TIPE : Utilisation de l'algorithme de Kuhn pour l'attribution de logements étudiants

Carole Vacherand

2022-2023

1 - Introduction

Les institutions d'enseignement supérieur étant en majorité situées dans les grandes villes, celles-ci concentrent les étudiants. Le logement étant un enjeu majeur dans les grandes villes, il l'est d'autant plus pour les étudiants que leur attribution se fait à grande échelle et sur un temps court. Il semble alors adapté d'utiliser un algorithme d'attribution.

Cependant, les préférences des étudiants en matière de logement ne sont pas toutes les mêmes. Il convient donc de trouver une manière de répartir les logements de telle sorte que les étudiants habitent un logement au plus proche de leur souhait. On ne s'intéressera pas ici à un éventuel ordre de priorité entre les étudiants ; seuls leurs vœux en matière de logement seront considérés.

L'objectif de ce document est donc d'appliquer l'algorithme de Kuhn-Munkres (aussi connu sous le nom d'algorithme hongrois) à l'attribution de logements à des étudiants demandeurs. On présentera d'abord la manière dont nous modéliserons ce problème. Nous mettrons ensuite en place l'algorithme et analyserons les résultats obtenus afin de conclure sur la viabilité de son utilisation dans un contexte concret.

2 - Modélisation du problème

Pour pouvoir attribuer leur logement, on commence à demander aux étudiants de dire s'ils préfèrent une caractéristique du logement (taille, étage, bâtiment, …). Notre objectif est donc de trouver la répartition des logements la plus proche des vœux des élèves. On implémentera pour cela l'algorithme en langage C.

A - Calcul des scores

Pour quantifier la satisfaction d'un élève par rapport au logement obtenu, on calcule un « coût », qui, qualitativement, est un score d'incompatibilité. Plus il est

bas, plus l'élève est satisfait (score 0 si le logement correspond exactement à ses attentes) ; plus il est grand, plus l'élève est insatisfait. On utilise des entiers, ce qui permet d'éviter des problèmes de précision avec les flottants, mais aussi de les interpréter comme des booléens, ce qui sera utile pour l'une des étapes de l'algorithme.

Ensuite, on regroupe ces coûts dans une matrice : une ligne correspond à un locataire et une colonne à une chambre. La case (i,j) contient le coût d'associer l'élève i à la chambre j. L'algorithme va ensuite effectuer des calculs sur cette matrice.

On notera désormais cette matrice C, ses coefficients seront les $C_{i,j}$.

Chambre \ Locataire	0	1	2	3
A	2	4	2	5
B	6	3	7	4
C	4	5	6	2
D	6	8	9	3

Exemple de matrice de coût pour n = 4.

B - Critère d'optimalité

On cherche ici à minimiser l'insatisfaction globale. En notant n le nombre de locataires (supposé égal au nombre de chambres, cas auquel l'on pourra toujours se ramener), on cherche donc à choisir une permutation σ de $[\![1, n]\!]$ telle que la somme

$$\sum_{i=0}^{n-1} C_{i,\sigma(i)}$$

soit minimale. Il est notable que l'algorithme de Kuhn permet précisément de remplir cette condition.

C – Cas simple

Il existe une situation dans laquelle le choix peut être fait immédiatement : lorsque l'on peut choisir σ telle que

$$\sum_{i=1}^{n-1} C_{i,\sigma(i)} = 0$$

En effet, cette somme étant positive, on sait alors qu'elle est minimale. Qualitativement, chaque étudiant a alors obtenu un logement correspondant exactement à ses vœux.

Chambre \ Locataire	0	1	2	3
A	0	4	0	5
B	1	0	2	1
C	0	3	2	0
D	1	5	4	0

Exemple d'une telle matrice et permutation choisie (chiffres entourés)

Cependant une telle permutation n'existe pas toujours. Le principe de l'algorithme de Kuhn est alors de se ramener à ce cas.

3 - Présentation de l'algorithme

L'algorithme est composé d'une initialisation puis d'une boucle.

Initialisation : - Calculer les scores à partir des données.

- Soustraire à chaque ligne le minimum de la ligne.

- Soustraire à chaque colonne le minimum de chaque colonne.

Boucle principale :

- Couvrir des lignes et des colonnes de telle sorte que tous les zéros de la matrice soient couverts ; en couvrant le minimum de lignes/colonnes possibles. Soit p le nombre de couvertures. Si p = n, sortir de la boucle.

- Ajouter aux éléments couverts le minimum des éléments non couverts.

-Soustraire à tous les éléments le minimum de la matrice.

Choix final : Choisir un ensemble de zéros de telle sorte que chaque colonne et chaque ligne en ait exactement un.

