Analyse multidimensionnelle des préférences, et application aux données de vote
LU2IN013 : Projet

Sira Lina Achouri, Ina Campan, Ida Rogie
Encadrant : Olivier Spanjaard

Février - Juin 2023

Table de matière

1 Introduction

1.1 Données et contexte

En 2017 et en 2022, une équipe de chercheurs a testé plusieurs méthodes alternatives de vote à l'occasion des élections présidentielles. C'est le projet *Voter Autrement* [7] (ou *Un Autre Vote*). La motivation était de remettre en question le système de vote actuel et montrer que d'autres méthodes peuvent représenter différemment l'opinion politique des électeurs.

Voici quelques-unes de ces méthodes de vote :

- **Vote par note** : les votants attribuent à chaque candidat une note dans un ensemble prédéfini (par exemple : $\{-1, 0, 1\}$ ou $\{0, 1, 2, 3\}$).
- **Jugement majoritaire** : les votants attribuent une mention verbale (comme *'très bien'*, *'bien'*, *'passable'*, ou encore *'insuffisant'*).
- **Méthode de Borda** : les votants classent les candidats par ordre de préférence.

C'est cette dernière méthode qui nous intéresse particulièrement car les votes sont exprimés sous forme de rangements complets de candidats. Les données de cette expérience sont publiques. Elles se présentent sous la forme d'un tableau dont les lignes représentent chaque participant à cette étude et les colonnes les candidats. Dans les cases, il y a des entiers allant de 1 à 11 (11 étant le candidat préféré) qui représentent leur préférence. La dernière colonne contient le nom du candidat pour lequel le participant a réellement voté au premier tour.
Certaines données disponibles concernant les élections présidentielles ne sont toutefois pas des ordres complets. Nous avons traité ce problème et nous l'expliquons dans la partie 3.2.

Nous avons aussi utilisé d'autres données qui représentent des préférences complètes sur *PrefLib* [4]. Ce sont par exemple des personnes qui ont classé des séries sur *Netflix* ou encore le classement d'arrivée des pilotes d'une course automobile. Les données sont représentées sous la forme d'un ficher *.soc* dans lequel un entier est attribué à chaque *'candidat'* ou *'pilote'*. Il y a par la suite les listes de toutes les préférences distinctes avec le nombre de *'votants'* ou *'courses'* qui ont eu cette préférence.

1.2 Problématique

C'est avec l'analyse multidimensionnelle que l'on va pouvoir utiliser ces données. Grâce à plusieurs algorithmes (déjà présents ou non dans une librairie existante), nous allons essayer de représenter les préférences dans un plan en dimension n (ici en dimension 1 ou en dimension 2). En prenant le cadre d'une élection, les candidats et les votants sont placés en sorte que chaque votant soit le plus proche possible de son candidat favori et le moins proche du candidat le moins apprécié. Plus généralement, la préférence d'un votant est décroissante avec sa distance aux candidats.

Sur la Figure 1, les candidats sont représentés par les points rouges A, B, C et D et les votes par les points bleus. Le point bleu en bas le plus à gauche représente le vote dont la

préférence est $[A, C, B, D]$. Le diamètre du disque est proportionnel au nombre de votants exprimant cette préférence.

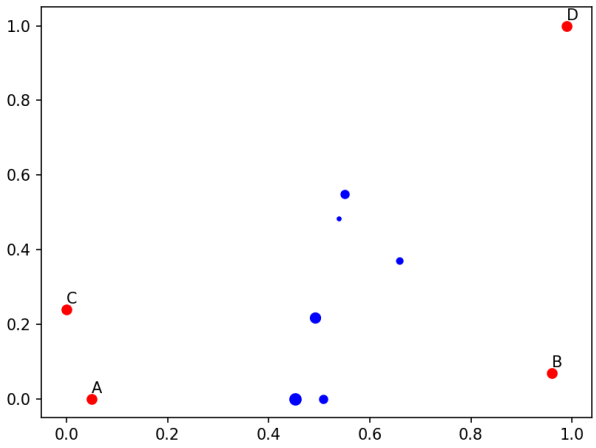

Figure 1: Exemple de graphe obtenue en dimension 2

Si l'on trace les médiatrices entre chaque paire de candidats, on obtient plusieurs zones, qui correspondent chacune à une préférence particulière (Figure 2). Il y a cependant plus d'ordres de préférences possibles que de zones décrites précédemment. Cela veut dire que certaines préférences ne peuvent pas être représentées dans le plan.

