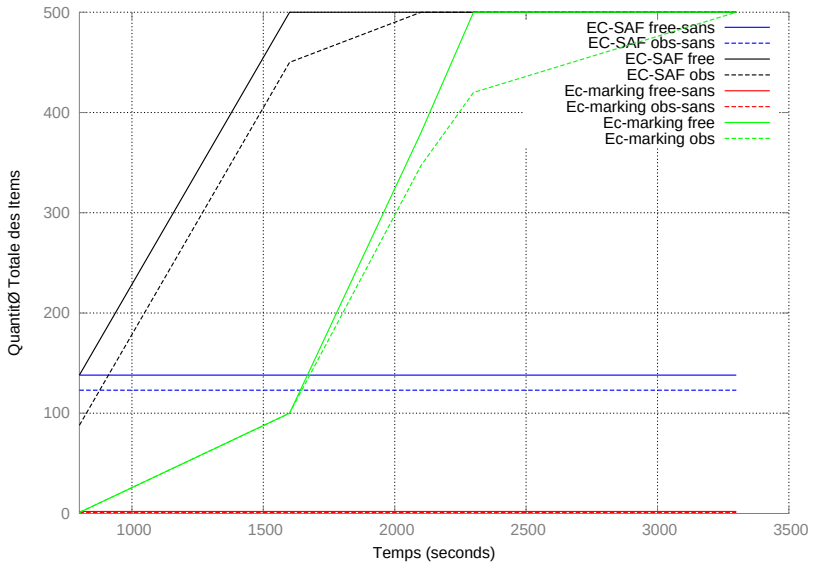


FIGURE 4.12 – Quantité d'énergie totale routourné par les quatres algorithmes dans des environnements avec et sans obstacles

durant les simulations sont dans le Tableau 4.21. Nous avons proposé trois scénarios fondamentaux :

– Scénario 2 a été proposé pour tester lequel des paramètres présentés dans la Section 2.4.2 peut influencé *l'efficacité d'énergie*. Il comporte cinq sous-scénarios. Le sous-scénario 1 montre l'évolution ou la variation de E_eff avec le temps. Sous-scénario 2 permet d'analyser l'influence du nombre d'agents sur les performances, quel nombre donne moins E_eff. Sous-scénario 3 permet de tester comment la taille de l'environnement peut influencer E_eff et y-a-t'il une taille minimale sous laquelle un nombre précis d'agents peut accomplir efficacement la tâche de foraging. Afin d'analyser si le regroupement des unités de nourritures à un seul endroit ou le distribuant sur plusieurs endroits améliore les performances, nous avons défini le sous-scénario 4, où nous varions la *densité de nourriture* de 1 à 10. Pour tester l'effet de la division de l'environnement dans des petits espaces de recherche, nous avons défini le sous-scénario 5, dans lequel nous varions le nombre de dépôts de 1 (un seul espace de recherche) à 64.

– Scénario 3 a été proposé pour montrer un échantillon de l'évolution de *l'énergie totale consommée* par le système durant le processus de foraging.

– Scénario 4 est défini pour étudier l'évolutivité de notre algorithme et si la variation de la *densité*

TABLE 4.21 – Paramètres de scénario 1, scénario 2, scénario 3 et scénario 4

Paramètre	**Valeur**
Scénario 1 *Nourriture totale retournée*	
Taille de l'environnement	1000 X 1000 cellules
Nombre des agents	300
Densité de nourriture	1 localisations
Concentration de nourriture	500 unités
Capacité de l'agent	1 unité
Nombre de dépôts	1
Scénario 2 E_{eff}	
Taille de l'environnement	100x100–1000x1000 cellules
Nombre des agents	100 – 1000
Densité de nourriture	1–10 localisation
Concentration de nourriture	500 unités
Capacité de l'agent	1 unité
Nombre de dépôts	1–64 dépôt
Scénario 3 *Énergie totale consommée*	
Taille de l'environnement	1000x1000 cellules
Nombre des agents	300
Densité de nourriture	2 localisation
Concentration de nourriture	500 unités
Capacité de l'agent	1 unité
Nombre de dépôts	1 dépôt
Scénario 4 *Analyse de scalabilité*	
Densité d'agents	0.00001–0.01
Densité de nourriture	2 localisations
Concentration de nourriture	400 unités
Capacité de l'agent	1–10 unités
Nombre de dépôts	1 dépôt

des agents affecte positivement ou négativement l'E_{eff}. Nous avons utilisé trois sous-scénarios où nous définissons dans chacun la *Densité des agents* comme *nombre des agents/Taille de l'environnement*. Dans le premier sous-scénario, nous avons gardé la *densité des agents* fixe à une valeur donnée, en faisant varier le *nombre des agents* et la *taille de l'environnement* simultanément. Dans le second scénario, nous augmentons la *densité des agents* en gardant le *nombre des agents* fixe et augmentant la *taille de l'environnement*. Dans le troisième sous-scénario, nous diminuons la *densité des agents* en augmentant le *nombre des agents* et gardant la *taille de l'environnement* fixe.

2.4.3 Résultats de simulation

Résultats dans le scénario 2

Les résultats dans la figure 4.13 montrent une diminution de E_{eff} dans les deux algorithmes avec le temps. Dans l'algorithme EC-SAF, lorsque les agents recherchent et déposent des phéromones, ils consomment plus d'énergie, mais lorsque la nourriture est atteinte la quantité totale de nourriture retournés augmente, ainsi le rapport diminue. Cependant, dans l'algorithme Ec-marking, l'E_{eff} est élevée à cause de la marche pseudo-aléatoire qui ralentit le processus d'exploration, après que la nourriture est atteinte, E_{eff} diminue considérablement puisque la quantité totale de nourriture retournée augmente. Les résultats obtenus avec EC-SAF (également avec Ec-marking) dans des environnement avec et sans obstacles sont très proches et leurs courbes se chevauchent dans les Figures 4.13 et 4.14.

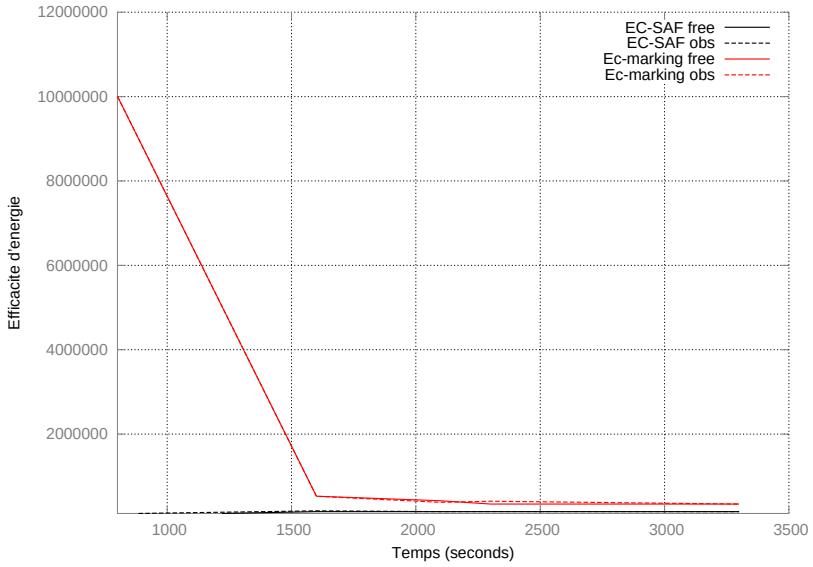


FIGURE 4.13 – Variation de l'E_{eff} durant le temps

La Figure 4.14(a) montre les résultats obtenus par EC-SAF et Ec-marking quand nous varions le nombre d'agent. La croissance de l'E_{eff} dans EC-SAF est constante et lente. Alors que, dans l'algorithme Ec-marking E_{eff} n'est pas stable. Elle augmente et diminue avec le temps, plus élevé avec 100 agents, où ces derniers passent plus de temps à l'exploration (temps de foraging est de 9907 secondes) et consomment donc plus d'énergie. Elle est beaucoup plus moins avec 300 agents (temps de foraging est 2325 secondes seulement). L'exploration rapide fournie par l'algo-

rithme S-MASA [150] dans l'algorithme EC-SAF contribue à réduire le temps d'exploration, ainsi la consommation d'énergie et E_{eff}.

