1/4- Projets Inter-Promos

W-WEB-404

ACO

Initiation au Machine Learning et au Path Finding

v1.0

EPITECH.
L'ECOLE DE L'INNOVATION ET DE L'EXPERTISE INFORMATIQUE

ACO

Sommaire

Détails administratifs

- Le projet est à réaliser en groupe de 3 à 4 personnes.
- Les sources doivent être rendues avec BLIH.
- Répertoire de rendu : **ACO**

Projet

Bienvenue dans ce sujet d'initiation au machine learning et au path finding.
Dans ce projet vous devrez implémenter un algorithme bien connu du machine learning : l'ACO (Ant Colony Organisation).

Le but est de réaliser un programme qui trouve le chemin le plus court entre une position et une autre, en simulant le fonctionnement "naturel" d'une fourmilière.
Ce type d'algorithme est très intéressant pour une raison principale : le changement.
En effet, si la source de nourriture venait à bouger ou si des murs étaient déplacés (imaginons une branche d'arbre tombant au sol), ce même algorithme, sans modifications, mettrais à jour rapidement le bon chemin.

Dans ce sujet vous aurez plusieurs Class Javascript à implémenter ainsi qu'un affichage en HTML/CSS a effectuer.

Avant de commencer

Sur une Map contenant un point de départ (la fourmilière) et une source de nourriture, des fourmis se déplacent.

Les fourmis n'ont pas *réellement* de cerveau : de fait leur capacité de calcul et de mémorisation est quasi nulle. Pourtant ces dernières sont capables de prouesses techniques en raison de leur grand nombre et de l'utilisation des phéromones qui sont des messages chimiques qu'elles déposent au sol.

"Un exemple classique de comportement collectif auto-organisé est l'exploitation des pistes de phéromones. Une fourmi seule n'a pas l'intelligence nécessaire pour choisir le plus court chemin dans un environnement complexe. De fait, c'est la colonie dans son ensemble (ou du moins les individus impliqués dans la recherche de nourriture) qui va choisir ce chemin."

- Wikipedia

Ainsi une fourmi arrivant sur une Case de la Map connaît la quantité de phéromones qui a été déposé et la met à jour - les fourmis déposant systématiquement des phéromones.

Les fourmis sont complètement *réactives*, c'est à dire qu'elles réagissent uniquement à cette quantité de phéromones. **Elles ne peuvent avoir des informations qu'uniquement sur les 4 cases autour d'elles et aucune autres cases de la Map.**

Pour le choix de déplacement vous aurez besoin de définir 4 variables heuristiques (vous devrez vous même définir les valeurs de ces variables pour adapter le comportement de la colonie).

Vous pouvez commencer avec ces valeurs :

- this.exploration = 0.8; // Degré de témérité de la fourmi et son goût pour l'exploration.
- this.confiance = 0.5; // Confiance dans le déplacement.
- this.evaporation = 0.9999; // Permet la modification du chemin en cas de changement.
- this.bruit = 0.7; // Explications détaillés dans le pdf fourni.

Les fourmis, en fonction de leur niveau de confiance vont choisir : soit d'aller à la case adjacente comportant le plus de phéromones, soit de se déplacer aléatoirement sur une des cases adjacentes libres.
Vous devez, grâce à une fonction mathématique, déterminer ce choix de déplacement.

Vous aurez plus d'informations sur ces variables et leurs fonctions dans ce pdf :
`https://hal.inria.fr/inria-00119238/PDF/article.pdf`

Cet article contient toute la logique et les fonctions mathématiques permettant de tout modéliser votre fourmilière. Vous devez mettre en application leur algorithme. Les 7 premières pages sont fondamentales.

Etape 1 : Travail de fourmis.

