# TP4 - Séquence de pygment

**Remise le 18/04/21 sur Moodle.**

## Consignes

- Le TP peuvent être faits par groupe de 2 au maximum.
- Toutes les consignes générales du cours (interdiction de plagiat, etc.) s'appliquent pour ce TP.
- Votre rapport doit faire au maximum 2 pages (titre, page de garde, et figures non compris).

## 1 Énoncé du problème

Vous êtes responsable de planifier la production chez PastaPasta. L'entreprise possède une ligne de production modulaire pour faire des pâtes. Elle accepte plusieurs types de farine (farine de blé, farine de riz, etc.), de moule (spaghetti, tortellini, etc.), d'additif (poudre de tomate, poudre d'épinard, etc.) et de paquets (250 grammes, 500 grammes, etc..). Utiliser un type spécifique requiert que le processus de production soit dans une configuration précise. Chaque jour, un lot de pâtes est produit. Pour passer d'une configuration à une autre, il y a un certain coût. Ainsi, passer d'un paquet $x$ à un un paquet $y$ n'engendre qu'un coût faible tandis que passer d'une farine à un paquet demande a un coût plus élevé (nettoyage complet pour éviter la présence du gluten résiduel). L'entreprise reçoit un certain nombre de commandes. Celle-ci ont chacune une date de livraison. Une commande peut être produite à l'avance. Il y a cependant des coûts de stockage par jour d'attente. Votre travail consiste à organiser le planning de production tel que le coût total (somme des coûts de changement des paramètres de la machine et des coûts de stockage) soit le plus bas possible.

### 1.1 Format

De manière plus complète, le problème est le suivant. Vous avez un certain nombre de jours maximum pour réaliser le planning, ainsi qu'un certain nombre de commandes à satisfaire. Chaque commande doit être remplie avant sa date de livraison. Le coût de stockage par jour est fixe et augmente quotidiennement (stocker 2 jours coutera donc 2 fois le coût par jour de stockage). Le coût de transition est également donné. Le coût de transition entre deux jours avec la même commande est nul.

Les fichiers contenant les instances du problème sont organisés en $3+2*C$ lignes ayant la signification suivante :

1. La première ligne contient un entier : le nombre $J$ de jours pour faire le planning.
2. La seconde ligne contient un entier : le nombre $C$ de produits possibles (nombre de configurations possible de la ligne de production).
3. Les $C$ lignes suivantes contiennent $J$ entiesr et décrivent pour chaque jour combien de commandes de ce type sont demandée.
4. La ligne suivante contient un entier : le coût de stockage $h$.
5. Pour finir, les $C$ dernières lignes contiennent $C$ entier, décrivant le coût de transition pour un type à un autre.

```
J
C
o_0^1 o_0^2 o_0^3 o_0^4 ... o_0^{J}
o_1^1 o_1^2 o_1^3 o_1^4 ... o_1^{J}
...
o_{C-1}^1 o_{C-1}^2 o_{C-1}^3 ... o_{C-1}^{J}
h
q^{1,0} q^{1,1} q^{1,2} q^{1,3} ... q^{1,C}
q^{2,0} q^{2,1} q^{2,2} q^{2,3} ... q^{2,C}
...
q^{C,0} q^{C,1} q^{C,2} q^{C,3} ... q^{C,C}
```

Le format d'une solution attendue comporte 2 lignes. La première ligne contient un entier qui donne le coût de la solution. L'autre ligne contient le planning des $J$ jours, avec chaque jour quel type est produit. Si un jour n'a pas de production, la valeur $-1$ est indiquée.

```
totValue
t_1 t_2 ... t_{J}
```

### 1.2 Exemple

L'instance suivante 2 produits différents et une production sur 5 jours.

```
5
2
0 1 0 0 1
1 0 0 0 1
2
0 5
3 0
```

Une possible solution au problème est la solution suivante.

```
15
1 0 1 -1 0
```

## 2 Implémentation

Vous avez a votre disposition un projet python ainsi qu'un environnement anaconda.

### 2.1 Instances de test

Vous avez à votre disposition 4 instances (2 supplémentaires sont cachées et serviront pour l'évaluation) sur lesquelles vous pouvez vous entrainer.

### 2.2 Description du projet à votre disposition

Trois fichier pythons sont mis à votre disposition. Le fichier `pygment.py` contient une classe modélisant une instance qui permet d'écrire la solution, de dessiner la solution, de calculer le coût, de vérifier une solution, etc. Le fichier `solver.py` contient une première version gloutonne de l'algorithme. Elle contient également la signature de la fonction de résolution plus avancée que vous devez implémenter. Vous pouvez utiliser la résolution naïve comme point de départ. Si vous avez besoin de fonction/classe annexe, veuillez les mettre dans le fichier `solver.py` ou modifier la classe dans `pygment.py`. Le fichier `main.py` permet de lancer le programme.

## 3 Ce qui vous est demandé

Votre mission consiste à compléter la fonction `solve_advance(pygment)` afin de résoudre une instance du problème en utilisant la recherche locale et des méta-heuristiques. Vous pouvez modifier le fichier `pygment.py` si vous voulez pour y ajouter des fonctions d'aide. Le temps d'exécution est limité à 10 minutes. Inspirez vous des techniques déjà vue en cours. Notez qu'il existe une multitudes de façons pour résoudre ce problème.

