RÉPUBLIQUE DU CAMEROUN
REPUBLIC OF CAMEROON
Peace – Work - Fatherland

UNIVERSITÉ DE DSCHANG
UNIVERSITY OF DSCHANG
Scholae Thesaurus DschangensisIbiCordum

BP 96, Dschang (Cameroun) – Tél./Fax (237) 233 45 13 811 Website : http://www.univ-dschang.org.
E-mail :udsrectorat@univ-dschang.org

FACULTE DES SCIENCES
FACULTY OF SCIENCE
Département de Mathematiques-Informatique
Department of Mathematics and Computer science

BP 67, Dschang (Cameroun)
Tél./Fax (237) 233451735
E-mail : dept-MI.sciences@univ-dschang.org

Année académique 2010-2021 ***Examen de fin de semestre***

Cours: IAR427 (Recherche Opérationnelle et Problème de satisfaction des contraintes) Durée: 3h
Classe: IN4 Tous les exercices sont obligatoires Téléphones interdits Examinateur : Dr SOH Mathurin

Exercice 1 : Evaluation des Connaissances générales (5 points =1,5 +1,5)

1. Expliciter la différence entre un problème d'optimisation (programmation linéaire) et un problème de satisfaction des contraintes
2. Une entreprise fabrique des armoires et des tables ; Une armoire nécessite 1h de travail et 9 m2 de bois ; Une table nécessite 1h de travail et 5m2 de bois ; • On dispose de 6h de travail et de 45 m2 de bois ; • Chaque armoire génère un profit de 80 00FCFA, et chaque table 5000FCFA.
   a)Formuler comme un problème de programmation linéaire en nombre entiers
   b) Le résoudre par la méthode de séparation -évaluation.

Exercice 2 Problème du stockage

Considérons n programmes $P_1, P_2, \ldots, P_n$ qui peuvent être stocker sur un disque dur de capacité D gigabytes.

— Chaque programme P_i a besoin s_i gigabytes pour être stocké et a une valeur v_i

— Tous les programmes ne peuvent pas être stockés sur le disque : $\sum_{i=1}^{n} s_i > D$.

Les programmes stockés dans le disque dur doivent maximiser la valeur totale, sans dépasser la capacité du disque dur. L'objectif est de concevoir un algorithme qui permet de calculer un tel ensemble.

Nous allons construire un tableau T dans lequel les lignes seront indexées par les programmes et les colonnes par les valeurs. L'élément $T[i, j]$ représentera la valeur maximale pour un disque dur de capacité j à l'aide des i premiers programmes.

1) Donner la formule de récurrence.
2) Donner l'algorithme utilisant la programmation dynamique.

Problème : (9points)

On s'intéresse à un distributeur automatique de boissons. L'utilisateur insère des pièces de monnaie pour un total de T centimes d'Euros, puis il sélectionne une boisson, dont le prix est de P centimes d'Euros (T et P étant des multiples de 10). Il s'agit alors de calculer la monnaie à rendre, sachant que le distributeur a en réserve E2 pièces de 2 €, E1 pièces de 1 €, C50 pièces de 50 centimes, C20 pièces de 20 centimes et C10 pièces de 10 centimes.

1. Après avoir Identifié les variables (les inconnues du problème), les domaines de valeur de ces variables, et les contraintes existant entre ces variables. Modélisez ce problème sous la forme d'un problème de satisfaction de contraintes.
2. Comment pourrait-on exprimer le fait que l'on souhaite que le distributeur rende le moins de pièces possibles ?

