Metaheuristics for Optimization
Series 4: Ant System and Traveling Salesman Problem

Romain Mencattini

7 novembre 2016

Table des matières

1 Le voyageur de commerce

Il s'agit du problème sur lequel nous allons tester l'algorithme du recuit simulé. Détaillons le de manière mathématique.
Nous avons un ensemble de villes : V. Où $v = \{x, y\} \in V$. est une ville qui avec une coordonnée x et une coordonnée y qui appratiennent à $\mathbb{N}$.

On veut visiter toutes les villes, une et une seule fois. Notre résultat sera une permutation de ces villes : $[v_1, v_2, ..., v_N]$. Une liste avec N villes. Où v_1 sera la première ville visitée, v_2 la deuxième etc.

On utilise la distance euclidienne :

$$euclidienne(v_1, v_2) = \sqrt{(v_{1_x} - v_{2_x})^2 + (v_{1_y} - v_{2_y})^2}$$

On veut minimiser la distance totale qui est :

$$d_{total} = \sum_{i=2}^{N} euclidienne(v_{i-1}, v_i)$$

2 Algorithme

Pour ce faire nous allons comparer plusieurs méthodes :
— Greedy
— Ant System séquentiel
— Ant System parallèle

2.1 Greedy

2.1.1 Algorithme

L'algorithme est relativement simple. C'est le même que celui programmé dans le tp3. Il faut :

1. Choisir une ville aléatoire, v_0.
2. Ajouter la ville la plus prochaine à notre solution.
3. Répéter l'étape 2. jusqu'à ce qu'on ait vue toutes les viles.

Suivant le choix du développeur, on peut choisir d'ajouter la première ville, à la fin pour avoir une boucle. Cet algorithme est totalement déterministe pour un V fixé et une ville initiale v_0 fixée.

2.1.2 Mesures et Résultats

Voici les résultats obtenus en moyenne lors de vingt lancés :

File	Mean	Std
./data/cities.dat	31.537453312384304	1.8045784058777552
./data/cities2.dat	3.387658063949946	0.2163689591198703
./data/cities50.dat	6.278780156847236	2.332163876429746e-15
./data/cities60.dat	6.302401870939252	0.047031657572111
./data/cities80.dat	6.2813443585626185	1.2394364910839084e-15
./data/cities100.dat	6.282070156684796	1.8110771949088597e-15

On remarque que l'algorithme trouve toujours la solution optimale pour le cercle. Ce qui est évident. En prenant le plus proche voisins, on trouve notre point à gauche (ou à droite), et ainsi de suite jusqu'à faire le tour du cercle. Il y a une déviation standard assez importante pour les deux premiers fichiers. Cela vient du faire que l'agorithme est déterministe, donc il dépend du point de départ. Ce qui n'est pas le cas des algorithmes fourmis.

2.2 Ant System Séquentiel

Les deux versions de cet algorithme (séquentielle et parallèle) sont très proches, nous allons donc expliquer en premier lieu la version séquentielle plus parallèle.

2.2.1 Algorithme

Voici le déroulement de l'algorithme :

1. on choisit une ville v_0 de départ pour la fourmis.
2. tant qu'il existe des villes non visitées, on en choisit une v_i et on l'ajoute.
3. marquer le chemin.

On répète cela pour chacune des m fourmis. Cela correspond à une itération. Une fois l'itération terminée, on met à jour la matrice des phéromones et on reprend les itérations jusqu'à en avoir exécuté t_{max}.

Nous allons maintenant expliquer les points cités. Pour le choix de la fourmis, nous les avons répartis une par ville, (en utilisatn le modulo). Il faut donc que n le nombre de villes, soit égal à m le nombre de fourmis. C'est dans ces conditions que les meilleurs résultats sont atteints.

Pour choisir parmis les villes, nous générons un tableau de probabilités (associée une probabilité à chaque ville),et nous choisissons la ville suivante en suivant ces probabilité. Voici comme elles sont générées :

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

Avec k, la $k^{\text{ième}}$ fourmis, i la ville i, j la ville j, et α,β des paramètres, et t l'itération courrante. On va donc favoriser avec τ les chemins avec beaucoup de phéromones ; et avec

η les villes les plus proches. Où τ est la matrice des phéromones, et η la matrice de l'inverse des distances entre villes. Donc $\eta_{ij} = \frac{1}{d_{ij}}$

Chaque fourmis va marquer le chemin de cette manière :

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L^{k}(t)} & \text{si la fourmis } k\text{utilise l'arc}(i,j)\text{dans son parcour} \\ 0 & \text{sinon} \end{cases}$$

Où Q est un paramètre. $L^k(t)$ est la longueur du chemine de la fourmis k à l'itération t. Mettre à jour la matrice des phéromones, soit τ, se fait comme suit :

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