TPs : Prog. Linéaire en Nombres Entiers avec SCIP (2)

Planification de services / “nurse rostering”

Un problème qu'on rencontre dans de nombreuses organisations consiste à planifier, sur une période pouvant aller jusqu'à plusieurs mois, les plannings de personnels en fonction des de leurs contrats de travail et de nécessités de service. C'est un problème complexe lorsqu'il n'y a pas de “pattern” régulier des périodes de travail ; dans un hôpital par exemple, il doit y avoir des infirmier-e-s tous les jours, et à toute heure de la journée, mais le nombre nécessaire varie suivant le jour et suivant l'heure. Il y a de nombreuses contraints ; par exemple, il faut respecter un certains nombre de jours de repos entre deux périodes de travail pour chaque infirmièr-e – et ce nombre peut dépendre du contrat de travail. Comme il arrive qu'il n'y ait pas solution qui satisfasse parfaitement toutes les contraintes, certaines contraintes sont affectées de *pénalités*. Le problème devient alors un problème d'optimisation : il s'agit de trouver un planning qui satisfasse “au mieux” les contraintes, c'est-à-dire qui minimise les pénalités des contraintes violées.

On vous demande de résoudre un ensemble d'instances de ce problème disponibles sur Moodle. Chaque instance est décrite dans un fichier `shift-scheduling-XX.zplread`, où XX désigne un nombre entier, le numéro de l'instance.

Chaque jour on a différents *services*, qui ont des identifiants alpha-numériques, par exemple “E" (pour *E*arly), “D” (pour *D*ay), “L” (pour *L*ate) dans l'instance 3, ou “a1”, ... “p2”, ... dans l'instance 24. Chaque personnel a aussi un identifiant alpha-numérique. Chaque personne ne peut effectuer qu'un service au plus par jour. Les jours de la période considérée sont numérotés à partir de 0, et on suppose que le premier jour est un lundi.

Le programme ZIMPL `shift-scheduling.zpl` lit une instance dans un fichier `shift-scheduling-XX.zplread`, et définit :

Le paramètre horizon

horizon	= le nombre de jours de la période à planifier
0 = premier jour = un lundi le nombre de jours est toujours un multiple de 7	

Des ensembles :

Personnes	ensemble des personnels disponibles
Days	ensemble des jours à planfier, de la forme $\{0, \ldots, \text{horizon} - 1\}$
Weekends	ensemble des week-ends, $= \{1, \ldots, \text{horizon}//7\}$ *(// = division entière)*
Services	ensembles des services à planifier chaque jour

Un tableau de variables de décision :

assigned$[p, d, s]$	= 1 si la personne p est affectée au service s le jour d = 0 sinon

Des paramètres / tableaux de paramètres :

duree[s]	durée du service s (en minutes)
MaxTotalMinutes[p]	durée totale max. de service de la personne p sur la période
MinTotalMinutes[p]	durée totale min. de service de la personne p sur la période
MaxWeekends[p]	nombre max. de week-ends travaillés pour p
(un week-end est travaillé si p travaille le samedi et/ou le dimanche)	
MaxShift[p, s]	nombre max. de services de type s que p peut assurer sur la période
requirement[d, s]	nombre de personnes requises pour le service s le jour d
dayOff[p, d]	$= 1$ si la personne p doit être de repos le jours d $= 0$ si p peut travailler ce jour-là

On vous demande de compléter ce fichier, en définissant des variables supplémentaires, l'objectif et les contraintes, afin de résoudre ce problème de planification. Les questions qui suivent vont vous guider pour cela.

Exercice 1 (Objectif, première version, simple)**.** Dans cette première version, on veut simplement minimiser l'écart entre le nombre de personnes requises pour chaque service, et le nombre de personnes affectées à ce service.

On suppose d'abord qu'on a une variable x, un paramètre fixé n, et qu'on veut minimiser l'écart entre x et n : on veut écrire un objectif linéaire qui soit équivalent à minimiser $|x - n|$.

Question 1.1. Si on introduit deux variables $y, z \geq 0$, avec la contrainte $x - y + z = n$, et qu'on minimise $y + z$, que valent y et z si :

- $x = 7$ et $n = 5$;
- $x = 2$ et $n = 5$.

Question 1.2. Compléter le fichier shift-scheduling.zpl afin de minimiser la somme des écarts entre les nombres de personnes requises, chaque jour et pour chaque service, et les nombres de personnes affectées chaque jours à chaque service. On pourra donc introduire deux tableaux de variables supplémentaires.

