Analyse d'un algorithme d'intelligence en essaim pour le fourragement

Amine Boumaza, Bruno Scherrer

Boumaza, A. and Scherrer, B. Analyse d'un algorithme d'intelligence en essaim pour le fourragement. In La Revue d'intelligence Artificielle, 22 (6) : 791-816, 2008.

Analyse d'un algorithme d'intelligence en essaim pour le fourragement

Amine Boumaza and Bruno Scherrer *
{amine.boumaza, bruno.scherrer}@loria.fr

Résumé

Nous présentons un algorithme d'intelligence en essaim pour résoudre le problème du fourragement dans le cas discret. Nous illustrons l'algorithme proposé à l'aide de simulations et nous faisons une analyse complète de convergence : nous démontrons que la population d'agents simples qui compose l'essaim calcule la solution d'un problème de contrôle optimal et que sa dynamique converge. Nous étudions le taux de convergence de l'algorithme en fonction de la taille de la population et donnons des arguments expérimentaux et théoriques qui suggèrent que ce taux de convergence est superlinéaire en fonction du nombre d'agents. En outre, nous expliquons comment ce modèle peut être étendu au cas où l'espace est continu et pour résoudre des problèmes de contrôle optimal en général. Nous argumentons qu'une telle approche peut être appliquée à tout problème qui implique le calcul du point fixe d'une contraction. Ceci permet de concevoir une grande classe d'algorithmes d'intelligence en essaim bien compris formellement.

1 Introduction

L'*intelligence en essaim* [2, 5] a été introduite dans les années 1990 comme une nouvelle approche, souvent bio-inspirée, pour la résolution de problèmes. La source d'inspiration de cette approche vient de l'observation du comportement des insectes sociaux : une population d'agents extrêmement simples, interagissant et communiquant indirectement à travers leur environnement, constitue un algorithme *massivement parallèle* pour résoudre une tâche donnée (par exemple le fourragement, l'attroupement ou *flocking*, la division de tâches, la capture de proies...). À propos d'une autre approche bio-inspirée pour la résolution *massivement parallèle* de problèmes, les réseaux de neurones, Kohonen a écrit [13][1] :

> *Une des tâches fondamentales de cette nouvelle science des réseaux de neurones est de démontrer par des analyses mathématiques, des simulations sur ordinateur, et même à travers des systèmes sensoriels ou de commande artificiels qu'il est possible de mettre en application des fonctions de traitement de l'information massivement parallèles en utilisant des composants dont les principes ne sont pas mystérieux mais déjà familiers de l'informatique, des sciences de la communication, et de l'automatique. Il n'y a rien dans le domaine des réseaux de neurones qui ne soit déjà connu ou suggéré, au moins en principe, par d'autres théories existantes.*

La motivation de cet article est d'exprimer un propos similaire au sujet de l'intelligence en essaim. Nous présentons un algorithme d'intelligence en essaim et montrons que des résultats des domaines du calcul parallèle et des processus stochastiques permettent de le comprendre et de l'analyser en profondeur.

*Equipe MAIA, LORIA Campus Scientifique BP 239, 54506 Vandœuvre-lès-Nancy CEDEX, France

[1] Dans le texte original : « One of the fundamental tasks of this new "neural networks" science is to demonstrate by mathematical analyses, computer simulations, and even working artificial sensory and control systems that it is possible to implement massively parallel information-processing functions using components the principles of which are not mysterious but already familiar from computer technology, communication science, and control engineering. There is nothing in the "neural network" area which were not known, in principle at least, from constructs already in use or earlier suggested. »

L'approche classique pour la résolution de problèmes et l'intelligence en essaim peuvent être vues comme deux manières alternatives de faire de la résolution de problèmes. En quoi ces approches sont-elles si différentes ? Comme évoqué par Sutton [22][2]), on pourrait :

caractériser la différence comme étant liée au dilemme familier entre obtenir un résultat clair, rigoureux d'un côté, et explorer les systèmes les plus intéressants, puissants de l'autre.

De manière schématique, la recherche sur l'intelligence en essaim s'appuie généralement sur une méthodologie expérimentale et étudie des problèmes réels difficiles tandis que la résolution de problèmes par l'ingénierie classique s'intéresse souvent à des preuves théoriques de convergence pour des problèmes « jouets ». La question qui nous intéresse ici n'est pas de juger ces approches (elles sont clairement complémentaires) mais d'essayer de tisser des liens entre les deux.

Dans la littérature, il existe relativement peu de travaux qui tentent de faire ce lien. Une exception intéressante est l'optimisation par colonies de fourmis pour les problèmes combinatoires (ACO) – probablement la méta heuristique la plus connue en intelligence en essaim – qui connaît de nombreuses analyses théoriques. Ces travaux sur l'ACO ont été récemment recensés dans [10]. Les auteurs y présentent des théorèmes pour la « convergence en valeurs » (l'ACO est garantie de trouver une solution optimale en un temps fini avec une probabilité qui peut être arbitrairement proche de 1), et pour la « convergence en solution » (avec un paramètre décroissant d'exploration bien conçu, l'ACO converge vers une solution optimale presque sûrement). Les auteurs font aussi le lien entre l'ACO et des approches « standards » comme la descente de gradient stochastique ou l'entropie croisée. Nous renvoyons le lecteur intéressé à [10] pour plus de détails.

Dans cet article, nous revenons à l'inspiration originale des algorithmes fourmis, où une population d'agents simples résout collectivement le problème du fourragement en utilisant un signal de phéromone comme moyen de communication indirect. L'algorithme que nous présentons ici est proche de plusieurs travaux de la littérature (Deneubourg *et al.*, 1989, 1990 ; Resnik, 1994 ; Panait *et al.*, 2004a, 2004b ; Simonin, 2005). Ces travaux s'inspirent de la modélisation du comportement de colonies de fourmis afin de proposer des algorithmes d'intelligence en essaim pour le fourragement. A l'exception de [21] au sujet duquel l'auteur a récemment fait une preuve de convergence [20] – nous verrons d'ailleurs plus loin que cette preuve est un cas très particulier de celle que nous donnons ici – la méthodologie employée dans ces travaux est purement expérimentale : de nombreuses simulations montrent les qualités intéressantes des algorithmes mais aucun résultat théorique n'étaye ces observations.

Dans cet article, nous décrivons un algorithme d'intelligence en essaim dont on peut prouver qu'il résout le problème du fourragement. Nous montrons formellement sa convergence et nous donnons une analyse théorique du taux de convergence en fonction du nombre d'agents. Cette analyse se fonde sur la théorie du contrôle optimal, le calcul parallèle et la théorie des graphes. A notre connaissance, c'est la première fois qu'un algorithme d'intelligence en essaim appliqué au problème du fourragement est présenté avec une analyse en termes de convergence *et* de taux de convergence.

Le plan de la suite de cet article est le suivant. Dans la section 2 nous décrivons notre algorithme et observons à l'aide de simulations que la dynamique de la population converge et qu'il y a un taux de convergence superlinéaire. La section 3 donne une preuve de la convergence et donne des éclaircissements sur l'influence des paramètres de l'algorithme présenté. Dans la section 4 nous analysons le taux de convergence et donnons des arguments qui expliquent pourquoi on observe un taux de convergence superlinéaire. La section 5 montre comment on peut étendre l'algorithme au cas du fourragement dans un espace continu et explique dans quelle mesure et avec quelles limites on peut adapter l'analyse des sections 3 et 4. Finalement, nous discutons (section 6) de la portée de notre algorithme et des relations étroites qu'il entretient avec certains des algorithmes que nous avons cités plus haut [17, 21, 20].

2 Un algorithme d'intelligence en essaim pour le fourragement

Dans cette section, nous décrivons un algorithme d'intelligence en essaim qui permet de résoudre le problème du fourragement. Nous exhibons des simulations qui montrent que les agents finissent par transporter la nourriture depuis une source vers leur « nid », et nous présentons des données

[2] Bien que l'auteur parle ici de la relation entre l'apprentissage artificiel « moderne » et le contrôle intelligent « classique », nous pensons que son propos correspond également à la relation entre l'intelligence en essaim et la résolution de problèmes. Dans le texte original : « [We could] characterize the split as having to do with the familiar dilemma of choo sing between obtaining clear, rigorous results on the one hand, and exploring the most interesting, powerful systems on the other. »

expérimentales qui montrent que l'efficacité de cet algorithme distribué est superlinéaire en fonction du nombre d'agents.

2.1 Description de l'algorithme

Considérons un ensemble de m agents qui évoluent sur les cellules d'une grille[3] 2D (l'environnement) et sur lesquelles ils mettent à jour des traces, que nous appellerons dans la suite « phéromones » par analogie aux phéromones réelles utilisées par les fourmis. L'environnement se compose de quatre types de cellules : une cellule *nid*, une *source de nourriture*, un certain nombre de cellules *indésirables* et le reste des cellules sont *libres*. Chaque cellule s de la grille stocke deux valeurs réelles, correspondant à deux types de phéromone : $\Phi_f(s)$ est la phéromone de nourriture et $\Phi_n(s)$ la phéromone du nid.

Un agent ne peut porter qu'une seule unité de nourriture et peut être dans deux états possibles : « chargé » ou « non chargé ». En évoluant dans l'environnement, l'état des agents change selon les règles naturelles suivantes : si un agent arrive à la cellule source de nourriture, son état devient « chargé », et s'il arrive au nid son état change à « non chargé ». Quand un agent dans l'état « chargé » arrive au nid, un compteur d'« unités de nourriture » est incrémenté. Comme nous le verrons par la suite, ce compteur servira comme mesure de performance globale pour la population.

Nous décrivons maintenant le fonctionnement de notre algorithme. Au départ,

- le compteur d'« unités de nourriture » est initialisé à zéro,
- les m agents sont initialisés arbitrairement (e.g. ils peuvent être tous au nid, ou initialisés aléatoirement uniformément sur la grille),
- tous les agents sont dans l'état « non chargé »,
- les phéromones sont initialisées arbitrairement (e.g. 0, aléatoire, etc).

