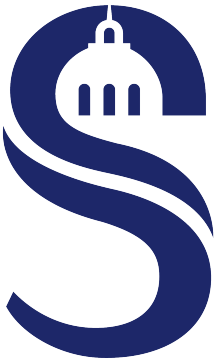


SORBONNE UNIVERSITÉ

Master Informatique

Rapport de projet
Gestionnaire de voeux

Etudiants :

Adan Bougherara

Vivien Demeulenaere

Encadrant :

Antoine Genitrini

Remis le 13 mai 2022

Table des matières

Gestionnaire de voeux

Résumé : Notre projet consiste à réécrire l'application logicielle de gestion des voeux d'UE des étudiants. Il y a quelques années, une application de gestion des voeux des étudiants a été développée par des étudiants du parcours ANDROIDE du master informatique de Sorbonne Université. Le but étant d'informatiser et par conséquent d'accélérer le processus de choix des UE tout en garantissant de respecter des contraintes liées à l'emploi du temps par exemple. Le coeur de l'application se base sur un solveur de contraintes qui permet de résoudre des modèles mathématiques. Notre projet comprend de revoir le développement de l'application qui a été enrichie au fil des années et dont le fil conducteur et la simplicité de l'architecture sont à revoir ainsi que d'actualiser le modèle mathématique construit pour que le solveur de contrainte trouve une solution optimale.

Mots-clés : Gestionnaire de voeux, Gurobi, Modélisation de problème, Solveur de contraintes

1 Introduction

Ce projet vise à revoir l'architecture et la conception d'un logiciel de gestion de voeux d'UE d'étudiants. Ce dernier avait été développé par des étudiants dans le cadre d'un projet du parcours ANDROIDE du master informatique de Sorbonne Université.
Il s'agira à la fois de revoir le modèle mathématique anciennement utilisé et d'améliorer l'expérience utilisateur avec notamment une interface graphique et une remontée plus explicite des erreurs.

2 Modélisation du problème

Notre problème se résume à optimiser une fonction possédant des variables de degré 1 avec différentes contraintes que doivent respecter ces variables.
Il s'agit donc d'un problème de type linéaire.
Ainsi, l'application doit contenir une partie dédiée à la modélisation du problème. Nous allons dans un premier temps décrire dans cette partie le modèle mathématique anciennement utilisé. Dans un second temps nous présenterons les modifications apportées à cet ancien modèle pour créer le nouveau.

2.1 Ancien modèle

2.1.1 Les variables de décisions

L'ancien modèle était composé de trois variables de décisions binaires :

$$x_{ijk} = \begin{cases} 1 & \text{si l'étudiant i est inscrit dans le groupe k de l'UE j} \\ 0 & \text{sinon} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{si l'étudiant i est inscrit dans l'UE j} \\ 0 & \text{sinon} \end{cases}$$

$$n_i = \begin{cases} 1 & \text{si l'étudiant i est totalement satisfait} \\ 0 & \text{sinon} \end{cases}$$

2.1.2 La fonction objectif

La fonction objectif consistait à maximiser la somme des variables y_{ij}. Il s'agissait donc de maximiser le nombre d'étudiants inscrits.

$$max\ z = \sum y_{ij}$$

2.1.3 Les contraintes

Le modèle était composé de 6 contraintes linéaires permettant de modéliser les différentes exigences auquel le résultat doit répondre.

— Chaque étudiant doit être inscrit dans les unités d'enseignement obligatoires dans son parcours :

$$\forall i \in Etudiant,\ \forall j \in UE_{i\_obligatoire},\ y_{ij} = 1$$

— Un étudiant ne peut pas être inscrit dans plusieurs groupes d'une même matière :

$$\forall i \in Etudiant, \forall j \in UE_i, \\ (\sum_{k \in Gr_j} x_{ijk}) - y_{ij} = 0$$

— Un groupe ne peut contenir plus d'étudiants qu'il n'a de places :

$$\forall j \in UE, \forall k \in Gr_j, \\ (\sum_{i \in Etudiant} x_{ijk}) \leq cap_{jk}$$

— Un étudiant ne peut pas être inscrit dans deux unités d'enseignement incompatibles :

$$\forall i \in Etudiant\ et\ toutes\ les\ incompatibilites\ ((UE_{j_1}, Gr_{k_1}), (UE_{j_2}, Gr_{k_2}))\ des\ UE\ de\ i, \\ x_{ij_1k_1} + x_{ij_2k_2} \leq 1$$

— Contrainte d'équilibre entre les groupes d'une unité d'enseignement :

$$\forall j \in UE, \forall k_1, k_2 \in Gr_j, \\ \left| \frac{\sum_{i \in Etudiant} x_{ijk_1}}{capGr_{k1}} - \frac{\sum_{i \in Etudiant} x_{ijk_2}}{capGr_{k2}} \right| \leq seuil$$

— Contrainte indiquant qu'un étudiant est satisfait s'il a une affectation complète :

$$\forall i \in Etudiant, v = taille(\{UE_i\}), \\ (\sum_{j \in E_i} y_{ij}) - v * n_i \geq 0$$

2.1.4 Utilisation du modèle dans l'application

Le solveur de contrainte utilisé pour résoudre le modèle était Gurobi. Le principe était de fournir au modèle, pour chaque étudiant, autant de voeux d'unités d'enseignement que de matières qu'il doit suivre. Initialement, on fournissait donc au modèle les premiers choix de chaque étudiant, puis on appelait le solveur. Si Gurobi ne trouvait pas de solution, c'est-à-dire si un certain nombre étudiants n'avaient pas d'affectations, il s'agissait alors de changer leurs voeux avec un algorithme d'unranking puis de rappeler le solveur avec ces nouvelles valeurs.

Par exemple, si un étudiant avait pour liste de voeux (PAF, CPA, CA, CPS, PC3R, APS) et qu'il devait suivre 4 unités d'enseignement, on donnait au modèle uniquement ses 4 premiers choix (PAF, CPA, CA, CPS). Si après l'appel à Gurobi son affectation n'était pas complète, l'algorithme d'unranking modifiait sa liste de voeux en (PAF, CPA, CA, PC3R). Puis si au prochain appel l'étudiant n'a toujours pas 4 matières affectées, on change encore sa liste de voeux, etc...

Enfin, s'il restait toujours des étudiants sans affectations après avoir sondé le solveur avec toutes leurs combinaisons de choix possibles, c'était l'affectation d'étudiants ayant réussi à avoir un contrat complet qui était choisie avec une certaine probabilité, jusqu'à ce que le solveur trouve une solution telle que tous les étudiants aient un contrat complet :
$z = (nombre\ d'étudiants) * (le\ nombre\ de\ matieres\ qu'ils\ doivent\ suivre)$
Ainsi, l'application effectuait plusieurs appels au solveur de contrainte Gurobi.

2.2 Nouveau modèle

Nous avons fait le choix de conserver l'usage du solveur Gurobi. Cependant, nous avons modifié le modèle afin de ne faire qu'un appel au solveur. En réalité, nous verrons par la suite qu'on effectue une deuxième passe avec une contrainte relâchée afin de marquer les contrats impossibles (cf. 3.4).
Pour ce faire, nous renseignons au modèle tous les choix d'unités d'enseignement de chaque étudiant. Par conséquent, nous avons dû ajouter des contraintes pour garantir que chaque étudiant doit avoir le bon nombre d'unités d'enseignement comme l'exige son parcours.
De plus, nous avons fait le choix de fournir à Gurobi un modèle plus complet en y ajoutant également les contraintes liées à l'emploi du temps, contrairement à l'ancien modèle qui recevait les couples des

UE incompatibles, qui étaient calculées par l'application au préalable, sans l'aide du solveur.

Nous avons conservé les variables de décisions x_{ijk} et y_{ij} de l'ancien modèle. La fonction objectif a été changée. Notre fonction objectif maximise la satisfaction globale des étudiants. Par conséquent, l'ancienne fonction objectif qui visait à maximiser les affectations complètes a été transformée en contraintes : chaque étudiant doit avoir une affectation complète. La fonction `creation_modele` s'occupe de créer le modèle. Elle reçoit en entrée le dictionnaire comprenant les UE, la liste des etudiants, ainsi que deux booléens *contrainte_relachee_affectation* et *contrainte_relachee_capacite*.
Le premier donne un pouvoir de décision sur l'ajout de la contrainte indiquant que chaque étudiant doit avoir un contrat complet. Autrement dit, positionnée à `True`, elle indique au modèle qu'on est prêt à accepter une solution partielle où tous les étudiants n'ont pas forcément d'affectation complète.
Le second donne un pouvoir de décision sur l'ajout de la contrainte liée à la capacité des groupes. Elle permet de cibler les affectations incomplètes dues à des incompatibilités liées à l'emploi du temps. Elle crée, ajoute les variables de décisions au modèle et les stocke dans des dictionnaires. Puis, elle ajoute la fonction objectif et les contraintes au modèle.

```
def creation_modele(dictionnaire_ue, liste_etudiants,
                    contrainte_relachee_affectation=False,
                    contrainte_relachee_capacite=False):
    modele = gp.Model("Affectation")
    """On ajoute les variables au modèle"""
    vars_x, vars_y, vars_x_par_ue, vars_creneaux_incompatibles,
    vars_cours_par_ue, vars_td_tme_par_ue = ajouter_vars(
                                        modele, dictionnaire_ue, liste_etudiants)
    """Puis on ajoute la fonction objectif"""
    ajouter_fonction_objectif(modele, vars_y)
    """Ainsi que les contraintes"""
    ajouter_contrainte_ue_obligatoires(modele, vars_y)
    if not contrainte_relachee_capacite:
        ajouter_contrainte_capacite_groupes(modele, vars_x_par_ue)
    ajouter_contrainte_groupe_unique(modele, vars_x_par_ue)
    ajouter_contrainte_nombre_ue_etudiant(modele, vars_y,
    contrainte_relachee_affectation)
    ajouter_contrainte_edt(modele, vars_creneaux_incompatibles)
    ajouter_contrainte_inscription_complete(modele, vars_cours_par_ue,
    vars_td_tme_par_ue, vars_y)
    return (modele, vars_x, vars_y, vars_x_par_ue)
```