Bon courage et bonne chance

ELEMENTS DE CORRECTION

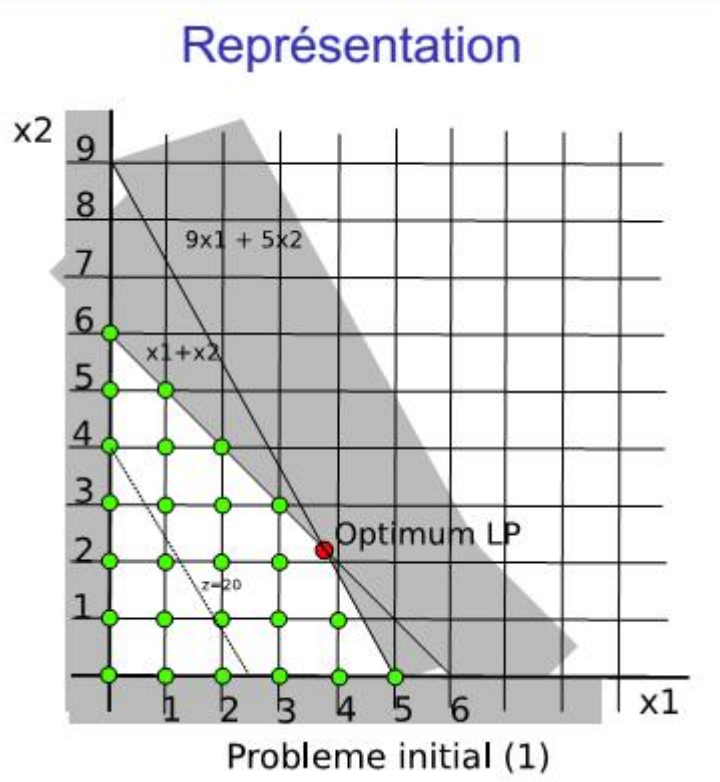
Exercice 1 : Evaluation des Connaissances générales (5 points =1,5 +1,5)

3. Expliciter la différence entre un problème d'optimisation (programmation linéaire) et un problème de satisfaction des contraintes
4. Une entreprise fabrique des armoires et des tables ; Une armoire nécessite 1h de travail et 9 m2 de bois ; Une table nécessite 1h de travail et 5m2 de bois ; • On dispose de 6h de travail et de 45 m2 de bois ; • Chaque armoire génère un profit de 80 00FCFA, et chaque table 5000FCFA.
   Formuler comme un problème de programmation linéaire en nombre entiers
   Le résoudre par la méthode de séparation -évaluation.

   Solution :
   On formule de la façon suivante :

$$\begin{aligned} \max z = \; & 8x_1 + 5x_2 \\ & x_1 + x_2 \leq 6 \\ & 9x_1 + 5x_2 \leq 45 \end{aligned}$$

$x_i \geq 0, x_i \in \mathbb{N}.$



Probleme initial (1)

- On commence par résoudre la formulation de relaxation du problème en entiers, c-à-d le problème en variables continues (LP). Ici, on obtient :

$$\begin{aligned} z &= \frac{165}{4} \\ x_1 &= \frac{15}{4} \\ x_2 &= \frac{9}{4} \end{aligned}$$

- Si la solution est en entiers, on s'arrête, on a trouvé l'optimal. Ici ce n'est pas le cas, mais la valeur z obtenue est une borne supérieure pour l'optimum en entiers.

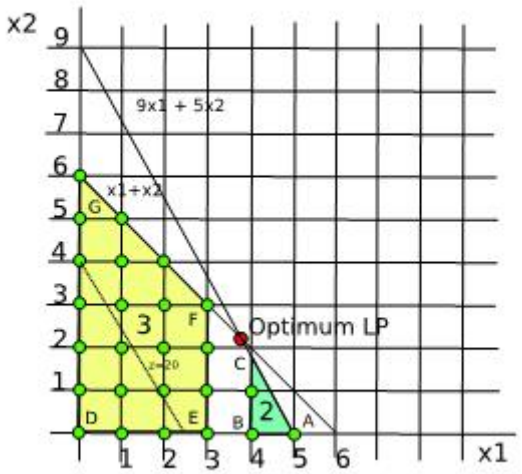
- Si la solution n'est pas entière, on partitionne le domaine : on choisit arbitrairement une variable qui est fractionnaire dans la solution optimale relaxée, par exemple ici x_1.
- On applique des contraintes supplémentaires dues à la nature de la variable, ici par exemple on impose soit $x_1 \leq 3$, soit $x_1 \geq 4$ (Question : pourquoi pouvons nous éliminer les solutions $3 < x_1 < 4$?)