A chaque pas de temps, chaque agent fait deux choses :

1. Il met à jour localement les taux de phéromones $\Phi_f(s)$ et $\Phi_n(s)$ de sa cellule courante s en utilisant ceux des quatre cellules voisines c'est-à-dire en utilisant uniquement une information locale. La mise à jour des taux de phéromones requiert uniquement la connaissance du maximum et de la moyenne des valeurs voisines. On note $\mathcal{V}(s)$ l'ensemble des voisins de s, et on définit :

$$\max_i(s) \equiv \max_{s' \in \mathcal{V}(s)} \Phi_i(s')$$

et

$$\text{avg}_i(s) \equiv \frac{1}{4} \sum_{s' \in \mathcal{V}(s)} \Phi_i(s')$$

où l'indice $i \in \{f, n\}$ spécifie laquelle des deux phéromones est considérée. La mise à jour se fait alors selon les calculs simples suivants :

$$\Phi_f(s) \leftarrow \begin{cases} -1 & \text{si } s \text{ est une cellule } \textit{indésirable} \\ 1 & \text{si } s \text{ est la } \textit{source de nourriture} \\ \beta\left(\alpha \text{max}_f(s) + (1-\alpha)\text{avg}_f(s)\right) & \text{sinon} \end{cases}$$

$$\Phi_n(s) \leftarrow \begin{cases} -1 & \text{si } s \text{ est une cellule } \textit{indésirable} \\ 1 & \text{si } s \text{ est le } \textit{nid} \\ \beta\left(\alpha \text{max}_n(s) + (1-\alpha)\text{avg}_n(s)\right) & \text{sinon} \end{cases}$$

avec $0 \leq \alpha \leq 1$ et $0 \leq \beta \leq 1$. Comme on le verra plus loin, on considère de plus que $\beta < 1$ quand $\alpha = 1$.

2. Il se déplace jusqu'à l'une de ses cellules voisines :
   - avec une probabilité ϵ ($0 \leq \epsilon \leq 1$), qu'on appellera le taux *d'exploration*, il avance sur une cellule choisie aléatoirement uniformément sur ses quatre voisines,
   - avec une probabilité $1 - \epsilon$, il avance sur la cellule avec la plus grande valeur de phéromone de Φ_f ou Φ_n en fonction de son état : s'il n'est « non chargé » alors il utilise Φ_f et s'il est « chargé » il utilise Φ_n.

Le paramètre β peut être considéré comme un paramètre d'*évaporation* et prend typiquement des valeurs proches de 1. Le paramètre α, que nous appelons paramètre de *bruit* pour des raisons qui sont explicitées plus tard, doit être fixé proche de 0 ou proche de 1. Deux cas particuliers correspondent

[3] Comme le lecteur comprendra dans la suite, des graphes plus complexes peuvent être utilisés, toutefois nous nous restreignons au cas d'une grille 2D pour simplifier la présentation.

aux choix de paramètres ($\alpha = 1$, $0 < \beta < 1$) et ($\alpha = 0$, $\beta = 1$). Dans le premier, l'équation de mise à jour des phéromones devient :

$$\Phi_i(s) \leftarrow \beta \max_i(s) \tag{1}$$

et dans le deuxième, l'équation devient :

$$\Phi_i(s) \leftarrow \text{avg}_i(s) \tag{2}$$

qui est une simple mise à jour linéaire. Comme il apparaîtra dans l'analyse, la question selon laquelle l'activité de l'ensemble des agents (mises à jour des phéromones et mouvements dans l'environnement) est faite de manière synchrone ou asynchrone n'a pas d'importance.

En résumé, pour identifier précisément une instance de l'algorithme (et pour que le lecteur ait la possibilité de reproduire les expériences décrites plus loin), les données suivantes doivent être spécifiées :

- l'environnement : l'ensemble des cellules et leur type (nid, nourriture, indésirable, libre),
- le nombre m d'agents,
- la manière d'initialiser les positions des agents et les valeurs des phéromones Φ_f et Φ_n sur l'environnement,
- le paramètre de bruit α, le paramètre d'évaporation β et le taux d'exploration ϵ.

L'algorithme que nous venons de décrire est constitué d'un ensemble d'agents extrêmement simples qui communiquent indirectement à travers l'environnement. Contrairement à d'autres algorithmes inspirés des fourmis, les phéromones ne nécessitent pas d'être évaporées en *chaque* cellule de l'environnement à *chaque* instant ; une forme d'évaporation a lieu à travers le paramètre β lors de la mise à jour locale des phéromones par un agent. De plus, à la différence d'algorithmes classiques qui utilisent souvent une seule phéromone, le nôtre en utilise deux : suivre Φ_f amène à la source de nourriture alors que suivre Φ_n amène au nid. Cette particularité n'est pas spécifique à notre algorithme : d'autres travaux de la littérature utilisent plusieurs phéromones [17, 24, 12]. D'une manière générale, les travaux qui utilisent une seule phéromone mettent souvent en œuvre des mécanismes *ad hoc* qui aident les agents à retrouver le chemin du nid. L'utilisation d'une deuxième phéromone compense le fait que nos agents ne mémorisent pas le chemin de retour au nid.

2.2 Simulations

Dans cette section nous illustrons le comportement de l'algorithme que nous venons de présenter à l'aide de simulations. Nous observons que l'activité des agents converge dans un certain sens, que nous précisons. Nous décrivons ensuite le protocole expérimental qui nous permet de mesurer le taux de convergence de l'algorithme et nous présentons l'évolution de ce taux en fonction du nombre d'agents. Nous commentons les résultats expérimentaux obtenus brièvement dans la mesure où une analyse est développée dans les sections 3 et 4.

Considérons une exécution typique de notre modèle. L'environnement est présenté à la figure 1. La taille de la population est $m = 256$ et les agents sont initialisés au nid. Enfin, les paramètres sont : $\epsilon = 0,75$, $\alpha = 0,99$, et $\beta = 0,9999$. La figure 1 montre l'évolution des agents qui, au cours du temps, se déplacent dans l'environnement à la recherche de nourriture. La source de nourriture est découverte par hasard par quelques agents isolés (itération 7 000). Très rapidement (entre les itérations 7 000 et 8 000), une grande proportion des agents se met à emprunter ce chemin. Plus tard, des agents trouvent par hasard un chemin alternatif (vers l'itération 26 000) ; ceux-ci sont rapidement rejoints par l'ensemble de la population (itération 27 000). Il n'y aura après plus de changement significatif : les agents feront des allers-retours en empruntant ce chemin entre le nid et la source.

Toutes choses étant égales par ailleurs, si on change la valeur du paramètre α, on peut observer des chemins de formes différentes, voire dans certains cas aucun chemin. La figure 2 illustre la distribution asymptotique des agents pour des valeurs de α différentes : il existe des chemins sauf pour le cas où $\alpha = 0,2$.

Supposons qu'un chemin émerge entre le nid et la source de nourriture. Si l'on peut observer visuellement ce chemin, il est plus intéressant d'en faire une mesure objective. Ceci peut être fait avec le compteur d'« unités de nourriture » introduit précédemment. En effet, on peut tracer la courbe de la quantité accumulée de nourriture rapportée au nid au cours du temps (voir la figure 3). Au bout d'un certain temps, on observe que la quantité accumulée de nourriture apportée au nid croît linéairement : ceci suggère que le comportement de fourragement des agents a convergé. Dans la section 3, nous fournirons des arguments théoriques qui caractérisent précisément cette convergence, et nous expliquerons en particulier pourquoi il n'y a pas de chemin pour certaines valeurs de α. La courbe de l'évolution de la quantité de nourriture montre expérimentalement, pour une simulation,

qu'il y a convergence ; elle peut de plus être exploitée pour mesurer un taux de convergence. Pour ce faire, nous procédons de la manière suivante : nous fixons un temps d'évolution maximum (t_{max}) suffisamment grand et nous effectuons une régression linéaire sur une portion supérieure de la courbe de quantité de nourriture (la partie linéaire de la courbe). La taille de la portion choisie est liée à un paramètre (sp), que nous fixons typiquement entre $0,25$ et $0,5$, ce qui prendre en compte entre le quart et la moitié supérieure de la courbe pour la régression linéaire.

Nous définissons alors le temps de convergence comme l'intersection de la droite de la régression et l'axe du temps[4] (voir la figure 3). Le taux de convergence est alors naturellement défini comme l'inverse du temps de convergence. Afin de déterminer l'influence de la taille de population sur le taux de convergence, nous mesurons celui-ci pour différentes valeurs de taille de population m. Une simulation consiste ainsi à lancer l'algorithme pour différentes valeurs de m en gardant fixes l'environnement et l'ensemble des autres paramètres (α, β, ϵ, t_{max} et sp). A l'issue de chaque simulation nous obtenons les valeurs du taux de convergence en fonction de m. Étant donnée la nature stochastique des simulations, nous reproduisons les simulations un certain nombre de fois de sorte à mesurer des statistiques sur les taux de convergence.

La figure 4 présente une courbe typique montrant le taux de convergence en fonction de la taille de la population. Pour ces expériences, nous avons repris l'environnement de la figure 2 avec les paramètres $\alpha = 0,9$, $\beta = 0,999999$, $\epsilon = 0,7$, $t_{max} = 100\ 000$, $sp = 0,5$ et les agents ont été initialisés uniformément sur l'environnement. Les statistiques (valeur médiane et écart absolu moyen) sont calculées à partir de 7 000 simulations. La figure 5 illustre des expériences similaires sur des environnements différents (avec 5 000 simulations).

Ces courbes illustrent expérimentalement le fait que lorsque le nombre d'agents croît linéairement, le taux de convergence croît de manière superlinéaire. En d'autres termes, si le nombre d'agents est doublé le taux de convergence croît d'un facteur supérieur à deux. Cette observation sera analysée dans la partie 4 : nous fournirons des arguments qui expliquent pourquoi le taux de convergence peut croître de manière superlinéaire.

