Faculté des sciences et de génie

Département d'informatique

IFT-4001 - IFT-7020

Optimisation Combinatoire

Travail pratique

Présenté au

Professeur Claude-Guy Quimper

Par

Claudia Laramée – 111 042 844 - cllar117

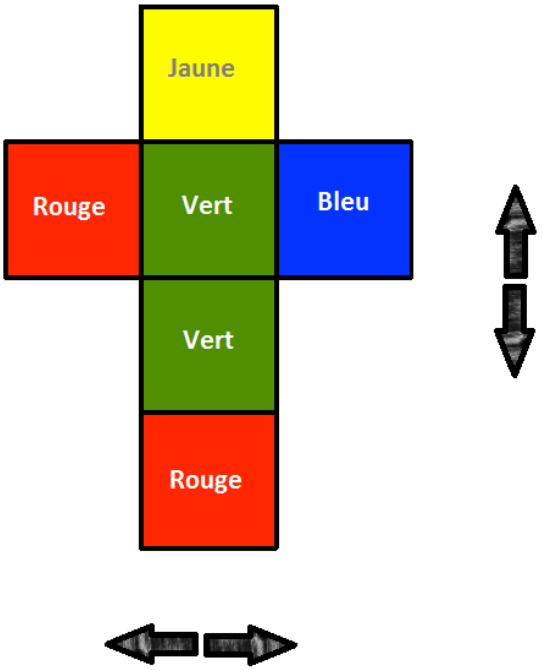
Maxime Leclerc – 999 050 537 - malec207

22 février 2014

Premier problème

Description du modèle

Pour définir le modèle servant à modéliser le problème des cubes, nous sommes partis du fait qu'il y a deux séquences de quatre couleurs possibles pour chaque face d'un cube. La première séquence est obtenue en parcourant le cube vers le haut ou le bas. La seconde est obtenue en parcourant le cube vers la gauche ou la droite. Il y a 6 faces différentes sur un cube fois 4 séquences de couleurs = 24 séquences de quatre couleurs.

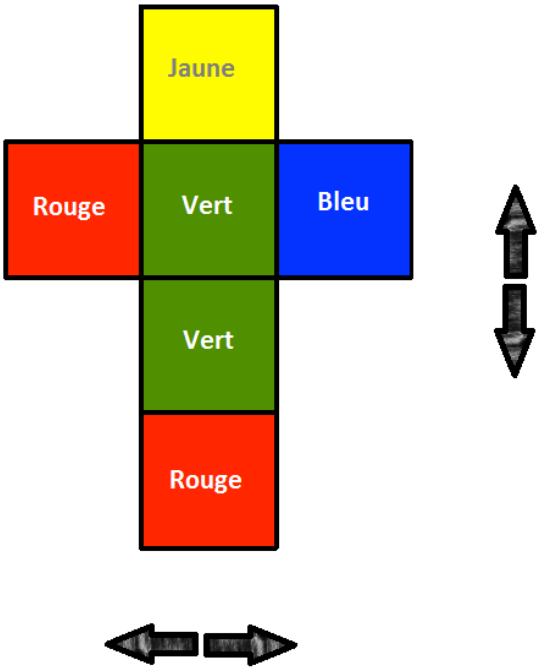


Pour le premier cube, nous avons :

Vers le haut	Bleu – jaune – rouge – vert (a)	Rouge – vert – vert – jaune (e)
Vers le bas	Bleu – vert – rouge – jaune (b)	Rouge – jaune – vert – vert (f)
Vers la gauche	Bleu – vert – rouge – rouge (c)	Rouge – rouge – vert – bleu (d)
Vers la droite	Bleu – rouge – rouge – vert (d)	Rouge – bleu – vert – rouge (c)

Vers le haut	Rouge – jaune – bleu – vert (b)	Jaune – rouge – vert – vert (e)
Vers le bas	Rouge – vert – bleu – jaune (a)	Jaune – vert – vert – rouge (f)
Vers la gauche	Rouge – rouge – bleu – vert (c)	Jaune – rouge – vert – bleu (a)
Vers la droite	Rouge – vert – bleu – rouge (d)	Jaune – bleu – vert – rouge (b)

Vers le haut	Vert – jaune – rouge – vert (e)	Vert – vert – jaune – rouge (e)
Vers le bas	Vert – vert – rouge – jaune (f)	Vert – rouge – jaune – vert (f)
Vers la gauche	Vert – rouge – rouge – bleu (c)	Vert – rouge – jaune – bleu (b)
Vers la droite	Vert – bleu – rouge – rouge (d)	Vert – bleu – jaune – rouge (a)



