# Métaheuristique pour l'optimisation TP4 120 crédit

Deniz Sungurtekin

November 2020

## 1 Introduction

L'objectif de ce travail pratique est de nous familiariser avec l'algorithme "Ant System" (AS) en résolvant le problème de "Traveling Salesman". Cette méthode s'inspire des comportements des fourmis pour trouver le chemin le plus court de leur nid à une source de nourriture en utilisant leurs phéromones afin de guider l'essaim. L'efficacité de cette méthode réside dans sa composition de multiple individus, qui grâce à leur interaction forme un système complex très robuste et adaptatif. Dans un premier temps, nous allons expliquer avec plus de détail le fonctionnement de l'algorithme. Ensuite, nous analyserons l'influence de ses paramètres, puis nous comparerons cette méthode à celle de l'algorithme Greedy et "Simulated Annealing" afin d'observer les temps d'exécutions et la qualité des résultats obtenus.

## 2 Méthodologie

L'idée de la méthode consiste à simuler le passage des fourmis à travers nos villes. Initialement, chaque fourmi fait le tour des villes de manière plus ou moins aléatoire en déposant des phéromones sur les chemins utilisés. Puisque la fourmi utilisant le chemin le plus court sera la première à effectuer le chemin du retour, ce chemin possédera plus de phéromone est aura par la suite plus de chance d'être choisi par d'autre fourmi. On aura donc, après un certain nombre d'itération, un système qui converge vers le chemin le plus court. Pour simuler cela, une fourmi k aura une certaine probabilité d'aller d'une ville i à j:

$$p_{ij}^{k}(t) = \begin{cases} \frac{(\tau_{ij}(t))^{\alpha}(\eta_{ij})^{\beta}}{\sum_{l \in J}(\tau_{il}(t))^{\alpha}(\eta_{il})^{\beta}} & \text{if } j \in J \\ 0 & \text{otherwise} \end{cases}$$

Où J est l'ensemble des villes non-visité par la fourmi k, $\eta_{i,j}$ est l'inverse de la distance entre la ville i et j, $\alpha$ et $\beta$ des paramètres de contrôle qui gère respectivement l'importance des phéromones et de l'information heuristique.

Finalement $\tau_{i,j}$ représente le taux de phéromone associé au chemin entre la ville i et j et se met à jour de la façon suivante:

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

où $\rho$ est le taux d'évaporation de la phéromone et $\Delta\tau_{i,j}^{k}(t)$ la quantité de phéromone déposée par la fourmi k:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L^k(t)} & \text{if ant } k \text{ used edge } (i,j) \text{ in its tour} \\ 0 & \text{otherwise} \end{cases}$$

Où Q est une constante donnée par une solution obtenue par l'algorithme Greedy et $L^k$ la longueur du tour construit par la fourmi k.

On veut donc reproduire cela en effectuant un certain nombre d'itération jusqu'à obtenir le plus court chemin tracé.

# 3 Résultat

Voici un exemple des résultats obtenus avec le fichier "cities.dat" et "cities2.dat":

cities.dat:

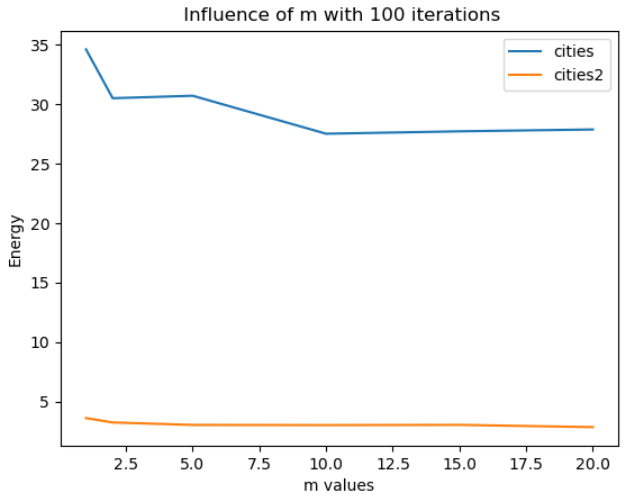
```
Corresponding Path:Energy,  ([16, 11, 4, 17, 10, 14, 6, 15, 12, 8, 5, 7, 2, 0, 1, 13, 9, 3], 27.515362563425345)
```

cities2.dat:

