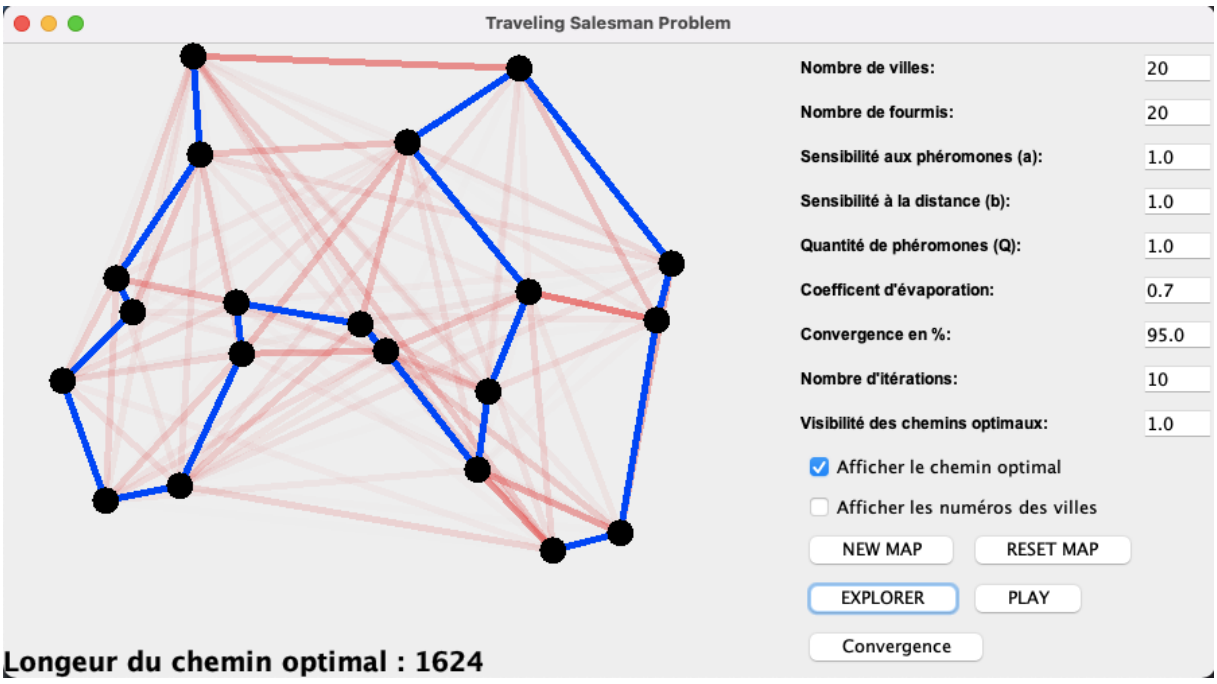
GUIDE DE L'ALGORITHME DES FOURMIS SUR LE PROBLÈME DU VOYAGEUR DE COMMERCE



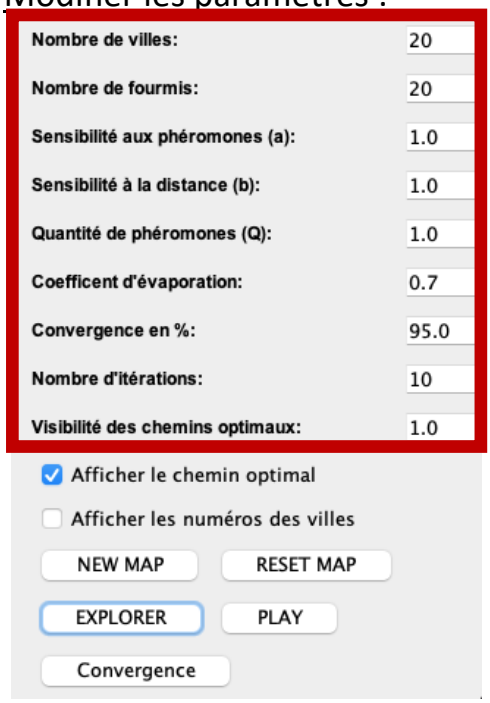
Damien Planchamp
Bourget du Lac

Table des matières :

Utilisation de l'interface

L'interface du programme permet de modifier tous les paramètres afin de laisser à l'utilisateur le pouvoir d'effectuer tous les tests possibles sans à avoir modifier le code du programme.

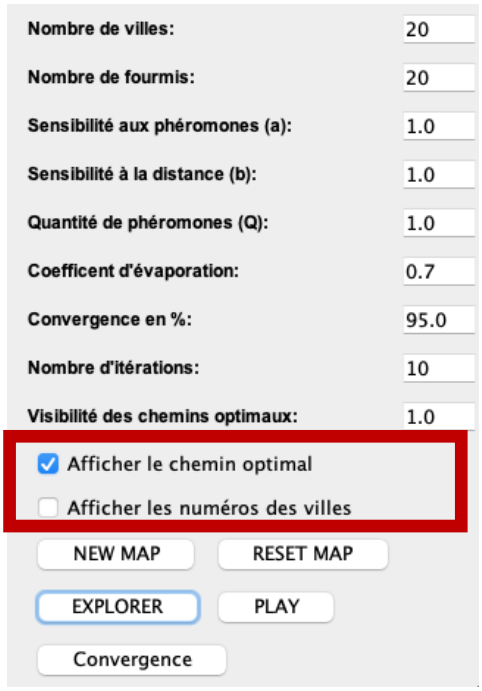
Modifier les paramètres :



Dans cette section, l'utilisateur peut modifier tous les paramètres relatifs à la recherche des chemins optimaux.
Si aucune valeur n'est inscrite, le programme utilisera la valeur inscrite précedement.

