SAE_R4_04_GroupeSAE-9_A21

Bastien ALLEGRE, Hugo BARBIERI, Emeric DIEUDONNE, Amaury GAU, Louis VICAT

2025-02-26

Introduction

Dans ce projet, nous abordons le problème d'affectation des stages aux étudiants en utilisant l'**algorithme hongrois**. L'objectif est d'attribuer à chaque étudiant un stage en fonction de ses préférences tout en optimisant l'affectation globale. cela se ramène à un problème **d'optimisation combinatoire**. Soit :

p le nombre d'étudiants,

n le nombre de stages avec $n \geq p$.

En cas de $n > p$, des étudiants fictifs peuvent être ajoutés afin d'obtenir une matrice carrée, sans influencer la solution optimale.

Modélisation du problème

Hypothèses

- Chaque étudiant classe les stages en fonction d'un **ordre de préférence** exprimé sous forme d'un coût.
- La matrice de préférences est **carrée**, si $n > p$ nous complétons par des étudiants ficitfs avec des couts sans influences sur l'optimisation
- L'objectif est de minimiser le **coût total d'affectation**. défini par :

$min \sum_{i=1}^{n} c_i, \sigma(i)$

où σ est une permutation représentant l'affectation.

Représentation matricielle

Pour modéliser ce problème, nous utilisons une matrice des préférences C de dimension nÖn, où chaque élément $C_{i,j}$ représente le coût (ou l'inverse de la préférence) de l'affectation de l'étudiant i au stage j. Ainsi, plus $C_{i,j}$ est faible, meilleure est la correspondance entre l'étudiant et le stage

Soit C la matrice des préférences où sont représenté la préférence de l'étudiant i pour le stage j.

$$C = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots \\ c_{n1} & c_{n2} & \dots & c_{nn} \end{bmatrix}$$

Algorithme utilisé pour résoudre ce problème :

Algorithme hongrois

L'algorithme hongrois est un algorithme de compléxité $\mathbf{O(n^3)}$. Il résound le problème d'affectation en minimisant la fonction coût.Il fonctionne en plusieurs étapes d'abord en ajustant une matrice de coûts pour

créer des zéros, puis en trouvant une affectation optimale :

Etape 1. Normalisation des préférences

Pour chaque ligne i, on soustrait le minimum de la ligne : $C'_{ij} = C_{ij} - min_j\{c_{ij}\}$ Puis, pour chaque colonne j, on soustrait le minimum de la colonne obtenue : $C''_{ij} = C'_{ij} - min_i\{c'_{ij}\}$ Ce processus garantit qu'il y aura au moins un zéro dans chaque ligne et chaque colonne.

Etape 2. Marquage des zéros

On identifie une première affectation en marquant les zéros de façon à ce que : - Chaque zéro marqué représente une affectation potentielle. - Une même ligne ou colonne ne peut contenir plus d'un zéro marqué.

Etape 3. Vérification de l'optimalité

Si l'affectation initiale compte n affectations (pour une matrice $n \times n$), la solution est optimale. Sinon, il faut procéder à un ajustement de la matrice.

Etape 4. Ajustement de la matrice :

Pour les éléments non couverts par les lignes ou colonnes contenant les zéros marqués :

1. Trouver la plus petite valeur non couverte δ
2. Soustraire δ de tous les éléments non couverts
3. Ajouter δ aux éléments situés à l'intersection des lignes et colonnes couvertes

Ces opérations permettent de créer de nouveaux zéros et, par itérations successives, d'améliorer l'affectation jusqu'à obtenir une solution optimale

Implémentation en R

```r
# Génération aléatoire d'une matrice représentant les préférences des stages pour les étudiants
set.seed(123)
n <- 4  # Nombre d'étudiants/stages
C <- matrix(sample(1:10, n*n, replace=TRUE), nrow=n, ncol=n)
print("Matrice initiale de la matrice représentant les préférences:")
```

```
## [1] "Matrice initiale de la matrice représentant les préférences:"
```

```r
print(C)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    3    6    9    9
## [2,]    3    5   10    9
## [3,]   10    4    5    9
## [4,]    2    6    3    3
```

```r
# Étape 1: Soustraction du minimum de chaque ligne
row_min <- apply(C, 1, min)
C <- sweep(C, 1, row_min, FUN="-")
print("Après soustraction du minimum de chaque ligne:")
```

```
## [1] "Après soustraction du minimum de chaque ligne:"
```

```r
print(C)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0    3    6    6
## [2,]    0    2    7    6
## [3,]    6    0    1    5
## [4,]    0    4    1    1
```

```r
# Étape 2: Soustraction du minimum de chaque colonne
col_min <- apply(C, 2, min)
C <- sweep(C, 2, col_min, FUN="-")
print("Après soustraction du minimum de chaque colonne:")
```

```
## [1] "Après soustraction du minimum de chaque colonne:"
```

```r
print(C)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0    3    5    5
## [2,]    0    2    6    5
## [3,]    6    0    0    4
## [4,]    0    4    0    0
```

```r
# Fonction pour trouver l'affectation optimale
find_assignment <- function(C) {
  n <- nrow(C)
  assignment <- rep(NA, n)
  covered_rows <- rep(FALSE, n)
  covered_cols <- rep(FALSE, n)

  repeat {
    # Marquage des zéros avec un X pour les affectations initiales
    for (i in 1:n) {
      for (j in 1:n) {
        if (C[i, j] == 0 && !covered_rows[i] && !covered_cols[j]) {
          assignment[i] <- j
          covered_rows[i] <- TRUE
          covered_cols[j] <- TRUE
          break
        }
      }
    }

    # Vérifier si une affectation complète est obtenue
    if (all(!is.na(assignment))) break

    # Ajustement de la matrice pour les éléments non couverts
    uncovered_values <- C[!covered_rows, !covered_cols, drop=FALSE]
    min_value <- min(uncovered_values)
    C[!covered_rows, !covered_cols] <- C[!covered_rows, !covered_cols] - min_value
    C[covered_rows, covered_cols] <- C[covered_rows, covered_cols] + min_value
  }
  return(assignment)
}

# Résolution du problème
resultat <- find_assignment(C)
print("Affectation optimale:")
```

```
## [1] "Affectation optimale:"
```

```
print(resultat)
```

```
## [1] 1 4 2 3
```

Résultats et interprétation

- **Matrice initiale** : Représente les coûts initiaux associés aux préférences.
- **Étapes de réduction** : Permettent de créer une matrice avec des zéros facilitant l'identification de l'affectation optimale.
- **Résultat final** : Le vecteur resultat indique pour chaque étudiant (par indice de ligne) le stage (par indice de colonne) qui minimise le coût total.

Conclusion et perspectives

Conclusion :

Nous avons utilisé l'algorithme hongrois pour résoudre le problème d'affectation des stages en minimisant le coût total. La méthode s'appuie sur une réduction itérative de la matrice des coûts et sur un marquage stratégique des zéros pour identifier une affectation optimale.

Perspectives d'amélioration :

- **Prise en compte de contraintes supplémentaires :**

intégrer des pondérations liées aux compétences des étudiants ou aux spécificités des stages.

- **Modification de certaines hypothèses :**

Remplacer l'hypothèse d'un coût unique par étudiant-stage par une modélisation multi-critères.

Gérer de manière dynamique le cas $n > p$ en attribuant un coût variable aux étudiants fictifs selon des critères définis.