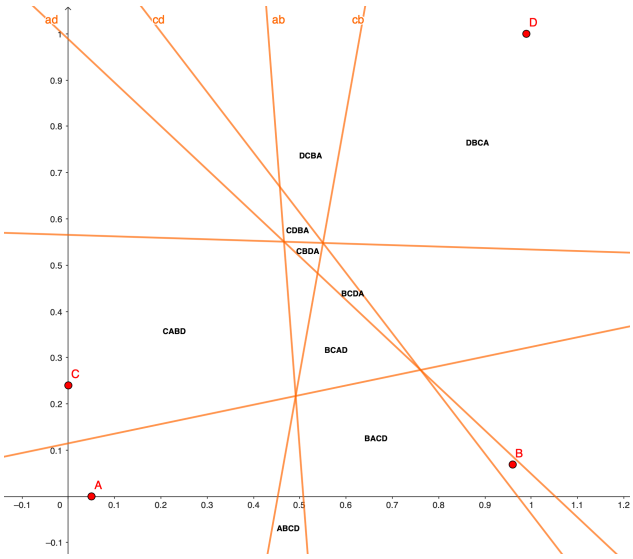

Figure 2: Graphe avec quelques zones de préférences

Il s'agit donc ici d'implémenter des algorithmes qui font en sorte que cette situation arrive le moins possible, c'est-à-dire que les préférences en entrée soient respectées le mieux possible.

2 Méthodes utilisées

L'objet de cette partie est d'introduire les notions mathématiques utilisés pour formaliser ce problème. On va faire appel aux deux méthodes suivantes pour résoudre le problème posé :

- La programmation mathématique **(PM)** [3]
- Le positionnement multidimensionnel non-métrique (*Non-metric multidimensional scaling* en anglais) **(NMDS)** [2]

en dimension 2 et en dimension 1.

2.1 La programmation mathématique

Grâce à l'article *A multidimensional spatial model for preference representation in multicriteria decision aiding* [3], nous avons pu établir un programme qui s'appuie sur un modèle mathématique; dans son cadre, on va distinguer des variables, des paramètres, des contraintes et des fonctions.

Les variables et les contraintes posées dans un espace de dimension 2 sont celles qui suivent:

On pose C (respectivement V) l'ensemble des candidats (respectivement votants).

Soit $n = 2$ la dimension de l'espace des solutions.

$$0 \leqslant c_i \leqslant 1, \quad \forall c \in C, \forall i \in \{0, 1\}$$
$$0 \leqslant v_i \leqslant 1, \quad \forall v \in V, \forall i \in \{0, 1\}$$

où c_i et v_i représentent des coordonnées dans le plan.

$$d_c^v = \sqrt{\sum_{i=0}^{n-1} (c_i - v_i)^2}, \quad \forall c \in C, \forall v \in V$$

où d_c^v donne la distance euclidienne entre un candidat c et un votant v.

$$0 \leqslant \sigma_{(c,c')}^v, \quad \forall (c, c') \in C^2, \forall v \in V, \text{ tels que } c > c' \text{ pour le votant } v$$

$$d_c^v \leqslant d_{c'}^v + \sigma_{(c,c')}^v, \quad \forall v \in V, \forall (c, c') \in C^2 \quad \text{avec} \quad c >_v c'$$

Les variables $\sigma_{(c,c')}^v$ sont des variables artificielles, rajoutées pour permettre que la condition $d_c^v \leqslant d_{c'}^v$ ne soit éventuellement pas respectée.

Pour un votant, la plus petite distance doit être observée par rapport à son premier choix dans la préférence, donc la plus grande distance correspond à sa dernière option. Ainsi, la fonction à minimiser est bien la somme des variables σ, car on cherche à minimiser la violations des contraintes. De plus, la nature quadratique des équations à résoudre est donnée

par la dimension 2 de l'espace.

Le passage en une seule dimension revient à poser $n = 1$ et garder seulement une seule coordonnée pour les candidats, ainsi que pour les votants.

