ANNEXE 1: Modèle linéaire

```
from pulp import LpProblem, LpVariable, lpSum, LpMinimize, LpBinary, value

# Données d'entrée (à adapter selon les besoins réels)
S = 5  # Nombre maximum de cours par créneau
G = 3  # Nombre de groupes (exemple : 3 groupes)
I = 4  # Nombre de matières (exemple : 4 matières)
K = 10  # Nombre de créneaux (exemple : 10 créneaux)
STP = 2  # Nombre maximum de TP par créneau

# Charge des matières pour chaque groupe (exemple)
rho_M = {0: {0: 2, 1: 3, 2: 1},  # Charge CM pour chaque matière et groupe
        1: {0: 2, 1: 1, 2: 2},
        2: {0: 1, 1: 2, 2: 1},
        3: {0: 3, 1: 2, 2: 2}}

rho_T = {0: {0: 1, 1: 2, 2: 1},  # Charge TP pour chaque matière et groupe
        1: {0: 1, 1: 1, 2: 1},
        2: {0: 1, 1: 1, 2: 1},
        3: {0: 2, 1: 1, 2: 1}}

rho_D = {0: {0: 1, 1: 1, 2: 1},  # Charge TD pour chaque matière et groupe
        1: {0: 1, 1: 1, 2: 1},
        2: {0: 1, 1: 1, 2: 1},
        3: {0: 1, 1: 1, 2: 1}}

# Disponibilité des enseignants (exemple)
D = {0: {k: 1 for k in range(K)},  # Enseignant 0 est disponible à tous les créneaux
    1: {k: 1 for k in range(K)},
    2: {k: 1 for k in range(K)},
    3: {k: 1 for k in range(K)}}

# Affectation des groupes et matières aux enseignants (exemple)
C_cm = {0: [(0, 0), (1, 0), (2, 0)],  # Enseignant 0 enseigne les CM des matières 0, 1, 2 pour
le groupe 0
        1: [(0, 1), (1, 1), (2, 1)],
        2: [(0, 2), (1, 2), (2, 2)],
        3: [(3, 0), (3, 1), (3, 2)]}

C_tp = {0: [(0, 0), (1, 0), (2, 0)],  # Enseignant 0 enseigne les TP des matières 0, 1, 2 pour le
groupe 0
        1: [(0, 1), (1, 1), (2, 1)],
        2: [(0, 2), (1, 2), (2, 2)],
        3: [(3, 0), (3, 1), (3, 2)]}
```

```
C_td = {0: [(0, 0), (1, 0), (2, 0)],  # Enseignant 0 enseigne les TD des matières 0, 1, 2 pour le groupe 0
        1: [(0, 1), (1, 1), (2, 1)],
        2: [(0, 2), (1, 2), (2, 2)],
        3: [(3, 0), (3, 1), (3, 2)]}

# Matrice des chevauchements de groupes (exemple)
A = [(0, 1), (1, 2)]  # Les groupes 0 et 1, ainsi que 1 et 2, se chevauchent

# Définir les variables de décision
X = LpVariable.dicts("X", ((g, j, k) for g in range(G) for j in range(I) for k in range(K)), cat=LpBinary)
Y = LpVariable.dicts("Y", ((g, j, k) for g in range(G) for j in range(I) for k in range(K)), cat=LpBinary)
Z = LpVariable.dicts("Z", ((g, j, k) for g in range(G) for j in range(I) for k in range(K)), cat=LpBinary)

# Définir le problème
prob = LpProblem("Emploi_du_temps", LpMinimize)

# Fonction objectif (minimiser le nombre de créneaux utilisés)
prob += lpSum(X[g, j, k] + Y[g, j, k] + Z[g, j, k] for g in range(G) for j in range(I) for k in range(K))

# Contraintes
# Contrainte A : Un groupe ne peut avoir qu'une seule séance dans un créneau
for k in range(K):
    for g in range(G):
        prob += lpSum(X[g, j, k] + Y[g, j, k] + Z[g, j, k] for j in range(I)) <= 1

# Contrainte B : La charge de chaque matière doit être complètement dispensée
for i in range(I):
    for g in range(G):
        prob += lpSum(X[g, i, k] for k in range(K)) == rho_M[i][g]  # CM
        prob += lpSum(Y[g, i, k] for k in range(K)) == rho_T[i][g]  # TP
        prob += lpSum(Z[g, i, k] for k in range(K)) == rho_D[i][g]  # TD

# Contrainte C : Un enseignant ne peut dispenser un cours que s'il est disponible
for i in range(I):
    for k in range(K):
        prob += lpSum(X[g, j, k] for (j, g) in C_cm[i]) + lpSum(Y[g, j, k] for (j, g) in C_tp[i]) + lpSum(Z[g, j, k] for (j, g) in C_td[i]) <= D[i][k]

# Contrainte D : Nombre maximum de cours par créneau
for k in range(K):
    prob += lpSum(X[g, j, k] + Y[g, j, k] + Z[g, j, k] for g in range(G) for j in range(I)) <= S

# Contrainte E : Nombre maximum de TP par créneau
```

```
for k in range(K):
    prob += lpSum(Y[g, j, k] for g in range(G) for j in range(I)) <= STP

# Contrainte F : Deux groupes qui se chevauchent ne peuvent pas avoir de séance dans le même créneau
for k in range(K):
    for (g1, g2) in A:
        prob += lpSum(X[g1, j, k] + X[g2, j, k] + Y[g1, j, k] + Y[g2, j, k] + Z[g1, j, k] + Z[g2, j, k] for j in range(I)) <= 1

# Résoudre le problème
prob.solve()

# Afficher les résultats
print("Statut de la solution:", prob.status)
for k in range(K):
    for g in range(G):
        for j in range(I):
            if value(X[g, j, k]) == 1:
                print(f"Groupe {g} a un CM de matière {j} dans le créneau {k}")
            if value(Y[g, j, k]) == 1:
                print(f"Groupe {g} a un TP de matière {j} dans le créneau {k}")
            if value(Z[g, j, k]) == 1:
                print(f"Groupe {g} a un TD de matière {j} dans le créneau {k}")
```