L'E_{eff} diminue avec l'augmentation de la taille de l'environnement. Cette dernière peut affecter les résultats quand elle est petite par rapport au nombre des agents, de ce cas ça peut diminuer le nombre des agents participants au transport de nourriture puisque la plus part de ces agents atteignent rapidement les frontières et donc arrêtent la recherche, même si la nourriture a été déjà localisée par d'autres agents. Quand la taille est suffisamment grande, plus d'agents vont contribuer au transport de nourriture, ce qui augmente la quantité totale de nourriture retournée et donc diminue E_{eff}. Dans Ec-marking, l'E_{eff} change avec le temps quand la taille de l'environnement est 800x800 cellules, elle augmente de façon spectaculaire, alors qu'elle été en diminutaion avec les autres tailles. Une représentation graphique est donnée par la Figure 4.14(b).

L'augmentation de la densité de nourriture permet une amélioration des performances des deux algorithmes. Elle contribue à l'augmentation de la quantité de nourriture retournée parce que à chaque augmentation de la densité de nourriture (nombre de sources) les agents seront distribués entre les différentes sources et la longueur des chemins à traverser par les agents diminue et l'énergie consommée est plus minimale. L'E_{eff} dans les deux algorithmes est proche mais avec l'augmentation de la densité de nourriture les résultats diminuent et s'éloignent les unes des autres. Elle sont très élevés avec Ec-marking. Les plots dans la Figure 4.14(c) présentent les résultats obtenues par les deux algorithmes dans des environnements avec et sans obstacles.

Dans le sous-scénario 5, quand nous divisons l'environnement en plusieurs sous-espaces par l'utilisation de plusieurs dépôts, les sous-espaces résultants sont plus petits et les chemins reliant la nourriture et le dépôt sont plus courts. L'E_{eff} est donc diminué puisque l'énergie consommée est petite et la quantité totale de nourriture retournée est plus grande. Dans Ec-marking, l'E_{eff} est plus élevée que dans EC-SAF (voir Figure 4.14(d)).

Résultats dans le scénario 3

La croissance de l'E_{eff} est constante durant le temps dans les deux algorithmes. Dans l'algorithme EC-SAF, *l'énergie totale consommée* est moins parce que les agents évitent les mouvements inutiles en évitant les cellules déjà visités, alors qu'il est plus élevé dans Ec-marking, puisque les agents

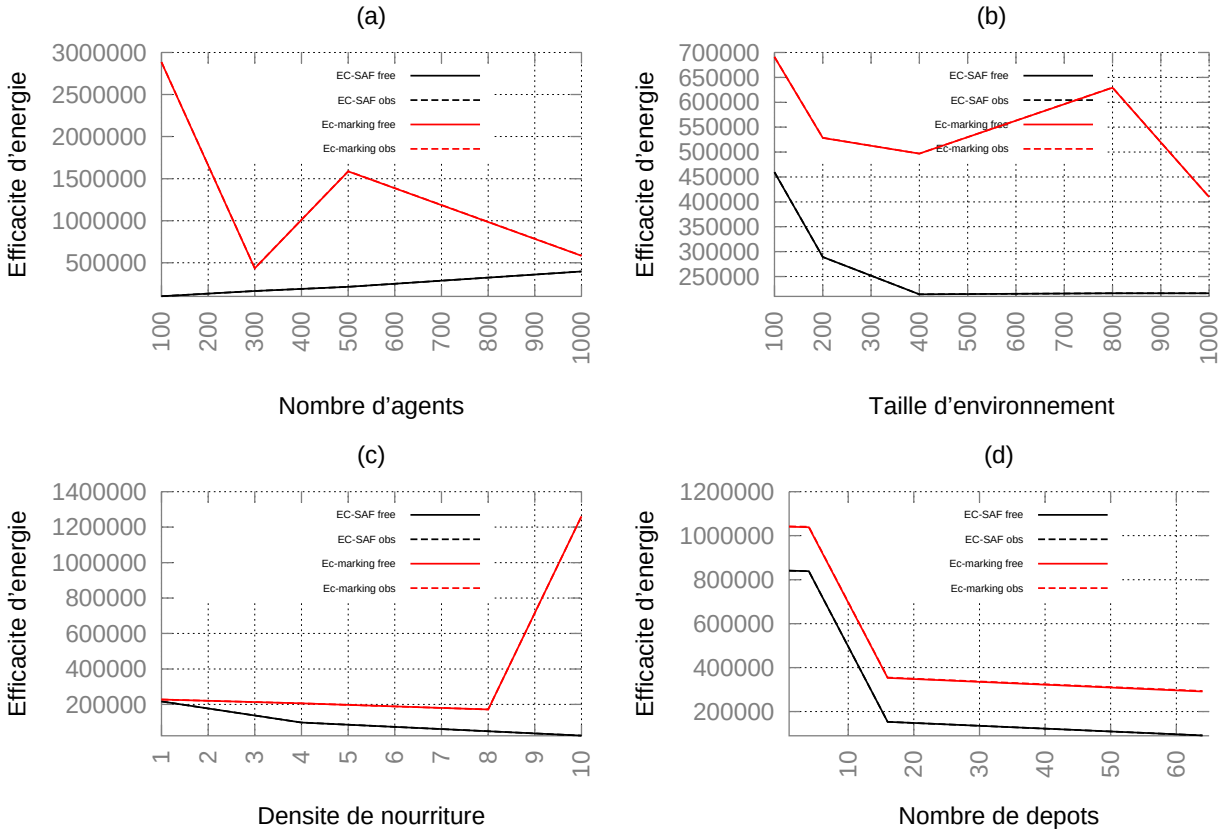


FIGURE 4.14 – Résultats de simulations montrant l'E_{eff} lorsque nous varions : (a) le nombre d'agents, (b) la taille de l'environnement, (c) la densité de nourriture et (d) le nombre de dépôts

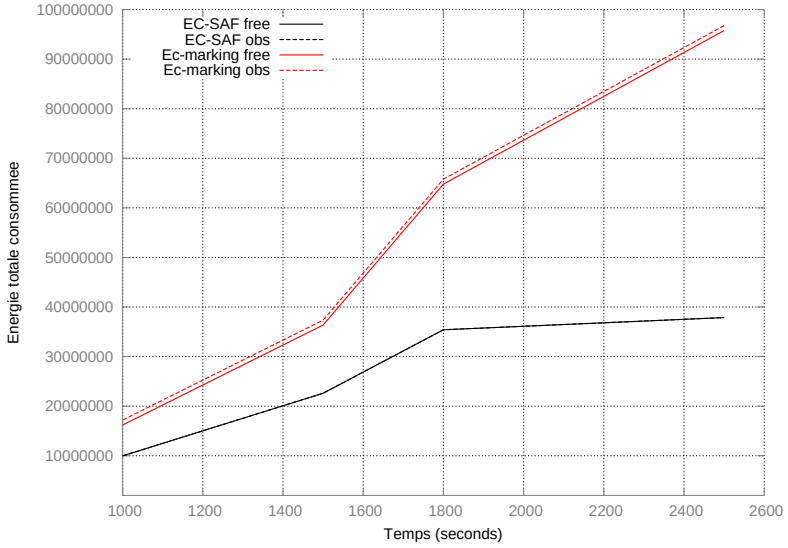


FIGURE 4.15 – Energie totale consommée durant le temps

doivent revenir aux cellules déjà visitée à plusieurs reprises afin d'atteindre les valeurs optimales de champ de potentiel artificiel (Figure 4.15). Puisque les résultats obtenus avec EC-SAF dans des environnements avec et sans obstacles sont très proches, leurs courbes se chevauchent dans la Figure 4.15.

