Cours de modélisation mathématique

Guilherme Dias da Fonseca

TP 3 – Voyageur de Commerce

1 Présentation abstraite

Entrée : Un ensemble de n points du plan $S = \{p_1, \ldots, p_n\}$.

Sortie : Une permutation $(q_1, \ldots, q_n)$ des points de S.

Objectif : Minimiser

$$\|q_n - q_1\| + \sum_{i=1}^{n-1} \|q_i - q_{i+1}\|$$

En d'autres termes, on est donné un ensemble S de n points et il faut trouver le polygone de périmètre minimum dont l'ensemble de sommets est S.

2 Présentation appliquée

Le problème du voyageur de commerce (Travelling Salesman Problem, TSP) s'applique par exemple à la logistique où on veut choisir l'ordre des livraisons d'un camion de façon à minimiser la distance du parcours. Dans ce projet, on considère la version Euclidienne du problème, où chaque ville est un point du plan et la distance entre deux villes est leur distance Euclidienne (dans la vrai vie il faut considérer les routes existantes). Un ensemble de n points nous est donné et on veut, idéalement, trouver le chemin le plus court pour connecter tous les points et retourner au point d'origine.

3 Exemple

Disons que l'entrée est l'ensemble de points représenté sur la figure 1. Une solution possible est la permutation

$$((0, 1.75), (.75, 1.5), (1.5, 2), (2.25, 1.5), (2.75, .25), (2, .5), (1.25, 1), (.75, .25), (.25, .75))$$

de longueur 8.27.

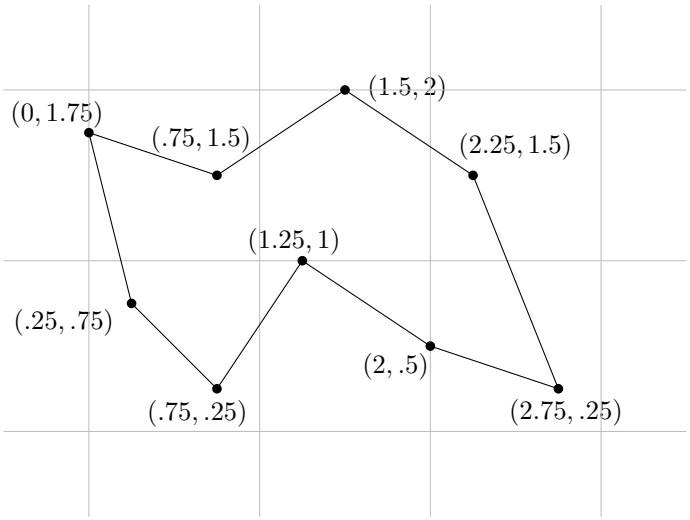

FIGURE 1 – Exemple de parcours.

4 Fichiers d'entrée

Le fichier d'entrée est textuel et consiste de deux flottants par ligne séparés par un espace, sauf la première ligne qui contient l'entier n. Les n lignes de deux flottants correspondent aux coordonnées x, y de chaque point.

Pour l'exemple on a le fichier :

```
9
0 1.75
1.5 2
.75 1.5
.25 .75
.75 .25
1.25 1
2 .5
2.25 1.5
2.75 .25
```

Vous allez trouver 4 fichiers d'entrée sur l'ENT, pour $n = 20, 100, 1000, 10000$. Les fichiers sont basés sur les coordonnées des villes de différents pays.

5 Fichier de sortie

Le format du fichier de sortie est presque le même que ce du fichier d'entrée. La première ligne du fichier de sortie est l'entier n. La deuxième est la longueur du tour. Ensuite on liste les points de l'entrée, mais selon l'ordre du parcours obtenu. Pour l'exemple on a :

```
9
8.270864494922767
0 1.75
.75 1.5
1.5 2
2.25 1.5
2.75 .25
2 .5
1.25 1
.75 .25
.25 .75
```

Peut importe quel est le premier point utilisé, mais on ne le répète pas à la fin. Par contre, pour calculer la longueur, il faut compter le retour au premier point.

6 Stratégies

Plus proche voisin La stratégie la plus simple est de choisir un point de départ et à chaque pas connecter le dernière point du chemin à son voisin le plus proche que n'est pas encore visité (figure 2). Un problème de cette stratégie est que les deux extrémités du chemin peuvent être très éloignées à la fin, ce qui rend le retour du tour très coûteux.

