Comportement artificiel d'une colonie de fourmis

proposé par Arnaud Sangnier (sangnier@liafa.univ-paris-diderot.fr)

Projet de Programmation 2013/2014 - 12 septembre 2013

1 Simulation de colonies de fourmis sur une grille

Une façon de résoudre des problèmes de recherche de chemins dans un graphe est d'utiliser un algorithme dit de colonies de fourmis. L'idée est la suivante : le but pour les fourmis est d'atteindre les sources de nourriture en évitant les obstacles et de ramener la nourriture à la colonie. Lors de leur déplacement, les fourmis déposent de la phéromone qui contient deux types d'information, soit la fourmi qui laisse la phéromone est partie de la colonie et n'a pas encore trouvé de nourriture, soit elle a trouvé de la nourriture et elle est en train de rentrer vers la colonie. Lorsque plusieurs fourmis se comportent ainsi sur un graphe, le comportement qui devrait être observé est que au bout d'un certain laps de temps, les fourmis finissent par trouver le plus court chemin menant de la colonie à la source de nourriture, ceci car les quantités de phéromone déposées sur les "bons" chemins augmentent. Si vous souhaitez des informations un peu plus détaillées, vous pouvez aller jeter un coup d'oeil sur la page web suivante : `http://fr.wikipedia.org/wiki/Algorithme_de_colonies_de_fourmis`.

2 Description du projet

Le but de ce projet est de programmer une interface graphique pour la visualisation d'algorithmes de colonies de fourmis. Les fourmis évolueront sur une grille de case en case en déposant de la phéromone sur les cases où elles passeront. Sur cette grille, il y aura également des cases spéciales correspondant soit à des sources de nourriture soit à des obstacles. La visualisation du comportement des fourmis sera dynamique, on verra ainsi les fourmis se déplacer sur la grille. Ce projet comprendra plusieurs phases de développement et libre à vous de proposer éventuellement d'autres directions.

2.1 Génération aléatoire de grilles et visualisation

La première phase de ce projet consistera à développer un algorithme de génération automatique de grilles sur lesquelles une colonie sera placée sur une certaine case, des obstacles et des sources de nourriture seront posés sur d'autres cases. Il faudra que votre algorithme de génération de grilles permettent de générer des grilles de différentes taille avec une densité plus ou moins forte d'obstacles et de nourriture. Vous devrez de plus garantir que depuis toute case sans obstacle de la grille, il existe un chemin (sans obstacle) vers toute autre case sans obstacle, ceci afin d'éviter que des sources de nourriture soient inaccessibles depuis la colonie. Vous devrez de plus programmer une interface graphique permettant la visualisation de votre grille.

2.2 Comportement des fourmis

Nous allons passer maintenant à la description du comportement des fourmis sur la grille. La position de départ des fourmis est la colonie. Au cours de leur déplacement, les fourmis laissent sur les cases non

occupées (par un obstacle ou par de la nourriture) de la phéromone de deux type, le premier type sera de la phéromone dite *nid* et le deuxième type de la phéromone dite *nourriture*. Sur les cases, la phéromone s'accumule (par exemple de façon additive en incrémentant d'une unité la quantité de phéromone présente sur la case). Les fourmis ne voient pas l'environnement et donc ne savent pas où se trouvent la nourriture ou la colonie et pour se repérer elles utilisent la phéromone présente sur les cases de la grille. Bien entendu au début de la simulation, aucune case de la grille ne comporte de la phéromone. À chaque coup, toutes les fourmis sorties de la colonie décident sur quelles cases elles doivent aller et ce en respectant les règles suivantes :

