Implémentation d'algorithmes de recherche opérationnelle

Laboratoire n^{o}4 (MTH6412B) – Dominique Orban

Jean-Noël Weller (1843975) & Juliette Tibayrenc (1800292)

28 novembre 2016

1 Introduction

Dans cette nouvelle étape du projet, nous avons implémenté l'algorithme de Rosenkrantz, Stearns et Lewis pour obtenir une approximation du résultat du problème du voyageur de commerce. Nous avons comparé les résultats obtenus avec différentes heuristiques et les résultats optimaux.

2 Implémentation de l'algorithme de Rosenkrantz, Stearns et Lewis

2.1 Préliminaires

Nous avons modifié l'algorithme de Prim pour pouvoir choisir le nœud racine à partir duquel construire l'arbre de recouvrement minimal. Nous avons implémenté en outre un parcours d'arbre en pré-ordre et un parcours d'ordre en largeur.

2.2 Algorithme

Les paramètres de l'algorithme sont le nœud racine, le type de parcours (profondeur `dfs` ou largeur `bfs`) et l'algorithme Prim ou Kruskal (`prim` ou `kruskal`) pour le recouvrement minimal. La fonction retourne un graphe représentant la tournée minimale approchée obtenue.

3 Résultats

Les résultats numériques détaillés sont présentés dans le tableau qui suit :

Instance	Source	Prim/Krusal	Poids	Écart à l'optimal
bayg29	14	Prim	2009	399
bays29	23	Kruskal	2396	376
brazil58	11/42	Kruskal	29031	3636
brg180	2	Prim	16280	14330
dantzig42	41	Prim	884	185
fri26	25	Kruskal	1075	138
gr120	78	Krusal	9611	2669
gr17	1	Prim	2224	139
gr21	0	Kruskal	3664	957
gr24	1/14	Kruskal	1563	291
gr48	12	Kruskal	6536	1490
hk48	6	Kruskal	14503	3042
swiss42	29	Kruskal	1587	314
pa561	0	Kruskal	3882	1119

On a constaté entre autres que, si les résultats changent selon la racine utilisée, ils ne varient pas beaucoup pour un même algorithme et un même type d'exploration. Ils ne sont pas optimaux et varient (peu) selon l'essai, comme noté précédemment à cause de l'existence d'arêtes distinctes de même poids.

Pour pa561, le plus grand des exemples fournis, on n'a pas cherché à appliquer notre algorithme en prenant successivement chaque nœud comme racine : le temps d'exécution du script aurait été trop important. On n'a donc testé l'algorithme qu'avec deux nœuds différents pour cet exemple. Il est d'ailleurs fortement déconseillé de chercher à faire tourner le premier `main.py` fourni sur cet exemple sans remplacer `len(roots)` par un nombre plus raisonnable, comme 1 ou 2, dans la dernière boucle…

Par ailleurs, brg180 semble être un cas particulier, pour des raisons que nous n'avons pas pu déterminer : tandis que les résultats obtenus pour les autres instances respectent certaines garanties de distance à l'optimal, ce n'est pas son cas, et il en est même très, très loin.

Les parcours obtenus pour les instances pour lesquelles sont fournies des coordonnées sont en annexe.

4 Programme principal : main

Comme précédemment, on peut l'exécuter depuis le terminal comme suit :

```
python ./main stsp/bayg29.tsp
```

(tout autre fichier tsp symétrique peut bien sûr être substitué à bayg29.tsp).

5 Autres changements

Nous avons corrigé certains éléments du code ne correspondant pas à la norme PEP8. Nous avons également revu le critère d'arrêt pour l'algorithme de Kruskal selon les conseils donnés : au lieu de vérifier que tous les nœuds ont été ajoutés à l'arbre minimal, nous vérifions que le nombre d'arêtes est le bon (nombre total de nœuds - 1).

Nous avons, comme conseillé, déplacé l'attribut `key` de la classe `Node` à la classe `DisjointSet`, afin de conserver seulement l'essentiel dans la classe `Node`.

Nous avons modifié la représentation d'un graphe avec Matplotlib pour afficher les noms des nœuds et les poids des arêtes. Le graphe résultant est bien sûr très peu lisible quand on souhaite représenter un graphe complet (mais c'était déjà le cas avant), mais on obtient un affichage bien plus intéressant pour les arbres minimaux.