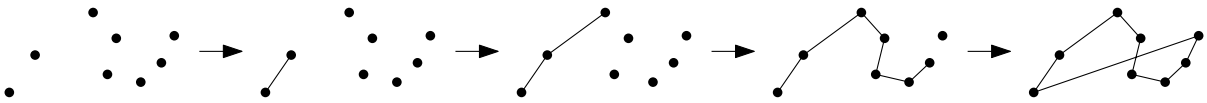

FIGURE 2 – Algorithme de plus proche voisin.

Incrémental On commence par choisir un petit sous-ensemble des points pour lesquels c'est facile de faire un bon tour. Par exemple on peut trouver trois points et faire un triangle. Une autre possibilité est d'utiliser l'enveloppe convexe (figure 3a) ou un tour pour un sous-ensemble des points. A chaque pas on trouve le point qu'on peut ajouter au tour en augmentant le moins sa longueur (figure 3b). On répète jusqu'à avoir ajouté tous les points.

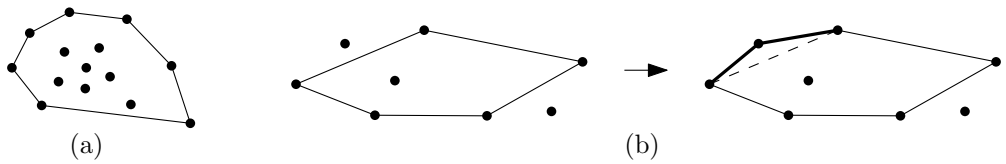

FIGURE 3 – (a) Enveloppe convexe. (b) Ajouter une ville au tour.

Local search Une fois qu'on a trouvé une bonne solution on peut essayer de la raffiner. On peut par exemple enlever les croisements d'arête ou changer le placement d'un point dans la liste (figure 4.)

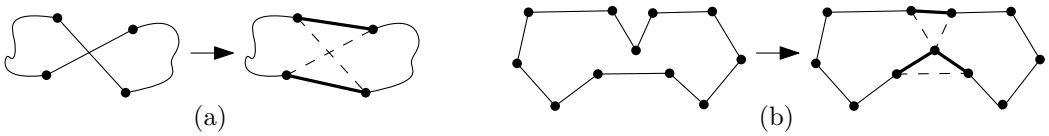

FIGURE 4 – Local search : (a) enlever un croisement et (b) changer un point de place.

Sketch Parfois les algos précédents sont trop lents. Dans ce cas on peut choisir un sous-ensemble de la solution avec un grille et l'utiliser pour guider notre solution (figure 9).

7 Notation

La notation sera basée sur le tableau ci-dessous, qui indique lq longueur du tour nécessaire pour avoir la note à gauche pour chaque fichier. La note du projet est la moyenne des notes des 4 fichiers.

Note	`20.in1`	`100.in1`	`1000.in1`	`10000.in1`
10	115	100000	135000	560000
11	108	90000	128000	540000
12	105	85000	124000	530000
13	103	84000	121000	525000
14	100	83000	119000	520000
15	99	82000	117000	510000
16	97	81000	115000	500000
17	95	80000	113000	490000
18	94	79000	111000	480000
19	93	78500	109000	470000
20	92	78000	108000	465000

La notation est automatique (sauf exceptions) et utilise le logiciel `grader1.py` disponible sur l'ENT. Pour exécuter ce logiciel, il faut être dan un dossier avec les fichiers d'entrée ainsi que votre code. Voici un exemple :

8 Exemples de solutions

Le logiciel pour tester la solution produit un fichier html qui représente la solution de manière graphique. Voici les 4 exemples sur les figures 5 à 9.

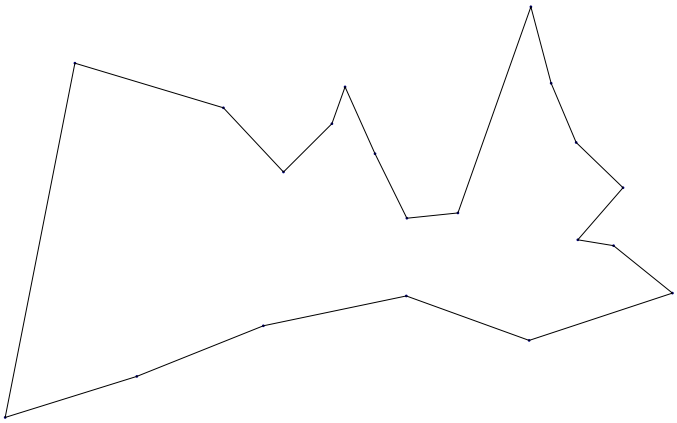

FIGURE 5 – 20 capitales européennes et leur tour de longueur 91.57.

