TP1 – IFT3335
Jeu de Sudoku
(à rendre au plus tard le lundi 19 mars avant 23 :59)

1. Buts du TP

1. Comprendre comment on peut formuler un problème d'IA comme un problème de recherche dans l'espace d'états
2. Comprendre comment implanter des algorithmes de recherche et des heuristiques

2. Le jeu de Sudoku

Ce jeu d'origine japonaise est représenté en une grille de 9x9, séparée en 9 carrés de 9 cases. Certaines cases contiennent déjà un chiffre au départ, de 1 à 9. Le but est d'arriver à remplir les cases vides de telle façon que chaque carré, chaque ligne et chaque colonne contiennent les chiffres 1-9 sans répétition. Autrement dit, on ne doit pas trouver 2 chiffres identiques dans un même carré, une ligne ou une colonne.

Voici une configuration de départ (à gauche) et sa solution (à droite) :

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Ce jeu est l'objet de nombreuses recherches, et beaucoup d'heuristiques ont été développées pour trouver des solutions de façon efficace. Vous pouvez trouver une bonne description du jeu et des heuristiques sur la page de Wikipédia :
http://fr.wikipedia.org/wiki/Sudoku
Une autre page Wikipedia en anglais donne une description plus courte, mais plus utile pour votre implantation dans ce TP :
https://en.wikipedia.org/wiki/Sudoku_solving_algorithms

3. Le cadre de base pour votre travail

Le but de ce TP n'est pas de créer un programme qui résout un Sudoku en un temps record, mais de pratiquer l'implantation de différents algorithmes de recherche dans l'espace d'états, et de comparer leur performance. Sudoku sert de support pour cette fin.
Vous pouvez trouver une implantation de base par Norvig ici :
http://www.norvig.com/sudoku.html
Le programme en Python (qui n'est plus disponible sur le site) est affiché sur Studium.
Dans cette implantation, il a utilisé 3 idées :

1. Recherche sous contrainte : En examinant les chiffres dans le même carré et dans les mêmes lignes et colonnes, on peut déterminer les chiffres qu'on peut mettre à

une position (chiffres candidats). Ces contraintes aident beaucoup à réduire les chiffres qu'on peut tenter à chaque position.

2. Recherche en profondeur avec retour en arrière (backtracking) : C'est une recherche en profondeur d'abord, qui permet de revenir en arrière sur un placement tentative fait avant, si jamais le jeu se bloque. L'algorithme correspond bien à l'algorithme de recherche en profondeur d'abord présenté dans le cours.
3. Ordonner les positions : Il ordonne les positions à essayer de telle manière qu'une position avec moins de chiffres candidats sera classée avant. On va donc tenter de placer un chiffre candidat d'abord à une telle position. La raison est expliquée dans la page web de Norvig.

Cette implantation fournit un cadre de base pour faire vos implantations. Elle fournit aussi une façon de modéliser le jeu en espace d'états. Le travail que vous devez faire est d'implanter quelques autres algorithmes pour le jeu, et de comparer leurs performances.

4. Travail à réaliser :

Voici les travaux que vous devez réaliser.

1. (15%) (Sur papier, dans votre rapport) Un des buts de ce TP est de renforcer l'aspect de modélisation en utilisant l'espace d'états. Ainsi, même si le problème a été formulé dans l'implantation de Norvig, on vous demande de mettre cette formulation sur papier explicitement. Notamment, vous devez décrire dans votre rapport les éléments suivants :
   - Définir la notion d'état pour ce problème et proposer une représentation (structure de nœud – voir dans le programme de Norvig) ;
   - Définir l'état de départ et l'état but (ou une fonction de vérification de but) ;
   - Définir la relation de successeur ;
   - Définir le coût d'étape (si nécessaire).

   Cette formulation doit figurer dans votre rapport.

2. (15%) Modifier l'implantation de Norvig en ajoutant un comptage de tentatives. Chaque fois un chiffre est placé dans une position (une tentative), on incrémente ce compte. Ce compte nous servira à analyser la complexité de différents algorithmes.

