Rapport Projet C : Organisation de journées culturelles

(spectacles + déjeuner + concerts)

MU Maxime (IMT Mines Alès / Élève)
18/03/2023

IMT Mines Alès
École Mines-Télécom

Table des matières

Lien GitHub : https://github.com/Ayfred/CulturalDayOrganization

Question 1

I. Contexte

L'objectif est de planifier **l'affectation de toutes les familles** enregistrées pour une semaine donnée tout en respectant un **ensemble de contraintes** et en **minimisant** une fonction de coût **non linéaire**. Cette planification doit être faite pour **chaque jour** de la semaine au cours d'une période d'une semaine.

II. Objectif

L'objectif de la société organisatrice est de réduire au minimum la somme des indemnités versées aux familles et aux personnels mobilisés tout en respectant les contraintes imposées.

La fonction à minimiser est la fonction objective (fonction économique) est donc :

$$\mathbf{F} = \sum_{f=0}^{f=nbFamille} \boldsymbol{Cout}(\mathbf{f}) + \sum_{j=0}^{j=6} \boldsymbol{Penalite}(\mathbf{j})$$

Minimiser la fonction objective revient à minimiser les termes dans la somme à savoir :

- $\sum_{f=0}^{f=nbFamille} \boldsymbol{Cout}(\mathbf{f})$
- $\sum_{j=0}^{j=6} \boldsymbol{Penalite}(\mathbf{j})$

III. Données

On dispose de cinq fichiers *.csv, un tableau contenant les données des familles à savoir la taille de la famille et leurs cinq préférences en ordre (de 0 à 6).

IV. Principe de résolution

Chaque famille est affectée à sa première préférence (numéro 1) dans l'ordre de priorité tant qu'il y a de la place disponible. Si une famille ne peut pas être affectée à son choix numéro 1, elle sera affectée à son choix numéro 2 s'il y a de la place, et ainsi de suite jusqu'à ce qu'elle ne puisse être affectée à aucun de ses choix. Dans ce cas, la famille sera affectée au jour où la participation est la plus faible. Ce processus permet de maximiser les affectations en fonction des préférences des familles tout en évitant une surcharge des jours où il y a déjà beaucoup de participants.

V. Explications

En accordant aux familles leurs choix prioritaires, on diminue les risques d'indemnités élevées car plus on attribue les choix moins préférés, plus les indemnités sont importantes. En outre, en affectant les familles au jour où la participation est la plus faible, les délais entre les affectations sont réduits, ce qui limite les éventuelles pénalités et réduit les coûts globaux.

VI. Réalisation

La méthode void « **readData** » prend en paramètres le nom « fileName » du fichier *.csv, et un tableau 3D « families » que l'on va remplir à partir des données du fichier. Cette méthode donne également le nombre de lignes et le nombre de colonnes que le tableau « families » possède.

La méthode « **assignFamilies** » prend en paramètres le nombre de lignes « num_rows », le nombre de colonnes « num_colomns » et le tableau « families ». Cette méthode va affecter les familles aux jours correspondant à leur choix. Elle va appeler les méthodes « indemnity » et « penality » pour le calcul des coûts. Elle retournera la valeur du coût total (fonction objectif) à savoir la somme des indemnités et des pénalités.

La méthode « **indemnity** » prend en paramètres le choix de la famille « family_choice » et le nombre de membres dans cette famille « number ». Elle retourne un entier correspondant aux indemnités dues à la famille.

La méthode « **penality** » prend en paramètres la capacité du jour n°j « capacityOfToday », la capacité du jour n°j+1 « capacityOfTomorrow » et le jour actuel j. Elle retourne un entier correspondant aux pénalités que doit la société à ses employés.

La méthode void « **showResults** » prend en paramètre un entier « i » correspondant au numéro de position du fichier csv de travail. La méthode consiste à afficher dans la console :

- Le nom du tableau qu'on travaille.
- Le nombre de lignes et le nombre de colonnes du tableau.
- La distribution des familles en fonction des jours.
- Le nombre total des personnes.
- Le résultat de la fonction objectif.

La méthode void « **resetData** » consiste à effacer tous les contenus des variables globales afin de travailler sur le fichier .csv suivant.

VII. Résultats (sans optimisations)

Avec « **max_capacity = 250** » :

Fichier « pb10.csv » :

Jours	0	1	2	3	4	5	6
Nombres de personnes	271	285	284	294	265	286	250

Coûts : 3331 €

Fichier « pb20.csv » :

Jours	0	1	2	3	4	5	6
Nombres de personnes	262	259	261	258	254	263	250

Coûts : 1928 €

Fichier « pb30.csv » :

Jours	0	1	2	3	4	5	6
Nombres de personnes	216	191	230	222	247	247	219

Coûts : 421 €

Fichier « pb40.csv » :

Jours	0	1	2	3	4	5	6
Nombres de personnes	212	218	238	250	237	165	172

Coûts : 939 €

Fichier « pb50.csv » :

Jours	0	1	2	3	4	5	6
Nombres de personnes	219	246	174	201	168	200	235

Coûts : 858 €

Avec « **max_capacity = 300** » :

Fichier « pb10.csv » :

Jours	0	1	2	3	4	5	6
Nombres de personnes	269	286	300	244	237	299	300

Coûts : 1330 €

Fichier « pb20.csv » :

Jours	0	1	2	3	4	5	6
Nombres de personnes	288	241	272	291	228	222	265

Coûts : 646 €

Fichier « pb30.csv » :

Jours	0	1	2	3	4	5	6
Nombres de personnes	211	187	230	215	275	249	205

Coûts : 202 €

Fichier « pb40.csv » :

Jours	0	1	2	3	4	5	6
Nombres de personnes	212	215	238	264	237	161	165

Coûts : 1152 €

Fichier « pb50.csv » :

Jours	0	1	2	3	4	5	6
Nombres de personnes	219	246	174	201	168	200	235

Coûts : 858 €

VIII. Interprétations

On remarque qu'avec une capacité de 250, les affectations des familles dépassent la capacité maximale et qui a donc un impact conséquent sur le coût total.

