

# Methods and Heuristics for Learning and Optimization

Romain Mencattini

7 novembre 2016

## Table des matières

<b>1</b>	<b>Le voyageur de commerce</b>	<b>3</b>
<b>2</b>	<b>Algorithme</b>	<b>3</b>
2.1	Recuit simulé . . . . .	3
2.2	Greedy . . . . .	4
<b>3</b>	<b>Mesures et Résultats</b>	<b>4</b>
3.1	Cities.dat . . . . .	4
3.2	Cities2.dat . . . . .	5
3.3	CitiesN.dat . . . . .	6
3.3.1	Cities50.dat . . . . .	7
3.3.2	Cities60.dat . . . . .	8
3.3.3	Cities80.dat . . . . .	9
3.3.4	Cities100.dat . . . . .	10
3.3.5	Remarques . . . . .	10

# 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 appartiennent à  $\mathbb{N}$ .

On veut visiter toutes les villes, une et une seule fois. Notre résultat sera une permutation de ces villes :  $[v_1, v_2, \dots, v_N]$ . 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 on va comparer plusieurs algorithmes.

## 2.1 Recuit simulé

Voici le principe de l'algorithme du recuit simulé, ou SA.

1. Choisir une solution initiale,  $s_0$ , qui est une permutation des villes.
2. Calculer la température initiale. Pour ce faire, on fait 100 transformations élémentaires aléatoires, et on regarde le  $\Delta$  d'énergie. En résolvant l'équation :

$$e^{\frac{-\Delta E}{T_0}} = 0.5$$

3. Effectuer une transformation élémentaire aléatoire pour obtenir  $s_1$ .
4. On applique la règle d'acceptation (de Métropolis) :

$$P = \begin{cases} 1 & \text{si } \Delta E < 0 \\ e^{\frac{-\Delta E}{T}} & \text{sinon} \end{cases}$$

On accepte si  $rand() < P$  où  $rand() \in [0, 1[$ . Si on accepte  $s_0 = s_1$ .

5. On considère qu'un équilibre est atteint lorsqu'on a accepté  $12n$  changements ou qu'il y a eu  $100n$  perturbations sans changements.
6. Lorsqu'on est en équilibre, on diminue la température :  $T_{k+1} = 0.9T_k$ .
7. On arrête l'algorithme lorsqu'on a diminué trois fois la température, sans qu'il y ait eu aucun changements de solutions. On considérera que la solution est suffisamment stable pour qu'elle ne change plus.

Il reste à expliquer ce que signifie une perturbation élémentaire. Cela consiste en une nouvelle permutation où on aura échangé deux villes.

## 2.2 Greedy

L'algorithme est plus simple que celui du recuit. Il faut :

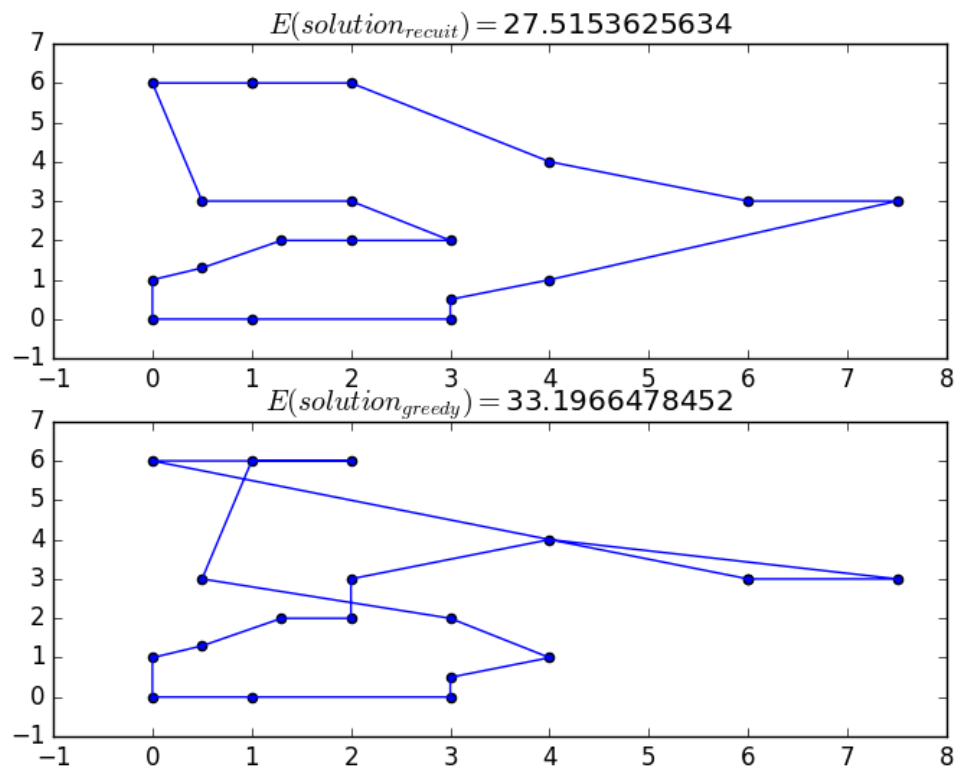
1. Choisir une ville aléatoire,  $v_0$ .
2. Ajouter la ville la plus proche à notre solution.
3. Répéter l'étape 2. jusqu'à ce qu'on ait vu toutes les villes.