Dans le cas général, on aura la **fonction objectif** suivante à minimiser :

$$\min(\sum_{v \in V} \sum_{(c,c') \in L^v} \sigma^v_{(c,c')}) \quad \text{avec } L_v \text{ l'ensemble des couples } (c, c') \text{ pour un votant.}$$

2.2 Non-metric multidimensional scaling

Le positionnement multidimensionnel non-métrique est une méthode d'analyse des données qui est conçue pour traiter des données de type classements et rangements. L'objectif principal est de préserver les relations de similarité ou de dissimilarité entre les objets d'origine lors de leur projection dans l'espace, que ce soit en dimension 1, 2 ou 3 [2].

On va alors créer une matrice symétrique, appelée de dissimilarité, de taille $n + m$, avec n le nombre total de candidats et m le nombre de votants. S'il y a un zéro dans une case (i, j), cela indique qu'il n'y a pas de mesure de dissimilarité entre les objets correspondant à la ligne i et la colonne j (par exemple, deux candidats ou deux votants). Les composantes non-nulles représentent les distances entre les objets, afin de mettre en évidence les candidats qui se suivent dans la préférence d'un votant.

Étant donné un placement des candidats et des votants dans l'espace, on calcule la distance Euclidienne d_{ij} entre chaque paire (candidat, votant). Cette distance n'a pas de raison d'être du même ordre de grandeur que les valeurs D_{ij} dans la matrice en entrée (rang du candidat j dans la préférence du votant i).

Pour "convertir" sur une même échelle, on fait une régression linéaire entre les d_{ij} et les D_{ij}. La valeur $\hat{d}_{ij} = \alpha + \beta D_{ij}$ ainsi obtenues (les valeurs de α et β sont retournées par la régression linéaire) est en quelque sorte la distance "convertie en rang".

On réalise ensuite la somme des carrés des écarts $\sum_{i,j}(d_{ij} - \hat{d}_{ij})^2$ pour calculer l'adéquation de la représentation euclidienne avec les données en entrée, en normalisant par $\sum_{i,j} d^2_{ij}$ pour éviter que la valeur du stress augmente avec le nombre de candidats (opération de moyenne).

Le positionnement multidimensionnel cherche à minimiser la fonction **stress** (qui associe une valeur à un positionnement des candidats et des votants), donnée par la formule générale [5]:

$$Stress = \sqrt{\frac{\sum_{i,j}(d_{ij} - \hat{d}_{ij})^2}{\sum_{i,j} d^2_{ij}}}$$

Cette fonction mesure à quel point les rangs obtenus sont proches des vraies préférences en entrée.

2.3 Évaluation des méthodes

Pour évaluer et comparer les résultats obtenus par les différentes méthodes, on utilise le **tau de Kendall**, un indice statistique qui mesure la proximité entre deux rangements et qui va prendre dans notre cas des valeurs entre 0 et 1.

Soient R_i la préférence dans le fichier en entrée et R'_i la préférence dans la représentation. On définit la fonction $d(R_i, R'_i)$ qui calcule le nombre de couples en désaccords.

On obtient ainsi le tau de Kendall avec la formule :

$$\tau = \frac{2}{mn(n-1)} \sum_{i=0}^{m} d(R_i, R'_i)$$

où $m =$ le nombre total de préférences et $n =$ le nombre de candidats.

En conclusion, un tau de valeur 0 indique que la représentation respecte les préférences données en entrée, alors qu'une valeur égale à 1 indique un désaccord total entre les données initiales et les résultats obtenus.

3 Conception et implémentation

Dans la suite, on va exploiter les moyens techniques disponibles afin de résoudre le problème posé. Les algorithmes sont fournis dans le langage Python et les principaux outils exploités pour implémenter les méthodes décrites dans la section précédente sont le solveur **Gurobi** [1] et la bibliothèque python **scikit-learn** [6], destinée à l'apprentissage automatique.

