Rapport de programmation par contraintes

Modélisation d'un problème d'emplois du temps

Pascal Brunot
Hadrien Cambazard

Sommaire

1. Données d'entrée

1.1. Pré-traitement

On calcule, par rapport aux informations d'entrées du problème fournies, un certain nombre de nouvelles données :

- **Nc** : le nombre de cours,
- **Ns** : le nombre de salles,
- **Ne** : le nombre d'étudiants,
- $LC_k = \{C_1,...,C_n\}$ **la liste des numéros des cours suivis par l'étudiant n°k**. les C_i ayant pour valeur les numéros des cours suivis. Calculable en $\mathrm{O}(Ne \cdot Nc)$
- ***Effectif(Ci)*** = nombre d'étudiants qui sont inscrits au cours Ci. On pose également Effectif(0)=0. Calculable en $\mathrm{O}(Nc)$.
- **La liste des salles possibles pour chaque cours** $LS_k = \{\delta_1,...,\delta_{Ns}\}$ (i.e les salles suffisamment grande par rapport au nombre d'étudiant suivant le cours et possédant les équipements requis pour ce cours). δ_i vaut 1 si la salle n°i est une salle possible pour le cours n°k. On obtient cette information associée à chaque cours par prétraitement sur les données. Calculable en $O(N_C \cdot N_S \cdot N_{équipements})$

La complexité de la phase de pré-traitement reste négligeable par rapport au reste du problème, nous ne déplaçons pas la difficulté vers le pré-traitement en calculant ces données.

1.2. Conventions

On numérote les journées de 1 à 5.

On numérote les créneaux de 1 à 45.

On utilisera des créneaux relatifs à une journée *j*: ils sont numérotés de 1 à 9. On passe donc d'un créneau relatif à un créneau absolu par la formule CreneauAbsolu = 9*(Journée-1)+CreneauRelatif.

On modélisera les contraintes de préférences en les considérant comme une fonction objectif à minimiser. On sait en effet que le problème n'est pas sur-contraint puisqu'il existe une solution satisfaisant les 6 contraintes. On pourrait donc modéliser ces contraintes comme étant des contraintes à part entière, mais au vu de leur difficulté on préfère les considérer comme des éléments d'une fonction objectif à minimiser. Ceci est pertinent au regard de leur sens : ce sont des contraintes de convenance pour les étudiants mais qui ne remettent pas en question la viabilité du planning.

2. Première modélisation

2.1. Variables

On choisit 2 jeux de variables qui correspondent au créneau et à la salle de chaque cours.

- $\forall i \in [1, Nc], salle_i$: indique la salle affectée au cours n°i. $salle_i$ prend ses valeurs dans le domaine D = [1..Ns].

- $\forall i \in [1, Nc], creneau_i$: indique le créneau affecté au cours n°i. $creneau_i$ prend ses valeurs dans le domaine D = [1..45].

Une solution du problème est caractérisée par l'affectation de chaque cours dans un créneau et une salle. On peut donc voir ces variables comme une seule série de variables $cours_i$ représentées par les couples ($salle_i$, $creneau_i$). $cours_i$ prendrai alors ses valeurs dans le domaine [1..45] × [1..Ns]

2.2. Contraintes dures

Contrainte 1 : Tous les cours suivis par un étudiant doivent être placés dans des créneaux différents. Il s'agit d'une contrainte globale qui impose à toutes les variables $creneau_i$ associées aux cours suivis par un même étudiant de prendre des valeurs différentes.

$$\forall e \in [1, Ne],\ allDifferent(\{creneau_{Ci} | C_i \in LC_e\})$$

Contrainte 2 : tous les cours doivent être placés dans une salle de taille suffisante et possédant les équipements nécessaires. Ainsi pour chaque cours on pose la contrainte suivante (La liste des salles possibles est indexée par la variable salle, on vérifie juste que la salle choisit pour le cours i est bien une salle possible pour ce même cours) :

$$\forall i \in [1..\mathrm{Nc}],\ \mathrm{LS_i}(salle_i) = 1$$

Contrainte 3 : tous les cours affectés dans un même créneau doivent avoir lieu dans des salles distinctes. Il s'agit d'une contrainte globale qui impose à toutes les couples ($creneau_i$, $salle_i$) associées aux cours de prendre des valeurs différentes.

$$allDifferent(\{(creneau_i, salle_i) | (i, j) \in [1..Nc] \times [1..Nc]\})$$

Remarque : on peut voir cette contrainte de différence entre couples comme une contrainte d'ordre strict entre des expressions numériques uniques associées à chaque couple (ex : 100 $salle_i + creneau_i$).

