Rapport du projet de programmation

COLONIE DE FOURMIS

Membres du groupe (CD1A) :

- Amira Said
- Amira Yanis
- Cherifi Rayan
- Salhi Younes
- Sidi Ahmed Ahmed

Sommaire :

I - Introduction :

Dans le cadre du module « Projet de programmation », nous avons construit un algorithme de recherche du chemin le plus court basé sur le concept des colonies de fourmis comme indique le nom du projet. Comme observé dans la nature, les fourmis après avoir quitté le nid et trouvé de la nourriture en explorant au hasard les alentours, elles utilisent les phéromones pour marquer leurs passages sur le chemin retour afin d'indiquer que cette piste est valable aux fourmis sortantes du nid. En revenant au nid ces mêmes fourmis renforcent l'odeur des phéromones sur cette piste qui devient plus attirante tout en ayant des fourmis choisissant d'autres chemins mais au bout d'un certain temps/certain nombre de fourmis, on remarque que la quasi-totalité des fourmis emprunte le même chemin qu'on considère optimal (ou presque).

Nous avons donc implémenté ceci sous forme d'une interface dédiée à un utilisateur afin d'effectuer différentes simulations de ce phénomène.

II – Parties traitées du cahier des charges :

- **Modification des terrains, rajout des obstacles et enregistrement des Maps** : Ce point a été pris en charge dans la simulation puisqu'au lancement l'utilisateur a devant lui une fenêtre de paramétrage initiale (préremplie avec des valeurs par défaut) dans laquelle il peut modifier les valeurs pour ensuite pouvoir accéder au plateau.

Sur ce dernier l'utilisateur doit placer le nid de la colonie, et la nourriture, respectivement. Il pourra par la suite placer des obstacles soit en cliquant sur les cases voulues ou en glissant le curseur par-dessus plusieurs et par la suite lancer la simulation ; Il peut modifier le plateau en cours de cette dernière en retirant ou rajoutant des obstacles.

En haut de la fenêtre de simulation, l'utilisateur dispose de l'option de sauvegarder sa map actuelle ou bien charger une autre déjà créé.

- **Utilisation du principe de l'orienté objet :** ce dernier a bien été respecté, cependant notre projet n'a pas eu besoin d'une structure orientée objet trop compliqué mais le travail a plus été au niveau algorithmique et graphique.

- **Représentation des fourmis et des chemins choisis :** le plateau étant une matrice initialement de couleur blanche le dépôt de phéromones par les fourmis lors de leur passage est représenté par un léger changement de couleur qui s'intensifient selon la densité des phéromones sur les tuiles, allant graduellement du blanc au rouge, le chemin optimal étant par conséquent celui de couleur rouge vif. Quant aux fourmis elles sont représentées par une icône se déplaçant de case en case suivant le chemin de la fourmi.

- **Vision dynamique du programme avec la possibilité de mettre en pause :** En lançant la simulation on peut facilement suivre les différents chemins empruntés par les fourmis et en ayant la possibilité de mettre en pause celle-ci et ce grâce à l'option « pause » dans la barre d'outils en haut de la fenêtre et le remettre en marche en rappuyant. Mais aussi mettre en pause pour effectuer des modifications sur les paramètres ou bien simplement les vérifier à travers le bouton « paramètres », ou bien relancer une toute nouvelle simulation grâce au bouton « restart ».

- **Des informations lors du clic sur les cellules** : Oui ceci a été implémentée, en faisant un clic droit sur les tuiles (puisque le clic gauche est pour rajouter des obstacles) un pop-up contenant des informations diverses sur la cellule apparaît tel que le nombres de fourmis, le taux de phéromones et peut éventuellement être modifiée pour afficher d'autres informations.

II – Algorithmes et choix techniques :

On s'est inspiré de la documentation (assez riche) présente en ligne pour la conception de notre algorithme.