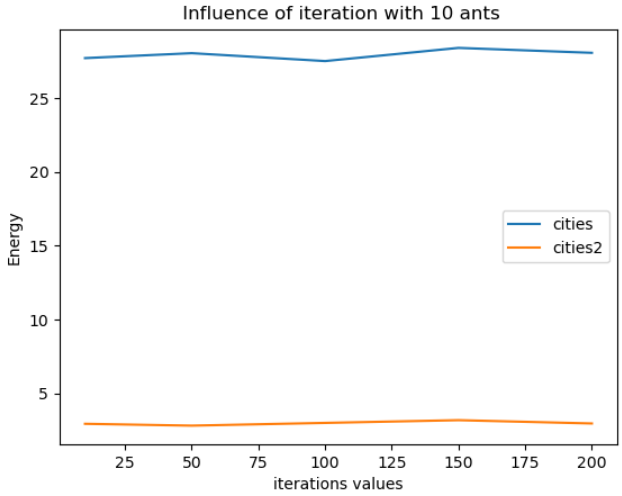
```
Corresponding Path:Energy,  ([34, 42, 46, 44, 41, 39, 33, 35, 31, 29, 26, 17, 22, 24, 20, 16, 14, 11, 9, 6, 18, 27, 30, 32, 23, 25, 36, 38, 28, 15, 19, 8, 12, 40, 43, 45, 48, 47, 4, 3, 5, 1, 2, 0, 7, 10, 13, 37, 21], 2.91084762405
```

Où chaque ville possède un id commençant par 0.

Analysons maintenant l'influence du nombre de fourmis, m sur la qualité de nos solutions:

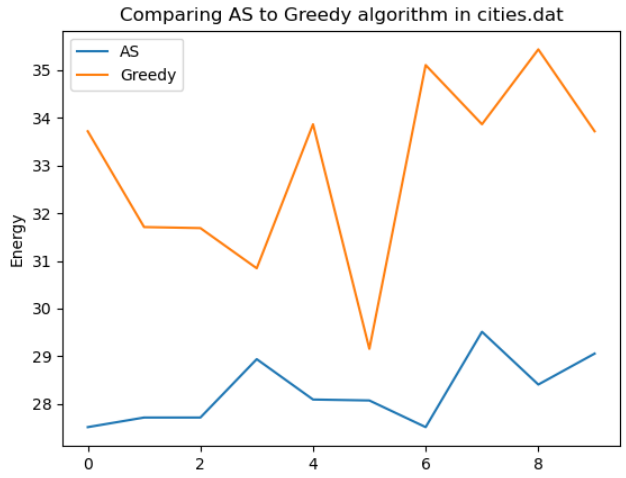



Dans notre cas, on observe que après m = 10, il n'y a pas réellement de progrès dans la qualité des solutions. C'est pourquoi je fixerai m = 10 pour analyser l'impact du nombre d'itération.

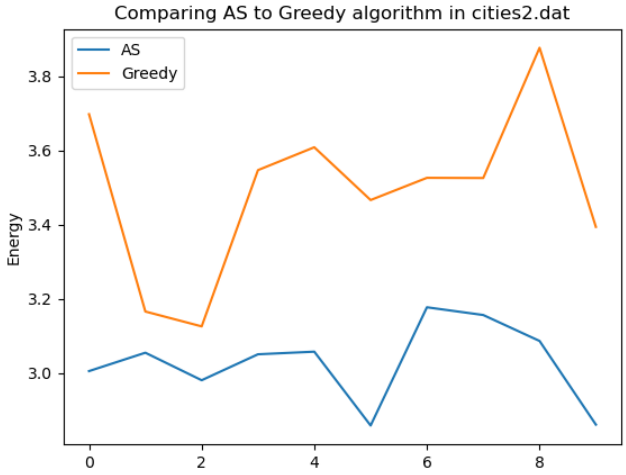



Étonnamment, le nombre d'itération ne semble pas vraiment influer sur la qualité des données puisqu'on obtient des meilleurs résultats avec 10 itérations. Ceci est sûrement dû à la faible taille du problème puisque celui-ci ne nécessite pas forcément beaucoup d'itération pour trouver un chemin optimal.