Résultats dans le scénario 4

Lorsque la densité de l'agent est fixe, l'évolution de *l'énergie totale consommée* est linéaire dans les deux algorithmes. Le temps de foraging augmente aussi, ce qui induit plus de temps d'exploration et plus de consommation d'énergie jusqu'à ce que les agents atteignent la quantité totale de nourriture. Varier les deux paramètres (*nombre des agents* et *taille de l'environnement*) au même temps n'affecte pas la croissance de E_{eff}. Tableau 4.22 et la Figure 4.16(a) présentent les résultats obtenus. Aussi dans ce scénario, les résultats obtenus avec EC-SAF (également avec Ec-marking) dans des environnements avec et sans obstacles sont très proches et leurs courbes se chevauchent dans les Figures 4.16(b) et 4.16(c).

Lorsque nous diminuons la densité des agents dans le sous-scénario 2, l'E_{eff} diminue de façon constante et lente dans l'algorithme EC-SAF. La diminution de l'E_{eff} est très faible, puisque le paramètre varié c'est la taille de l'environnement, qui n'a pas d'influence significative sur E_{eff}. Alors que la croissance de E_{eff} n'est pas stable dans Ec-marking, elle évolue de façon linéaire jusqu'à ce que la densité des agents est 0.00004, elle augmente alors de façon spectaculaire à des valeurs plus élevées, et diminue après lorsque la densité est de 0,0001 (voir Figure 4.16(b)).

TABLE 4.22 – Résultats de scénario 3 quand nous fixons la densité des agents à 0.01

Densité des agents	0.01	0.01	0.01	0.01
Nombre des agents	100	2500	3600	6400
Taille de l'environnement	100x100	500x500	600x600	800x800
E_{eff} **environnement sans obstacles**				
EC-SAF	*13794,9*	*190071,5*	*394295,3*	*620946,4*
Ec-marking	720445,6	8262832,8	444642500,0	11616062,5
E_{eff} **environnement avec obstacles**				
EC-SAF	*14940,4*	*205509,8*	*463935,6*	*807330,3*
Ec-marking	758390,1	16526165,5	444652499,0	20202282,6

Dans le sous-scénario 3, plus d'agents avec des chemins de longueur plus courte permet un temps de foraging minimale et moins d'E_{eff}. Elle atteint sa valeur minimale avec la densité 0.0001, au dessous de cette valeur l'augmentation dans le nombre des agents fournit des chemins de longueur plus longs jusqu'à la source de nourriture. Le temps d'exploration augmente alors et aussi l'E_{eff}. Dans l'algorithme Ec-marking, la croissance dans l'E_{eff} est constante mais reste très élevée en comparaison avec celle de l'algorithme EC-SAF (voir Figure 4.16(c)).

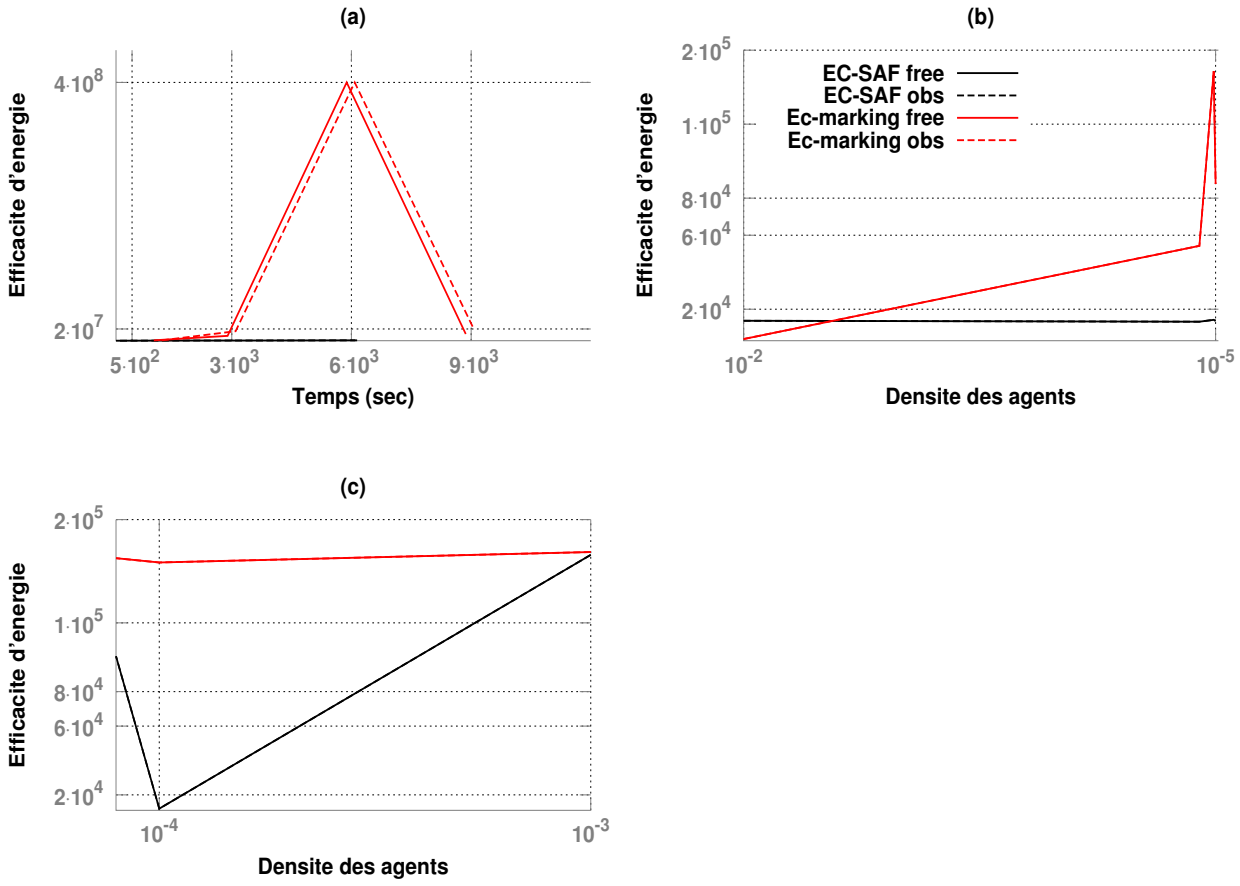


FIGURE 4.16 – Resultats de simulation dans le scénario 3 representent les variations de l'E_{eff} quand nous : *(a) fixons la densité des agents, (b) diminuons la densité des agents, (c) augmentons la densité des agents*

L'évolution de l'E_{eff} dans l'algorithme EC-SAF est constante quand nous varions la densité des agents et n'est pas stable dans l'algorithme Ec-marking. EC-SAF est évolutif et il peut être utilisé avec des systèmes de grande taille avec un grand nombre d'agents et des tailles d'environnements plus larges. Cependant, il présente des résultats meilleurs lorsque certains paramètres de simulation

sont ajustés. Le Tableau 4.23 présente les résultats obtenus en terme de l'E_{eff} et le T_{forag} pour les deux algorithmes lorsque nous varions la densité des agents dans des environnements avec et sans obstacles.

