Examen HAI710I – Fondements de l'IA symbolique

Session 2 - 27 mars 2023

Durée 2h - Aucun document autorisé

Barème indicatif (sur 21) : ex. 1 (4 points), ex. 2 (4 points), ex. 3 (4 points), ex. 4 (3 points), ex. 5 (4 points), ex. 6 (2 points).

Exercice 1. Recherche heuristique

Répondez de manière précise et concise aux 8 questions suivantes (3-4 lignes par question max) :

On s'intéresse à un problème de recherche sur un **espace infini** d'états disposant d'une fonction de coût g d'une solution définie comme la somme du coût c de chaque action permettant d'arriver à la solution. On suppose ici que le coût de toute action est supérieur ou égal à 1.

Pour chacune des questions suivantes de complétude, vous justifierez votre réponse :

1. Est-on sûr de trouver une solution (s'il en existe une) avec la recherche en largeur ?
2. Est-on sûr de trouver une solution (s'il en existe une) avec la recherche en profondeur ?
3. Est-on sûr de trouver une solution (s'il en existe une) avec la recherche de coût min ?

On dispose de plus d'une heuristique h monotone pour ce problème.

4. Précisez ce qu'est une heuristique monotone.
5. Peut-on dire que h est admissible ?

Pour chacune des questions suivantes sur l'optimalité de la recherche, vous justifierez votre réponse :

6. Est-on sûr de trouver une solution optimale (s'il en existe une) avec la recherche de coût min ?
7. Est-on sûr de trouver une solution optimale (s'il en existe une) avec la recherche gloutonne ?
8. Est-on sûr de trouver une solution optimale (s'il en existe une) avec la recherche A* (précisez si vous considérez ou non la détection des états répétés) ?

Exercice 2. Modélisation CSP

On souhaite développer un logiciel de planification de session d'examens pour un centre de formation. Ce centre de formation organise ses examens par groupe d'étudiants partageant les mêmes matières. Chaque groupe d'étudiants doit passer 3 épreuves d'examen. Une épreuve d'examen occupe 1 groupe d'étudiants, sur 1 matière, dans 1 salle, pendant 1 demi-journée et est surveillée par 1 enseignant.

On souhaite planifier la session de printemps sur 3 journées (donc 6 demi-journées : D_1, D_2, D_3, D_4, D_5, D_6) avec les données suivantes :

- il y a 4 groupes d'étudiants G_1, G_2, G_3 et G_4 qui ont les épreuves suivantes :
  - G_1 doit passer les épreuves de Math, Info et Physique,
  - G_2 doit passer les épreuves de Math, Physique et Chimie,
  - G_3 et G_4 doivent passer les épreuves de Physique, Chimie et Biologie,
- il y a 3 salles disponibles pour chaque demi-journée : S_1, S_2, S_3 ;
- il y a 5 enseignants susceptibles de surveiller : E_1, E_2, E_3, E_4, E_5 ;
- on souhaite que les enseignants ne surveillent que des épreuves qu'ils enseignent et :

— E_1 et E_2 enseignent Math et Physique,
— E_3 enseigne Math et Info,
— E_4 enseigne Physique et Chimie,
— E_5 enseigne la Biologie.

On a donc à décider de la date, de la matière, de la salle et du surveillant pour chacune des 12 épreuves d'examen : Ep_iG_j pour i allant de 1 à 3 (3 épreuves d'examen pour chaque groupe) et j allant de 1 à 4 (4 groupes d'étudiants).

— les bonnes matières pour chaque groupe ;
— pas deux examens pour un groupe au même moment ;
— pas d'enseignant surveillant une matière qu'il n'enseigne pas ;
— pas deux groupes dans la même salle au même moment ;
— pas deux surveillances pour un enseignant au même moment.

Modéliser par un réseau de contraintes ce problème de planification de la session d'examens de printemps, c'est-à-dire définit l'ensemble X de variables, les domaines $D(x)$ de chaque variable $x \in X$, et l'ensemble C de contraintes permettant d'obtenir une planification correcte. Vous préciserez le nombre de contraintes créées, l'arité des contraintes et le nombre de tuples par contraintes.