3.1 Conception

Il est maintenant question de comprendre comment les méthodes décrites précédemment avec des formules mathématiques sont mises en œuvre via des algorithmes. On fournit ci-dessous les instructions pour NMDS, qu'on a traduit dans le langage python :

Données : nbCandidats, tabOrders
Résultat : tab (tableau aux coordonnées des objets en dimension 2)
nbVoters = calcul du nombre total de votants;
Création d'une matrice de taille **nbCandidats + nbVoters**;
pour toutes les *préférences* **faire**
  pour tous les *votants* **faire**
    si *préférence exprimée* **alors**
      matrice[k + i + nbcandidats][tabOrders[i][j] - 1] = j;
      matrice[tabOrders[i][j] - 1][k + i + nbcandidats] = j;
    fin
  fin
fin
Appliquer la transformation NMDS avec le paramètre *n_components* = 2;
Retour de la matrice des coordonnées;

Algorithme 1 : NMDS en dimension 2

Exemple de construction de la matrice :

On prend en entrée 4 candidats et 2 préférences complètes distinctes, exprimées chacune par un seul votant : [1, 2, 3, 4] et [3, 2, 4, 1].

Dans la matrice, les quatre premières lignes correspondent aux candidats, suivies de deux lignes pour représenter les votants. On construit une matrice symétrique, en cherchant l'indice des candidats dans les listes des préférences des votants.

$$M_{\text{NMDS}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 0 & 4 & 3 \\ 1 & 2 & 3 & 4 & 0 & 0 \\ 4 & 2 & 1 & 3 & 0 & 0 \end{bmatrix}$$

3.2 Implémentation et analyse des algorithmes

Par définition, un solveur est un logiciel mathématique, éventuellement sous la forme d'un programme informatique autonome ou d'une bibliothèque de logiciels, qui résout un programme mathématique.

Ainsi, pour le première méthode on a fait appel à Gurobi [1], qui est un solveur de programme mathématique, et pour la deuxième méthode, on importe la bibliothèque python : *from sklearn.manifold import MDS* [2].

Nos fichiers sont organisés de la manière suivante :

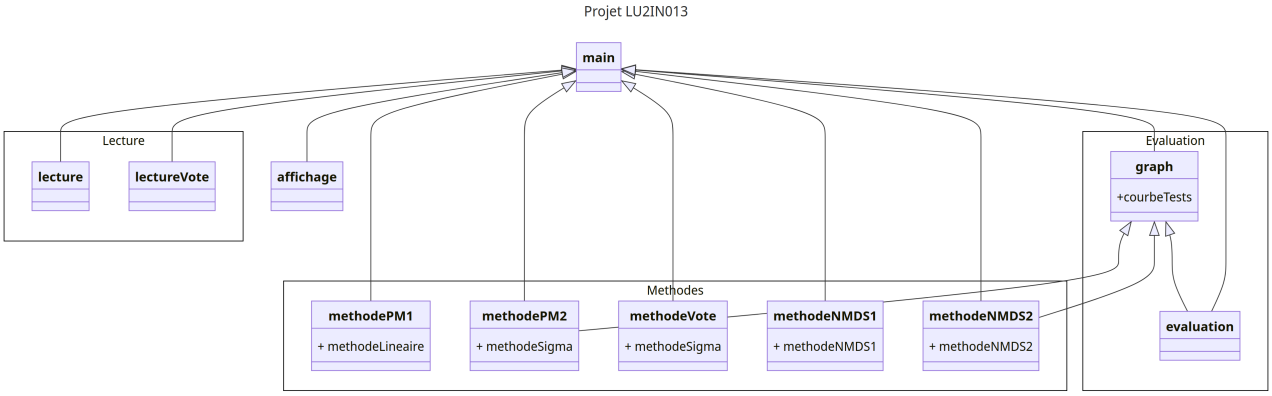

Figure 3: Organisation des fichiers

Les données de votes chargées depuis *VoterAutrement* présentent des préférences incomplètes. Un candidat peut par exemple effectuer le rangement suivant : [Jean-Luc Mélenchon, Philippe Poutou, Nathalie Arthaud] (Votant 35854 du fichier *stv111.csv*), sans indiquer d'ordre de préférence sur les autres candidats. Pour la lecture de ces données, on choisit de mettre à `-1` le rang du candidat non exprimé dans la préférence. Pour l'exemple ci-dessus, la liste associée est : [10, 11, 1, -1, -1, -1, -1, -1, -1, -1, -1] où 1, 10 et 11 sont les identifiants des candidats exprimés.