Séparation

On a donc maintenant deux sous-problèmes :

2. Problème initial + contrainte $x_1 \geq 4$
3. Problème initial + contrainte $x_2 \leq 3$.

Représentation (2)



Sous-problemes 2+3

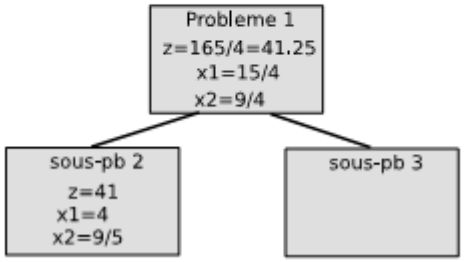
Récursion

- On constate qu'on a, ce faisant, éliminé la région contenant l'optimum de LP.
- On a maintenant deux problèmes qu'on peut de nouveau résoudre en LP.
- On en choisi un arbitrairement parmi ceux non résolus, par exemple ici le problème 2.
- La solution de relaxation (LP) pour la région 2 est

$$\begin{aligned} z &= 41 \\ x_1 &= 4 \\ x_2 &= \tfrac{9}{5} \end{aligned}$$

  Correspondant au point C.
- On a créé la première branche d'un *arbre*.
- Comme x_2 est toujours fractionnaire, on décide de séparer sur cette variable, on sépare donc la région 2 entre deux zones : celle pour $x_2 \geq 2$ et celle pour $x_2 \leq 1$.

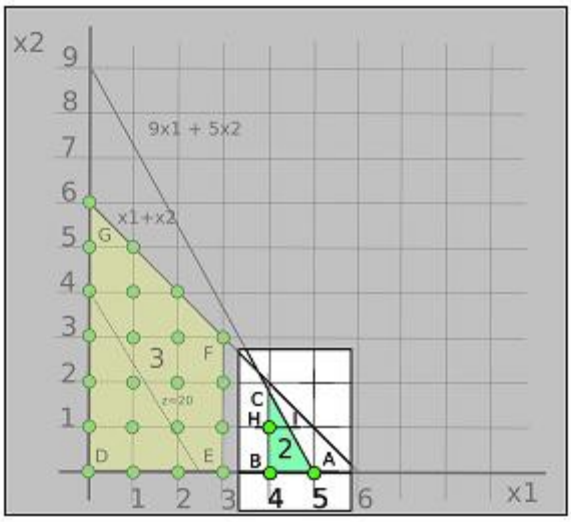
Arbre (1+2)



Nouvelle séparation (4 + 5)

- Nous voici avec deux nouveaux sous-problèmes
  4. Problème 2 + contrainte $x_2 \geq 2$
  5. Problème 2 + contrainte $x_2 \leq 1$.
- Il est préférable d'explorer l'arbre en profondeur d'abord (on verra plus tard pourquoi), on choisi donc de regarder l'un ou l'autre des nouveaux sous-problèmes, par exemple le sous-problème 4. Or ce problème n'est pas réalisable en entiers.

Représentation (4 et 5)



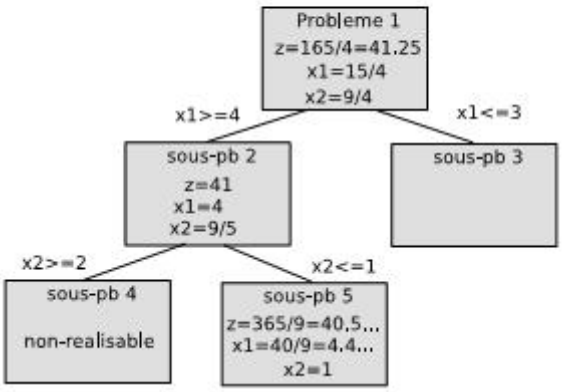
Évaluation (4 + 5)

- On constate que le problème 4 (région $x_2 \geq 2$) n'est pas réalisable.
- En résolvant le problème de relaxation LP lié au sous-problème 5, on trouve l'optimum *I* avec

$$\begin{aligned} z &= 365/9 = 40.555\ldots \\ x_1 &= \tfrac{40}{9} = 4.444\ldots \\ x_2 &= 1 \end{aligned}$$

- il faut donc de nouveau séparer sur x_1, avec les contraintes
  6. Problème 5 + contrainte $x_1 \geq 5$
  7. Problème 5 + contrainte $x_2 \leq 4$.