3 Analyse de la convergence

Dans cette section, nous analysons l'algorithme d'agents décrit dans la section précédente, et nous prouvons sa convergence. Pour comprendre ce que fait l'algorithme et pourquoi il y a émergence de chemins, nous allons en premier lieu faire un petit détour par la théorie du contrôle optimal stochastique à temps discret, et plus particulièrement par le cadre des processus décisionnels de Markov (PDM). Un PDM est un processus stochastique contrôlé qui vérifie la propriété de Markov avec des récompenses (valeurs numériques) affectées à des états. Plus formellement, un PDM est un quadruplet $\langle S, A, T, R\rangle$ où S est *l'espace des états*, A est *l'espace des actions*, T est la *fonction de transition* et R est la *fonction de récompense*. La fonction T est la distribution de probabilités de transition d'états conditionnellement aux actions. Pour tous états s et s' et pour toute action a, $T(s, a, s')$ est la probabilité de passer de l'état s à l'état s', si l'action a est effectuée :

$$T(s, a, s') \stackrel{def}{=} \Pr(s_{t+1} = s' | s_t = s, a_t = a). \tag{3}$$

La fonction $R(s) \in \mathbb{R}$ est la récompense instantanée d'être dans l'état s. Dans le cadre des PDM, le *problème du contrôle optimal* consiste à trouver une *politique*, c'est-à-dire une application $\pi : S \to A$ des états vers les actions, qui maximise l'espérance de la récompense accumulée à long terme, aussi appelée fonction de valeur de la politique π :

$$V^{\pi}(s) = \mathrm{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t . R(s_t) | s_0 = s\right]. \tag{4}$$

où la notation E_{π} désigne l'espérance sur les trajectoires aléatoires induites par le fait qu'on suit la politique π[5]. Nous considérons ainsi un horizon temporel infini. Les récompenses futures sont exponentiellement actualisées par un facteur $\gamma \in [0, 1]$.

Pour tout PDM, il a été démontré [18] qu'il existe une unique fonction de valeur optimale V^* qui est l'unique point fixe de l'opérateur B suivant[6] :

$$\forall W \in \mathbb{R}^S, [B.W](s) = \max_a \left(R(s, a) + \gamma . \sum_{s'} T(s, a, s') . W(s') \right). \tag{5}$$

[4] Ce calcul est analogue à celui de la constante de temps d'une capacité en électricité.

[5] En chaque état s on choisit l'action $a = \pi(s)$. Le processus est alors une chaîne de Markov sur S.

[6] Cet opérateur est souvent appelé opérateur de Bellman.

Une fois la fonction de valeur V^* calculée, une politique optimale π^* peut immédiatement être dérivée :

$$\pi^*(s) \in \arg\max_a \left(R(s,a) + \gamma . \sum_{s'} T(s,a,s').V(s') \right). \quad (6)$$

L'opérateur de Bellman, B ci-dessus, a un unique point fixe car c'est une *contraction* de facteur de contraction au plus γ : i.e. pour toute paire de fonctions réelles U, U' sur S,

$$\|BU - BU'\| \leq \gamma \|U - U'\|,$$

où $\|.\|$ est la norme infinie sur S : $\|U\| = \max_s |U(s)|$. Une approche standard pour résoudre le problème du contrôle optimal utilise une procédure itérative séquentielle qui s'appelle *Itérations sur les valeurs* [18]. Elle consiste à initialiser arbitrairement une fonction V^0 et à itérer le processus suivant :

$$\text{Pour tout état } s \in S, \text{ faire : } V^{t+1}(s) \leftarrow [BV^t](s).$$

Grâce à la propriété de contraction, cette séquence est garantie de converger asymptotiquement vers la fonction de valeur optimale V^*, depuis laquelle on peut déduire la politique optimale π^* (cf équation 6). De plus, une telle séquence a un taux de convergence linéaire au plus γ :

$$\|V^{t+1} - V^*\| \leq \gamma . \|V^t - V^*\| \leq \gamma^{t+1} . \|V^0 - V^*\| \quad (7)$$

Dans un ouvrage sur le parallélisme [3], les auteurs expliquent qu'une version asynchrone d'*Itérations sur les valeurs* :

$$\text{Choisir (aléatoirement) un état } s \in S \text{ et faire : } V(s) \leftarrow [BV](s), \quad (8)$$

convergera aussi vers le point fixe V^*, *tant que tous les états continuent d'être choisis*. Dans le cas asynchrone, on peut réécrire l'équation 7 sous la forme :

$$\|V^{k_t} - V^*\| \leq \gamma \, \|V^{k_{t-1}} - V^*\| \leq \gamma^t \|V^0 - V^*\| \, , \quad (9)$$

où $k_0, k_1 \ldots$ est une suite croissante telle que $k_0 = 0$ et que tous les éléments de S (tous les états) sont mis à jour au moins une fois entre les instants k_t et $k_{t+1} - 1$ pour tout t (voir [4] p. 27). Chaque fois que tous les états ont été mis à jour, on sait que V s'approche de V^* avec un taux linéaire au plus γ. La preuve de ce résultat repose sur la propriété de contraction. Bien que le but dans [3] était de proposer des implantations parallèles efficaces sur des calculateurs parallèles, le nôtre est légèrement différent : nous allons établir un lien entre ce que la population d'agents calcule dans l'algorithme que nous avons présenté et des problèmes de contrôle optimal. En effet, nous allons montrer que les traces de phéromones Φ_f et Φ_n correspondent chacune à la fonction de valeur d'un problème de contrôle. Ceci constitue notre premier résultat :

Proposition 1 *Considérons l'algorithme décrit dans la section 2.1. Si le taux d'exploration ϵ est strictement positif, alors les phéromones Φ_f (resp. Φ_n) convergent asymptotiquement vers la fonction de valeur optimale du PDM $M_f = \langle S, A, T_f, R_f \rangle$ (resp. $M_n = \langle S, A, T_n, R_n \rangle$) avec le facteur d'actualisation β où :*

- *S est l'ensemble des points de la grille auquel on ajoute un « état final ».*
- *A est l'ensemble des quatre mouvements cardinaux (nord, sud, est, ouest).*
- *La transition T_f (resp. T_n) est définie comme suit :*
  1. *quand on est sur une cellule libre ou sur le nid (resp. sur une cellule libre ou sur la source de nourriture) et qu'on choisit une des quatre directions $a \in A$, la probabilité d'aller dans la direction a est $\alpha + \frac{1-\alpha}{4} = \frac{3\alpha+1}{4}$, tandis que la probabilité d'aller dans l'une des trois autres directions est $\frac{1-\alpha}{4}$.*
  2. *à partir de tout autre état, i.e. cellules indésirables, source de nourriture et l'« état final » (resp. cellules indésirables, le nid et l'« état final ») il y a, quelle que soit l'action choisie, une probabilité 1 d'arriver à l'« état final », qui est un état absorbant.*
- *La récompense R_f (resp. R_n) est définie comme suit : pour tous les états s correspondant à une cellule libre ou le nid (resp. une cellule libre ou la source de nourriture), la récompense est 0. Pour l'état correspondant à la source (resp. au nid), la récompense est 1. La récompense est -1 pour les états correspondant aux cellules indésirables et 0 pour l'« état final ».*

La preuve de ce résultat consiste simplement à vérifier que l'algorithme est une version asynchrone de l'algorithme *Itérations sur les valeurs* pour les problèmes que nous venons de décrire : concrètement il faut vérifier, pour toutes les cellules, que les mises à jour des traces Φ_f et Φ_n sont identiques à

celles que ferait l'opérateur de Bellman (équation 8). Ceci n'est qu'une simple question de réécriture. Par exemple, pour la mise à jour de $\Phi_i(s)$ quand s est une cellule libre nous avons :

$$\begin{aligned}
& \Phi_i(s) \leftarrow \beta\left(\alpha \mathrm{max}_i(s) + (1-\alpha)\mathrm{avg}_i(s)\right) \\
\Leftrightarrow \quad & \Phi_i(s) \leftarrow \beta\left(\alpha \max_{s'\in\mathcal{V}(s)} \Phi_i(s') + \frac{1-\alpha}{4} \sum_{s'\in\mathcal{V}(s)} \Phi_i(s')\right) \qquad (10) \\
\Leftrightarrow \quad & \Phi_i(s) \leftarrow \beta \max_{s'\in\mathcal{V}(s)} \left(\alpha\Phi_i(s') + \frac{1-\alpha}{4} \sum_{s''\in\mathcal{V}(s)} \Phi_i(s'')\right) \\
\Leftrightarrow \quad & \Phi_i(s) \leftarrow \beta \max_{s'\in\mathcal{V}(s)} \left(\left(\alpha + \frac{1-\alpha}{4}\right)\Phi_i(s') \right. \\
& \qquad \left. + \frac{1-\alpha}{4} \sum_{s''\in\mathcal{V}(s)\setminus\{s'\}} \Phi_i(s'')\right) \\
\Leftrightarrow \quad & \Phi_i(s) \leftarrow \beta \max_{a\in A}\left(\sum_{s'\in S} T_i(s,a,s')\Phi_i(s)\right) \qquad (11)
\end{aligned}$$

Nous laissons le soin au lecteur intéressé de vérifier l'équivalence dans les autres cas. Enfin la condition $\epsilon > 0$ assure que tous les états sont visités et mis à jour continuellement, ce qui implique la convergence de cette version asynchrone vers la fonction de valeur optimale.

Examinons de plus près ce que cela veut dire et particulièrement pourquoi nous observons l'émergence de chemins entre le nid et la source de nourriture lors des simulations. Résoudre M_f (resp. M_n) signifie trouver la politique qui générera des trajectoires qui évitent (en moyenne) les cellules indésirables pour lesquelles la récompense est -1 et qui atteignent la source de nourriture (resp. le nid) pour lesquelles la récompense est 1 ; à cause du facteur d'actualisation $\beta < 1$, l'optimisation tente aussi de minimiser le temps pour arriver à la source de nourriture (resp. au nid). En d'autres termes, M_f (resp. M_n) est une formulation naturelle du problème de contrôle suivant : « aller à la source de nourriture (resp. au nid) tout en évitant les cellules indésirables » en supposant qu'il y a du bruit dans les transitions. Comme on le voit dans la définition de T_f (resp. T_n), le niveau de bruit est lié au paramètre α : c'est pour cette raison que nous l'avons appelé paramètre de bruit.

