Intelligence Artificielle : Logique et Contraintes - Examen - session n° 1

Date : 13/03/2018 **Durée :** 2h30
Documents autorisés : 1 copie double MANUSCRITE de résumé de cours à votre initiative
Barème : 3 + 5 + 6 + 6 (le barème est juste est indicatif)
Il est conseillé de lire le sujet dans son intégralité avant de commencer.
Les exercices sont indépendants. Vous pouvez les traiter dans l'ordre de votre choix.
Toute réponse doit être **justifiée**, sinon elle ne sera pas prise en compte.

Exercice 1 (Questions de cours)
Répondre en quelques lignes (5 maxi) aux questions suivantes :

1. Un CSP qui est 2-cohérent est-il nécessairement 3-cohérent ?
   Un CSP qui est 3-cohérent est-il nécessairement 2-cohérent ?
   Prouvez le ou donnez un contre exemple.
2. Expliquez pourquoi, lors de la résolution d'un CSP par un algorithme de recherche, choisir d'affecter en priorité la variable la plus contrainte avec la valeur la moins contrainte est une bonne heuristique.
3. Décrire en quoi consiste la contrainte globale `element(...)` et illustrez son fonctionnement sur un exemple pertinent.

Exercice 2 (Formalisation de problème)

Quatre jeunes juges tout juste sortis de l'école nationale de la magistrature se retrouvent à l'occasion d'une rencontre amicale et échangent leurs points de vue sur leur première année d'expérience. Ils étaient tous quatre de très bons étudiants et occupaient les 4 premières places dans le classement de sortie de l'école, ce qui leur a plus ou moins permis de choisir le lieu de leur première affectation. On sait qu'au moment de leur candidature, ils étaient tous âgés de 27 à 29 ans et que deux d'entre eux avaient 28 ans. On sait également que :

a) André à 29 ans.
b) Lucie était moins bien classée que le candidat affecté à Paris.
c) Les deux premiers ont choisi une affectation dans des villes au sud de la France (Lyon ou Marseille).
d) Les deux ayant 28ans se suivaient dans le classement final.
e) Didier, qui a 28 ans était moins bien classé que celui qui est allé à Lyon.
f) Ce n'est pas Claire qui est allée à Lyon.
g) Le plus jeune candidat était classé au moins 2 rangs derrière Claire et s'est retrouvé affecté à Lille.

1. Formaliser l'énoncé précédent sous forme d'un problème de satisfaction de contraintes. Bien préciser le sens de vos variables et comment vous modélisez les différentes parties de l'énoncé. Vous ferez attention à ne pas utiliser de contraintes disjonctives.
2. Dessinez le graphe de contraintes de ce problème.
3. Ce CSP est-il cohérent par Sommet ? cohérent par arc ? Justifiez votre réponse
4. Trouver une solution pour ce problème. Vous pouvez procéder comme vous le souhaitez... mais expliquez et détaillez votre manière d'obtenir cette solution.

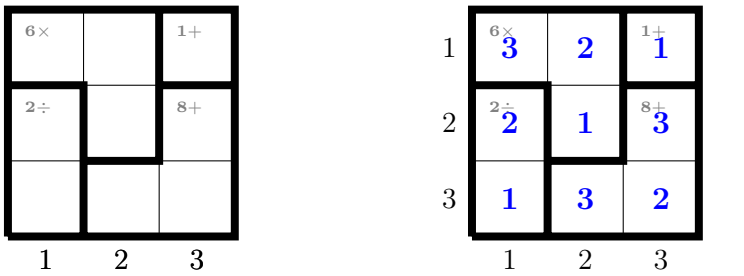
Exercice 3 On considère le CSP $\mathcal{P} = (X, D, C)$ caractérisé par :

- $X = \{A, B, C, D, E, F, G, H, I\}$
- $D = d^9$, avec $d_x = 1..3$,
- $C = \{\ C \neq E,\ E \neq 2,\ D \neq G,\ A \geq H + 2, B = I,$
  $A \neq B, A \neq C, B \neq C, D \neq E, D \neq F, E \neq F, G \neq H, G \neq I, H \neq I\ \}$

