

# Algorithmes – Emploi du temps

Coudray – Julien – Tran

26 mars 2014

## 1 Pré-traitements

Avant de réaliser l’emploi du temps, nous procédons à des vérifications sur les données d’entrées afin de détecter toutes les incohérences. Ainsi, nous éliminons au préalable une partie des traitements qui n’aboutiront pas.

### 1.1 Le nombre de professeurs

La première vérification concerne le nombre de professeurs en entrée. Nous vérifions qu’il y a assez de professeurs pour dispenser les cours de chaque classe. Ainsi pour un cours donné, l’algorithme somme les disponibilités des professeurs puis compare le résultat au nombre de classe, en tenant compte du fait que pour des cours de 4h, le nombre de disponibilités nécessaire est doublé.

Soit  $n$  le nombre de professeurs pouvant donner un cours  $c$  et  $m$  le nombre de classe devant suivre ce cours. Pour un cours de 2 heures, nous avons :

$$\sum_{i=0}^n \text{dispo}_{\text{prof}_i} > m$$

Pour un cours de 4 heures, nous avons :

$$\frac{\sum_{i=0}^n \text{dispo}_{\text{prof}_i}}{2} > m$$

Pour un cours dispensé sur 4h, le nombre de disponibilités nécessaire est doublé. Pour simplifier nos calculs, nous divisons par 2 le nombre de disponibilités trouvées pour un professeur pour le comparer aux disponibilités nécessaires pour dispenser le cours.

L’opération est répétée pour l’ensemble des cours.

---

**Algorithme 1 : Pré-traitement nombre de professeurs**

---

```
for all Cours do
  idCours ← identifiant de Cours
  idPromo ← identifiant de la promotion recevant Cours
  nbClasses ← nombre de classe de la promotion idPromo
  for all Profs do
    if Profs donne le cours idCours then
      for all CreneauxProf do
        if Profs est disponible then
          nbCreneaux ← nbCreneaux + 1
        end if
      end for
    end if
  end for
  if Cours est sur 4h then
    nbCreneaux ← nbCreneaux/2
  end if
  if nbClasses > nbCreneaux then
    display (Erreur sur le nombre de professeurs pour la promo idPromo)
    EXIT FAILURE
  end if
end for
display (Nombre de professeurs ok)
```

---

## 1.2 Le nombre de cours total sur le semestre

La seconde vérification porte sur le nombre d'heures de cours à dispenser à une classe. Ce nombre ne doit pas excéder la totalité des heures du semestre. Le programme somme l'ensemble des cours que possède une classe et le compare au nombre d'heures total du semestre.

$$\sum_{i=0}^n nbHeures_{cours_i} \leq s * c * h$$

Avec :

- $n$  le nombre de cours d'une classe
- $p$  la classe considérée
- $s$  le nombre de semaines du semestre
- $c$  le nombre de créneaux sur une semaine
- $h$  le nombre d'heures d'un créneau

Une fois ces pré-traitements réalisés, nous pouvons commencer la conception de l'emploi du temps.

---

**Algorithme 2 : Pré-traitement nombre d'heures sur le semestre**

---

```
for all Classes do  
    listCours  $\leftarrow$  ensemble des cours que suit une classe  
    for all cours in listCours do  
        nbHours  $\leftarrow$  nbHours + nombre d'heures du cours cours  
    end for  
    if nbHours > (nombre de semaines du semestre * nombre de créneaux par semaine *  
    nombre d'heures par créneau) then  
        display(Erreur, trop d'heures pour la classe Classes)  
        EXIT FAILURE  
    end if  
end for  
display(Nombre d'heures de cours ok)
```

---

## 2 Réalisation de l'emploi du temps

Il est difficile de trouver un ordonnancement en résolution exacte, donc pour faciliter la génération de l'emploi du temps, nous avons mis en place une fonction de répartition aléatoire des cours dans la semaine. Cette fonction est rappelée un certain nombre de fois (nombre d'itérations à définir), mais le programme s'arrête si une solution est trouvée avant la dernière itération.

