Cahier des Charges
P-ANDROIDE

Auteurs :
Laura Greige
Thibault Gigant

Encadrant :
Olivier Spanjaard

2015 – 2016

Université
Pierre-et-Marie-Curie

Introduction

Habituellement, lors d'un vote, l'électeur est amené à choisir un unique candidat parmi une multitude. Ainsi, il est très facile de compartimenter les votants à partir de leur vote, connaissant les affiliations de chaque candidat. En revanche, lorsqu'il est donné la possibilité aux électeurs de ne plus voter pour un seul candidat, mais pour un sous-ensemble d'entre eux qu'il approuverait, la tache se complique. Avec cette procédure de vote, qu'on dit par approbation, il peut être intéressant de voir le problème sous un autre angle. On peut étudier les différents votes formulés et tenter d'en extraire un axe « gauche-droite » classant les candidats les uns par rapport aux autres en fonction de leur proximité.

Il existe des algorithmes pour résoudre ce problème, et dans ce projet deux principales méthodes seront utilisées :

— Réaliser un « Branch & Bound » sur l'ensemble des bulletins de vote pour identifier un sous-ensemble le plus large possible de bulletins cohérents avec un axe.

— Utiliser un algorithme de sériation permettant de calculer l'axe qui crée le moins d'incohérences possibles dans une matrice de similarité entre les candidats, créée grâce aux bulletins de vote.

Ces deux approches seront plus amplement détaillées dans la partie suivante de ce cahier des charges.

Les deux approches : Branch & Bound et Sériation

Branch & Bound

Pour trouver la solution optimale du problème, il faut utiliser une méthode exacte, comme le « Branch & Bound » qui sera représenté par un arbre de recherche binaire. Chaque sous-problème créé au cours de l'exploration est désigné par un nœud qui représente les bulletins contenus dans le sous-ensemble. Les branches de l'arbre symbolisent le processus de séparation, elles représentent la relation entre les nœuds (ajouter le bulletin i dans le sous-ensemble ou non). Cette méthode arborescente nous permettra donc d'énumérer toutes les solutions possibles.

L'algorithme d'énumération complète des solutions peut être illustré par une arborescence de hauteur n, où à chaque nœud on considère les 2 valeurs possibles pour un bulletin. En chacune des $2n$ feuilles, on a une solution possible qui correspond ou non à une solution admissible dont on peut trouver l'axe correspondant (si ce dernier existe) et on retient la meilleure solution obtenue, qui dans ce cas, sera l'ensemble le plus large de bulletins cohérent avec un axe.

Pour améliorer la complexité du « Branch & Bound », seules les solutions potentiellement de bonne qualité seront énumérées, les solutions ne pouvant pas conduire à améliorer la solution courante ne sont pas explorées.

Le « Branch & Bound » est basé sur trois principes :

— **Principe de séparation** Le principe de séparation consiste à diviser le problème en un certain nombre de sous-problèmes qui ont chacun leur ensemble de solutions réalisables. En résolvant tous les sous-problèmes et en prenant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial. Ce principe de séparation est appliqué de manière récursive à chacun des sous-ensembles tant que celui-ci contient plusieurs solutions.

Remarque : La procédure de séparation d'un ensemble s'arrête lorsqu'une des conditions suivantes est vérifiée :
– on sait que l'ensemble ne contient aucune solution admissible (cas où l'un des bulletins n'est pas cohérent avec les axes trouvés) ;
– on connaît une solution meilleure que toutes celles de l'ensemble ;

— **Principe d'évaluation** Le principe d'évaluation a pour objectif de connaître la qualité des nœuds à traiter. La méthode de « Branch and Bound » utilise deux types de bornes : une borne inférieure de la fonction d'utilité du problème initial et une borne supérieure de la fonction d'utilité. La connaissance d'une borne inférieure du problème et d'une borne supérieure de la fonction d'utilité de chaque sous-problème permet d'arrêter l'exploration d'un sous-ensemble de solutions qui ne sont pas candidats à l'optimalité.

— **Parcours de l'arbre** Le type de parcours de l'arbre permet de choisir le prochain nœud à séparer parmi l'ensemble des nœuds de l'arborescence. L'exploration en profondeur privilégie les sous-problèmes obtenus par le plus grand nombre de séparations appliquées au problème de départ, c'est-à-dire aux sommets les plus éloignés de la racine (= de profondeur la plus élevée). L'obtention rapide d'une solution admissible en est l'avantage.

Sériation

La deuxième méthode qui sera utilisée appliquera un algorithme de sériation. Le principe est de calculer une matrice dont chaque élément représente la dissimilarité entre les candidats. Celle-ci peut être calculée de plusieurs manières, et chacune sera étudiée et comparée aux autres, pour déterminer laquelle correspond le plus à la recherche de l'axe voulu. Cette valeur est un réel compris entre 0 et 1, tel que 0 représente une similarité parfaite, et 1 une dissimilarité totale. Ainsi, l'élément en position ij de la matrice donnera la dissimilarité entre le candidat i et le candidat j. Bien entendu, cette matrice est parfaitement symétrique et a pour diagonale principale des 0. Elle peut donc être considérée comme une matrice strictement triangulaire supérieur, puisque la diagonale ne changera pas et la partie inférieure n'est que le miroir de la partie supérieure.

A partir de cette matrice, pour trouver un axe parmi les candidats, il suffit alors de trouver une permutation des lignes et des colonnes (donc des candidats) qui rajoute une caractéristique à cette matrice. En effet, cette permutation doit rendre les éléments sur les lignes et les colonnes croissants au fur et à mesure que l'on s'éloigne de la diagonale. Pour ceci, deux solutions s'imposent :

— **La force brute** qui prend toutes les permutations possibles de l'axe des candidats, calcule un score à partir de la matrice de dissimilarité de chaque permutation, et renvoie celle donnant le score minimal. Cette méthode naïve sera implémentée, mais étant donnée sa complexité exponentielle, puisqu'elle demande de calculer le score des $n!$ (n étant le nombre de candidats) axes possibles, elle ne sera utilisée que pour vérifier la validité de la solution suivante, avec un nombre de candidats restreint.

— **La programmation dynamique** Algorithme inspiré des travaux de Hubert. Il permet, un peu comme avec l'algorithme de « Branch & Bound » vu précédemment, de n'explorer que les sous-ensembles des permutations qui semblent les plus prometteuses.

Conclusion

Pour implémenter ces différents algorithmes, le langage *Python* a été choisi, en utilisant la libraire *Sage* qui reprend et étend les fonctionnalités de nombreux packages préexistants. En effet, ce langage possède quantité d'outils permettant de coder rapidement et facilement les algorithmes souhaités. De plus, pour produire une interface graphique, *Python* possède une bibliothèque nommée *Tkinter*, permettant de réaliser des interfaces simples, mais dont la portabilité sur les OS les plus courants est assurée. Cette interface permettra à l'utilisateur de choisir parmi les différentes méthodes qui seront implémentées, ainsi que le fichier où sont retranscrits les différents bulletins récoltés à une élection.

Enfin, les outils développés dans ce projet seront testés sur une base de données nommée *PrefLib* disponible sur `http://www.preflib.org/`. Le travail ainsi effectué n'est donc pas uniquement théorique, il a aussi des applications réelles.