2.3. Fonction objectif

Définition

On cherche à minimiser la somme des pénalités pour chaque contrainte de préférences.

$$\min f = P_1 + P_2 + P_3$$

Contrainte de préférence P1

On caractérise les créneaux en fin de journée par le reste de la division de leur numéro par 9 : il vaut 0 pour un créneau en fin de journée, et prend des valeurs entre 1 et 8 dans les autres cas. On peut utiliser une fonction de type $Numero \mapsto \max((1 - Numero \text{ MOD } 9), 0)$ ou bien utiliser un tableau de booléens tel que T[0]=1, T[1..8]=0.

$$P_1 = \sum_{j=1}^{Nc} Effectif(j) \cdot T[creneau_i \text{ MOD } 9]$$

Contrainte de préférence P2

Soit $\delta_{X,Y} = \begin{cases} 1 \text{ si } x = y \\ 0 \text{ sinon} \end{cases}$. On peut définir $\delta_{X,Y}$ par $(X,Y) \mapsto 1 - \min(|X - Y|,1)$ pour X et Y entiers.

Dés lors, on peut exprimer si un étudiant suit un cours dans un créneau d'une journée donné facilement :

Soit *Acours(e,k,c)* une variable qui prend la valeur 1 si l'étudiant e a cours dans le *c*-ième créneau de la journée *k* et 0 autrement.

$$Acours(e,k,c) = \sum_{l \in LS_e} \delta_{creneau_l, 9\cdot(k-1)+c}$$

Si un étudiant suit trois cours d'affilée, le produit $\prod_{i=0}^{2} Acours(e,k,c+i)$ vaut 1 . On remarque que si l'étudiant suit quatre cours d'affilée, cela signifie qu'il suit deux fois trois cours d'affilée ($\prod_{i=0}^{2} Acours(e,k,c+i) + \prod_{i=1}^{3} Acours(e,k,c+i)$ =2). Il suffit donc de considérer tous les triplets.

$$P_2 = \sum_{e=1}^{Ne} \sum_{j=1}^{5} \sum_{c=1}^{7} Acours(e,j,c) \cdot Acours(e,j,c+1) \cdot Acours(e,j,c+2)$$

Contrainte de préférence P3

Le nombre de cours suivis par un étudiant e dans une journée j est donné par $\sum_{c=1}^{9} Acours(e,j,c)$. Il faut compter un point de pénalité si ce nombre de cours vaut 1.

Donc :

$$P_3 = \sum_{e=1}^{Ne} \sum_{j=1}^{5} \delta_{\sum_{c=1}^{9} Acours(e,j,c),1}$$

3. Deuxième modélisation

3.1. Présentation

Cette modélisation est très inspirée des techniques de modélisation en programmation linéaire. Le nombre élevé de variables binaires la rend néanmoins difficilement utilisable.

3.2. Variables

$\delta_{i,j,k} \in \{0,1\}$: indique si le cours *i* à lieu au créneau *j* dans la salle *k* .

3.3. Formulation des contraintes

Contrainte 1

Les cours choisis par un étudiant ne sont pas placés dans le même créneau.

$$\forall e \in [1, Ne], \forall k \in LC_e, \forall c \in [1, Nc] \quad \sum_{S=1}^{Ns} \delta_{k,c,S} \leq 1$$

Contrainte 2

Les cours suivis par un étudiant sont placé dans une salle possible (conformément à sa taille et ses équipement) et un créneau.

$$\forall e \in [1, Ne], \forall i \in LS_e \quad \sum_{s=1}^{Ns} \sum_{c=1}^{Nc} \delta_{i,c,s} = 1$$

Note : si des cours ne sont suivis par aucun étudiant, ils ne seront pas planifiés.