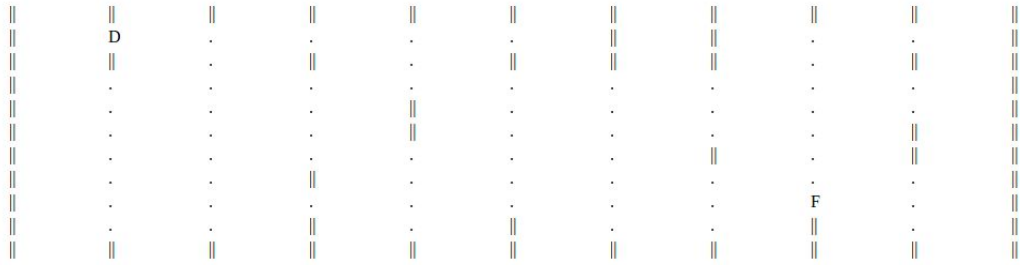
Rendu : ACO/index.html, ACO/Map.js, optionnel : ACO/style.css

Vous devez créer une Class Map(var height, var width) qui prend en paramètre deux entiers et qui génère une Map de taille height * width (ou 10x10 par défaut).

La map sera composé ainsi :

- Aucune ouverture sur l'extérieur, il y a des murs tout autour de la map.
- De manière aléatoire : des murs à l'intérieur de la map et zones vide sur lesquels les fourmis peuvent se déplacer.
- Une seule position "D" (0, 0) qui est le point de départ de toutes les fourmis.
- Et enfin une seule position "F" qui est la Food, l'endroit que les fourmis doivent rechercher qui est placé de manière aléatoire sur la map. La case Food à un attribut foodLeft qui est égal à 100 lors de l'instanciation.

Au final voici une map juste mais qui reste extrêmement basique :



Vous devez pouvoir modifier votre map en JS et les changements doivent être répercutés sur le front HTML.

Indice : Votre class Map peut contenir une class Case qui possède un attribut CASE_TYPE par exemple (et plus si nécessaire).

Etape 2 : Je suis une fourmi.

Rendu : ACO/ant.js

Écrire une Class Ant.
Chaque objet Ant représente une fourmi.

De fait, les fourmis ont des valeurs propres à chacune d'entre elles.

Elle possède plusieurs attributs :
Un attribut state qui peut être "EMPTY" ou "FULL", il représente le fait que la fourmi est chargée en nourriture (retour vers le nid) ou si elle est vide (cherche la source de nourriture).

- Un objet Case possédant au moins les attributs X et Y et représentant la position de la fourmi sur la Map.
- Un attribut phéromone qui vaut 0 par default.
- Les Getter/Setter de la class.
- ....

Il vous faut créer les méthodes permettant aux fourmis de se déplacer (et de faire un affichage de la Map). => L'affichage doit se faire via la classe Map.

Quelques exemples de méthodes utiles :

- getMovePossibility();
- moveRandPossible();
- moveToThisCase();
- majPheromone();
- pickFood();
- deliverFood();
- avgPheroScanNext();
- ...

Cette liste est fournie à titre indicatif, vous devez créer vos propres méthodes.

Etape 3 : La reine à faim.

Rendu : ACO/Anthill.js

Fonctionnement :

Les fourmis vont chercher la source de nourriture en se déplaçant de manière aléatoire et en laissant des phéromones sur leurs passages.

Quand elles trouvent la source de nourriture, elles récupèrent 1 de food sur le tas de nourriture puis elle repartent vers la position du nid.

Lorsqu'elles trouvent le nid, elles déposent la food et repartent dans l'autre sens ; et ce, jusqu'à ce que la source de nourriture soit épuisée (100/1 passages par défaut).

Vous devez obligatoirement envoyer plusieurs fourmis.
Une fourmi ne peut pas faire plus de 3 aller-retour.

Les fourmis mettent à jour les phéromones puis se déplacent, et ce, jusqu'à ce que la réserve de nourriture soit épuisée.

La fourmilière doit pouvoir exécuter un code semblable à celui-ci pour chaque fourmi :

```
while (type_of_this_case != "FOOD")
	{
		ant_colony[i].majPheromone(?);
		type_of_this_case = ant_colony[i].moving(?);
	}
ant_colony[i].pickFood();

while (type_of_this_case != "DEPARTURE")
	{
		ant_colony[i].majPheromone(?);
		type_of_this_case = ant_colony[i].moving(?);
	}
ant_colony[i].deliver();
```

Etape 4 : A pas de fourmi.

