ORGANISATION FESUP 2026

Répartition des élèves en fonction de leurs voeux
&
Répartition des présentations dans les salles

doc de travail du 14 janvier 2026

OBJECTIF DE LA RÉPARTITION:

Chaque demi-journée, environ 1000 élèves du secondaire viendront assister à 4 présentations (sous forme de conférences, de tables rondes ou de flashs métiers).

Ils ont formulé 5 voeux chacun, leurs 2 premiers voeux seront systématiquement respectés. Une présentation peut être répétée, si besoin.

D’un point de vue logistique, il faut répartir les étudiants dans les salles de l’école de sorte qu’ils puissent suivre des présentations en accord avec leurs voeux, tout en respectant la capacité d’accueil de chaque salle.

On résout le problème indépendamment pour chacune des demi-journées.

LES ENTREES DU PROBLÈME:

- Pour chaque élève, on connaît ses 5 voeux (et sa vague d'arrivée: première vague pour les lycées de proximité ou deuxième vague pour ceux qui ont de la route):

Entrée pour les élèves:

	voeu 1	voeu 2	voeu 3	voeu 4	voeu 5	vague
élève 1	prés. 5	prés. 3	prés. 6	prés. 12	prés. 21	2
élève 2	prés. 11	prés. 28	prés. 5	prés. 12	prés. 21	1
élève 3	prés. 11	prés. 5	prés. 8	prés. 12	prés. 30	1
⋮	⋮	⋮	⋮			⋮
élève E	prés. 26	prés. 5	prés. 11	prés. 6	prés. 12	2

- Pour chaque salle utilisée pour l'évènement, on connaît sa capacité:

Entrée pour les salles:

salle	capacité
3020	111
3021	212
3022	212
⋮	⋮
A116	48

LES SORTIES DU PROBLÈME:

On doit produire:

- Pour les intervenants: les créneaux et salles dans lesquels chaque présentation est donnée

Résultat pour les intervenants :

	temps 1	temps 2	temps 3	temps 4	temps 5
présentation 1	J110	X	J107	X	X
présentation 2	X	J022	J022	J022	X
⋮					
présentation P	J108	J109	J109	X	X

- Pour les élèves: les 4 présentations qu'ils vont suivre ainsi que les salles dans lesquelles les présentations seront données

Résultat pour les élèves :

	temps 1	temps 2	temps 3	temps 4	temps 5
élève 1	X	prés. 5	prés. 12	prés. 3	prés 21
élève 2	prés. 5	prés. 28	prés. 12	prés. 11	X
élève 3	prés. 12	prés. 5	prés. 8	prés. 11	X
⋮	⋮	⋮	⋮	⋮	⋮
élève E	X	prés. 5	prés. 6	prés. 11	prés. 26

TRADUCTION EN UN PROBLÈME D'OPTIMISATION

Un élève est noté e (au total, il y a E élèves par demi-journée).

Une présentation est notée p (au total, il y a P présentations: 19 conférences + 6 tables rondes + 6 flash-métier)

Chaque demi-journée est décomposée en 4 temps (permettant donc à un élève de suivre 4 présentations différentes). Un temps est noté t, le nombre total de temps est $T = 4$.

Nous cherchons à déterminer à la fois:

- **l'affectation des élèves** a_e, avec l'affectation $a_e(e, p, t)$ une fonction binaire indiquant que l'élève e suit la présentation p au temps t si $a_e(e, p, t) = 1$, et qu'il ne suit pas cette présentation si $a_e(e, p, t) = 0$;

- **l'affectation des présentations aux différentes salles** a_s, avec l'affectation $a_s(p, s, t)$ une fonction binaire indiquant que la présentation p est donnée dans la salle s au temps t si $a_s(p, s, t) = 1$, ou qu'elle n'est pas donnée si $a_s(p, s, t) = 0$.

Les inconnues du problème correspondent donc à (a_e, a_s).