En chaque état, l'action optimale est celle pour laquelle le max est atteint dans l'équation 11 ci-dessus, et il est aisé de voir que cette action optimale est celle qui mènera à la cellule pour laquelle le max est atteint dans l'équation 10 : c'est la cellule avec la plus grande valeur de phéromone. Les valeurs des phéromones convergent asymptotiquement vers les fonctions de valeur optimales Φ_f et Φ_n. Par conséquent, les mouvements des agents qui sont (c.f. la description de l'algorithme dans la section 2.1) un mélange de mouvements aléatoires (avec un poids ϵ) et des mouvements dans la direction la plus concentrée en phéromone (avec un poids $1-\epsilon$), convergent vers un mélange d'actions aléatoires et d'actions optimales. Plus ϵ est petit, plus net sera le chemin observé quand les phéromones auront convergé (à la limite $\epsilon = 0$ tous les agents suivent la politique optimale). Cependant, plus ϵ est petit, plus le temps que mettront les agents à visiter toutes les cellules sera grand, et par conséquent la convergence des phéromones sera ralentie. Ce compromis est typique dans les problèmes de contrôle optimal et est connu comme le dilemme exploration-exploitation (voir par exemple [23] pour une discussion approfondie).

Lors de la description de l'algorithme dans la section 2.1, nous avons écrit à propos des paramètres α et β qu'ils devaient satisfaire $0 \leq \alpha \leq 1$ et $0 \leq \beta \leq 1$ avec la condition que $\beta < 1$ si $\alpha = 1$. Nous pouvons à présent expliquer cette condition du point de vue du contrôle optimal : choisir le facteur d'actualisation β strictement inférieur à 1 dans un PDM garantit que la performance à horizon temporel infini (équation 4) est bornée et que l'opérateur de Bellman est une contraction. Dès lors qu'il y a du bruit dans le modèle de transition des PDM M_f et M_n (*i.e.* dès que $\alpha < 1$) il y a une probabilité 1 d'atteindre en un temps fini l'« état final » absorbant (qui a une récompense 0). Ceci est suffisant pour garantir que le critère de performance est borné et que l'opérateur de Bellman est une contraction. Toutefois, dans le cas purement déterministe ($\alpha = 1$) le facteur d'actualisation β doit être strictement inférieur à 1.

Nous pouvons enfin expliquer l'influence du paramètre de bruit α. Quand α est égal à 1 (et la mise à jour est l'équation 1), il n'y a pas de bruit dans le modèle de transition et les politiques optimales correspondent au chemin le plus court (par rapport à la distance de *Manhattan*, voir la

figure 2, cas $\alpha = 1$) entre le nid et la source de nourriture. Quand on réduit α, le niveau de bruit augmente et les politiques optimales induisent des trajectoires plus arrondies (elles tentent de rester loin des états indésirables). Quand α continue à décroître, il y a tellement de bruit que la probabilité d'atteindre un état indésirable en essayant d'arriver à la source de nourriture (ou au nid) devient trop grande pour n'importe quelle politique. Par conséquent, le comportement optimal consiste à rester loin des états indésirables sans tenter d'atteindre la source de nourriture (ou le nid) : dans ce cas il est préférable d'avoir une récompense nulle (état libres) qu'une récompense -1 (états indésirables). Ceci explique pourquoi il n'y a pas de chemin sur la figure 2 pour $\alpha = 0,6$. Quand α décroît encore plus et se rapproche de 0, il se produit un phénomène à première vue étrange : des chemins apparaissent de nouveau. Nous pouvons expliquer ceci de deux manières : 1) quand le bruit devient trop grand, l'influence des actions diminue et le contrôleur optimal ne peut même plus rester éloigné des états indésirables ; alors, tel un kamikaze qui sait qu'il va mourir, la stratégie optimale consiste à nouveau à essayer d'atteindre la source de nourriture (ou le nid). 2) À la limite où $\alpha = 0$ (la mise à jour est l'équation 2), les taux de phéromones, qui vérifient une équation du type $\Phi_i(s) = \text{avg}^i(\mathcal{V}(s))$ sont équivalentes à la version discrète d'une fonction harmonique, et il est connu que monter le gradient d'une fonction harmonique peut être utilisé pour la navigation [7]. Une discussion approfondie de l'influence du paramètre α est développée dans [6].

4 Analyse du taux de convergence

Nous avons vu que l'algorithme proposé converge en expliquant qu'il est une instance de calcul asynchrone de points fixes de deux contractions : les traces de phéromones qui résultent des mises à jour locales faites par les agents se stabilisent vers les fonctions de valeur optimales de deux problèmes de contrôle, ce qui guide les agents entre le nid et la source de nourriture. Le but de cette partie est l'étude du *taux de convergence* de ce processus en fonction de la taille de la population. Pour cette analyse, nous allons décrire quelques objets et quelques résultats concernant les chaînes de Markov sur les graphes. Nous verrons qu'ils permettent de comprendre l'observation (faite dans la section 2.2) que le taux de convergence est superlinéaire : doubler le nombre d'agents peut accélérer le processus d'un facteur supérieur à deux.

Pour l'analyse du taux de convergence, nous revenons à l'équation 9 qui décrit, en général, le taux de convergence du calcul asynchrone du point fixe d'une contraction. Par souci de clarté nous réécrivons cette équation :

$$\|V^{k_t} - V^*\| \leq \gamma \|V^{k_{t-1}} - V^*\| \leq \gamma^t \|V^0 - V^*\|, \tag{12}$$

Nous rappelons au lecteur que $k_0, k_1 \ldots$ est une suite croissante telle que $k_0 = 0$ et que tous les éléments de s (tous les états) sont mis à jour au moins une fois entre k_t et $k_{t+1} - 1$ pour tout t. On peut constater que le taux de convergence est lié à la variable aléatoire k_t : plus k_t croît lentement, plus le processus converge vite. Comme nous nous intéressons au taux de convergence en fonction de la taille m de la population, nous allons expliciter cette dépendance en écrivant k_t^m. Il nous faut donc étudier $k_{t+1}^m - k_t^m$.

La bonne nouvelle au sujet de $k_{t+1}^m - k_t^m$ est que c'est un objet connu de la littérature probabiliste. Considérons l'espérance $E[k_{t+1}^1 - k_t^1]$ pour le cas $m = 1$: c'est le temps moyen nécessaire à un agent pour visiter toutes les cellules. Plus formellement, si nous considérons l'environnement comme un graphe G (il y a un nœud pour chaque cellule et une connexion entre deux nœuds voisins), c'est le temps moyen nécessaire à une chaîne de Markov (qui décrit les positions de cet agent) pour visiter tout les nœuds du graphe G, et cette quantité est appelée le *temps de couverture du graphe G par la chaîne de Markov* [1]. Dans le cas général $m > 1$, $E[k_{t+1}^m - k_t^m]$ est le temps moyen pour que plusieurs marches aléatoires parallèles visitent tous les nœuds du graphe G, et cette quantité s'appelle le *temps de couverture du graphe G par m chaînes de Markov parallèles* [1].

La mauvaise nouvelle au sujet de $k_{t+1}^m - k_t^m$ est que dans le cas *général* il est très difficile de calculer le temps de couverture d'un graphe par une chaîne de Markov donnée ayant une distribution initiale donnée : il est calculable en temps exponentiel et la question selon laquelle il pourrait être calculable approximativement en temps polynomial est ouverte [11]. Il est encore plus difficile de calculer le temps de couverture d'un graphe pour plusieurs chaînes de Markov. Pour notre algorithme, le calcul paraît d'autant plus complexe que les chaînes de Markov et la distribution des agents varient en fonction du temps : les probabilités de transitions dépendent des traces de phéromones qui elles mêmes sont mises à jour par les chaînes de Markov. Il y a donc peu d'espoir qu'on puisse estimer *précisément* le taux de convergence de notre algorithme. Une analyse asymptotique générale de la convergence (dans laquelle on considère que a) les phéromones ont convergé et b) les agents

ont atteint une distribution stationnaire) pourrait être poursuivie et constitue un sujet pour des recherches futures.

Néanmoins, en consultant la littérature sur le temps de couverture, nous avons trouvé le résultat intéressant suivant [1] :

Proposition 2 *Sur un graphe régulier*[7] *à n nœuds, considérons m marches aléatoires indépendantes et équilibrées*[8]*, chacune commençant à des nœuds distribués aléatoirement uniformément sur le graphe. Appelons C^m le temps auquel tous les nœuds du graphe ont été atteints par au moins une marche aléatoire*[9]*. Alors à mesure que la taille du graphe $n \to \infty$ et tant que le nombre de marches vérifie $m \geq 6 \log n$, nous avons :*

$$E\left[C^{[m]}\right] \leq \frac{(25 + o(1))n^2 \log^2 n}{m^2}.$$

Le point intéressant est la dépendance en $\frac{1}{m^2}$: ceci implique que (à mesure que $n \to \infty$ et $m \geq 6 \log n$) le temps de couverture est borné par une fonction qui est inversement quadratique en fonction du nombre de marches. Sur des graphes simples comme un cycle de taille n, les auteurs de [1] expliquent que la borne ci-dessus est fine : dans ces cas, le temps de couverture est inversement quadratique en fonction du nombre de marches.

En utilisant la discussion ci-dessus, nous pouvons « traduire » cette proposition en quelque chose ayant un sens dans le cadre de notre algorithme :

Proposition 3 *Considérons l'algorithme décrit dans la section 2.1 sur un environnement torique avec n cellules, avec un taux d'exploration $\epsilon = 1$ et une initialisation des agents uniforme sur la grille. À mesure que la taille de l'environnement $n \to \infty$ et tant que le nombre d'agents vérifie $m \geq 6 \log n$, le temps que mettent les traces de phéromones pour réduire d'un facteur β leur distance à la limite est borné par $\frac{(25+o(1))n^2 \log^2 n}{m^2}$ qui est une fonction inversement quadratique en fonction du nombre m d'agents.*

L'hypothèse de l'environnement torique permet d'avoir un graphe régulier, et $\epsilon = 1$ permet d'avoir des marches aléatoires équilibrées. Notons que l'initialisation uniforme des agents correspond à la distribution stationnaire des marches, de sorte que la distribution des agents est stationnaire au cours du temps. Notre proposition n'est donc qu'un corollaire de la proposition 2. La borne est fine quand l'environnement est un graphe cycle de taille n, c'est-à-dire un long couloir cyclique : dans ce cas, le taux de convergence est approximativement quadratique en fonction du nombre d'agents. Nous avons clairement dans ce cas un taux de convergence superlinéaire. Les expériences que nous avons effectuées suggèrent que la superlinéarité se produit pour d'autres valeurs de ϵ et dans d'autres environnements. Étendre ce résultat à des situations plus générales est l'objet de recherches futures.

