INFO-F310 - Algorithmique et recherche opérationnelle

Dimitrios Papadimitriou
dimitrios.papadimitriou@ulb.be

Hugo Callebaut
hugo.callebaut@ulb.be

1 Problème

Le problème que l'on vous demande d'analyser est celui du sac à dos multiple (en anglais, multiple knapsack). Le problème du sac à dos jouit de très nombreuses applications concrètes dans divers domaines. En logistique, il permet d'optimiser le chargement des marchandises, en garantissant que la valeur maximale est transportée sans dépasser les limites de poids de chargement. En finance, il est couramment utilisé pour la sélection de portefeuille, où les investisseurs doivent choisir une combinaison d'actifs pour maximiser leurs rendements tout en respectant les contraintes de risque. En outre, il apparaît dans les problèmes d'allocation des ressources dans la gestion de parcs informatiques et de réseaux de télécommunication (contraintes de capacité), la gestion de projet et la budgétisation. Récemment, il est également apparu dans le domaine de l'écologie pour l'optimisation du captage et du stockage du carbone mise en oeuvre par différentes organisations ainsi que l'optimisation des quantités d'énergies renouvelables utilisées par différentes organisations.

Le problème du sac à dos vient avec deux variantes : la plus connue consiste a maximiser le profit (le gain ou le rendement) des items/objets sélectionnés. Dans ce cas, la formulation du modèle de programmation entière du problème du sac à dos unidimensionnel est la suivante

$$(maxKP)\ \max \sum_{j=1}^{n} p_j x_j \tag{1.1}$$

$$s.t. \sum_{j=1}^{n} w_j x_j \leq C \tag{1.2}$$

$$x_j \in \{0,1\} \qquad \forall j \in \{1,2,\ldots,n\} \tag{1.3}$$

Une instance de sac à dos maximale est donnée comme un ensemble N comprenant n articles (indexé par j allant de 1 à n) ayant chacun un profit $p_j > 0$ et un poids $w_j > 0$ associés, ainsi qu'une valeur de capacité $C > 0$ associée au sac à dos. Le problème consiste à sélectionner un sous-ensemble $\hat{N} \subseteq N$ d'articles de poids total d'au plus C (1.2) dont la somme de leur profit respectif est la plus élevée possible, donc la somme de leur profit est maximisée. Pour éviter les solutions triviales, les hypothèses suivantes sont considérées : $w_j \leq C, j \in 1,\ldots,n$ et $\sum_{j=1}^{n} w_j > C$. Dans ce cas, la variable binaire $x_j \in \{0,1\}, \forall j$ (1.3) modélise la présence (et donc la sélection) ou non de l'article j dans le sac à dos. Si $x_j = 1$, alors l'article j est inclus dans le sac à dos, sinon $x_j = 0$ indique que l'article n'est pas inclus dans le sac à dos. Par conséquent, cette variante du problème du sac à dos est appelée sac à dos 0/1.

Dans un problème de sacs à dos multiples, le problème consiste à sélectionner un sous-ensemble d'articles/-d'objets parmi un ensemble N donné, dans plusieurs sacs à dos unidimensionnels. Si nous avons à disposition un ensemble N de n articles et un ensemble M de m sacs à dos unidimensionnels de capacité $C_i, i = 1,\ldots,m$, nous obtenons le problème des sacs à dos unidimensionnels multiples.

La version moins connue est celle de la minimisation du coût des items/objets sélectionnés sachant qu'une taille minimum doit être atteinte par le sous-ensemble des objets sélectionnés. Cette taille minimum peut être interprétée comme une demande de taille minimum devant être fournie. Une instance de sac à dos minimal est donnée comme un ensemble N de n articles (indexé par j allant de 1 à n) ayant chacun un coût $c_j > 0$ (au lieu d'un profit p_j) et une taille $s_j > 0$ associés. Le problème est de sélectionner un sous-ensemble d'items d'une taille totale d'au moins $D > 0$ dont le coût est le plus faible possible, donc la somme de leur coût est

minimisée. Le programme en nombres entiers pour le sac à dos minimal est présenté ci-dessous. La variable binaire $x_j \in \{0, 1\}$ modélise la présence (et donc la sélection) ou non de l'article j dans le sac à dos.

$$(MinKP)\ \min \sum_{j=1}^{n} c_j x_j \tag{1.4}$$

$$s.t. \sum_{j=1}^{n} s_j x_j \geq D \tag{1.5}$$

$$x_j \in \{0, 1\} \qquad \forall j \in \{1, 2, \ldots, n\} \tag{1.6}$$

Ce problème de minimisation est faisable si et seulement si $\sum_{j=1}^{n} s_j \geq D$. Il est a noter que le problème consistant à sélectionner un sous-ensemble d'articles $\hat{N} \subseteq N$ de taille totale au moins égal à C dont la somme de leur profit respectif est la moins élevée possible, (sélection des articles les moins profitables), ne fait pas partie de l'énoncé. Le dual du problème de maximisation de profit ne correspond pas à celui de la minimisation du manque à gagner.

