Algorithmique
M2 Data Science
Projets

Vincent Runge

Section 1

Les règles du jeu

Choix optimal d'un projet

- Chaque groupe comporte *3 étudiants*.
- Pour *n* groupes et *n* projets, on demande aux étudiants d'établir une liste de préférences.

C'est un problème algorithmique !!!
Comment répartir au mieux les projets pour maximiser la satisfaction des étudiants ?

Remplir ce tableau avec vos choix par équipe

- **On décidera avec l'algorithme génétique du groupe M2_DS_2022_2023 !**
- me l'envoyer **maintenant** avec un exemple fonctionnel

Travail à réaliser

- Une présentation beamer / Rmd présentation / ppt (20 points)
- Un rapport au **format Rmd** (rendu pdf ou html) et un Package (20 points)

Le package (du type **M2algorithmique**) doit contenir les éléments suivants :

1) Solution naïve R fonctionnelle
2) Solution améliorée R fonctionnelle
3) Evaluation sur des simulations de la complexité
4) Le code en C++ et comparaison en temps avec le code R
5) Un package fonctionnel et bien documenté sur github

Les dates

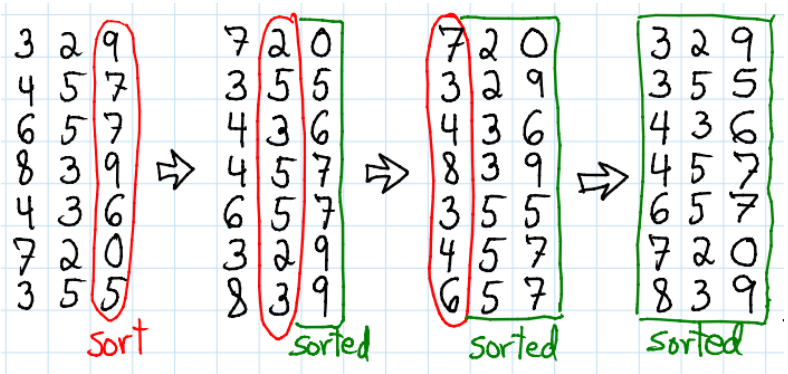
- projets proposés le 27 octobre
- classement de 1 à 8 pour les projets le même jour

1) Rendu du rapport le **? (début janvier)** (à vincent.runge@univ-evry.fr)
2) Présentation de 20 min devant un jury du LaMME (2 ou 3 chercheurs) le **? (mi-janvier)** et 5-10 min de questions.
3) Une étape de suivi (au minimum une) **début décembre** avec moi ou votre tuteur (s'organiser avec lui)

