Rendu 03 - Mathématique de la décision

Groupe BMR

16 Octobre 2019

1 Objectif

1.1 Description

L'objectif de ce projet est de trouver la meilleure méthode pour répartir la promotions des IG5 en groupe de 2 et 3 étudiants, selon 18 projets PIFE.

Pour ce faire, on obtient les préférences de chaque élève sur les autres élèves et les projets disponible. Une préférence est exprimé ainsi : Très bien (TB), Bien (B), Assez Bien (AB), Passable (P), Insuffisant (I), A Rejeter (AR).

Ainsi nos données sont sous forme de deux matrices, la première représentant les préférences de chaque élève selon les autres élèves et la seconde représentant les préférences des élèves selon les projets disponible.

L'objectif étant de satisfaire un maximum les élèves, il nous faut mettre en place une échelle de satisfaction. On considère donc les niveaux de satisfaction suivant : maximale, élevée, moyenne, faible, nulle. Ainsi :

- Satisfaction maximale : l'élève est dans groupe de 1 ou 2 autres élèves pour lesquels il a mis une préférence TB, et qu'il traite un sujet qu'il a aussi préféré à TB.
- Satisfaction élevée : l'élève est dans un groupe de 1 ou 2 autres élèves pour lesquels il a mis une préférence B ou TB, et qu'il traite un sujet qu'il a aussi préféré à B ou TB.
- Satisfaction moyenne : l'élève est associé avec un élève ou un projet préféré à P ou AB
- Satisfaction faible : l'élève est associé avec un élève ou un projet préféré à I.
- Satisfaction nulle : l'élève est associé avec un élève ou un projet préféré à AR.

Pour maximiser la satisfaction des élèves, il nous faut donc les répartir tel qu'on maximise les préférences entre les élèves de chaque groupe, tout en maximisant les préférences de chaque élève avec son projet.

Afin de réaliser cette répartition, il convient d'affecter en premier lieu, les élèves ayant les préférences les plus basses. En effet si on arrive à satisfaire ceux qui ont le moins de chance de l'être, il nous reste alors plus de possibilités pour essayer de satisfaire un maximum d'élèves.

