Laboratoire 5 - Métaheuristiques de construction

Exercice 1 - Explication de concepts

Expliquez brièvement les concepts suivants.

1. Les difficultés qui peuvent survenir lorsqu'on souhaite obtenir une solution initiale pour une recherche locale. Quels mécanismes peuvent être utilisés pour mitiger ces difficultés ?
2. Les différences fondamentales entre une recherche en faisceau de largeur k et une stratégie basée sur k redémarrages.
3. L'impact du paramètre α dans la conception de la liste restreinte de candidat pour la métaheuristique GRASP.
4. L'intérêt des phéromones dans la métaheuristique ACO. Comment les phéromones se mettent à jour au fil des itérations ?
5. La force idéale d'une perturbation dans une recherche locale itérée.

Exercice 2 : Problème de placement de machines - Implémentation

Dans cet exercice, on s'intéresse au même problème de placement de machines qu'au laboratoire précédent. A toutes fins utiles, son énoncé est tout de même rappelé. Soit I un ensemble de jobs, soit J un ensemble de sites dans un atelier, et soit M un ensemble de machines. Chaque job doit passer sur un certain nombre connu de machines, dans un ordre également connu. A chaque job est donc associé un sous-ensemble ordonné de M. Notons également que chaque job démarre à la première machine qui doit la traiter, passe sur les machines qui lui sont assignées, puis termine son parcours à la dernière machine qu'elle visite. L'objectif est de placer les machines de M sur les sites de J de façon à minimiser la distance totale parcourue par les jobs. Les distances entre chaque paire de sites sont toutes connues. Une instance du problème est représentée par un fichier de $4 + M + J$ lignes ayant la forme suivante.

```
J
M

d(0,0) d(0,1) ... d(0,M)
d(1,0) d(1,1) ... d(1,M)
...
d(M,0) d(M,1) ... d(M,M)

o[0]
o[1]
...
o[J]
```

De façon similaire, une solution est représentée par un fichier 2 lignes, où $s[m]$ est le site associé à la machine m. Un jeu de 4 instances, similaires à la semaine passée, vous est donné.

```
C
s[0], s[1], ..., s[M]
```

Cet exercice a pour objectif de vous montrer une implémentation concrète des métaheuristiques de construction vues au cours : recherche locale en faisceau [1], GRASP, optimisation par colonies de fourmis, et recherche locale itérée. Tous les codes vous sont fournis, à l'exception de celui associé au dernier scénario, où vous serez invités à expérimenter plusieurs choix de conception vus au cours. Pour les autres scénarios, il vous est demandé de comprendre l'implémentation et d'expérimenter avec. Notez que ces implémentations comportent plusieurs sources d'inefficacités, que ce soit au niveau des choix de conception, de l'équilibre diversification/intensification, et de l'implémentation. Un de vos objectifs est d'apporter un regard critique sur ces implémentations et d'en déceler les possibilités d'amélioration sur base des techniques vues au cours. Plusieurs fichiers vous sont fournis :

— `atelier.py` : définition du problème de placement de machines.
— `local_search.py` : implémentation d'une recherche locale pour le problème.
— `beam_search.py` : implémentation d'une recherche locale avec construction d'une solution initiale par recherche en faisceau
— `local_beam_search` : Implémentation d'une recherche locale en faisceau.
— `grasp.py` : implémentation d'un algorithme GRASP.
— `ils.py` : implémentation d'un algorithme de recherche locale itérée.
— `aco.py` : implémentation d'un algorithme de recherche par colonie de fourmis.
— `solver_advanced.py` : pour l'implémentation de votre métaheuristique de construction personnalisée.

Vous avez également à votre disposition 4 fichiers d'instances nommés selon le schéma `atelier_J_M.txt`. Deux petites instances pour comprendre le code, un instance moyenne pour voir plus en détail l'influence des paramètres et enfin une grande instance pour les plus courageux.

Cet exercice est divisé en 6 scénarios que vous devez exécuter via la commande suivante en remplaçant *Si* par l'un des scénarios (**S1**, ..., **S6**). Ces scénarios sont détaillés dans le code. Chaque méthode a finalement plusieurs paramètres que vous pouvez calibrer.

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

Pour chaque scénario, faites attention à bien identifier les différentes étapes clés propres à chaque algorithme. Faites varier les paramètres et amusez vous!

1. A strictement parler, ce n'est pas une métaheuristique de construction mais une amélioration du *hill climbing*.