Laboratoire 3 - Modélisation mathématique et recherche locale

Exercice 1 - Explication de concepts

Expliquez brièvement les concepts suivants.

1. la différence entre un problème combinatoire de satisfaction et d'optimisation.
2. la différence entre une recherche constructive et perturbative.
3. la différence entre une recherche complète et incomplète.
4. La différence entre une contrainte dure et une contrainte molle.
5. La différence entre une fonction objectif et une fonction d'évaluation.

Exercice 2 - Planification de l'impression de livres

L'entreprise *Cachette* a récemment fait l'acquisition d'une machine de dernière génération pour la production de livres. Sur base d'un fichier digital remis, la machine procède automatiquement à l'impression du livre en format papier. La machine n'est capable de n'imprimer qu'un seul livre à la fois. Considérons L l'ensemble des devant être imprimés. Pour chaque livre $l \in L$, on sait que le fichier digital n'est disponible qu'à partir du jour $r_l \in \mathbb{N}$. Sans perte de généralité, on peut considérer que le jour actuel est le jour 0. Ainsi, une impression d'un livre de peut pas commencer avant cette date. De plus, l'impression de chaque a sa propre durée, nommée $d_l \in \mathbb{N}$. Finalement, chaque livre a une date limite $f_l \in \mathbb{N}$, à laquelle il est souhaite que le livre soit complètement imprimé. Si un livre est imprimé après sa date limite, un *retard* est engendré, ce qui cause un coût pour l'entreprise. L'objectif de fixer un horaire d'impression afin de minimiser la somme des jours de retard engendré. A cette fin, elle peut décider du jour auquel chaque livre va commencer son impression.

Pour cette question, il vous est demandé de formuler mathématiquement ce problème, et de proposer un modèle de recherche locale pour sa résolution. Pour vous aider, vous pouvez considérer la situation suivante comportant 5 livres.

Livre (l)	Jour de disponibilité (r_l)	Durée d'impression (d_l)	Date de remise (f_l)
1	0	4	7
2	1	3	6
3	4	5	11
4	8	6	13
5	8	2	16

TABLE 1 – Situation d'exemple pour le problème d'impression de livres

1. Modélisez mathématiquement ce problème comme un COP. En particulier, veillez à bien définir vos variables de décision, vos domaines, vos contraintes, et votre fonction objectif.
2. Proposez un un espace de recherche afin qu'il soit exploré par une recherche locale. Indiquez le plus précisément possible quelle est sa taille.
3. Indiquez quelles sont les contraintes dures et molles relatives à l'espace de recherche considéré.
4. Proposez une méthode pour obtenir une solution initiale. Représentez visuellement votre solution.

5. Proposez une fonction d'évaluation pour caractériser la qualité d'une solution.
6. Proposez une fonction de voisinage. Indiquez le plus précisément possible quelle est sa taille.
7. Proposez une fonction de sélection. Indiquez qu'elle est la complexité d'une sélection.
8. Effectuez deux itérations de recherche locale avec votre modèle.

Exercice 3 : Carré magique - implémentation d'une recherche locale

Dans cet exercice, on s'intéresse au problème du carré magique vu au cours. Soit n un entier, le problème du carré magique consiste à placer les nombres de 1 à n^2 dans une grille carrée de taille $n \times n$ de sorte que la somme des nombres présents sur chaque ligne, chaque colonne, et les deux diagonales soit identique.

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. Tous les codes vous sont fournis, à l'exception de celui associé au dernier scénario, dans lequel vous devrez essayer de résoudre le carré magique de taille 6×6 avec la méthode de votre choix. Pour les autres scénarii, il vous est demandé de comprendre l'implémentation et d'expérimenter avec. Plusieurs fichiers vous sont fournis :

— `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 des redémarrages aléatoires.
— `simulated_annealing.py` : Implémentation d'un algorithme de *simulated annealing*. Cette technique sera vue prochainement, vous pourrez sur ces scénarios plus tard.

Cet exercice est divisé en 9 scénarios que vous devez exécuter via la commande suivante en remplaçant *Si* par l'un des scénarios (**S1**, . . ., **S9**). Ces scénarios sont détaillés dans le code.

```
python3 main.py --scenario Si
```

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.