5 Extension de l'algorithme au cas continu

Jusqu'ici nous avons considéré des agents évoluant dans un environnement discret. Le but de cette section est de montrer que notre modèle peut être adapté à une tâche de fourragement dans un espace continu, et dans laquelle les agents peuvent se déplacer continûment dans toutes les directions $\theta \in [0, 2\pi]$. Pour ce faire, notre approche sera particulièrement naturelle : nous allons décrire une variante continue du problème de navigation dans le cadre du contrôle optimal à temps continu, expliquer que ce problème continu peut être approché par un algorithme distribué et asynchrone, et en déduire l'algorithme correspondant. Bien que nous l'ayons présenté dans l'ordre inverse, c'est d'ailleurs par une démarche analogue que nous avons construit l'algorithme discret présenté précédemment. Ainsi, cette partie a aussi pour but d'illustrer la méthodologie que nous suivons pour construire des algorithmes d'intelligence en essaim. Pour ne pas compliquer inutilement notre travail, nous nous restreignons ici à la moitié du problème de fourragement : trouver des chemins vers une source de nourriture. L'autre moitié peut être implémentée de la même manière que ce que nous avons fait pour le cas discret : utiliser un état interne pour chaque agent et basculer entre les deux problèmes de contrôle sous-jacents : « aller vers la source de nourriture » / « rentrer au nid ».

Un modèle naturel pour la navigation continue est maintenant décrit. Considérons un système défini au temps t par sa position dans les zones libres d'un environnement, *i.e.* son état $x(t) \in \bar{\Omega}$ (l'espace d'états) où $\bar{\Omega} \subset \mathbb{R}^2$ est la fermeture d'un ouvert Ω et où $\partial\Omega$ est son bord ($\bar{\Omega} = \Omega \cup \partial\Omega$).

[7] Un graphe régulier est un graphe dont les nœuds ont tous le même degré, *i.e.* chaque nœud est connecté au même nombre de nœuds.

[8] Une marche aléatoire équilibrée sur un graphe est une marche aléatoire dont les transitions sont des distributions uniformes sur les voisins.

[9] $E[C^m]$ est donc le temps de couverture du graphe par ces marches aléatoires parallèles.

Les bords de l'environnement sont décomposés en deux ensembles : $\partial\Omega = \mathcal{B} \cup \mathcal{F}$: $\mathcal{B}$ est l'ensemble des « bords indésirables » et $\mathcal{F}$ est l'ensemble des « bords nourriture ». A chaque instant, le système est contrôlé par une commande de direction $\theta(t) \in [0, 2\pi]$ (l'espace de contrôle). Nous considérons que la dynamique de ce système est régie par une équation différentielle stochastique :

$$dx = \overrightarrow{u}(\theta(t))\ dt + \sigma\ dw$$

où w est un mouvement brownien de dimension 2, σ est une constante positive qui correspond au niveau de bruit dans l'environnement, et $\overrightarrow{u}(\theta)$ est un vecteur unitaire de direction $[0, 2\pi]$ (c'est-à-dire que le contrôle est de vitesse constante unitaire). Nous considérons le cas où l'horizon temporel est infini. Pour tout état initial $x(0)$, pour toute loi de contrôle $\theta(.)$ et pour la trajectoire (aléatoire) $x(.)$ qui en résulte, nous notons T le temps pour que la trajectoire $x(.)$ sorte de Ω, avec la convention que $T = \infty$ si la trajectoire reste indéfiniment dans Ω. Nous utilisons une récompense pour évaluer la qualité des trajectoires : en général, il est usuel de définir une récompense tout le long de la trajectoire $r : \Omega \to \mathbb{R}$ et une récompense terminale $R : \partial\Omega \to \mathbb{R}$ pour le moment où la trajectoire atteint le bord. Pour notre problème de navigation, nous posons $r(x) = 0$ ($\forall x \in \Omega$), $R(x) = -1$ sur les « bords indésirables » ($\forall x \in \mathcal{B}$) et $R(x) = 1$ sur les « bords nourriture » ($\forall x \in \mathcal{F}$). De manière similaire au cas discret, nous introduisons une fonction de valeur optimale qui correspond à la quantité maximale de renforcement sur les trajectoires avec un facteur d'actualisation ($\gamma \in [0, 1]$) :

$$\begin{aligned} J^*(x) &= \max_{\theta(.)} E_{\theta(.)}\left[\int_0^T \gamma^t r(x(t))\ dt + \gamma^T R(x(T))\right] \\ &= \max_{\theta(.)} E_{\theta(.)}\left[\gamma^T R(x(T))\right] \end{aligned}$$

où $E_{\theta(.)}$ est l'espérance sur les trajectoires induites par la loi de contrôle $\theta(.)$. Le terme intégral disparaît entre la première et la deuxième ligne car $r(\cdot) = 0$. Vu que la fonction $t \mapsto \gamma^t$ est décroissante et étant donnée notre définition de la récompense terminale (-1 sur les « bords indésirables » et 1 sur les « bords nourriture »), maximiser cette quantité revient à la fois à 1) maximiser le temps pour toucher un « bord indésirables » et 2) minimiser le temps pour atteindre un « bord nourriture ».

La théorie du contrôle optimal permet de montrer que la fonction de valeur ci-dessus vérifie une équation aux dérivées partielles, l'équation de Hamilton-Jacobi-Bellman :

$$J^*(x)\ \ln(\gamma)\ + \max_{\theta\in(0,2\pi)} \{\nabla J^*(x)\cdot\overrightarrow{u}(\theta)\} + \frac{\sigma^2}{2}\,\Delta J^*(x) = 0 \tag{13}$$

pour $x \in \Omega$ avec les conditions au bord $\forall x \in \partial\Omega, J^*(x) = R(x)$. Nous avons noté ∇J^* le gradient de J^* et ΔJ^* le Laplacien de J^*. De manière analogue au cas discret, le contrôle optimal (la direction optimale $\theta^*(x)$ lorsqu'on est à la position x) est celle pour laquelle le max est atteint. Dans notre cas, il est facile de voir que le contrôle optimal $\theta^*(x)$ est la direction qui monte le gradient $\nabla J^*(x)$ de la fonction de valeur le plus rapidement : $\overrightarrow{u}(\theta^*(x)) = \frac{\nabla J^*(x)}{\|\nabla J^*(x)\|}$. En effet, $\nabla J^*(x) \cdot \overrightarrow{u}(\theta)$ est maximal lorsque θ est dans la direction de plus haute pente de $\nabla J^*(x)$.

En pratique, il n'est pas possible de calculer des solutions analytiques à l'équation 13, et une approche standard consiste à construire un schéma aux différences finies. Pour ce faire, nous suivons la démarche de [14]. Etant donnée une résolution $\delta > 0$, nous construisons une grille Σ^δ et son bord $\partial\Sigma^\delta$ sur le domaine Ω. Pour cette résolution, la fonction de valeur J est approchée par une fonction J^δ définie sur $\Sigma^\delta \cup \partial\Sigma^\delta$ et le contrôleur optimal est celui qui remonte le gradient d'une interpolation (par exemple une triangulation) de J^δ sur Ω.

Un tel processus de discrétisation a d'intéressantes propriétés : on peut montrer que l'approximation J^δ de la fonction de valeur J^* est la fonction de valeur d'un problème de contrôle à temps discret, c'est-à-dire d'un certain PDM. Ainsi nous retombons sur le cas PDM que nous avons exploité dans les sections précédentes. En conséquence, il est facile de construire un algorithme d'intelligence en essaim qui calcule J^δ : il suffit que ce modèle implémente une version asynchrone de l'algorithme *Itérations sur les valeurs*. La mise à jour des phéromones doit correspondre à l'opérateur de Bellman, opérateur que nous décrivons maintenant.

Notons $\cos^+$, $\cos^-$, $\sin^+$ et $\sin^-$ les parties positives et négatives des fonctions cos et sin : $\cos^\pm(\theta) = \max(\pm\cos\theta, 0)$ et $\sin^\pm(\theta) = \max(\pm\sin\theta, 0)$. De plus, définissons les probabilités de transition d'un point de la grille de discrétisation (x, y) vers ses quatre voisins lorsqu'on choisit d'aller dans la direction θ :

$$\begin{cases} p_\theta[(x,y),(x\pm\delta,y)] = \frac{1}{N_\theta}\left[\frac{\sigma^2}{2} + \delta\cos^\pm\theta\right] \\ p_\theta[(x,y),(x,y\pm\delta)] = \frac{1}{N_\theta}\left[\frac{\sigma^2}{2} + \delta\sin^\pm\theta\right] \end{cases} \tag{14}$$

où $N_\theta = \delta(\cos^+\theta + \cos^-\theta + \sin^+\theta + \sin^-\theta) + 4\sigma^2/2 = \delta(|\cos\theta| + |\sin\theta|) + 2\sigma^2$ peut être vu comme un facteur qui assure que la somme des probabilités de transition est 1. Enfin, notons $\tau(\theta) = \frac{\delta^2}{N_\theta}$, quantité qui peut être interprétée comme le temps requis pour aller du point (x, y) vers l'un de ses voisins lorsqu'on va dans la direction θ. Alors, on peut montrer que l'opérateur de Bellman associé à J^δ est :

$$B^\delta[W](x,y) = \gamma^{\tau(\theta^W_{x,y})} \sum_{(x',y')\in\Sigma^\delta} p_{\theta^W_{x,y}}\left[(x,y),(x',y')\right] W(x',y')$$

pour $(x, y) \in \Sigma^\delta$ et $B^\delta[W] = R$ sur $\partial\Sigma^\delta$, où $\theta^{J^\delta}_{x,y}$ est l'angle qui correspond à la direction de plus grande pente de J^δ au point (x, y) lorsqu'on considère une interpolation linéaire par morceaux de J^δ sur Ω. La notation $\theta^{J^\delta}_{x,y}$ peut paraître un peu lourde, mais nous l'utilisons car il est important de se souvenir que l'angle correspondant au contrôle optimal dépend des coordonnées (x, y) et de la fonction de valeur J^δ.