1.2 Exemple

1.2.1 enumération des possibilités

```
il y a  3 répartitions possibles en prenant  4  élèves

voici les enumerations :
1  :  [['d', 'b'], ['c', 'a']]  ,  2  :  [['a', 'b'], ['c', 'd']]
3  :  [['a', 'd'], ['c', 'b']]


il y a  10 répartitions possibles en prenant  5  élèves

voici les enumerations :
1  :  [['c', 'a', 'b'], ['e', 'd']]  ,  2  :  [['c', 'a', 'd'], ['e', 'b']]
3  :  [['d', 'a', 'b'], ['e', 'c']]  ,  4  :  [['c', 'd', 'b'], ['e', 'a']]
5  :  [['c', 'a'], ['e', 'b', 'd']]  ,  6  :  [['a', 'd'], ['e', 'b', 'c']]
7  :  [['a', 'b'], ['e', 'c', 'd']]  ,  8  :  [['c', 'b'], ['e', 'a', 'd']]
9  :  [['c', 'd'], ['e', 'a', 'b']]  ,  10  :  [['b', 'd'], ['e', 'a', 'c']]


il y a  25 répartitions possibles en prenant  6  élèves

voici les enumerations :
1  :  [['b', 'f'], ['c', 'a'], ['e', 'd']]  ,  2  :  [['a', 'f'], ['c', 'b'], ['e', 'd']]
3  :  [['a', 'b'], ['c', 'f'], ['e', 'd']]  ,  4  :  [['d', 'f'], ['c', 'a'], ['e', 'b']]
5  :  [['a', 'f'], ['c', 'd'], ['e', 'b']]  ,  6  :  [['a', 'd'], ['c', 'f'], ['e', 'b']]
7  :  [['d', 'f'], ['a', 'b'], ['e', 'c']]  ,  8  :  [['b', 'f'], ['a', 'd'], ['e', 'c']]
9  :  [['b', 'd'], ['a', 'f'], ['e', 'c']]  ,  10  :  [['d', 'f'], ['c', 'b'], ['e', 'a']]
11  :  [['b', 'f'], ['c', 'd'], ['e', 'a']]  ,  12  :  [['b', 'd'], ['c', 'f'], ['e', 'a']]
13  :  [['b', 'd'], ['c', 'a'], ['e', 'f']]  ,  14  :  [['a', 'd'], ['c', 'b'], ['e', 'f']]
15  :  [['a', 'b'], ['c', 'd'], ['e', 'f']]  ,  16  :  [['c', 'a', 'b'], ['e', 'd', 'f']]
17  :  [['c', 'a', 'f'], ['e', 'b', 'd']]  ,  18  :  [['a', 'd', 'f'], ['e', 'b', 'c']]
19  :  [['c', 'a', 'd'], ['e', 'b', 'f']]  ,  20  :  [['a', 'b', 'f'], ['e', 'c', 'd']]
21  :  [['d', 'a', 'b'], ['e', 'c', 'f']]  ,  22  :  [['c', 'b', 'f'], ['e', 'a', 'd']]
23  :  [['c', 'd', 'f'], ['e', 'a', 'b']]  ,  24  :  [['d', 'b', 'f'], ['e', 'a', 'c']]
25  :  [['e', 'd', 'b'], ['e', 'a', 'f']]
```

```
il y a  105 répartitions possibles en prenant  7  élèves

voici les enumerations :
1  :  [['c', 'g', 'f'], ['d', 'b'], ['a', 'e']]  ,  2  :  [['g', 'b', 'f'], ['d', 'c'], ['a', 'e']]
3  :  [['c', 'b', 'f'], ['d', 'g'], ['a', 'e']]  ,  4  :  [['c', 'g', 'b'], ['d', 'f'], ['a', 'e']]
5  :  [['g', 'f'], ['d', 'b', 'c'], ['a', 'e']]  ,  6  :  [['c', 'f'], ['d', 'b', 'g'], ['a', 'e']]
7  :  [['c', 'g'], ['d', 'b', 'f'], ['a', 'e']]  ,  8  :  [['b', 'f'], ['d', 'c', 'g'], ['a', 'e']]
9  :  [['g', 'b'], ['d', 'c', 'f'], ['a', 'e']]  ,  10  :  [['c', 'b'], ['d', 'f', 'g'], ['a', 'e']]
11  :  [['c', 'g', 'f'], ['e', 'b'], ['a', 'd']]  ,  12  :  [['g', 'b', 'f'], ['e', 'c'], ['a', 'd']]
13  :  [['c', 'b', 'f'], ['e', 'g'], ['a', 'd']]  ,  14  :  [['c', 'g', 'b'], ['e', 'f'], ['a', 'd']]
15  :  [['g', 'f'], ['e', 'b', 'c'], ['a', 'd']]  ,  16  :  [['c', 'f'], ['e', 'b', 'g'], ['a', 'd']]
17  :  [['c', 'g'], ['e', 'b', 'f'], ['a', 'd']]  ,  18  :  [['b', 'f'], ['e', 'c', 'g'], ['a', 'd']]
19  :  [['g', 'b'], ['e', 'c', 'f'], ['a', 'd']]  ,  20  :  [['c', 'b'], ['e', 'f', 'g'], ['a', 'd']]
21  :  [['c', 'g', 'f'], ['e', 'd'], ['a', 'b']]  ,  22  :  [['g', 'd', 'f'], ['e', 'c'], ['a', 'b']]
23  :  [['c', 'd', 'f'], ['e', 'g'], ['a', 'b']]  ,  24  :  [['c', 'g', 'd'], ['e', 'f'], ['a', 'b']]
25  :  [['c', 'f'], ['e', 'd', 'g'], ['a', 'b']]  ,  26  :  [['c', 'g'], ['e', 'd', 'f'], ['a', 'b']]
27  :  [['g', 'f'], ['e', 'c', 'd'], ['a', 'b']]  ,  28  :  [['d', 'f'], ['e', 'c', 'g'], ['a', 'b']]
29  :  [['g', 'd'], ['e', 'c', 'f'], ['a', 'b']]  ,  30  :  [['c', 'd'], ['e', 'f', 'g'], ['a', 'b']]
31  :  [['g', 'b', 'f'], ['e', 'd'], ['a', 'c']]  ,  32  :  [['g', 'd', 'f'], ['e', 'b'], ['a', 'c']]
33  :  [['d', 'b', 'f'], ['e', 'g'], ['a', 'c']]  ,  34  :  [['d', 'g', 'b'], ['e', 'f'], ['a', 'c']]
