EISTI Ecole Internationale des Sciences du Traitement de l'Information	**CORRIGE DUTD DE PROGRAMMATION DYNAMIQUE**
Rédigé par : Hervé de Milleville	Ref : CORR-TD-PRO-DYN
Revu par : Jean-Paul Forest et Yannick Le Nir	Approuvé par :

1 Rappels

La programmation dynamique est basée sur le principe de Bellmann : toute sous solution d'une solution optimale est elle-même optimale.
En fait, ce principe n'est vérifié que si la fonction objectif est décomposable au sens suivant :
$f(x_1,...,x_n) = h\big(g(x_1,...,x_{n-1}), x_n\big)$ où h est une fonction de R^2 dans R ayant la propriété suivante :

$$\forall \beta, la\ \ fonction\ définie\ \ par\ \alpha \longrightarrow h_\beta(\alpha) = h(\alpha,\beta)\ est\ croissante$$

Si ce principe est vérifié alors : $Max\ f(x_1,...,x_n) = h\big(Max\ g(x_1,...,x_{n-1}), x_n\big)$

2 Exercices

2.1 Isopérimétrie

L'extrait qui suit provient de l'article wikipédia Isopérimétrie.

*En géométrie euclidienne, l'**isopérimétrie** est initialement l'étude des propriétés des formes géométriques du plan qui partagent le même périmètre, ce qui se généralise ensuite dans les autres espaces euclidiens.*
En particulier, le problème le plus classique consiste à déterminer la forme géométrique plane qui maximise son aire avec un périmètre fixé. La réponse est intuitive, c'est le disque, mais malgré son apparence anodine ce problème fait appel à des théories sophistiquées pour obtenir une démonstration rigoureuse (particularité qu'il partage par exemple avec le théorème de Jordan qui indique qu'une boucle tracée sans croisement divise le plan en deux parties). Pour cette raison on simplifie parfois le problème isopérimétrique en limitant les surfaces autorisées, par exemple en se restreignant aux seuls quadrilatères ou triangles, ce qui donne alors respectivement le carré et le triangle équilatéral.

***De manière générale, le polygone à n côtés ayant la plus grande surface, à périmètre donné, est celui qui se rapproche le plus du cercle : c'est le polygone régulier**.*

2.1.1 Enoncé :

Sur la base de cet article, nous allons résoudre un problème équivalent.
On donne une corde de longueur l. On forme avec cette corde un polygone de n côtés. On cherche parmi tous ces polygones celui maximise le produit des côtés.

2.1.2 Questions

- Modéliser ce problème
- Montrer qu'il est dans la classe des problèmes de programmation dynamique
- Résoudre ce problème en appliquant le principe de Bellmann

2.1.3 Corrigés

Modéliser ce problème :

Variables de décisions : x_1, x_2, …,x_n les différentes longueurs des côtés du polygone.

Paramètres : l la longueur de la ficelle et n le nombre de côtés du polygone.

Contraintes : $\sum_{i=1}^{n} x_i = l$ et $\forall i \in \{1,...,n\}\ x_i \geq 0$

Fonction objectif : $f(x_1,...,x_n)=\prod_{i=1}^{n} x_i$

Montrer qu'il est dans la classe des problèmes de programmation dynamique :

On pose $g_n(x_1,...,x_{n-1})=\prod_{i=1}^{n-1} x_i$ et $h(\alpha,\beta)=\alpha*\beta$. On a bien $f(x_1,...,x_n)=h_n(g_n(x_1,...,x_{n-1}),x_n)$

La fonction $h_\beta(\alpha)=\alpha*\beta$ est croissante pour β positif ou nul. La fonction objectif est donc décomposable au sens de Bellmann.

Résoudre ce problème en appliquant le principe de Bellmann :

Raisonnons par récurrence :

Cas n = 2 : le problème se ramène à $Max_{0\le x\le l} t(x)=x*(l-x)$

$t'(x)=l-2x \Rightarrow t'(x)=0 \Leftrightarrow x=l/2$

$t''(x)=-2<0 \Rightarrow x=l/2$ est un maximum.

Cas n ≥ 3. Supposons qu'au rang n-1 la solution optimale est $x_1 = ... = x_{n-1} = l/(n-1)$.