- si la fourmi ne porte pas de nourriture et si il n'y a pas de case voisine comportant de la phéromone *nourriture*, elle choisit au hasard une case voisine sans obstacle où aller ;
- si la fourmi ne porte pas de nourriture et si il y a des cases voisines comportant de la phéromone *nourriture*, elle choisit au hasard une case voisine où aller parmi les cases ayant le plus de phéromone *nourriture* ;
- si la fourmi porte de la nourriture et si il n'y a pas de case voisine comportant de la phéromone *nid*, elle choisit au hasard une case voisine sans obstacle où aller (normalement ce cas ne devrait pas arriver, vu que toute fourmi vient de la colonie) ;
- si la fourmi porte de la nourriture et si il y a des cases voisines comportant de la phéromone *nid*, elle choisit au hasard une case voisine où aller parmi les cases ayant le plus de phéromone *nid* ;
- si une fourmi portant de la nourriture a une case voisine correspondant à la colonie, elle va sur cette case ;
- si une fourmi ne portant pas de nourriture a une case voisine sur laquelle se trouve de la nourriture, alors elle va sur cette case.

De plus lorsqu'une fourmi chargée de nourriture arrive à la colonie, elle relâche la nourriture et si une fourmi non chargée arrive à une case nourriture, elle prend de la nourriture. Finalement, lorsqu'une fourmi chargée de nourriture arrive sur une case, elle laisse de la phéromone *nourriture* et si elle n'est pas chargée de nourriture elle laisse de la phéromone *nid*.

Vous devrez programmer un tel comportement de fourmis et ensuite afficher le résultat de la simulation graphiquement, en faisant évoluer de façon automatique votre grille à chaque coup. Sur la grille , vous devrez afficher les différentes fourmis ainsi que la phéromone (une idée pour afficher les différentes quantités de phéromone pourrait être d'utiliser un dégradé de couleur). Dans votre simulation, vous devrez laisser le choix à l'utilisateur de donner la quantité de fourmis dans la colonie ainsi que la vitesse de la simulation (un coup par seconde, un coup toutes les demi-seconde, etc). Il serait bien que l'utilisateur, grâce, par exemple, à des champs ou des boutons de l'interface graphique, puisse augmenter ou réduire à volonté la vitesse de la simulation.

Pour cette première partie de la simulation, on pourra supposer que la phéromone déposée ne disparaît jamais, que les fourmis ne meurent jamais et que les sources de nourriture sont infinies (elles ne se tarissent donc jamais).

2.3 Modification dynamique

Dans la description précédente, les obstacles et les sources de nourriture sont fixés au début de la simulation. On pourrait penser à une évolution dynamique par exemple en autorisant l'utilisateur à placer au fur et à mesure de la simulation des obstacles ou en l'autorisant à les supprimer (par exemple en cliquant sur la grille avec la souris). Aussi, on pourrait imaginer que les sources de nourriture ne sont pas infinies, mais que chacune d'elle comporte une certaine quantité de nourriture et que, quand cette quantité a été consommée, la source correspondante disparaît ; là aussi on pourrait autoriser l'utilisateur a rajouter des sources de nourriture manuellement. Finalement, pour qu'un tel comportement dynamique fonctionne le mieux possible, il faudra aussi faire en sorte qu'au fur et à mesure de la simulation, les quantités de phéromone les plus vieilles disparaissent (il faudra donc donner un âge à la phéromone et par exemple dire qu'au bout d'un certain nombre de coups, la phéromone déposée disparaît).

2.4 Quelques conseils

Le sujet ci-dessus vous donne un certain nombre d'indications sur la façon dont votre programme devra fonctionner mais tout n'est pas explicité dans les moindres détails. Réfléchissez bien à votre spécification avant de programmer (par exemple, comment incrémenter les doses de phéromones, dans quelles directions les fourmis peuvent-elles évoluer sur la grille, elles pourraient par exemple se déplacer en diagonal ou pas, pour la dernière partie, quelles sont les règles d'évolution dynamique pour la phéromone, pour la nourriture, etc).

2.5 Extensions possibles

Le sujet de ce projet vous donne des lignes directrices, mais n'hésitez pas à proposer des modifications ou des améliorations (à discuter avec votre encadrant). Par exemple, on pourrait supposer plus que deux types de phéromones ou bien un certain taux de fourmis défaillantes qui ne respectent pas les règles de l'algorithme ou encore supposer que certains obstacles sont franchissables mais avec une certaine pénalité d'attente (pour représenter un terrain non plat par exemple).