Nous pouvons constater qu'il n'y a en fait que six séquences différentes de longueur sept, mais dont les positions de départ sont décalées les unes par rapport aux autres. Nous pouvons donc regrouper chaque groupe de quatre séquences décalées de cette façon :

(a) Bleu – jaune – rouge – vert – bleu – jaune – rouge

(b) Bleu – vert – rouge – jaune – bleu – vert – rouge

(c) Bleu – vert – rouge – rouge – bleu – vert – rouge

(d) Bleu – rouge – rouge – vert – bleu – rouge – rouge

(e) Rouge – vert – vert – jaune – rouge – vert – vert

(f) Rouge – jaune – vert – vert – rouge – jaune – vert

Nous pouvons donc construire un vecteur de 6 fois 7 = 42 éléments pour chaque cube qui représentent toutes les séquences de couleurs possibles pour celui-ci.

Maintenant, pour résoudre le problème, il suffit de trouver quatre variables *i, j, k, l* dont les domaines sont les indices des vecteurs de [0,42[avec les contraintes suivantes :

- cube1[i], cube2[j], cube3[k] et cube4[l] sont différents
- cube1[i +1], cube2[j + 1], cube3[k + 1] et cube4[l + 1] sont différents
- cube1[i + 2], cube2[j + 2], cube3[k + 2] et cube4[l + 2] sont différents
- cube1[i + 3], cube2[j + 3], cube3[k + 3] et cube4[l + 3] sont différents

Cependant, il faut restreindre les valeurs possibles des indices pour qu'ils ne permettent pas de « mélanger » deux séquences distinctes pour un même cube : ceci est une conséquence du fait que nous avons joint bout à bout toutes les séquences possibles dans un cube.

Voici donc le domaine de *i, j, k, l* qui tient compte de cette spécification : {0,1,2,3,7,8,9,10,14,15,16,17,21,22,23,24,28,29,30,31,35,36,37,38}.

Avec ce nouveau domaine, on s'assure de trouver un sous-vecteur pour chaque cube de façon à ce que chaque élément d'un sous-vecteur soit différent avec de l'élément des autres sous-vecteurs tout en respectant les séquences possibles pour chaque vecteur.

Analyse du modèle

Le modèle comporte quatre vecteurs de 42 éléments, que l'on peut considérer comme quatre variables avec un domaine de 42 éléments énumérés.

Le modèle comporte également quatre variables (*i, j, k, l*) qui ont un domaine de 24 éléments énumérés chacun.

Le modèle comporte aussi quatre contraintes « alldifferent », telles que mentionnées plus haut, qui comparent quatre éléments des vecteurs chacune.

Il est à noter que dû aux limitations du langage de programmation de Choco 3, le modèle implémenté en Java possède 22 variables et 20 contraintes selon Choco 3.

Analyse des résultats obtenus avec Choco 3

Solution obtenue :

Cube 1	Cube 2	Cube 3	Cube 4
Rouge	Bleu	Vert	Jaune
Vert	Jaune	Bleu	Rouge
Vert	Rouge	Jaune	Bleu
Jaune	Bleu	Rouge	Vert

Nombre de retours arrière : 12

Temps de résolution : 0,048s

Voici les statistiques complètes de Choco 3 :

– Complete search –

Solutions : 1

Building time : 0,182s

Initialisation : 0,000s

Initial propagation : 0,029s

Resolution : 0,048s

Nodes : 11

Backtracks : 12

Fails : 7

Restarts : 0

Max depth : 5

Propagations : 303 + 0

Memory : -1mb

Variables : 22

Constraints : 20

Deuxième problème

Description du modèle

Employé \ Heure	09:00	09:30	10:00	…	15:30	16:00	16:30	Somme	Heures
Employe 1	X(1, 1)	X(1, 2)	…					**Somme X(1, j)**	2 * SommeX(1, j)
Employe 2	X(2, 1)	…						**Somme X(2, j)**	…
Employe 3	…							**…**	
Employe 4									
Employe 5									

Offre (Somme)	Somme X(i, 1)	X(i, 2)	…					
Demande	Y(1)	Y(2)	…					
Perte	20 * ((Somme X(i, 1)) – Y (1))	…						

Dans le tableau ci-haut, x(i, j) représente le choix pour l'employé i à la période j de travailler ou non. Le domaine est [0, 1]. x(i, j) = 1 si l'employé travaille et x(i, j) = 0 si l'employé ne travaille pas.

Nous avons 5 employés donc le domaine de i est [1, 5]. Nous avons aussi 16 périodes donc le domaine de j est [1, 16]. Les périodes de travail débutent à 9h00 et se terminent à 16h59.

