Recherche heuristique, représentation et résolution de connaissances
Corrigé

1. **Modélisation états - sur papier** *Le problème des 8 dames* : placer 8 dames d'un jeu d'échecs sur un échiquier 8x8 sans qu'elles se menacent mutuellement (donc deux dames ne doivent jamais partager la même rangée, colonne, ou diagonale).
   Modélisez le problème de 8 dames en détaillant l'espace d'états, les actions, le modèle de transition, l'état initial, le test de réussite ou les états but (plusieurs solutions sont possibles). Détaillez les premiers deux niveaux de votre arbre de recherche après la racine à partir d'un état initial. Analysez l'arbre de recherche correspondant en calculant le nombre d'états, le facteur de branchage, la profondeur, et le nombre de feuilles.

   Solution :

   Solution 1 - configurations partielle :
   - Etats : toutes les configurations possibles pour placer entre 0 et 8 dames sur l'échiquier.
   - Actions : placer une nouvelle dame dans une case vide (sauf si toutes les 8 dames sont déjà placées, dans ce cas aucune action est possible)
   - Fonction de transition :(définition assez naturelle...)
   - Etat initial : échiquier vide
   - Test de réussite : toutes les dames ont été placée et il n'y a pas de 2 dames sur la même colonne, ni sur la même ligne, ni sur la même diagonale

   Pour l'analyse de l'arbre de recherche :
   - Nb états : 1 état sans dames, 64 états possibles avec une seule dame sur l'échiquier, 64×63 pour placer deux dames...$(64! - 56!)$ états possibles avec 8 dames sur l'échiquier. Donc en total $1 + 64 + 64 \times 63 + \ldots + (64! - 56!)$. C'est dans l'ordre de 64^8 (=nombre de cases puissance nombre de dames).
   - Branchage : 64, car c'est le max nombre de cases vides où placer une dame dans la racine (ensuite le branchage est de 63 au niveau 2, ... jusqu'à 57 avant les feuilles). Donc de l'ordre du nombre de cases
   - Profondeur : 8. Une fois placées les 8 dames je suis dans une feuille (pas d'autres action possible)
   - Nb feuilles : $(64! - 56!)$, toutes les possibilités de placer 8 dames sur l'échiquier

   Observer que dans cette solution il n'y a pas de cycle, donc je peux utiliser DFS en TREE-SEARCH.

   Solution 2 - configurations complètes (type recherche locale) :
   Cette solution genère un arbre très grand, mais c'est intéressant car les solutions peuvent etre plus proches de la racine :

— Etats : toutes les configurations possibles de placer 8 dames sur l'échequier.
— Actions : bouger une dame de sa case à une autre case vide.
— Transition (...)
— Etat initial : une configuration avec les 8 dames sur l'échequier
— Test de réussite : il n'y a pas de 2 dames sur la même colonne, ni sur la même ligne, ni sur la même diagonale

Pour l'analyse de l'arbre de recherche :

— Nb états : 64 !-56 ! donc de l'ordre de nb de cases à la nb de dames
— Branchage : 56×8, car pour l'expansion d'un noeud je choisi une dame et je la place dans une case libre (64 cases moins 8 dames déjà en place).
— Profondeur : cycles possibles, donc c'est nécessaire de se souvenir des états visités. La profondeur est de l'ordre du nombre d'états.
— Nb feuilles (=nb états buts) : 92 (calculé par force brute, voir `https://en.wikipedia.org/wiki/Eight_queens_puzzle`)

2. **Recherche non informée - sur machine** Implementation et comparaison de DFS et de BFS pour le jeu du taquin : voir notebook sur l'espace du cours.
3. **Recherche heuristique - sur papier** Rappellez-vous de la modélisation en espace d'états du jeu de taquin. Considerez les deux heuristiques présentées en cours (slide 37 cours recherche heuristique) :
   — h1 : nombre de tuiles mal placées
   — h2 : distance de Manhattan avec l'état but (pour chaque tuile, compter combien de cases elle doit croiser pour arriver à sa case dans l'état but)

   Déroulez A* sur au moins 3/4 niveaux de l'arbre avec h1 et h2 à partir de l'état initial ci-dessous, et comparez visuellement avec le fonctionnement de BFS ou DFS.