TABLE 4.23 – Variation dans la densité des agents

	0,01	0,001	0,0001	0.0004	0.00001	0.00006
environnement sans obstaclese						
E_{eff} *EC-SAF*	*13794,9*	*159473,4*	*14321,1*	*13278,5*	*100580,6*	*13796,1*
T_{forag} *EC-SAF*	*100*	*4245*	*461*	*501*	*2943*	*775*
E_{eff} Ec-marking	3905,3	161152,8	148589,7	54236	157545	87699,1
T_{forag} Ec-marking	453	4232	4232	794	3948	4516
environnement avec obstacles						
E_{eff} *EC-SAF*	*13797,4*	*159475,9*	*14323,6*	*13281*	*100583,1*	*13798,6*
T_{forag} *EC-SAF*	*202*	*4340*	*565*	*635*	*3043*	*877*
E_{eff} Ec-marking	3930,2	161177,8	148614,7	54261	157570	87724,1
T_{forag} Ec-marking	658	2023	4432	105	4184	4713

Résumé des résultats

La variation de certains paramètres donne des résultats triviaux et ne nécessite pas des simulations pour en déduire tel que : *la capacité de l'agent* et *la concentration de nourriture*. Quand nous varions *la capacité de l'agent*, nous aurons une diminution dans le temps de foraging, une diminution dans l'énergie consommée (du fait que le nombre de visites aux sources diminue) et une augmentation dans la quantité totale de nourriture retournée (du fait que la capacité de l'agent est augmentée), alors l'E_{eff} diminue avec l'augmentation de la capacité de l'agent. Malheureusement, quand nous augmentons *la concentration de nourriture* les agents ont besoin de plus de temps pour récupérer toute la nourriture et ils consomment donc plus d'énergie et l'E_{eff} augmente. Cependant, plusieurs paramètres affectent l'E_{eff} et doivent être ajustés ensemble pour améliorer les performances. *Le nombre des agents* peut limiter l'évolutivité de l'algorithme proposé quand nous les utilisons dans un environnement de petite taille et avec une source unique de nourriture. Dans ce cas, les chemins à traverser par les agents sont très longs, cela permet des temps d'exploration plus longs, moins d'agents participants au transport, plus de consommation d'énergie et moins de nourriture retournée. Alors que, la division de l'environnement par l'utilisation de plusieurs dépôts à des sous-environnements de petite taille fournit des chemins plus courts entre la nourriture et le dépôt ce qui

permet moins de consommation d'énergie et plus de nourriture retournée. L'algorithme EC-SAF fournit des résultats stables et efficaces lorsque nous utilisons un grand nombre d'agents avec des environnements de taille large, et plusieurs sources de nourriture pour assurer plus de recrutement d'agents dans le transport de nourriture et réduire la longueur des chemins à traverser.

3 Conclusion

La validation par implémentation et comparaison des approches proposées dans le chapitre précédent ont été sujet du chapitre courant. Nous avons implémenté une plateforme de foraging multi-agent à travers laquelle nous avons testé les approches proposées. Les résultats obtenus dans les différents problèmes par les différents algorithmes proposés ont été comparés. Les avantages que nous concluront à travers les simulations et les comparaisons faites inclues : (1) la large dispersion des agents induite par l'algorithme S-MASA offre des temps d'exploration plus rapide, (2) la marche selon la dynamique des tourbillons d'eaux offre des chemins optimaux et plus courts construites par l'expansion d'une vague de concentration de phéromones, et (3)la large coopération par recrutement implicite dans les algorithmes de foraging offre des temps de transport plus rapide et donc du foraging rapide.

Depuis les simulations, toutes nos propositions ont été approuvées et validées. Les résultats ont été encourageants et notre souci est de penser à une façon de l'application réelle des approches proposées.

Conclusion générale et perspectives

1 Bilan des travaux et apports de la thèse

Nous avons étudié le problème de la coordination d'un ensemble d'agents réactifs dans le but d'améliorer les performances de groupe par coopération à travers la coordination réactive. Nous avons proposé un algorithme pour la coordination d'un ensemble d'agents réactifs (S-MASA) et un algorithme pour la résolution du problème de foraging multi-agents (C-CMFA). Dans la première contribution nous avons proposé un algorithme de coordination à base de stigmergie et des dynamiques des tourbillons d'eaux. Les agents tournent alors autour d'un axe central (le dépôt) et autour d'eux mêmes et déposent des phéromones digitales pour marquer les régions comme déjà visitées. Ce mouvement produit une large dispersion des agents et une exploration plus rapide.

L'algorithme S-MASA a été testé par des simulations dans différentes configurations d'environnements avec et sans obstacles et donne de bons résultats en temps nécessaire pour localiser toute les échantillons (problème de recherche des échantillons multi-agents) ou pour couvrir l'environnement (problème de couverture multi-agents). Une autre contribution de ce travail est la proposition d'un algorithme pour la résolution du problème de foraging multi-agents. Dans ce dernier problème deux phases sont à définir : la première étant la phase d'exploration et pour cette phase nous avons utilisé notre algorithme d'exploration (S-MASA) et la deuxième phase c'est le retour au dépôt avec les ressources collectées, nous avons proposé un nouveau comportement permettant le retour et nous bénéficions dans cette phase des chemins optimaux créés par l'algorithme S-MASA dans la phase d'exploration, une première version utilise un marquage passif (Algorithme C-CMFA) et une deuxième version utilise un marquage dynamique (Algorithme C-SAF). Une dernière contribution concerne l'adaptation de l'algorithme de foraging pour faire face aux limitations d'énergie d'un

agent pour que ce dernier puisse revenir au dépôt pour se recharger et continuer ses tâches. Chacun des algorithmes à fait l'objet des simulations dans des différentes configurations environnementales, avec des comparaisons avec des algorithmes existants à l'issue desquelles nos algorithmes ont montré leur supériorité.

2 Perspectives

Les perspectives que nous tentons de réaliser dans le future proche sont :

– Adapter l'algorithme d'exploration S-MASA aux environnements dynamiques ;

– D'après les résultats de simulations, nous conclurons que l'algorithme de foraging donne de bons résultats sous l'ajustement de certains paramètres, et nous voulons dans la prochaine version l'adapter d'une façon qu'il sera non dépendant des paramètres de la simulation.

Dans le futur lointain, nous pensons à une mise en œuvre robotique de nos contributions. Une des principales difficultés dans la mise en œuvre matérielle des mécanismes de coordination à base de stigmergie est la mise en œuvre de la phéromone elle-même et malgré les approches proposées, la mise en œuvre de phéromone est encore à ses débuts et la plupart des travaux se trouvent dans les laboratoires de recherche. Un des travaux que nous croyons qu'il constitue un exemple important de la mise en œuvre réelle de notre approche est celui dans [110]. Nous tentons d'utiliser une plateforme de robotique mobile tel que ARGoS [106] afin d'étudier quelles modifications sont nécessaires avant la mise en œuvre réelle de nos algorithmes.

Liste des travaux publiés

1 Revues internationales (avec comité de lecture)

1. O. ZEDADRA, H. SERIDI, N. JOUANDEAU AND G. FORTINO *A Cooperative Switching Algorithm for Multi-Agent Foraging*. Engineering Applications of Artificial Intelligence, Volume 50, 302–319, 2016.

2. O. ZEDADRA, H. SERIDI, N. JOUANDEAU AND G. FORTINO *An Energy-aware Algorithm for Large Scale Foraging Systems*, Scalable Computing : Practice and Experience, Volume 16, No. 4, 2015. URL : https ://www.scpe.org/index.php/scpe/index

2 Conférences internationales (avec comité de lecture)

1. O. ZEDADRA AND N. JOUANDEAU AND H. SERIDI *Cooperative c-marking agents for the foraging problem*, In the Fourth International Conference on Advances in System Simulation, IARIA, (Octobre 2012), pp. , Lisbonne (Portugal).

2. O. ZEDADRA, N. JOUANDEAU AND H. SERIDI, *Collaborative Foraging using a new Pheromone and Behavioral Model*, The First International Symposium on Informatics and its Applications (February 2014), pp. 20, M'sila (Algérie).

3. O. ZEDADRA, N. JOUANDEAU, H. SERIDI AND G. FORTINO *Stigmergic MASA : A Stigmergy Based Algorithm for Multi-Target Search*, In 1st workshop on Multi-Agent Systems and Simulation, 4th Joint Agent-oriented Workshops in Synergy, Federated Conference on Computer Science and Information Systems, (Septembre 2014), pp. 1515-1523, Warsaw (Pologne).

