# Algorithmes – Emploi du temps

Coudray – Julien – Tran

17 mars 2014

## 1 Pré-traitements

Avant de commencer à réaliser l'emploi du temps proprement parler, il faut vérifier que nos données d'entrées ne vont pas nous mener vers une non solution. Ces pré-traitements vont porter sur le nombre de professeur et la quantité des cours sur le semestre.

### 1.1 Le nombre de professeur

La première vérification va concerner le nombre de professeur en entrée. La question qu'il faut se poser est : il y a t'il assez de professeur pour donner chaque cours à toutes les promotions concernées ? Le principe de cet algorithme est de faire la somme des disponibilités de l'ensemble des professeurs pouvant donner un cours. Cette somme est ensuite comparée au nombre de classes qui va suivre ce cours.
Si le cours est sur 4 heures, alors la somme des disponibilités sera divisée par 2, car il va nécessiter 2 créneaux consécutifs dans la semaine pour donner le cours.

Soit $n$ le nombre de professeur pouvant donner un cour $c$ et $m$ le nombre de classe devant suivre ce cours. Nous devons avoir pour un cours de 2 heures :

$$\sum_{i=0}^{n} dispo_{prof_i} > m$$

Dans le cas d'un cours de 4 heures :

$$\frac{\sum_{i=0}^{n} dispo_{prof_i}}{2} > m$$

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

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

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

## 1.2 Le nombre de cours total sur le semestre

La seconde vérification va porter sur le nombre d'heure de cours d'une classe sur le semestre. Ce nombre ne doit pas excéder la totalité des heures du semestre. Le principe est de faire la somme de l'ensemble des cours que va avoir une classe et de la comparer au nombre d'heure du semestre.

Soit $n$ le nombre de cours d'une classe $p$, $s$ le nombre de semaine du semestre, $c$ le nombre de créneau sur une semaine et $h$ le nombre d'heure d'un créneau :

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

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

```
for all Classes do
    listCourses ← ensemble des cours que suit une classe
    for all courses in listCourses do
        nbHours ← nbHours+ nombre d'heure du cours courses
    end for
    if nbHours > (nombre de semaine 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'heure de cours ok)
```

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

# 2 Réalisation de l'emploi du temps

La réalisation de l'emploi du temps se déroule en deux étapes. Elles seront réalisées pour chaque promotion indépendamment. Chaque classe d'une promotion va suivre le même programme avec la même liste d'enseignants. C'est pourquoi, chaque promotion va avoir un emploi du temps sur le semestre. A savoir, la planification des cours indiquant la semaine de début et de fin. Enfin, un emploi du temps final où sont placer les cours sur leurs créneaux. Il sera réaliser à l'aide du planning du semestre en le créant semaine par semaine.

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

**for all** *Promo* **do**
  $idCourses \leftarrow$ liste de tous les cours que doivent suivre la promotion *Promo*
  repartitionCoursSemestre($idCourses$, $programmeSemestre$)
  repartitionCoursPromotions(*Promo*, $programmeSemestre$)
**end for**

## 2.1 Répartition du programme sur le semestre

La première étape de l'algorithme de résolution de planification de l'emploi du temps est de répartir de l'ensemble du programme de la promotion sur le semestre. Cette étape consiste à indiquer pour chaque cours la date de début et de fin semaine.

La répartition se déroule en deux étapes :
— Le trie des cours
— Le placement des cours sur le semestre

L'objectif est de répartir au mieux les cours sur le semestre. Il faut réussir à placer le maximum de cours les uns à la suite des autres. Les cours les plus longs sont donc placés en premier. Puis pour chaque nouveau cours, il faut vérifier si il est possible de pouvoir le débuter après un cours déjà placé. Si ce n'est pas le cas, le cours sera placé en début de semestre.

Un cours de 4 heures impose plus de contraintes. Il s'agit d'un cours où le professeur et la classe doivent avoir deux créneaux consécutifs libre et sans interruption (fin de journée, pause de midi). C'est pourquoi un cours de 4 heures doit être planifié sur le semestre avant un cours de 2 heures.

Les cours vont tout d'abord être séparés en deux : les cours de 4 heures et les cours de 2 heures. Ensuite pour chacun des groupes un trie est effectué de manière décroissante par rapport à la durée d'un cours en nombre de semaine.

Pour planifier les cours sur le semestre, la liste des cours triés de 4 heures puis de 2 heures vont être parcouru. Pour chaque nouveau coures à placer, il faut vérifier si il est possible de l'intégrer après un cours déjà planifié. Il est possible de le faire uniquement si la durée du nouveau cours ne va pas dépasser la fin du semestre. Auquel cas le cours sera placé en début de semestre.

$$semaineDebut_{coursPlace} + nbSemaine_{coursPlace} + nbSemaine_{nouveauCours} \leq nbSemaine_{semestre}$$

Chaque élément du semestre va avoir les informations suivantes :
— L'identifiant du cours
— Le numéro du début de la semaine
— Le nombre de semaine du cours
— Le cours qui le suit

**Algorithme 4 :** Algorithme principale de la répartition des cours sur le semestre

**Require:** liste *idCourse*, liste *programmeSemestre*
  **for all** *idCourse* **do**
    **if** *idCourse* est un cours sur 2h **then**
      *idCourses*2 ← pushback *idCourses*
    **else**
      *idCourses*4 ← pushback *idCourses*
    **end if**
  **end for**
  Trie de *idCourses*2 par nombre de semaine de cours décroissant
  Trie de *idCourses*4 par nombre de semaine de cours décroissant
  *idCourses* est vider
  *idCourses* ← pushback *idCourses*4
  *idCourses* ← pushback *idCourses*2
  *programmeSemestre* ← repartitionDesCours(*idCourses*)

**Algorithme 5 :** repartitionDesCours($idCourses$)

**Require:** liste $idCourses$ triée par nombre de semaine d'un cours et par cours de 4H et 2H
initialisation de $coursProgrammes$
**for all** $idCourses$ **do**
  **for all** $coursProgrammes$ **do**
    **if** $programmeSemestre$ a cours placé **then**
      checkNextCourse($idCourses$, $coursProgrammes$)
      **if** $idCourses$ a été programmé **then**
        $coursPlace \leftarrow true$
        BREAK
      **end if**
    **end if**
  **end for**
  **if** $coursPlace == false$ **then**
    $coursProgrammes \leftarrow$ pushback $idCourses$ en début de semestre
  **end if**
**end for**
**return** $coursProgrammes$

**Algorithme 6 :** checkNextCourse($idCourses$, $coursProgrammes$)

**if** $coursProgrammes$ a un cours après lui déja **then**
  checkNextCourse($idCourses$, $coursProgrammes$ du cours suivant)
**else if** $semaineDebut_{coursProgrammes} + nbSemaine_{coursProgramme} + nbSemaine_{idCourses} \leq nbSemaine_{semestre}$ **then**
  $coursProgramme \leftarrow$ pushBack $idCourses$
**end if**