Rendu : Rendu : ACO/Ant.js, ACO/Anthill.js, ACO/Map.js, ACO/index.html, ACO/style.css,

Cette étape concerne le rendu final.

Lorsque vos fourmis ont épuisée la source de nourriture, vous devez afficher le chemin le plus court vers cette nourriture.
Pour cela, vous devez créer une fourmi qui a une confiance totale en ses pairs et qui va donc naturellement suivre le chemin contenant le plus de phéromones.

Ex:

- this.exploration = 1.1;
- this.confiance = 1.1;

Le résultat final peut ressembler à cet affichage :

‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
‖	**0.00 / 10.00**	**7.40 / 9.04**	8.11 / 8.19	7.13 / 6.95	‖	0.78 / 4.33	‖	.	‖	‖
‖	6.87 / 9.00	**8.76 / 9.00**	**9.10 / 8.86**	8.43 / 8.04	‖	1.13 / 5.94	0.00 / 3.60	.	.	‖
‖	1.30 / 7.63	8.56 / 8.82	**9.29 / 8.84**	9.30 / 8.70	8.47 / 7.93	7.29 / 7.18	.	‖	.	‖
‖	‖	8.49 / 8.13	**9.38 / 8.77**	**9.48 / 8.75**	9.33 / 8.61	8.48 / 7.85	‖	.	.	‖
‖	0.00 / 6.79	8.05 / 8.51	9.36 / 8.74	**9.52 / 8.74**	**9.55 / 8.71**	9.39 / 8.52	7.94 / 7.19	‖	‖	‖
‖	0.00 / 6.75	6.09 / 8.37	9.05 / 8.68	9.46 / 8.67	**9.59 / 8.68**	**9.61 / 8.60**	8.85 / 7.82	‖	5.86 / 4.29	‖
‖	‖	6.39 / 7.67	7.40 / 8.39	8.63 / 7.99	9.55 / 8.55	**9.70 / 8.55**	**9.73 / 8.30**	9.09 / 6.88	8.15 / 6.03	‖
‖	0.00 / 5.41	3.72 / 7.25	0.00 / 7.11	‖	8.83 / 7.78	**9.70 / 8.34**	**9.88 / 7.60**	**10.00 / 0.00**	8.93 / 5.03	‖
‖	0.00 / 4.89	0.00 / 6.00	0.00 / 5.83	5.78 / 5.83	7.89 / 7.02	8.78 / 7.51	8.99 / 6.88	9.00 / 5.69	7.64 / 4.78	‖
‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖

‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
‖	**PATH**	.	‖	.	‖	.	.	.	‖	‖
‖	**PATH**	‖	.	.	.	‖	‖	.	‖	‖
‖	**PATH**	.	‖	‖	‖	.	.	.	.	‖
‖	**PATH**	.	.	.	.	.	.	.	.	‖
‖	**PATH**	‖	‖	.	.	.	‖	.	‖	‖
‖	**PATH**	**PATH**	.	‖	.	‖	.	.	.	‖
‖	.	**PATH**	**PATH**	**PATH**	**PATH**	**PATH**	**PATH**	**PATH**	.	‖
‖	‖	.	‖	.	.	‖	.	**PATH**	.	‖
‖	.	‖	.	.	‖	.	.	.	.	‖
‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖

Légende :
En Bleu : La proximité par rapport au nid (noté en unité phéromone).
En Rouge : La proximité par rapport à la source de nourriture (noté en unité phéromone).
En Gras : Le chemin le plus court trouvé par la dernière fourmi.

Bonus: Où est le Prince ?

Cette étape concerne la partie bonus.

Il y a de nombreux bonus à réaliser et tous seront valorisé.

Vous pouvez tout aussi bien améliorer le code, le design de la map, les options pour lancer le programme...
Quelques idée de bonus :

- Permettre des changements dans la structure de la map de maniére dynamique.
- Une map avec des sprites
- Pouvoir modifier à la volé les variables heuristiques.
- Soyez imaginatif...

Bonne chance !