Section 2

Les projets

Projet 1 : Radix sort

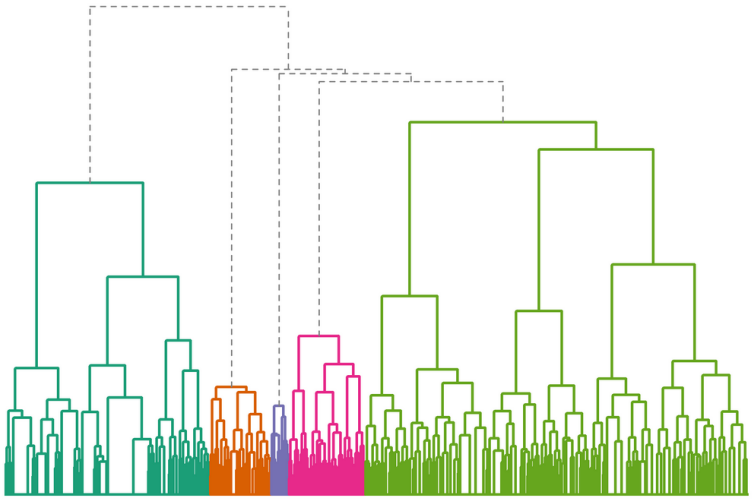


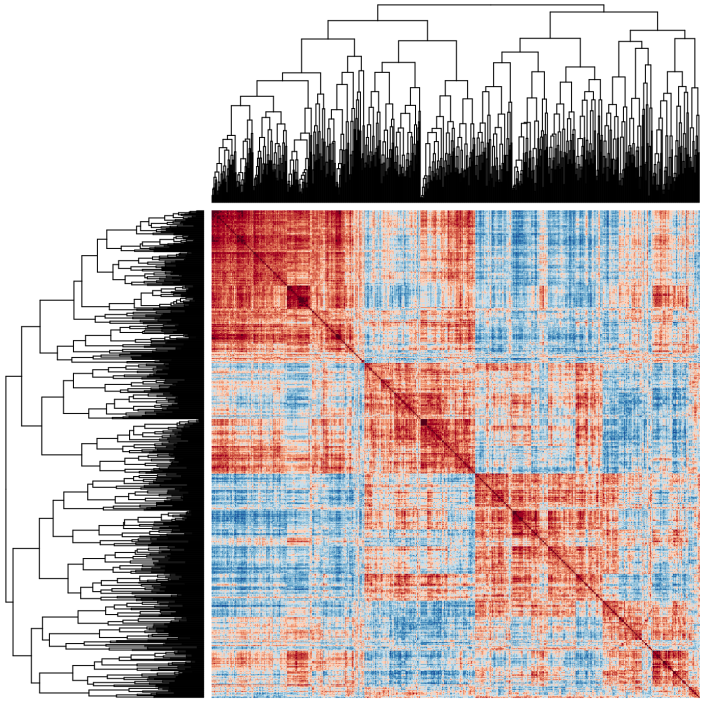
Page wikipédia

Il est possible de trier des éléments en temps linéaire (!) en procédant digit par digit.

Comparer les performances du radixsort à un tri en $n\log(n)$ de votre choix sur différentes tailles et types de données. Mettre en évidence les cas favorables à l'un ou l'autre des algorithmes.

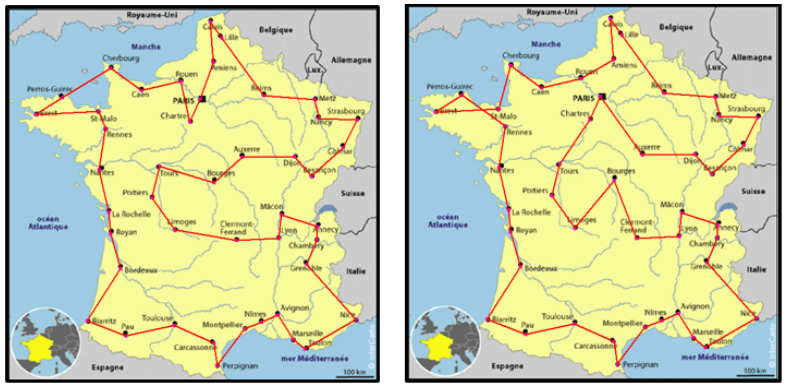
Projet 2 : Classification ascendante hiérarchique





Une **référence** et plus d'info ici. L'objectif minimal est de passer d'une complexité $O(n^3)$ à $O(n^2)$.

Projet 3 : Voyageur de commerce (TSP)

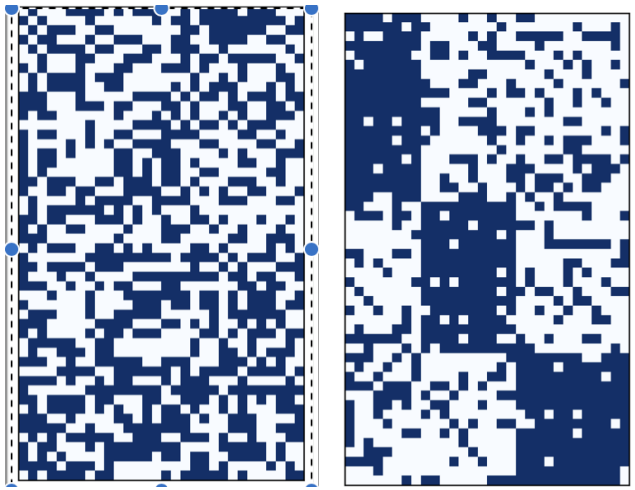


Ce problème est bien connu pour être NP-difficile

1) Implémenter l'algorithme naïf, celui du **branch and bound** (voir ici) et de **Held-Karp** et comparer les complexités (sur des simulations). Etudier en particulier pour combien de villes au plus on peut résoudre le problème en un temps raisonnable (1h au plus)
2) Utiliser le meilleur de ces 2 algorithmes pour estimer les performances des algorithmes heuristiques vus en cours (jeudi 27 octobre) pour un nombre de villes grands.