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.

## 3 Mesures et Résultats

Voici les résultats obtenus en moyenne lors de dix lancers.

### 3.1 Cities.dat



On remarque que les deux solutions sont assez bonnes, (visuellement et en terme d'énergie). On peut noter que le recuit est meilleur dans ce cas. Il s'agit d'un seul lancé pour

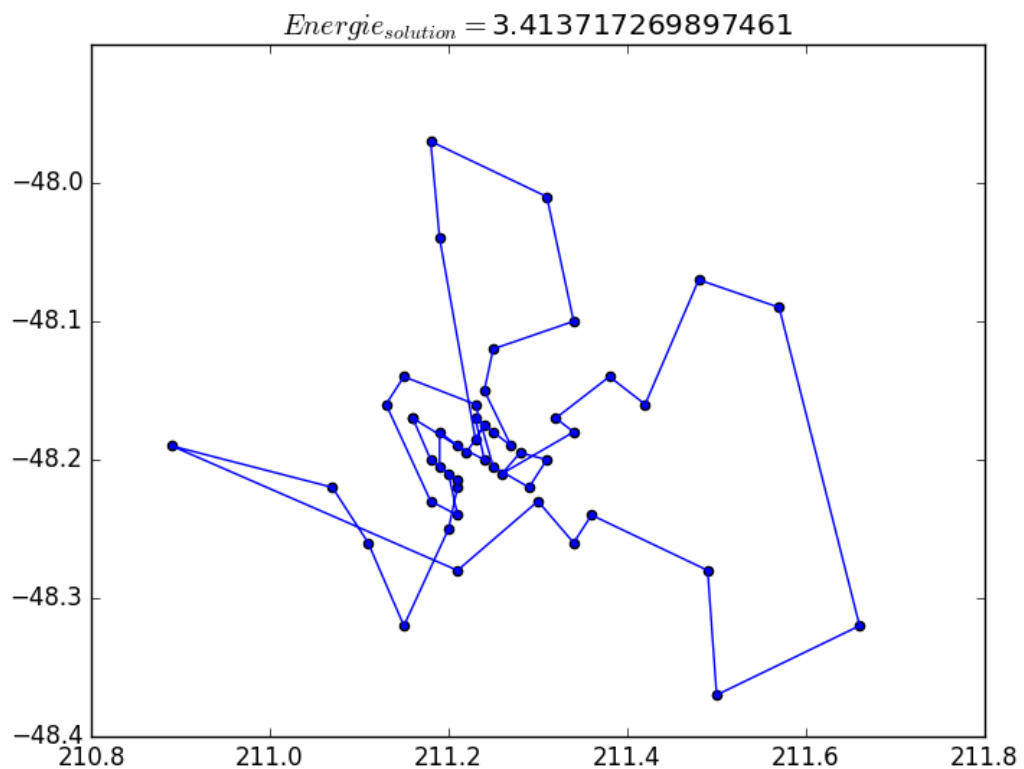
illustré, ce à quoi ça ressemble. Voici le tableau avec la moyenne :

Cities.dat		
	Moyenne	Déviatiion standard
Greedy	33.2325687408	1.77688
Recuit	31.3042554855	1.71704

On voit que le recuit est en moyenne meilleur. Il n'a que peu de déviation standard. Environ 3% de la valeur moyenne d'énergie. Les deux algorithmes sont donc assez stable.