Contrainte 3 :

Une salle n'est utilisée que par au plus un cours, à un créneau donné.

$$\forall c \in [1, Nc], \forall s \in [1, Ns], \quad \sum_{k=1}^{Nc} \delta_{c,k,s} \leq 1$$

3.4. Fonction objectif

On garde la terminologie (P1,P2,P3) définie en 2.3.

Première contrainte de préférence P1

$$P_1 = \sum_{i=1}^{5} \sum_{c=1}^{Nc} \sum_{s=1}^{Ns} \delta_{c,9\cdot i,s}$$

Seconde contrainte de préférence P2

L'idée est de compter les triplets de cours consécutifs (cf. 2.3).

$$P_2 = \sum_{e=1}^{NE} \sum_{i=1}^{5} \sum_{c=1}^{7} \left[\left(\sum_{l \in LC_e} \sum_{j=1}^{NS} \delta_{l,9(i-1)+c,j} \right) \cdot \left(\sum_{l \in LC_e} \sum_{j=1}^{NS} \delta_{l,9(i-1)+c+1,j} \right) \cdot \left(\sum_{l \in LC_e} \sum_{j=1}^{NS} \delta_{l,9(i-1)+c+2,j} \right) \right]$$

Troisième contrainte de préférence P3

Pour compter le nombre de cours pour un étudiant *e* et une journée *j* donnés, on utilise l'expression $\sum_{c \in LC_e} \sum_{d=1}^{9} \sum_{s=1}^{Ns} \delta_{c,9(j-1)+d,s}$ (c représente un cours dans la liste des cours choisis par l'étudiant, d représente créneaux de la journée j et s les salles). Il faut ensuite compter une pénalité de 1 si cette expression est égale à 1. On utilise la fonction définie en 2.3 (symbole de Kronecker). On obtient donc :

$$P_3 = \sum_{e=1}^{Ne} \sum_{j=1}^{5} \left[1 - \min\left(\left| \left(\sum_{c \in LC_e} \sum_{d=1}^{9} \sum_{s=1}^{Ns} \delta_{c,9(j-1)+d,s} \right) - 1 \right|, 1 \right) \right]$$

4. Troisième modélisation

4.1. Variables

On choisit cette fois une seule variable qui associe les numéros de cours pour chaque couple (créneau, salle).

$\forall (i,j) \in [1,45] \times [1,Ns], C_{ij}$: indique le numéro du cours affecté au créneau n°i et à la salle n°j.

On décide par convention que C_{ij} vaut 0 si aucun cours n'est placé à cet endroit. Son domaine est donc D = [0..Nc].

4.2. Contraintes dures

Contrainte 1 : Pour un créneau donné (i fixé), on s'assure que la variable C_{ij} (j variant pour les différentes salles) ne prend qu'une fois la valeur d'un cours suivi par un même étudiant. On a ainsi pour chaque étudiant et chaque créneau.

$$\forall e \in [1, Ne] \forall i \in [1,45] \sum_{l \in LC_e} occur(\{C_{i,j} | j \in [1, Ns]\}, l) \leq 1$$

Contrainte 2 :

Etant donné que LS_k représente la liste des salles possibles du cours n°k, on peut exprimer la contrainte 2 de la façon suivante :

(Cette fois c'est l'ensemble des salles possibles qui est indexée par la variable C_{ij})

$$\forall (i,j) \in [1,45] \times [1,Ns],\ LS_{C_{i,j}}(j) = 1$$

Contrainte 3 :

Comme un cours n'a lieu qu'à une seule salle et à un seul créneau, il faut que les C_{ij} prennent tous des valeurs différentes, excepté si C_{ij} =0. On utilise une contrainte globale qui nous donne le nombre d'occurrence d'une valeur parmi l'ensemble de toutes les variables. Dans l'hypothèse où un cours n'est pas forcément suivi par un étudiant, il faut remplacer l'inégalité par une égalité.

$$\forall k \in [1, Nc], occur(\{C_{i,j}\}, k) \leq 1$$

4.3. Fonction objectif

Contraire de préférence 1

On dispose du tableau Effectif(i) qui donne le nombre d'élèves qui suivent le cours numéro i. On observera que les numéros des derniers créneaux de claque journée sont un multiple de 9.

