UNIVERSITE
DE LIMOGES

FACULTE DES SCIENCES
ET TECHNIQUES

Faculté des Sciences
& Techniques

INTELLIGENCE
ARTIFICIELLE

Filière : Master 1 ACSYON/CRYPTIS

Problème du Voyageur de commerce

Rédigé par :

Rami MECHI

Mohamed Hedi KALAI

Cleque Marlain MBOULOU

Professeur :

Karim TAMINE

Année académique 2022-2023

TABLE DES MATIÈRES

CHAPITRE

1

PROBLÈME DU VOYAGEUR DE COMMERCE

Le problème du voyageur de commerce (TSP en anglais) est l'un des problèmes d'optimisation les plus connus et les plus étudiés en informatique et en mathématiques. Il a des applications dans divers domaines tels que la logistique, la planification de tournées, la conception de circuits électroniques, la bio-informatique, etc.

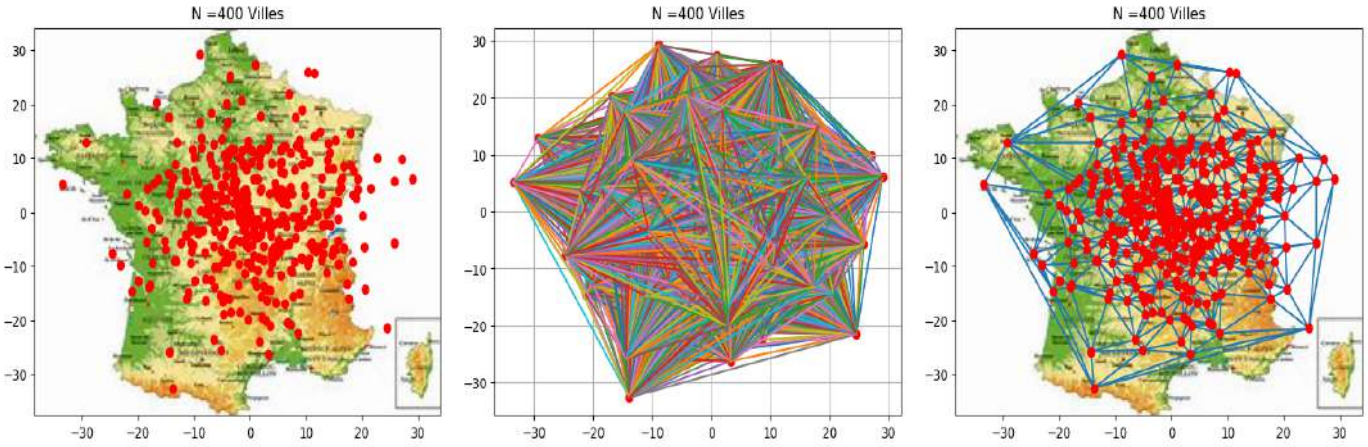
Le problème consiste à trouver le plus court circuit hamiltonien (un circuit passant une fois par chaque ville) dans un graphe complet pondéré. Le graphe représente les villes et les poids des arêtes représentent les distances entre les villes. Le but est de trouver le circuit qui minimise la somme des distances entre les villes visitées. Le problème devient rapidement complexe lorsque le nombre de villes augmente, car le nombre de circuits possibles croît de façon exponentielle.

Dans ce projet, nous allons étudier deux approches pour résoudre le problème du voyageur de commerce : **un algorithme de colonie de fourmis** et **un algorithme génétique**

1.1 Générations des données

Afin d'évaluer la complexité du problème lorsque le nombre de villes augmente, nous avons commencé par générer les coordonnées de chaque ville. Ensuite, nous avons calculé les distances entre chaque paire de villes en utilisant une matrice M symétrique à diagonale nulle, en utilisant la distance euclidienne.

Pour visualiser l'impact de l'augmentation du nombre de villes sur le problème, nous avons créé les affichages suivants :



Pour implémenter les algorithmes de colonies de fourmis et génétiques, nous avons créé une population P de la taille souhaitée, représentant le nombre d'individus pour chaque algorithme. Cette population est constituée de chromosomes pour l'algorithme génétique et de fourmis pour l'Algorithme de colonies de fourmis.

Distance parcouru par cycle

Nous avons créé une fonction nommée **dist_trajet()** qui calcule et renvoie une liste de distances pour chaque trajet (cycle) parcouru par chaque individu (chromosome ou fourmi) dans une population, en utilisant une matrice de distances en entrée.