Maintenant que l'opérateur de Bellman B^δ est explicité, il est facile de construire un algorithme qui implémente le calcul de son point fixe. C'est ce que nous faisons maintenant. Les agents sont situés dans l'environnement original (ils ont des coordonnées qui peuvent prendre des valeurs continues) et s'y déplacent en mettant à jour des taux de phéromones qui correspondent aux valeurs de la fonction J^δ. Plus précisément, à chaque instant, chaque agent fait exactement deux choses :

- il met à jour la valeur de phéromone $J^\delta(x, y)$ du point (x, y) qui lui est le plus proche sur la grille comme suit :
$$J^\delta(x,y) \leftarrow [B^\delta J^\delta](x,y)$$
c'est-à-dire en utilisant uniquement les valeurs des phéromones des voisins du point (x, y) ;
- il se déplace (on peut considérer des déplacements de longueur infinitésimale dl ou d'une longueur fixée l) :
  - avec une probabilité ϵ (le taux d'exploration), il se déplace dans une direction choisie aléatoirement uniformément dans $[0, 2\pi]$;
  - avec une probabilité $1 - \epsilon$, il se déplace dans la direction qui monte le plus vite le gradient de l'interpolation locale des phéromones J^δ (autrement dit elle suit le contrôle optimal correspondant à J^δ).

Comme nous avons simplement modélisé une moitié du problème (la recherche de la source de nourriture), nous supposons de plus que les agents qui arrivent à une source de nourriture sont réinitialisés au nid.

Une simulation typique de ce modèle est illustrée à la figure 6, dans un environnement avec un nid et une source de nourriture. Tous les agents sont initialisés au nid. Comme attendu, on observe l'émergence d'un chemin entre le nid et la source de nourriture. Ce chemin est « renforcé » au fur et à mesure que le temps passe.

Dans ce cas, il est également possible de faire une analyse de la convergence. Si le taux d'exploration ϵ est strictement positif, les agents passent indéfiniment à proximité de chacun des points de la grille de discrétisation. La population constitue donc une version asynchrone de l'algorithme *Itérations sur les valeurs* et les phéromones convergent vers la solution approchée J^δ. On peut également caractériser le taux de convergence en utilisant une notion généralisée du temps de couverture. Dans le cas que nous venons de décrire, où les agents se déplacent dans l'espace continu et mettent à jour la valeur de phéromone du point de la grille le plus proche, le processus dynamique qui décrit la manière avec laquelle les points de la grille sont mis à jour n'est plus une simple chaîne de Markov mais une chaîne de Markov cachée : les points de la grille qui sont mis à jour sont une fonction de la position (réelle et continue) de chaque agent, qui est elle-même une chaîne de Markov dans l'environnement continu. Le temps de couverture qu'il faut considérer est donc celui d'une chaîne de Markov cachée. Malheureusement, il n'y a à notre connaissance pas de résultat dans la littérature concernant ce type de temps de couverture. Si nous avons pu observer expérimentalement que nous avons ici encore un taux de convergence superlinéaire, nous n'avons pas trouvé dans la littérature de résultat théorique correspondant.

6 Discussion et conclusion

Nous avons présenté un algorithme d'intelligence en essaim qui peut être lié au cadre du contrôle optimal, et nous avons prouvé sa convergence. Nous avons aussi analysé le taux de convergence de cet algorithme en fonction de la taille de la population : nous avons donné des arguments qui montrent, à l'aide de simulations et analytiquement (dans le cas discret) que le taux de convergence est superlinéaire en le nombre d'agents. À première vue, la superlinéairité peut apparaître comme une propriété très intéressante. Il faut toutefois la relativiser : la superlinéairité est due au fait que

de petites populations sont *particulièrement* inefficaces pour explorer l'espace et faire converger les traces de phéromones. En fait, l'analyse de convergence que nous avons faite suggère clairement (voir les équations 7 et 9) que la méthode la plus rapide pour calculer la fonction de valeur optimale est la méthode synchrone : à chaque pas de temps, la valeur s'approche de sa limite à un taux linéaire γ. La lenteur relative de notre algorithme doit être considérée comme le prix de la décentralisation pour mettre à jour les phéromones. D'autres stratégies d'exploration (moins aléatoires) comme par exemple celle décrite dans [21], pourraient être envisagées afin de pallier cette lenteur.

L'approche que nous avons présentée dans cet article est très flexible. Il est, par exemple, facile d'aborder des problèmes dans lesquels l'environnement contient plusieurs sources de nourriture chacune en quantité finie, qui décroît en fonction des visites des agents. Ces quantités peuvent être incorporées dans le modèle du contrôle optimal à travers la fonction récompense aux états sources. La récompense évolue alors au cours du temps, et les traces de phéromones, qui sont constamment mises à jour, suivront cette fonction de valeur optimale « changeante ». On peut aussi utiliser différents paramètres pour le voyage aller et le voyage retour des agents : ceci pourrait modéliser des stratégies différentes selon que les agents transportent ou non de la nourriture (on pourrait en effet souhaiter une approche plus sûre pour le retour). Enfin, si l'environnement est statique, on peut faire décroître lentement le taux d'exploration ϵ vers 0, de sorte que le comportement des agents converge finalement vers la politique optimale déterministe.

Le travail que nous venons de présenter est proche de travaux récemment publiés dans la littérature.

- Dans [21], l'auteur présente un algorithme multi-agent qui calcule le plus court chemin entre un nid et n'importe quel point d'une grille sous la forme d'une implantation asynchrone de l'algorithme de Bellman-Ford : la mise à jour locale est de la forme $U(x) \leftarrow 1+\min_{x' \in \mathcal{V}(x)} U(x')$. Modulo le changement de variable $U \leftrightarrow \frac{\log(V)}{\log(\beta)}$, ceci est équivalent à notre mise à jour des traces de phéromones avec $\alpha = 1$ (eq. 1). L'auteur utilise ce mécanisme comme une base pour construire un système de fourragement distribué efficace (il utilise un mécanisme original pour l'exploration et la coopération avec une seconde trace binaire).
- Dans [17], les auteurs considèrent la problématique du fourragement et proposent d'identifier, comme nous le faisons ici, la fonction de valeur de problèmes de contrôle à des taux de phéromones. Ils utilisent des versions asynchrones d'*Itérations sur les valeurs* ainsi que certaines variantes algorithmiques (dites d'apprentissage par renforcement) pour résoudre le problème du contrôle optimal [22]. Ils présentent de nombreuses expériences qui montrent que ce genre de systèmes est robuste dans des environnements contenant des obstacles, où les positions du nid et des sources de nourriture varient au cours du temps, et s'il y a plusieurs chemins à trouver (ils utilisent, comme nous, un taux de phéromone par chemin à trouver).

Il y a des différences importantes entre ces travaux et le nôtre :

- ces travaux concernent uniquement des problèmes de navigation déterministe. L'étude que nous avons faite permet de considérer des environnements où les déplacements sont stochastiques ;
- a notre connaissance, il n'y a pas d'arguments concernant la convergence pour les algorithmes de [17] tandis qu'une version étendue de [21] contient une preuve qui repose sur le même genre d'arguments que celui que nous donnons dans ce papier [20][10].
- enfin, la superlinéarité du taux de convergence est observée expérimentalement dans [20] mais pas dans [17], et dans aucun cas elle n'est abordée théoriquement.

En d'autre termes, notre travail a pour ambition de généraliser et d'approfondir théoriquement ceux de [17, 21, 20].

Au-delà du problème simple du fourragement, il est facile de voir qu'on peut construire des algorithmes d'intelligence en essaim du même genre que celui que nous avons présenté pour résoudre n'importe quel problème de contrôle optimal. Tant que le problème est formulé comme un PDM, nous savons qu'il peut être résolu par une version asynchrone d'*Itérations sur les valeurs*, et construire l'algorithme correspondant est immédiat : il suffit que ces agents évoluent dans l'environnement et mettent à jour une phéromone qui joue le rôle de la fonction de valeur partout où elles passent. Si l'on veut satisfaire la contrainte que « les agents doivent prendre leurs décisions en utilisant uniquement l'information locale », il est facile de voir que cette approche fonctionnera tant que les transitions, dans le PDM, sont *locales* dans l'espace d'états. Nous pouvons aller encore plus loin : toute notre analyse (la convergence et le taux de convergence) repose sur la propriété de « contraction » qui est vérifiée par l'opérateur de Bellman. Ceci suggère que tout problème qui implique le calcul du point fixe d'une contraction (par exemple, on trouve des contractions dans des problèmes tels que la recherche du zéro d'une fonction, ou l'optimisation [15]) sur un espace fini a une solution naturelle

[10] Comme [20] considère un environnement déterministe la preuve de convergence diffère : l'auteur montre que la fonction de valeur optimale est atteinte en un temps fini presque sûrement tandis que dans notre cas nous avons seulement une convergence asymptotique.

par un algorithme fourmi : les agents évoluent dans l'espace fini et effectuent de manière distribuée et asynchrone le calcul du point fixe de cette contraction. La construction de systèmes d'intelligence en essaim résolvant des problèmes autres que le fourragement constitue ainsi l'une des perspectives immédiates de ce travail.

Références

[1] D. Aldous and J. Fill. *Reversible Markov Chains and Random Walks on Graphs*. Monograph in preparation, 1996.

[2] G. Beni and J. Wang. Swarm intelligence. In RSJ press., editor, *Seventh Annual Meeting of the Robotics Society of Japan*, pages 425–428, 1989.

[3] D.P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation : Numerical Methods*. Prentice hall, 1989.

[4] D.P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[5] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence : from natural to artificial systems*. Oxford University Press, Inc., New York, NY, USA, 1999.

[6] A. Boumaza and B. Scherrer. Optimal control subsumes harmonic control. In *Proceedings of the 2007 IEEE International Conference on Robotics a nd Automation*. IEEE, April 2007.

[7] C.I. Connolly and R. Grupen. On the applications of harmonic functions to robotics. *Journal of Robotic and Systems*, 10(7) :931–946, October 1993.

[8] J. L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Insect Behavior*, 3 :159–168, 1990.