3. (20%) Implanter l'algorithme Hill-Climbing pour ce problème. Pour cela, à partir de la configuration de départ, on remplit chaque carré avec un de chiffres possibles au hasard, mais en vérifiant les contraintes pour le même carré. Ceci va probablement engendre des conflits sur les lignes ou des colonnes. Ensuite, on utilise Hill-Climbing pour tenter d'améliorer la configuration le plus possible, en inter-changeant (swap) deux des chiffres remplis dans un carré. L'amélioration consiste à réduire le nombre de conflit global (sur les lignes et les colonnes). L'algorithme est brièvement décrit dans la page Wikipedia en anglais.

4. (20%) Utiliser le recuit simulé (simulated annealing). Hill-Climbing peut être coincé sur un optimum local sans pouvoir arranger tous les chiffres correctement (i.e. d'arriver à une solution). On peut améliorer cet algorithme en utilisant le recuit simulé. Cet algorithme est décrit dans le cours, et aussi dans un article par Lewis (voir sur Studium). Vous êtes demandés à implanter une version simplifiée de l'algorithme décrit par Lewis. Notamment, vous utilisez la même stratégie pour diminuer la température : $t_{i+1} = \alpha \cdot t_i$ avec α=0.99. Pour la température initiale, Lewis la détermine selon la dérivation standard des gains produits par un petit nombre d'essais. Pour ce

devoir, vous pouvez utiliser une stratégie simplifiée en la fixant à une valeur. La valeur de 3 peut être testée d'abord. Vous pouvez faire varier cette température de départ pour tester le comportement de l'algorithme. L'article parle aussi de recuit simulé homogène (homogeneous SA). Vous n'en tenez pas compte dans ce devoir.

5. (20%) Utiliser d'autres heuristiques. Ici, on ne vous impose pas quelles heuristiques à utiliser, mais vous devez en implanter au moins une. Vous devez chercher dans les articles sur le Web (ainsi que les articles sur Studium) pour trouver une ou quelques heuristiques à implanter. Ces heuristiques devraient permettre d'améliorer la performance des algorithmes implantés décrits en haut. Un des articles que vous pouvez lire est celui d'Angus Johnson dans http ://www.angusj.com/sudoku/hints.php, qui décrit un ensemble d'heuristiques possibles. Même si le nombre d'heuristiques à implanter n'est pas exigé, on vous encourage à essayer de différentes heuristiques.

6. (10%) Comparer les algorithmes en les faisant fonctionner sur 100 configurations de départ (voir la liste des configurations sur Studium). On compare le nombre de nœuds explorés avant de trouver la solution (ceci nécessite une modification du programme de Norvig pour compter le nombre de nœuds explorés). On compare aussi le pourcentage de jeux réussis (pour lequels l'algorithme trouve la solution). Faites votre analyse personnelle selon ce que vous observez dans ces tests (la complexité de l'algorithme, le taux de succès, …).

5. À rendre

Ce TP est à faire en groupe de **2 personnes**. Vous avez deux choses à rendre : Un rapport et des programmes.

1. Un rapport contenant votre formulation du problème, une brève description de vos implantations et une analyse de fonctionnements sur les 100 exemples. Indiquez dans votre rapport également le programme correspondant à chaque algorithme que vous avez implanté. Indiquez dans votre rapport comment vos programmes doivent fonctionner. Les programmes que vous rendez doivent s'exécuter correctement.
2. Les programmes que vous avez implantés.

Gare au plagiat :

- Il existe beaucoup de programmes de Sudoku sur le Web. Vous ne pouvez pas prendre une implantation sur le Web comme votre devoir. Si vous rendez un tel programme pris directement sur le Web, vous aurez la note de 0.
- Si 2 groupes rendent les mêmes programmes (ou les mêmes programmes masqués), les deux groupes auront 0.

Date de remise :

La date limite pour la remise est le 19 mars avant 23:59. Vous devez remettre le rapport et les programmes sur Studium.

Évaluation :

Ce TP compte pour 15% dans la note globale. Chaque jour de retard entraîne 2 points de pénalité (sur 15 points).

6. Ressources Sur le Web

- 1000 configurations de départ sur le site Sodoku Garden (d'où les 100 configurations de test sont prises). Chaque configuration occupe une ligne, composée de 81 chiffres, correspondant aux 81 cases, avec les 9 lignes concatenées. 0 signifie que la case et vide (à remplir par votre programme).
- Prof. Gordon Royle fournit un grand nombre de configurations de départ ici : http://staffhome.ecm.uwa.edu.au/~00013890/sudokumin.php.
- Vous pouvez pratiquer en ligne ici : http://www.websudoku.com