Project : Aifants !

AI for ants !

Année 2025-2026

Jean-Baptiste Caignaert ©

Image issue de du jeu SimAnts sur Super Nintendo

Contexte et objectif

L'objectif de ce projet est de simuler une colonie de fourmis évoluant sur une grille 2D contenant des sources de nourriture, des obstacles et des zones mortelles. Les fourmis doivent apprendre (par Q-learning) à :

- trouver la nourriture et ramener des ressources au nid ;
- laisser et suivre des phéromones pour guider leurs congénères ;
- éviter les zones dangereuses (mortelles) et optimiser leurs trajets.

Vous allez développer en langage Python/Java/C/C++ (ou autre à faire valider) un programme qui va optimiser des déplacements de fourmis pour amener l'ensemble de la nourriture au nid.

L'univers est décrit par une grille de case de taille x*y, où se déplacent des fourmis pouvant transporter de la nourriture. L'univers commence au temps 0 et se termine au temps 100.000 (unité de temps S) ou quand toutes les ressources de nourriture sont finies (le premier des deux arrivant en premier).

Détails des fourmis

Les fourmis se déplacent dans les 4 directions possibles. Les fourmis ont des caractéristiques et des visions différentes.

. Elles sont décrites par :

- Une charge maximale de nourriture : capacité maximum de nourriture qu'elles peuvent transporter
- Une vitesse de déplacement (nombre d'unité de déplacement par case(M) en unité de temps (S))
- Un champs de vision, nombre de case qu'elles peuvent voir (détecter les objets)

Le nid est décrit par :

- Nombre de fourmis qu'elle peut mettre sur le terrain autour d'elle
- Quantité de nourriture récupérée
- Quantité de fourmis disponibles de chaque catégories

Une source de nourriture est décrite par :

- Quantité de nourriture restante

Il existera aussi sur le terrain des zones où les fourmis ne peuvent aller (dit “mur”) et les zones de danger (dite “mortel”). Si une fourmis se rend sur une zone mortelle, elle cesse d’exister de suite.

Il peut y avoir jusqu’à 10 fourmis sur la même case en même temps.

Programme à réaliser:

Univers à coder :

3 catégories de fourmis qui possèdent un facteur d’actualisation, un facteur d’apprentissage et un facteur ε-greedy (% de chance d’aller au hasard ou de suivre la meilleure destination).

Une vision de 1 permet à une fourmis de ne pas réellement exécuter un déplacement mais de mettre à jour la récompense “comme si” elle l’avait fait. Cela sert à indiquer les zones mortelles (c’est la seule utilisation permise de cette vision). Les récolteuses peuvent mourir si elles vont sur une zone mortelle.

Exploratrice :

- Capacité maximale : 10 unité de nourriture
- Vitesse : 1 case les 5S
- Vision 1 case de distances

Combattante:

- Capacité maximale : 10 unité de nourriture
- Vitesse : 1 case les 5S
- Vision 1 case de distance

Récolteuse:

- Capacité maximale : 100 unité de nourriture
- Vitesse : 1 case les 10S
- Vision 0 case de distance

Chaque fourmis possède 2 modes :

“recherche de nourriture” quand elle a 0 unité de nourriture

“recherche du nid” quand elle a >0 unité(s) de nourriture

Phéromones :

Chaque case possède deux taux de phéromone, qui est mis à jour par les fourmis à chaque déplacement, avec l’algorithme de Q learning.. Un taux vers la nourriture et un taux vers le nid

A chaque tour de jeu les phéromones se dissipent de 1%. (indice des matrices de Q learning qui tendent vers 0)

Chaque type de recherche utilise des phéromones différentes et donc soit une Q table “nourriture”, soit la Q table “nid”.

Chaque Q table est initialisée à 0 sur toutes les cases. Les déplacements vers les murs sont enlevés.

La quantité de phéromone laissée est calculée par la formule de Q learning. La récompense du Nid et de la nourriture est mise à 1000 de manière arbitraire (paramètre modifiable). La récompense d’une zone mortelle est à -500. La récompense d’une case autre est à -1. Les murs ne sont pas possibles

Nid :

Votre programme devra gérer le nid : à quel moment, quels fourmis seront mises “dehors”. Une fourmis mise dehors ne peut rerentrer que si elle possède de la nourriture. A chaque rentrée de nourriture, le programme décide si elle laisse la fourmis ressortir de suite ou plus tard.

Univers :

Écrivez un programme qui décrit un univers simple, le nid contiendra: 2 exploratrices,1 combattante ,3 récolteuses.

Nid					Mur			
					Mur			Nourriture 20000
					Mur			
		Mur			Zone Mortelle			
		Zone Mortelle						
					Mur			
	Mur							
								Nourriture (1000)

Algorithmes :

Nous vous invitons à faire de multiples tests pour vérifier que tout est cohérent.

Votre programme devra dérouler à chaque unité de temps la situation de l’univers. Il est indispensable d’avoir une représentation graphique claire pour voir à chaque unité de temps la situation globale (notamment les fourmis). Il est demandé d’avoir un traçage de l’historique et de

pouvoir donc « revenir en arrière » de manière facile et fluide. Il faut qu'on puisse à un instant t déterminer quelles fourmis sont présentes,, leurs caractéristiques. Il faut qu'on puisse visualiser clairement les 2 Q-tables et les 2 types de phéromones sur la carte.

Il est demandé que TOUS les paramètres puissent être mis comme paramètre de votre programme et appelé par un autre programme englobant. Ce "méta programme" devra trouver la meilleure combinaison pour optimiser le délai moyen.

Le méta programme envoi aussi bien la carte, que les caractéristiques mais ne décide rien dans les gestion des déplacements par exemple.

Les paramètres du Q learning de chaque fourmis par exemple seront modifiables.

Récapitulatif :

- Méta programme envoi ensemble des paramètres de la simulation à Programme gestion fourmis
- Programme gestion fourmis renvoie au méta programme le temps de fin de la simulation

Sur vos principaux différents algorithmes expliqués, il faudra préciser la complexité calculée. La complexité devra être calculée sur l'outil de comparaison de vitesse de fin de la simulation.

Il est demandé une fois la simulation du sujet donnée, de tester avec d'autres simulations et d'autres paramètres.

Rendu:

Le projet devra être rendu de manière individuelle, avec le code (comprenant toutes les librairies utilisables), le document expliquant les choix techniques et la vision de la complexité.

Le code devra être accompagné obligatoirement avec deux fichiers :

-résultat : présentant les résultats du déroulement de votre programme, il est attendu que vous décriviez précisément ce qui fonctionne, ne fonctionne pas.

-utilisation : décrivant précisément les commandes à faire pour lancer votre programme. Cela doit pouvoir être lancé par un non informaticien (ayant le langage de programmation sur son ordinateur)

Les applications bonus suivantes seront gratifiées :

-représentations graphiques claire avec utilisation de la souris

-pouvoir créer sa propre map facilement

-animation