4. O. ZEDADRA, H. SERIDI, N. JOUANDEAU AND G. FORTINO, *A Distributed Foraging Algorithm Based on Artificial Potential Field*, 12th IEEE International Symposium on Programming and Systems, (Avril 2015), pp. 1-6, Algiers (Algérie).
5. G. FORTINO, O. ZEDADRA, N. JOUANDEAU AND H. SERIDI, *A decentralized ant colony foraging model using only stigmergic communication*, In Proceedings of XV Workshop Dagli Oggetti agli Agenti (WOA 2014), vol. 1260. CEUR, 2014, Catania (Italy).
6. O. ZEDADRA AND N. JOUANDEAU AND H. SERIDI AND G. FORTINO *Design and analysis of cooperative and non cooperative stigmergy-based models for foraging*, In IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD), (May 2015), pp. 85-90, Calabria (Italy).
7. O. ZEDADRA AND N. JOUANDEAU AND H. SERIDI AND G. FORTINO , In Federated Conference on Computer Science and Information Systems, 9th International Workshop on Multi-Agent Systems and Simulation (MAS&S'15), (Septembre 2015), Lodz (Pologne).

3 Conférences nationales (avec comité de lecture)

1. O. ZEDADRA AND H. SERIDI *coordination d'agents réactifs dans un environnement incertain*, Première Journée Doctorale, 2011, Guelma, Algérie.
2. O. ZEDADRA AND H. SERIDI *Phéromones digitales pour la résolution du problème de fourragement*, Première Journée nationale des Sciences et Technologies de l'Information et de la Communication (JSTIC 2012), Guelma, Algérie, 2012.
3. O. ZEDADRA AND H. SERIDI *An on-line Multi-Robot Coverage Algorithm*, Deuxième Journée nationale des Sciences et Technologies de l'Information et de la Communication (JSTIC 2013), Guelma, Algérie, 2013.
4. O. ZEDADRA AND H. SERIDI *Multi-Traget Search using Stigmergic Comunication*, Troisième Journée nationale des Sciences et Technologies de l'Information et de la Communication (JSTIC 2014), Guelma, Algérie, 2014.

Bibliographie

[1] *Dictionnaire Hachette langue française*. Hachette Educ (Eds.), 2003. 8

[2] Philip E Agre and David Chapman. Pengi : An implementation of a theory of activity. In *AAAI*, volume 87, pages 286–272, 1987. 11

[3] Richard A Altes. Wavelets, tomography, and line-segment image representations. In *San Diego'90, 8-13 July*, pages 268–278. International Society for Optics and Photonics, 1990. 39

[4] Monica Anderson and Nikolaos Papanikolopoulos. Implicit cooperation strategies for multi-robot search of unknown areas. *Journal of Intelligent and Robotic Systems*, 53(4) :381–397, 2008. 3

[5] Ronald C Arkin. Motor schema based navigation for a mobile robot : An approach to programming by behavior. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 264–271. IEEE, 1987. 33

[6] Ronald C Arkin. *Behavior-based robotics*. MIT press, 1998. 8, 11, 56

[7] Tucker Balch. Grid-based navigation for mobile robots. *The Robotics Practitioner*, 2(1) :6–11, 1996. 69

[8] Tucker Balch. Taxonomies of multirobot task and reward. *Robot teams : From diversity to polymorphism*, pages 23–35, 2002. 46

[9] Jerome Barraquand, Bruno Langlois, and J-C Latombe. Numerical potential field techniques for robot path planning. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(2) :224–241, 1992. 39, 45, 51, 55, 56, 57

[10] George Keith Batchelor. *An introduction to fluid dynamics*. Cambridge university press, 2000. 53

[11] Gerardo Beni and Jing Wang. Swarm intelligence in cellular robotic systems. In *Robots and Biological Systems : Towards a New Bionics ?*, pages 703–712. Springer, 1993. 32

[12] Maren Bennewitz, Wolfram Burgard, and Sebastian Thrun. Optimizing schedules for prioritized path planning of multi-robot systems. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 271–276. IEEE, 2001. 38

[13] Aurélie Beynier. *Une contribution à la résolution des Processus Décisionnels de Markov Décentralisés avec contraintes temporelles*. PhD thesis, Université de Caen, 2006. 17, 18

[14] DW Boeringer and DH Werner. A comparison of particle swarm optimization and genetic algorithms for a phased array synthesis problem. In *Antennas and Propagation Society International Symposium, 2003. IEEE*, volume 1, pages 181–184. IEEE, 2003. 39

[15] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence : from natural to artificial systems*. Number 1. Oxford university press, 1999. 24, 33, 34

[16] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics : a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1) :1–41, 2013. 36, 37

[17] Rodney A Brooks. A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of*, 2(1) :14–23, 1986. 33, 58

[18] Rodney A Brooks. Intelligence without representation. *Artificial intelligence*, 47(1) :139–159, 1991. 2

[19] Rodney A Brooks and Jonathan H Connell. Asynchronous distributed control system for a mobile robot. In *Cambridge Symposium_Intelligent Robotics Systems*, pages 77–84. International Society for Optics and Photonics, 1987. 11

[20] Rodney Allen Brooks. *Cambrian intelligence : the early history of the new AI*, volume 97. Mit Press Cambridge, MA, 1999. 11

[21] James Bruce and Manuela Veloso. Real-time randomized path planning for robot navigation. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2383–2388. IEEE, 2002. 39

[22] Olivier Buffet. *Une double approche modulaire de l'apprentissage par renforcement pour des agents intelligents adaptatifs*. PhD thesis, Université Henri Poincaré-Nancy I, 2003. 16

[23] John Canny. A voronoi method for the piano-movers problem. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 530–535. IEEE, 1985. 39

[24] John Canny. A new algebraic method for robot motion planning and real geometry. In *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pages 39–48. IEEE, 1987. 39

[25] Y Uny Cao, Alex S Fukunaga, and Andrew Kahng. Cooperative mobile robotics : Antecedents and directions. *Autonomous robots*, 4(1) :7–27, 1997. 40, 46

[26] Wayne F Carriker, Pradeep K Khosla, and Bruce H Krogh. The use of simulated annealing to solve the mobile manipulator path planning problem. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 204–209. IEEE, 1990. 39

[27] Cristiano Castelfranchi. Guarantees for autonomy in cognitive agent architecture. In *Intelligent agents*, pages 56–70. Springer, 1995. 21

[28] Brahim Chaib-Draa, Imed Jarras, and Bernard Moulin. Systèmes multi-agents : principes généraux et applications. *Edition Hermès*, pages 1030–1044, 2001. 12, 14

[29] Brahim Chaib-Draa, Bernard Moulin, René Mandiau, and P Millot. Trends in distributed artificial intelligence. *Artificial Intelligence Review*, 6(1) :35–66, 1992. 11, 21

[30] Alexis Champion. Mécanisme de coordination multi-agent fondé sur des jeux : applicationa la simulation comportementale de trafic routier en situation de carrefour. *P h D thesis, Université de V alenciennes et du H ainaut-Cambrésis*, 2(00) :3, 2003. ix, 20, 22, 23

[31] Sarun Chattunyakit, Toshiaki Kondo, Itthisek Nilkhamhang, Teera Phatrapornnant, and Itsuo Kumazawa. Two foraging algorithms for a limited number of swarm robots. In *SICE Annual Conference (SICE), 2013 Proceedings of*, pages 1056–1061. IEEE, 2013. 45, 48

[32] De Hai Chen and Yu Ming Liang. Behavior-based fuzzy control of obstacle avoidance for indoor mobile robot. In *Applied Mechanics and Materials*, volume 34, pages 482–486. Trans Tech Publ, 2010. 39

[33] Philip R Cohen and Hector J Levesque. Intention is choice with commitment. *Artificial intelligence*, 42(2) :213–261, 1990. 11

[34] A. Liang D. Liu, X. Zhou and H. Guan. A swarm intelligence based algorithm for distribute search and collective cleanup. In *IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS)*, volume 2, pages 161–165. IEEE, 2010. 38

