TP Optimisation – session 3 – Tournoi

28 novembre 2024

Ce sujet démarre le tournoi d'optimisation avec un problème d'optimisation combinatoire classique qui est NP-difficile.

1 Le problème du sac à dos (ou *knapsack*)

On dispose d'objets que l'on souhaite sélectionner ou non sachant qu'on ne peut pas tous les prendre à cause de contraintes de *poids* (ou de capacité). D'autre part, chaque objet a une *valeur* (ou profit) et on souhaite maximiser la sommes des valeurs des objets sélectionnés.

Le sac à dos est défini par un poids maximum W. Pour chacun des n objets, on aura une valeur p_i et un poids w_i avec $1 \leq i \leq n$. Le problème consiste à trouver un ensemble X contenant les indices des objets sélectionnés qui maximise

$$\sum_{i \in X} p_i$$

tout en respectant les contraintes de poids

$$\sum_{i \in X} w_i \leq W.$$

Il s'agit du *sac à dos de base* qui sera le cas principal du tournoi (les cas avec plusieurs dimensions et plusieurs sac à dos seront traités ultérieurement). Un exemple d'instance pour ce problème : on a une contrainte de poids $W = 10$ pour le sac à dos et $n = 5$ objets de valeurs $\{1, 100, 200, 500, 900\}$ et de poids $\{8, 3, 4, 6, 7\}$. Dans ce cas, la solution optimale consiste à sélectionner les objets d'indices 2 et 5 pour une valeur totale de 1000.

Ce problème peut se rencontrer dans plusieurs situations pratiques. Par exemple, si l'on souhaite préparer une valise pour un voyage, on peut associer un poids qui représente le volume pour chaque objet et une valeur qui représente l'utilité de chaque objet que l'on souhaite emmener.

2 Objectifs du tournoi

Il faudra concevoir les *stratégies algorithmiques* les plus simples et rechercher puis implémenter les stratégies les plus avancées. Pour chaque stratégie mise en œuvre, il faudra proposer une *instance pathologique*, c'est-à-dire une instance pour laquelle le rapport de la valeur objectif de la solution générée avec l'optimal est le plus grand possible (pour les stratégies non-optimales seulement).

3 Format et structure du code

Les instances suivront le format textuel suivant :

```
max_time #nb_knapsack #nb_dim #nb_item
capacity11 capacity12 ...
capacity21 capacity22 ...
...
profit1 weight11 weight12 ...
profit2 weight21 weight22 ...
...
```

Voici la description des différents champs :

max_time nombre de secondes maximum dédiées au traitement de cette instance

#nb_knapsack nombre de sac à dos (fixé à 1 pour l'instant, le cas où ce sera supérieur sera traité ultérieurement)

#nb_dim nombre de dimensions (fixé à 1 pour l'instant, le cas où ce sera supérieur sera traité ultérieurement)

#nb_item nombre d'objets

Pour l'exemple précédent, on a :

```
1 1 1 5
10
1 8
100 3
200 4
500 6
900 7
```

Le code qu'il faudra développer devra fournir une solution dans la limite de temps donnée dans l'instance et une limite d'espace générale de 10 Go maximum. Cependant, ces limites n'ont pas besoin d'être prises en compte dans les soumissions étudiantes car elles seront garanties lors de l'évaluation. En effet, lors de l'évaluation, le code sera exécutée de la façon suivante :

```
ulimit -Sv 10000000
SECONDS=$(head -n 1 input.txt | cut -f1 -d" ")
timeout ${SECONDS} python script.py input.txt output.txt
```

Le fichier résultat doit contenir la liste des indices des objets sélectionnés. C'est `1 4` pour la solution optimale dans l'exemple précédent.