On en déduit que $Max_{l=\sum_{i=1}^{n} x_i} f(x_1,...,x_n)=Max_{0\le x\le l} t(x)=\left(\frac{l-x}{n-1}\right)^{n-1}*x$

$$t'(x)=\left(\frac{l-x}{n-1}\right)^{n-1}-\frac{n-1}{n-1}*\left(\frac{l-x}{n-1}\right)^{n-2}*x=\left(\frac{l-x}{n-1}\right)^{n-2}\left(\frac{l-x}{n-1}-x\right)=\left(\frac{l-x}{n-1}\right)^{n-2}*\left(\frac{l-n*x}{n-1}\right)$$

$t'(x)=0 \Leftrightarrow x=\frac{l}{n}$ ou $x=l$.

Pour x =l, il s'agit d'un minimum car un côté est de longueur l et tous les autres sont de longueur nulle. La surface est donc nulle.

Pour $x = l/n$, on a donc $x_n=\frac{l}{n}$ et $x_1=...=x_{n-1}=\frac{l-l/n}{n-1}=\frac{l}{n}$. Il s'agit bien d'un maximum car la fonction t est concave. Pour le vérifier, il faut étudier le signe de t"(x).

En conclusion : le polygone qui maximise le produit des côtés pour un périmètre donné est le polygone régulier.

2.1.4 Pour finir, un petit problème de réflexion

On considère un polygone convexe régulier à n côtés. Chaque côté est de longueur a. On peut démontrer que la surface et le périmètre sont donnés par les formules qui suivent :

$$S=\frac{na^2}{4\tan(\frac{\pi}{n})} \quad \text{et} \quad P=na$$

- Montrer que si on prend deux polygones réguliers de même périmètre alors celui de plus grande surface est celui qui a le plus grand nombre de côtés.
- En déduire que parmi les figures régulières d'un périmètre donné, celle qui a la plus grande surface est un cercle.

2.2 Choix d'investissement

Une entreprise peut investir dans 4 projets. Elle dispose d'un budget maximum de 7 M€. Elle peut répartir son budget entre ces différents projets. Pour simplifier le problème, on suppose que chaque montant investi est un nombre entier de millions d'euros.

Montant investi en M€	Rendement			
	I1	I2	I3	I4
0.00	0.00	0.00	0.00	0.00
1.00	0.56	0.50	0.30	0.50
2.00	0.90	0.82	0.50	0.66
3.00	1.30	1.10	0.80	0.84
4.00	1.56	1.30	1.00	0.96
5.00	1.80	1.50	1.24	1.06
6.00	2.04	1.60	1.46	1.12
7.00	2.26	1.70	1.64	1.16

Vous devez aider cette entreprise à investir afin d'optimiser la somme des rendements.

2.2.1 Questions

- Modéliser ce problème
- Montrer qu'il est dans la classe des problèmes de programmation dynamique
- Résoudre ce problème en appliquant la méthode des tableaux

2.2.2 Corrigés

Modéliser ce problème :

Variables de décisions : x_1, x_2, x_3 et x_4 les montants investis dans chaque projet.

Paramètres : $r_{i,x}$ le rendement du i^{ème} projet pour un investissement de x millions d'euros.

Contraintes : $\sum_{i=1}^{n} x_i \leq 7$ et $\forall i \in \{1,...,n\} x_i \in \mathrm{N}$

Fonction objectif : $f(x_1,...,x_n) = \sum_{i=1}^{n} r_{i,x_i}$

Montrer qu'il est dans la classe des problèmes de programmation dynamique :

La fonction objectif est de la forme $f(x_1,...,x_n) = \sum_{i=1}^{n} f_i(x_i)$. Cette fonction est donc décomposable.

Résoudre ce problème en appliquant la méthode des tableaux :

Dans la i^{ème} colonne, on n'investit que dans les i premiers projets. Dans chaque colonne, on a des lignes formés de trois valeurs :

- 1ère valeur : montant total investi
- 2ème valeur : rendement optimal pour le montant investi associé
- 3ème valeur : montant total investi de la colonne précédente pour ce rendement optimal. Cette colonne permet de retrouver le chemin optimal à la fin de l'algorithme