ILLUSTRATION:

- **l'affectation des élèves** a_e, avec l'affectation $a_e(e, p, t)$ une fonction binaire indiquant que l'élève e suit la présentation p au temps t si $a_e(e, p, t) = 1$, et qu'il ne suit pas cette présentation si $a_e(e, p, t) = 0$;

Affectation des étudiants : $a_e(e, p, t)$

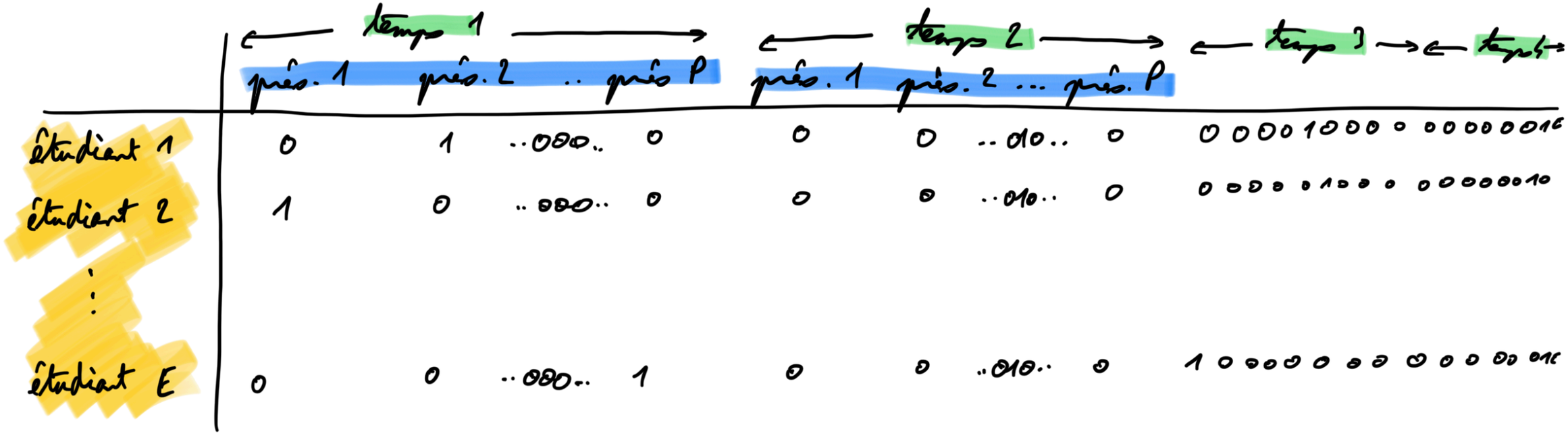
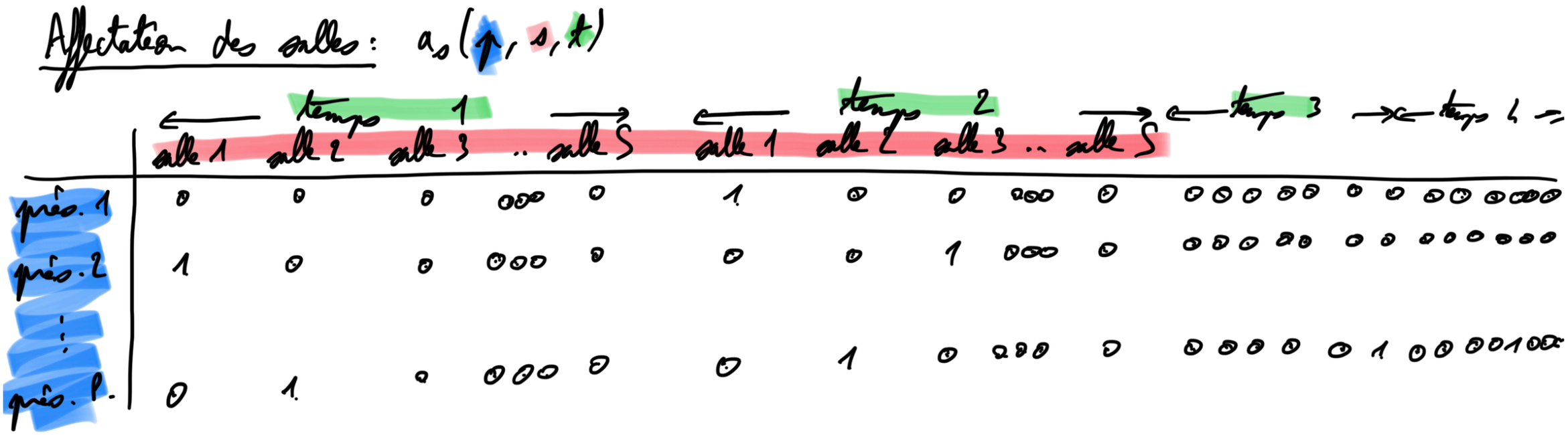


ILLUSTRATION:

- **l'affectation des présentations aux différentes salles** a_s, avec l'affectation $a_s(p, s, t)$ une fonction binaire indiquant que la présentation p est donnée dans la salle s au temps t si $a_s(p, s, t) = 1$, ou qu'elle n'est pas donnée si $a_s(p, s, t) = 0$.



