3 Complexité et approximation : le problème du sac à dos (en OCAML)

On rappelle que le problème SUBSETSUM est défini comme suit :

Instance : $x_0, \dots, x_{n-1} \in \mathbb{N}^*$ et $S \in \mathbb{N}$

Question : existe-t-il $I \subseteq [0 \dots n-1]$ tel que $\sum_{i \in I} x_i = S$?

On appelle *objet* un couple (v, w) d'entiers naturels non nuls, où v est la *valeur* de l'objet et w son *poids.*
On définit alors les deux problèmes de décision suivants :

- 0/1-KNAPSACK

  Instance : n objets (v_i, w_i) et deux entiers naturels V et W

  Question : existe-t-il $I \subseteq [0 \dots n-1]$ tel que $\sum_{i \in I} v_i \geq V$ et $\sum_{i \in I} w_i \leq W$?
- KNAPSACK

  Instance : n objets (v_i, w_i) et deux entiers naturels V et W

  Question : existe-t-il $x_0, \dots, x_{n-1} \in \mathbb{N}$ tels que $\sum_{i=0}^{n-1} x_i v_i \geq V$ et $\sum_{i=0}^{n-1} x_i w_i \leq W$?

3.1 Réductions

On admet que le problème SubsetSum est NP-complet.

1. Montrer que les problèmes KNAPSACK et 0/1-KNAPSACK sont dans NP.
2. En réduisant SUBSETSUM, montrer que 0/1-KNAPSACK est NP-complet.

   On cherche désormais à montrer que KNAPSACK est NP-complet. Pour cela, on considère une instance $F = (X, S)$, avec $X = x_0, \dots, x_{n-1}$ de SUBSETSUM et l'on définit :
   - $B = 1 + \max(x_0, \dots, x_{n-1}, S)$;
   - $y_i = (2^n + 2^i)nB$ pour $0 \leq i < n$;
   - $y'_i = y_i + x_i$ pour $0 \leq i < n$;
   - $S' = \left(n2^n + \sum_{i=0}^{n-1} 2^i\right) nB + S$.

   On considère alors l'instance G de KNAPSACK où :
   - $V = S'$
   - $W = S'$
   - les $2n$ objets sont les couples (y_i, y_i) et (y'_i, y'_i) pour $0 \leq i < n$. *Chaque objet a donc une valeur égale à son poids.*
3. On suppose dans cette question que F est une instance positive de SUBSETSUM. Montrer que G est une instance positive de KNAPSACK.

   On suppose à présent que $\sum_{i=0}^{n-1} \lambda_i y_i + \sum_{i=0}^{n-1} \lambda'_i y'_i = S'$, avec les λ_i et λ'_i dans $\mathbb{N}$. On note $N = \sum_{i=0}^{n-1} (\lambda_i + \lambda'_i)$.
4. Montrer que $S' \geq 2^n nBN$ puis que $S' < 2^n(n+1)B$.
   En déduire que que $N \leq n$.

5. Montrer qu'on a les deux égalités suivantes (on pourra utiliser l'unicité du quotient et du reste dans la division euclidienne de S' par nB) :
   - $S = \sum_{i=0}^{n-1} \lambda'_i x_i$
   - $N2^n + \sum_{i=0}^{n-1}(\lambda_i + \lambda'_i)2^i = n2^n + \sum_{i=0}^{n-1} 2^i$

   On admet le résultat suivant :
   si $\sum_{i=0}^{n-1} \alpha_i 2^i \equiv \sum_{i=0}^{n-1} 2^i \mod 2^n$ et si $\sum_{i=0}^{n-1} \alpha_i \leq n$, alors $\alpha_0 = \cdots = \alpha_{n-1} = 1$.
6. Montrer que Knapsack est NP-complet.

3.2 Algorithmes d'approximation

Dans cette partie et la suivante, on s'intéresse à la variante « optimisation » des problèmes Knapsack et 0/1-Knapsack :

- 0/1-Knapsack$_o$
  Instance : n objets (v_i, w_i) et un entier W
  Solution : une partie $I \subseteq [0 \ldots n-1]$ telle que $\sum_{i \in I} w_i \leq W$
  But : maximiser $\sum_{i \in I} v_i$
- Knapsack$_o$
  Instance : n objets (v_i, w_i) et un entier W
  Solution : n entiers naturels $\lambda_0, \ldots, \lambda_{n-1}$ tels que $\sum_{i=0}^{n-1} \lambda_i w_i \leq W$
  But : maximiser $\sum_{i=0}^{n-1} \lambda_i v_i$

Dans toute la suite, on suppose que tous les w_i sont inférieurs ou égaux à W (sans perte de généralité, puisque les objets ne vérifiant pas cette condition sont inutiles et peuvent facilement être éliminés).

