Cahier des Charges
P-ANDROIDE

Auteurs :
Laura Greige
Thibault Gigant

Encadrants :
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 première partie de ce cahier des charges. Quant à la seconde partie, elle sera dédiée à la réalisation qui sera faite de ce problème.

1 Les deux approches : Branch & Bound et Sériation

1.1 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 contenu 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 permettrait 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 candidates 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.

1.2 Sériation

2 Réalisation

2.1 Implémentation des algorithmes

Nous avons choisi d'implémenter les algorithmes en Python, en utilisant la libraire Sage qui reprend et étend les fonctionnalités de nombreux packages sous-jacentes.

2.2 Interface Utilisateur