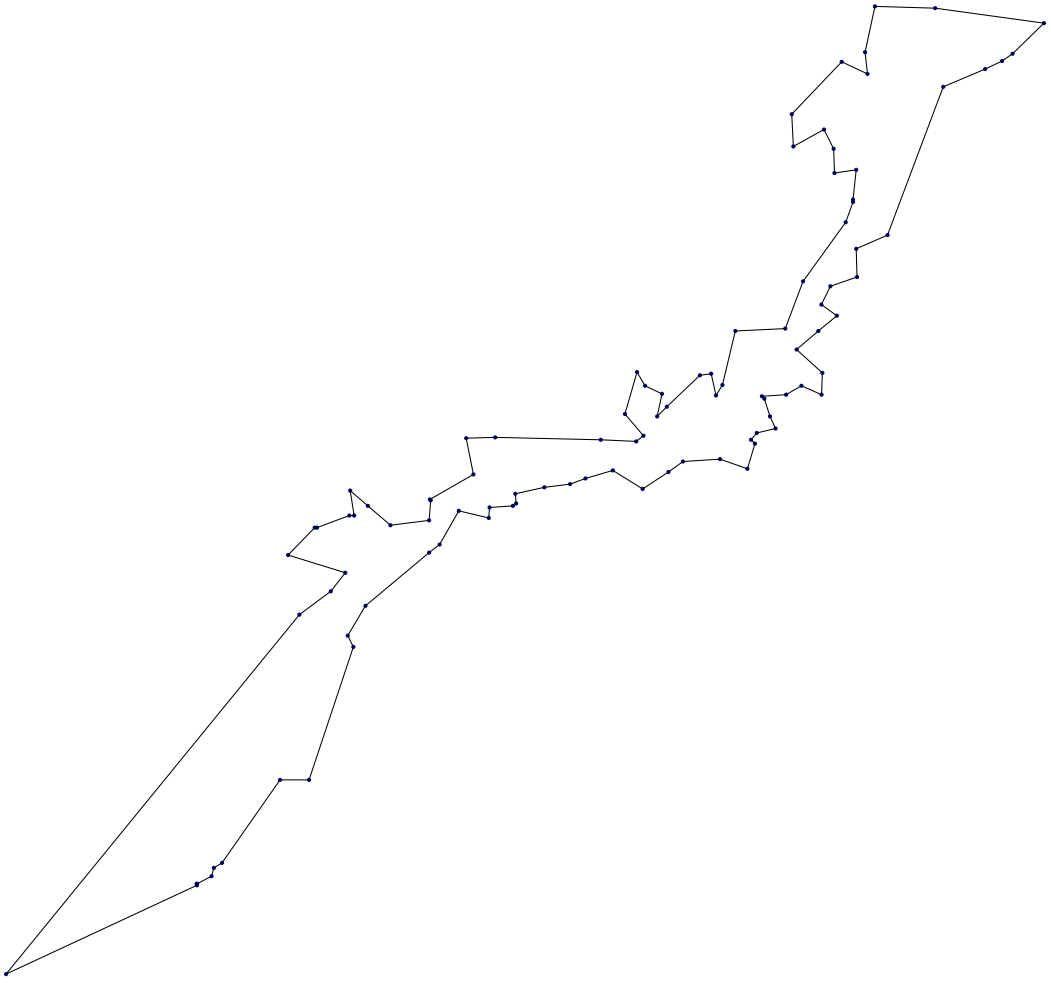

FIGURE 6 – 100 villes du Japon et leur tour de longueur 77993.

FIGURE 7 – 1000 villes de la Grèce et leur tour de longueur 107154.