$$P_1 = \sum_{j=1}^{Ns} \sum_{k=1}^{5} Effectif(C_{9k,j})$$

Contrainte de préférence 2

Soit Acours(e,k,c) une variable qui prend la valeur 1 si l'étudiant *e* a cours dans le *c*-ième cours de la journée k, et 0 autrement. On a :

$$Acours(e,k,c) = \sum_{l \in LS_e} \sum_{j=1}^{NS} \delta_{(C_{9\cdot(k-1)+c,j}),l}$$

(avec $\delta_{X,Y} = \begin{cases} 1 \text{ si } X = Y \\ 0 \text{ sinon} \end{cases}$, voir 2.3 pour une définition explicite)

$$P_2 = \sum_{e=1}^{Ne} \sum_{j=1}^{5} \sum_{c=1}^{7} Acours(e,j,c) \cdot Acours(e,j,c+1) \cdot Acours(e,j,c+2)$$

Contrainte de préférence 3

Soit NC(e,k) le nombre de cours suivis par l'étudiant *e* dans la journée *k.*

On a $NC(e,k) = \sum_{i=9k-8}^{9k} \sum_{j=1}^{NS} \sum_{l \in LC_e} \delta_{C_{i,j},l}$

Il faut compter une pénalité de 1 lorsque cette expression vaut 1, pour chaque journée et chaque étudiant. En utilisant l'expression du symbole de Kronecker définie en 2.3, on obtient :

$$P_3 = \sum_{e=1}^{Ne} \sum_{k=1}^{5} \left(1 - \min\left(\left|NC(e,k) - 1\right|, 1\right)\right)$$

5. Autres modélisations envisagées

Dans le même ordre que la modélisation 1, on peut imaginer ajouter des variables et donc diminuer leurs domaines respectifs. On obtient par exemple en ajoutant la variable journée à la modélisation 1 : $salle_i$ de domaine [1..Ns], $creneau_i$ de domaine [1..9], $journée_i$ de domaine [1..5] ce qui faciliterai l'expression des contraintes faisant appel à la journée notamment dans le calcul de la note.

A l'inverse on peut utiliser une variable unique pour chaque cours, de type $cours_i$ de domaine [1..Ns×45] qui permet d'exprimer facilement la contrainte 3 par un simple allDifferent, mais rend la n°1 moins explicite car il devient mal aisé de retrouver le créneau et la salle à partir de la valeur prise par la variable. C'est néanmoins une modélisation tout à fait envisageable.

6. Modélisation retenue

6.1. Remarque sur la fonction objectif

Notre utilisation du symbole de Kronecker, peut se traduire de manière plus efficace que $(X,Y) \mapsto 1 - \min(|X-Y|,1)$. On peut modéliser ceci en Jsolver avec la fonction count, qui permet de compter le nombre d'occurrences d'une valeur parmi un tableau de variables ; il suffirait de sélectionner la bonne composante du vecteur retourné. En Choco, on peut utiliser occur de la même façon. Enfin, il est toujours possible d'implémenter une nouvelle contrainte qui compterait itérativement le nombre d'élements ; néanmoins on perd les optimisations faites sur occur par les solveurs. Notons que le recours à occur permet de diminuer le nombre de sommes : occur travaille sur des vecteurs là où le symbole de Kronecker traite des éléments simples.

6.2. Limites des modélisations

On éliminera d'emblée la modélisation 2 qui ne fait appel à aucune contrainte globale et possède un nombre « astronomique » de variables et de contraintes élémentaires. On préfère retenir les modélisations utilisant des contraintes globales dans la mesure où celles-ci sont plus efficaces car elles donnent une vision plus tôt du problème à satisfaire. Cette vision globale portant sur un ensemble de variable évite de descendre dans l'arbre de recherche car on sait d'emblée que l'ensemble des contraintes ne sera pas satisfaisable même si une partie des contraintes élémentaires qui constituent la contrainte globale sont satisfaisables. On possède une vision plus générale du problème et donc plus discriminante. Par ailleurs des algorithmes spécifiques et plus efficaces sont utilisés pour les traiter. Même si les contraintes de cette modélisation sont linéaires, elles portent en revanche sur des variables entières, impliquant une résolution par un algorithme de type Branch and Bound (donc non polynomial), mais avec 180000 variables (cf. tableau ci-dessous), c'est irréaliste. Ce n'est donc pas une modélisation judicieuse dans ce contexte.