1.2 Algorithme de colonies de fourmis

L'algorithme fonctionne en simulant la recherche de nourriture des fourmis. Chaque fourmi part d'une ville aléatoire et visite les villes suivantes en choisissant à chaque fois la ville suivante en fonction de la probabilité associée à chaque ville. La probabilité de choisir une ville dépend de la distance entre la ville actuelle et la ville candidate, ainsi que de la quantité de phéromone déposée sur le chemin menant à la ville candidate. Plus la distance est courte et plus la quantité de phéromone est grande, plus la probabilité de choisir cette ville est élevée. Un modèle expliquant ce comportement est le suivant :

- Les éclaireuses parcourent plus ou moins au hasard chaque ville une et une seule fois pour générer la population P
- Si une éclaireuse ont parcouru toute les villes, elle rentre plus ou moins à la ville de départ, en laissant sur son chemin une piste de phéromones.
- Les phéromones étant attractives, les fourmis passant à proximité ont tendance à suivre, de façon plus ou moins directe, cette piste.
- En revenant sur la ville initiale, ces mêmes fourmis renforcent la piste en déposant plus de phéromones.
- Si deux pistes sont possibles pour atteindre la même la ville suivante, la plus courte sera parcourue par plus de fourmis que la longue piste.

- La piste courte sera donc de plus en plus renforcée, et donc de plus en plus attractive.

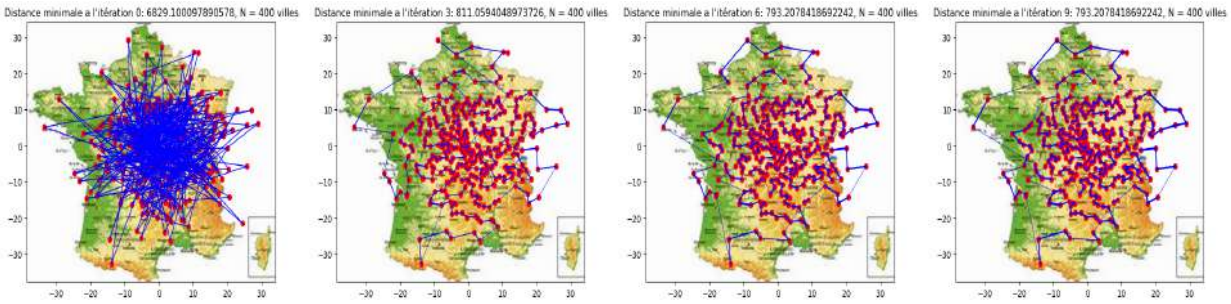
Dans la fonction **pheromone()**, une fois les hyperparamètres initialisés, nous calculons la quantité de phéromone qui sera présente sur chaque piste en prenant en compte le processus d'évaporation des phéromones. Ce calcul est effectué en utilisant les formules définies dans le document donné, présent également sur wikipédia.

Probailité de choix d'une ville non-visitée

La probabilité de visiter une ville$_{i+1}$ à partir d'une ville$_i$ est déterminée en fonction de trois facteurs : la visibilité de la ville$_{i+1}$ depuis la ville$_i$, la quantité de phéromone présente sur la piste reliant i et $i+1$, ainsi que deux hyperparamètres α et β qui pondèrent l'importance relative de chaque facteur (visibilité et quantité de phéromone).
Afin d'introduire un peu de hasard, nous utilisons pour le choix la ville$_{i+1}$, procédé de sélection par roulette.

Affichage

Dans une fonction **affichage()** nous affichons le meilleur trouvé après chaque itération de notre algorithme, ainsi que sa distance. On effectue un teste pour une population de 400 individus et nous avons les result at suivant :



On peut voir que ici l'algorithme converge très bien vers une solution optimale, passant de 6829.100097890578 à 824.9225086882711, à la première itération puis passe à 793.2078418692242 à la sixième itération.

La forte diminution de la distance minimale du meilleur trajet à chaque itération s'explique par le choix des hyperparamètres α et β. Nous avons remarqué qu'en choisissant α grand et β petit, les résultats étaient nettement meilleurs. Cela signifie que la visibilité des villes voisines est plus importante que la quantité de phéromones déposée sur la piste. Cependant, cette observation reste à approfondir.

1.3 Algorithme Génétique

L'algorithme génétique pour le problème du voyageur de commerce (TSP) est un algorithme évolutionnaire qui utilise des techniques d'évolution biologique pour résoudre le problème de l'optimisation de la distance parcourue par le voyageur de commerce.