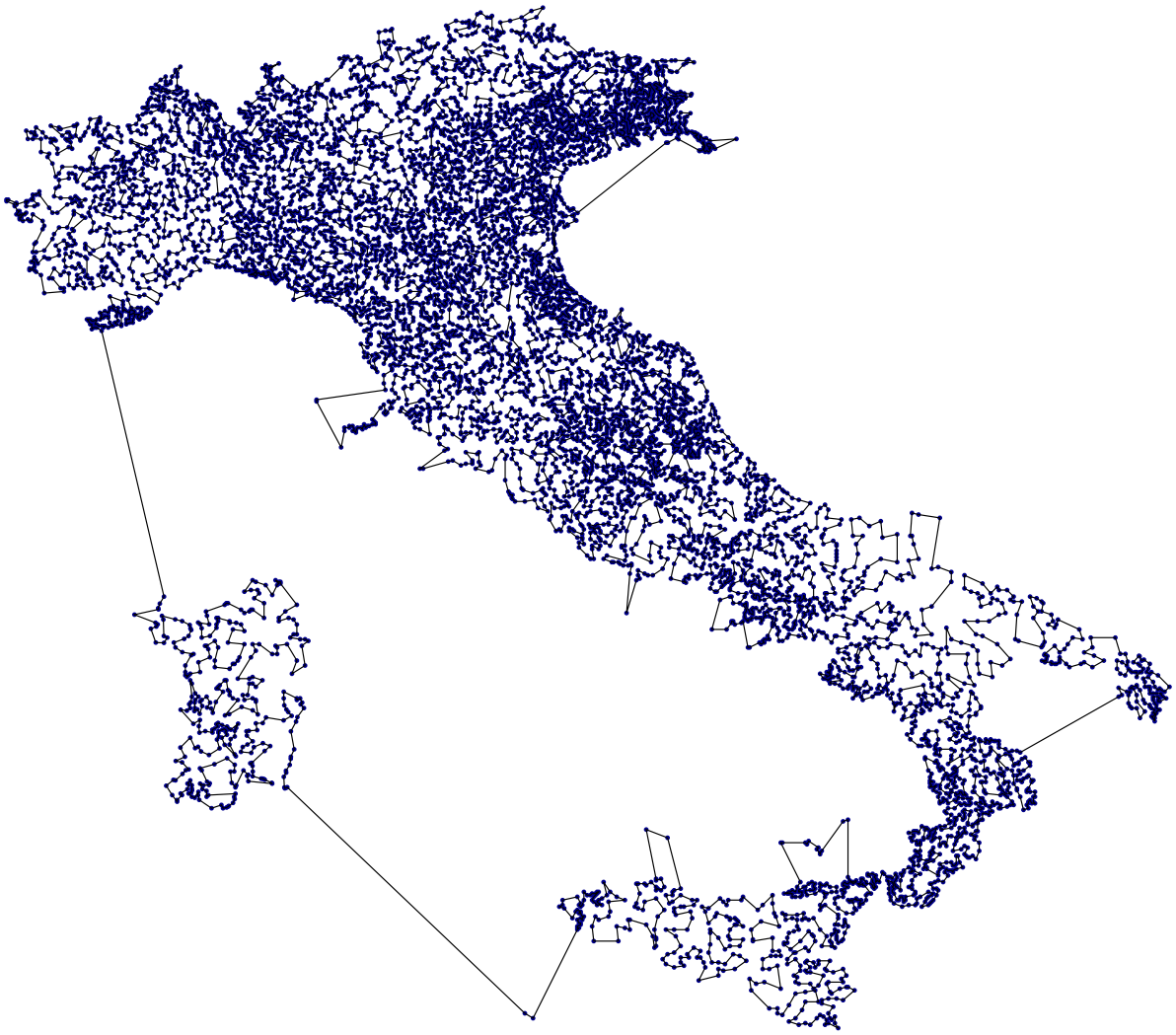

FIGURE 8 – 10000 villes de l'Italie et leur tour de longueur 467194. On peut repérer des arêtes très longues qui ne devraient pas exister dans une solution optimale.