Afin de clarifier l'explication, nous rappelons que le programme de l'année est un ensemble de matières (Algèbre, Analyse, Electricité, etc.), et chaque matière est un ensemble de cours (Algèbre : 14 cours de 2h, Analyse 12 cours de 2h, etc.), répartis sur les créneaux de la semaine. De plus, chaque promotion (B1, B2, B3, etc.) est constituée d'un certain nombre de classes (B1A, B2C, M1B, etc.). Enfin, un créneau correspond à 2h dans la semaine (Lundi 8h30-10h30, Mercredi 14h-16h, etc.), un cours de 2h se place donc sur un créneau, et un cours de 4h sur 2 créneaux.

Afin d'optimiser la répartition des cours dans la semaine, nous répartissons au préalable les matières du programme, et les enseignants correspondant sur le semestre, et ce pour chaque promotion. Ainsi, chaque classe d'une même promotion suivra les mêmes cours chaque semaine, ce qui garantit l'homogénéité de l'emploi du temps.

Pour se faire, nous déterminons la semaine à laquelle doit se dérouler le premier cours de chaque matière, ainsi que le nombre de semaines nécessaires. Nous pouvons alors répartir les cours sur chaque semaine à l'aide de la fonction de répartition aléatoire. Cette fonction sera rappelée jusqu'à trouver une solution dans laquelle tous les cours ont été placés correctement. En cas d'échec (aucune solution parfaite), le programme donnera la solution dans laquelle un minimum de cours n'ont pas été placés correctement. Ces cours pourront toutefois être placés manuellement par la suite.

---

**Algorithme 3 : Principe général de conception des emplois du temps**

---

```
meilleurEDT  $\leftarrow \infty$ 
repeat
  for all Promo do
    idMatières  $\leftarrow$  liste contenant les id des matières que doivent suivre Promo
    programmeSemestre  $\leftarrow$  liste contenant la répartition des idMatières sur le semestre
    planningOk  $\leftarrow$  booléen indiquant si la génération de l'emploi du temps a rencontré des erreurs
  end for
  if planningOk then
    if nbCoursNonPlaces < meilleurEDT then
      meilleurEDT  $\leftarrow$  nbCoursNonPlaces
      Ecriture des emplois du temps dans les fichiers
    end if
  end if
until meilleurEDT > 0 and cmpt < nombreIteration
```

---

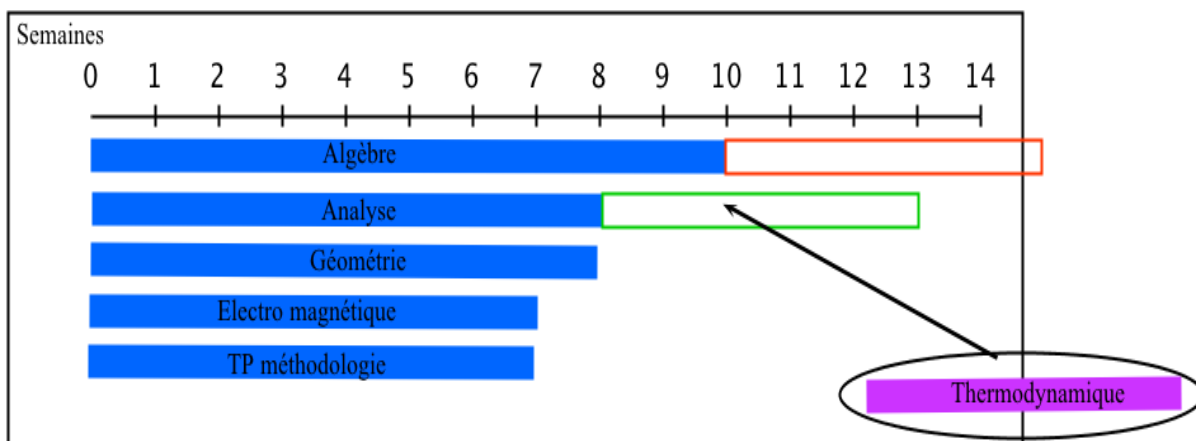
## 2.1 Répartition du programme sur le semestre

Afin d'optimiser la répartition des cours sur chaque semaine, nous commençons par répartir les matières sur les semaines du semestre. Pour chaque matière, nous allons donc indiquer la semaine dans laquelle doit commencer le premier cours, ainsi que le nombre de semaines durant lequel elle va être enseignée.