Le processus commence par la génération d'une population initiale de solutions aléatoires, représentées sous forme de chromosomes comme dans les étapes précédentes. Chaque chromosome représente une séquence ordonnée de villes visitées par le voyageur. La population est ensuite évaluée en calculant la distance totale parcourue par chaque chromosome à l'aide de la fonction **dist_trajet()** utilisée précédemment. Les individus à croiser sont sélectionnés par roulette, de manière à obtenir des enfants améliorés. Des mutations sont ensuite appliquées aux enfants pour améliorer davantage leur qualité. Les enfants obtenus sont alors insérés dans la population de base, et le processus est répété jusqu'à ce qu'une condition d'arrêt soit atteinte, telle qu'un nombre prédéfini d'itérations.
Voici les resultats pour une population de 15 individus :

L'algorithme marche bien, car nous pouvons remarquer la diminution de la distance minimale an fontion du nombre d'itérations. Par contre, nous pouvons remarqué que celle-ci va beaucoup moins vite que celle de l'algorithme génétique.

1.4 Comparaison des deux algorithmes

Pour comparer les deux algorithmes, nous avons décider d'évaluer la distance minimale trouver pour un nombre d'itération fixé et une taille de population croissante. Nous évaluons aussi le temps mit par chaque algorithme en fonction de la taille de la population.

Comparant les deux algorithmes pour une même population de 15 individus :

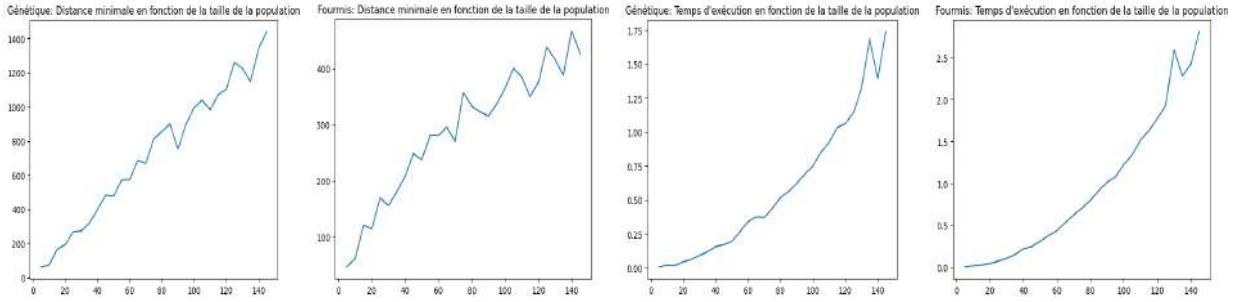
Algorithme génétique


```
Un chemin optimale trouvé par l'algorithme génétique est: [8, 7,
    0, 10, 11, 14, 5, 9, 1, 6, 12, 13, 2, 3, 4]
Un trajet optimale est long de 110.17714661418748 Km
Nombre d'itération:   10
Temps d'excétution:  3.945485830307007 secondes
```

Algorithme de colonie de fourmis


```
**********************************************************
Un chemin optimale trouvé par l'algorithme a base de fourmis est:
   [8, 7, 4, 0, 5, 14, 11, 10, 9, 1, 2, 3, 13, 12, 8]
Un trajet optimale est long de 87.78080484982941 Km
Nombre d'itération:   10
Temps d'excétution:  3.6722428798675537 secondes
**********************************************************
```

Comparaison des deux algorithmes pour une taille de population croissante : temps et distances minimales trouvées



Dans le cas où la population est de taille $N = 15$ les deux algorithmes s'approchent de la solution optimale, bien que l'algorithme de colonie de fourmis semble trouver la solution optimale en un meilleurs temps d'exécution.

En considérant des tailles de populations croissantes, l'algorithme de colonie de fourmis est largement meilleur quand les tailles deviennent très grandes, en terme d'approche de la solution optimale plus vite.

1.5 Conclusion

En conclusion, le problème du voyageur de commerce est un défi passionnant de l'optimisation, pour lequel il n'existe pas d'algorithme optimal connu en temps polynomial pour trouver une solution exacte dans tous les cas. Cependant, dans ce projet, nous avons pu concevoir et implémenter deux méthodes d'optimisation pour calculer une solution au problème. La première méthode, basée sur les algorithmes génétiques, utilise des concepts tels que le codage du chromosome, le croisement, la mutation et la sélection pour trouver

une solution approchée. La seconde méthode, basée sur le principe d'une colonie de fourmis, repose sur le comportement des agents fourmis (déplacement, dépôt de phéromone, évaporation, etc) pour trouver une solution approchée également. En implémentant ces deux méthodes et en testant leur performance sur plusieurs cas d'exemples de graphes, nous avons pu comparer les résultats et déterminer leur efficacité en fonction de la taille du graphe. Cela montre que ces méthodes offrent des approches intéressantes pour résoudre le problème du voyageur de commerce et peuvent être utilisées dans diverses applications de planification de tournées.

BIBLIOGRAPHIE

[1] **Algorithme de colonies de fourmis** [en ligne]

[2] **Algorithme génétique** [Youtube]

[3] **Algorithme génétique** [Youtube]