2.2.1 Les variables de décisions

Définition formelle

$$x_{ijk} = \begin{cases} 1 & \text{si l'étudiant } i \text{ est inscrit dans le groupe/cours } k \text{ de l'UE } j \\ 0 & \text{sinon} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{si l'étudiant } i \text{ est inscrit dans l'UE } j \\ 0 & \text{sinon} \end{cases}$$

2.2.2 La fonction objectif

Définition formelle

$$max\ z\ =\ \sum_{i\ \in\ Etudiant}\ \sum_{j\ \in\ UE_i}\ (y_{ij}\ *\ \alpha_j)$$

Avec α le coefficient pour prioriser les voeux des étudiants dans le modèle.

code

```
def ajouter_fonction_objectif(modele, vars_y):
    expr = gp.LinExpr()
    for etu, variables in vars_y.items():
        cpt = 1
        coeff = 10000
        for var in variables:
            expr.add(coeff*var)
            coeff -= (cpt * (etu.nb_ue_a_total - cpt))
            cpt += 1
    modele.setObjective(expr, GRB.MAXIMIZE)
    modele.update()
```

2.2.3 Les contraintes

— Chaque étudiant doit être inscrit dans les unités d'enseignement obligatoires dans son parcours :

Définition formelle

$$\forall i\ \in\ Etudiant,\ \forall j\ \in\ UE_{obligatoire},\ y_{ij} = 1$$

code (cf. section 6.1.1 de l'annexe, page 15)

— Un étudiant ne peut pas être inscrit dans plusieurs groupes d'une même matière :

Définition formelle

$$\forall i\ \in\ Etudiant,\ \forall j\ \in\ UE,\ (\sum_{k\ \in\ Gr_j}\ x_{ijk})\ \leq\ 1$$

code (cf. section 6.1.2 de l'annexe, page 15)

— Un groupe ne peut contenir plus d'étudiants qu'il n'a de places :

Définition formelle

$$\forall j\ \in\ UE,\ \forall k\ \in\ Gr_j,\ (\sum_{i\ \in\ Etudiant}\ x_{ijk})\ \leq\ cap_{jk}$$

code (cf. section 6.1.3 de l'annexe, page 15)

— Un étudiant doit suivre le bon nombre d'UE :

Définition formelle

$$\forall i\ \in\ Etudiant,\ (\sum_{j\ \in\ UE}\ y_{ij})\ =\ \alpha_i$$

Avec α_i le nombre de matières que l'étudiant i doit suivre.

code (cf. section 6.1.4 de l'annexe, page 15)

— Un étudiant ne peut pas être inscrit dans deux groupes (d'UE différentes) ayant un créneau en commun :

Définition formelle

Soient $g1$ un groupe d'une UE notée $u1$ et $g2$ un groupe d'une UE notée $u2$ étant sur un même créneau (avec $u1\ \neq\ u2$),

$$\forall i\ \in\ Etudiant,\ x_{iu1g1}\ +\ x_{iu2g2}\ \leq\ 1$$

code (cf. section 6.1.5 de l'annexe, page 16)

— Un étudiant inscrit dans un groupe d'une UE doit être inscrit dans un cours de cette UE :

Définition formelle

$$\forall i \in Etudiant,\ \forall j \in UE,\ (\sum_{k \in Gr_j} x_{ijk}) = (\sum_{c \in Cours_j} x_{ijc})$$

code (cf. section 6.1.6 de l'annexe, page 16)

3 Fonctionnement de l'application

Cette section vise à décrire le fonctionnement de l'application. Cette dernière utilise une interface graphique basée sur le module `Tkinter` de python.

3.1 Données d'entrée du problème

L'application nécessite deux fichiers d'entrée. L'un d'entre eux est dédié aux voeux des étudiants et l'autre aux créneaux des cours. Ils doivent tous deux être chargés via le sous menu `Fichier` de l'application. Une description plus détaillée de leur contenu est disponible plus bas.
Il est à noter que le nom des colonnes des différents fichiers d'entrée peuvent être modifiés dans le fichier `parametres.py` de l'application. De plus, l'ordre des colonnes n'importe pas. Enfin ces fichiers doivent adopter un format proche de celui d'un *csv* à ceci près que le séparateur peut être une virgule, un point virgule ou tout autre caractère ajouté à la variable `delimiteurs_fichier` du fichier `parametres.py`.

3.1.1 Fichier de voeux

Le fichier de voeux doit impérativement contenir une colonne pour le parcours, une pour le numéro d'étudiant, i colonnes pour les voeux obligatoires, j colonnes pour les voeux que l'étudiant souhaite en priorité et k voeux de substitution. Il n'y a pas de contraintes sur i, j, k si ce n'est que ces nombres doivent être au moins égaux à un. De plus, si l'on opte pour plusieurs voeux d'un de ces trois types, il sera nécessaire de nommer les colonnes de la forme suivante : *colonneN* avec *colonne* le nom de la colonne et N le numéro de cette dernière.

Pour que l'application puisse fonctionner correctement, on veillera à nommer ces colonnes sans discontinuité dans la numérotation. Par exemple pour les UE obligatoires, les colonnes *oblig*1, *oblig*2 et *oblig*3 peuvent convenir mais si l'on remplace *oblig*3 par *oblig*4, cette dernière colonne sera ignorée.

Une fois les entêtes des colonnes définis, il est maintenant possible de les remplir. Chacune des lignes du fichier représente les voeux d'un étudiant. Il est donc impératif que les colonnes contenant l'intitulé de leur parcours ainsi que leur numéro d'étudiant soient renseignées. Celui-ci devra être unique au sein d'un même parcours.

Ensuite, les colonnes de voeux peuvent être remplies par des intitulés d'UE dont le nom est spécifié dans le fichier dédié aux créneaux horaire. Cependant, leur remplissage n'est pas obligatoire et peut être adapté en fonction du nombre de voeux de chacun. La seule contrainte sur ce point est que la somme des UE obligatoires et des UE souhaitées doit être exactement égal au nombre d'UE que l'étudiant doit suivre. Il est néanmoins conseillé de fournir un certain nombre de voeux de substitution afin d'éviter des affectations partielles en raison de groupes saturés ou de créneaux horaire incompatibles.

3.1.2 Fichier des créneaux de cours

Le fichier des créneaux de cours doit contenir des colonnes pour renseigner le numéro de l'UE, son nom, ainsi que son numéro de groupe. De plus, il est nécessaire d'avoir une colonne par nombre de

groupe afin d'en renseigner sa capacité. Enfin, pour chacun de ces groupes on prendra soin d'avoir une colonne de cours, une de TD, et une de TME. Ces colonnes seront remplies d'un créneau horaire représenté par un entier. A l'instar du fichier de voeux, la numérotation des colonnes doit être réalisée sans discontinuité.

On pourra ainsi représenter chacune des UE par une ligne dans le fichier. Quelques critères sur ces dernières s'appliquent comme l'unicité de leur identifiant, de leur nom ou encore l'interdiction du caractère _ ou des espaces dans leur nom.

3.2 Traitement des entrées

Lorsque les fichiers d'entrée sont fournis à l'application, leur chemin est stocké dans les paramètres de cette dernière. Aucune analyse sur le format des fichiers n'est alors encore effectué. S'ils posent problèmes, des erreurs pourront éventuellement être déclenchées au moment du calcul des affectations. A l'inverse, si les fichiers conviennent, leur traitement pourra commencer.

3.2.1 Cas d'erreur

Le cahier des charges de ce projet exige une gestion des erreurs fine et transparente afin de palier celle de l'ancien modèle qui était assez brutale et qui laissait souvent l'utilisateur dans le flou.
Nous avons corrigé cela en ciblant les erreurs qui peuvent être déclenchées lors de l'utilisation de l'application et en affichant un message explicite selon l'erreur.
Lors du traitement des fichiers, deux erreurs inhérentes au mauvais format des fichiers peuvent être levées. Il s'agit de `ColonneInexistante` et `FormatDeFichierNonReconnu`, toutes deux définies dans le fichier `erreurs.py`.
La première se déclenche lorsque l'application souhaite accéder à une colonne dont l'entête est inexistant. La deuxième quant à elle peut se déclencher si l'on a essayé de lire le fichier avec tous les séparateurs présents dans la variable `delimiteurs_fichier` du fichier `parametres.py` et que l'on ne parvient toujours pas à lire plus d'une colonne.
Dans les deux cas un message d'erreur apparaîtra pour spécifier le problème via une boîte de dialogue. De plus, celles-ci seront inscrites dans un fichier de log avec la date et l'heure à laquelle elles ont eu lieu. Le nom de ce fichier peut être spécifié dans le `Parametres.py`.

3.2.2 Cas nominal

Si aucune erreur n'est levée par l'application, le traitement des données peut commencer. Pour éviter de lire les fichiers plusieurs fois, les informations qu'ils contiennent sont gardées dans des objets de type `UE` ou `Etudiant` selon la donnée impliquée. Les `UE` sont alors regroupées dans un dictionnaire dont la clé est leur intitulé afin d'en accélérer l'accès. Les `Etudiant` sont quant à eux ajoutés à une liste que l'on ne parcourra qu'une seule fois.
Une fois la collecte des données effectuée, il est possible d'instancier les variables du modèle décrit dans la section 2.1.4. Ce travail est réalisé dans la fonction `ajouter_vars` du fichier `solveur.py`. Il est à noter que les variables sont par la suite rangées dans un dictionnaire dont la clé est l'étudiant qu'elles concernent, mais aussi sous d'autres formats dont notamment un dictionnaire regroupant les variables liées aux cours par créneau. Ce choix un peu plus gourmand en espace mémoire a été réalisé afin de limiter le temps de recherche des variables lors de la création des contraintes mais aussi pour faciliter l'ajout de ces dernières par un autre utilisateur.

3.3 Ajout de contraintes personnalisées

3.3.1 Principe

Nous avons vu que le solveur s'appuie sur certaines contraintes fixes comme celle empêchant qu'un étudiant suive deux UE sur le même créneaux. Cependant, il peut être souhaitable d'étendre le code pour ajouter des contraintes spécifiques à un semestre ou encore pour prendre en compte d'éventuels

changements dans le programme du master.

C'est donc dans cette optique que le fichier `contraintesPersonnalisées.py` a été ajouté. Il contient notamment une fonction `ajouter_contraintes` qui sera appelée lors de l'ajout des contraintes par le solveur. Celle-ci prend en argument le modèle auquel on souhaite ajouter une contrainte ainsi qu'un certain nombre de dictionnaires contenant les variables de décision du problème. Celle-ci sont explicitées ci-dessous.