[35] Keith S Decker and Victor R Lesser. *Environment centered analysis and design of coordination mechanisms*. PhD thesis, Citeseer, 1995. 21

[36] Yves Demazeau. La plate-forme paco et ses applications. *2èmes Journée Nationale du PRC-IA sur les Systèmes Multi-Agents, PRC-IA, Montpellier, France*, 1993. 18

[37] Yves Demazeau and J-P Müller. *Decentralized Ai*, volume 2. Elsevier, 1990. 10

[38] Jean Louis Deneubourg, Paul Louis Clip, and Scott S Camazine. Ants, buses and robots-self-organization of transportation systems. In *From Perception to Action Conference, 1994., Proceedings*, pages 12–23. IEEE, 1994. 39

[39] Jean-Louis Deneubourg and Simon Goss. Collective patterns and decision-making. *Ethology Ecology & Evolution*, 1(4) :295–311, 1989. 26

[40] Jean-Louis Deneubourg, Guy Theraulaz, Ralph Beckers, P Bourgine, and E Varela. Swarm made architectures. In *1st European Conference on Artificial Life*, pages 123–133. MIT Press, 1992. 24

[41] Daniel C Dennett. *The intentional stance*. MIT press, 1989. 11

[42] Marco Dorigo, Mauro Birattari, and Manuele Brambilla. Swarm robotics. *Scholarpedia*, 9(1) :1463, 2014. 35

[43] Marco Dorigo and Luca Maria Gambardella. Ant colony system : a cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*, 1(1) :53–66, 1997. 34

[44] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system : optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B : Cybernetics, IEEE Transactions on*, 26(1) :29–41, 1996. 2, 52

[45] Alexis Drogoul. *De la simulation multi-agent à la résolution collective de problèmes*. PhD thesis, Université Paris VI, 1993. 26

[46] Gregory Dudek, Michael RM Jenkin, and Evangelos Milios. A taxonomy of multirobot systems. In T. Balch and Natick MA L. Parker, editors. A.K. Peters, editors, *Robot Teams : From Diversity to Poly-morphism*, pages 3–22, 2002. 46

[47] Edmund H Durfee, Victor R Lesser, and Daniel D Corkill. Coherent cooperation among communicating problem solvers. *Computers, IEEE Transactions on*, 100(11) :1275–1291, 1987. 1

[48] Edmund H Durfee, Victor R Lesser, and Daniel D Corkill. Trends in cooperative distributed problem solving. *Knowledge and Data Engineering, IEEE Transactions on*, 1(1) :63–83, 1989. 21

[49] Nicole El Zoghby, Valeria Loscri, Enrico Natalizio, Véronique Cherfaoui, et al. Robot cooperation and swarm intelligence. *Wireless Sensor and Robot Networks : From Topology Control to Communication Aspects*, pages 168–201, 2014. 33

[50] István Engedy and Gábor Horváth. Artificial neural network based mobile robot navigation. In *Intelligent Signal Processing, 2009. WISP 2009. IEEE International Symposium on*, pages 241–246. IEEE, 2009. 39

[51] Jacques Ferber. Les systèmes multi-agents : un aperçu général. *Techniques et sciences informatiques*, 16(8), 1997. 27, 28

[52] Jacques Ferber and Eric Jacopin. The framework of eco-problem solving. *Decentralized AI*, 2 :181–193, 1990. 2

[53] G. Fortino, O. Zedadra, N. Jouandeau, and H. Seridi. A decentralized ant colony foraging model using only stigmergic communication. In *Proceedings of XV Workshop Dagli Oggetti agli Agenti (WOA 2014)*, volume 1260. CEUR, 2014. 45

[54] Yoav Gabriely and Elon Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence*, 31(1-4) :77–98, 2001. 52

[55] Julia Rose Galliers. *A theoretical framework for computer models of cooperative dialogue, acknowledging multi-agent conflict*. PhD thesis, Open University, 1988. 23

[56] S. K. Ghosh and R. Klein. Online algorithms for searching and exploration in the plane. *Computer Science Review*, 4(4) :189–201, 2010. 38

[57] Arnaud Glad. *Etude de l'auto-organisation dans les algorithmes de patrouille multi-agent fondés sur les phéromones digitales*. PhD thesis, Université Nancy II, 2011. 12

[58] Plerre-P Grassé. La reconstruction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. la théorie de la stigmergie : Essai d'interprétation du comportement des termites constructeurs. *Insectes sociaux*, 6(1) :41–80, 1959. 24

[59] Michael J Greene and Deborah M Gordon. How patrollers set foraging direction in harvester ants. *The American Naturalist*, 170(6) :943–948, 2007. 44

[60] Zahia Guessoum. A hybrid agent model : a reactive and cognitive behavior. In *Autonomous Decentralized Systems, 1997. Proceedings. ISADS 97., Third International Symposium on*, pages 25–32. IEEE, 1997. 11

[61] Yi Guo and Lynne E Parker. A distributed and optimal motion planning approach for multiple mobile robots. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 3, pages 2612–2619. IEEE, 2002. 38

[62] Noam Hazon, Fabrizio Mieli, Gal Kaminka, et al. Towards robust on-line multi-robot coverage. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1710–1715. IEEE, 2006. 39

[63] James A Hendler, Austin Tate, and Mark Drummond. Ai planning : Systems and techniques. *AI magazine*, 11(2) :61, 1990. 22

[64] Francis Heylighen. Stigmergy as a universal coordination mechanism : components, varieties and applications. *Human Stigmergy : Theoretical Developments and New Applications. Springer. Retrieved from http ://pespmc1. vub. ac. be/papers/stigmergy-varieties. pdf*, 2015. 24

[65] N. Hoff, R. Wood, and R. Nagpal. Distributed colony-level algorithm switching for robot swarm foraging. In *Springer Distributed Autonomous Robotic Systems*, pages 417–430. 2013. 44, 48

[66] Nicholas R Hoff III, Amelia Sagoff, Robert J Wood, and Radhika Nagpal. Two foraging algorithms for robot swarms using only local communication. In *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*, pages 123–130. IEEE, 2010. 43, 48

[67] Geoffrey Hollinger, Sanjiv Singh, Joseph Djugash, and Athanasios Kehagias. Efficient multi-robot search for a moving target. *The International Journal of Robotics Research*, 28(2) :201–219, 2009. 52

[68] Ferber. J. *Les systèmes multi-agents. Vers une intelligence collective*. InterEditions, Paris, 1995. ix, 2, 3, 9, 10, 11, 12, 13, 15, 16, 17, 19, 23, 25, 33

[69] Nicholas R Jennings, Katia Sycara, and Michael Wooldridge. A roadmap of agent research and development. *Autonomous agents and multi-agent systems*, 1(1) :7–38, 1998. 27

[70] Miguel Juliá, Arturo Gil, Luis Payá, and Oscar Reinoso. Potential field based integrated exploration for multi-robot teams. In *ICINCO-RA (2)*, pages 308–314, 2008. 38

[71] Lydia E Kavraki, Mihail N Kolountzakis, and J-C Latombe. Analysis of probabilistic roadmaps for path planning. *Robotics and Automation, IEEE Transactions on*, 14(1) :166–171, 1998. 38

[72] James Kennedy. *Swarm intelligence*. Springer, 2006. 24

[73] James Kennedy. Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer, 2010. 39

[74] James Kennedy, James F Kennedy, Russell C Eberhart, and Yuhui Shi. *Swarm intelligence*. Morgan Kaufmann, 2001. 32

[75] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1) :90–98, 1986. 24, 33, 39, 56

[76] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012. 38

[77] Boris Lau, Christoph Sprunk, and Wolfram Burgard. Efficient grid-based spatial representations for robot navigation in dynamic environments. *Robotics and Autonomous Systems*, 61(10) :1116–1130, 2013. 69

[78] Steven M LaValle. Rapidly-exploring random trees a new tool for path planning. 1998. 39