[9] J. L. Deneubourg, S. Goss, and N. R. Franks. The blind leading the blind : Modeling chemically mediated army ant raid patterns. *Insect Behavior*, 2 :719–725, 1989.

[10] Marco Dorigo and Christian Blum. Ant colony optimization theory : a survey. *Theoretical Computer Science*, 344(2-3) :243–278, 2005.

[11] Uriel Feige and Yuri Rabinovich. Deterministic approximation of the cover time. *Random Struct. Algorithms*, 23(1) :1–22, 2003.

[12] S. Guerin and D. Kunkle. Emergence of constraint in self-organizing systems. *Nonlinear Dynamics, Psychology, and Life Sciences*, 8(2) :131–146, 2004.

[13] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, 1988.

[14] H.J. Kushner and P.G. Dupuis. *Numerical Methods for Stochastic Control Problems in Continuous Time*. Springer Verlag, 1992.

[15] D. Luenberger. *Linear and nonlinear programming*. Addison-Wesley, New York, 1989.

[16] L. Panait and S. Luke. Ant foraging revisited. In *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE-IX)*, 2004.

[17] L. Panait and S. Luke. A pheromone-based utility model for collaborative foraging. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2004.

[18] M. Puterman. *Markov Decision Processes*. Wiley, New York, 1994.

[19] M. Resnik. *Turtles termites and traffic jams*. MIT Press, 1994.

[20] O. Simonin. Personal communication.

[21] O. Simonin. Construction of numerical potential fields with reactive agents. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2005.

[22] R.S. Sutton. Artificial intelligence as a control problem : Comments on the relationship between machine learning and intelligent control. In *IEEE International Symposium on Intelligent Control*, pages 500–507, 1988.

[23] R.S. Sutton and A.G. Barto. *Reinforcement Learning, An introduction*. BradFord Book. The MIT Press, 1998.

[24] M. Wodrich and G. Bilchev. Cooperative distributed search : The ants' way. *Control and Cybernetics*, 26, 1997.

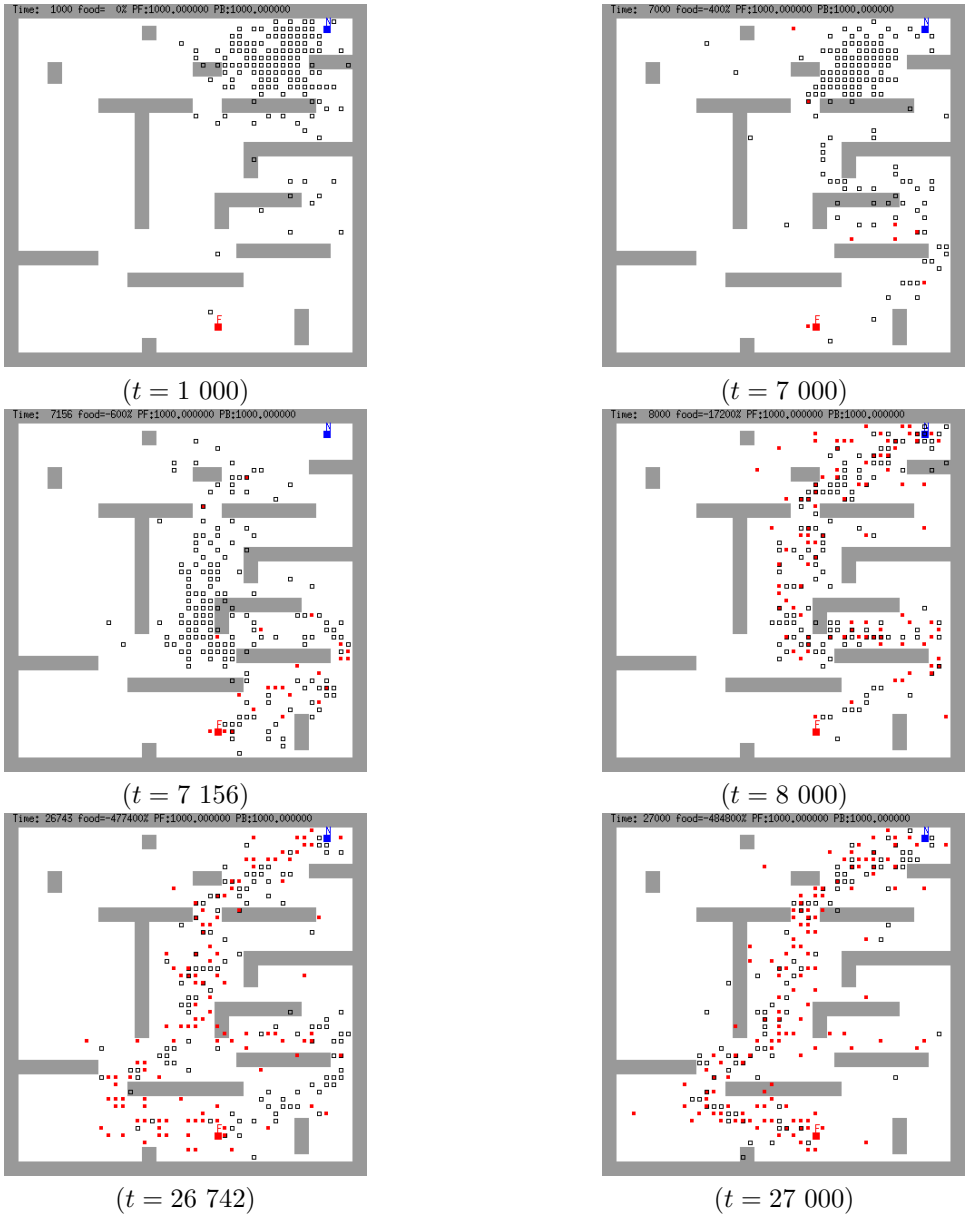

FIG. 1 – Simulation de l'algorithme : les carrés pleins représentent les agents chargés et les carrés vides représentent les agents qui ne le sont pas. La source de nourriture est en bas à gauche et le nid est en haut à droite. Voir le texte

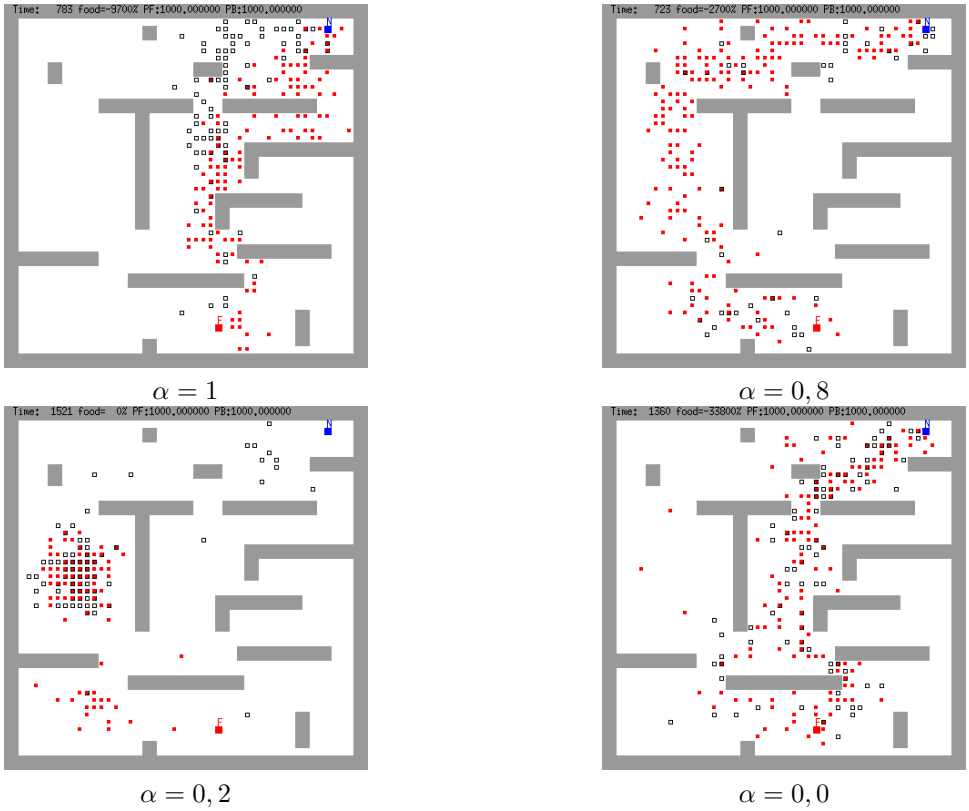

FIG. 2 – Distribution limite des agents en fonction de la valeur de α : différentes valeurs du paramètre donnent différents chemins entre le nid et la source et des fois pas de chemin

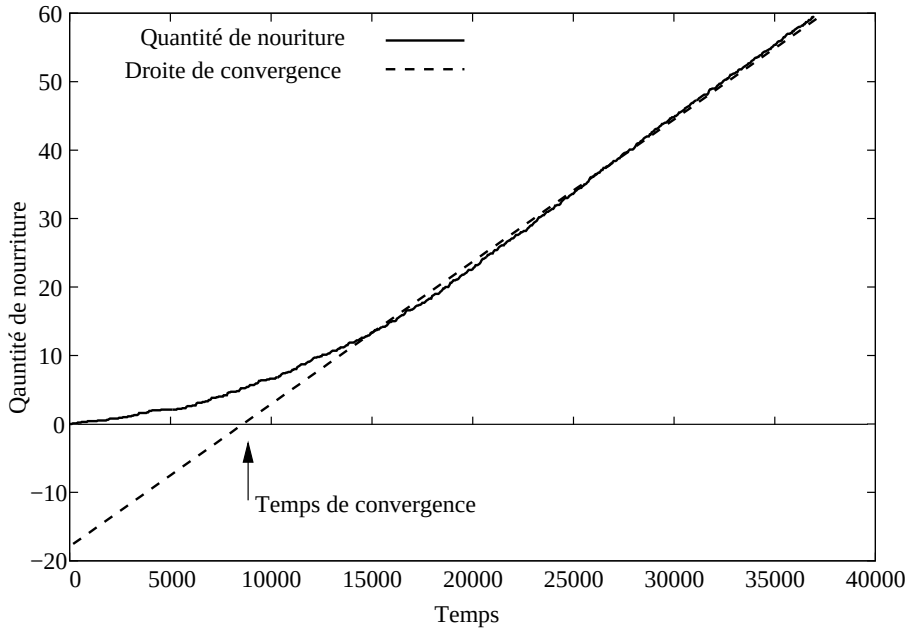

FIG. 3 – Mesure de la convergence : unités de nourriture accumulée au nid au cours du temps normalisée par le nombre d'agents. Pour une simulation on mesure le temps de convergence comme l'inverse du temps de l'intersection de la droite de convergence et l'axe des abscisses (la régression linéaire est effectuée sur la partie linéaire de la courbe)