Chaque employé doit travailler entre 5 et 7 heures par jour incluant une pause obligatoire de 30 minutes. Nous avons donc les contraintes :

Pour tous les i [1, 5]: $\Sigma x(i, j) \geq 9$ Chacune des 5 sommes se fait sur tous les j [1, 16]

Pour tous les i [1, 5]: $\Sigma x(i, j) \leq 13$ Chacune des 5 sommes se fait sur tous les j [1, 16]

Pour toutes les périodes, il doit y avoir au moins un employé. Ceci donne la contrainte suivante :

Pour tous les j [1, 16]: $\Sigma x(i, j) \geq 1$ Chacune des 16 sommes se fait sur tous les i [1, 5]

Dans le tableau précédent, y (j) représente le nombre d'employés souhaité pour la période j. Selon l'énoncé, nous avons :

Heure	9		10		11		12		13		14		15		16	
Nombre d'employés requis	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Nombre d'employés souhaité	1	2	3	4	5	4	2	3	4	3	5	5	4	3	3	3

Idéalement, nous aimerions avoir le nombre d'employés souhaité à chacune des périodes. Cependant, cela n'est pas nécessairement possible. Il en résulte que l'entreprise perd un montant pour chaque demi-heure où le nombre d'employés présents est différent du nombre d'employés souhaité. Cette perte est égale à x * 20$ lorsqu'il y x employés de plus ou x employés de moins que le nombre d'employés souhaités pour une période donnée. Nous cherchons à minimiser cette perte. Calculons premièrement l'offre d'employés présents.

Pour tous les j [1, 16] $\text{Offre}(j) = \Sigma x(i, j)$

Chacune des 16 sommes se fait sur tous les i [1, 5]

Calculons maintenant la perte. Ceci nous donne la valeur suivante que nous voulons minimiser :

Perte = $20 * \Sigma \; | \; \text{Offre}(j) - y(j) \; |$ La somme des valeurs absolues se fait sur tous les j [1, 16].

Analyse du modèle

Le modèle comporte une matrice de n = 5 employés par p = 16 périodes. Ceci nous donne n * p = 80 variables entières dont le domaine est [0, 1].

Choco requiert la création de variables additionnelle afin d'exprimer notre objectif. Nous avons donc une variable entière de plus qui représente la perte totale. Son domaine est :

[0, (n = 5) * (p = 16) * 20] = [0, n * p * 20] = [0, 1600]

Nous avons ensuite besoin de 3 variables constantes entières pour représenter le nombre d'employés requis (domaine [0, n = 5]), le nombre de périodes de travail par employé minimum (domaine [0, p = 16]) et le nombre de périodes de travail par employé maximum (domaine [0, p = 16]).

Nous avons aussi besoin de p = 16 variables constantes entières pour représenter le nombre d'employés souhaités. Leurs domaines sont [0, n = 5].

Choco requiert la création de p = 16 variables entières pour représenter la somme du nombre d'employés par période. Leurs domaines sont [0, n = 5]. Choco requiert aussi la création de p = 16 variables entières constantes pour représenter le nombre d'employés souhaité par période. Leurs domaines sont [0, n = 5]. De plus, Choco requiert la création de p = 16 variables entières pour représenter la perte par période. Leurs domaines sont [0, n = 5] dans notre modèle. Nous multiplions ensuite ces valeurs par 20.

Choco requiert la création de p = 16 contraintes pour représenter la somme du nombre d'employés par période. Choco aussi la création de p = 16 contraintes pour représenter le nombre d'employés souhaité par période. De plus, Choco requiert la création de p = 16 contraintes pour représenter la perte par période.

Nous avons p = 16 contraintes pour s'assurer que nombre d'employés correspond au nombre d'employés minimum requis. Nous avons n = 5 contraintes pour s'assurer que nombre de périodes travaillées par employés soit au moins le minimum spécifié (exemple : 5 heures). Nous avons n = 5 contraintes pour s'assurer que nombre de périodes travaillées par employés soit au plus le maximum spécifié (exemple : 7 heures).

Nous avons n = 5 contraintes pour s'assurer que le pattern de périodes travaillées corresponde à l'énoncé du problème. Dans l'énoncé fourni, un employé peut avoir une journée qui ressemble à ceci :

(a)	N'a pas commencé à travailler	0*
(b)	Commence à travailler	1*
(c)	Travaille 2 heures	1111
(d)	En pause pour 30 minutes	0
(e)	Travaille 2 heures	1111
(f)	Fini de travailler	1*
(g)	A terminé le travail	0*

Ceci correspond à l'expression régulière suivante : **"0*1*1111011111*0*"**

Nous avons une dernière contrainte qui calcule la perte totale et la stocke dans la variable de perte.

Analyse Θ

Au total, nous avons (n * p = 80) + 1 + 3 + (4 * p = 4 * 16 = 64) variables, ou encore :

4 + 4 * p + n * p variables = 148 variables.