[79] Emmanuelle Le Strugeon. *Une méthodologie d'auto-adaptation d'un système multi-agents cognitifs*. PhD thesis, 1995. 18, 21

[80] Jong-Hyun Lee and Chang Wook Ahn. Improving energy efficiency in cooperative foraging swarm robots using behavioral model. In *Bio-Inspired Computing : Theories and Applications (BIC-TA), 2011 Sixth International Conference on*, pages 39–44. IEEE, 2011. 44

[81] Jong-Hyun Lee, Chang Wook Ahn, and Jinung An. A honey bee swarm-inspired cooperation algorithm for foraging swarm robots : An empirical analysis. In *Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME International Conference on*, pages 489–493. IEEE, 2013. 44, 48

[82] Kim D Listmann, Volker Willert, et al. Discoverage : A new paradigm for multi-robot exploration. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 929–934. IEEE, 2010. 38, 39

[83] Xin Ma, Qin Zhang, and Yibin Li. Genetic algorithm-based multi-robot cooperative exploration. In *Control and Automation, 2007. ICCA 2007. IEEE International Conference on*, pages 1018–1023. IEEE, 2007. 37

[84] Thomas W Malone. Modeling coordination in organizations and markets. *Management science*, 33(10) :1317–1332, 1987. 18

[85] Thomas W Malone et al. *What is coordination theory ?* Citeseer, 1988. 19

[86] R Mandiau. Modélisation et évaluation d'organisations multi-agents. *Habilitation à diriger les recherches, Université de Valenciennes et du Hainaut-Cambrésis*, 2000. 18

[87] Ellips Masehian and Mohammad Reza Amin-Naseri. Sensor-based robot motion planning a tabu search approach. *Robotics & Automation Magazine, IEEE*, 15(2) :48–57, 2008. 39

[88] Zhigang Meng, Beiji Zou, and Yu Zeng. Considering direct interaction of artificial ant colony foraging simulation and animation. *Journal of Experimental & Theoretical Artificial Intelligence*, 24(1) :95–107, 2012. 44, 48

[89] Hui Miao and Yu-Chu Tian. Robot path planning in dynamic environments using a simulated annealing based approach. In *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, pages 1253–1258. IEEE, 2008. 39

[90] Peter Miller. The genius of swarms : The study of swarm intelligence is providing insights that can help humans manage complex systems, from truck routing to military robots. *National Geographic*, 212(1) :126, 2007. 32

[91] Marvin Lee Minsky. *The society theory on mind*. Audiovisual Center of the University of Geneva, 1986. 1, 24

[92] Sifat Momen. Ant-inspired decentralized task allocation strategy in groups of mobile agents. *Procedia Computer Science*, 20 :169–176, 2013. 44, 48

[93] Sana Moujahed. Approche multi-agents auto-organisée pour la résolution de contraintes spatiales dans les problèmes de positionnement mono et multi-niveaux. *These de l'université de Technologie de Belfort-Montbéliard et de l'Université de Franche-Comté*, 2007. 33

[94] Richard Moussa. *SEGMENTATION MULTI-AGENTS EN IMAGERIE BIOLOGIQUE ET MÉDICALE : APPLICATION AUX IRM 3D*. PhD thesis, Université Bordeaux I ; Université Sciences et Technologies-Bordeaux I, 2011. 15

[95] FM Muller, PM Franca, and CM Guidini. A diversification strategy for a tabu search heuristic to solve the multiprocessor scheduling problem with sequence dependent setup times. In *Proceedings of the Meeting on CAD/CAM Robotics and Factories of the Future*, pages 217–222, 1995. 39

[96] H Jürgen Müller. Negotiation principles, foundations of distributed artificial intelligence. *GMP OHare, and NR Jennings, New York : John Wiley & Sons*, 1996. ix, 23

[97] N. Jouandeau et H. Seridi O. Zedadra. Collaborative foraging using a new pheromone and behavioral model. In *The First International Symposium on Informatics and its Applications*, page 20, 2014. 45

[98] G. H. Orians and N. E. Pearson. On the theory of central place foraging. *Analysis of ecological system*, pages 155–177, 1979. 38

[99] Sascha Ossowski. *Co-ordination in artificial agent societies : social structures and its implications for autonomous problem-solving agents*. Springer-Verlag, 1999. ix, 21

[100] Esben H Ostergaard, Gaurav S Sukhatme, and Maja J Matari. Emergent bucket brigading : a simple mechanisms for improving performance in multi-robot constrained-space foraging tasks. In *Proceedings of the fifth international conference on Autonomous agents*, pages 29–30. ACM, 2001. 41, 46

[101] Liviu Panait and Sean Luke. A pheromone-based utility model for collaborative foraging. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 36–43. IEEE Computer Society, 2004. 24, 44

[102] Joey K Parker, Ahmad R Khoogar, and David E Goldberg. Inverse kinematics of redundant robots using genetic algorithms. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 271–276. IEEE, 1989. 39

[103] H Van Dyke Parunak, LCDR Michael Purcell, Fleet Composite Squadron SIX, and Mr Robert O'Connell. Digital pheromones for autonomous coordination of swarming uav's. *Ann Arbor*, 1001 :48105–1579, 2002. 34

[104] Yuanteng Pei, Matt W Mutka, and Ning Xi. Coordinated multi-robot real-time exploration with connectivity and bandwidth awareness. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5460–5465. IEEE, 2010. 38

[105] M Peyvandi, M Zafarani, and E Nasr. Comparison of particle swarm optimization and the genetic algorithm in the improvement of power system stability by an sssc-based controller. *Journal of Electrical Engineering and Technology*, 6(2) :182–191, 2011. 39

[106] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, et al. Argos : a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm intelligence*, 6(4) :271–295, 2012. 110

[107] Lenka Pitonakova, Richard Crowder, and Seth Bullock. Understanding the role of recruitment in collective robot foraging. *MIT Press*, 2014. 44, 48, 98

[108] Jim Pugh and Alcherio Martinoli. Inspiring and modeling multi-robot search with particle swarm optimization. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pages 332–339. IEEE, 2007. 52

[109] M. Figueiredo R. Calvo, J. R. de Oliveira and R. A. F. Romero. Bio-inspired coordination of multiple robots systems and stigmergy mechanims to cooperative exploration and surveillance tasks. In *IEEE 5th International Conference on Cybernetics and Intelligent Systems (CIS)*, pages 223–228, 2011. 37

[110] Bijan Ranjbar-Sahraei, Gerhard Weiss, and Ali Nakisaee. A multi-robot coverage approach based on stigmergic communication. In *Multiagent System Technologies*, pages 126–138. Springer, 2012. 110

[111] Anand S Rao, Michael P Georgeff, et al. Bdi agents : From theory to practice. In *ICMAS*, volume 95, pages 312–319, 1995. 12

[112] Craig W Reynolds. Flocks, herds and schools : A distributed behavioral model. *ACM Siggraph Computer Graphics*, 21(4) :25–34, 1987. 24, 25

[113] Brian Rooks. Robots reach the home floor. *Industrial Robot : An International Journal*, 28(1) :27–28, 2001. 36

[114] Stuart J Russell. Rationality and intelligence. *Artificial intelligence*, 94(1) :57–77, 1997. ix, 8, 9

[115] Mitul Saha and Pekka Isto. Multi-robot motion planning by incremental coordination. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5960–5963. IEEE, 2006. 38

[116] Gustavo Sanchez and Jean-Claude Latombe. Using a prm planner to compare centralized and decoupled planning for multi-robot systems. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 2112–2119. IEEE, 2002. 38

[117] SHAHAR Sarid and AMIR Shapiro. H-mrstm : A competitive search algorithm for heterogeneous group of robots. *Int. J. of Factory Automation, Robotics and Soft Computing*, (3) :51–58, 2009. 52

[118] John A Sauter, Robert Matthews, H van Dyke Parunak, and Sven Brueckner. Evolving adaptive pheromone path planning mechanisms. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems : part 1*, pages 434–440. ACM, 2002. 24