Arbre (4+5)



Séparation (6+7)

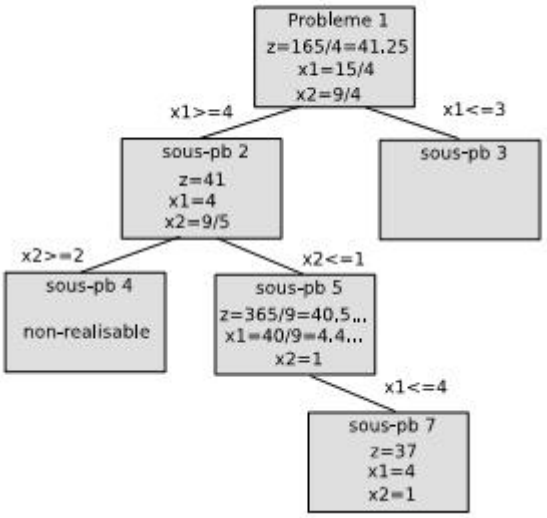
- Nous voici avec deux nouveaux problèmes, comme d'habitude, on va éliminer la solution fractionnaire en séparant de part et d'autre.
- Nous procédons en profondeur d'abord parmi les nouveaux sous-problèmes non-résolus (6 et 7). Nous choisissons arbitrairement 7.
- La solution de relaxation LP est maintenant :

$$\begin{aligned} z &= 37 \\ x_1 &= 4 \\ x_2 &= 1 \end{aligned}$$

Cette solution est réalisable et correspond au point *H*, c'est une *solution candidate* qui nous donne une *borne inférieure* sur notre résultat final.

- Il est inutile de continuer à séparer sur cette branche.

Arbre (7)



Evaluation (6)

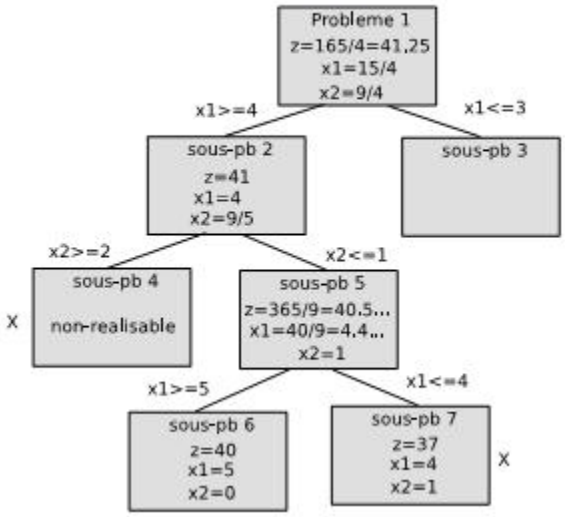
- On continue à évaluer en profondeur d'abord, soit maintenant le sous-problème 6.
- On trouve la solution correspondant au point *A* :

$$\begin{aligned} z &= 40 \\ x_1 &= 5 \\ x_2 &= 0 \end{aligned}$$

Il s'agit également d'une solution réalisable candidate. La valeur de la borne inférieure de notre problème est donc maintenant 40.

- La solution 7 n'est donc pas optimale.
- Il est inutile de séparer davantage sur 6.