- `vars_x` de la forme $etudiant_i : x_{ijk}$
- `vars_y` de la forme $etudiant_i : y_{ij}$
- `vars_x_par_ue` de la forme UE : $k : [x_{i_1 nom_{\text{UE}} k}, ..., x_{i_n nom_{\text{UE}} k}]$
- `vars_creneaux_incompatibles` de la forme $etudiant_i : creneau : [x_{ij\text{cours1}}, ..., x_{ij\text{coursn}}]$
- `vars_cours_par_ue` de la forme $etudiant_i$: UE : $[x_{inom_{\text{UE}}\text{cours1}}, ..., x_{inom_{\text{UE}}\text{coursn}}]$
- `vars_td_tme_par_ue` de la forme $etudiant_i$: UE : $[x_{inom_{\text{UE}}k}]$

avec i l'identifiant de l'$etudiant_i$ et nom_{UE} l'intitulé de UE.

3.3.2 Exemple de contrainte

Afin d'illustrer l'ajout d'une contrainte personnalisée, on se propose d'ajouter une contrainte interdisant à un étudiant de suivre à la fois l'UE de ml et celle de mll. On définit alors la fonction auxiliaire suivante :

```
def ajouter_contrainte_speciale_ml_mll(modele, vars_y):
    for etu, variables in vars_y.items():
        y_ml = solveur.chercher_variable_par_ue(variables, "ml")
        y_mll = solveur.chercher_variable_par_ue(variables, "mll")
        if (y_ml and y_mll):
            modele.addConstr(gp.quicksum([y_ml, y_mll]) <= 1)
```

On utilise alors la fonction `chercher_variable_par_ue` disponible dans `solveur.py` qui permet de retourner à partir d'une liste de variables de la forme y_{ij} et d'une UE ayant pour intitulé nom_{UE} la variable pour laquelle $j = nom_{\text{UE}}$ ou `False` si cette dernière n'est pas présente.
Il s'agit ensuite d'effectuer l'appel sur les variables y_{ij} de chaque $etudiant_i$. On peut maintenant ajouter une contrainte au modèle visant à s'assurer que la somme du retour des deux appels avec ml et mll n'excède pas un.
Il est tout de même à noter que si l'une des variables n'est pas présente, il ne faut pas ajouter la contrainte au modèle.

Enfin, il ne reste plus qu'à appeler la fonction dans `ajouter_contraintes` en ajoutant éventuellement une autre fonction annexe afin d'y regrouper toutes les contraintes du semestre.

```
def ajouter_contraintes(modele, vars_x, vars_y, vars_x_par_ue,
                vars_creneaux_incompatibles, vars_cours_par_ue, vars_td_tme_par_ue):
    ajouter_containtes_M1_S2(modele, vars_x, vars_y, vars_x_par_ue,
                        vars_creneaux_incompatibles, vars_cours_par_ue, vars_td_tme_par_ue)

def ajouter_containtes_M1_S2(modele, vars_x, vars_y, vars_x_par_ue,
                    vars_creneaux_incompatibles, vars_cours_par_ue, vars_td_tme_par_ue):
    ajouter_contrainte_speciale_ml_mll(modele, vars_y)
```

3.4 Génération d'affectations

Une fois les données d'entrée récupérées et les contraintes personnalisées définies, le gestionnaire de voeux peut ajouter les contraintes citées dans la section 2.1.4 et dans les contraintes ajoutées par

l'utilisateur.
Cependant, la génération des affectations n'est pas encore lancée. En effet, une première passe est effectuée afin de marquer les étudiants dont les voeux sont incohérents *i.e.* ceux qui ont formulé des voeux pour des UE dont les créneaux horaires se chevauchent. Ce marquage est réalisé en résolvant le modèle en ignorant les contraintes de capacité de groupe et en autorisant certains étudiants à avoir une affectation partielle. Ainsi, on considère tous ceux ayant obtenu moins d'UE que ce qu'ils sont censés avoir comme ayant formulé des voeux incohérents. Ils seront ensuite signalés à l'utilisateur lors de l'affectation des voeux.
Une fois cette première passe terminée, le solveur cherche à résoudre le problème avec toutes les contraintes énumérées précédemment, puis s'il ne parvient pas à trouver une solution, il réessaiera en autorisant des affectations partielles. Enfin, si malgré le relâchement des contraintes aucune solution n'est trouvée, une message d'erreur sera généré. Notons tout de même que ce cas ne devrait pas arriver avec les contraintes de base du modèle, mais qu'il peut être dû à une mauvaise utilisation des contraintes personnalisées.

4 Résultats à partir d'un exemple concret

Dans cette partie, on se propose d'illustrer à l'aide d'un exemple concret de quelle manière l'application doit être utilisée.

4.1 Lancement de l'application

Pour lancer l'application, il est nécessaire que la machine hôte possède *Python 3*. Une licence gurobi valide ainsi que les modules `gurobipy` et `Tkinter` sont également nécessaires.
Une fois toutes ces conditions réunies, il suffit d'ouvrir un terminal dans le dossier contenant le code source de l'application et d'y taper la commande `python gestionnaire_de_voeux.py`. Une fenêtre semblable à celle de la figure 1 devrait alors apparaître.

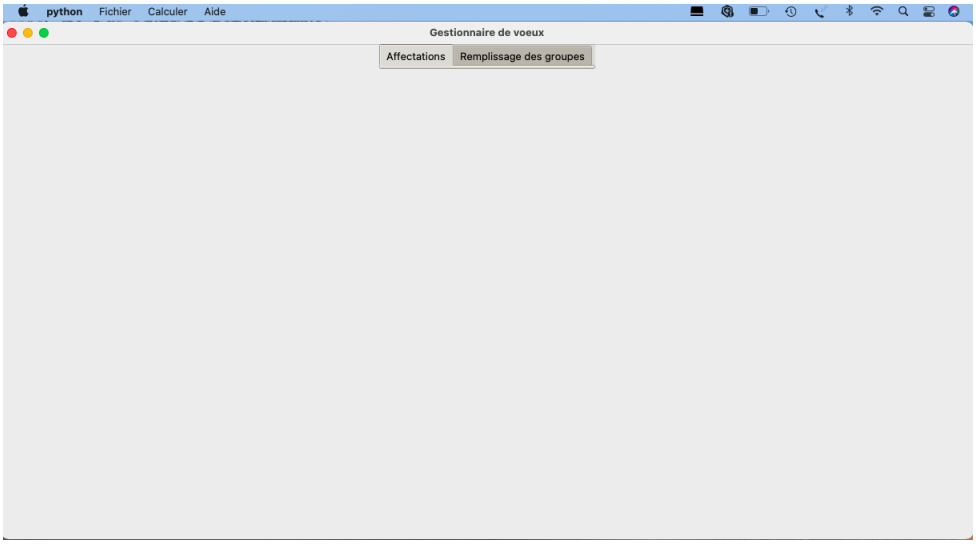


FIGURE 1 – Interface de l'application

4.2 Chargement des fichiers

Les fichiers peuvent être fournis à l'application via le sous menu *Fichier*. Ce sous menu donne ensuite la possibilité de charger le fichier contenant les voeux ou celui contenant l'emploi du temps. Une boîte de dialogue signale alors le bon déroulement de l'opération (cf figure 2). Dans cette démonstration nous fournirons les fichiers 6.2.1 et 6.2.2 de l'annexe.

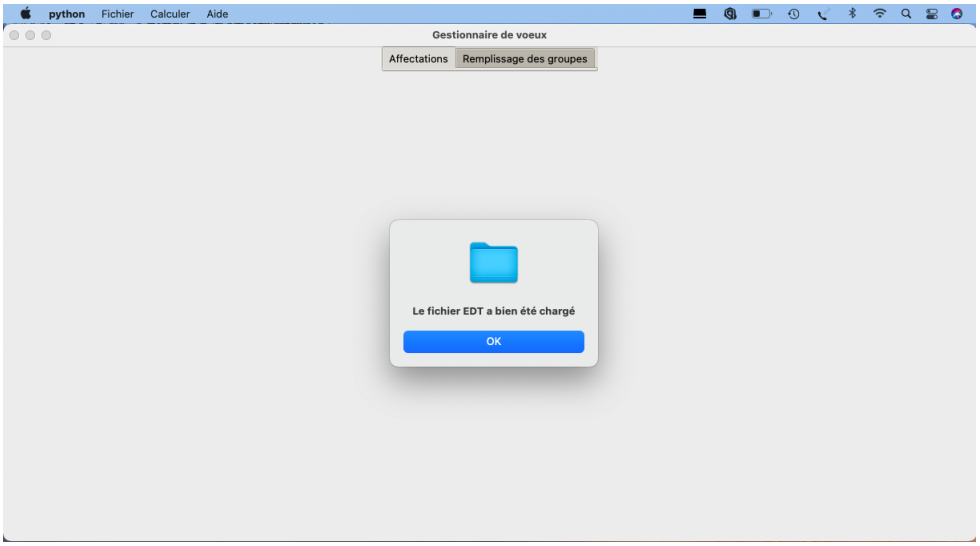


FIGURE 2 – Message de confirmation du chargement du fichier des créneaux horaire

4.3 Génération des affectations

Pour générer les affectations, il suffit maintenant d'accéder à l'option *Résoudre le modèle* dans le sous-menu *Calculer*. Sur notre exemple, un avertissement s'affiche signalant que trois étudiants ont une affectations partielle (cf figure 3).