La modélisation 1 s'appuie sur des contraintes globales de type *allDifferent.* Les contraintes 1 et 2 semblent très efficaces. Grâce à un prétraitement sur les données, la contrainte 2 permet de réduire directement les domaines des variables $salle_i$ en excluant d'emblée toutes les valeurs impossibles (on pourrait pratiquement éliminer ces valeurs des domaines de chaque variable $salle_i$ dès le début). Le *allDifferent* sur les cours suivis par un élève travaille sur environ une vingtaine de cours et reste donc très raisonnable. En revanche la contrainte 3 semble difficile à mettre en œuvre. Il s'agit d'un *allDifferent* sur tous les couples possibles (créneau, salle), soit Nc×Nc couples. On prenant l'instance 9 qui représente le pire des cas c'est à dire 440 cours, on obtient 193600 couples, ce qui représente plusieurs milliards de contraintes élémentaires entre couples ! (On compte n(n-1)/2 contraintes élémentaires pour un *allDifferent* portant sur n variables). Même si la contrainte procède par des algorithmes spécifiques, on a le sentiment que cette contrainte unique n'est pas adaptée et sera très difficile à évaluer. L'alternative à cette contrainte globale est d'écrire Nc×(Nc-1)/2 contraintes du type :

$$\forall Ci, \forall Cj, Ci \neq Cj, creneau(Ci) = creneau(Cj) \Rightarrow salle(Ci) \neq salle(Cj)$$

On perd alors l'avantage d'utiliser une contrainte globale. Cette modélisation utilisant deux jeux de variables ne nous semble donc pas la plus judicieuse.

La modélisation 3 s'appuie quant à elle sur des contraintes globales de type *occur.* La contrainte 3 s'exprime cette fois plus simplement et porte dans le pire des cas (11 salles, instance 9) sur 495 variables. La contrainte 2 reste aussi adaptée que pour la modélisation 1 et permet d'éliminer les cours impossibles pour chaque C_{ij} du fait de la salle (les domaines de chaque C_{ij} sont restreint d'emblée). C'est cette fois la contrainte 1 qui est semble limitante dans cette modélisation. Néanmoins, elle semble moins délicate à mettre en œuvre que ne l'était la contrainte 3 dans la modélisation 1. En fait, on fait intervenir pour chaque élève 45x « le nombre de cours suivis par l'élève » contraintes de type *occur*. Néanmoins, cela représente dans le pire des cas (instance 7, 350 élèves) 15750 contraintes. On pourrait également construire des groupes d'étudiants suivant les mêmes cours durant la phase de pré-traitement afin de limiter le nombre d'étudiants.

6.3. Conclusion

Des trois modélisations proposées ici en détail, la modélisation 3 nous semble la plus judicieuse même si elle possède (relativement au nombre d'élèves et de cours) plus de contraintes que la modélisation 1 (voir tableau). La deuxième modélisation évoquée au 5 est également une modélisation qui pourrait donner des résultats. Le seul moyen de valider un modèle reste encore de le tester.

6.4. Dimensions

Modélisation	**Nombre de variables**	**Domaine des variables**	**Nombre de contraintes** (avec #(LSe) ≈ Nc/20)	**Exemple : contraintes Ne=200, Nc=400, Ns=10**	**Exemple : variables Ne=200, Nc=400, Ns=10**
1	2×Nc	[1,45] et [1,Ns]	Ne + Nc + **1***	601	800
2	Nc×Ns×45	{0,1}	Ne×Nc²/20 + Ne×Nc/20+Nc×Ne	244000	180000
3	45×Ns	[1,Nc]	45×Ne + 45×Ns + Nc	9850	450

Note : le AllDiff global de la modélisation 1 touche 160000 couples de variables.