1er projet			2ème projet			3ème projet			4 ème projet
0.00	0	-	0.00	0	0	0.00	0	0	A finir mais inutile
1.00	0.56	-	1.00	0.56	1.00	1.00	0.56	1.00	A finir mais inutile
2.00	0.90	-	2.00	1.06	1.00	2.00	1.06	2.00	A finir mais inutile
3.00	1.30	-	3.00	1.40	2.00	3.00	1.40	3.00	A finir mais inutile
4.00	1.56	-	4.00	1.80	3.00	4.00	1.80	4.00	A finir mais inutile
5.00	1.80	-	5.00	2.12	3.00	5.00	2.12	5.00	A finir mais inutile
6.00	2.04	-	6.00	2.40	3.00	6.00	2.42	5.00	A finir mais inutile
7.00	2.26	-	7.00	2.66	4.00	7.00	2.70	6.00	7.00	2.72	5.00

Comment lit-on le résultat :

- 4ème colonne, le montant optimal est 2.72 M€. Le montant investi avec les 3 premiers projets est de 5 M€ (donc on a investi 2 M€ dans le 4ème projet).
- 3ème colonne, on lit la ligne avec 5 M€ investi dans les 3 premiers projets. On en déduit le montant investi avec les 2 premiers projets soit 5 M€ (donc on a investi 0 M€ dans le 3ème projet).
- 2ème colonne, on lit la ligne avec 5 M€ d'investi dans les 2 premiers projets. On en déduit le montant investi avec le premiers projet soit 3 M€ (donc on a investi 2 M€ dans le 2ème projet).
- Le parcours inverse est terminé car on connait le montant investi dans le 1er projet soit 3 M€.

Pour bien comprendre la méthode des tableaux, nous expliquons comment on a obtenu les tableaux.

Le premier tableau est très simple car on n'utilise que le 1er projet. Il y a 8 lignes car on peut investir 0 M€, 1 M€, …, 7 M€. La première colonne contient la quantité de ressources utilisées (en l'occurrence le montant investi). La deuxième colonne contient le rendement optimum pour le montant investi. Sachant qu'on n'utilise que le 1er projet il suffit pour remplir cette colonne de recopier les rendements de la colonne P1 qu'on trouve dans les données initiales. La troisième colonne n'a pas de sens pour ce premier tableau car il n'y a pas de tableau précédent.

Les trois autres tableaux sont remplis à l'aide des données initiales et du tableau précédent. Prenons l'exemple du triplet sur fond jaune du troisième tableau. La première valeur indique qu'on dépense 3 M€ avec les trois projets. On a donc 4 montants possibles pour le 3ème projet : 0 M€, 1 M€, 2 M€ ou 3 M€. Dans les données initiales, on sait que les rendements respectifs pour ce troisième projet sont 0.00 M€, 0,30 M€, 0,50 M€ et 0,80 M€.

- Si on dépense 0 M€ pour le 3ème projet alors on a dépensé 3 M€ pour les 2 premiers projets. Dans le tableau précédent, le rendement optimum pour 3 M€ est de 1.40 €. Le rendement total est donc de 0 M€ + 1,40 M€ = 1.40 M€
- Si on dépense 1 M€ pour le 3ème projet alors on a dépensé 2 M€ pour les 2 premiers projets et on a un rendement de 0,30 € pour ce 3ème projet. Dans le tableau précédent, le rendement optimum pour 2 M€ est de 1.06 €. Le rendement total est donc de 0,30 M€ + 1,06 M€ = 1.36 M€
- Si on dépense 2 M€ pour le 3ème projet alors on a dépensé 1 M€ pour les 2 premiers projets et on a un rendement de 0,50 € pour ce 3ème projet. Dans le tableau précédent, le rendement optimum pour 1 M€ est de0.56 €. Le rendement total est donc de 0,50 M€ + 0.56 M€ = 1.06 M€
- Si on dépense 3 M€ pour le 3ème projet alors on a dépensé 0 M€ pour les 2 premiers projets et on a un rendement de 0,80 € pour ce 3ème projet. Dans le tableau précédent, le rendement optimum pour 0 M€ est de 0.00 €. Le rendement total est donc de 0,80 M€ + 0.00 M€ = 0;80 M€

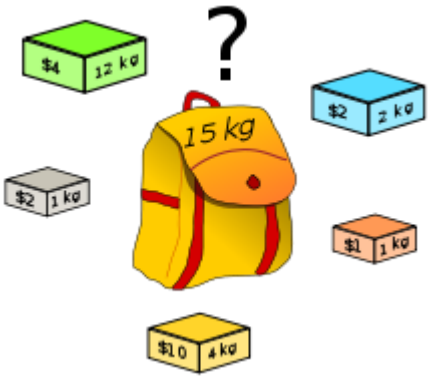
La rendement optimum est donc 1.40 M€ et correspond à utiliser 3 M€ pour les deux premiers projets. On obtient donc le triplet (0,30 M€, 1,40 M€, 0,30 M€).