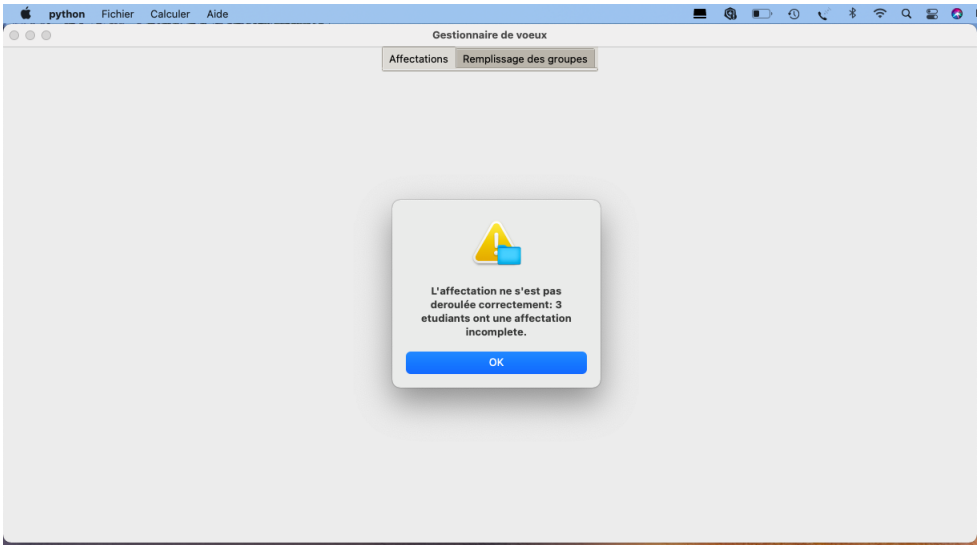
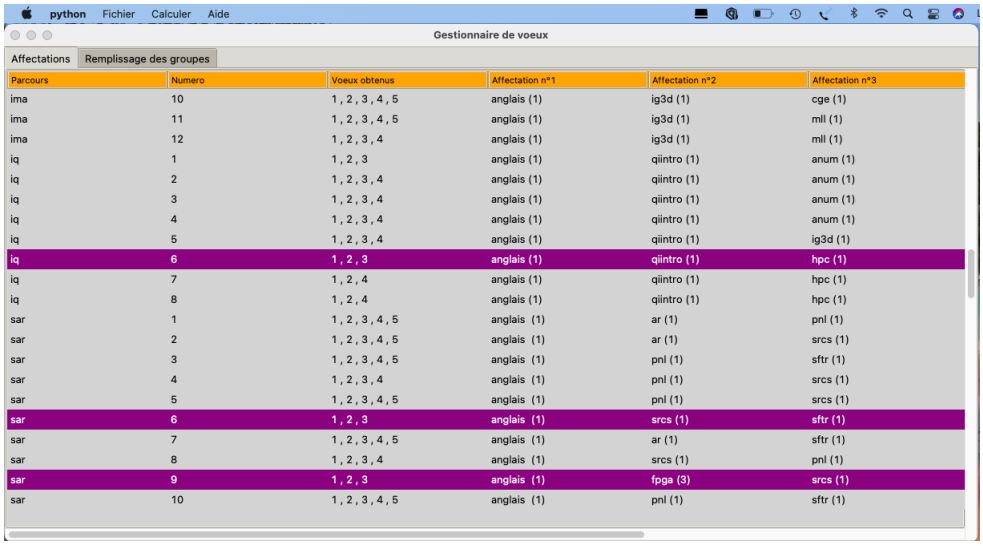


FIGURE 3 – Message d'avertissement

Les affectations sont ensuite affichées dans l'application. Chaque étudiant se voit alors attribué des UE ainsi qu'un groupe de TD et de TME qui est précisé entre parenthèses.
En descendant un peu dans les résultats, on remarque que les étudiants ayant obtenus une affectation partielle sont marqués en violet afin de signaler un problème de chevauchement de leur voeux (cf figure 4).

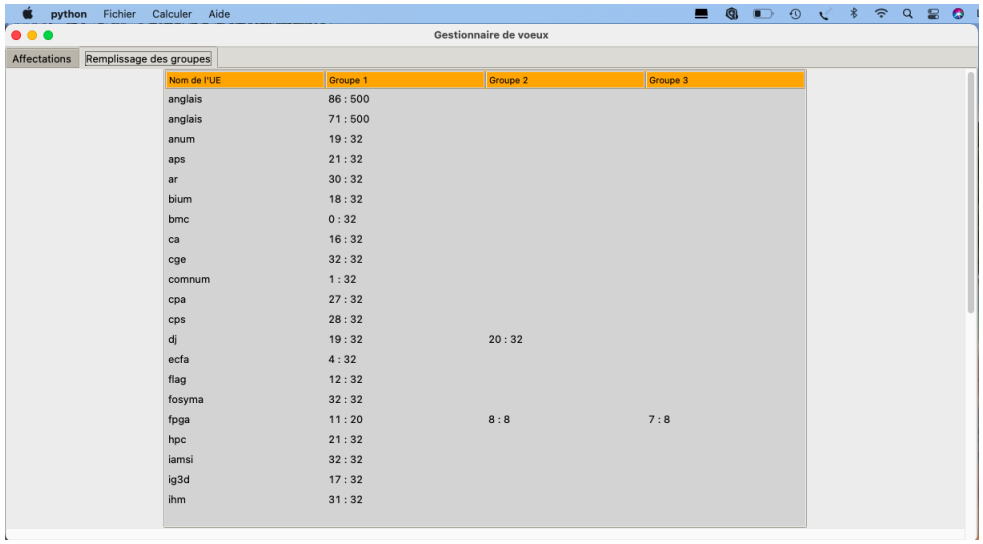


Parcours	Numero	Voeux obtenus	Affectation n°1	Affectation n°2	Affectation n°3
ima	10	1, 2, 3, 4, 5	anglais (1)	ig3d (1)	cge (1)
ima	11	1, 2, 3, 4, 5	anglais (1)	ig3d (1)	mll (1)
ima	12	1, 2, 3, 4	anglais (1)	ig3d (1)	mll (1)
iq	1	1, 2, 3	anglais (1)	qiintro (1)	anum (1)
iq	2	1, 2, 3, 4	anglais (1)	qiintro (1)	anum (1)
iq	3	1, 2, 3, 4	anglais (1)	qiintro (1)	anum (1)
iq	4	1, 2, 3, 4	anglais (1)	qiintro (1)	anum (1)
iq	5	1, 2, 3, 4	anglais (1)	qiintro (1)	ig3d (1)
iq	6	1, 2, 3	anglais (1)	qiintro (1)	hpc (1)
iq	7	1, 2, 4	anglais (1)	qiintro (1)	hpc (1)
iq	8	1, 2, 4	anglais (1)	qiintro (1)	hpc (1)
sar	1	1, 2, 3, 4, 5	anglais (1)	ar (1)	pnl (1)
sar	2	1, 2, 3, 4, 5	anglais (1)	ar (1)	srcs (1)
sar	3	1, 2, 3, 4, 5	anglais (1)	pnl (1)	sftr (1)
sar	4	1, 2, 3, 4	anglais (1)	pnl (1)	srcs (1)
sar	5	1, 2, 3, 4, 5	anglais (1)	pnl (1)	srcs (1)
sar	6	1, 2, 3	anglais (1)	srcs (1)	sftr (1)
sar	7	1, 2, 3, 4, 5	anglais (1)	ar (1)	sftr (1)
sar	8	1, 2, 3, 4	anglais (1)	srcs (1)	pnl (1)
sar	9	1, 2, 3	anglais (1)	fpga (3)	srcs (1)
sar	10	1, 2, 3, 4, 5	anglais (1)	pnl (1)	sftr (1)

FIGURE 4 – Résultat des affectations

4.4 Remplissage des groupes

Une autre fonctionnalité du logiciel permet d'afficher le remplissage des groupes. Elle est accessible via le sous-menu *Calculer*. Elle produit un affichage similaire à celui de la figure 5.



Nom de l'UE	Groupe 1	Groupe 2	Groupe 3
anglais	86 : 500		
anglais	71 : 500		
anum	19 : 32		
aps	21 : 32		
ar	30 : 32		
bium	18 : 32		
bmc	0 : 32		
ca	16 : 32		
cge	32 : 32		
comnum	1 : 32		
cpa	27 : 32		
cps	28 : 32		
dj	19 : 32	20 : 32	
ecfa	4 : 32		
flag	12 : 32		
fosyma	32 : 32		
fpga	11 : 20	8 : 8	7 : 8
hpc	21 : 32		
iamsi	32 : 32		
ig3d	17 : 32		
ihm	31 : 32		

FIGURE 5 – Résultat des affectations

4.5 Export des résultats

Enfin, il sera possible d'exporter les affectations ainsi que le taux de remplissage des groupes. Cela se fera via le sous-menu *Fichier*. Une boîte de dialogue s'ouvrira alors afin de laisser le choix à l'utilisateur de l'emplacement où sauvegarder les fichiers en question (cf 6).

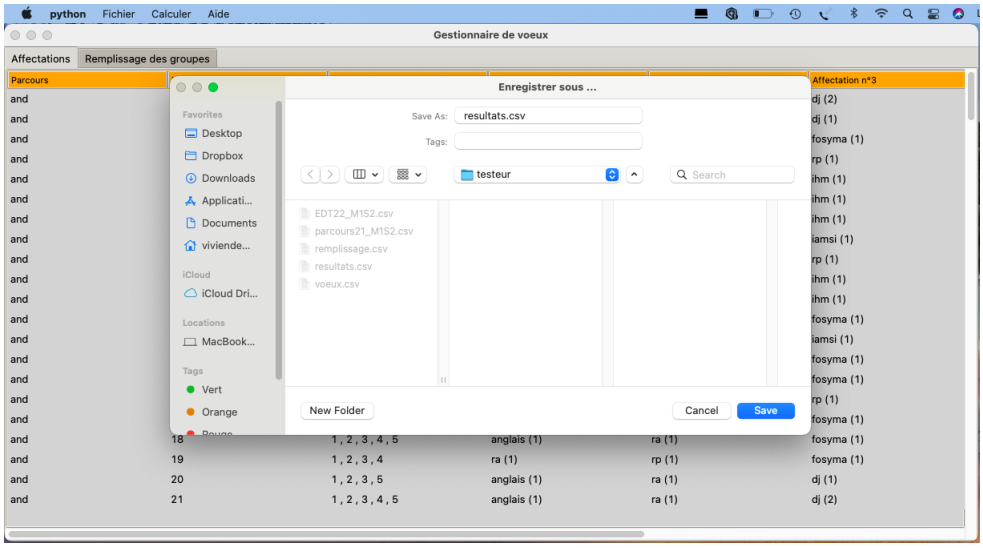


FIGURE 6 – Export du fichier contenant les affectations

5 Conclusion

Au cours de ce projet, nous avons épuré l'application précédente en nous concentrant exclusivement sur la gestion des voeux des étudiants. Ainsi, le nombre de fichiers est passé de 13 à 10 avec des noms plus explicites et des méthodes commentées. Les fichiers d'entrée ont eux aussi été réduits. En effet, seuls deux fichiers sont nécessaires : l'un contenant les voeux des étudiants et l'autre contenant les créneaux horaires sur lesquels les cours ont lieu.
De plus, nous avons pris soin d'améliorer l'expérience utilisateur en incorporant une interface graphique, en faisant remonter les erreurs lorsqu'il y en a et en réduisant drastiquement le temps d'exécution. Par ailleurs dans la version précédente il était possible que la résolution ne se termine pas. Cette amélioration a été rendue possible en modifiant le modèle mathématique et en transmettant directement tous les voeux des étudiants d'un coup.
Il est à noter qu'il serait possible d'utiliser un solveur de contrainte *open source* comme *GLPK*. Ce dernier dispose d'ailleurs d'un module module Python nommé `PyGLPK`. Cette modification serait facilement réalisable car les modules fonctionnent de manière similaire. Toutefois, il est à noter que *Gurobi* est plus efficace que *GLPK* dans la plupart des cas.