Du point de vue complexité (en temps), le problème du sac à dos 0–1 et ses variantes telles que le problème du sac à dos 0-1 multiples sont des problèmes NP-difficiles. Le problème du sac à dos 0-1 unique admet un schémas d'approximation en temps entièrement polynomial (FPTAS) tandis que la variante multiple n'admet qu'un schéma d'approximation en temps polynomial (PTAS). Bien que la programmation dynamique permettent de résoudre ces problèmes en un temps pseudo-polynomial, respectivement, $O(n \cdot C)$ et $O(n \cdot m \cdot C)$, la programmation linéaire offre de nombreux avantages lors de leur résolution. En effet, les méthodes exploitant la relation primale-duale fournissent une technique intéressante pour concevoir des algorithmes d'approximation pour les problèmes NP-difficiles, en s'appuyant sur la dualité du programmation linéaire pour trouver des valeurs à un facteur garanti près de la valeur optimale.

2 Taches

On vous demande de

1. — Pour la version 0/1 à sac-à-dos unidimensionnels multiples du problème minKP de i) Formuler le modèle de programmation entière, ii) Définir précisément les paramètres, les variables, les contraintes, et la fonction objectif du modèle, et iii) Spécifier les hypothèses afin d'éviter les solutions triviales, et les conditions de faisabilité
   — Chercher un contexte d'utilisation du problème maxKP à sac à dos unidimensionnels multiples (tel que logistique, allocation de ressources, etc.). Justifier en expliquant dans ce cadre la signification des paramètres (C, des vecteurs p et w), de l'ensemble N d'items/objets à disposition, de ceux sélectionnés $\hat{N}$, et de l'ensemble M de sac à dos unidimensionnels.
   — Chercher un contexte d'utilisation du problème minKP à sac à dos unidimensionnels multiples (tel que logistique, satisfaction de demandes, etc.). Justifier en expliquant dans ce cadre la signification des paramètres (D, des vecteurs c et s), de l'ensemble N d'items/objets à disposition, de ceux sélectionnés $\hat{N}$, et de l'ensemble M de sac à dos unidimensionnels.
   Note : si vos exemples sont extraits ou s'inspirent d'un manuel ou d'un article veuillez citer leurs références.
2. Formuler la relaxation linéaire des modèles minKP à sac a dos unidimensionnel unique et multiples. Elaborer votre réponse en fonction de la qualité de la relaxation. A partir de cette formulation du programme linéaire primal, établir le programme linéaire dual pour chacun de ces deux modèles.
   — Dans le formulation du dual du programme linéaire, les variables duales correspondront aux contraintes du primal du programme linéaire et les contraintes du dual de ce programme linéaire aux variables x_j (x_{ij}) du primal du programme linéaire.
3. Analyser au moyen des instances fournies (voir Section 3), pour le modèle minKP à sac à dos unidimensionnel unique et celui à sac à dos unidimensionnels multiples, la relation primale-duale en tenant compte de la classe de complexité du problème. Dans votre analyse, on vous demande au minimum de :
   — Vérifier notamment l'égalité des valeurs optimales du primal et du dual pour la relaxation linéaire.

— Discuter l'écart d'intégralité (integrality gap) observé entre la solution entière optimale et celle de la relaxation linéaire.
— Commenter ce que cette observation peut suggérer concernant la complexité du problème ou la qualité des bornes fournies par la relaxation linéaire.
— Expliquer brièvement comment on pourrait résoudre la relaxation linéaire du problème en temps polynomial sans utiliser un solveur linéaire (et donc sans utiliser le simplexe).

4. Interprétez les solutions obtenues par votre programme sur les instances `multi_minKP_15_3_out.txt` et `multi_minKP_15_5_out.txt`.