Arbre (6)



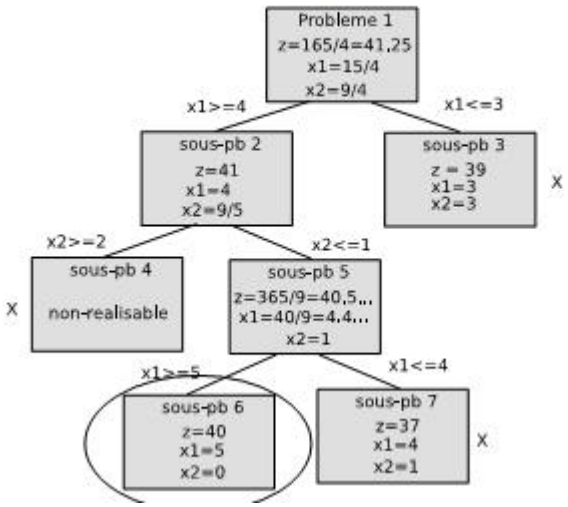
Évaluation (3)

- Il reste à évaluer la solution pour (3). On trouve la solution correspondant au point *F* :

$$\begin{aligned} x_1 &= 3 \\ x_2 &= 3 \\ z &= 39 \end{aligned}$$

- Ce résultat est inférieur à 40, notre borne inf, donc cette branche de l'arbre de peut pas produire un meilleur résultat que celui déjà connu correspondant au point *A*.
- Il ne reste plus de nœud de l'arbre à explorer, on a donc trouvé notre optimum en nombre entier : fabriquer 5 armoires et 0 table pour 40€.

Arbre (3)



5. vvv

Exercice 2 Problème du stockage

Considérons n programmes $P_1, P_2, \ldots, P_n$ qui peuvent être stocker sur un disque dur de capacité D gigabytes.

— Chaque programme P_i a besoin s_i gigabytes pour être stocké et a une valeur v_i

— Tous les programmes ne peuvent pas être stockés sur le disque : $\sum_{i=1}^{n} s_i > D$.

Les programmes stockés dans le disque dur doivent maximiser la valeur totale, sans dépasser la capacité du disque dur. L'objectif est de concevoir un algorithme qui permet de calculer un tel ensemble.

Nous allons construire un tableau T dans lequel les lignes seront indexées par les programmes et les colonnes par les valeurs. L'élément $T[i, j]$ représentera la valeur maximale pour un disque dur de capacité j à l'aide des i premiers programmes.

3) Donner la formule de récurrence.

4) Donner l'algorithme utilisant la programmation dynamique.

Corrigé Donner la formule de récurrence.

Correction

Soit OPT une solution optimale avec un disque dur de capacité D et un ensemble $\mathcal{P}$ de n éléments. Notons $T[i, j]$ la valeur maximale pour un disque dur de capacité j à l'aide des i premiers objets.

1. Si le $n$ième élément appartient à la solution optimale, cela signifie, que la solution optimale privée du $n$ième élément est aussi une solution optimale avec un disque dur de capacité $D - s_n$ et un ensemble $\mathcal{P}$ privé du $n$ième élément.

$$T[n, D] = v_n + T[n-1, D - s_n]$$

2. Si le $n$ième élément n'appartient pas à la solution optimale, cela signifie, que la solution optimale OPT est aussi une solution optimale avec un disque dur de capacité D et un ensemble $\mathcal{P}$ privé du $n$ième élément.

$$T[n, D] = T[n-1, D]$$

Donc par récurence on peut en déduire pour tout $i \in \{1, \ldots, n\}$

Si $j < v_i$, alors $T[i, j] = T[i-1, j]$ sinon $T[i, j] = max(T[i-1, j], T[i-1, j - s_i] + v_i)$

On supposera que $\forall j \in \{1, \ldots, D\}$, on a $T[0, j] = 0$

Donner l'algorithme utilisant la programmation dynamique.

Correction

Entrée : un ensemble d'objects $\mathcal{P} = \{1, \ldots, n\}$. L'objet i a une valeur v_i et un poids s_i.