TRADUCTION EN UN PROBLÈME D'OPTIMISATION

Afin d'affecter au mieux les élèves et les salles aux différentes présentations, nous pouvons définir une fonction objectif qui récompense une affectation qui tient compte des préférences des élèves (les élèves assistent aux présentations associées à leurs quatre premiers voeux) et une affectation qui minimise le nombre de présentations à réaliser (pour éviter, autant que possible, de se répéter).

Les élèves ont formulé des voeux représentés par la fonction c: la présentation p est le premier choix de l'élève e lorsque $c(e,1) = p$, la présentation p est le deuxième choix de l'élève e lorsque $c(e,2) = p$, et ainsi de suite.

On peut considérer la fonction objectif suivante qui est d'autant plus faible qu'on a affecté les élèves à des présentations qui les intéressent:

$$f_{\text{obj},e}(a_e) = \sum_{e=1}^{E} \sum_{p=1}^{P} \sum_{t=1}^{T} a_e(e,p,t) \cdot [c(e,5) = p]$$

où la notation $[c(e,5) = p]$ représente la valeur 1 lorsque $c(e,5) = p$ et 0 sinon (on veut donc limiter le nombre d'étudiants pour qui le choix 5 est imposé).

Pour l'affectation des salles, on cherche à minimiser le nombre de présentations. Si on dispose de S salles, on peut définir:

$$f_{\text{obj},s}(a_s) = \frac{1}{P \cdot S \cdot T + 1} \sum_{p=1}^{P} \sum_{s=1}^{S} \sum_{t=1}^{T} a_s(p,s,t)$$

Avec cette définition, puisque a_s prend des valeurs binaires (0 ou 1), on a $0 \leq f_{\text{obj},s}(a_s) < 1$ et la priorité est donnée à l'affectation des élèves (une meilleure affectation, dans laquelle un seul étudiant voit son affectation améliorée, est plus récompensée que n'importe quelle amélioration de l'affectation des présentations dans les différentes salles; par contre, à affectation équivalente pour les étudiants, on retient la solution qui limite le nombre de présentations à donner en regroupant les étudiants dans des présentations qui ont lieu au même moment.

Précisons maintenant les contraintes du problème…

CONTRAINTES DU PROBLÈME D'OPTIMISATION

Les choix 1 et 2 des élèves doivent être respectés:

$$\forall e,\ \sum_{t=1}^{T} a_e(e, c(e,1), t) = 1 \text{ et } \sum_{t=1}^{T} a_e(e, c(e,2), t) = 1$$

(ils suivent la présentation correspondant à leurs deux premiers choix à au moins l'une des sessions)

Les présentations qu'ils suivent correspondent à leurs 5 voeux:

$$\forall e,\ \sum_{t=1}^{T} a_e(e, c(e,1), t) + \sum_{t=1}^{T} a_e(e, c(e,2), t) + \sum_{t=1}^{T} a_e(e, c(e,3), t) + \sum_{t=1}^{T} a_e(e, c(e,4), t) + \sum_{t=1}^{T} a_e(e, c(e,5), t) = T$$

Une présentation n'est pas suivie plus d'une fois par un même étudiant:

$$\forall e, \forall p,\ \sum_{t=1}^{T} a_e(e, p, t) \leq 1$$

Les affectations sont binaires:

$\forall e, \forall p, \forall t, 0 \leq a_e(e, p, t) \leq 1,\ \forall p, \forall s, \forall t, 0 \leq a_s(p, s, t) \leq 1$ et $a_e(e, p, t)$ comme $a_s(p, s, t)$ sont entiers

Un étudiant suit une présentation et une seule par session pendant laquelle il est présent (en fonction de sa vague d'arrivée):

$$\text{si vague} = 1, \forall e,\ \forall t,\ 1 \leq t \leq 4,\ \sum_{p=1}^{P} a_e(e, p, t) = 1 \text{ et } \forall e,\ \sum_{p=1}^{P} a_e(e, p, 5) = 0; \text{ si vague} = 2,$$

$$\forall e,\ \forall t,\ 2 \leq t \leq 5,\ \sum_{p=1}^{P} a_e(e, p, t) = 1 \text{ et } \forall e,\ \sum_{p=1}^{P} a_e(e, p, 1) = 0$$