- Convergence : cherche les chemins ayant une longueur avec une correspondante égale ou supérieure.
- Nombre d'itérations : Nombre de fois où on répete l'action explore du modele(voir explorationAction p8) et aide à la convergence(voir convergenceAction p8).
- Visibilité des chemins optimaux : met en avant les routes avec le plus de phéromones.

Modifier l’affichage :



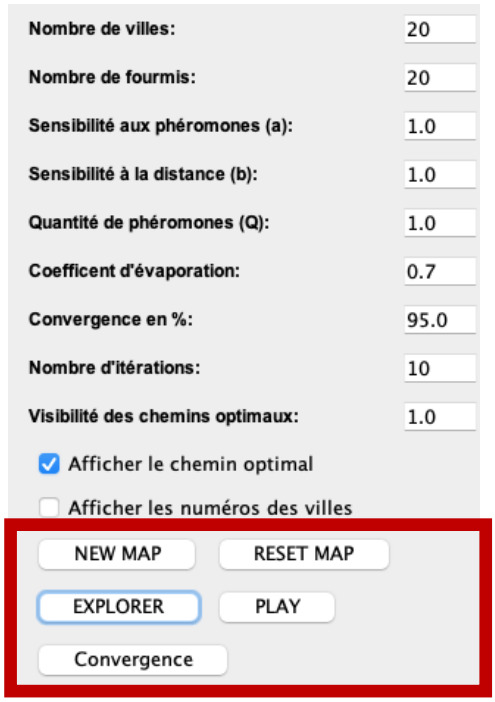
Dans cette partie, il est possible de modifier l’affichage du terrain.
Par défaut on affiche tous les chemins que les fourmis ont parcourus et le meilleur chemin basé sur la quantité de phéromones.

- La visibilité des chemins optimaux correspond aux trajets que les fourmis ont aléatoirement pris, ils sont représentés en rouge avec une transparence équivalente à la quantité de phéromones. Plus il y en a, plus le chemin sera visible.
  La valeur de la visibilité permet de mettre en avant les meilleurs chemins, plus cette valeur est élevée plus on verra les meilleurs chemins.

- La checkbox « Afficher le chemin optimal », permet d’afficher en bleu le meilleur chemin reliant toutes les villes basées sur la quantité de phéromones.

- La checkbox « Afficher les numéros des villes » permet d’afficher leur numéro sur elles.

Utilisation des boutons :



Chacun de ses boutons a un usage spécifique, cela permet à l'utilisateur d'effectuer certaines actions plus rapidement.

- NEW MAP : Ce bouton permet de configurer une nouvelle carte avec les paramètres inscrits plus haut

- RESET MAP : Ce bouton permet de réinitialiser le terrain en gardant la même structure des villes

- EXPLORER : Ce bouton permet de lancer la recherche des fourmis, il effectue autant de recherches que d'itérations (paramètre Nombre d'itérations).
  De plus il stocke la meilleure combinaison pour trouver le meilleur chemin.

- PLAY : Tant que ce bouton est appuyé, on effectue une exploration. A la différence du bouton EXPLORER, on voit l'évolution du terrain.

- Convergence : Cherche un chemin optimal en fonction des paramètres Convergence et Nombre d'itérations. Si la recherche prends trop de temps, la recherche s'arrete et affiche le message «Pas de réponse ».

Documentation du code :

Cette partie explique l'utilisation des méthodes complexes de la classe Area.
Toutes les fonctions Get et Set, ont uniquement un usage graphique que l'on ne développera pas.

On expliquera également la méthode exploreAction et convergenceAction de la classe ControlPanel

La méthodologie utilisée est basé sur la thèse « Algorithmes de fourmis artificielles : applications à la classification et à l'optimisation » de Nicolas Monmarché (2000) https://tel.archives-ouvertes.fr/tel-00005186/document

Les Objets Ant, Road, City du package Ressource, sont des objets pour faciliter l'usage du modèle (Area), ils ne sont pas complexes.
Il faut juste savoir que :

- Ant contient une liste de City qu'il a traversé et une liste de Road qu'il a pris.
- Road contient deux City (CityA et CityB), sa longueur, sa quantité de phéromones, et ses coordonnées
- City contient son nom (c'est son id) et ses coordonnées.