Les résultats moyens sont un peu éloigné du meilleur résultat trouvé plus haut. Il y a moins de 10% de différence, ce qui signifie qu'on trouve de bon solution, assez proche de l'optimum global.

### 3.2 Cities2.dat



Les points centraux étant très proches, cela peut engendré beaucoup de différences d'un lancé à un autre. Il est également plus compliqué de trouver l'optimum.

Voici le tableau avec la moyenne :

Cities2.dat		
	Moyenne	Déviati�n standard
Greedy	3.46352212408	0.166237
Recuit	3.54229936614	0.171651

Le point positif est que la d viation standard est tr s faible. On a confirmation que les algorithmes sont stables sur plusieurs lancements.

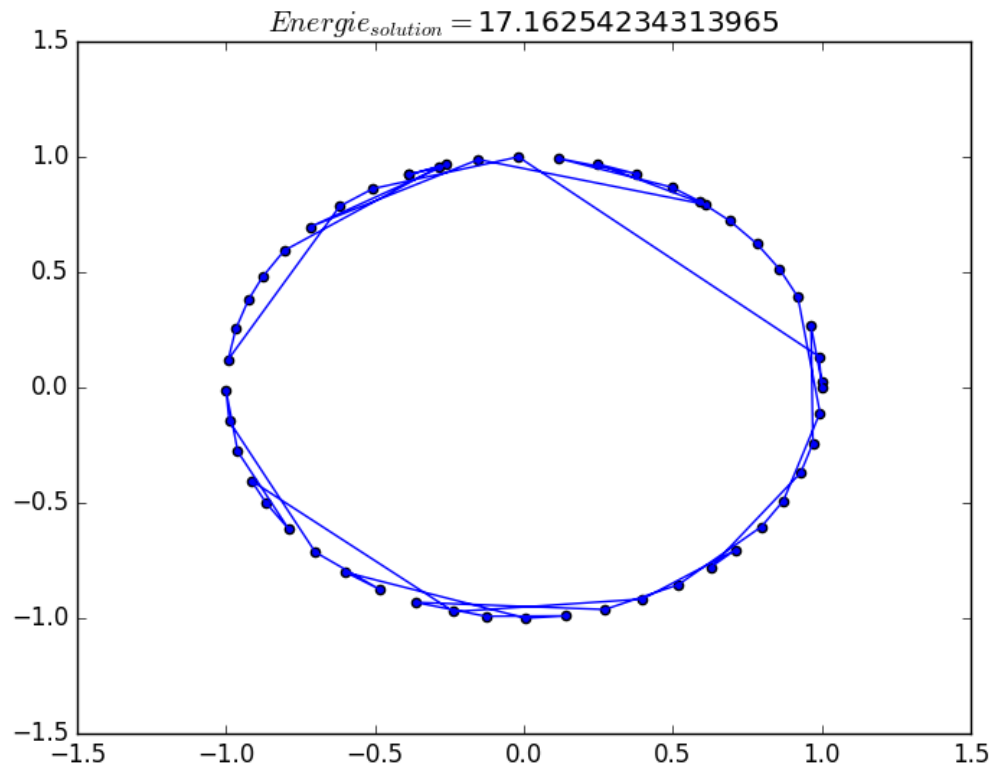
Cette fois ci, le *greedy* est l g rement meilleur. Nous pensons que cela vient des points centraux tr s proches qui peuvent perturber l'algorithme du recuit et engendr  des r sultats moins bon.

### 3.3 CitiesN.dat

Pour les versions avec  $n = \{50, 60, 80, 100\}$ , nous avons g n r  des points en cercle, afin de savoir le r sultat optimal et visualiser   quel point nos algorithmes trouvent la solution optimale.

L'algorithme *greedy* trouve toujours l'optimum global, car en prenant le point le plus proche dans un cercle, on trouve la meilleure solution. Ce n'est donc pertinent que pour nous donner la valeur optimal comme comparaison.

