Optimisation de colonie de fourmis

Guillaume Noguera

June 11, 2017

Abstract

Le but du projet était de modéliser et réaliser une simulation de colonie de fourmis en suivant le modèle algorithmique de l'intelligence d'essaim. En effet, les fourmis suivent des règles simples qui leurs permettent de déterminer le chemin le plus court entre le terrier et une source de nourriture. Une implémentation de cette logique est connue et utilise des graphes pondérés : au vu des délai, ce laboratoire utilise un système de grille. De plus, les résultats ne sont que peu probant.

1 Implémentation

1.1 Comportement d'une fourmi

Les fourmis ne sont pas douées d'une grande intelligence, mais sont capable de rapidement trouver le chemin le plus court entre leur fourmillière et une source de nourriture. En effet, une fourmi se déplace de façon plutôt aléatoire et ce jusqu'à trouver quelque chose à manger. Ensuite, elle va parcourir le chemin inverse en "marquant" le sol de phéromones.

Une fourmi rencontrant ces phéromones saura ainsi que de la nourriture est obtenable en suivant ces phéromones, et renforcera ces derniers. Néanmoins, il existe une chance qu'une fourmi dissidente décide de ne pas suivre le chemin : ainsi, elle créera peut-être un nouveau chemin.

Les phéromones perdant en intensité dans le temps (et les fourmis ayant tendance à préférer les chemins composés des phéromones les plus intenses), les chemins plus longs seront naturellement moins attirants et disparaitront progressivement. Ainsi, après moultes itérations, le chemin optimal est trouvé.

1.2 Structure de donnée

La classe $Tile$ représente une case du tableau. Elle possède des coordonées x et y et peut être occupée par un obstacle. On y stocke les valeurs de phéromones laissées par les fourmis. La classe Map s'occupe de stocker les différentes instances de $Tile$ dans un tableau à deux dimensions. Elle est utilisée par les fourmis pour obtenir des informations sur leur environnement. La classe $MemoryNode$ est une petite classe utilitaire utilisée par les fourmis pour garder en tête le chemin parcouru. Pendant chaque itération, les fourmis ajoutent un $MemoryNode$ au tableau ($this.memory$). Elles utilisent ensuite ce tableau pour revenir à la fourmillière.

Enfin, la classe Ant se charge évidemment de représenter les fourmis, qui suivront la routine établie par $Ant :: update()$.

1.3 Boucle principale

(Par ordre de priorité)

- Si la fourmi est à la fourmillière et qu'elle rapporte de la nourriture, elle laisse sa nourriture sur place et repart dans une direction aléatoire.
- Si la fourmi est proche de la fourmillière et qu'elle apporte de la nourriture, elle ira forcément à la fourmillière le tour d'après.
- si la fourmi rapporte de la nourriture et qu'elle tombe sur des phéromones, elle en dépose elle-même et continue dans sa direction.
- Si la fourmi rapporte de la nourriture, elle parcourt le chemin inverse stocké dans sa mémoire.
- Si la fourmi trouve de la nourriture, elle marque le terrain et se met à parcourir le chemin inverse.
- Si la fourmi cherche de la nourriture et se trouve sur des phéromones, elle marque le terrain et a une chance de suivre les phéromones.
- Sinon, la fourmi se ballade aléatoirement.

(Algorithme trouvé sur le github suivant : https://github.com/andreasjansson/ants_simulation

2 Problèmes rencontrés

Malheureusement et en l'état, la simulation ne fonctionne pas comme voulu. Plusieurs méthodes de détection des phéromones ont étées tentées (globale, directionelle ..), et je n'arrive pas à déterminer si le soucis vient d'un soucis de paramètres ou d'un défaut de l'algorithme. Je continuerai néanmoins à améliorer l'algorithme, le problème m'intéressant et n'ayant pas trouvé d'implémentation correcte du problème.

Quelques pistes à explorer néanmoins :

- Le chemin aléatoire est son parcours inverse ont tendance à générer des grandes zones de phéromones très denses, perturbant les autres fourmis - Il faudrait modifier la mémoire pour éviter ces allers-retours.
- Les variables $Tile :: pheromoneDecrement$, $Ant :: pheromoneIncrement$, et le nombre de fourmis peuvent entrer en compte, mais le projet n'a pas bénéficié de tests assez long pour déterminer si ces mauvais coefficients pouvaient être la cause du soucis.
- Modification plus profonde de l'algorithme ?

3 Conclusion

Malgré la déception liée à l'absence de résultats intéressants, j'ai trouvé que sujet était intéressant (en plus d'être formateur concernant l'utilisation de Javascript). Les bases du projet marchent sans bug notable et je pense donc m'y intéresser à nouveau après cette période remplie.