Area :

Cette classe est le modèle de notre code.
Elle contient donc :

- la liste des fourmis (colony)
- la liste des villes (cities)
- la liste des routes (map)
- et toutes les constantes nécessaires pour les calculs de l'algorithme des fourmis

La méthode exploration() :

Elle permet de créer tout les chemins possibles avec leurs phéromones

Fonctionnement :

- Indique qu'on commence à afficher des routes (beginning = false)
- Pour chaque fourmi on effectue les taches suivantes :
  - on récupère la fourmi
  - on choisit une ville au hasard pour placer la fourmi
  - on effectue une boucle allant au nombre de routes nécessaires pour avoir un chemin reliant toute les villes (cities.size -1)
    - on récupère la route choisie par la fourmi
    - on l'ajoute à sa mémoire
    - on déplace la fourmi vers l'autre côté de la route
- On met à jour les phéromones sur les routes
- On supprime la mémoire des fourmis pour pouvoir recommencer
- On met à jour l'affichage

La méthode getPossibleRoad(Ant ant, City city) :

Elle renvoie tous les chemins que la fourmi peut prendre à partir de la ville passée en paramètre

Fonctionnement :

- On récupère tous les chemins possibles depuis la ville
- On élimine les chemins qui mènent vers des villes que la fourmi a déjà traversées

La méhode getRoad(ArrayList<Road> roadPossible) :

Choisit une route parmi la liste des routes passées en paramètre.
Le choix est aléatoire, la probabilité d'une route est :

$$p_{ij} = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\nu_{ij}]^{\beta}}{\sum_{l \notin L_k(i)} [\tau_{il}(t)]^{\alpha} \cdot [\nu_{il}]^{\beta}} & \text{si } j \notin L_k(i) \\ 0 & \text{sinon} \end{cases}$$

Avec :

- Tij la quantité de phéromones de la route
- α la sensibilité au phéromone
- Vij la longueur de la route
- **β** la sensibilité à la longueur
- Σ la somme du calcul précédent

La méthode addingPheromone() :

Elle rajoute sur chaque route la quantité de phéromones définit par

$$\tau_{ij} \leftarrow \tau_{ij}(1-\rho) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

Avec :

- Tij la quantité de phéromones de la route
- ρ la constante d'évaporation

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L^k} & \text{si la fourmi } k \text{ est passée par l'arc } (i,j) \\ 0 & \text{sinon} \end{cases}$$

Avec :

- Q la quantité de phéromones
- Lk la longueur du chemin (chemin = suite de routes)
-

La méthode foundTheBestPath() :

Renvoie le meilleur chemin que les fourmis ont trouvé.
On utilisera une nouvelle fourmi qui partira d'une ville où il y a une route avec le plus de phéromones.
Puis elle choisira la prochaine route la plus riche en phéromones, où elle continuera jusqu'à qu'elle l'ait fait un chemin reliant toutes les villes.

ControlPanel :

Cette classe permet de contrôler notre modèle (Area). C'est également elle qui, par l'intermédiaire d'Area, chercher le meilleur chemin parmi toutes les combinaisons de structure d'Area (chaque nouvelle exploration change la structure d'Area).

exploreAction() :

Cette méthode va appeler area.explore() autant de fois que d'itérations.
Elle compare à chaque fin d'itération la structure d'Area avec la meilleure structure trouvée jusqu'à maintenant.

Fonctionnement :

- Recupere le nombre d'itération
- Initialise la variable bestLenght (elle retiendra la meilleur longueur)
- Initialise l'objet bestArea de type Area (elle retiendra la meilleur configuration)
- On effectue une boucle allant au nombre d'itération
  - On explore
  - On donne à bestLenght et bestArea leur premiere valeur (uniquement une fois)
  - On compare si la longueur du nouveau chemin est meilleur que celle de bestLenght
  - Si oui
    - on donne à bestLenght la longueur du nouveau meilleur chemin
    - on stocke la structure d'area
- On donne à area la strucure de bestArea

convergeAction() :

Cette méthode va chercher le chemin avec les paramètres Convergence et Nombre d'itération.
Elle effectue une exploration et compare le chemin précédant avec le nouveau si ça correspond en termes de pourcentage de convergence.
Elle vérifiera les N chemins suivants (N étant le Nombre d'itérations).
Elle continuera sa recherche jusqu'à avoir remplis cette condition.
Remarque : pour fluidifier le programme, un compteur est mis en place pour éviter une boucle infini

Fonctionnement :

- On effectue une boucle jusqu'à qu'on trouve une correspondante
    - On explore
    - On calcule la convergence du nouveau chemin (calculConvergence)
    - Si calculConvergence est supérieur ou égale à la convergence de l'utilisateur
        - On compte qu'il y a une correspondante en plus d'affilé
    - Sinon
        - On remet notre compteur à 0

CONCLUSION :

L'algorithme des fourmis était très intéressant développer, malgré le temps de recherche pour comprendre son fonctionnement.

La partie la plus compliquée a été la réalisation des algorithmes pour choisir un chemin, car il fallait bien comprendre comment fonctionnait les foncions mathématiques.

D'une certaine manière, ce projet m'a fait penser à du développement d'IA, car on y retrouve la notion de neurones (les villes ou City), de poids (les routes ou Road), et de solutions (le chemin ou bestPath). Il manque la notion d'apprentissage pour avoir un véritable réseau neuronal.