Une seule présentation, au plus, est affectée à une salle donnée, pour chaque temps de présentation:

$$\forall s, \forall t,\ \sum_{p=1}^{P} a_s(p, s, t) \leq 1$$

La capacité des salles est respectée: le nombre d'étudiants suivant la présentation p dans la salle s n'excède pas la capacité de la salle s si la salle est bien affectée à cette présentation ou aucun étudiant ne peut suivre cette présentation dans cette salle sur ce créneau:

$$\forall p, \forall t,\ \sum_{e=1}^{E} a_e(e, p, t) \leq \sum_{s=1}^{S} \text{capacité}(s) \cdot a_s(p, s, t)$$

RESOLUTION DU PROBLÈME D'OPTIMISATION

Résolution du problème d'optimisation:

$$\arg\min_{(a_e, a_s)} \sum_{e=1}^{E} \sum_{p=1}^{P} \sum_{t=1}^{T} a_e(e,p,t) \cdot [c(e,5) = p] \; + \; \frac{1}{P \cdot S \cdot T + 1} \sum_{p=1}^{P} \sum_{s=1}^{S} \sum_{t=1}^{T} a_s(p,s,t)$$

sous les contraintes:

$$\forall e, \; \sum_{t=1}^{T} a_e(e, c(e,1), t) = 1 \text{ et } \sum_{t=1}^{T} a_e(e, c(e,2), t) = 1$$

$$\forall e, \; \sum_{t=1}^{T} a_e(e, c(e,1), t) + \sum_{t=1}^{T} a_e(e, c(e,2), t) + \sum_{t=1}^{T} a_e(e, c(e,3), t) + \sum_{t=1}^{T} a_e(e, c(e,4), t) + \sum_{t=1}^{T} a_e(e, c(e,5), t) = T$$

$$\forall e, \forall p, \; \sum_{t=1}^{T} a_e(e,p,t) \leq 1$$

$$\forall e, \forall p, \forall t, \; 0 \leq a_e(e,p,t) \leq 1, \; \forall p, \forall s, \forall t, \; 0 \leq a_s(p,s,t) \leq 1 \text{ et } a_e(e,p,t) \text{ comme } a_s(p,s,t) \text{ sont entiers}$$

$$\forall e, \forall t, \; \sum_{p=1}^{P} a_e(e,p,t) = 1$$

$$\forall s, \forall t, \; \sum_{p=1}^{P} a_s(p,s,t) \leq 1$$

$$\forall p, \forall t, \; \sum_{e=1}^{E} a_e(e,p,t) - \sum_{s=1}^{S} \text{capacité}(s) \cdot a_s(p,s,t) \leq 0$$

Il s'agit d'un programme linéaire entier (la fonction objectif est linéaire en a_e et a_s, les contraintes d'égalité et d'inégalité sont toutes linéaires).

La solution optimale peut être déterminée avec un algorithme déterministe (cutting plane, branch and bound), par exemple scipy.optimize.milp.

GUIDE D'UTILISATION DE LA FONCTION SCIPY.OPTIMIZE.MILP

documentation de la fonction:

https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.milp.html

la fonction est basée sur la bibliothèque open-source HiGHS:

https://highs.dev/

UTILISATION DE LA FONCTION MILP()

La fonction milp() permet de résoudre un problème du type:

$$\arg\min_{x} \sum_{i=1}^{N} c(i) \cdot x(i) \text{ c'est à dire } \arg\min_{\boldsymbol{x}} \boldsymbol{c}^t \boldsymbol{x}$$

sous les contraintes:

$$\forall i,\ \ell \leq x(i) \leq u$$

$$\text{les } x(i) \text{ sont tous entiers}$$

$$\forall i,\ b_l(i) \leq \sum_{j=1}^{n} A(i,j) \cdot x(j) \leq b_u(i), \text{ c'est à dire } \boldsymbol{b}_l \leq \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}_u$$

Pour cela, il faut fournir:

- en premier paramètre: le tableau c qui a la même dimension que nos inconnues (a_e, a_s), c'est à dire $N = EPT + PST$ (soit environ 150000 inconnues)

- préciser la contrainte que les inconnues sont entières: integrality=np.ones(N) (rq: on pourrait résoudre un problème dans lequel certaines inconnues sont entières et d'autres réelles)

- préciser les contraintes de bornes: bounds=(np.zeros(N),np.ones(N))

- préciser les contraintes d'inégalité: constraints=opt.LinearConstraint(A, b_l, b_u)

où opt.LinearConstraint() permet de définir les contraintes.

