Projet – Gerrymandering

De l'art du redécoupage électoral

Projet de programmation M1

À rendre pour le 04/01/2016, 23h59

Le Gerrymandering est une technique utilisée en politique pour favoriser une élection en redécoupant le territoire. Le but est de couper le territoire en k régions différentes afin de gagner les élections dans plus de $k/2$ régions afin d'être élu. On suppose dans la suite qu'on a seulement deux partis, A et B, que le territoire est rectangulaire et qu'on connaît pour chaque ville le nombre de personnes qui voteront pour le parti A et le nombre de personnes qui voteront pour le parti B. La figure 1, tirée de l'article wikipedia, montre un exemple de redécoupage du territoire afin que les rouges gagnent 3 régions sur 5 alors qu'ils sont minoritaires sur l'ensemble du territoire.

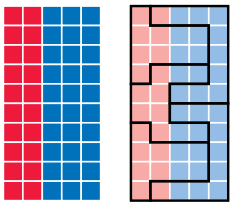

FIGURE 1 – Comment redécouper le territoire en 5 régions où 3 sont gagnées par les rouges

Modélisation

On utilisera des fichiers textes pour représenter la configuration du territoire. La structure du fichier sera la suivante :

```
n m
Nom1 x y #A #B
Nom2 x y #A #B
...
NomP x y #A #B
```

où n est la largeur du territoire, m sa hauteur. Chaque ville est repérée dans le territoire par des coordonnées entières avec x entre 0 et $n-1$ et y entre 0 et $m-1$ et on vous donne le nombre de votant pour le parti A et le nombre de votant pour le parti B dans chaque ville. Par exemple :

```
5 5
Rennes 0 1 25 30
Paris 2 1 55 10
Lille 4 0 10 45
Lyon  2 3 5 55
Marseille 2 4 47 22
```

Minimum attendu

Votre programme devra pouvoir lire les fichiers décrits à la partie précédente. On pourra l'appeler directement depuis la ligne de commande, par exemple, si votre programme s'appelle `a.out`, l'exemple suivant devra traiter le problème pour la configuration décrite dans le fichier `FICHIER` :

`./a.out FICHIER`

Le problème dans toute sa généralité est un problème très complexe et vous n'arriverez pas à obtenir le découpage optimal pour toutes les situations. On va alors s'intéresser à deux cas particuliers.

Découpage en trois rectangles On commence par un cas particulier où on cherche à découper le territoire en trois régions connexes qui doivent toutes être des rectangles. Votre programme devra être capable de trouver, s'il existe, un tel découpage qui avantage le parti A. S'il existe plusieurs découpages possibles, on choisira celui qui est "le plus équilibré" pour le nombre de votants. À vous de décider (ou laissez le choix à l'utilisateur) ce que vous considérez comme équilibré.

Découpage en deux parties équilibrés Supposons qu'on ait n villes, n pair. On cherche désormais à découper le territoire en deux sous-ensembles D_1 et D_2 de villes tels que $|D_1| = |D_2| = n/2$ et tels que le parti A gagne dans les deux districts. Les questions suivantes peuvent vous aider à trouver un algorithme par programmation dynamique. On numérote les villes $\{v_1, \dots, v_n\}$ et on note $S(j, k, x, y)$ la variable qui vaut 1 si il existe $D_1 \subseteq \{v_1, \dots, v_j\}$ avec $|D_1| = k$ et où x personnes ont voté pour le parti A dans D_1 et y personnes ont voté pour le parti A dans $D_2 = \{v_1, \dots, v_j\} \setminus D_1$.

1. Comment calculer $S(j+1, k, x, y)$ à partir des $S(j, \dots)$?
2. Comment peut-on en déduire s'il existe un découpage équilibré ?
3. Comment peut-on retrouver ce découpage ?

Aller plus loin

Le problème est tellement vaste qu'on peut en faire un peu plus. Je donne quelques pistes ici pour les plus curieux. Rien ne vous empêche de proposer vos propres généralisations/idées :

— Essayer de trouver des découpages en k rectangles. On pourra commencer par 4 rectangles.
— Améliorer le modèle pour qu'on puisse tenir compte d'incertitudes sur le nombre de votants. On cherchera alors à découper le territoire pour avoir une bonne probabilité d'être élu.
— Essayer et critiquer des heuristiques pour trouver des découpages connexes qui ne sont pas des rectangles. On commencera par couper en deux. Par exemple, on peut chercher les deux villes où le parti A a le plus gros avantage et faire grossir les régions autour de chaque ville. Ou on pourrait partir d'un découpage en deux non-connexe, comme décrit à la partie précédente, et d'essayer de le rendre connexe, etc.

Modalités

Ce projet peut être réalisé *en binôme* (c'est d'ailleurs un bon exercice de mener un projet de programmation à deux). Il est à rendre par mail (`fcapelli@math.univ-paris-diderot.fr`) avant le **04/01/2016, 23h59**. Il comportera le code source de vos programmes, *indenté et commenté*. Ce code source doit compiler avec `gcc -Wall`. La lisibilité et l'organisation du code seront un facteur important de la note finale. Il doit être accompagné d'un rapport de 4 à 8 pages et de fichiers exemple sur lesquels vous avez testé votre algorithme.

Le rapport devra présenter une justification des structures de données utilisées, une explication de vos algorithmes (fonctionnement, analyse de complexité, approximation etc.) et la présentation de quelques exemples bien choisis pour illustrer les avantages, désavantages ou limites des méthodes choisies. Si vous avez des solutions qui ne marchent pas toujours, des bugs, c'est la bon endroit pour en parler et commenter vos méthodes. Je préfère voir que vous êtes conscients de la présence d'un bug que vous n'arrivez pas à résoudre et des limites de vos méthodes plutôt que vous les cachiez.

Une soutenance de 30mn par binôme sera organisée à la rentrée en janvier où vous me présenterez brièvement votre travail à l'oral pendant 10mn et où je vous poserai des questions pendant 20mn.