La répartition se déroule en deux étapes :

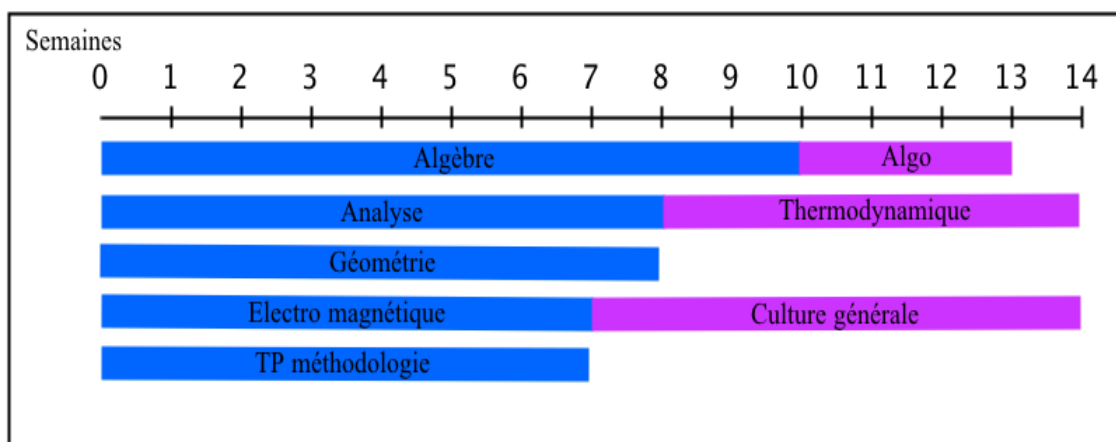
- Le tri des matières
- Le placement des matières sur le semestre

L'objectif est de répartir les cours sur le semestre de manière homogène. Il faut donc réussir à placer le maximum de matières les unes à la suite des autres. Pour se faire, nous trions les matières de la plus longue à la plus courte (en nombre de semaines), puis nous les plaçons les unes après les autres dans le semestre, en faisant en sorte de les placer à la suite d'une autre matière dès que possible.



Un cours de 4 heures impose plus de contraintes. En effet, il s'agit d'un cours où le professeur et la classe doivent avoir en commun deux créneaux libres consécutifs dans la même demi-journée. C'est pourquoi un cours de 4 heures doit être planifié sur le semestre avant un cours de 2 heures.

Pour se faire, les matières vont être séparées en deux listes : une pour les cours de 4 heures et une autre pour les cours de 2 heures, et nous effectuons les répartitions sur le semestre comme expliqué précédemment.



Pour chaque matière, nous avons besoin des informations suivantes :

- L'identifiant de la matière
- Le numéro de la semaine dans laquelle sera donné premier cours
- Le nombre de semaines nécessaires pour enseigner l'ensemble de la matière
- La matière qui la suit, s'il y en a une

---

**Algorithme 4 :** Algorithme principal de la répartition des matières sur le semestre

---

**Require:** liste *idMatières*, liste *programmeSemestre*

```
for all idMatières do
  if idMatières est un cours sur 2h then
    idMatières2  $\leftarrow$  pushback idMatières
  else
    idMatières4  $\leftarrow$  pushback idMatières
  end if
end for
Tri de idMatières2 par nombre de semaines de matière décroissant
Tri de idMatières4 par nombre de semaines de matière décroissant
idMatières est vidé
for all idMatières4 do
  idMatières  $\leftarrow$  pushback idMatières4
end for
for all idMatières2 do
  idMatières  $\leftarrow$  pushback idMatières2
end for
return programmeSemestre  $\leftarrow$  repartitionDesCours(idMatières)
```