   Bonus : bien que la programmation dynamique fournisse une méthode de choix pour la résolution de ces problèmes, donner (et justifier) trois raisons objectives pour lesquelles la programmation linéaire reste attractive pour la résolution de problèmes de type sac à dos (en particulier, lorsque la formulation du problème comprend des contraintes multiples).

3 Instances

Un répertoire contenant les instances `mono_minKP_n_m.txt` pour le problème minKP à sac a dos unidimensionnel unique et `multi_minKP_n_m.txt` pour le problème minKP à sac a dos unidimensionnels multiples vous est fourni. Les fichiers sont au format suivant (le template est disponible à la Figure 1) :

— `n` et `m` dans le nom du fichier correspondent respectivement au nombre d'objets et de sacs à dos.

Chaque fichier contient cinq lignes.

— La première ligne du fichier contient l'entier m, i.e. le nombre de sacs à dos (le nombre de demandes).
— La seconde ligne contient l'entier n, i.e. le nombre d'objets.
— La troisième ligne contient m entiers séparés par des espaces correspondant aux tailles des m demandes.
— la quatrième ligne contient n entiers séparés par des espaces correspondant aux poids des n objets.
— La dernière ligne contient n entiers séparés par des espaces correspondant aux coûts des n objets.

```
NB_KNAPSACKS
NB_ITEMS
DEMAND_SIZE_0 DEMAND_SIZE_1 ... DEMAND_SIZE_M
WEIGHT_ITEM_0 WEIGHT_ITEM_1 ... WEIGHT_ITEM_N
COST_ITEM_0 COST_ITEM_1 ... COST_ITEM_N
```

FIGURE 1 – Template des fichiers d'instance

4 Documents à remettre

1. Génération du modèle :
   Un script python3 nommé `minKP.py` prenant en paramètres en ligne de commande le nom d'une instance dans le même dossier et un paramètre p égal à 0, 1 ou 2. Ce script doit permettre de générer, résoudre et afficher la solution optimale du programme linéaire associée à cette instance à l'aide de la librairie PuLP comme vu en TP. La valeur de p indique quel modèle générer (vous pouvez vous référer à la première ligne du fichier d'instance pour savoir si vous devez utiliser la version à sac à dos unique ou multiple) :
   — si $p = 0$, résoudre le modèle primal en nombres entiers
   — si $p = 1$, résoudre la relaxation du modèle primal
   — si $p = 2$, résoudre le dual de la relaxation

   La résolution du primal en nombre entiers de l'instance `mono_minKP_15_1.txt` se fera donc avec la commande :

   ```
   python3 minKP.py mono_minKP_15_1.txt 0
   ```

2. Rapport :
   Un **rapport scientifique** rédigé en LaTeX et compilé au format `pdf` qui détaille vos formulations et vos résultats. Ce rapport doit contenir au minimum :
   - Vos noms, prénoms, matricules et année d'étude.
   - Une réponse à tous les points de la Section 2.
   - Vous pouvez ajouter toute information que vous pensez être utile au rapport ainsi que toutes les hypothèses que vous avez faites. Nous insistons sur le fait de remettre un **rapport scientifique**, un esprit critique est donc demandé lors de la réalisation du projet.

5 Consignes de remise

Ce travail doit être réalisé en groupe. Chaque groupe est constitué de 1 ou 2 étudiants.
Pour remettre votre projet sur l'UV du cours INFO-F310, nous vous demandons de

- créer sur votre machine un répertoire local intitulé `NOM1_NOM2` (exemple : `PAPADIMITRIOU_CALLEBAUT`, sans espace) dans lequel vous placez les fichiers demandés (sans y inclure de fichier de données).
- compresser ce répertoire via un utilitaire d'archivage produisant un fichier `.zip` (aucun autre format de compression n'est accepté)
- soumettre le fichier archive `.zip`, et uniquement ce fichier, sur l'UV (une seule soumission par groupe)

Le projet est à remettre pour le **16 mai 2025 à 23h59** sur l'UV.
Tout manquement aux consignes ou retard sera sanctionné directement d'une note de **0/20** pour le projet.
Il est à souligner que toute tentative de fraude ou de plagiat détectée dans le code ou le rapport sera sévèrement sanctionnée (dans ce dernier cas, pour tous les groupes impliqués).

Le code python3 compte pour 25% de la note du projet et le rapport scientifique pour 75% de cette note. Veuillez tenir compte que la note du projet est automatiquement reportée pour la note finale de seconde session.