Projet 4 : *Biclustering Algorithm*

Les matrices binaires de très grande dimension sont très présentes en **bioinformatique**. Le biclustering consiste à permuter les lignes et les colonnes d'une matrice pour faire apparaître des groupes aux caractéristiques homogènes.



Les algorithmes existants sont cependant assez lents. Coder deux solutions existantes et comparer leur complexité. (naïve, récursif, prog. dynamique,. . .)

Projet 5 : Dijkstra's algorithm

Trouver le plus court chemin entre deux noeuds d'un graphe

1) Algorithme naïf à implémenter : l'algorithme de Bellman-Ford ou une méthode heuristique (algo glouton)
2) Présenter des exemples d'application de l'algoritme de Dijkstra

Références :

Un référence pour l'algorithme de Dijkstra

Voir aussi une application possible
https://en.wikipedia.org/wiki/Seam_carving

Projet 6 : Détection d'une rupture en ligne

L'article Romano et al. 2022 (et code) propose un algorithme, FOCuS, pour la détection d'une rupture en ligne considérant simultanément toutes les tailles de fenêtres possibles.

A chaque nouvelle observation y_n l'algorithme permet de retrouver la rupture τ minimisant

$$\sum_{i=1}^{t}(y_i - \bar{y}_{1:t})^2 + \sum_{i=t+1}^{n}(y_i - \bar{y}_{t+1:n})^2,$$

avec $\bar{y}_{t:t'} = \sum_{i=t}^{t'} y_i/(t' - t + 1)$.

FOCuS est proche de l'algorithme de Melkman pour **identifier les points sur l'enveloppe d'un ensemble de n points**. Le projet est d'implémenter FOCuS en R/C++ et de comparer ses performances (algorithmiques et statistiques) à celle de l'approche séquentielle de Page. **Avec Rigaill, Pishchagina, Romano (UK)**

Projet 7 : Détection de rupture avec noyaux

Harchaoui et Cappé 2007 ainsi qu'Arlot et al. 2019 ont proposé l'approche, KernSeg, basée sur des noyaux reproduisant, pour la détection de rupture dans des espaces généraux.

Elle permet par exemple de considérer la détection de ruptures

1) dans des signaux audio ou vidéo
2) dans des réseaux variants au cours du temps
3) dans des chaînes de caractères ou encore
4) dans la distribution d'un signal univarié.

Celisse et al. 2019 (et ici) ont proposé un algorithme de programmation dynamique implémentant KernSeg.

Projet 7 : Détection de rupture avec noyaux #2

KernSeg a pour complexité $O(n^2)$. Cette complexité est problématique pour des profils de grandes tailles ($n > 10^4$).

Le projet est d'implémenter la technique d'élagage classique PELT à ce problème KernSeg pour le noyau Gaussien. Il faudra alors comparer le temps d'exécution avec ou sans élagage pour des signaux simulés avec ou sans ruptures.

Avec Rigaill, Liehrmann

Projet 8 : Algorithme de Chew-Liu

Algorithme qui sert à déterminer les indépendances conditionnelles entre variables.

1ère étape : construction du graphe de variables

- Noeuds : Variables
- Poids des arêtes : Information mutuelle sur la paire de noeuds.

2ème étape : Réduction du graphe; Créer un arbre couvrant de poids maximal sur le graphe de l'étape précédente.

Objectifs : Implémenter l'algorithme de Chew-Liu, basé sur un algorithme naïf sur la réduction du graphe. Comparer l'algorithme naïf à l'algorithme de Kruskal pour la réduction du graphe (résultats, complexité, etc).

Appliquer ces méthodes sur un jeu de données, et interpréter les résultats.
Une référence

avec De Santiago