---

---

**Algorithme 5 :** repartitionDesMatières(*idMatières*)

---

**Require:** liste *idMatières* triée par nombre de semaines d'une matière et par cours de 4h et 2h

```
initialisation de programmeSemestre
for all idMatières do
  for all Matières in programmeSemestre do
    if Matières a été placé then
      checkNextCourse(idCourses, Matières)
      if idMatières a été programmé then
        coursPlace  $\leftarrow$  true
        BREAK
      end if
    end if
  end for
  if coursPlace == false then
    programmeSemester  $\leftarrow$  pushback idMatières en le programmant en début de semestre
  end if
end for
return programmeSemestre
```

---

---

**Algorithme 6 : checkNextCourse(*idMatières*, *Matières*)**

---

```
if Matières a une autre matière après lui then
    checkNextCourse(idMatières, Matières du Matières suivant)
else if  $semaineDebut_{coursProgrammes} + nbSemaine_{coursProgramme} + nbSemaine_{idMatières} \leq nbSemaine_{semestre}$  then
    programmeSemestre  $\leftarrow$  pushBack idMatières en le programmant après le cours Matières
end if
```

---

Après avoir réalisé cet emploi du temps, nous pouvons commencer à placer les cours sur les créneaux des classes concernées.

## 2.2 Répartition des cours sur leurs créneaux

A partir de l'emploi du temps du semestre, nous allons planifier les cours sur les créneaux des classes. A la fin de chaque semaine, les classes d'une même promotion doivent être au même point du programme. Ainsi, l'emploi du temps est réalisé en parallèle pour chaque promotion semaine par semaine.

Pour chacune des semaines, nous allons récupérer la liste des cours à dispenser depuis l'emploi du temps semestriel. A partir des semaines de début et de fin de cours, nous en déduisons s'il doit être donné sur cette semaine.

Avec ce programme, nous allons pouvoir commencer à placer les cours sur les différents créneaux des classes. Cela va être fait en trois étapes :

- Si le cours a déjà été placé à la semaine précédente
- La récupération de la liste des professeurs pouvant enseigner les cours de la semaine
- Le placement des nouveaux cours du semestre

Dans le cas où un cours n'a pu être placé faute de créneaux disponibles, nous avons décidé de le mettre dans une liste contenant l'ensemble des cours non placés et de ne pas tenter de le replacer sur les semaines suivantes. Ainsi cette liste contiendra l'identifiant du cours, l'identifiant du professeur et les semaines où le cours n'a pu être placé.

### 2.2.1 Placement de cours déjà fixé la semaine précédente

Après avoir récupéré le programme de la semaine, nous vérifions si l'un des cours a déjà été placé la semaine précédente. Si c'est le cas, nous allons vérifier que le professeur ayant donné le cours est toujours disponible sur le créneau et nous plaçons le cours.

Dans le cas où le cours a pu être redonner pour toutes les classes de la promotion, nous pouvons supprimer le cours dans le programme de la semaine. Sinon, soit il s'agit d'un nouveau cours, soit toutes les classes ne l'ont pas reçu. Ce dernier cas arrive lorsqu'un professeur n'est plus disponible ou quand le cours n'a pu être placé pour toutes les classes de la promotion.

Ceci va permettre à une classe d'avoir le même cours sur le même créneaux avec le même professeur semaine après semaine.

---

**Algorithme 7 :** Algorithme principal de la répartition des cours sur les créneaux des classes

---

**Require:** le programme du semestre *prog*  
**for all** *semaine* du semestre **do**  
    *progSemaine*  $\leftarrow$  getProgrammeSemaine(*prog*, *semaine*)  
    placementAncienCours(*progSemaine*, *listeClasses*, *semaine*)  
    **if** il y a des cours à placer encore dans la semaine **then**  
        *profSemaine*  $\leftarrow$  getProfSemaine(*progSemaine*)  
        placementNouveauCours(*listeClasses*, *progSemaine*, *profSemaine*, *semaine*)  
    **end if**  
    **if** une erreur est survenu dans la réalisation du planning **then**  
        **return** 0  
    **end if**  
**end for**  
**return** 1

---

---

**Algorithme 8 :** Méthode pour récupérer le programme d'une semaine

---

**Require:** le programme du semestre *prog* et la semaine du semestre *semaine*  
**for all** *cours* du programme **do**  
    **if** semaineDebut de *cours*  $\leq$  *semaine* **and** semaine fin de *cours*  $>$  *semaine* **then**  
        *progSemaine*  $\leftarrow$  pushback *cours*  
    **end if**  
**end for**  
**return** *progSemaine*