Quant à l'évaluation, pour appliquer l'algorithme, nous avons besoin de tableaux de taille uniforme. Nous avons donc décidé de déplacer les candidats non classés à la fin des tableaux : un tableau $[10, 5, 7, 9, 8, 11, 1, 2, 3, 4, 6]$ retourné par le solveur où seuls les candidats d'identifiants 1, 10 et 11 étaient exprimés dans la préférence initiale devient donc $[10, 11, 1, 5, 7, 9, 8, 2, 3, 4, 6]$. Les préférences exprimées seront toujours mieux classées que les préférences non exprimées, cela ne va donc pas impacter le calcul du tau de Kendall.

Il est important de noter que les algorithmes à contraintes quadratiques comme celui de la méthode PM sont de complexité exponentielle. Concernant NMDS, la complexité est exponentielle mais, le pire cas n'arrivant presque jamais, la réponse de l'algorithme est en pratique presque instantanée.

4 Expérimentations numériques

4.1 Données

L'interface du programme fonctionne comme suit : l'utilisateur saisit un fichier (en format *.csv* ou *.soc*) dont la lecture permet d'extraire les nombres de candidats et de votants, les noms des candidats et le tableau des préférences uniques précédées par leurs pondérations (nombre de votants exprimant cette préférence). Après le choix de la méthode et du temps de calcul maximum souhaité, les nouvelles coordonnées des votants et des candidats sont récupérées et utilisées pour produire un affichage. Si l'utilisateur souhaite évaluer la méthode, les nouvelles préférences sont recalculées à partir des distances des votants par rapport aux candidats. On applique par la suite la formule de la partie 2.3 pour afficher le tau de Kendall.

Pour illustration, on applique le programme au fichier `stv111.csv` de *Voter Autrement* [7], qui contient des données de vote issues du premier tour de l'élection présidentielle de 2017. Les votants, au nombre de **10946**, classent par ordre de préférence les **11** candidats qui se sont présentés. L'exécution se fait avec les **400** préférences les plus fréquentes. Elle aboutit aux affichages suivants pour les candidats, avec un temps de calcul maximum fixé à deux heures pour Gurobi :