6 Annexe

6.1 Les contraintes

6.1.1 contrainte des UE obligatoires

```
def ajouter_contrainte_ue_obligatoires(modele, vars_y):
    for etu, variables in vars_y.items():
        for i in range(etu.nb_ue_obligatoires):
            nom = str(variables[i]) + " == 1"
            modele.addConstr(variables[i] == 1, name=nom)
    modele.update()
```

6.1.2 contrainte du groupe unique

```
def ajouter_contrainte_groupe_unique(modele, vars_x_par_ue):
    for ue, dico in vars_x_par_ue.items():
        #La contrainte ne concerne pas les UE avec un groupe ou moins
        if ue.nb_groupes <= 1:
            continue
        for i in range(len(dico[0])):
            somme_variables = []
            for k in range(ue.nb_groupes):
                somme_variables.append(dico[k][i])
            modele.addConstr(gp.quicksum(somme_variables) <= 1)
    modele.update()
```

6.1.3 contrainte de la capacité d'un groupe

```
def ajouter_contrainte_capacite_groupes(modele, vars_x_par_ue):
    for ue, dico in vars_x_par_ue.items():
        for k, variables in dico.items():
            modele.addConstr(gp.quicksum(variables) <= ue.capacites_groupes[k])
    modele.update()
```

6.1.4 contrainte du nombre d'UE par étudiant

```
def ajouter_contrainte_nombre_ue_etudiant(modele, vars_y, contrainte_relachee):
    """Ajout d'une contrainte pour s'assurer qu'un etudiant suit bien le bon
    nombre d'UE"""
    if contrainte_relachee:
        for etu, variables in vars_y.items():
            nb_ue = etu.nb_ue_a_suivre
            modele.addConstr(gp.quicksum(variables) <= nb_ue)
    else:
        for etu, variables in vars_y.items():
            nb_ue = etu.nb_ue_a_suivre
            modele.addConstr(gp.quicksum(variables) == nb_ue)
    modele.update()
```

6.1.5 contrainte d'emploi du temps

```
def ajouter_contrainte_edt(modele, vars_creneaux_incompatibles):
    for etu, dico in vars_creneaux_incompatibles.items():
        for creneau, variables in dico.items():
            if len(variables) <= 1 or creneau == Parametres.creneau_poubelle:
                continue
            modele.addConstr(gp.quicksum(variables) <= 1)
    modele.update()
```

6.1.6 contrainte du créneau de cours

```
def ajouter_contrainte_inscription_complete(modele, vars_cours_par_ue,
vars_td_tme_par_ue, vars_y):
    for etu, dico in vars_cours_par_ue.items():
        cpt = 0
        for ue, variables in dico.items():
            modele.addConstr(gp.quicksum(variables) == gp.quicksum(
            vars_td_tme_par_ue[etu][ue]))
            modele.addConstr(vars_y[etu][cpt] == gp.quicksum(variables))
            cpt += 1
    modele.update()
```

6.2 Les fichiers d'entrée

6.2.1 Fichier des créneaux de cours

id_ue	intitule	nb_groupes	capac1	capac2	capac3	capac4	capac5	cours1	cours2	td1	tme1	td2	tme2	td3	tme3	td4	tme4	td5	tme5
1	anglais	1	500					1		2									
2	anglais	1	500					13		14									
3	anum	1	32					11		12									
4	aps	1	32					14		11	12								
5	ar	1	32					7		3	4								
6	bium	1	32					3		11	12								
7	bmc	1	32					23		24	7								
8	ca	1	32					1		6	7								
9	cge	1	32					-1											
10	comnum	1	32					22		23	24								
11	cpa	1	32					22		23	24								
12	cps	1	32					9		3	4								
13	dj	2	32	32				3		16	17	21	22						
14	ecfa	1	32					4	17	1	2								
15	flag	1	32					21		1	2								
16	fosyma	1	32					4		6	7								
17	fpga	3	20	8	8			9		3	4	6	7	6	7				
18	hpc	1	32					22		6	7								
19	iamsi	1	32					13		6	7								
20	ig3d	1	32					9		18	19								
21	ihm	1	32					9		11	12								
22	ioc	1	40					16		21	22								
23	isec	1	32					16		23	24								
24	ml	2	32	32				14		6	7	16	17						
25	mll	1	32					-1		11	12								
26	mmcn	1	32					17		21	22								
27	multi	1	40					12		23	24								
28	paf	1	32					13		18	19								
29	pc2r	1	32					2		16	17								
30	pnl	1	40					6		18	19								
31	qiintro	1	32					13		23	24								
32	ra	1	50					14		18	19								
33	rital	1	32					4		18	19								
34	rout	1	32					1		18	19								
35	rp	1	32					23		16	17								
36	sam	1	32					9		21	22								
37	sbas	1	32					6		3	4								
38	sdm	1	32					6		7	8								
39	sftr	1	32					11		16	17								
40	srcs	1	32					12		23	24								
41	projet	1	500					-1											

6.2.2 Fichier de voeux

parcours	num	oblig1	oblig2	oblig3	cons1	cons2	cons3	cons4	cons5	equiv1	equiv2	equiv3	equiv4	equiv5
and	1	anglais	ra		dj	fosyma	rp			ml	iamsi	ihm		