Les scripts implémentant les stratégies algorithmiques proposées devront respecter les contraintes suivantes :

- Chaque script ne doit accepter que deux arguments : le nom du fichier contenant l'instance, puis le nom du fichier dans lequel le résultat sera écrit.
- Les sorties standard et d'erreur sont ignorées et peuvent être utilisées à des fins de débogage.
- Les seules dépendances autorisées sont celles indiquées dans le code fourni en exemple (chaque autre dépendance devra être validée par l'enseignant).
- Un ensemble de 20 scripts maximum peut être soumis (un script par stratégie algorithmique). Lors de l'évaluation, tous les scripts fournis seront utilisés et c'est la meilleure solution qui sera considérée pour le classement.
- Chaque script devra expliquer et décrire en commentaire la stratégie implémentée et l'instance pathologique associée.
- Le nommage des scripts et de leurs instances pathologiques devra suivre le schéma suivant : `nom_algo.py` pour le script et `nom_algo.txt` pour l'instance.
- Pour chaque stratégie algorithmique où il est possible de fournir une instance pathologique avec un rapport arbitrairement élevé, on s'arrêtera à 1000 et lorsque ce rapport dépasse cette valeur, on le tronquera à 1000 lors de l'évaluation (pas de malus si le rapport dépasse, mais pas de bonus non-plus).

Le code fourni en exemple (`basic_greedy`) lit une instance, applique une solution gloutonne basique et génère un fichier résultat. Pour cette solution gloutonne basique, l'instance donnée plus haut est un cas pathologique car le glouton retourne 1 alors que l'optimal est à 1000.

4 Évaluation

L'évaluation portera sur 5 catégories de stratégies algorithmiques :

glouton les heuristiques non-garanties (hormis celles qui s'appuient sur un programme linéaire) ;

algorithme d'approximation les algorithmes d'approximation (hormis ceux qui s'appuient sur un programme linéaire) ;

programmation linéaire les solutions basées sur un programme linéaire ;

multidimensionnel les solutions pour le cas à plusieurs dimensions ;

multiple les solutions pour le cas multiple.

Chaque catégorie sera notée sur 4 points et prendra en compte les stratégies implémentées et les instances pathologiques fournies (pour les stratégies non-optimales). Les 3 premières catégories concernent le sac à dos de base (une seule dimension avec un seul sac à dos). Elles sont donc le thème de la première séance de TP dédiée au tournoi et du premier rendu intermédiaire. La quatrième catégorie est le thème de la seconde séance de TP dédiée au tournoi et du second rendu intermédiaire. La cinquième catégorie est le thème de la troisième séance de TP dédiée au tournoi.

Les rendus intermédiaires permettent d'avoir un classement intermédiaire. Pour la restitution finale, des instances pathologiques, des instances de grandes tailles et les instances pathologiques des autres étudiants seront utilisées.

Points bonus (uniquement pour le rendu final) :

- +1 pour les 3 étudiants classés en premiers : le score est le rang moyen sur chaque instance.
- +1 pour chaque étudiant qui bat les stratégies algorithmiques enseignantes pour au moins une instance.
- +1 pour chaque étudiant qui soumettra au moins une instance plus pathologique que les instances enseignantes.

Points malus (uniquement pour le rendu final) :

- -1 pour chaque modification manuelle dans le code nécessaire pour le faire fonctionner.
- -1 pour une soumission avec un problème de nommage ou d'une archive zip plutôt que les scripts et instances directement.
- -1 si le maximum de 20 scripts est dépassé et les scripts au delà du vingtième sont ignorés.
- -1 pour l'utilisation d'une dépendance non-autorisée et le script impacté n'est pas évalué.
- -1 pour une instance pathologique invalide (valeur nulle ou négative) et l'instance est ignorée.

5 Déroulement

— 3 séances de TP dédiées au tournoi : 19/11/2025, 3/12/2025 et 10/12/2025.
— 2 rendus intermédiaires : 2/12/2024 et 9/12/2024 à 23h59 (script et instances pour sac à dos de base, puis même chose pour la version multidimensionnelle).
— Le rendu final le 8/1/2025 à 23h59.
— La restitution le 9/1/2025 de 13h30 à 15h.

6 Pour commencer

En première étape, il est possible d'adapter le glouton basique fourni dans `greedy_dumb.py` de manière à obtenir une meilleure solution pour l'instance `greedy_dumb.txt`. Pour cela, il faut copier les 2 fichiers en les renommant par le nom d'une meilleure stratégie algorithme que vous concevrez puis implémenterez. Il faudra aussi adapter l'instance pathologique pour obtenir le cas le plus défavorable pour la nouvelle stratégie. Il faudra penser à décrire la nouvelle stratégie et son instance pathologique en commentaire dans le script python. Après avoir recommencé ce processus plusieurs fois, vous aurez une collection de scripts python et de fichiers textuels. Ce sont ces fichiers qu'il faudra soumettre lors des différents rendus.