---

---

**Algorithme 9 :** Méthode pour placer les cours précédemment planifier

---

**Require:** le programme de la semaine *prog*, la liste des classes *classes*, la semaine du semestre *semaine*  
*nbCourseAjout*  $\leftarrow$  0  
*nouveauCours*  $\leftarrow$  *false*  
**if** La première semaine à déjà été planifié **then**  
    **for all** *cours* du programme de la semaine **do**  
        coursDejaProgrammeAvant(*cours*, *classes*, *nbCoursAjout*, *nouveauCours*)  
        **if** *nbCoursAjout* = nombre *classes* **then**  
            *coursASupprimer*  $\leftarrow$  pushback *cours*  
        **else**  
            *nouveauCours*  $\leftarrow$  *faux*  
        **end if**  
    *nbCourseAjout*  $\leftarrow$  0  
    **end for**  
    **for all** *coursASupprimer* **do**  
        *progSemestre*  $\leftarrow$  supprimer *progSemestre*(*coursASupprimer*)  
    **end for**  
**end if**

---



---

**Algorithme 10 :** Méthode pour savoir si un cours a déjà été programmé avant

---

**Require:** le cours de la semaine *cours*, la liste des classe *classes* , la semaine du semestre *semaine*

```
for all classes do
  if classes a reçu le cours la semaine semaine – 1 then
    ajoutDuCours(classes, cours, semaine)
    nbCoursAjoute ← nbCoursAjoute + 1
  end if
end for
if nbCoursAjoute = nombre de classes then
  nouveauCours ← faux
else
  nouveauCours ← vrai
end if
```

---

---

**Algorithme 11 :** Méthode pour ajouter le cours par rapport à la semaine d'avant

---

**Require:** la classe *classe*, la matière *cours*, la semaine du semestre *semaine*

```
idProf ← identifiant du professeur donnant cours la semaine – 1
creneau ← créneau de cours la semaine – 1
if cours est sur 4 heures then
  if prof est disponible à semaine, creneau and prof est disponible
  semaine, creneau + 1 and classe est disponible à semaine, creneau and classe est
  disponible semaine, creneau + 1 then
    planification cours avec prof sur semaine et creneau
    planification cours avec prof sur semaine et creneau + 1
  end if
else
  if prof est disponible à semaine, creneau and prof est disponible semaine, creneau + 1
  then
    planification cours avec prof sur semaine et creneau
  end if
end if
```

---

### 2.2.2 Planification des nouveaux cours du semestre

A partir du programme du semestre, nous récupérons l'ensemble des professeurs pouvant enseigner la liste des matières. Tous les professeurs ne se verront pas forcément attribuer un cours car plusieurs professeurs peuvent enseigner le même cours.

Pour tous les cours restant à planifier, nous allons à chaque fois trouver le couple promotion-professeur ayant le moins de créneaux en communs. En effet si nous plaçons des couples ayant plus de disponibilités avant un couple qui en a moins, il pourrait bloquer l'ensemble des disponibilités de ce dernier.

Ensuite, nous sélectionnons un créneaux aléatoirement parmi les choix possibles pour placer un cours. Nous réalisons un tirage aléatoire pour pouvoir essayer plusieurs combinaisons.

Dans le cas d'un cours de 4 heures, il se peut que nous n'ayons aucun créneau permettant de mettre les 4 heures à la suite. Dans ce cas nous mettons le cours dans la liste des cours n'ayant pu être planifié. Auquel cas, nous plaçons le cours de la classe sur le créneau en modifiant les disponibilités du professeurs.