Dernière remarque : Dans la 4ème colonne, nous n'avons pas besoin de calculer les rendements optimums pour des montants investis inférieur à 7 M€ car on a tout intérêt à tout dépenser.

2.3 Problème du sac à dos

En algorithmique, le **problème du sac à dos**, noté également **KP** (en anglais, *Knapsack Problem*) est un problème d'optimisation combinatoire. Il modélise une situation analogue au remplissage d'un sac à dos, ne pouvant supporter plus d'un certain poids, avec tout ou partie d'un ensemble donné d'objets ayant chacun un poids et une valeur. Les objets mis dans le sac à dos doivent maximiser la valeur totale, sans dépasser le poids maximum.

CORRIGE DUTD DE PROGRAMMATION DYNAMIQUE



Le problème du sac à dos : quelles boîtes choisir afin de maximiser la somme emportée tout en ne dépassant pas les 15 kg autorisés ?

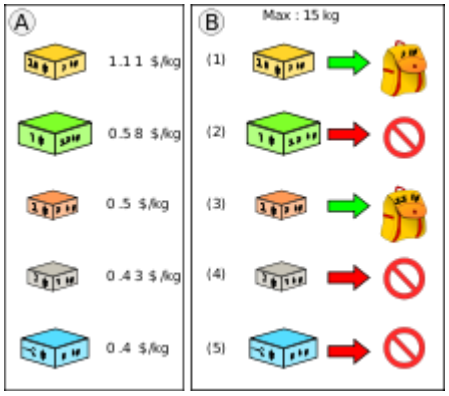
L'article et l'image et empruntée sur le site wikipédia sur le thème du problème du sac à dos

2.3.1 Questions

- Modéliser ce problème
- Commenter l'algorithme glouton qui suit :

```
trier les objets par ordre décroissant
d'efficacité
w_conso := 0

pour i de 1 à n
  si w[i] + w_conso ≤ W alors
    x[i] := 1
    w_conso := w_conso + w[i]
  sinon
    x[i] := 0
  fin si
fin pour
```



- Montrer qu'il est dans la classe des problèmes de programmation dynamique
- Résoudre ce problème en appliquant la méthode des tableaux
- Que cela change-t-il dans la modélisation du problème si on propose plusieurs boites de chaque type ?

2.3.2 Corrigés

Modéliser le problème :

Paramètres :

- W le poids maximum à emporter
- w_i le poids du $i^{ème}$ objet pour i ∈ {1, 2, 3, 4, 5}
- v_i l'efficacitédu $i^{ème}$ objet pour i ∈ {1, 2, 3, 4, 5}

Variables de décisions : x_1, x_2, x_3, x_4 et x_5 le choix ou pas du $i^{ème}$ objet. X_i est donc égal à 0 ou 1.

Contraintes : $\sum_{i=1}^{n} x_i * w_i \leq W$ et $\forall i \in \{1,...,n\} x_i \in \mathrm{N}$

Fonction objectif : $f(x_1,...,x_n) = \sum_{i=1}^{n} v_{i_i} * x_{i_i}$

Commenter l'algorithme glouton qui suit :

Il risque de ne pas balayer toutes les solutions car il ne remet pas en cause des choix précédents, contrairement à la méthode des tableaux de la programmation dynamique.

Montrer qu'il est dans la classe des problèmes de programmation dynamique :

La fonction objectif est de la forme $f(x_1,...,x_n) = \sum_{i=1}^{n} f_i(x_i)$. Cette fonction est donc décomposable.

2.4 Programmer la méthode des tableaux

2.4.1 Enoncé

On vous demande d'écrire un programme pour résoudre des problèmes de programmation dynamique. On se restreindra au cas suivants :

- La fonction objectif est de la forme $f(x_1, \ldots, x_n) = f_1(x_1) + \ldots + f_n(x_n)$
- Chaque variable x_i ne prend que des valeurs entières.
- Il n'y a qu'une seule contrainte de la forme $\sum_{i=1}^{n} c_i.x_i \leq \lambda$