parcours	num	oblig1	oblig2	oblig3	cons1	cons2	cons3	cons4	cons5	equiv1	equiv2	equiv3	equiv4	equiv5
and	2	anglais	ra		ml	dj	fosyma			ihm	rp	iamsi		
and	3	ra			ihm	fosyma	dj			rp	ml	iamsi		
and	4	anglais	ra		ml	rp	dj			iamsi	ihm	fosyma		
and	5	anglais	ra		ihm	fosyma	dj			ml	rp	iamsi		
and	6	anglais	ra		ihm	dj	ml			iamsi	fosyma	rp		
and	7	anglais	ra		ihm	dj	ml			iamsi	fosyma	rp		
and	8	anglais	ra		iamsi	fosyma	dj			rp	ml	ihm		
and	9	anglais	ra		rp	dj	ml			ihm	fosyma	iamsi		
and	10	anglais	ra		ihm	fosyma	rp			iamsi	ml	dj		
and	11	anglais	ra		ihm	fosyma	dj			ml	rp	iamsi		
and	12	anglais	ra		fosyma	ihm	dj			rp	ml	iamsi		
and	13	anglais	ra		fosyma	iamsi	dj			ml	rp	ihm		
and	14	anglais	ra		fosyma	rp	ml			dj	ihm	iamsi		
and	15	anglais	ra		ml	fosyma	dj			ihm	iamsi	rp		
and	16	anglais	ra		rp	ihm	fosyma			dj	iamsi	ml		
and	17	anglais	ra		ml	fosyma	dj			iamsi	ihm	rp		
and	18	anglais	ra		fosyma	dj	rp			ml	ihm	iamsi		
and	19	ra			rp	fosyma	ihm			dj	ml	iamsi		
and	20	anglais	ra		dj	ml	fosyma			iamsi	ihm	rp		
and	21	anglais	ra		dj	rp	fosyma			ml	iamsi	ihm		
and	22	anglais	ra		ihm	dj	fosyma			rp	iamsi	ml		
and	23	anglais	ra		dj	ihm	fosyma			iamsi	rp	ml		
and	24	anglais	ra		ihm	fosyma	dj			ml	rp	iamsi		
and	25	anglais	ra		fosyma	dj	iamsi			ihm	rp	ml		
and	26	anglais	ra		rp	ml	dj			iamsi	ihm	fosyma		
and	27	anglais	ra		ihm	iamsi	fosyma			rp	dj	ml		
and	28	anglais	ra		ml	ihm	fosyma			dj	rp	iamsi		
and	29	anglais	ra		ml	ihm	fosyma			dj	iamsi	rp		
and	30	anglais	ra		ihm	fosyma	dj			rp	iamsi	ml		
and	31	anglais	ra		rp	iamsi	ihm			ml	dj	fosyma		
and	32	anglais	ra		fosyma	ihm	rp			dj	iamsi	ml		
and	33	anglais	ra		fosyma	ihm	rp			dj	iamsi	ml		
and	34	anglais	ra		fosyma	ihm	rp			dj	ml	iamsi		
and	35	anglais	ra		ihm	dj	ml			rp	fosyma	iamsi		
and	36	anglais	ra		ihm	dj	ml			rp	fosyma	iamsi		
and	37	anglais	ra		dj	fosyma	ihm			iamsi	rp	ml		
and	38	anglais	ra		fosyma	rp	dj			ihm	ml	iamsi		
and	39	anglais	ra		dj	fosyma	rp			ml	iamsi	ihm		
and	40	anglais	ra		rp	ml	dj			fosyma	iamsi	ihm		
and	41	anglais	ra		rp	fosyma	ihm			dj	ml	iamsi		
and	42	anglais	ra		fosyma	rp	dj			iamsi	ml	ihm		
and	43	anglais	ra		fosyma	dj	iamsi			ihm	ml	rp		
and	44	anglais	ra		ml	rp	dj			fosyma	iamsi	ihm		
dac	1	anglais			ml	rital	iamsi	bium		sam				
dac	2	anglais			ml	rital	sam	bium		iamsi				
dac	3	anglais			ml	iamsi	rital	bium		sam				
dac	4	anglais			ml	bium	rital	iamsi		sam				
dac	5	anglais			ml	rital	sam	bium		iamsi				
dac	6	anglais			ml	rital	bium	sam		iamsi				
dac	7	anglais			sam	rital	ml	iamsi		bium				
dac	8	anglais			ml	rital	iamsi	sam		bium				
dac	9	anglais			ml	rital	sam	bium		iamsi				
dac	10	anglais			ml	iamsi	rital	bium		sam				
dac	11	anglais			ml	iamsi	rital	sam		bium				
dac	12	anglais			iamsi	sam	ml	rital		bium				
dac	13	anglais			ml	bium	rital	sam		iamsi				
dac	14	anglais			rital	sam	iamsi	ml		bium				
dac	15	anglais			ml	bium	rital	sam		iamsi				
dac	16	anglais			ml	bium	iamsi	rital		sam				
dac	17	anglais			ml	iamsi	rital	bium		sam				
dac	18	anglais			sam	ml	iamsi	bium		rital				
dac	19	anglais			ml	rital	bium	sam		iamsi				
dac	20	anglais			ml	bium	rital	iamsi		sam				
dac	21	anglais			ml	sam	bium	rital		iamsi				
dac	22	anglais			iamsi	ml	bium	sam		rital				
dac	23	anglais			ml	rital	iamsi	bium		sam				
dac	24	anglais			ml	rital	iamsi	sam		bium				
dac	25	anglais			ml	rital	iamsi	sam		bium				
ima	1	ig3d			rp	iamsi				ihm	dj	mll		
ima	2	anglais	ig3d		cge	ihm	fosyma			mll	iamsi	ml	hpc	
ima	3	anglais	ig3d		anum	ml	hpc			ra	flag	dj	iamsi	
ima	4	anglais	ig3d		cge	iamsi	ihm			mll	rital	fosyma	hpc	
ima	5	anglais	ig3d		ra	ml	rital			cge	rp	hpc	fosyma	
ima	6	anglais	ig3d		mll	rital	ml			anum	dj	fosyma	iamsi	
ima	7	anglais	ig3d		mll	rp	rital			cge	iamsi	ihm	hpc	
ima	8	anglais	ig3d		ra	fosyma	ml			anum	dj	iamsi	rital	
ima	9	anglais	ig3d		ra	dj	iamsi			mll	rital	hpc	fosyma	
ima	10	anglais	ig3d		cge	ml	dj			mll	rp	iamsi	flag	
ima	11	anglais	ig3d		mll	iamsi	rp			ra	dj	rital	ihm	
ima	12	anglais	ig3d		mll	hpc	ml			ra	rp	rital	iamsi	
iq	1	anglais	qiintro		anum	flag				pc2r	ig3d			
iq	2	anglais	qiintro		anum	ig3d				sdm	flag			
iq	3	anglais	qiintro		anum	hpc				pc2r	sdm			
iq	4	anglais	qiintro		anum	hpc				sdm	pc2r			
iq	5	anglais	qiintro		ig3d	hpc				flag				
iq	6	anglais	qiintro		hpc	sdm				pc2r				
iq	7	anglais	qiintro		flag	hpc				ig3d				
iq	8	anglais	qiintro		flag	hpc				ig3d				
sar	1	anglais			ar	pnl	srcs	sftr		multi	ioc	fpga		
sar	2	anglais			ar	srcs	pnl	sftr		ioc	multi	fpga		
sar	3	anglais			pnl	sftr	srcs	ar		fpga	ioc	multi		
sar	4	anglais			pnl	srcs	ar	fpga		sftr	ioc	multi		
sar	5	anglais			pnl	srcs	ar	ioc		sftr	multi	fpga		
sar	6	anglais			srcs	sftr	multi	ioc		fpga				
sar	7	anglais			ar	sftr	srcs	pnl		ioc	multi	fpga		
sar	8	anglais			srcs	pnl	ioc	sftr		multi	ar	fpga		
sar	9	anglais			fpga	srcs	multi	ar		sftr				
sar	10	anglais			pnl	sftr	srcs	ar		multi	ioc	fpga		
sar	11	anglais			pnl	srcs	ar	sftr		fpga	ioc	multi		
sar	12	anglais			pnl	ar	sftr	ioc		fpga	multi	srcs		
sar	13	anglais			pnl	srcs	sftr	ar		ioc	fpga	multi		
sar	14	anglais			pnl	sftr	multi	ioc		ar	srcs	fpga		
sar	15	anglais			ar	sftr	srcs	ioc		multi	fpga	pnl		
sar	16	anglais			pnl	sftr	srcs	multi		ar	fpga	ioc		
sar	17	anglais			srcs	sftr	ar	multi		fpga	ioc	pnl		
sar	18	anglais			srcs	pnl	ar	sftr		ioc	multi	fpga		
sar	19	anglais			pnl	srcs	ar	sftr		multi	ioc	fpga		
sar	20	anglais			pnl	srcs	ar	sftr		multi	ioc	fpga		
sar	21	anglais			pnl	srcs	sftr	ar		multi	ioc	fpga		
sar	22	anglais			srcs	pnl	ar	sftr		fpga	ioc	multi		
sar	23	anglais			pnl	ioc	ar	fpga		sftr	srcs	multi		
sar	24	anglais			pnl	ar	srcs	ioc		sftr	fpga	multi		

parcours	num	oblig1	oblig2	oblig3	cons1	cons2	cons3	cons4	cons5	equiv1	equiv2	equiv3	equiv4	equiv5
sar	25	anglais			pnl	srcs	ar	sftr		multi	fpga	ioc		
sar	26	anglais			pnl	srcs	ioc	multi		sftr	ar	fpga		
sar	27	anglais			sftr	ar	srcs	fpga		pnl	multi	ioc		
sar	28	anglais			pnl	ioc	ar	srcs		multi	sftr	fpga		
sar	29	anglais			pnl	srcs	ar	sftr		fpga	ioc	multi		
sar	30	anglais			pnl	ar	sftr	srcs		ioc	fpga	multi		
sesi	1	anglais	cge		pnl	ar	multi	ioc		comnum	fpga			
sesi	2	anglais	cge		fpga	multi	ioc	pnl		comnum	projet			
sesi	3	anglais	cge		ioc	multi	pnl	fpga		comnum	ar			
sesi	4	anglais	cge		projet	multi	fpga	ioc		pnl	ecfa			
sesi	5	anglais	cge		multi	ioc	projet	fpga		ecfa	comnum			
sesi	6	anglais	cge		projet	ecfa	fpga	multi		ioc	comnum			
sesi	7	anglais	cge		projet	ioc	multi	pnl		fpga	ar			
sesi	8	anglais	cge		fpga	multi	projet	ioc		pnl	ar			
sesi	9	anglais	cge		projet	fpga	ioc	multi		pnl	ecfa			
sesi	10	anglais	cge		projet	multi	pnl	fpga		ar	ioc			
sesi	11	anglais	cge		multi	fpga	ioc	pnl		projet	ar			
sesi	12	anglais	cge		multi	fpga	ioc	projet		ecfa	comnum			
sesi	13	anglais	cge		ioc	projet	pnl	fpga		ar	multi			
sesi	14	anglais	cge		ioc	multi	projet	ecfa		comnum	fpga			
sesi	15	anglais	cge		fpga	multi	projet	ar		ioc	pnl			
sesi	16	anglais			multi	rout	projet	pnl		ar	comnum			
sesi	17	anglais	cge		multi	ioc	pnl	projet		ar	fpga			
sesi	18	anglais	cge		multi	ioc	pnl	projet		fpga	ar			
sesi	19	anglais	cge		multi	fpga	ioc	pnl		ar	comnum			
sesi	20	anglais	cge		multi	fpga	ioc	pnl		ar	comnum			
sesi	21	anglais	cge		multi	fpga	ioc	pnl		ar	comnum			
sesi	22	anglais	cge		fpga	pnl	multi	ioc		projet	rout			
sesi	23	anglais	cge		ioc	fpga	comnum	projet		multi	ecfa			
sesi	24	anglais	cge		ioc	multi	projet	ecfa		comnum	fpga			
sesi	25	anglais	cge		ioc	fpga	pnl	multi		projet	ecfa			
sesi	26	anglais	cge		ecfa	comnum	fpga	ioc		multi	projet			
sesi	27	anglais	cge		multi	projet	pnl	fpga		ioc	ar			
sesi	28	anglais	cge		ioc	fpga	projet	multi		rout	pnl			
sesi	29	anglais	cge		multi	fpga	ioc	pnl		ar	comnum			
sfpn	1	anum				flag	hpc	fpga		ml	pnl			
sfpn	2	anglais	anum			flag	hpc	isec		pnl	ml	fpga		
sfpn	3	anglais	anum			flag	hpc	isec		fpga	ml	pnl		
sfpn	4	anglais	anum			isec	flag	hpc		ml	pnl	fpga		
sfpn	5	anglais	anum			flag	hpc	isec		pnl	fpga	ml		
sfpn	6	anglais	anum			hpc	flag	isec		ml	pnl	fpga		
sfpn	7	anglais	anum			isec	flag	hpc		ml	fpga	pnl		
sfpn	8	anglais	anum			isec	hpc	flag		pnl	fpga	ml		
sfpn	9	anglais	anum			isec	flag	hpc		fpga	pnl	ml		
sfpn	10	anglais	anum			flag	hpc	isec		ml	fpga	pnl		
sfpn	11	anglais				hpc				fpga	pnl	ml		
sfpn	12	anglais	anum			isec	flag	hpc		ml	pnl	fpga		
sfpn	13	anglais	anum			hpc	isec	flag		pnl	fpga	ml		
stl	1				ca	paf								
stl	2				paf	pc2r	aps	cpa		ca	cps			
stl	3				paf					cps	cpa			
stl	4				pc2r	cpa	cps	aps		paf	ca			
stl	5				pc2r	cpa	cps	aps		paf	ca			
stl	6				pc2r	cpa	cps	aps		paf	ca			
stl	7				aps	paf				cpa	cps			
stl	8				pc2r	cpa	cps	aps		paf	ca			
stl	9				pc2r	cps	cpa	paf		ca	aps			
stl	10				ca	cps	cpa	paf		pc2r	aps			
stl	11				ca	cps	cpa	paf		pc2r	aps			
stl	12				paf	cpa	pc2r	cps		aps	ca			
stl	13				cps	pc2r	cpa	aps		ca	paf			
stl	14				cps	pc2r	cpa	aps		ca	paf			
stl	15				cpa	cps	ca	paf		aps	pc2r			
stl	16				aps	pc2r	paf			ca	cpa			
stl	17				paf	aps	pc2r			ca	cps			
stl	18				paf	pc2r	aps	ca		cps	cpa			
stl	19				cpa	cps	paf	ca		pc2r	aps			
stl	20				cpa	cps	paf	ca		pc2r	aps			
stl	21				paf	pc2r	aps	cpa		ca	cps			
stl	22				cpa	cps	ca	paf		pc2r	aps			
stl	23				pc2r	paf				cps	ca			
stl	24				ca	cpa	paf	aps	pc2r	cps				
stl	25				paf	cpa	cps	pc2r		aps	ca			
stl	26				pc2r	paf				aps	ca			
stl	27				paf	pc2r	aps	cps		cpa	ca			
stl	28				ca	paf	aps	cpa		pc2r	cps			
stl	29				cps	pc2r	cpa	aps		ca	paf			
stl	30				paf	aps	pc2r	ca		cpa	cps			
stl	31				cps	pc2r	paf	cpa		aps	ca			
stl	32				cps	pc2r	cpa	ca		aps	paf			
stl	33				cps	pc2r	cpa	ca		aps	paf			
stl	34				pc2r	cps	paf	aps		cpa	ca			
stl	35				cps	cpa	ca	aps		pc2r	paf			
stl	36				aps	cps	pc2r	ca		paf	cpa			
stl	37				cps	pc2r	cpa	ca		paf	aps			
stl	38				pc2r	cps	cpa	paf		aps	ca			
stl	39				paf	pc2r	cpa	cps		aps	ca			
stl	40				paf	pc2r	cps			aps	cpa			
stl	41				paf	pc2r	aps	cps		ca	cpa			