[119] John A Sauter, Robert Matthews, H Van Dyke Parunak, and Sven A Brueckner. Performance of digital pheromones for swarming vehicle control. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 903–910. ACM, 2005. 24

[120] Yoav Shoham. Agent-oriented programming. *Artificial intelligence*, 60(1) :51–92, 1993. 8

[121] Yoav Shoham. An overview of agent-oriented programming. *Software agents*, 4, 1997. 9

[122] O. Simonin, F. Charpillet, and E. Thierry. Revisiting wavefront construction with collective agents : an approach to foraging. *Swarm Intelligence*, pages 113–138, 2014. 25, 39, 45, 48, 51, 55, 56, 57

[123] Olivier Simonin. *Le modèle satisfaction-altruisme : coopération et résolution de conflits entre agents situés réactifs, application à la robotique*. PhD thesis, Montpellier 2, 2001. ix, 10, 12, 13, 25, 33, 41

[124] Olivier Simonin. Construction of numerical potential fields with reactive agents. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1351–1352. ACM, 2005. 39, 55, 56

[125] Olivier Simonin. *Contribution à la résolution collective de problème (Modèles d'auto-organisation par interactions directes et indirectes dans les SMA réactifs et robotiques)*. PhD thesis, Université Henri Poincaré-Nancy I, 2010. 2, 24, 62

[126] Reid G Smith. The contract net protocol : High-level communication and control in a distributed problem solver. *IEEE Transactions on computers*, (12) :1104–1113, 1980. 18

[127] Ricard V Solé, Eric Bonabeau, Jordi Delgado, Pau Fernández, and Jesus Marín. Pattern formation and optimization in army ant raids. *Artificial life*, 6(3) :219–226, 2000. 53

[128] M Soria. Le banc de poissons : expression d'une motivation sociale ou comportement collectif d'un groupe auto-organisé. *Auto-organisation et comportement. Hermes, Paris*, pages 141–156, 1997. 25

[129] Luc Steels. Cooperation between distributed agents through self-orcamsation. 1989. 11, 26

[130] Anthony Stentz. Optimal and efficient path planning for partially-known environments. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 3310–3317. IEEE, 1994. 38

[131] David JT Sumpter and Madeleine Beekman. From nonlinearity to optimality : pheromone trail foraging by ants. *Animal behaviour*, 66(2) :273–280, 2003. 24

[132] I. Tanev T. Kuyucu and K. Shimohara. Evolutionary optimization of pheromone-based stigmergic communication. In *Applications of Evolutionary Computation*, pages 63–72. Springer, 2012. 38

[133] Shigeru Takano and Einoshin Suzuki. New object detection for on-board robot vision by lifting complex wavelet transforms. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 911–916. IEEE, 2011. 39

[134] Herbert G Tanner and Amit Kumar. Towards decentralization of multi-robot navigation functions. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 4132–4137. IEEE, 2005. 39

[135] Vincent Thomas. *Proposition d'un formalisme pour la construction automatique d'interactions dans les systèmes multi-agents réactifs*. PhD thesis, Université Henri Poincaré-Nancy I, 2005. 2, 14

[136] Alan M Turing. Computing machinery and intelligence. *Mind*, pages 433–460, 1950. 1

[137] George Vachtsevanos and Henry Hexmoor. A fuzzy logic approach to robotic path planning with obstacle avoidance. In *Decision and Control, 1986 25th IEEE Conference on*, pages 1262–1264. IEEE, 1986. 39

[138] U. Wilensky. Netlogo. http ://ccl.northwestern.edu/netlogo/,. In *Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL*, 1999. 44, 77

[139] Alan FT Winfield. Foraging robots. In *Encyclopedia of Complexity and Systems Science*, pages 3682–3700. Springer, 2009. ix, 40, 43, 46

[140] Alan FT Winfield, Serge Kernbach, and Thomas Schmickl. Collective foraging : cleaning, energy harvesting and trophallaxis. *Handbook of Collective Robotics : Fundamentals and Challenges, Pan Stanford Publishing, Singapore*, pages 257–300, 2011. 42

[141] Michael Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2009. 27

[142] Michael Wooldridge and Nicholas R Jennings. Agent theories, architectures, and languages : a survey. In *Intelligent agents*, pages 1–39. Springer, 1995. 7, 18

[143] Kai M Wurm, Cyrill Stachniss, and Wolfram Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1160–1165. IEEE, 2008. 39

[144] Q Xue, YP Chien, YUANLI FENG, and MJ Palakal. A task planner for intelligent mobile robots using subgoal priority information. *International Journal on Artificial Intelligence Tools*, 3(01) :1–21, 1994. 39

[145] Yap Ping Yean and RM Kuppan Chetty. An efficient grid based navigation of wheeled mobile robots based on visual perception. In *Trends in Intelligent Robotics, Automation, and Manufacturing*, pages 128–135. Springer, 2012. 69

[146] Naigong Yu and Bo Li. Study on mobile robot mapping based on binocular vision and voronoi diagram. In *Electrical and Control Engineering (ICECE), 2011 International Conference on*, pages 860–863. IEEE, 2011. 39

[147] Jing Yuan, Yalou Huang, Tong Tao, and Fengchi Sun. A cooperative approach for multi-robot area exploration. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1390–1395. IEEE, 2010. 37

[148] Miriam Zacksenhouse, Rui JP DeFigueiredo, and Don H Johnson. A neural network architecture for cue-based motion planning. In *Decision and Control, 1988., Proceedings of the 27th IEEE Conference on*, pages 324–327. IEEE, 1988. 39

[149] O Zedadra, N Jouandeau, and H Seridi. Cooperative c-marking agents for the foraging problem. In *the Fourth International Conference on Advances in System Simulation, IARIA*, 2012. 45

[150] O. Zedadra, N. Jouandeau, H. Seridi, and G. Fortino. S-MASA : A stigmergy based algorithm for multi-target search. In M. Paprzycki M. Ganzha, L. Maciaszek, editor, *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*, volume 2 of *Annals of Computer Science and Information Systems*, pages 1477–1485. IEEE, 2014. 38, 45, 51, 103

[151] O. Zedadra, N. Jouandeau, H. Seridi, and G. Fortino. Design and analysis of cooperative and non-cooperative stigmergy-based models for foraging. In *Proceedings of the 19th IEEE International Conference on Computer Supported Cooperative Work in Design*, 2015. xii, 45, 48, 51, 62, 64

[152] O Zedadra, H Seridi, N Jouandeau, and G Fortino. A distributed foraging algorithm based on artificial potential field. In *Programming and Systems (ISPS), 2015 12th International Symposium on*, pages 1–6. IEEE, 2015. xii, 45, 48, 51, 55, 60

[153] O Zedadra, H Seridi, N Jouandeau, and G Fortino. Energy expenditure in multi-agent foraging : An empirical analysis. In *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 1773–1778. IEEE, 2015. 46, 48, 51

[154] O Zedadra, H Seridi, N Jouandeau, and G Fortino. An energy-aware algorithm for large scale foraging systems. *Scalable Computing : Practice and Experience*, 16(4) :–, 2015 (à paraitre). xii, 46, 51, 65, 68, 72

[155] Ouarda Zedadra, Hamid Seridi, Nicolas Jouandeau, and Giancarlo Fortino. A cooperative switching algorithm for multi-agent foraging. *Engineering Applications of Artificial Intelligence*, 50 :302–319, 2016. xii, 41, 51, 62, 64, 87

[156] Juanping Zhao and Xiuhui Fu. Improved ant colony optimization algorithm and its application on path planning of mobile robot. *Journal of Computers*, 7(8) :2055–2062, 2012. 39

[157] Qidan Zhu, Yongjie Yan, and Zhuoyi Xing. Robot path planning based on artificial potential field approach with simulated annealing. In *Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on*, volume 2, pages 622–627. IEEE, 2006. 56