6 Conclusion

Pour ce laboratoire, nous avons enfin assemblé les différents algorithmes précédemment implémentés (Prim, Kruskal) ou étudiés en cours (algorithmes d'exploration) pour construire des solutions au problème du voyageur de commerce. Nos algorithmes semblent fonctionner, mais nos solutions sont encore trop lentes pour pouvoir choisir le nœud le plus intéressant pour les graphes trop grands (c'est-à-dire pa561, dans notre sélection d'exemples). Nous constatons également que les résultats obtenus ne sont pas les meilleurs possibles, et qu'ils dépendent à la fois du choix entre Prim et Kruskal et du type d'exploration utilisé, et peuvent compter une part d'aléatoire, à cause des arêtes distinctes de même poids présentes dans la plupart des graphes fournis.

A Parcours obtenus

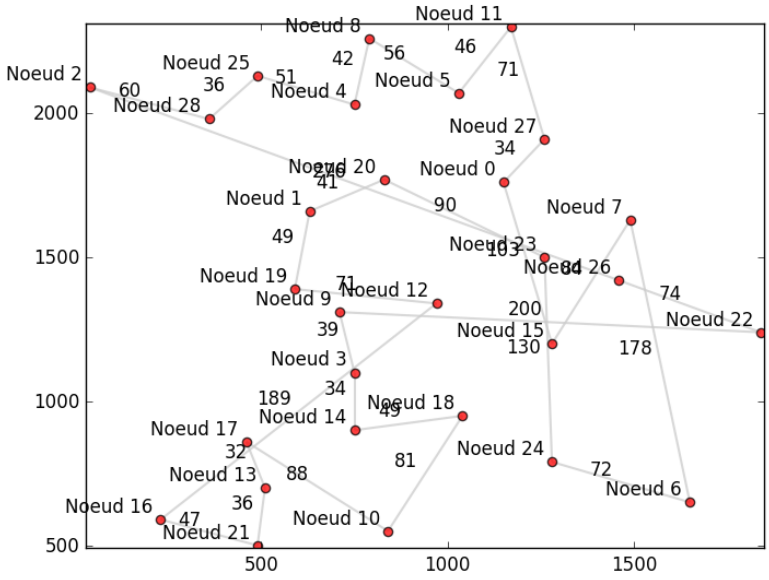

FIG. 1: Meilleure tournée obtenue pour bayg29

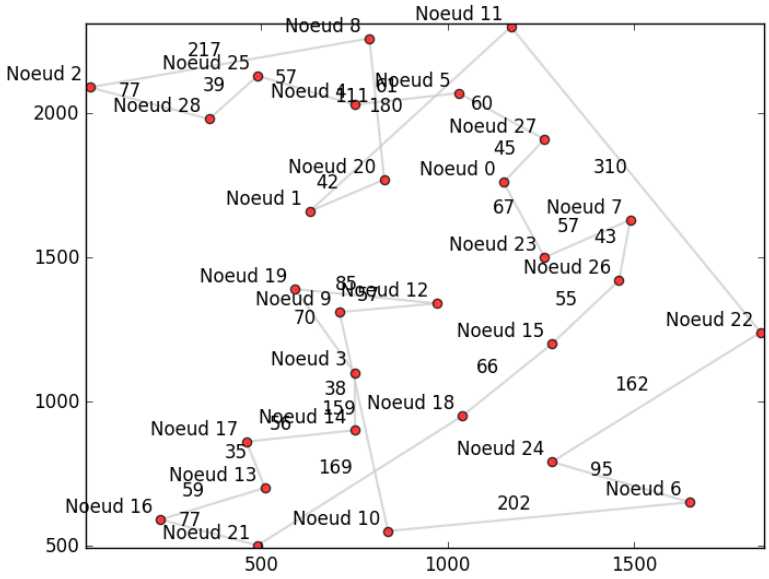

FIG. 2: Meilleure tournée obtenue pour bays29

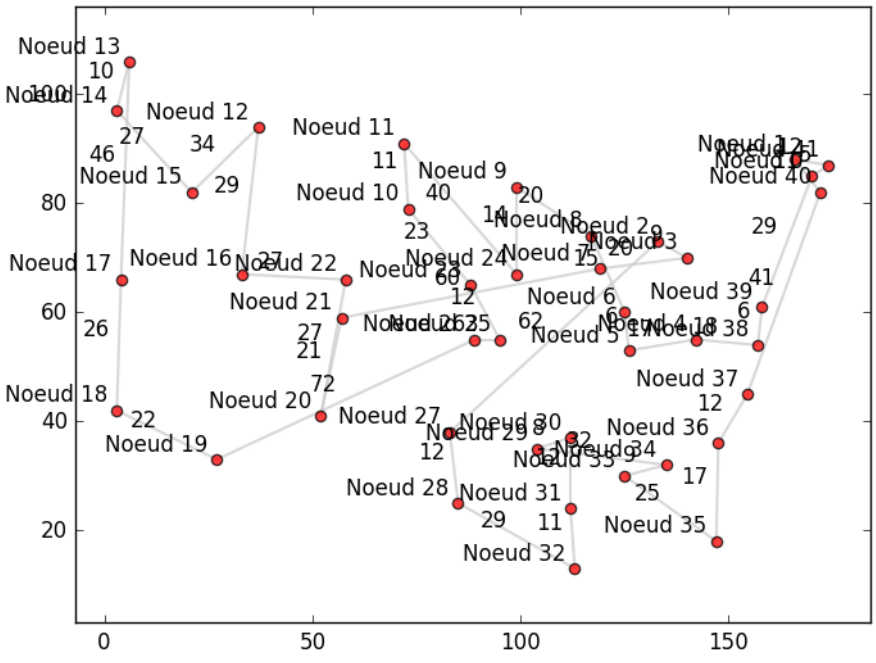

FIG. 3: Meilleure tournée obtenue pour dantzig42

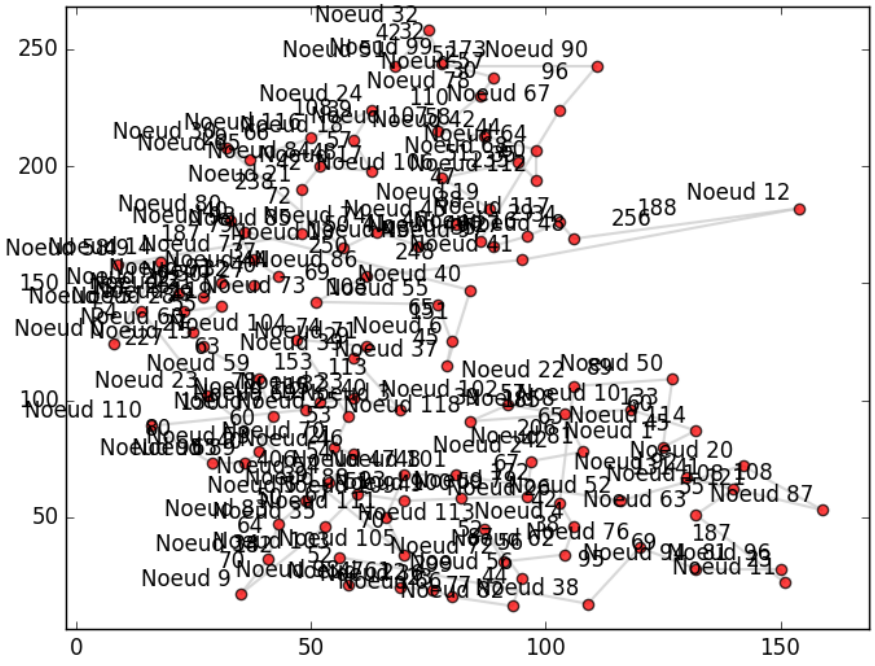

FIG. 4: Meilleure tournée obtenue pour gr120

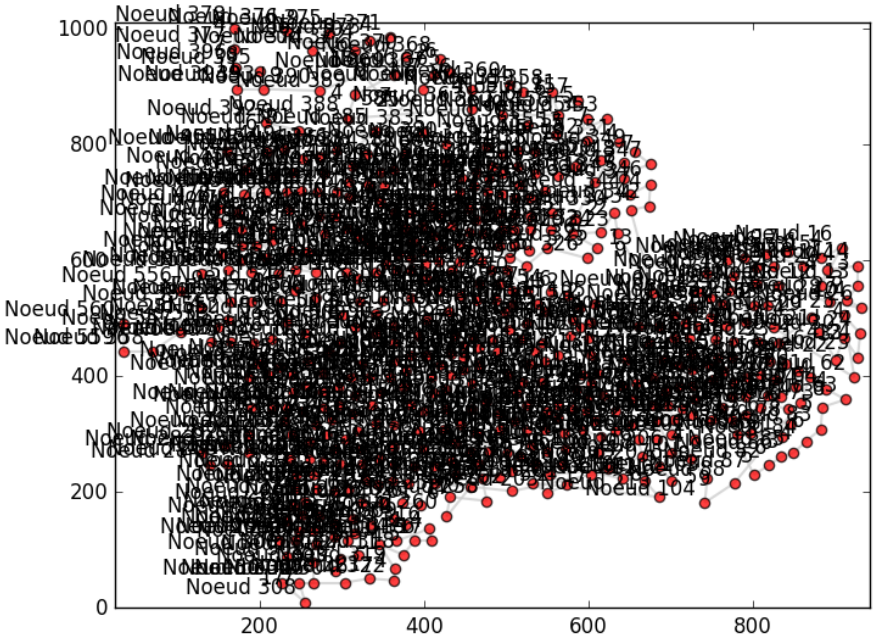

Fig. 5: Meilleure tournée obtenue pour pa561