Sortie : un entier

1. Intialiser tous les éléments du tableau T à zéro.
2. Pour tout i allant de 1 à n
   (a) Pour tout j allant de 1 à D

$$2\text{ (a).1 Si } j < v_i, \text{ alors } T[i,j] = T[i-1,j]$$
$$\text{sinon } T[i,j] = max(T[i-1,j], T[i-1, j-s_i] + v_i)$$

3. Retourner $T[n, D]$

□

Exercice 3 : (7points=1+2+1+2+1)

On s'intéresse à un distributeur automatique de boissons. L'utilisateur insère des pièces de monnaie pour un total de *T* centimes d'Euros, puis il sélectionne une boisson, dont le prix est de *P* centimes d'Euros (*T* et *P* étant des multiples de 10). Il s'agit alors de calculer la monnaie à rendre, sachant que le distributeur a en réserve *E2* pièces de 2 €, *E1* pièces de 1 €, *C50* pièces de 50 centimes, *C20* pièces de 20 centimes et *C10* pièces de 10 centimes.

3. Modélisez ce problème sous la forme d'un problème de Satsisfaction de contraintes.

Pour cela, identifier les variables (les inconnues du problème), les domaines de valeur de ces variables, et les contraintes existant entre ces variables.

4. Comment pourrait-on exprimer le fait que l'on souhaite que le distributeur rende le moins de pièces possibles ?

Solution :

Pour modéliser ce problème sous la forme d'un CSP, il s'agit d'identifier les variables (les inconnues du problème), les domaines de valeur de ces variables, et les contraintes existant entre ces variables. Ici, *T, P, E2, E1, C50, C20* et *C10* sont des "données" du problème (correspondant aux paramètres en entrée). Ce que l'on doit déterminer (nos inconnues), c'est la quantité de pièces de 2 et 1 Euro, ainsi que de 50, 20 et 10 centimes à rendre. On a donc 5 variables. Pour modéliser notre problème sous la forme d'un CSP *(X,D,C)*, vous devez

- donner un nom à chacune de ces variables, et définir *X* comme étant l'ensemble de ces 5 variables ;
- définir pour chacune de ces 5 variables son domaine de valeur, sachant que la quantité de pièces retournées, pour un type de pièce donné, est comprise entre 0 et le nombre de pièces de ce type que l'on a en réserve ;
- définir les contraintes (il n'y en a qu'une... elle spécifie que la somme à retourner doit être égale à la somme insérée moins le prix à payer).

On définit le CSP *(X,D,C)* tel que

- *X = {XE2, XE1, XC50, XC20, XC10}*

où *XE2* désigne le nombre de pièces de 2 Euros à retourner, *XE1* désigne le nombre de pièces de 1 Euro à retourner, ...

- Les domaines spécifient que la quantité de pièces retournées, pour un type de pièce donné, est comprise entre 0 et le nombre de pièces de ce type que l'on a en réserve :

D(XE2) = {0,1,...,E2}
D(XE1) = {0,1,...,E1}
D(XC50) = {0,1,...,C50}
D(XC20) = {0,1,...,C20}
D(XC10) = {0,1,...,C10}

- Les contraintes spécifient que la somme à retourner doit être égale à la somme insérée moins le prix à payer :

*C = { 200*XE2 + 100*XE1 + 50*XC50 + 20*XC20 + 10*XC10 = T-P }*

Dans cette modélisation, nous avons utilisé une contrainte arithmétique linéaire sur les entiers. Cette contrainte est globale dans le sens où elle fait intervenir toutes les variables du problème. Une autre modélisation possible consisterait à définir le domaine des variables par l'ensemble des entiers positifs ou nuls, et d'ajouter en contraintes que la quantité de pièces retournées doit être inférieure ou égale au nombre de pièces en réserve (pour chaque type de pièce différent).

Pour exprimer le fait que l'on souhaite que le distributeur rende le moins de pièces possibles, on pourrait ajouter à ce CSP une fonction "objectif" à minimiser

