Algorithmes – Emploi du temps

Coudray – Julien – Tran

3 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 directement vers une non solution.

1.1 Le nombre de professeur

La première vérification va concerner le nombre de professeur que nous avons en entrée. La question qu'il faut se poser est : il y a t'il assez professeur pour donner chaque cours à chaque promotion ? Le principe de cette algorithme est de faire la somme des créneaux de disponibilités de l'ensemble des profs pouvant donner un cours et le comparer avec le nombre de classes qui va devoir suivre ce cours.

Algorithme 1 : Pré-traitement nombre de professeurs

```
for all Courses do
  idCourse ← id_course[Courses]
  idPromo ← id_promo[Courses]
  nbClasses ← nombre de classe de la promo idPromo
  for all Profs do
    if Profs donne le cours idCourse then
      for all ProfSlots do
        if Profs est displonnible then
          nbSlots ← nbSlots + 1
        end if
      end for
    end if
  end for
  if Courses est sur 4h then
    nbSlots ← nbSlots/2
  end if
  if nbClasses > nbSlots 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 total que va suivre une classe sur l'ensemble du semestre. Ce nombre d'heure ne doit pas excéder le nombre d'heure maximum par rapport au nombre total de créneaux du semestre. Le principe est de faire la somme de l'ensemble des cours que va avoir la classe et la comparer au nombre d'heure maximum du semestre.

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 réalisé, nous pouvons commencer à réaliser l'emploi du temps de l'école.

2 Réalisation de l'emploi du temps

La réalisation de l'emploi du temps se déroule en plusieurs étapes. Il sera réalisé pour chaque promotion indépendamment. Une promotion ayant plusieurs classes qui vont suivre le même programme sur l'ensemble du semestre, avec le même nombre d'heure et la même liste de professeur pouvant enseigner la liste des matières. Puis pour chaque promotion, réaliser un emploi du temps sur le semestre. A savoir planifier l'ensemble des cours avec une semaine de départ et une semaine de fin. Enfin, placer les cours sur les créneaux en parcourant semaines par semaines sur l'ensemble des classes simultanément.

Algorithme 3 : Principe général de conception de l'emploi du temps

```
for all IdPromo do
    for all Classes do
        if idPromo de Classes = IdPromo then
            idCourses ← liste de tous les cours que doivent suivre la promotion IdPromo
            BREAK
        end if
    end for
    repartitionCoursSemestre(idCourses, programmeSemestre)
    repartitionCoursPromotions(IdPromo, 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 de l'école est à partir de l'ensemble du programme de la promo de placer les cours à une date de début et de fin sur le semestre. Dans un premier temps il va falloir donner un trie d'insertion des cours sur le programme du semestre. Il faut séparer les cours de deux et quatre heures, puis les trier dans l'ordre décroissant du nombre de semaines sur lequel va se dérouler le cours. Ensuite la répartition des cours sur le semestre en essayant toujours de placer un cours à la suite d'un fini. Dans le cas où il est impossible, le cours débutera au début du semestre.

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*)