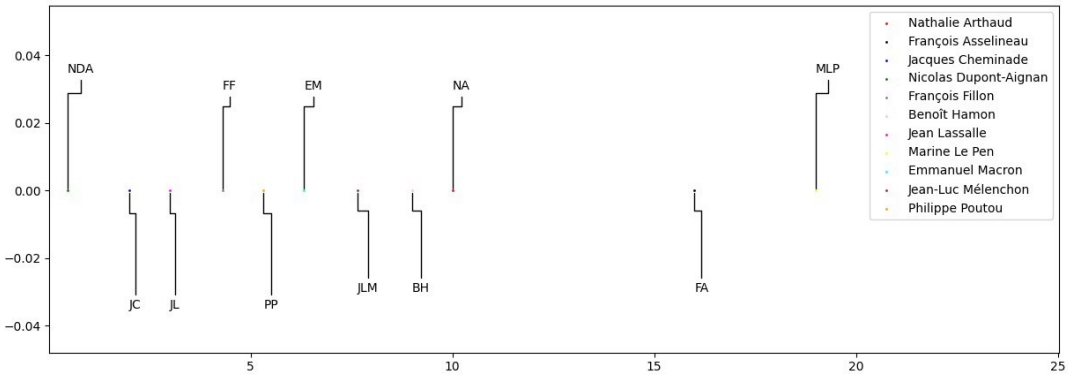

Figure 4: PM1 appliqué à stv11.csv : tau = 0.03

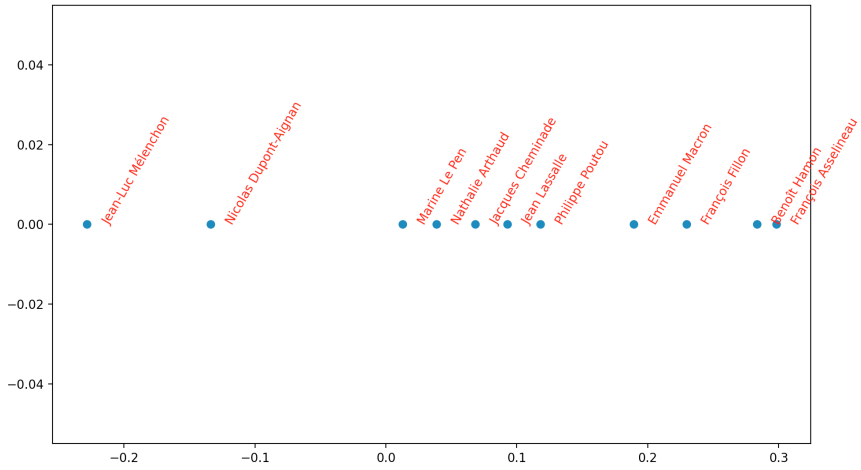

Figure 5: NMDS1 appliqué à stv11.csv : tau = 0.14

4.2 Comparaison des méthodes en dimension 2

Une autre option fournie par le programme est celle d'analyse des méthodes en dimension 2 sur un catalogue de fichiers *.soc*. Elle permet d'afficher les courbes Temps d'exécution / Nombre de candidats pour la programmation mathématique `(courbe rouge)` et *Non-Metric Multidimensional Scaling* `(courbe bleue)` appliquées sur des instances euclidiennes à nombre borné de préférences uniques (on se limite aux dix préférences les plus fréquentes).

Voici ce que l'on obtient après l'exécution avec un temps de calcul maximum de 400 secondes (lecture des fichiers non comprise), sur des instances à 5, 6, 7 et 8 candidats, avec 6 fichiers *.soc* pour chaque nombre de candidats :

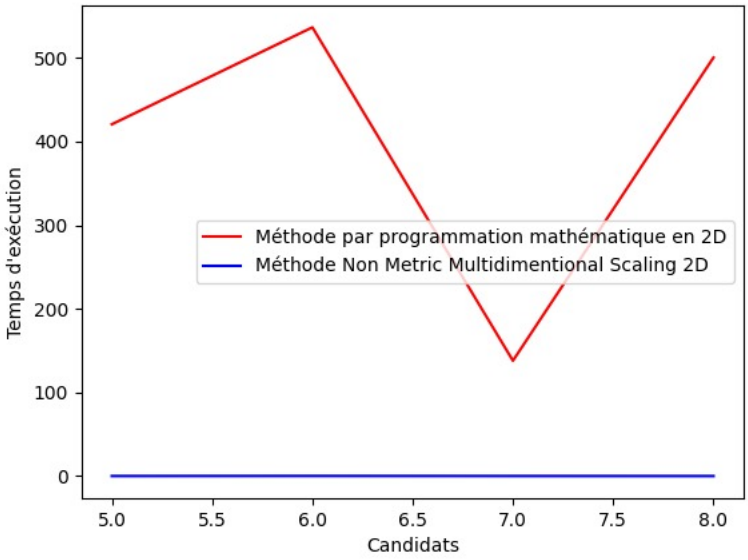

Figure 6: Courbes de temps d'exécution en fonction du nombre de candidats

Avec un nombre plus grand de fichiers en entrée et en utilisant un serveur performant en ligne, on aurait pu obtenir une courbe croissante.

Nous avons également comparé les valeurs de tau de Kendall obtenues lors de cette exécution (Figure 7). On observe un meilleur tau de Kendall pour la programmation mathématique (entre 0.0 et 0.1). Cette méthode donne donc des résultats plus pertinents en comparaison avec la méthode NMDS, pour cet indicateur.

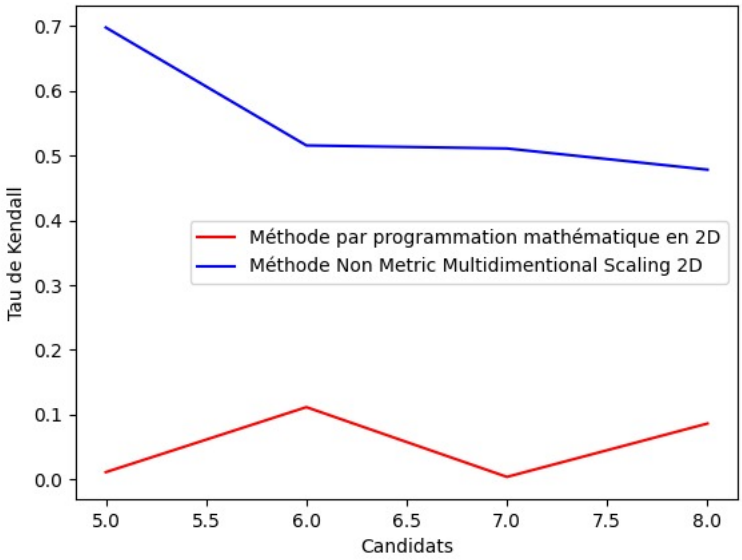

Figure 7: Courbes de tau de Kendall en fonction du nombre de candidats

4.3 Exemple de dégénérescence

On remarque un problème de dégénérescence lors de l'exécution du programme sur certains fichiers. En dimension 2, ce problème se manifeste dans l'affichage : les candidats s'entassent au même endroit et les votants s'agencent en un cercle autour des candidats.

