Laboratoire 3 (partie 1/2) - Recherche locale

Exercice 1 : Résolution d'un sudoku

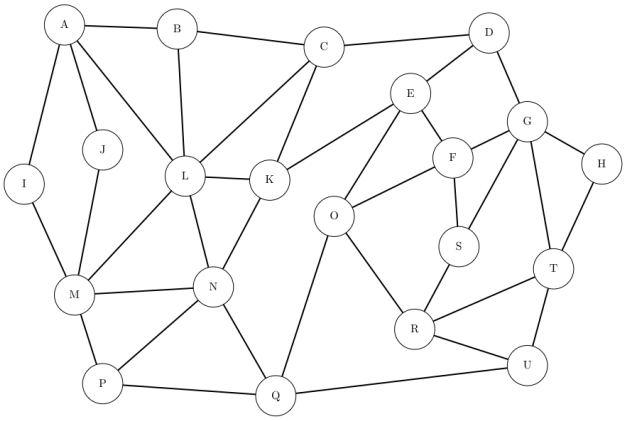
5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Le jeu du Soduku consiste à remplir un carré de 9×9 cases (ou plus pour des cas plus compliqués), de sorte que chaque ligne, colonne, et sous-carré 3×3, ne comportent qu'une et une seule fois chaque chiffre de 1 à 9. Il vous est demandé de proposer une méthode basée sur la recherche locale pour résoudre une situation arbitraire de Sudoku. Concrètement, vous devez réaliser les étapes suivantes :

1. Présentez ce problème comme un CSP (*constraints satisfaction problem*). Identifiez quelles sont les variables de décisions, domaines, et contraintes du problème. Il n'est pas demandé de fournir une description mathématique du problème.
2. Définissez l'ensemble des solutions possibles en vue d'une résolution basée sur la recherche locale. Indiquez quelles sont les contraintes dures ou molles de votre représentation ainsi que la taille de l'espace des solutions.
3. Proposez une méthode pour obtenir une solution initiale.
4. Proposez une fonction d'évaluation pour évaluer la qualité d'une solution.
5. Proposez une fonction de voisinage, de validité, et de sélection que vous pouvez utiliser afin d'améliorer votre solution initiale. Identifier quelle est la taille de votre voisinage.
6. Vous décidez d'implémenter l'algorithme du *hill climbing* pour résoudre ce problème. Est-ce que vous avez la garantie que votre solution soit faisable ? Dans le cas contraire, indiquez pourquoi et proposez une façon concrète de mitiger ce problème.

Exercice 2 : Le temps des fêtes

Pour le temps des fêtes, vous souhaitez organiser une soirée entre amis chez vous. Vos amis ont des relations assez difficiles entre eux, et certains sont en conflit. Chaque conflit est réciproque, c'est-à-dire que si Alice est en conflit avec Bernard, Bernard sera également en conflit avec Alice. L'ensemble des conflits est représenté comme un graphe, où chaque nœud correspond à un ami, et chaque arête indique la présence d'un conflit. Pour vos 21 amis, le graphe des conflits est présenté ci-dessous.



Afin que tout le monde profite au mieux de la soirée, vous décidez de n'inviter que des amis qui ne sont pas en conflit. Malheureusement, suite aux nouvelles mesures sanitaires, vous êtes également limité à n'inviter que 10 amis. Votre problème est le suivant : *"Est-il possible d'inviter exactement 10 amis de sorte à ce qu'il n'y ait aucun conflit ?"*. Par exemple, il n'est pas possible d'inviter à la fois Alice (A) et Bernard (B), par contre il est possible d'inviter Alice (A) et Claude (C).

Pour cette question, il ne vous est pas demandé de résoudre complètement ce problème, mais simplement de proposer une résolution basée sur la recherche locale et d'exécuter quelques étapes de l'algorithme.

1. Proposez un modèle de recherche locale en vue de résoudre ce problème. En particulier, veillez à bien définir les éléments suivants :
   (a) Votre espace de recherche. Indiquez le plus précisément possible quelle est sa taille.
   (b) Les contraintes dures et molles relatives à l'espace de recherche que vous avez considéré.
   (c) Une méthode pour obtenir une solution initiale.
   (d) Votre fonction d'évaluation pour caractériser la qualité d'un état.
   (e) Votre fonction de voisinage.
   (f) Votre critère pour sélectionner un voisin dans le voisinage.
2. Exécutez deux itérations d'une recherche locale sur base de votre formulation. Vous ne devez pas indiquer tous les voisins de votre voisinage, prenez en juste 5 au maximum (qui sont cohérents avec la définition de votre voisinage), et faites la sélection parmi ces 5 voisins selon votre critère. En repartant de la figure de départ en 3 exemplaires, illustrez (a) votre solution initiale, (b) la solution obtenue après la première itération, (c) la solution obtenue après la deuxième itération. Pour ces trois solutions, indiquez la valeur de votre fonction d'évaluation.
3. Proposez une méthode pour éviter que votre algorithme se retrouve bloqué dans un minimum local.

Exercice 3 : Le chantier spatio-naval de Naboo

La compagnie JarJar Bink's & Son LLC est un armateur bien connu tant de l'empire galactique que de l'alliance rebelle. Cette entreprise est le leader inter-galactique de la construction de vaisseaux légers d'interception et de transport. Ils sont capables de produire un certain nombre de *vaisseaux* dont le Chasseur TIE, le A-wing, le X-wing, le Y-wing, ou encore le redoutable TIE Avenger.