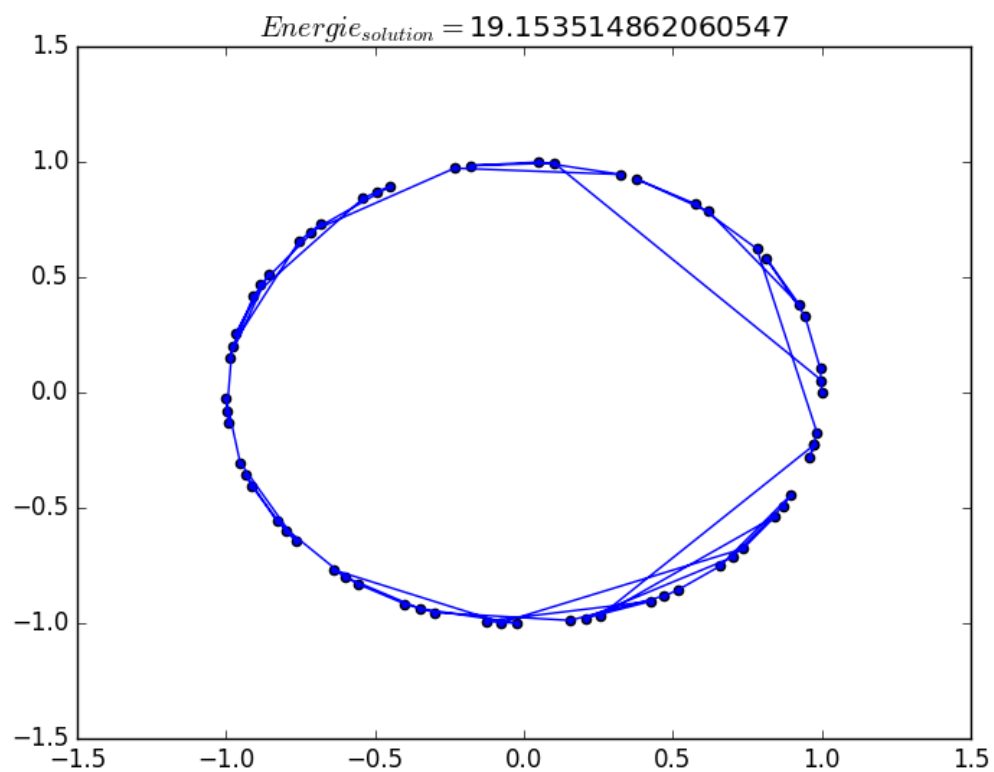
### 3.3.1 Cities50.dat



Voici le tableau avec la moyenne :

Cities50.dat		
	Moyenne	Déviati�n standard
Greedy	6.27877907753	1.50789e-07
Recuit	16.9541293144	1.70877

## 3.3.2 Cities60.dat

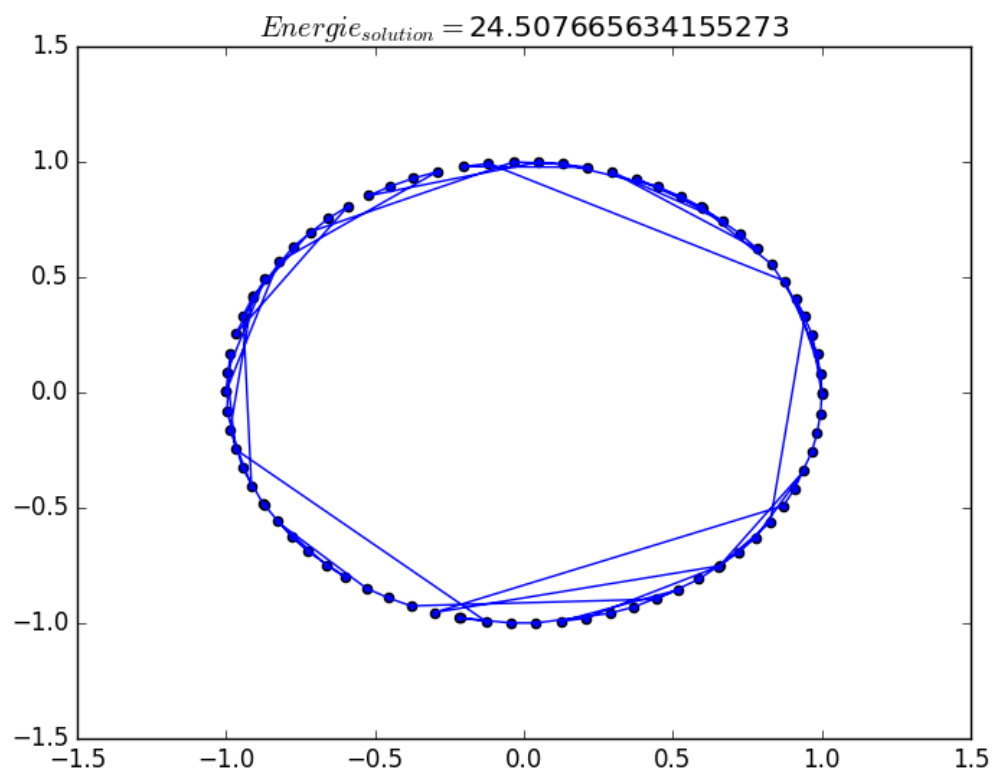


Voici le tableau avec la moyenne :