1. Dessinez le graphe de contraintes correspondant à $\mathcal{P}$
2. On suppose que l'on recherche les solutions de ce problème en utilisant l'algorithme du backtrack standard (BT) et l'on étudie le contexte de l'affectation partielle $A = \{< F, 2 >, < D, 1 >, < E, 3 >, < C, 1 >, < B, 2 >\}$. Construire la portion de l'arbre de recherche développé par BT et correspondant au prolongement de A (et juste cette portion) en affectant les variables restantes dans l'ordre I, G, H, A. Pour préciserez bien à chaque étape, quelles sont les contraintes testées et le résultat de chaque test, comme indiqué en td.
3. On souhaite à présent trouver **toutes** les solutions de ce problème à l'aide d'un algorithme de recherche hybride. Après avoir traité les contraintes unaires de façon adéquate, appliquez l'algorithme du *forward checking* (FC) en affectant les variables dans l'ordre $B, A, C, H, I, G, D, E, F$. Vous indiquerez bien l'ordre d'exploration des noeuds de l'arbre de recherche et détaillerez les propagation effectuées à chaque étape.
4. Si avant de lancer la recherche précédente on vous permettait de réviser quelques arcs (5 au maximum), lesquels choisiriez vous ? Justifiez votre choix et précisez l'impact que cela aurait sur l'arbre précédent.

Exercice 4 (Opl - Jeu Kenken)

On s'intéresse ici au jeu du Kenken, dans lequel il faut trouver une façon remplir une grille de taille $n * n$ avec des nombres de 1 à n en tenant compte de certaines contraintes. La grille a pour particularité d'être divisée en différentes zones constituées de cases contigües. Pour chaque zone, la case la plus haute et la plus à gauche de la zone contient un chiffre et un symbole d'opération arithmétique (cf Fig 1a).



(a) la grille énoncé (b) ... et sa solution

Figure 1 – Une instance de Kenken 3×3

.

Le but est alors de remplir la grille de telle façon que les nombres situées sur une même ligne ou sur une même colonne soient tous différents et que pour chaque zone, il existe une façon d'ordonner les cases de la zone de manière à ce que la valeur indiquée dans la case en haut à gauche, corresponde au résultat d'une expression arithmétique (sans parenthèses) n'utilisant que l'opérateur indiqué. La figure 1b décrit une solution possible pour le problème correspondant à la grille 1a. On peut bien vérifier que l'on a bien $6 = 3 \times 2 \times 1$, $2 = 2 \div 1$ et $8 = 3 + 2 + 3$ (on admettra que lorsqu'une zone est réduite à une case, il n'y a rien à vérifier).

Afin de concevoir un solveur générique capable de résoudre n'importe quel problème de ce type, un certain nombre de structures de données ont été introduites dans le but de faciliter la description d'une instance de problème.

```
// ---- Structures pour la description d'une instance Kenken -------
tuple OpRes {
   string operateur;    // l'opérateur à appliquer ("+" "-" "*" ou "/")
   int resultat;        // le résultat de l'opération
}

tuple Case {
   int l;    // ligne
   int c;    // colonne
}
```

Chaque instance du problème est alors décrite, par deux entiers décrivant respectivement la taille de la grille et le nombre de zones différentes) et deux tableaux, décrivant pour chaque zone d'une part, l'ensemble des cases définissant la zone, et d'autre part l'opérateur à appliquer sur les cases de cette zone et le résultat qui doit être obtenu.

Par exemple l'instance de problème correspondant à la figure 1a peut à présent être décrite par :

```
n = 3;
nbZones=4;
casesZone = [
              {<1,1>,<1,2>,<2,2>},
              {<1,3>},
              {<2,1>,<3,1>},
              {<2,3>,<3,2>,<3,3>}
              ];
res = [
       <"*",6>,
       <"+",1>,
       <"/",2>,
       <"+",8>
      ];
```

On souhaite à présent concevoir un modèle Opl permettant de résoudre n'importe quel problème de ce type, décrit sous ce format .

1. Quelles sont les variables/domaine(s) à introduire pour pouvoir formaliser ce problème ?
2. Ecrire le code Opl permettant de modéliser ce problème.
   NB : Ne pas détailler les contraintes dans cette question (cf question suivante).
3. Décrire le code modélisant les contraintes à respecter (vous ferez attention attention au fait que les opérateurs ne sont pas tous commutatifs).