Un exemple où nous avons observé ce problème est lors de l'exécution de la méthode NMDS avec le fichier 00049-00000001.soc (Figure 8). On note que le tau de Kendall correspondant est de 0.47.

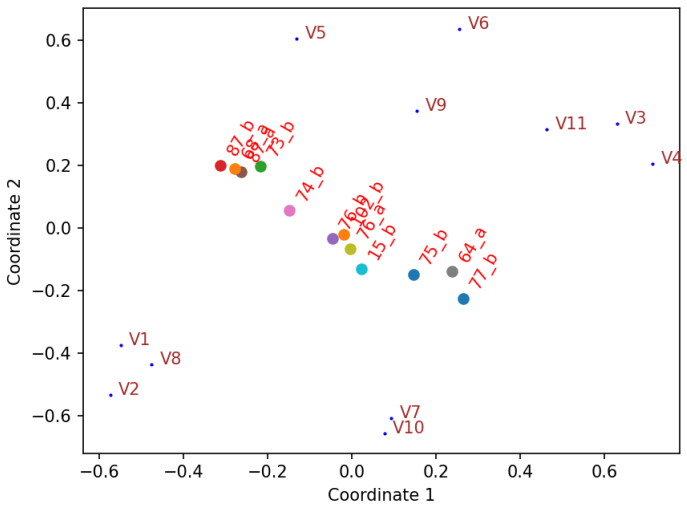

Figure 8: NMDS2 appliqué à 00049-00000001.soc

Pour cet exemple, nous observons l'affichage suivant (Figure 9) avec le Programme Mathématique. Cette méthode donne un tau de Kendall de 0.21.

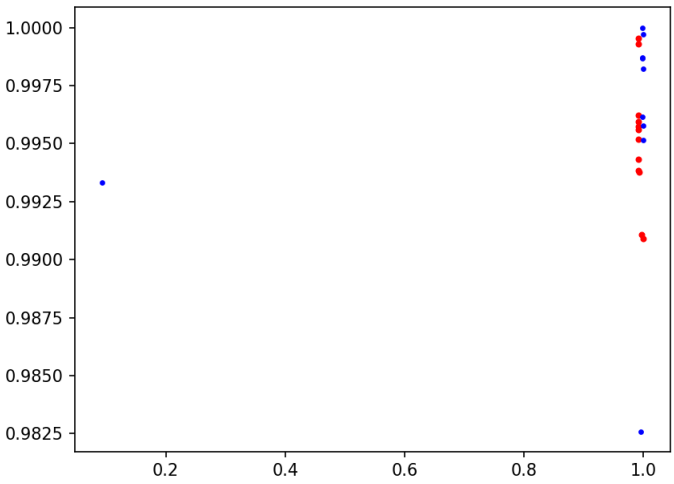

Figure 9: PM2 appliqué à 00049-00000001.soc

Pour régler le problème de la dégénérescence, nous avons ajouté une contrainte qui empêche un cluster des candidats, après avoir aussi agrandit l'intervalle des coordonnées :

```
model.addConstrs(
    ((coordonneescandidats[i, 0] - coordonneescandidats[j, 0])**2 +
    (coordonneescandidats[i, 1] - coordonneescandidats[j, 1])**2 >=1
        for i in range(nbcandidats)
        for j in range(i+1, nbcandidats)),
        name = "contrainte_dist_min_candidats")
```

L'affichage obtenu avec cette contrainte correspond à la Figure suivante :

