M1 Info - Résolution Collective de Problèmes
TP 2 - Ant Colony Optimization

Tom Jorquera - tom.jorquera@irit.fr

> Les algorithmes de colonies de fourmis sont des algorithmes inspirés du comportement des fourmis et qui constituent une famille de métaheuristiques d'optimisation.
>
> Initialement proposé par Marco Dorigo et al. dans les années 1990, pour la recherche de chemins optimaux dans un graphe, le premier algorithme s'inspire du comportement des fourmis recherchant un chemin entre leur colonie et une source de nourriture. L'idée originale s'est depuis diversifiée pour résoudre une classe plus large de problèmes et plusieurs algorithmes ont vu le jour, s'inspirant de divers aspects du comportement des fourmis.
>
> En anglais, le terme consacré à la principale classe d'algorithme est « Ant Colony Optimization » (abrégé ACO). Les spécialistes réservent ce terme à un type particulier d'algorithme. Il existe cependant plusieurs familles de méthodes s'inspirant du comportement des fourmis. En français, ces différentes approches sont regroupées sous les termes : algorithmes de colonies de fourmis, optimisation par colonies de fourmis, fourmis artificielles, ou diverses combinaisons de ces variantes.
>
> Extrait de l'article *Algorithme de colonies de fourmis* - Wikipedia fr
> `http://fr.wikipedia.org/wiki/Algorithme_de_colonies_de_fourmis`

Dans ce TP, nous allons commencer à développer notre propre algorithme Ant Colony Optimization (ACO). Les agents-fourmis auront pour objectif de récolter des ressources dispersées dans l'environnement de la fourmilière.

Préparation de l'environnement

Dans cette première partie, nous allons générer l'environnement dans lequel évolueront les fourmis. NetLogo vous permet d'attribuer des variables supplémentaires à vos différents agents patches à l'aide du mot-clé *patches-own* (des mots-clés similaires existent pour les autres types d'agents). Vous utiliserez cette primitive pour représenter le fait que chaque patch peut contenir de la nourriture.

Vous créerez ensuite une procédure d'initialisation qui attribue aléatoirement une quantité fixée de nourriture avec une chance de 5%, et utiliserez les capacités des agents patchs pour différencier visuellement les patchs contenant de la nourriture (à l'aide de couleurs différentes par exemple).
Dans un second temps, vous permettrez à l'utilisateur d'ajuster les différents paramètres d'attribution de nourriture à l'aide d'éléments de l'interface graphique.
Enfin vous utiliserez la primitive *diffuse* pour permettre des générer des environnements où la nourriture est regroupée en différents tas. Là encore, vous fournirez à l'utilisateur les moyens de régler les paramètres de génération.

La fourmilière

Le mot-clé *breed* vous permet de déclarer une sous-espèce d'agent turtle. Les sous-espèces vous offrent de nombreuses facilités, comme le fait d'associer des variables uniquement à une sous-espèce particulière ou de faire exécuter une commande uniquement par les agents de cette sous-espèce.
Déclarez une sous-espèce pour vos agents fourmis, puis attribuez leur une variable représentant la nourriture transportée par une fourmi.

Pour ce tp, on supposera que la fourmilière est toujours crée au milieu de la carte (vous pouvez utiliser une couleur distincte pour représenter le patch contenant la fourmilière). Modifier votre procédure de setup pour que des agents fourmis soit créés sur la fourmilière, puis faites-les se déplacer aléatoirement sur la carte.
Vous ajouterez ensuite le comportement suivant : lorsqu'une fourmi se déplace sur un patch contenant de la nourriture, il en prélève une quantité déterminée et la ramène à la fourmilière. Tout en retournant à la fourmilière la fourmi doit déposer des phéromones sur son chemin (vous devrez ajouter une variable à vos patchs pour représenter la quantité de phéromones déposée). En vous inspirant du TP précédent, vous modifierez les fourmis pour qu'elles utilisent les phéromones de la manière suivante : les fourmis en exploration qui perçoivent des patchs contenants phéromones dans leur voisinage auront plus de chance de se diriger ces derniers.
En utilisant une variable globale (mot-clé *globals*), vous fournirez à l'utilisateur le moyen de visionner la quantité de nourriture encore présente dans l'environnement et la quantité de nourriture récoltée. Vous présenterez ces résultats tout d'abord à l'aide d'un élément *Monitor*, puis à l'aide d'un élément *Plot* (pour plus d'informations sur l'utilisation du Plot, référez-vous en particulier à `http://ccl.northwestern.edu/netlogo/docs/programming.html#plotting`).