

**Etude comparative des heuristiques**

S2.02 - Exploration algorithmique d'un problème

Léo Bourdin / Antoine Lindimer / Romain Barabant

IUT DIJON AUXERRE  Département informatique

# Critères de comparaison

## Complexité algorithmique

La complexité algorithmique représente le nombre de calculs basiques que fera l’algorithme selon la taille du problème. Pour un même algorithme, la complexité peut parfois être légèrement différente selon la qualité du code.

**Pour ce critère** : le meilleur algorithme sera celui avec la complexité algorithmique la plus petite.

## Temps d’exécution

Le temps d’exécution représente la durée que prendra l’algorithme pour calculer l’ordre de la tournée finale. La Stopwatch mesurera et affichera cette durée en milliseconde pour chaque algorithme.

**Pour ce critère** : le meilleur algorithme sera celui avec le temps d’exécution moyen le plus petit.

*Remarque : pour que ce critère soit sensé, on exécutera chaque algorithme sur l’ensemble du jeu d’essai afin qu’il soit soumis à différentes situations.*

## Distance

La distance représente la distance totale de la tournée finale trouvée par l’algorithme.

**Pour ce critère** : le meilleur algorithme sera celui avec la distance moyenne le plus petit.

*Remarque : pour que ce critère soit sensé, on exécutera chaque algorithme sur l’ensemble du jeu d’essai afin qu’il soit soumis à différentes situations.*

# Jeu d’essai de graphes

<Author>Antoine

Comme susdit, le jeu d’essai aura pour but de faire rencontrer aux algorithmes des situations diverses et variées afin de récolter des données plus précises, donc utiles pour la comparaison.

Pour créer un jeu d’essai pertinent, nous avons commencé par recréer les graphes que nous avions étudié en séance de TP. Puis nous avons décidés de créer nos propres graphes afin d’obtenir des résultats exploitables.

Avec des graphes simples (où l’on peut obtenir facilement un résultat en faisant tourner les algorithmes à la main), les résultats ne sont pas très intéressants. Le temps d’exécution est inférieur à 10 ms dans tous les cas, on observe tout de même de petites différences dans les résultats des tournées. Ces résultats étaient à prévoir puisque nous les avions observés durant les séances de découvertes.

En utilisant le graphe fourni avec le sujet (GrapheSimple2.gph), nous avons pu tester nos algorithmes et observer des résultats différents en fonction de l’algorithme.

Pour obtenir des résultats intéressant nous avons commencé par créer des graphes ou tous les sommets sont reliés entre eux, ce sont des cliques. Nous avons essayé de créer une clique de degré 5 « à la main », cela s’est révélé très long et inutile car il semble impossible de créer des cliques de degré supérieur à 10 « à la main ». Nous avons donc créé un algorithme qui permet de générer automatiquement dans la console les lignes qui permettent de créer un graphe avec un fichier en .gph dans le projet.

Cet algorithme imprime dans la console :

* L’usine qui est numéroté 0
* La liste des magasins
* La liste des routes entre tous les sommets

Pour les sommets, les coordonnées sont générées en se basant sur le cercle trigonométrique. On place dans un premier temps l’usine à un abscisse définie et avec une ordonnée de 0, puis on place les sommets sur le cercle à intervalle régulier. En ce qui concerne les routes, avec deux boucles imbriquées, on peut générer chaque route entre tous les sommets. Pour ce qui est de la pondération, elle est aléatoire et comprise entre 1 et le nombres de sommets.

# Mesure qualitative des algorithmes

## Plus proche voisin

## Insertion proche

## Insertion loin

## Recherche locale

## Heuristiques personnelles

# Conclusion

# Correction (provisoire

inverser condition

stocker distanceTournee

renvoyer un couple de valeur (ne pas utiliser un attribut)

cas particulier pour jeu d'essais