Déplacement des fourmis :

Pour accélérer le fonctionnement de la simulation nous avons choisi d'utiliser le concept de multi-threading pour les fourmis, et ceci rends le comportement de chaque fourmi indépendante. Au lancement de la simulation toutes les fourmis sortent de la colonie aléatoirement à une vitesse qu'on peut modifier dans les paramètres et donc, une fourmi k placée sur la cellule i à l'instant t va choisir sa tuile j de destination en fonction de la visibilité nij de cette tuile et de la quantité de τij phéromones déposée sur la tuile adjacente. Ce choix est fait aléatoirement, avec une probabilité de choisir la tuile j calculée par :

$$P_{i,j}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{l \epsilon N_i^k(t)} [\tau_{il}(t)]^{\alpha} \cdot \eta_{il}^{\beta}} & si\ j \in N_i^k \\ 0 & sinon \end{cases}$$

Pour permettre aux fourmis d'explorer d'autres chemins possible et donc éviter une solution certes rapide mais pas optimale, aucun chemin n'est inondé de phéromones et aucun chemin ne devra être totalement invisible, on peut donc aussi contrôler le niveau de phéromone de chaque chemin pour le maintenir entre des bornes

minimum (phero_min) et maximum (phero_max) et pour éviter le parcours de vieux chemins nous avons rajouté deux paramètres du système : taux d'évaporation et temps d'évaporation pour rendre les vieux chemins (plus longs) moins visibles.

Nos fourmis sont aussi programmées en mémorisation, celles-ci se rappellent leurs parcours de manière efficace pour éviter d'osciller entre deux cellules, et déposer des phéromones au retour quand elles auront trouvé de la nourriture. Une fois cela est fait, la fourmi est réinitialisée et recommence le même procédé.

Paramètres des déplacements : Quand la valeur du paramètre alpha est grande on a moins de fourmis qui s'éloignent des traces de phéromones, quant au beta, une grande valeur de beta pousse les fourmis à prendre des chemins de plus en plus droits, et d'éviter de passer par la diagonale.

Le taux d'évaporation influence la vitesse de convergence, car les fourmis perdent rapidement les traces de phéromones et sont donc obliger à recommencer jusqu'au point on le nombre des fourmis qui arrive à la destination est grand, pour que les phéromones s'accumulent et ne disparaissent pas avant. Cela est inversement vrai pour le temps d'évaporation, les fourmis ne peuvent distinguer entre l'ancien et le nouveau chemin, car la densité des phéromones est

élevée et ne diminue pas assez vite pour distinguer le nouveau chemin.

Ces paramètres (taux d'évaporation – temps d'évaporation – vitesse des fourmis – alpha – beta) ont des valeurs optimales qu'on a essayé de trouver et mettre par défaut mais on s'est vite rendu compte qu'elles peuvent différer selon la taille du plateau, à l'utilisateur donc de trouver des valeurs optimales pour les différents plateaux qu'il va créer.

Dépôt des phéromones :

Au début nous avions mis le dépôt des phéromones tel que la quantité déposée sur chaque tuile est calculé de la sorte :

$$Q = \frac{Quantité}{longueur\ du\ chemin^{a}}$$

Mais cela étant contre nature (la fourmi ne sait pas faire de division euclidienne) nous avons mis en place une quantité fixe qui est déposée sur chaque tuile du chemin.

IV – Problèmes rencontrées :

Programmation :

- Problème 1 : Que les fourmis évitent les anciens chemins et suivent le chemin trouvé plus récemment. On a pu régler ça grâce au système d'évaporation des phéromones.

- Problème 2 : Eviter que les fourmis s'éloignent trop du parcours et qu'elles ne trouvent pas de chemins. On a pour cela ajouter le fait que les fourmis peuvent mourir (elles sont réinitialisées) et sont mieux orientées grâce au alpha et beta.

-Problème 3 : Gérer l'exécution parallèle des threads en ce qui concerne l'accès a la mémoire, et on a pu arranger ça avec les variables volatiles de Java.

-Problème 4 : Trouver un juste milieu pour les différents paramètres de l'algorithme (taux d'évaporation, temps de déplacement des fourmis, temps d'évaporations …).

-Problème 5 : L'affichage du taux de phéromones de la tuile qui est géré avec l'intensification d'une couleur (rouge).

Hors programmation :

Pour ce qui s'agît de l'organisation, nous avons utilisé l'outil git. Mais ce dernier a causé aussi quelques soucis étant des novices à l'utilisation de cet outil. Le projet n'est pas celui ou le partage des taches est facile car on a eu deux gros morceaux, la conception de l'algorithme et l'interface graphique. On a commencé par la première et celle-ci nous a pris du temps car nous avons changé la structure de données plusieurs fois en suivant les propositions de chacun avant de décider finalement une structure finale. En travaillant souvent sur les mêmes fichiers nous avons à maintes reprises fait face à des conflits sur git qu'on a pu finalement arranger.

V– Représentation graphique du modèle des classes