Chacun de ces chasseurs possède une liste fixe d'*options* dont certaines peuvent être commune à plusieurs types d'appareil. Parmis ces options, on retrouve le propulseur ionique XC-4N, le canon laser EM-1919, ou encore le drone de diagnostic R2-D2.

Chaque option o_i est associée à une équipe d'ouvriers essentiellement composée de Gungans. Les Gungans étant de nature indolente, chaque équipe à le droit à un temps de repos formalisé par le ratio suivant p_i/q_i : sur q_i vaisseaux les uns à la suite des autres dans la chaine de production, au maximum p_i vaisseaux doivent nécessiter l'option o_i.

Par ailleurs, pour chaque classe, la compagnie reçoit une commande de d_i vaisseaux devant être construits. L'objectif est de planifier l'ordre de construction de tous les vaisseaux de sorte que le critère de productivité p_i/q_i ne soit jamais dépassé pour chacune des options o_i dans la chaine d'assemblage. Chaque construction de vaisseaux prend une durée d'un jour, et l'entièreté des constructions doit être réalisée après T jours au maximum. A chaque jour, au maximum un vaisseau peut-être construit.

Une visualisation de ce problème pour une situation comportant 6 vaisseaux, 5 options, un horizon de 10 jours, et des valeurs spécifiques de capacité/demandes est disponible ci-dessous :

Options	1	2	3	4	5	Demande
Vaisseau 1	■		■	■		1
Vaisseau 2				■		1
Vaisseau 3		■			■	2
Vaisseau 4		■		■		2
Vaisseau 5	■		■			2
Vaisseau 6	■	■				2
Capacité	1/2	2/3	1/3	2/5	1/5	

N'étant pas familier avec les outils de pointe en intelligence artificielle, JarJar Bink's a initialement agencé la construction de façon séquentielle selon le numéro des vaisseaux. Chaque type de vaisseau est construit successivement jusqu'à ce que la demande soit satisfaite. Cela donne la solution initiale suivante. Bien que cette méthode permette de satisfaire les contraintes de demandes (figure de gauche), les contraintes de capacité de sont pas respectées (figure de droite). Les conflicts sont indiqués en rouge.

Jours	1	2	3	4	5	6	7	8	9	10	Demande
Vaisseau 1	■										1
Vaisseau 2		■									1
Vaisseau 3			■	■							2
Vaisseau 4					■	■					2
Vaisseau 5							■	■			2
Vaisseau 6									■	■	2

Vue de la construction en fonction des vaisseaux

Jours	1	2	3	4	5	6	7	8	9	10	Capacité
Option 1	■						■	■	■	■	1/2
Option 2			■	■	■	■			■	■	2/3
Option 3	■						■	■			1/3
Option 4	■	■			■	■					2/5
Option 5			■	■							1/5

Vue de la construction en fonction des options

Il vous est demandé d'utiliser la recherche locale pour améliorer cette solution initiale. Pour les questions suivantes, une explication en français appuyée par une représentation visuelle sur base des schémas précédent suffit. Aucun modèle mathématique n'est demandé.

1. Proposez une méthode pour repérer les conflits concernant les contraintes de capacité d'options.
2. Proposez une fonction d'évaluation pour évaluer la qualité d'une solution.
3. Proposez une fonction de voisinage, de validité, et de sélection afin d'améliorer cette solution de sorte que la contrainte de demande reste dure (i.e., votre voisinage ne doit pas contenir de solutions permettant de violer cette contrainte). Une explication en français appuyée par une représentation visuelle suffit.
4. Effectuez 2 itérations de recherche locale suivant votre voisinage et représentez graphiquement les nouvelles solutions obtenues.

Exercice 4 : Implémentation d'une recherche locale

⚠Cette question est hors-matière pour l'examen.

Cet exercice a pour objectif de vous montrer une implémentation concrète d'un algorithme de recherche locale. Le problème considéré est celui du carré magique vu au cours. Aucun code n'est demandé, sauf pour la dernière partie qui vous propose de réaliser votre propre algorithme. Les autres questions vous demande essentiellement de comprendre l'implémentation et d'expérimenter avec. Vous avez à votre disposition une implémentation python dans le sous-dossier `code-lab3`. Le dossier contient plusieurs fichiers :

— `magic_square.py` : définition du problème du carré magique.

— `local_search.py` : Implémentation d'une recherche locale.

— `local_search_with_restarts.py` : Implémentation d'une recherche locale avec restarts.

— `simulated_annealing.py` : Implémentation d'un algorithme de *simulated annealing*.

Cet exercice est divisé en 9 scénarios que vous devez exécuter via la commande suivante : *python3 main.py –scenario Si*, en remplaçant *Si* par l'un des scénarios (**S1**, ..., **S9**). Ces scénarios sont détaillés dans le code. Pour chaque scénario, faites attention à bien identifier la fonction d'évaluation, le voisinage, le résultat obtenu, ainsi que les différentes variantes de l'algorithme. Le scénario **S9**, vous demande de résoudre un carré magique de taille 6×6