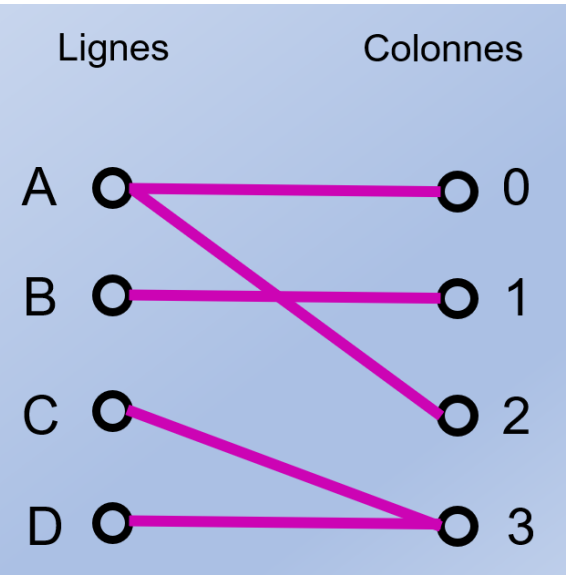
Algorithme de Kuhn.

L'étape cruciale est celle de la 'couverture' des zéros. Pour la réaliser, on interprète la matrice comme une demi-matrice d'adjacence d'un graphe biparti :

$$G = ((A \cup B), E) \text{ où } A = \{A_i, i \in [\![1; n]\!]\} \text{ et } B = \{B_i, i \in [\![1; n]\!]\}$$

$$C_{i,j} = 0 \iff (A_i, B_j) \in E$$

Chambre \ Locataire	0	1	2	3
A	0	2	0	3
B	3	0	4	1
C	2	3	4	0
D	3	5	6	0



Une matrice et le graphe biparti correspondant.

Pour calculer la couverture cherchée, il suffit alors de déterminer une couverture minimale dans ce graphe. Le graphe étant biparti, cette couverture est calculable à partir d'un couplage maximum, par la preuve constructive du théorème de König :

> *Théorème de König.*
>
> Soit G un graphe biparti. Le cardinal d'un couplage maximum est égal au cardinal d'une couverture minimale.

Preuve :

Soit $G = ((A \cup B), E)$ un graphe biparti. Soit $C \subset E$ un couplage maximum de ce graphe.

On pose $Z = A \setminus C$ et U l'ensemble des sommets atteignables à partir de Z par des chemins alternants.

On pose $Z' = Z \cup U$ et enfin $S := (A \setminus Z') \cup (B \cap Z')$.

Alors S est une couverture minimale de cardinal $Card(C)$.

Pour le montrer, nous avons besoin d'un lemme intermédiaire.

> *Lemme 1 :*
> $\forall (a,b) \in (A \times B) \cap C \quad a \in Z \iff b \in Z$

Preuve du lemme 1 :

$\Longrightarrow$: Soit (a,b) une arête du couplage. On suppose que $a \in Z$. Alors a appartient à Z car il est accessible par un chemin alternant (en effet il ne peut être ajouté à Z dès le début car il est saturé). Ce chemin a dernièrement emprunté une arête couplée, donc $b \in Z$.

$\Longleftarrow$: Soit (a, b) une arête du couplage. On suppose que $b \in Z$. Alors a peut être atteint par un chemin alternant à partir de Z, en passant par b. Donc $a \in Z$.

Nous pouvons donc maintenant prouver que S défini plus tôt est bien une couverture, de cardinal souhaité.

S est une couverture :

Soit (a, b) une arête de G. Montrons qu'elle est couverte par S.

Si $a \notin Z$: alors $a \in S$ donc (a, b) est couverte.

Sinon $a \in Z$:

Si $(a, b) \in C$: alors b est atteignable à partir d'un sommet de Z par un chemin alternant. Donc $b \in Z$, par construction de S on a $b \in S$ donc (a, b) est couverte.

Sinon $(a, b) \notin C$: de même.

S est de cardinal $Card(C)$*:*

On a nécessairement $Card(S) \geqslant Card(C)$, car chaque arête de C doit être couverte par un sommet distinct (en effet, elles n'ont aucun sommet en commun).