6.3 Fichiers de sortie

6.3.1 Fichier des affectations

parcours	num	oblig1	oblig2	oblig3	cons1	cons2	cons3	cons4	cons5	equiv1	equiv2	equiv3	equiv4	equiv5
and	1	anglais	ra		dj	fosyma	rp			ml	iamsi	ihm		
and	2	anglais	ra		ml	dj	fosyma			ihm	rp	iamsi		
and	3	ra			ihm	fosyma	dj			rp	ml	iamsi		
and	4	anglais	ra		ml	rp	dj			iamsi	ihm	fosyma		
and	5	anglais	ra		ihm	fosyma	dj			ml	rp	iamsi		
and	6	anglais	ra		ihm	dj	ml			iamsi	fosyma	rp		
and	7	anglais	ra		ihm	dj	ml			iamsi	fosyma	rp		
and	8	anglais	ra		iamsi	fosyma	dj			rp	ml	ihm		
and	9	anglais	ra		rp	dj	ml			ihm	fosyma	iamsi		

parcours	num	oblig1	oblig2	oblig3	cons1	cons2	cons3	cons4	cons5	equiv1	equiv2	equiv3	equiv4	equiv5
and	10	anglais	ra		ihm	fosyma	rp			iamsi	ml	dj		
and	11	anglais	ra		ihm	fosyma	dj			ml	rp	iamsi		
and	12	anglais	ra		fosyma	ihm	dj			rp	ml	iamsi		
and	13	anglais	ra		fosyma	iamsi	dj			ml	rp	ihm		
and	14	anglais	ra		fosyma	rp	ml			dj	ihm	iamsi		
and	15	anglais	ra		ml	fosyma	dj			ihm	iamsi	rp		
and	16	anglais	ra		rp	ihm	fosyma			dj	iamsi	ml		
and	17	anglais	ra		ml	fosyma	dj			iamsi	ihm	rp		
and	18	anglais	ra		fosyma	dj	rp			ml	ihm	iamsi		
and	19	ra			rp	fosyma	ihm			dj	ml	iamsi		
and	20	anglais	ra		dj	ml	fosyma			iamsi	ihm	rp		
and	21	anglais	ra		dj	rp	fosyma			ml	iamsi	ihm		
and	22	anglais	ra		ihm	dj	fosyma			rp	iamsi	ml		
and	23	anglais	ra		dj	ihm	fosyma			iamsi	rp	ml		
and	24	anglais	ra		ihm	fosyma	dj			ml	rp	iamsi		
and	25	anglais	ra		fosyma	dj	iamsi			ihm	rp	ml		
and	26	anglais	ra		rp	ml	dj			iamsi	ihm	fosyma		
and	27	anglais	ra		ihm	iamsi	fosyma			rp	dj	ml		
and	28	anglais	ra		ml	ihm	fosyma			dj	rp	iamsi		
and	29	anglais	ra		ml	ihm	fosyma			dj	iamsi	rp		
and	30	anglais	ra		ihm	fosyma	dj			rp	iamsi	ml		
and	31	anglais	ra		rp	iamsi	ihm			ml	dj	fosyma		
and	32	anglais	ra		fosyma	ihm	rp			dj	iamsi	ml		
and	33	anglais	ra		fosyma	ihm	rp			dj	iamsi	ml		
and	34	anglais	ra		fosyma	ihm	rp			dj	ml	iamsi		
and	35	anglais	ra		ihm	dj	ml			rp	fosyma	iamsi		
and	36	anglais	ra		ihm	dj	ml			rp	fosyma	iamsi		
and	37	anglais	ra		dj	fosyma	ihm			iamsi	rp	ml		
and	38	anglais	ra		fosyma	rp	dj			ihm	ml	iamsi		
and	39	anglais	ra		dj	fosyma	rp			ml	iamsi	ihm		
and	40	anglais	ra		rp	ml	dj			fosyma	iamsi	ihm		
and	41	anglais	ra		rp	fosyma	ihm			dj	ml	iamsi		
and	42	anglais	ra		fosyma	rp	dj			iamsi	ml	ihm		
and	43	anglais	ra		fosyma	dj	iamsi			ihm	ml	rp		
and	44	anglais	ra		ml	rp	dj			fosyma	iamsi	ihm		
dac	1	anglais			ml	rital	iamsi	bium		sam				
dac	2	anglais			ml	rital	sam	bium		iamsi				
dac	3	anglais			ml	iamsi	rital	bium		sam				
dac	4	anglais			ml	bium	rital	iamsi		sam				
dac	5	anglais			ml	rital	sam	bium		iamsi				
dac	6	anglais			ml	rital	bium	sam		iamsi				
dac	7	anglais			sam	rital	ml	iamsi		bium				
dac	8	anglais			ml	rital	iamsi	sam		bium				
dac	9	anglais			ml	rital	sam	bium		iamsi				
dac	10	anglais			ml	iamsi	rital	bium		sam				
dac	11	anglais			ml	iamsi	rital	sam		bium				
dac	12	anglais			iamsi	sam	ml	rital		bium				
dac	13	anglais			ml	bium	rital	sam		iamsi				
dac	14	anglais			rital	sam	iamsi	ml		bium				
dac	15	anglais			ml	bium	rital	sam		iamsi				
dac	16	anglais			ml	bium	iamsi	rital		sam				
dac	17	anglais			ml	iamsi	rital	bium		sam				
dac	18	anglais			sam	ml	iamsi	bium		rital				
dac	19	anglais			ml	rital	bium	sam		iamsi				
dac	20	anglais			ml	bium	rital	iamsi		sam				
dac	21	anglais			ml	sam	bium	rital		iamsi				
dac	22	anglais			iamsi	ml	bium	sam		rital				
dac	23	anglais			ml	rital	iamsi	bium		sam				
dac	24	anglais			ml	rital	iamsi	sam		bium				
dac	25	anglais			ml	rital	iamsi	sam		bium				
ima	1	ig3d			rp	iamsi				ihm	dj	mll		
ima	2	anglais	ig3d		cge	ihm	fosyma			mll	iamsi	ml	hpc	
ima	3	anglais	ig3d		anum	ml	hpc			ra	flag	dj	iamsi	
ima	4	anglais	ig3d		cge	iamsi	ihm			mll	rital	fosyma	hpc	
ima	5	anglais	ig3d		ra	ml	rital			cge	rp	hpc	fosyma	
ima	6	anglais	ig3d		mll	rital	ml			anum	dj	fosyma	iamsi	
ima	7	anglais	ig3d		mll	rp	rital			cge	iamsi	ihm	hpc	
ima	8	anglais	ig3d		ra	fosyma	ml			anum	dj	iamsi	rital	
ima	9	anglais	ig3d		ra	dj	iamsi			mll	rital	hpc	fosyma	
ima	10	anglais	ig3d		cge	ml	dj			mll	rp	iamsi	flag	
ima	11	anglais	ig3d		mll	iamsi	rp			ra	dj	rital	ihm	
ima	12	anglais	ig3d		mll	hpc	ml			ra	rp	rital	iamsi	
iq	1	anglais	qiintro		anum	flag				pc2r	ig3d			
iq	2	anglais	qiintro		anum	ig3d				sdm	flag			
iq	3	anglais	qiintro		anum	hpc				pc2r	sdm			
iq	4	anglais	qiintro		anum	hpc				sdm	pc2r			
iq	5	anglais	qiintro		ig3d	hpc				flag				
iq	6	anglais	qiintro		hpc	sdm				pc2r				
iq	7	anglais	qiintro		flag	hpc				ig3d				
iq	8	anglais	qiintro		flag	hpc				ig3d				
sar	1	anglais			ar	pnl	srcs	sftr		multi	ioc	fpga		
sar	2	anglais			ar	srcs	pnl	sftr		ioc	multi	fpga		
sar	3	anglais			pnl	sftr	srcs	ar		fpga	ioc	multi		
sar	4	anglais			pnl	srcs	ar	fpga		sftr	ioc	multi		
sar	5	anglais			pnl	srcs	ar	ioc		sftr	multi	fpga		
sar	6	anglais			srcs	sftr	multi	ioc		fpga				
sar	7	anglais			ar	sftr	srcs	pnl		ioc	multi	fpga		
sar	8	anglais			srcs	pnl	ioc	sftr		multi	ar	fpga		
sar	9	anglais			fpga	srcs	multi	ar		sftr				
sar	10	anglais			pnl	sftr	srcs	ar		multi	ioc	fpga		
sar	11	anglais			pnl	srcs	ar	sftr		fpga	ioc	multi		
sar	12	anglais			pnl	ar	sftr	ioc		fpga	multi	srcs		
sar	13	anglais			pnl	srcs	sftr	ar		ioc	fpga	multi		
sar	14	anglais			pnl	sftr	multi	ioc		ar	srcs	fpga		
sar	15	anglais			ar	sftr	srcs	ioc		multi	fpga	pnl		
sar	16	anglais			pnl	sftr	srcs	multi		ar	fpga	ioc		
sar	17	anglais			srcs	sftr	ar	multi		fpga	ioc	pnl		
sar	18	anglais			srcs	pnl	ar	sftr		ioc	multi	fpga		
sar	19	anglais			pnl	srcs	ar	sftr		multi	ioc	fpga		
sar	20	anglais			pnl	srcs	ar	sftr		multi	ioc	fpga		
sar	21	anglais			pnl	srcs	sftr	ar		multi	ioc	fpga		
sar	22	anglais			srcs	pnl	ar	sftr		fpga	ioc	multi		
sar	23	anglais			pnl	ioc	ar	fpga		sftr	srcs	multi		
sar	24	anglais			pnl	ar	srcs	ioc		sftr	fpga	multi		
sar	25	anglais			pnl	srcs	ar	sftr		multi	fpga	ioc		
sar	26	anglais			pnl	srcs	ioc	multi		sftr	ar	fpga		
sar	27	anglais			sftr	ar	srcs	fpga		pnl	multi	ioc		
sar	28	anglais			pnl	ioc	ar	srcs		multi	sftr	fpga		
sar	29	anglais			pnl	srcs	ar	sftr		fpga	ioc	multi		
sar	30	anglais			pnl	ar	sftr	srcs		ioc	fpga	multi		
sesi	1	anglais	cge		pnl	ar	multi	ioc		comnum	fpga			
sesi	2	anglais	cge		fpga	multi	ioc	pnl		comnum	projet			