Comparons désormais les résultats obtenus sur 10 résolution du problème avec Greedy et AS. Voici donc la distance des chemins obtenus à chaque itération: Avec le fichier cities.dat:




Avec le fichier cities2.dat:

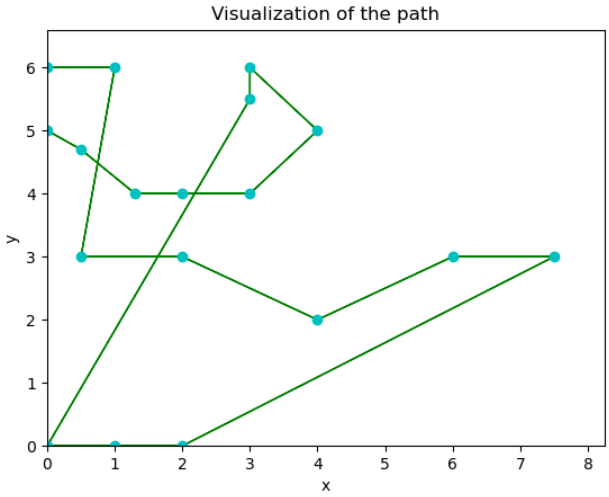



On remarque clairement un meilleur bilan pour AS, qui se montre beaucoup plus stable dans la qualité de ses résultats. Cela s'explique par le fait que la méthode Greedy dépend fortement de la configuration de son état initial pour obtenir un chemin "optimal", puisque celui-ci ne fait que prendre le voisin le plus proche en partant de la première ville qui lui a été assignée.

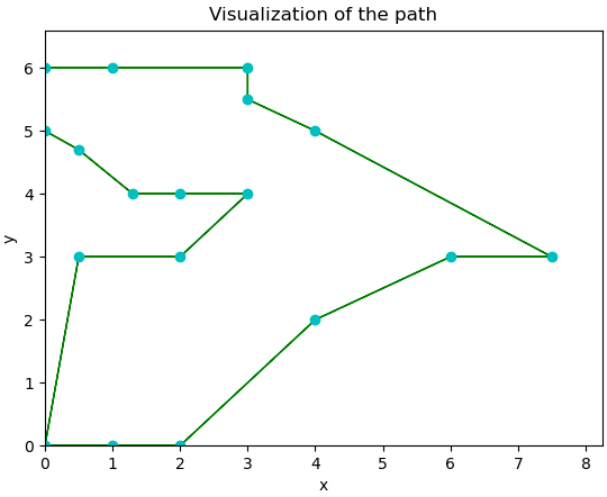
Voici maintenant le résultat et la visualisation des meilleurs chemins obtenus par Greedy et AS:

**cities.dat:**

Greedy:

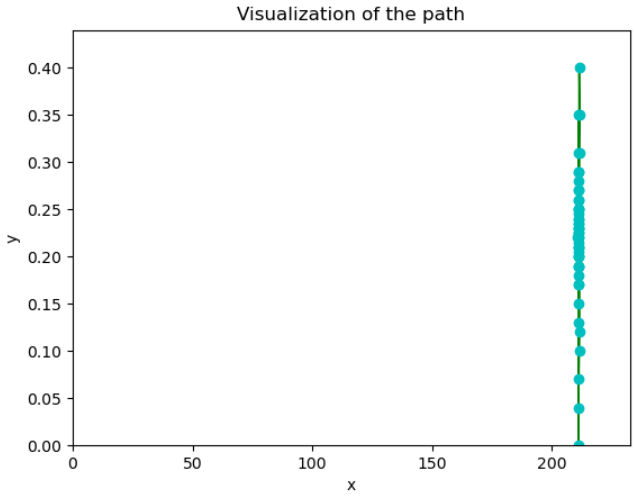



AS:

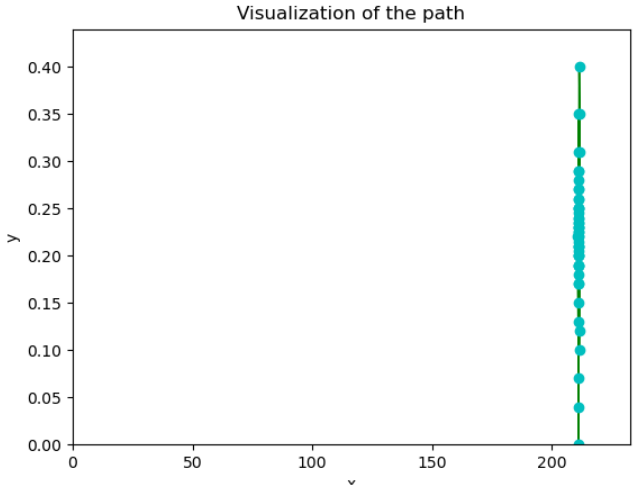


**cities2.dat:**

Greedy:

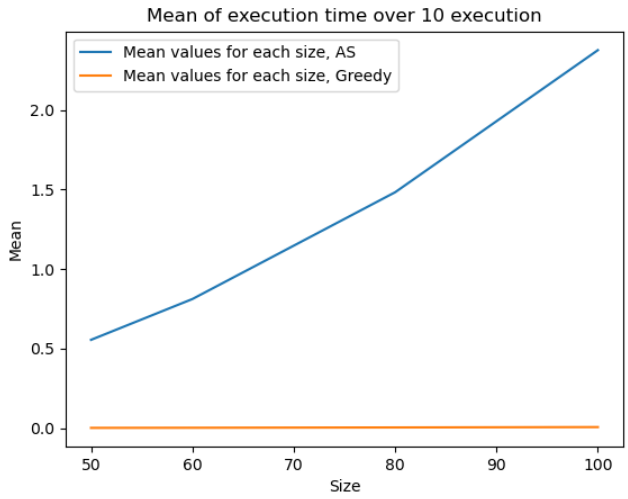


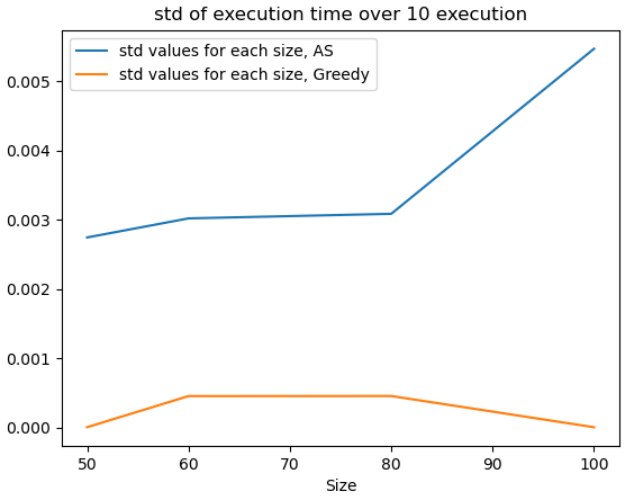

AS:




Pour le fichier cities.dat, on visualise clairement que le chemin obtenue par AS est plus optimale que celui de Greedy. Cependant pour cities2.dat, il est beaucoup plus difficile d'observer cela, puisque les coordonnés en x de nos villes ne varient que très peu.

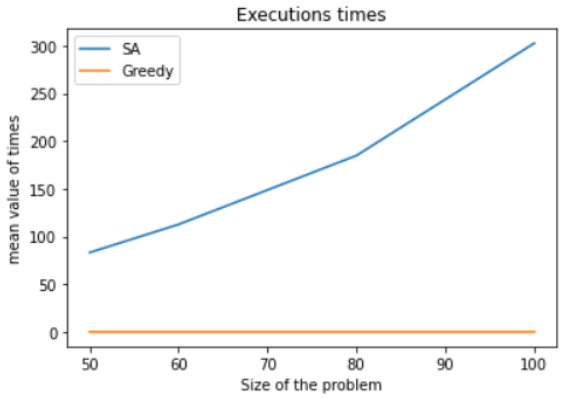
Observons désormais les temps d'exécutions de nos algorithmes pour différentes tailles du problème: 50,60,80 et 100:




On remarque clairement que le temps d'exécution pour AS croît linéairement alors que celui de Greedy reste négligeable pour les tailles données. Cependant, le temps d'exécution de AS reste minime en sachant que la qualité des résultats est nettement meilleur.

Dès lors. il nous ai finalement possible de comparer AS et SA:




On distingue que le temps d'exécution de SA est énorme par rapport à celui de AS. De plus, la qualité des résultats est meilleur pour AS puisqu'on obtient sur 10 itérations une valeur moyenne d'énergie de 28,36 contre seulement 29,45 pour le fichier cities.dat et une valeur moyenne de 2,98 contre 3,04 pour cities2.dat.