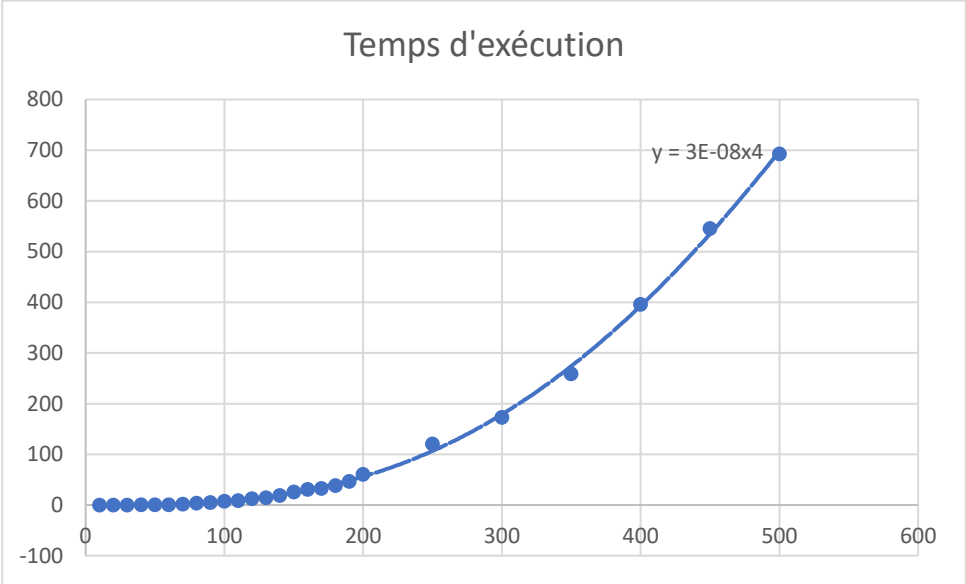
Il suffit donc de montrer que $Card(S) \leqslant Card(C)$ pour montrer que la couverture est minimale, et on aura de plus l'égalité des cardinaux.

C'est le cas car chaque arête de C a exactement un de ses sommets dans S par construction.

Une fois cette couverture minimale calculée, on a deux issues : si elle est de taille n, le théorème affirme qu'il existe un couplage parfait du graphe. On peut donc bien choisir exactement un zéro par ligne et par colonne, donc faire une attribution définitive et sortir de la boucle. Sinon, on effectue des opérations qui garantissent que l'on fait apparaître un zéro supplémentaire, et l'on relance la boucle.

A - Résultats obtenus

L'algorithme s'exécute en temps raisonnable (jusqu'à 11 minutes) pour n allant jusqu'à 500. On retrouve sa complexité théorique $O(n^4)$ (courbe d'approximation), qui est due en majeure partie au calcul du couplage maximum (les autres opérations étant linéaires ou quadratiques).



Temps d'exécution (en secondes) de l'algorithme en fonction de n.

B - Voies d'amélioration

L'algorithme étant déjà optimal pour le problème que nous cherchons à résoudre, il n'est pas possible d'obtenir de meilleurs résultats sans changer la définition de l'optimalité que nous avons adoptée.

On peut cependant penser à des améliorations sur le plan de la complexité. En effet, pour calculer le couplage maximal, ce qui est l'opération qui prend le plus de temps, on a ici utilisé l'algorithme d'Edmonds qui est en $O(|E||V|^2)$. Il existe un algorithme de complexité inférieure, l'algorithme de Hopcroft-Karp, en $O(|E||V|^{1/2})$; cependant sa complexité dépend en grande partie de la densité du graphe, ce qui pose un problème ici. Par exemple, si un grand nombre de locataires émet exactement les mêmes vœux, il est possible d'obtenir un grand nombre de zéros dans la matrice après initialisation, et donc un graphe très dense. Cela pourrait réduire l'intérêt de l'utilisation de cet algorithme.

4 – Conclusion

Le temps raisonnable d'exécution de cet algorithme permet de conclure que son utilisation dans un contexte concret est possible. Cependant, quelques-unes de ses caractéristiques expliquent que son usage ne soit pas systématique dans cette situation. Il est notamment impossible d'introduire un quelconque ordre de priorité dans les choix des locataires. On peut aussi citer le fait que les attributions se font au hasard au sein d'un groupe d'élèves qui auraient fait les mêmes vœux,

sans considération pour d'autres critères comme les affinités qui n'auraient pas été pris en compte.

Bibliographie :

Cindy Fouraker, Akanksha Kartik, Devin Noh, Harika Veda (Carnegie Mellon University) : Optimizing Freshmen Dorm Assignment : (projet de fin du cours Operations Research II) https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjhw4j-suP8AhUIXaQEHQwoD74QFnoECA8QAQ&url=https%3A%2F%2Fwww.math.cmu.edu%2F~af1p%2FTeaching%2FOR%2FProjects%2FP64%2FORFinalProject.pdf&usg=AOvVaw2Sd4ySfPDdqpWVbCcFFWsz

H. W. Kuhn : The Hungarian method for the assignment problem : Naval Research Logistics, 2: 83-97