Choco semble créer des variables supplémentaires car son décompte total est de 159 variables. Selon notre analyse, nous avons Θ (n * p) variables.

Nous avons (4 * p = 4 * 16 = 64) + (3 * n = 3 * 5 = 15) + 1 contrainte, ou encore :

1 + 4 * p + 3 * n = 80 contraintes.

Encore une fois, semble créer des contraintes supplémentaires car son décompte total est de 106 contraintes. Selon notre analyse, nous avons Θ (n + p) contraintes.

Analyse des résultats obtenus avec Choco 3

Solution obtenue :

Heure	09:00	09:30	10:00	10:30	11:00	11:30	12:00	12:30	13:00	13:30	14:00	14:30	15:00	15:30	16:00	16:30	Somme	Heures
Employe 1	0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	**11**	6
Employe 2	1	1	1	1	1	1	1	0	1	1	1	1	0	0	0	0	**11**	6
Employe 3	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	**13**	7
Employe 4	0	1	1	1	1	1	1	1	0	1	1	1	1	0	0	0	**11**	6
Employe 5	0	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	**12**	6,5
Offre (Somme)	1	2	3	4	5	5	5	3	4	3	5	5	4	3	3	3	**58**	
Demande	1	2	3	4	5	4	2	3	4	3	5	5	4	3	3	3	**54**	
Perte	0 $	0 $	0 $	0 $	0 $	20 $	60 $	0 $	0 $	0 $	0 $	0 $	0 $	0 $	0 $	0 $	**80 $**	

Nombre de retours arrière : 22,859

Temps de résolution : 0.727s

Voici les statistiques complètes de Choco 3 :

– Complete search -

Solutions : 17

Minimize perte = 80,

Building time : 0.275s

Initialisation : 0.000s

Initial propagation : 0.005s

Resolution : 0.727s

Nodes : 11,446

Backtracks : 22,859

Fails : 11,413

Restarts : 0

Max depth : 35

Propagations : 492,153 + 0

Memory : -3mb

Variables : 159

Constraints : 106

Nous n’avons pas utilisé de redémarrage (restart).

Cependant, nous avons utilisé l’heuristique **domOverWDeg**. Selon Mami et al. (Mami, I., Bellahsene, Z., & Coletta, R. (2013). A Declarative Approach to View Selection Modeling. In Transactions on Large-Scale Data-and Knowledge-Centered Systems X (pp. 115-145). Springer Berlin Heidelberg.), cet heuristique est conçu de la manière suivante.

La stratégie choisie la variable qui possède le plus petit ratio R :

R = TailleDuDomaine / (SommeDuCompteurD’Échecs * NombreContraintesNonInstanciées)

– TailleDuDomaine est la taille du domaine actuelle.

– SommeDuCompteurD’Échecs est la somme des compteurs d’échecs causés par chaque contrainte depuis le début de la recherche.

– NombreContraintesNonInstanciées est le nombre de contraintes non instanciées qui incluent la variable.

Solution optimale

La solution que nous avons trouvée est optimale. Premièrement, nous savons que tous les employés doivent travailler au moins 5 heures. Ils doivent donc commencer au plus tard à 12h00 et finir au plus tôt à 13h59. De plus, chaque employé doit prendre une pause de 30 minutes. Si on examine le nombre d'employés souhaité entre 12h00 et 13h59, on voit qu'il varie entre 2 et 4 employés. Nous y trouvons 4 périodes. Il y doit y avoir 2 + 3 + 4 + 3 employés au total pour ces périodes pour un total de 12 présences. Cependant, chacun des 5 employés ne peut prendre qu'une seule pause dans cette période. Nous avons donc une offre minimum de (4 – 1) * 5 = 15 présences. La perte minimum pour cette période est donc de (15 présences – 12 présences) * 20$ = 60$.

Heure	9		10		11		12		13		14		15		16	
Nombre d'employés requis	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Nombre d'employés souhaité	1	2	3	4	5	4	2	3	4	3	5	5	4	3	3	3

De plus, cette perte initiale assume qu'aucun employé ne prendra de pause en dehors de 12h00 et 13h59. Cependant, la demande de 11h00 à 11h59 est de 5 employés et 4 employés. Nous ne pouvons pas avoir 5 employés présents puis seulement 4 employés à moins qu'un de ces employés ne prenne une pause. Or, nous avons déjà utilisé toutes les pauses pour la période de 12h00 et 13h59. Nous aurons donc nécessairement une perte additionnelle de 20$ pour 11h00 à 11h59. La perte totale minimum est donc de 80$ ce qui correspond bien au résultat de notre programme.