On considère l'algorithme glouton suivant :

- on trie les objets par valeur décroissante de v_i/w_i ;
- on considère les objets dans cet ordre, et l'on prend chaque objet autant de fois que possible (étant donné la capacité qui reste disponible).

Par exemple, pour l'instance $o_0 = (10, 4)$, $o_1 = (20, 6)$, $o_2 = (3, 3)$ et $W = 17$:
- on considère o_1 en premier et l'on fixe $\lambda_1 = 2$;
- on considère ensuite o_0 et l'on fixe $\lambda_0 = 1$;
- on considère finalement o_2 et l'on fixe $\lambda_2 = 0$.

7. On note v^* la valeur totale d'une solution optimale pour une instance $(v_0, \ldots, v_{n-1}), (w_0, \ldots, w_{n-1}), W$ (que l'on suppose triée par v_i/w_i décroissants). Montrer que $v^* \leq \frac{v_0}{w_0} W$.
8. Montrer que l'algorithme 1 (au verso) fournit une $\frac{1}{2}$-approximation pour Knapsack (on pourra commencer par justifier que $2\lambda_0 w_0 \geq (1 + \lambda_0) w_0 > W$).

Début algorithme

Trier les objets par v_i/w_i décroissant.
$\lambda \leftarrow (0, \dots, 0)$ (taille n)
$R \leftarrow W$
Pour $i = 0$ à $n-1$ **Faire**
  Poser $\lambda[i]$ maximal tel que $\lambda[i] \cdot w_i \leq R$
  $R \leftarrow R - \lambda[i] \cdot w_i$
Renvoyer λ

Algorithme 1 : Algorithme glouton pour KNAPSACK$_o$

On considère une instance de KNAPSACK donnée par deux tableaux `v` et `w` de même longueur n et un entier `capacity`.

On suppose disposer d'une fonction `by_ratio : int array -> int array -> int array` qui prend en entrée les tableaux `v` et `w` et renvoie un tableau `indices` de taille n tel que :

— les `indices` sont exactement les entiers de 0 à $n-1$;

— si `i < j`, alors $\dfrac{v.(indices.(i))}{w.(indices.(i))} \geq \dfrac{v.(indices.(j))}{w.(indices.(j))}$.

9. Écrire une fonction OCaml `greedy` telle que l'appel `greedy v w capacity` renvoie le tableau `lambda` obtenu par l'algorithme ci-dessus. Les tableaux `v` et `w` seront supposés de même longueur et correspondent respectivement aux valeurs et poids des objets ; l'entier `capacity` correspond à W.

```
val greedy : int array -> int array -> int -> int array
```

10. Déterminer la complexité de `greedy` en fonction du nombre n d'objets.
11. En considérant l'instance avec deux objets $(1, 1)$ et $(N-1, N)$ et $W = N$. Montrer qu'en appliquant le même principe pour 0/1-KNAPSACK$_o$, on n'obtient pas une α-approximation (quel que soit $\alpha > 0$).

On considère la *relaxation* FRACTIONAL-0/1-KNAPSACK$_o$ du problème 0/1-KNAPSACK$_o$ dans laquelle on s'autorise à prendre une quantité fractionnelle de chaque objet :

Instance : n objets (v_i, w_i) et un entier W

Solution : n **rationnels** $\lambda_0, \dots, \lambda_{n-1} \in [0, 1]$ tels que $\sum_{i=0}^{n-1} \lambda_i w_i \leq W$

But : maximiser $\sum_{i=0}^{n-1} \lambda_i v_i$

12. On considère un ensemble $(v_i, w_i)_{0 \leq i < n}$ d'objets, et l'on note V^* la valeur optimale associée pour 0/1-KNAPSACK$_o$ et V_f^* la valeur optimale associée pour FRACTIONAL-0/1-KNAPSACK$_o$. Montrer que $V^* \leq V_f^*$.
13. Proposer un algorithme glouton très simple résolvant FRACTIONAL-0/1-KNAPSACK$_o$ de manière optimale (ce que l'on justifiera très rapidement), en temps $O(n \log n)$.