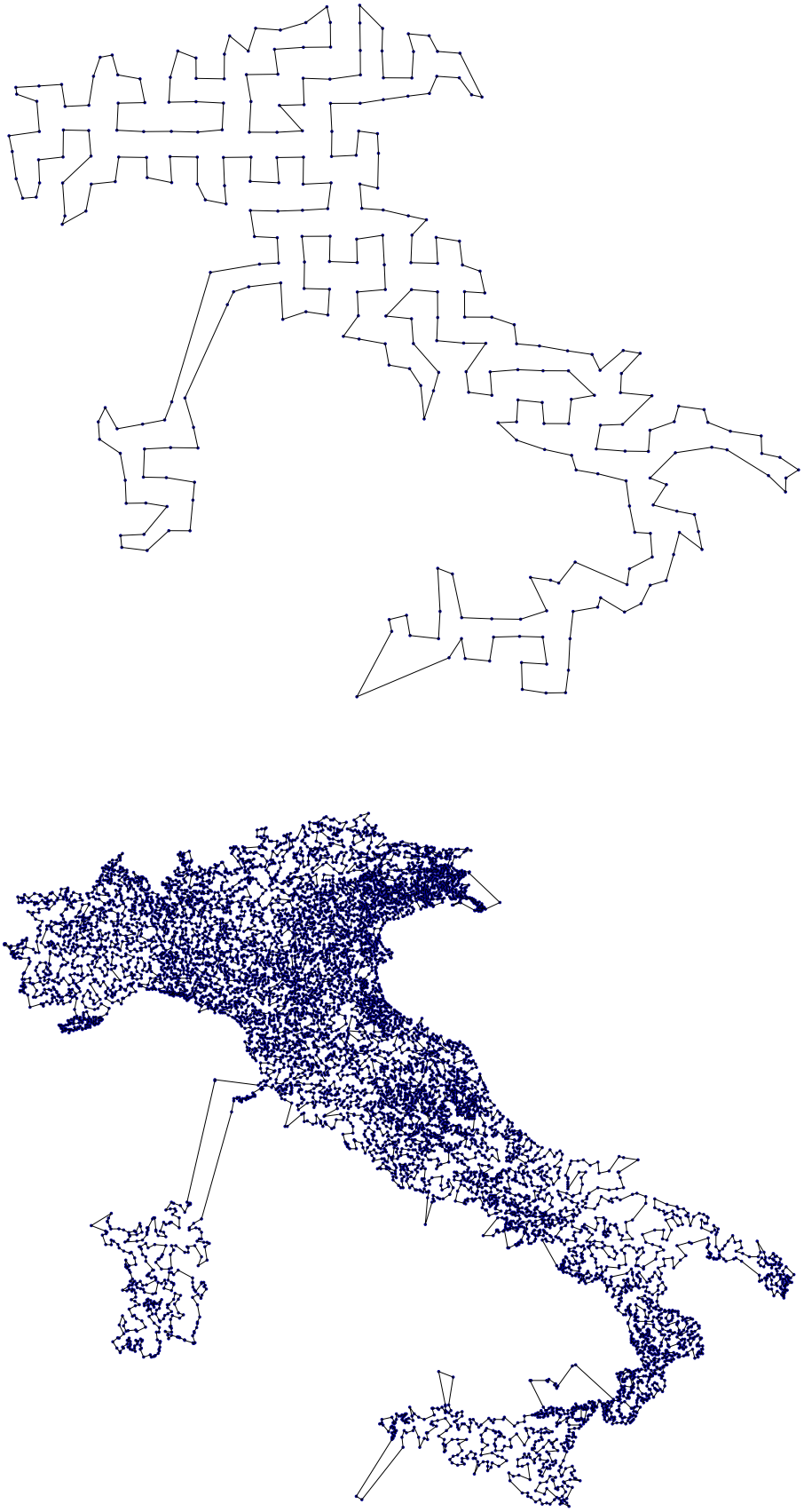

FIGURE 9 – Sketch de l'Italie et un tour de longueur 461643 fait selon le sketch.

9 Enveloppe Convexe

Je vous donne un code Python 3 pour calculer l'enveloppe convexe si vous voulez l'utiliser.

```
def ch(v):
  def ccw(p,q,r):
    x = ((q[0]-p[0]) * (r[1]-p[1])) - ((r[0]-p[0]) * (q[1]-p[1]))
    return x > 0    # True if p,q,r are oriented counterclockwise
  def halfch(v):
    hull = [v[0]]
    for p in v[1:]:
      while len(hull)>=2 and ccw(p, hull[-1], hull[-2]):
        hull.pop()
      hull.append(p)
    return hull
  top = halfch(sorted(v))
  bottom = halfch(sorted(v, reverse=True))
  return top[0:-1] + bottom[0:-1]
```