Nous vous demandons d'utiliser au moins l'une des métaheuristiques suivantes : *tabu search*, algorithmes génétiques, *GRASP*, *iterated local search* ou *ant colony optimization*, *VNS*, *LNS*, *ALNS*. Vous êtes libre de choisir celle de votre choix et d'y importer n'importe quelle amélioration (solution initiale heuristique, restarts, beamsearch, etc.).

**ATTENTION :** vous devez utiliser une autre métaheuristique que celle utilisée lors du TP 3, ou du moins y intégrer un mécanisme supplémentaire. Par exemple, si vous avez utilisé ILS lors du TP3, il est permis de faire une méthode combinant GRASP + ILS. Si vous avez déjà utilisé GRASP + ILS, vous pouvez faire une autre hybridation, comme ACO + ILS, ou GRASP + LNS.

## 4 Evaluation

Vous serez évalués en fonction de plusieurs critères : la clarté de votre implémentation (commentaire, etc.), la capacité de votre code à résoudre les instances données (plus vous êtes proche d'une solution faisable, plus vous aurez de points) et votre rapport (explication claire de vos algorithmes et de vos choix conceptuels). La répartition précise des points est la suivante :

- 6 points sur 10 seront attribués pour la qualité de votre code. Donne t-il une solution faisable meilleure qu'une solution naïve ? Donne t-il une solution d'une bonne qualité ? Chaque instance rapporte 1 points. **Attention : un code qui ne compile pas ou qui retourne une solution incohérente sur une instance ne vous donnera aucun point pour l'instance en question.**
- 3 points sur 10 seront attribués pour la qualité de votre rapport. Est-ce que votre modèle et vos choix de conception sont clairement expliqués ?
- 1 point sur 10 sera attribué à la clarté de votre implémentation.
- 2 points bonus seront attribués au groupe ayant le meilleur résultat sur les deux instances cachées. En cas d'égalité pour une instance, les points seront répartis.

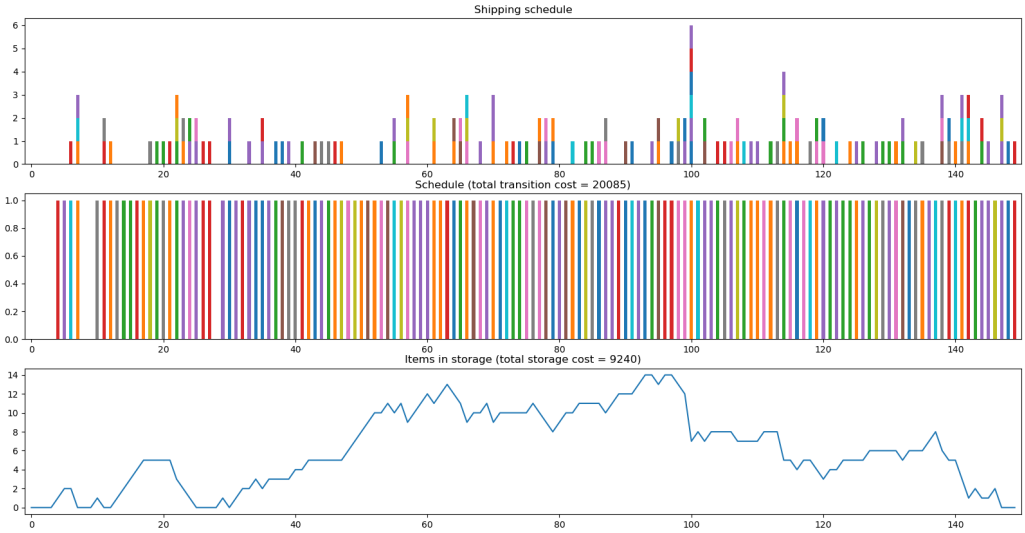
# 5 Remise

Vous devez remettre une archive zip contenant :

- vos fichiers `solver.py` et `pygment.py`, contenant votre algorithme ainsi que vos fonctions et classes annexes
- les solutions obtenues sur chacune des instances à votre disposition (les fichiers solution doivent être nommés "solutionX" avec X la lettre correspondant à l'instance)
- un rapport de 2 pages (titre et images non compris) expliquant votre solution

Vous pouvez, pour vous motiver entre groupe, partager sur slack le cout (pas les solutions exactes) de la meilleure solution que vous avez.

# 6 Visualisation




La visualisation mise à votre disposition contient 3 graphiques.

- le premier graphique montre le planning de livraison (chaque type de batch est représenté avec une couleur différente). La hauteur de chaque barre donne le nombre de batch à livrer pour ce jour, les couleurs donne quels types.
- le dexième graphique montre votre planning de production. Chaque barre donne la couleur de l'objet produit ce jour là. Dans le titre se trouve également le cout de toutes les transitions de votre solution. Hint : plus les couleurs sont éparpillées, plus il y aura de coût de transition.
- le troisième graphique donne l'évolution des stocks. La courbe donne l'évolution de la fonction $s[i] = s[i-1] + p[i] - l[i]$ où $s[i]$ est le stock au jour $i$, $p[i]$ est la production au jour $i$ (0 ou 1) et $l[i]$ est la livraison au jour $i$ (taille de la barre dans le premier graphique au même jour).