35  :  [['b', 'f'], ['e', 'd', 'g'], ['a', 'c']]  ,  36  :  [['g', 'b'], ['e', 'd', 'f'], ['a', 'c']]
37  :  [['g', 'f'], ['e', 'b', 'd'], ['a', 'c']]  ,  38  :  [['d', 'f'], ['e', 'b', 'g'], ['a', 'c']]
39  :  [['g', 'd'], ['e', 'b', 'f'], ['a', 'c']]  ,  40  :  [['b', 'd'], ['e', 'f', 'g'], ['a', 'c']]
41  :  [['c', 'b', 'f'], ['e', 'd'], ['a', 'g']]  ,  42  :  [['c', 'd', 'f'], ['e', 'b'], ['a', 'g']]
43  :  [['d', 'b', 'f'], ['e', 'c'], ['a', 'g']]  ,  44  :  [['d', 'c', 'b'], ['e', 'f'], ['a', 'g']]
45  :  [['c', 'b'], ['e', 'd', 'f'], ['a', 'g']]  ,  46  :  [['c', 'f'], ['e', 'b', 'd'], ['a', 'g']]
47  :  [['d', 'f'], ['e', 'b', 'c'], ['a', 'g']]  ,  48  :  [['c', 'd'], ['e', 'b', 'f'], ['a', 'g']]
49  :  [['b', 'f'], ['e', 'c', 'd'], ['a', 'g']]  ,  50  :  [['b', 'd'], ['e', 'c', 'f'], ['a', 'g']]
51  :  [['c', 'g', 'b'], ['e', 'd'], ['a', 'f']]  ,  52  :  [['c', 'g', 'd'], ['e', 'b'], ['a', 'f']]
53  :  [['d', 'g', 'b'], ['e', 'c'], ['a', 'f']]  ,  54  :  [['d', 'c', 'b'], ['e', 'g'], ['a', 'f']]
55  :  [['c', 'b'], ['e', 'd', 'g'], ['a', 'f']]  ,  56  :  [['c', 'g'], ['e', 'b', 'd'], ['a', 'f']]
57  :  [['g', 'd'], ['e', 'b', 'c'], ['a', 'f']]  ,  58  :  [['c', 'd'], ['e', 'b', 'g'], ['a', 'f']]
59  :  [['g', 'b'], ['e', 'c', 'd'], ['a', 'f']]  ,  60  :  [['b', 'd'], ['e', 'c', 'g'], ['a', 'f']]
61  :  [['d', 'f'], ['c', 'b'], ['a', 'e', 'g']]  ,  62  :  [['b', 'f'], ['c', 'd'], ['a', 'e', 'g']]
63  :  [['b', 'd'], ['c', 'f'], ['a', 'e', 'g']]  ,  64  :  [['g', 'd'], ['c', 'b'], ['a', 'e', 'f']]
65  :  [['b', 'd'], ['c', 'g'], ['a', 'e', 'f']]  ,  66  :  [['g', 'b'], ['c', 'd'], ['a', 'e', 'f']]
67  :  [['b', 'f'], ['c', 'g'], ['a', 'd', 'e']]  ,  68  :  [['g', 'f'], ['c', 'b'], ['a', 'd', 'e']]
69  :  [['g', 'b'], ['c', 'f'], ['a', 'd', 'e']]  ,  70  :  [['e', 'f'], ['c', 'b'], ['a', 'd', 'g']]
71  :  [['b', 'f'], ['c', 'e'], ['a', 'd', 'g']]  ,  72  :  [['b', 'e'], ['c', 'f'], ['a', 'd', 'g']]
73  :  [['g', 'e'], ['c', 'b'], ['a', 'd', 'f']]  ,  74  :  [['b', 'e'], ['c', 'g'], ['a', 'd', 'f']]
75  :  [['g', 'b'], ['c', 'e'], ['a', 'd', 'f']]  ,  76  :  [['d', 'f'], ['c', 'g'], ['a', 'b', 'e']]
77  :  [['g', 'f'], ['c', 'd'], ['a', 'b', 'e']]  ,  78  :  [['g', 'd'], ['c', 'f'], ['a', 'b', 'e']]
79  :  [['e', 'f'], ['c', 'g'], ['a', 'b', 'd']]  ,  80  :  [['g', 'f'], ['c', 'e'], ['a', 'b', 'd']]
81  :  [['g', 'e'], ['c', 'f'], ['a', 'b', 'd']]  ,  82  :  [['e', 'f'], ['d', 'g'], ['a', 'b', 'c']]
83  :  [['g', 'f'], ['d', 'e'], ['a', 'b', 'c']]  ,  84  :  [['g', 'e'], ['d', 'f'], ['a', 'b', 'c']]
85  :  [['e', 'f'], ['d', 'c'], ['a', 'b', 'g']]  ,  86  :  [['c', 'f'], ['d', 'e'], ['a', 'b', 'g']]
87  :  [['c', 'e'], ['d', 'f'], ['a', 'b', 'g']]  ,  88  :  [['g', 'e'], ['d', 'c'], ['a', 'b', 'f']]
89  :  [['c', 'e'], ['d', 'g'], ['a', 'b', 'f']]  ,  90  :  [['c', 'g'], ['d', 'e'], ['a', 'b', 'f']]
91  :  [['d', 'f'], ['b', 'g'], ['a', 'c', 'e']]  ,  92  :  [['g', 'f'], ['b', 'd'], ['a', 'c', 'e']]
93  :  [['g', 'd'], ['b', 'f'], ['a', 'c', 'e']]  ,  94  :  [['e', 'f'], ['b', 'g'], ['a', 'c', 'd']]
95  :  [['g', 'f'], ['b', 'e'], ['a', 'c', 'd']]  ,  96  :  [['g', 'e'], ['b', 'f'], ['a', 'c', 'd']]
97  :  [['e', 'f'], ['d', 'b'], ['a', 'c', 'g']]  ,  98  :  [['b', 'f'], ['d', 'e'], ['a', 'c', 'g']]
99  :  [['b', 'e'], ['d', 'f'], ['a', 'c', 'g']]  ,  100  :  [['g', 'e'], ['d', 'b'], ['a', 'c', 'f']]
101  :  [['b', 'e'], ['d', 'g'], ['a', 'c', 'f']]  ,  102  :  [['g', 'b'], ['d', 'e'], ['a', 'c', 'f']]
103  :  [['b', 'e'], ['d', 'c'], ['a', 'f', 'g']]  ,  104  :  [['c', 'e'], ['d', 'b'], ['a', 'f', 'g']]
105  :  [['c', 'b'], ['d', 'e'], ['a', 'f', 'g']]
```