parcours	num	oblig1	oblig2	oblig3	cons1	cons2	cons3	cons4	cons5	equiv1	equiv2	equiv3	equiv4	equiv5
sesi	3	anglais	cge		ioc	multi	pnl	fpga		comnum	ar			
sesi	4	anglais	cge		projet	multi	fpga	ioc		pnl	ecfa			
sesi	5	anglais	cge		multi	ioc	projet	fpga		ecfa	comnum			
sesi	6	anglais	cge		projet	ecfa	fpga	multi		ioc	comnum			
sesi	7	anglais	cge		projet	ioc	multi	pnl		fpga	ar			
sesi	8	anglais	cge		fpga	multi	projet	ioc		pnl	ar			
sesi	9	anglais	cge		projet	fpga	ioc	multi		pnl	ecfa			
sesi	10	anglais	cge		projet	multi	pnl	fpga		ar	ioc			
sesi	11	anglais	cge		multi	fpga	ioc	pnl		projet	ar			
sesi	12	anglais	cge		multi	fpga	ioc	projet		ecfa	comnum			
sesi	13	anglais	cge		ioc	projet	pnl	fpga		ar	multi			
sesi	14	anglais	cge		ioc	multi	projet	ecfa		comnum	fpga			
sesi	15	anglais	cge		fpga	multi	projet	ar		ioc	pnl			
sesi	16	anglais			multi	rout	projet	pnl		ar	comnum			
sesi	17	anglais	cge		multi	ioc	pnl	projet		ar	fpga			
sesi	18	anglais	cge		multi	ioc	pnl	projet		fpga	ar			
sesi	19	anglais	cge		multi	fpga	ioc	pnl		ar	comnum			
sesi	20	anglais	cge		multi	fpga	ioc	pnl		ar	comnum			
sesi	21	anglais	cge		multi	fpga	ioc	pnl		ar	comnum			
sesi	22	anglais	cge		fpga	pnl	multi	ioc		projet	rout			
sesi	23	anglais	cge		ioc	fpga	comnum	projet		multi	ecfa			
sesi	24	anglais	cge		ioc	multi	projet	ecfa		comnum	fpga			
sesi	25	anglais	cge		ioc	fpga	pnl	multi		projet	ecfa			
sesi	26	anglais	cge		ecfa	comnum	fpga	ioc		multi	projet			
sesi	27	anglais	cge		multi	projet	pnl	fpga		ioc	ar			
sesi	28	anglais	cge		ioc	fpga	projet	multi		rout	pnl			
sesi	29	anglais	cge		multi	fpga	ioc	pnl		ar	comnum			
sfpn	1	anum				flag	hpc	fpga		ml	pnl			
sfpn	2	anglais	anum			flag	hpc	isec		pnl	ml	fpga		
sfpn	3	anglais	anum			flag	hpc	isec		fpga	ml	pnl		
sfpn	4	anglais	anum			isec	flag	hpc		ml	pnl	fpga		
sfpn	5	anglais	anum			flag	hpc	isec		pnl	fpga	ml		
sfpn	6	anglais	anum			hpc	flag	isec		ml	pnl	fpga		
sfpn	7	anglais	anum			isec	flag	hpc		ml	fpga	pnl		
sfpn	8	anglais	anum			isec	hpc	flag		pnl	fpga	ml		
sfpn	9	anglais	anum			isec	flag	hpc		fpga	pnl	ml		
sfpn	10	anglais	anum			flag	hpc	isec		ml	fpga	pnl		
sfpn	11	anglais				hpc				fpga	pnl	ml		
sfpn	12	anglais	anum			isec	flag	hpc		ml	pnl	fpga		
sfpn	13	anglais	anum			hpc	isec	flag		pnl	fpga	ml		
stl	1				ca	paf								
stl	2				paf	pc2r	aps	cpa		ca	cps			
stl	3				paf					cps	cpa			
stl	4				pc2r	cpa	cps	aps		paf	ca			
stl	5				pc2r	cpa	cps	aps		paf	ca			
stl	6				pc2r	cpa	cps	aps		paf	ca			
stl	7				aps	paf				cpa	cps			
stl	8				pc2r	cpa	cps	aps		paf	ca			
stl	9				pc2r	cps	cpa	paf		ca	aps			
stl	10				ca	cps	cpa	paf		pc2r	aps			
stl	11				ca	cps	cpa	paf		pc2r	aps			
stl	12				paf	cpa	pc2r	cps		aps	ca			
stl	13				cps	pc2r	cpa	aps		ca	paf			
stl	14				cps	pc2r	cpa	aps		ca	paf			
stl	15				cpa	cps	ca	paf		aps	pc2r			
stl	16				aps	pc2r	paf			ca	cpa			
stl	17				paf	aps	pc2r			ca	cps			
stl	18				paf	pc2r	aps	ca		cps	cpa			
stl	19				cpa	cps	paf	ca		pc2r	aps			
stl	20				cpa	cps	paf	ca		pc2r	aps			
stl	21				paf	pc2r	aps	cpa		ca	cps			
stl	22				cpa	cps	ca	paf		pc2r	aps			
stl	23				pc2r	paf				cps	ca			
stl	24				ca	cpa	paf	aps	pc2r	cps				
stl	25				paf	cpa	cps	pc2r		aps	ca			
stl	26				pc2r	paf				aps	ca			
stl	27				paf	pc2r	aps	cps		cpa	ca			
stl	28				ca	paf	aps	cpa		pc2r	cps			
stl	29				cps	pc2r	cpa	aps		ca	paf			
stl	30				paf	aps	pc2r	ca		cpa	cps			
stl	31				cps	pc2r	paf	cpa		aps	ca			
stl	32				cps	pc2r	cpa	ca		aps	paf			
stl	33				cps	pc2r	cpa	ca		aps	paf			
stl	34				pc2r	cps	paf	aps		cpa	ca			
stl	35				cps	cpa	ca	aps		pc2r	paf			
stl	36				aps	cps	pc2r	ca		paf	cpa			
stl	37				cps	pc2r	cpa	ca		paf	aps			
stl	38				pc2r	cps	cpa	paf		aps	ca			
stl	39				paf	pc2r	cpa	cps		aps	ca			
stl	40				paf	pc2r	cps			aps	cpa			
stl	41				paf	pc2r	aps	cps		ca	cpa			

6.3.2 Fichier de remplissage des groupes

id_ue	intitule	nb_groupes	capac1	capac2	capac3	capac4	capac5	cours1	cours2	td1	tme1	td2	tme2	td3	tme3	td4	tme4	td5	tme5
1	anglais	1	500					1		2									
2	anglais	1	500					13		14									
3	anum	1	32					11		12									
4	aps	1	32					14		11	12								
5	ar	1	32					7		3	4								
6	bium	1	32					3		11	12								
7	bmc	1	32					23		24	7								
8	ca	1	32					1		6	7								
9	cge	1	32					-1											
10	comnum	1	32					22		23	24								
11	cpa	1	32					22		23	24								
12	cps	1	32					9		3	4								
13	dj	2	32	32				3		16	17	21	22						
14	ecfa	1	32					4	17	1	2								
15	flag	1	32					21		1	2								
16	fosyma	1	32					4		6	7								
17	fpga	3	20	8	8			9		3	4	6	7	6	7				
18	hpc	1	32					22		6	7								
19	iamsi	1	32					13		6	7								
20	ig3d	1	32					9		18	19								
21	ihm	1	32					9		11	12								
22	ioc	1	40					16		21	22								
23	isec	1	32					16		23	24								
24	ml	2	32	32				14		6	7	16	17						
25	mll	1	32					-1		11	12								
26	mmcn	1	32					17		21	22								
27	multi	1	40					12		23	24								
28	paf	1	32					13		18	19								
29	pc2r	1	32					2		16	17								
30	pnl	1	40					6		18	19								
31	qiintro	1	32					13		23	24								
32	ra	1	50					14		18	19								
33	rital	1	32					4		18	19								
34	rout	1	32					1		18	19								
35	rp	1	32					23		16	17								
36	sam	1	32					9		21	22								
37	sbas	1	32					6		3	4								
38	sdm	1	32					6		7	8								
39	sftr	1	32					11		16	17								
40	srcs	1	32					12		23	24								
41	projet	1	500					-1											

6.4 Bibliographie

Références

[1] AMOUSSOU, M., DIALLO, M., AND KHEIREDDINE, A. Projet ANDROIDE Optimisation des inscriptions aux UE, 2018.

[2] GUROBI OPTIMIZATION, LLC. Gurobi Optimizer Reference Manual, 2022.