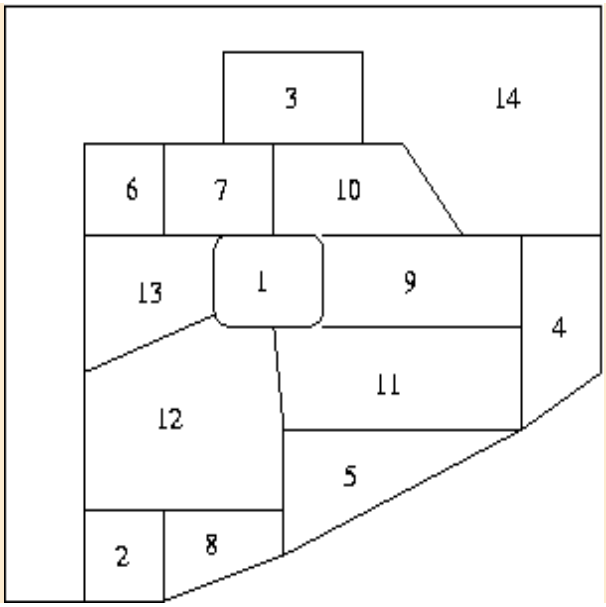
f(X) = XE2 + XE1 + XC50 + XC20 + XC10

Dans ce cas, résoudre le CSP revient à chercher une affectation de *X* complète et consistante qui minimise *f(X)*.

Exercice 3 : (8points =0,5+1+0,5+1+1+1+1+1+1+1)

Exercice 2 : Coloriage d'une carte

Il s'agit de colorier les 14 régions de la carte ci-dessous, de sorte que deux régions ayant une frontière en commun soient coloriées avec des couleurs différentes. On dispose pour cela des 4 couleurs suivantes : bleu, rouge, jaune et vert.



Modélisez ce problème sous la forme d'un CSP.

Solution :

Pour modéliser ce problème sous la forme d'un CSP, il s'agit d'identifier les variables (les inconnues du problème), les domaines de valeur de ces variables, et les contraintes existant entre ces variables.

Ici, ce que l'on doit déterminer (nos inconnues), c'est la couleur de chaque région. On aura donc une variable pour chaque région, chacune de ces variables pouvant prendre comme valeur une des 4 couleurs.

Les contraintes spécifient que deux régions voisines ne doivent pas être de la même couleur.

Corrigé de l'exercice 2

On définit le CSP *(X,D,C)* tel que

- *X = {X1, X2, ..., X14}*

  (On associe une variable *Xi* différente par région *i* à colorier.)

- *pour tout Xi élément de X, D(Xi) = { bleu, rouge, vert, jaune }*

  (Chaque région peut être coloriée avec une des 4 couleurs.)

- *C = { Xi ≠ Xj / Xi et Xj sont 2 variables de X correspondant à des régions voisines }*

  (2 régions voisines doivent être de couleurs différentes.)

Pour être plus précis, on peut définir explicitement les relations de voisinage entre régions, par exemple à l'aide d'un prédicat *voisines/2*, tel que *voisines(X,Y)* soit vrai si *X* et *Y* sont deux régions voisines. Ce prédicat peut être défini en extension, en listant l'ensemble des couples de régions ayant une frontière en commun :

voisines(X,Y) <=> (X,Y) élément-de {(1,7), (1,9), (1,10), (1,11), (1,12), (1,13), (2,8), (2,12), (2,14), (3,7), (3,10), (3,14), (4,9), (4,11), (4,14), (5,8), (5,11), (5,12), (6,7), (6,13), (6,14), (7,1), (7,3), (7,6), (7,10), (7,13), (7,14), (8,2), (8,5), (8,12), (9,1), (9,4), (9,10), (9,11), (10,1), (10,3), (10,7), (10,9), (10,14), (11,1), (11,4), (11,5), (11,9), (11,12), (12,1), (12,2), (12,5), (12,8), (12,11), (12,13), (12,14), (13,1), (13,6), (13,7), (13,12), (13,14), (14,2), (14,3), (14,4), (14,6), (14,7), (14,10), (14,12), (14,13)}
on peut alors définir l'ensemble des contraintes *C* de la façon suivante :

C = { Xi ≠ Xj / Xi et Xj sont 2 variables différentes de X et voisines(Xi,Xj) = vrai }

Ce problème de coloriage d'une carte est un cas particulier du problème du coloriage des sommets d'un graphe (deux sommets adjacents du graphe doivent toujours être de couleurs différentes). De nombreux problèmes "réels" se ramènent à ce problème de coloriage d'un graphe : problème des examens, d'emploi du temps, de classification, ...