1	2	3
4	5	
6	7	8

Solution :

Voir pdf avec l'arbre de recherche et le calcul des H1 et H2. Si on utilise A* avec H1 on developpe les états avec H1=3 d'abord, puis H1=4. Si on utilise A* avec H2 on developpe d'abord n'importe quel dans la première couche, on obtient que des H2=7 donc on doit continuer avec tous les H2=6 d'abord. Vous pouvez visualiser la frontière au tableau par exemple et projeter le pdf. Vous pouvez visualiser aussi BFS et DFS. L'idée est de montrer que l'heuristique nous permets de rester proche de notre état et explorer d'une facon "moins impulsive" (?)

4. **Recherche heuristique - sur machine** Implementation et comparaison de A* avec deux heuristiques : voir notebook sur l'espace du cours.

5. **SAT et CSP - sur papier** Modélisez le problème de résoudre une grille de Sudoku avec des formules propotionnelles (SAT). Analysez la modélisation en terme de nombre de variables et nombre de formules.

1	2		
		2	1
2	4		
		4	2

Solution :

Modélisation en SAT :

Variables $x_{i,j,k}$ pour i, j, k entre 1 et 4. Signification : $x_{i,j,k} = 1$ si et seulement si la case à la ligne i et colonne j est égale à k. Exemple, dans notre état initial $x_{2,3,2} = 1$ mais $x_{2,3,3} = 0$

- Contraintes de representation de l'état initial : $x_{1,1,1} = 1$ AND $x_{1,2,2} = 1$ AND $x_{2,3,2} = 1$ AND $x_{2,4,1} = 1$ AND $x_{3,1,2} = 1$ AND $x_{3,2,4} = 1$ AND $x_{4,3,4} = 1$ AND $x_{4,4,2} = 1$ (écrire avec des virgules c'est aussi ok, expliquer que toutes les formules qu'on écrit seront connecté avec des AND)
- Solution 1 pour es contraintes pour qu'une case soit remplie avec un et un seul chiffre :
  $x_{i,j,1}$ OR $x_{i,j,2}$ OR $x_{i,j,3}$ OR $x_{i,j,4}$
  $x_{i,j,1}$ ALORS NOT $(x_{i,j,2} OU x_{i,j,3} OU x_{i,j,4})$
  $x_{i,j,2}$ ALORS NOT $(x_{i,j,1} OU x_{i,j,3} OU x_{i,j,4})$
  $x_{i,j,3}$ ALORS NOT $(x_{i,j,1} OU x_{i,j,2} OU x_{i,j,3})$
  $x_{i,j,4}$ ALORS NOT $(x_{i,j,1} OU x_{i,j,2} OU x_{i,j,3})$
  à répéter pour tout i, j
- Solution 2 pour les contraintes pour qu'une case soit remplie avec un et un seul chiffre : On peut utiliser une nouvelle definition : p XOR q = (p AND NOT q) OR (NOT p AND q). Cela veut dire qu'une et une seule de deux variables peut etre vraie. On écrit XOR(p,q,r...) pour une et une seule variables entre p, q, r.. doit etre vraie. La contrainte qu'il y aille un et un seul valeur pour la cellule i, j devient donc :

$$XOR(x_{i,j,1}, x_{i,j,2}, x_{i,j,3}, x_{i,j,4})$$

- Solution 3 pour les contraintes pour qu'une case soit remplie avec un et un seul chiffre : enumérer toutes les affectations possibles pour chaque case :
  ($x_{i,j,1}$ AND NOT $(x_{i,j,2} OU x_{i,j,3} OU x_{i,j,4})$ OU
  ($x_{i,j,2}$ AND NOT $(x_{i,j,1} OU x_{i,j,3} OU x_{i,j,4})$ OU
  ($x_{i,j,3}$ AND NOT $(x_{i,j,1} OU x_{i,j,2} OU x_{i,j,4})$ OU
  ($x_{i,j,4}$ AND NOT $(x_{i,j,1} OU x_{i,j,2} OU x_{i,j,3})$ OU
- Contraintes pour qu'il n'y ait pas de meme chiffre sur la meme ligne i, ce qui est équivalent à demander qu'il y ait toutes les chiffres sur chaque ligne :
  $x_{i,1,1}$ OR $x_{i,2,1}$ OR $x_{i,3,1}$ OR $x_{i,4,1}$ AND
  $x_{i,1,2}$ OR $x_{i,2,2}$ OR $x_{i,3,2}$ OR $x_{i,4,2}$ AND
  $x_{i,1,3}$ OR $x_{i,2,3}$ OR $x_{i,3,3}$ OR $x_{i,4,3}$ AND
  $x_{i,1,4}$ OR $x_{i,2,4}$ OR $x_{i,3,4}$ OR $x_{i,4,4}$
  à instantier pour toutes les lignes i