Cities60.dat		
	Moyenne	Déviati�n standard
Greedy	6.31829538345	0.051879
Recuit	19.7690262794	2.64644



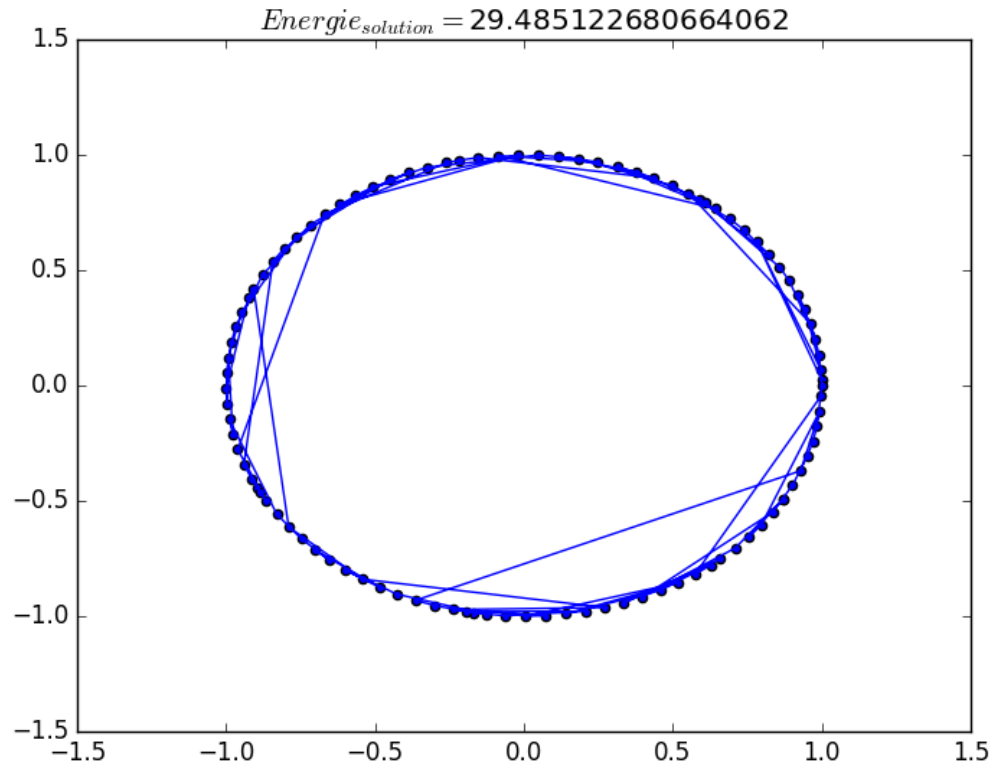
### 3.3.3 Cities80.dat



Voici le tableau avec la moyenne :

Cities80.dat		
	Moyenne	Déviati�n standard
Greedy	6.28134675026	5.43678e-07
Recuit	27.2583179474	2.54419

### 3.3.4 Cities100.dat



Voici le tableau avec la moyenne :

Cities100.dat		
	Moyenne	Déviatiion standard
Greedy	6.28206834793	3.69356e-07
Recuit	31.6346078873	2.16477

### 3.3.5 Remarques

On voit que la valeur de l'algorithme *greedy* tend vers  $d\pi$  où  $d = 2$ . On tend donc bien vers le périmètre d'un cercle comme valeur optimale. Concernant le recuit, on remarque que la majorité de la solution est bien placée, sauf certains points (environ 10% de  $n$ ), ce qui fait augmenter l'énergie.

On peut imaginer que le paysage étant très irrégulier cela influencer le résultat. Il est également possible que les valeurs des paramètres d'acceptations peuvent jouer. Ou bien la manière de diminuer la température.