Tandis qu'avec une capacité de 300, il semblerait que seulement certains jours sont affectés par la saturation et a donc un impact moins conséquent sur le coût total. Cela se voit sur les autres fichiers, on remarque une baisse des coûts totaux. On peut remarquer que le fichier pb40.csv présente une augmentation des coûts, qui est due aux pénalités engendrées par les importantes variations de population entre les jours de la semaine.
Bien que mon algorithme d'affectation des familles permet d'optimiser les coûts en affectant les familles de manière efficace, il comporte une limite : si la distribution des familles est déséquilibrée entre deux jours consécutifs, cela peut entraîner une augmentation des coûts.

IX. Optimisations de l'algorithme

Je me suis intéressé à l'optimisation de mon algorithme avec deux méthodes :

- Méthode avec le tri par **ordre croissant** : on trie la liste des familles par ordre croissant par rapport au nombre de membres de famille (première colonne). Cette méthode est **efficace** que **si** les familles sont dites des « petites » familles en termes de membres de famille. Ici dans les cinq fichiers, on a plus de « petites » familles que de « grandes » familles. Dans le cas contraire, si on avait plus de « grandes » familles que de « petites » familles, alors on aurait adopté la méthode de tri par ordre décroissant.

  Méthodes : « **compare** » compare deux valeurs a et b.
  « **sortDataAscending** » trie les valeurs de la matrice par ordre croissant (tri bulle).

Pour le pb10.csv :
Capacité/jour maximale : 250 | Coût Total = **1562 €**
Capacité/jour maximale : 300 | Coût Total = **883 €**

Pour le pb20.csv :
Capacité/jour maximale : 250 | Coût Total = **757 €**
Capacité/jour maximale : 300 | Coût Total = **646 €**

Pour le pb30.csv :
Capacité/jour maximale : 250 | Coût Total = **344 €**
Capacité/jour maximale : 300 | Coût Total = **202 €**

Pour le pb40.csv :
Capacité/jour maximale : 250 | Coût Total = **1242 €**
Capacité/jour maximale : 300 | Coût Total = **1152 €**

Pour le pb50.csv :
Capacité/jour maximale : 250 | Coût Total = **858 €**
Capacité/jour maximale : 300 | Coût Total = **858 €**

- Méthode de tri **aléatoire** : on trie la liste des familles de manière aléatoire par rapport au nombre de membres de familles. (**Itération à ne pas dépasser : environ 550** sinon l'algorithme nous donnera un résultat égal à **0**)

  Méthodes : « **shuffle** » trie de manière aléatoire les valeurs de la matrice.

  « **findMinimalTotalCost** » trouve dans la liste des coûts totaux le minimal.

Pour le pb10.csv :
Capacité/jour maximale : 250 | Coût Total = **1130 €**
Capacité/jour maximale : 300 | Coût Total = **599 €**

Pour le pb20.csv :
Capacité/jour maximale : 250 | Coût Total = **887 €**
Capacité/jour maximale : 300 | Coût Total = **646 €**

Pour le pb30.csv :
Capacité/jour maximale : 250 | Coût Total = **225 €**
Capacité/jour maximale : 300 | Coût Total = **202 €**

Pour le pb40.csv :
Capacité/jour maximale : 250 | Coût Total = **373 €**
Capacité/jour maximale : 300 | Coût Total = **1152 €**

Pour le pb50.csv :
Capacité/jour maximale : 250 | Coût Total = **858 €**
Capacité/jour maximale : 300 | Coût Total = **858 €**

On obtient un meilleur résultat en adoptant la **méthode de tri aléatoire** quel que soit la capacité journalière.

On remarque également que les résultats pour le pb40.csv et pb50.csv car ils sont dus au fait que toutes les familles ont été attribuées à leur choix préféré, on ne plus optimiser le résultat avec mon algorithme. Cependant en adoptant une autre approche qui serait de prioriser l'égalité des distributions pourrait optimiser ces coûts.

Question 2

Les familles sont assignées dans l'ordre décroissant des pourcentages, puis attribuées au jour avec la plus faible participation pour une meilleure gestion du flux et pour éviter une surcharge les jours où il y a déjà beaucoup de participants. Cette stratégie de répartition des familles aide à répartir équitablement les coûts de l'organisation sur tous les jours de l'événement, ce qui peut contribuer à réduire les coûts globaux.
Par exemple pour l'événement 1, on affecte les familles au jour 4 avec le moins de participants 13.32%.
Idem pour l'événement 2, on affecte les familles au jour 6 avec le moins de participants 13.83%.
Et pour l'événement 3, on affecte les familles au jour 1 avec le moins de participants 12.74%

Une option d'amélioration consiste à prioriser l'affectation des familles avec le nombre de membres le moins élevé afin de minimiser les indemnités à verser.