Question 1.3. Lancer l'optimisation. Quelle est la valeur de l'objectif pour la solution optimale retournée.

Exercice 2 (Contraintes simples)**.** Ajouter maintenant les contraintes suivantes :

1. personne ne peut faire plus d'un service par jours
2. pour chaque personne p et chaque jour d, si dayOff[p, d] $= 1$ alors p ne doit être affectée à aucun service ce jour-là
3. chaque personne p ne peut prendre plus de MaxShift[p, s] fois le service s sur la période
4. la durée totale de service (en minutes) de la personne p doit être dans l'intervalle [MinTotalMinutes[p], MaxTotalMinutes[p]]

Exercice 3 (Séquences max.)**.** Les instances ont aussi les deux paramètres suivants :

MaxConsecutiveShifts[p]	nombre max. de jours de service consécutifs pour la personne p
ForbiddenSeq[$s_1 s_2$]	$= 1$ s'il ne faut pas que le service s_1 un jour soit suivi du service s_2 le lendemain $= 0$ si le service s_1 un jour peut être suivi du service s_2 le lendemain

Exprimer des contraintes linéaires pour que :

1. à aucun moment durant la période, le nombre. de jours de service consécutifs pour la personne p ne dépasse MaxConsecutiveShifts[p] ;
2. si p effectue le service s_1 un certain jour d, et si ForbiddenSeq[s_1, s_2] $= 1$, alors p ne doit pas faire le service s_2 le jour $d + 1$.

Exercice 4 (Séquences min.). Les instances ont aussi les deux paramètres suivants :

MinConsecutiveDaysOff[p]	durée min. de toute séquence de jours non travaillés par p
MinConsecutiveShifts[p]	nombre min. de jours de service consécutifs pour la personne p

Question 4.1. Exprimer des contraintes conditionnelles pour que :

1. à aucun moment durant la période il y ait une séquence de jours non travaillés par p qui fasse moins de p jours ;
2. à aucun moment durant la période, le nombre. de jours de service consécutifs pour la personne p ne soit inférieur à MinConsecutiveShifts[p].
   Indication 1 : Pour contraindre qu'il n'y ait pas par exemple de séquence de repos de moins de 3 jours qui commence un certain jour d, on peut écrire

```
vif assigned[p,d-1,s] == 1 and assigned[p,d,s] == 0
  then assigned[p,d+1,s] +  assigned[p,d+2,s]  == 0
  end ;
```

   Indication 2 : C'est plus facile de travailler avec, pour chaque personne p et chaque jour d, une variable booléenne w_{pd} qui vaudra 1 si p travaille le jour d, 0 sinon, c'est-à-dire que $w_{pd} = 1$ quand assigned$[p, d, s]$ vaut 1 pour au moins l'un des services s. S'il y a trois services, on peut modéliser cela avec la double contrainte :

$$w_{pd} \leq \text{assigne}[p, d, E] + \text{assigne}[p, d, D] + \text{assigne}[p, d, L] \leq 3 \times w_{pd}$$

Question 4.2. On observe que les contraintes conditionnelles précédentes génèrent beaucoup de nouvelles variables et de contraintes conditionnelles. Réexprimer directement les contraintes ci-dessus à l'aide de contraintes conditionnelles, en observant par exemple que si, pour p, le nombre minimum de jours consécutifs non travaillés est 3, alors il suffit d'interdire les séquences de la forme $Off - On - Off$, $Off - On - On - Off$, $Off - On - On - On - Off$.

Exercice 5 (Objectif avec pénalités). Les instances ont aussi des paramètres qui pondèrent les écarts, en terme de nombre de personnels, par rapport aux nombres requis pour chaque service ; ainsi que des paramètres qui permettent de prendre en compte certaines préférences des personnels :

belowCoverPen[d, s]	pénalité quand pas assez de personnels pour le service s le jour d pénalité totale = belowCoverPen[d, s]× nb. pers. manquantes
aboveCoverPen[d, s]	pénalité quand pas trop de personnels pour le service s le jour d pénalité totale = aboveCoverPen[d, s]× nb. pers. en trop
prefOff[p, d, s]	pénalité si p effectue le service s le jour d
prefOn[p, d, s]	pénalité si p n'effectue pas le service s le jour d

Modifier l'objectif pour que, au lieu de minimiser simplement l'écart entre les nombres de personnes requises et les nombres de personnes affectées à chaque service, on minimise la somme de toutes ces pénalités.