REMARQUES IMPORTANTES

1) Nous avons deux jeux d'inconnues: a_e et a_s, qui sont des tableaux tridimensionnels. Pour la fonction milp(), il y a une seule inconnue 1D. Il faut donc aplatir nos tableaux et les concaténer.

2) La matrice $\mathbf{A}$ a autant de lignes qu'il y a de contraintes et autant de colonnes que d'inconnues. La stocker et la manipuler serait trop coûteux pour notre problème. Nous allons donc devoir utiliser une représentation *creuse*: scipy.sparse.coo_array().

```
# Exemple de création d'un tableau creux:

row  = np.array([0, 3, 1, 0])

col  = np.array([0, 3, 1, 2])

data = np.array([4, 5, 7, 9])

A = coo_array((data, (row, col)), shape=(4, 4))

# le tableau A correspond alors à:

array([[4, 0, 9, 0],
       [0, 7, 0, 0],
       [0, 0, 0, 0],
       [0, 0, 0, 5]])
```

3) Le solver dispose de plusieurs options qui peuvent être utiles, notamment l'option 'disp' pour afficher l'état de l'optimisation au cours du calcul, 'presolve' qui peut éventuellement accélérer les choses, 'time_limit' pour définir un budget de temps.

4) Le résultat renvoyé est une structure contenant notamment un état ('status') qui renseigne sur le résultat (solution optimale, limite de temps/d'itérations atteinte, problème sans solution, …)

ETAPES POSSIBLES POUR LE TRAVAIL

1. Prise en main de la fonction scipy.optimize.milp()

    résoudre un problème simple, par ex:

    $\arg\min_{a} \; 5 \cdot a(1) - 3 \cdot a(2)$ tel que $0 \leq a(1) \leq 1$ et $0 \leq a(2) \leq 1$ et a entier et $a(1) + a(2) \leq 1$

2. Définition de la fonction de coût et des contraintes pour notre problème afin d'utiliser scipy.optimize.milp()

    (en s'inspirant de l'exemple donné)

3. Génération de données de test

    Partir d'un résultat (répartition aléatoire de présentations dans des salles, certaines salles n'étant pas utilisées, puis génération d'un programme de visite des élèves en respectant la capacité des salles), puis en déduire des voeux (il faut choisir un voeu en plus des 4 présentations suivies par les élèves).

4. Ecriture d'un code qui vérifie la faisabilité d'une solution (c'est à dire, le respect des différentes contraintes) et qui compare la solution obtenue à celle utilisée pour générer les données de test + benchmark du temps de calcul pour vérifier le passage à l'échelle.

4. Lecture des fichiers .csv produits par l'appli web et mise en forme pour appel de la fonction milp()

5. Mise en forme des résultats (tableau intervenants et tableau élèves)

ETAPES POSSIBLES POUR LE TRAVAIL

1. Prise en main de la fonction scipy.optimize.milp()

résoudre un problème simple, par ex:

$\arg\min_{a} \; 5 \cdot a(1) - 3 \cdot a(2)$ tel que $0 \leq a(1) \leq 1$ et $0 \leq a(2) \leq 1$ et a entier et $a(1) + a(2) \leq 1$

```
import scipy.optimize as opt
import scipy.sparse as sp
import numpy as np


c = np.array([5,-3])
n = c.size
A = sp.coo_array(([1,1],([0,0],[0,1])),shape=(1,2),dtype=float)
res =
opt.milp(c,integrality=np.ones(n),bounds=(np.zeros(n),np.ones(n)),constraints=opt.LinearConstraint(A,
-np.inf, 1))
print(res)
        message: Optimization terminated successfully. (HiGHS Status 7: Optimal)
        success: True
         status: 0
            fun: -3.0
              x: [ 0.000e+00  1.000e+00]
 mip_node_count: 0
 mip_dual_bound: -3.0
        mip_gap: 0.0
```