1.2.2 nombre enumerations en fonction du nombre d'élèves

```
il y a  385  répartitions possibles en prenant  8  élèves
il y a  1540  répartitions possibles en prenant  9  élèves
il y a  7245  répartitions possibles en prenant  10  élèves
il y a  32725  répartitions possibles en prenant  11  élèves
```

2 Méthode

2.1 Algorithme enumeration

2.1.1 Description

Dans cet algorithme, nous donnons tous les cas posssibles de groupe de 2 et de 3 que nous pouvons former avec un nombre d'élèves donné. Pour chacun de ses cas, nous allons donner toute les énumérations possibles grace à la fonction listerEnumeration située plus bas dans ce document.

2.1.2 Algo

Data: liste d'élèves
Result: énumérations, sous la forme d'une matrice
if *Si le nombre d'éléve est inférieur à 2* **then**
  Retourner un tableau vide;
else
  On calcule le nombre maximum de groupe possibles en fonction de notre liste d'élèves
  On calcule le nombre minimum de groupe possibles en fonction de notre liste d'élèves
  On prend comme formation initiale celle avec le maximum de binomes (max binomes, min trinomes)
  On retient le nombre de groupe de cette répartion
  On initialise un tableau pour stocker nos énumérations
  while $nombreDeGroupe \geq nombreDeGroupeMinimum$ **do**
    On ajoute au tableau des énumérations l'énumération calculer par la fonction lister_enumerations(liste d'élèves, répartition))
    On passe à la prochaine répartition (-3 Binomes + 2 Trinomes)
    On modifie le nombre de groupe suite à la nouvelle répartition
  end
end
On retourne le tableau des énumérations

Algorithm 1: enumeration

2.2 Algorithme lister_enumeration

2.2.1 Description

On va prendre les 2 premiers élèves de la liste des élèves qu'on va mémoriser et enlever de la liste des éleves. Puis on va relancer récursivement la fonction sur le reste des élèves. On ajoutera ensuite ses 2 élèves au resultat de la récursivité. Ce résultat donnera une énumération possible. On recommence ensuite en changeant le 2ème éléve enregistré dans la 1ere étape.

2.2.2 Algo

Data: liste d'éleves et une formation[binome,trinome]
Result: liste d'enumérations
On copie la liste d'élèves;
if *possibilite de faire qu'un binome* **then**
  Retourner les 2 élèves de la liste;
else
  if *possibilite de faire qu'un trinome* **then**
    Retourner les 3 élèves de la liste;
  else
end
On enleve un eleve eleve1 de la liste d'élèves
enumerations = [];
if *il y a des binomes dans la formations* **then**
  for *eleve2 in listeEleves* **do**
    liste2 = liste des élèves sans l'eleve1 et l'eleve2;
    list_enum = fonction récursive : on liste les autres énumérations avec un binome en moins dans la formation et l'eleve 1 et 2 en moins dans la liste d'élèves.
    for *i in taille list_enum* **do**
      on ajoute les 2 élèves mémorisés
    end
    On ajoute list_enum au tableau énumérations
  end
end
if *il y a des trinomes dans la formation* **then**
  for *eleve2 in liste2* **do**
    liste2 = liste des élèves sans l'eleve1 et l'eleve2;
    for *eleve3 in liste2* **do**
      if *eleve2 dejà traité* **then**
        liste3 = liste des élèves sans eleve1, eleve2 et eleve3 list_Enum = fonction récursive : on liste les autres énumérations avec un binome en moins dans la formation et l'eleve 1, 2 et 3 en moins dans la liste d'élèves.
        for *i in list_enum* **do**
          On ajoute les 3 eleves à la list_enum
        end
        On ajoute list_enum au tableau enumerations
      end
    end
  end
end
On retourne enumerations

Algorithm 2: listerEnumération