Correction TD 8 : Algorithmes gloutons

Pascal Vanier

May 4, 2013

Exercice 1 : Le problème du rendu de monnaie

1. Pour rendre la monnaie sur 263 centimes d'euros, on rend une pièce de 200, de 50, de 10, de 2 et de 1.

2. L'algorithme glouton pour ce problème est le suivant : on rend toujours la pièce de la plus grande valeur que l'on peut. On suppose que les valeurs des pièces sont stockées par ordre croissant dans un tableau `p` :

```
rendre_pieces(entier S):
-- rendu := 0
-- i := 0
-- r := tableau rempli de 0 de même taille que p
-- tant que (S-rendu != 0) faire
---- si p[i] > S-rendu alors
------ i := i+1
---- sinon alors
------ r[i] := r[i] + 1
------ rendu := rendu + p[i]
-- retourner r
```

   Attention! Cet algorithme ne termine pas toujours, essayez de trouver quels sont les cas dans lesquels il ne termine pas.

3. Montrons que l'algorithme est optimal :

   - Montrons d'abors la propriété de choix glouton : Supposons qu'il existe $s > 200$ pour lequel le rendu de monnaie optimal ne contient pas de pièce de 200. On peut avoir au maximum une pièce de 100, car sinon ou pourrait remplacer les deux pièces de 100 par une de 200, au maximum une pièce de 50, 4 pièces de 10, 1 pièce de 5 et 2 pièces de 2 (sinon on peut remplacer par 1 pièce de 5 et une pièce de 1) et 1 pièce de 1. La somme maximale que l'on peut donc obtenir de manière optimale sans pièce de 200 est $1 + 2 \times 2 + 5 + 4 \times 10 + 50 + 100 = 200$ qui est inférieure ou égale à 200 (et qui pourrait être remplacée par 200).
   - Montrons la propriété de sous-structure optimale. Soit P une solution optimale pour une somme s et c le choix glouton, $P' = P \setminus \{c\}$ est une solution optimale pour $s - c$, car si il y a une meilleure solution P'' pour $s - c$, alors $P'' \cup \{c\}$ est une meilleure solution que P pour s.

4. Si les pièces ont comme valeurs 1,3,4, alors l'algorithme glouton va rendre la monnaie sous la forme d'une pièce de 4 et de deux pièces de 1, alors que rendre deux pièces de 3 aurait été optimal, il ne fournit donc pas la solution optimale.

Exercice 2 : Faire le plein sans panne sèche

La méthode glouton pour ce problème consiste à utiliser à chaque fois la dernière station avant la panne sèche. Montrons maintenant que cet algorithme est optimal.

Notons la solution optimale $y_1, \ldots, y_p$ et la solution gloutonne $g_1, \ldots, g_q$, on a donc $p \leq q$. Comme l'on prend la dernière station avant d'être en panne, on a g_1 qui est au moins aussi loin que y_1, inductivement, on déduit que g_k est au moins aussi loin que y_k, et donc que l'algorithme glouton donne la solution optimale.

Exercice 3 : Un problème d'ordonnancement

1. Soit H un ordonnancement, on peut diviser H en deux sous-ensembles de tâches F et G, où F est le sous-ensemble des tâches effectuées avant l'échéance et est donc un ensemble indépendant et G contient le reste des tâches. Notons p_I la somme des pénalités d'un ensemble de tâches I. On a la relation suivante :

   $$p_E = p_G + p_F$$

   Le calcul d'un ordonnancement optimal revient à trouver un ordonnancement minimisant p_G, ce qui revient à trouver un ordonnancement maximisant p_F. Comme F est indépendant, le calcul d'un ordonnancement optimal revient à trouver un ensemble indépendant maximal de E.

2. Soit F un ensemble indépendant optimal, par définition, toute les tâches de F ont des pénalités inférieures ou égales à w_i, on peut donc remplacer n'importe quelle tâche de F dont la date d'échance est inférieure à d_i par i, et obtenir ainsi un ensemble indépendant dont la somme des pénalités est plus grande ou égale à celle de F.

3. Voici un algorithme glouton pour le problème :

   ```
   ordonnancement(tableau d, tableau w)
   -- L := tri des i par w[i] décroissants
   -- F := vide
   -- pour i=1 jusqu'à n faire :
   ---- si F + L[i] indépendant alors
   ------ F := F + L[i]
   -- retourner F
   ```

   Si F est ordonné par i croissants et plus par poids, alors vérifier que F est indépendant se fait en temps au plus n.

   Montrons que cet algorithme est optimal, après avoir montré la propriété du choix glouton à la question 2, montrons maintenant la propriété de sous-structure optimale : Soit F une solution optimale, et i la tâche effectuée le plus tôt, alors $F' = F \setminus \{i\}$ est une solution optimale pour le problème $E' = \{j : d_j > t_i\}$ où t_i est la date à laquelle i est effectuée dans F : En effet, supposons que F' ne soit pas optimal pour E', et que la solution optimale soit G, alors $G \cup \{i\}$ serait une solution optimale pour E et F n'en serait pas une.

Exercice 4 Un algorithme glouton possible est de trier l'ensemble par ordre décroissant et d'ajouter ensuite chaque nombre au sous-ensemble dont le somme est la plus petite. Cela donne respectivement pour les deux ensmbles de l'exemple :

- $E_1 = \{10, 7, 5, 3, 2\}$ et $E_2 = \{9, 8, 5, 3, 2\}$ les sommes sont les mêmes : 27.
- $E_1 = \{1003, 771, 486, 281, 83\}$ et $E_2 = \{885, 854, 734, 121, 62\}$ les sommes ne sont pas les mêmes mais sont proches : 2624 et 2656.

L'algorithme glouton n'est pas optimal mais est une 4/3-approximation polynômiale. Par ailleurs, ce problème est NP-complet.