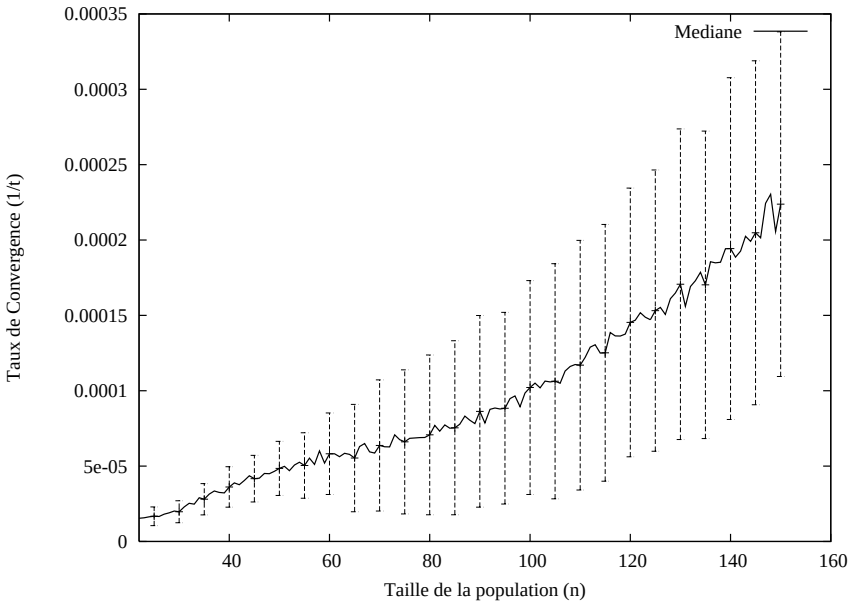

FIG. 4 – Taux de convergence en fonction de la taille de la population : le résultat représente la valeur médiane du taux sur 7 000 simulations menées sur l'environnement de la figure 2 avec les paramètres suivants : $\alpha = 0,9$, $\beta = 0,999999$, $\epsilon = 0,7$, $t_{max} = 100\ 000$ et $sp = 0,5$

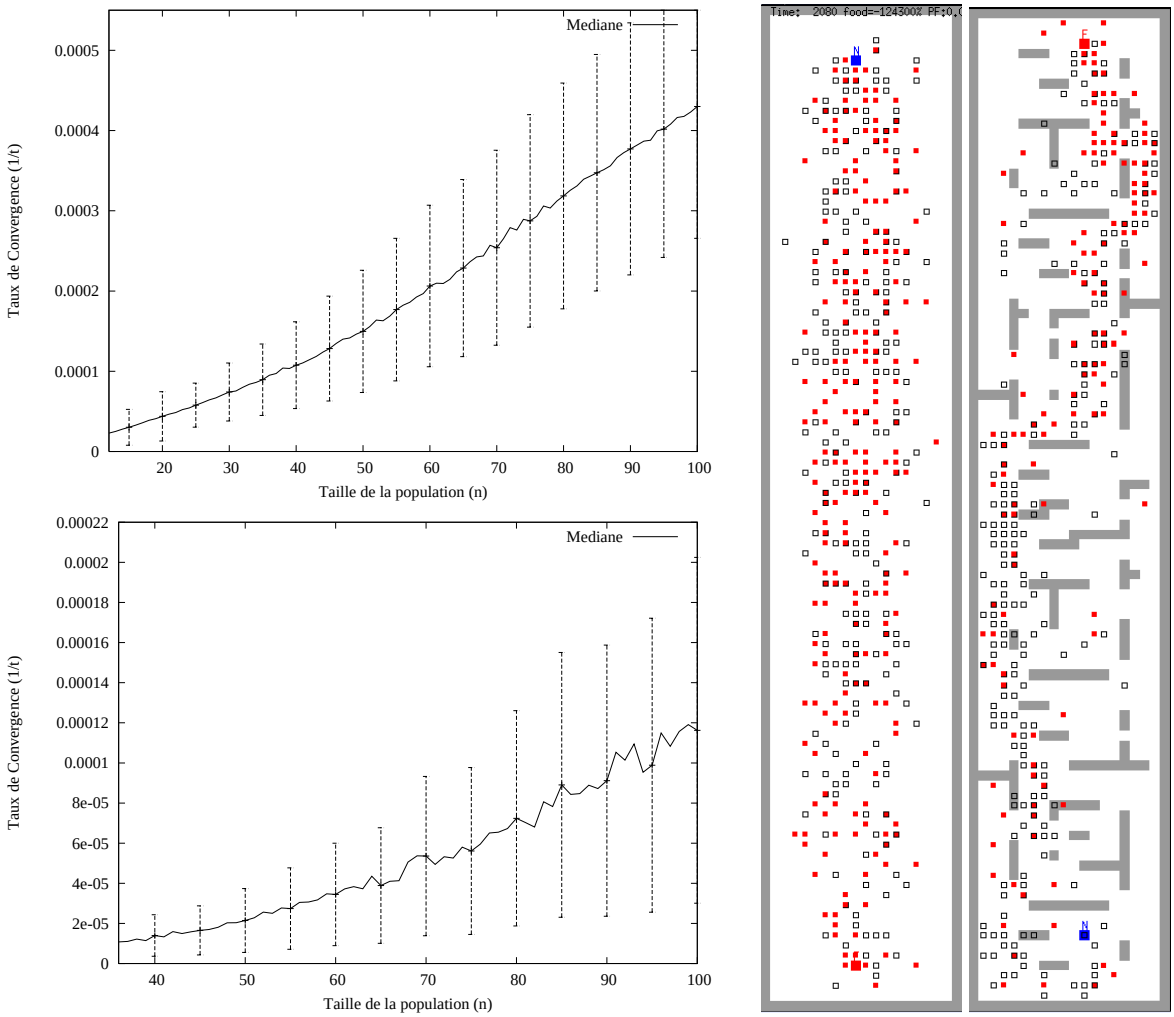

FIG. 5 – Taux de convergence en fonction de la taille de la population : les courbes représentent la valeur médiane sur 5 000 simulations. Celle du haut correspond aux simulations sur l'environnement de gauche et celle du bas a l'environnement de droite. Dans ces deux simulations les valeurs des paramètres de l'algorithme sont : $\alpha = 0,9$, $\epsilon = 0,8$, $\beta = 0,999999$, $t_{max} = 100\ 000$ et $sp = 0,5$

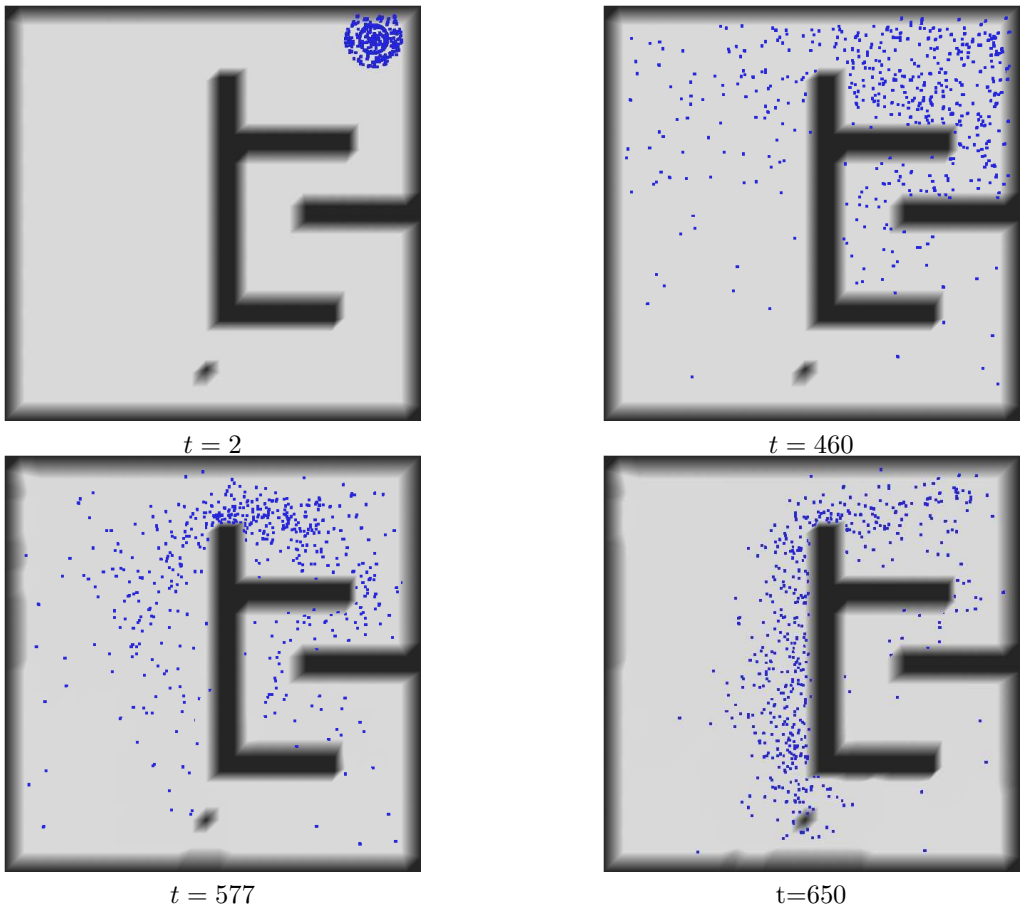

FIG. 6 – Expérience typique montrant la convergence et l'émergence d'un chemin : quatre « photos » d'une exécution de l'algorithme continu dans un environnement contenant 30×30 points grilles et 512 agents