Exercice 3. Résolution de CSP

Soit le réseau de contraintes (X, D, C) binaires où :

— $X = \{v, w, x, y, z\}$
— $D(v) = D(w) = D(x) = D(y) = D(z) = \{a, b, c\}$
— $C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$

c_1

x	y
b	c
b	a
c	b
a	c

c_3

w	z
c	b
a	c
a	a

c_5

x	v
b	c
c	a
a	b
a	c

c_2

y	z
b	c
b	a
c	b
c	c
a	c

c_4

v	w
b	c
c	c
c	a
a	b

c_6

w	y
b	a
c	b
c	c
a	b

1. Appliquez l'algorithme de backtrack à la recherche d'une solution en prenant l'ordre alphabétique pour les variables et les valeurs. Vous dessinerez l'arbre de recherche en indiquant les contraintes violées.
2. Quelle est cette première solution trouvée ?
3. Calculez la fermeture arc-consistante de ce réseau.
4. Appliquez l'algorithme de Forward Checking pour trouver <u>toutes les solutions</u> sur cette fermeture arc-consistante en utilisant l'heuristique (dynamique) *dom* + *deg* + *alpha* pour l'ordre des variables et l'ordre alphabétique pour les valeurs. Vous préciserez bien à chaque étape comment les domaines de chaque variable évoluent. On rappelle que *dom* consiste à sélectionner en priorité la variable ayant le plus petit domaine (dynamique), que *deg* consiste à sélectionner en priorité la variable impliquée dans le plus grand nombre de contraintes et *alpha* correspond à l'ordre alphabétique ; *dom* + *deg* + *alpha* consiste à utiliser *dom* et en cas d'égalité *deg* pour départager puis, si encore égalité, *alpha*.
5. Donnez toutes les solutions trouvées.

Exercice 4. Règles propositionnelles positives

Soit la base de faits BF $= \{A, B\}$ et la base de règles BR suivante :

$R1 : B \wedge C \rightarrow D$
$R2 : D \wedge A \rightarrow G$
$R3 : B \wedge I \rightarrow H$
$R4 : G \wedge F \rightarrow H$
$R5 : A \wedge H \rightarrow I$
$R6 : A \wedge G \rightarrow H$
$R7 : F \rightarrow E$
$R8 : A \rightarrow C$

1. Utilisez l'algorithme *à base de compteurs* pour calculer BF^*, la base de faits saturée par les règles. Indiquer : (1) l'état des compteurs à la fin de l'algorithme, et (2) le contenu de BF^*.
2. Votre réponse à la question 1 vous permet-elle de dire si le symbole H est prouvable en chaînage arrière ? Pourquoi ?
3. Essayez de prouver H en chaînage arrière. Dessinez l'*arbre de recherche* construit par un algorithme qui considère les règles par numéro croissant et les symboles par ordre d'apparition dans l'hypothèse de la règle considérée. Vous indiquerez sur chaque feuille traitée : échec, (appartient à) BF, déjà prouvé, boucle. Finalement, H est-il prouvé ?

Exercice 5. Règles propositionnelles avec négation par défaut

On considère des règles avec négation par défaut (ou monde clos). On rappelle qu'une dérivation menant d'une base de faits BF à une certaine base de faits BF^i est dite *persistante* si aucune règle de la dérivation n'est bloquée dans BF^i (autrement dit, pour toute règle $H+, H^- \rightarrow C$, on a $H^- \cap BF^i = \emptyset$). Une base de faits BF^i est *saturée* si aucune règle ne s'applique de façon utile sur BF^i (autrement dit, aucun nouvel atome ne peut être produit).

Soit BR la base de règles suivante :

$R_1 : \cancel{U} \wedge P \rightarrow T$
$R_2 : T \wedge not\ P \rightarrow S$
$R_3 : T \wedge not\ S \rightarrow Q$
$R_4 : S \rightarrow R$
$R_5 : \cancel{U} \rightarrow P$
$R_6 : R \wedge P \rightarrow S$

1. Soit la base de connaissances (BF, BR) avec $BF = \{T\}$. Quelles sont toutes les bases de faits saturées que l'on peut obtenir avec des dérivations complètes ? Parmi ces bases de faits saturées, lesquelles peuvent être obtenues par des dérivations persistantes ?
2. Calculez le graphe de précédence des symboles associé à BR. On rappelle que seuls les symboles apparaissant en conclusion de règle sont pertinents. Comment ce graphe vous permet-il que conclure que BR est stratifiable ?
3. Donnez une stratification de BR en vous appuyant sur le graphe de la question précédente.
4. Qu'appelle-t-on "dérivation qui suit une stratification" ?
5. Quelle est la propriété fondamentale assurée par le fait qu'une base de règles soit *stratifiable* ?
6. Partant de BF $= \{U\}$, quelle base de faits saturée est obtenue par une dérivation qui suit la stratification que vous avez donnée à la question 3 ?

Exercice 6. Règles du premier ordre avec négation par défaut

On considère une base de faits généalogiques contenant les prédicats *femme/1*, *homme/1* et *aEnfant/2*. Intuitivement, *aEnfant(X, Y)* signifie que X a pour enfant Y. En utilisant ces prédicats, donner une ou plusieurs règles définissant le prédicat *pasDeuxParentsConnus/1* tel que *pasDeuxParentsConnus*(X) est vrai si X n'a pas ses deux parents connus. Vous pouvez définir d'autres prédicats si nécessaire. Vous pouvez utiliser la syntaxe logique classique ou celle de la programmation logique (Clingo).