---

**Algorithme 12 : Méthode pour ajouter un nouveau cours**

---

**Require:** *progSemaine, profSemaine, listClasses*  
*nbCours*  $\leftarrow$  nombre de cours dans *progSemaine* \* nombre de classe dans *listClasses*  
**for** *i* := 0 **to** *nbCours* **do**  
    *meilleurConnexion*(*progSemaine, profSemaine, listClasses, semaine*)  
    **if** on trouve une connexion **then**  
        *ajoutCours*(*progSemaine, profAAjouter, classesAAjouter, semaine*)  
    **else**  
        **return** 0  
    **end if**  
**end for**  
**return** 1

---

---

**Algorithme 13 : Méthode pour trouver la plus meilleur connexion**

---

**Require:** *profSemaine, profSemaine, listClasses, semaine*  
*buf*  $\leftarrow$  23  
**for all** *profSemaine* **do**  
  **for all** *listClasses* **do**  
    *nbConnections*  $\leftarrow$  nbCreneauCommun(*profSemaine, listClasses, semaine*)  
    **if** *nbConnections* > 0 **and** *nbConnections* < *buf* **then**  
      *buf*  $\leftarrow$  *nbConnections*  
      *profAAjouter*  $\leftarrow$  *profSemaine*  
      *promoAAjouter*  $\leftarrow$  *listClasses*  
    **end if**  
  **end for**  
**end for**

---

---

**Algorithme 14 : Méthode pour compter le nombre de connection**

---

**Require:** *prof, classe, semaine, progSemaine*)  
**for all** *cours* donnés par *prof* **do**  
  **if** *promo* doit recevoir *cours* sur *semaine* **and** *cours* n'a pas encore été placé pour *promo* sur *semaine* **then**  
    *coursPossible*  $\leftarrow$  vrai  
    BREAK  
  **else**  
    *coursPossible*  $\leftarrow$  faux  
  **end if**  
**end for**  
**if** *coursPossible* **then**  
  **return** *nbConnection*  $\leftarrow$  somme des disponibilité commune de *prof* et *promo*  
**end if**  
**return** -1

---

---

**Algorithme 15 :** Méthode pour ajouter un cours à une classe

---

**Require:** *progSemaine, profAAjouter, classesAAjouter, semaine*

**for all** *cours* de *profAAjouter* **do**

**if** *promo* doit recevoir *cours* sur *semaine* **and** *cours* n'a pas encore été placé pour *promo* sur *semaine* **then**

*creationCours*(*prof*, *promo*, *cours*, *semaine*)

        BREAK

**end if**

**end for**

---

---

**Algorithme 16 : Méthode pour créer le cours à la classe**

---

```
if cours n'a pas été programmé à semaine – 1 pour classes then
    ajout de cours dans la liste des cours non planifié
else if cours est sur 4 heures then
    for all creneau do
        if classe est libre à semaine, creneau and classe est libre à semaine, creneau + 1 and
        prof est libre à semaine, creneau and prof est libre à semaine, creneau + 1 then
            creneauxPossibles ← pushback creneau
        end if
    end for
    if creneauxPossibles n'est pas vide then
        creneau ← choix aléatoire dans creneauxPossibles
        mise en place du cours et des données (cours, classe, prof, semaine, creneau)
        mise en place du cours et des données (cours, classe, prof, semaine, creneau + 1)
    end if
    ajout de cours dans la liste des cours non planifié
else
    for all creneau do
        if classe est libre à semaine, creneau and prof est libre à semaine, creneau then
            creneauxPossibles ← pushback creneau
        end if
        creneau ← choix aléatoire dans creneauxPossibles
        mise en place du cours et des données (cours, classe, prof, semaine, creneau)
    end for
end if
```

---

### 3 Déplacements des cours

Une fois l'emploi du temps réalisé, nous avons la possibilité de déplacer des cours ou de placer manuellement les cours qui n'ont pu être placé par le programme.

#### 3.1 Déplacement d'un cours existant

Dans un premier, il faut choisir la classe pour laquelle nous souhaitons effectuer le changement. Ensuite on sélectionne le cours puis une liste de créneaux possible apparait en fonction de la classe et du professeur. En sélectionnant le créneau, nous allons effectuer les modification de planning de la classe et du professeur.

#### 3.2 Ajout d'un cours non placé

Lorsque nous voulons ajouter un cours non placé, la liste de ceux-ci va apparaitre. Nous sélectionnons le cours, puis comme précédemment on sélectionne un créneaux possible pour placer ce cours. Nous effectuons enfin les modifications sur la classe et le professeur.