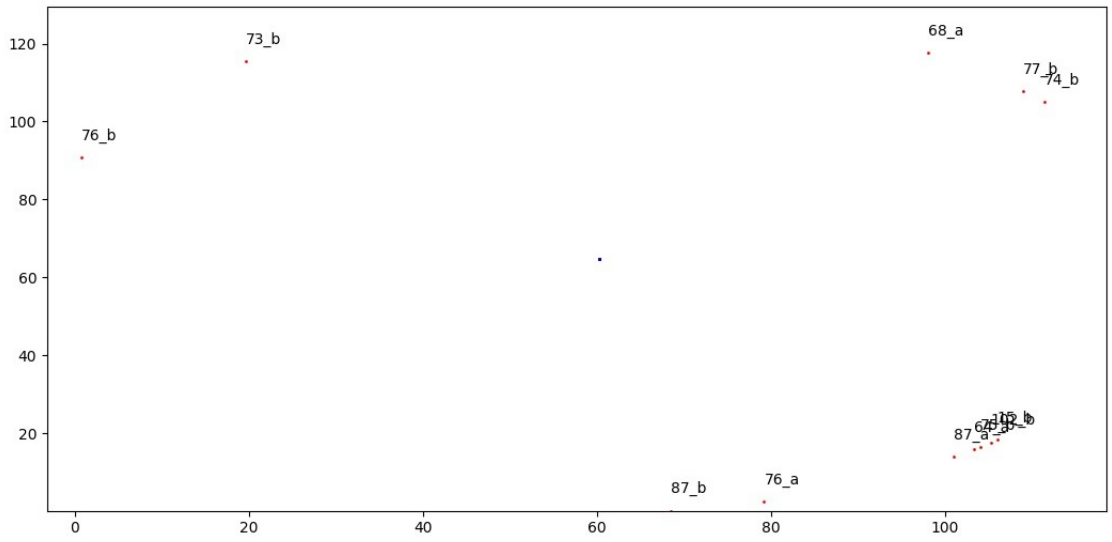

Figure 10: PM2 appliqué à 00049-00000001.soc avec la contrainte

La même contrainte, mais en dimension 1, a été également ajoutée pour les données de la présidentielle 2017, en fixant les coordonnées des candidats et des votants à l'intervalle $[0, 100]$.

Ajoutons également que la méthode NMDS ne donne pas toujours le même résultat avec un même fichier, ce qui n'est pas le cas pour la programmation mathématique.

5 Conclusion

Grâce aux données construites avec la méthode de Borda, nous avons pu placer les candidats dans un plan. Pour cela, nous avons utilisé deux méthodes différentes : la programmation mathématique et la méthode NMDS (cette dernière s'appuie sur un algorithme de gradient). L'évaluation de ces deux méthodes pour les mêmes données nous indique que la PM respecte mieux les contraintes de distances entre les votants et les candidats. Cependant, ces méthodes sont sujettes à des dégénérescences : les candidats sont proches les uns des autres, on ne peut pas bien distinguer les préférences de chaque candidat.

L'approche que nous avons utilisée pour ce projet n'est pas la seule possible. Nous avons fait le choix de minimiser les distances entres les coordonnées des candidats et des votants. On aurait aussi pu choisir de minimiser le nombre de désaccords, ce qui donnerait un tau de Kendall plus petit. Cette approche ordinale s'accompagne néanmoins d'un désavantage calculatoire : les variables données au solveur Gurobi [1] seront de type binaire, ce qui donnerait une complexité exponentielle.

Bibliographie

[1] Bob Bixby and Zonghao Gu and Ed Rothberg. *Gurobi Optimization*. Version 20.0.1. Jan. 1, 2008. URL: https://www.gurobi.com.

[2] Sean Dolibas. *MDS: Multidimensional Scaling — Smart Way to Reduce Dimensionality in Python*. 2021. URL: https://towardsdatascience.com/mds-multidimensional-scaling-smart-way-to-reduce-dimensionality-in-python-7c126984e60b.

[3] Arwa Khannoussi, Antoine Rolland, and Julien Velcin. "A multidimensional spatial model for preference representation in multi-criteria decision aiding". In: (2022), pp. 14–15.

[4] *PrefLib: A Library for Preferences*. URL: https://www.preflib.org/datasets.

[5] Vivien Rossi. *ordination methods for analyzing ecological data*. 2010.

[6] *sklearn.manifold.MDS*. 2007-2023. URL: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html.

[7] *Voter Autrement*. URL: https://vote.imag.fr/results.