— Contraintes pour qu'il n'y ait pas de meme chiffre sur la meme colonne j ce qui est équivalent à demander qu'il y ait toutes les chiffres sur chaque colonne :
$x_{1,j,1}$ OR $x_{2,j,1}$ OR $x_{3,j,1}$ OR $x_{4,j,1}$ AND
$x_{1,j,2}$ OR $x_{2,j,2}$ OR $x_{3,j,2}$ OR $x_{4,j,2}$ AND
$x_{1,j,3}$ OR $x_{2,j,3}$ OR $x_{3,j,3}$ OR $x_{4,j,3}$ AND
$x_{1,j,4}$ OR $x_{2,j,4}$ OR $x_{3,j,4}$ OR $x_{4,j,4}$
à instantier pour toutes les colonnes j

— Contraintes pour qu'il n'y ait pas de meme chiffre sur le meme carré, ce qui est équivalent à demander qu'il y ait toutes les chiffres dans chaque carré :
$x_{1,1,1}$ OR $x_{1,2,1}$ OR $x_{2,1,1}$ OR $x_{2,2,1}$ AND
$x_{1,1,2}$ OR $x_{1,2,2}$ OR $x_{2,1,2}$ OR $x_{2,2,2}$ AND
$x_{1,1,3}$ OR $x_{1,2,3}$ OR $x_{2,1,3}$ OR $x_{2,2,3}$ AND
$x_{1,1,4}$ OR $x_{1,2,4}$ OR $x_{2,1,4}$ OR $x_{2,2,4}$
à adapter pour toutes les autres trois carrés

Important ! Pour terminer il faut expliquer que les SAT solvers veulent les formules en CNF (des conjonctions de clauses, où une clause est une disjonction de litteraux). C'est une terminologie qu'ils retrouverons dans le TP.

6. **Extra.** Modélisez le problème de résoudre une grille de Sudoku en détaillant l'espace d'états, les actions, le modèle de transition, les états but (plusieurs solutions sont possibles). Détaillez les premiers deux niveaux de l'arbre de de recherche après la racine à partir de l'état initial ci-dessous. Analysez l'arbre de recherche correspondant en calculant le nombre d'états, le facteur de branchage, la profondeur, et le nombre de feuilles.

Solution :

Modélisation comme espace d'états :

— Etats : toutes configurations partielles de 1,2,3,4 sur la grille.
— Actions : remplir une case vide avec un élément de 1,2,3,4
— Transition (...)
— Etat initial : la configuration ci-dessous
— Test de réussite : pas de meme chiffre sur la même ligne, pas de meme chiffre sur la même colonne, pas de même chiffre dans chacun de quatre sous-carrés

Pour l'analyse de l'arbre de recherche :

— Nb états : 4^16
— Branchage : 16×4
— Profondeur : 16
— Nb feuilles : 4^{16}

L'alternative des configurations complètes est aussi possible ici (style recherche locale). L'arbre est donc plus large et plus profond, mais les solutions peuvent être plus proches de la racine.

7. **Extra.** Modélisez la recherche d'un chemin Hamiltonien dans le graphe suivant en détaillant l'espace d'états, les actions, le modèle transition, l'état but. Analysez l'arbre de recherche correspondant. Devéloppez des heuristiques admissibles. Modélisez le problème comme un SAT et CSP. Implementez BFS, DSF, A*, SAT, CSP et comparez les temps d'execution sur un graphe de taille suffisamment grande.