

## Concepts généraux de prog dynamique.

① Représentation d'une solution réalisable d'un problème comme une suite finie de décisions.

sac à dos : on a une suite de décisions. 1 décision concerne 1 objet.

On appelle une étape avec 1 décision  $\rightarrow$  1 phase

phase k  $\rightarrow$  objet k.

nombre d'étapes appelé : pour n  $\rightarrow$  Horizon

$D_k$   $\rightarrow$  l'ensemble des décisions possibles à la phase k.

$D_k = \{$  on prend k, on ne prend pas k  $\}$

② Définir l'état du système à chaque phase :

l'état résume ce qu'on a besoin de connaitre des décisions prises des phases 1 à k-1 pour savoir ce qu'on a le choix de décider par la suite.

sac à dos : l'état E à la phase k c'est la liste restante dans le sac (le poids restant) et pas la liste des objets mis dans le sac.

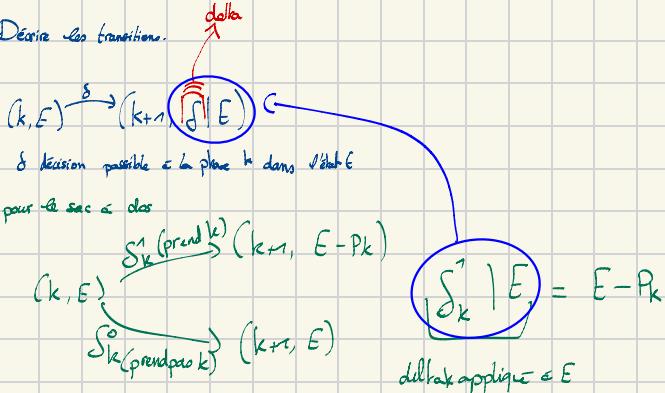
en gros l'état résume ce qu'on a besoin de savoir pour les futures décisions.

$E_k$  : l'ensemble des états possibles à la phase k.

$E_0$  : état initial

$\hookrightarrow$  pour le sac à dos :  $E_0 = P$ , P est le poids max.

③ Décrire les transitions.



④ profit/coust et solution optimale

→ chaque décision  $s_i$  prise dans la phase  $k$  à l'état  $E$  a un:

(coust)  
profit

c( $s_i, k, E$ )  
w( $s_i, k, E$ )

Le problème d'optimisation doit se formuler en:

trouver la meilleure suite n décisions à partir de l'état  $E_0$  qui maximise une fonction du type  $\boxed{\Sigma}$  Max/Min/TI des profits immédiats des décisions.

$F_k(E)$  = valeur optimale d'une suite de décisions sur les phases  $k$  à  $n$  sachant qu'à la début de la phase  $k$  on est dans l'état  $E$ .  
pour le sac à dos → valeur optimale d'un sac parmi les objets de  $k$  à  $n$  qui pèse au plus  $E$ .

On suppose que pour tous les  $E$  possibles  $F_n(E)$  se calcule facilement

pour le sac à dos:

$$F_m(E) \begin{cases} w \in P_m \\ 0 \text{ sinon} \end{cases}$$

si maximisation :

$$F_k(E) = \max_{\substack{\text{max} \\ \text{chacune des décisions}}} \underbrace{(w(s_i, k, E) + F_{k+1}(s_i | E))}_{\text{delta}} \quad \text{formule de récurrence.} \Rightarrow \text{relation de récurrence.}$$

$$F_k(E) \leftarrow F_{k+1}(E) \text{ ou } P_k(E)$$

si minimisation :

$$F_k(E) = \min_{s \in D_k(E)} (c(s, k, E) + F_{k+1}(s | E))$$

Algorithmique générique de programmation dynamique.

Étape 1 : initialisation

calculer  $F$  pour tous les  $E$  possibles à la phase  $n$  ( $E \in \mathcal{E}_n$ )

## Etape 2 : Calcul des $F_k(E)$

Pour  $k$  de  $n-1$  à 1 pour chaque  $E \in \mathcal{E}_k$ , on calcule  $F_k(E)$  avec  $\textcircled{*}$  (solution normale)

## Etape 3 : Calculer la solution optimale.

$$E = E_0$$

pour  $k$  de 1 à  $n-1$ :

on regarde pour quel  $S$

$$F_k(E) = w(S, k, E) + F_{k+1}(S|E)$$

ce  $S$  est la décision prise à la phase  $k$

$$E = S|E$$

On regarde quelle est la dernière décision qui a conduit à la valeur  $F_k(E)$

1)

prise sur machine  $k, 1$   
tâche  $i_k$  → non prise sur machine  $k, 0$

$T$  durée max par machine.

$a_i, b_i$

$w_i$  si tâche  $i_k = i$

- pas deux tâches sur deux machines

- max  $T$  sur la somme des tâches faites par machine.

$\max w_i$  des tâches faites

2)

step 1 : phases

phase = tâche

horizon =  $n - le nb$  de tâches

step 2 : décisions

3 décisions possibles:

- $k$  sur machine 1
- $k$  sur machine 2
- $k$  non fait.

step 3 : états :

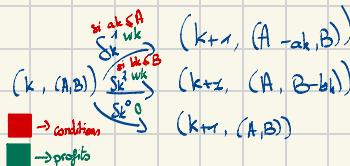
$(A, B)$

$$E_0 = (M, M)$$

step 4 : transitions

$A \rightarrow$  durée restante sur machine 1

$B \rightarrow$  durée restante sur machine 2



→ condition

→ profit

Exercice 3.5 Ordonnancement sur deux machines de profit maximum

Un atelier de production dispose de deux machines qui peuvent fonctionner en parallèle. En début de mois un ensemble de commandes (tâches) est proposé à l'atelier pour réalisation. La plupart du temps, il y en a trop pour être réalisées. Le responsable d'atelier doit donc décider quelles sont les commandes qui sont acceptées par son atelier (celles qui peuvent être réalisées dans le mois), et sur quelle machine elles s'effectuent. Pour une tâche  $i$  la durée n'est pas la même selon la machine qu'on utilise (durées respectives  $a_i$  et  $b_i$ ). De plus, le fait d'accepter une tâche  $i$  induit un profit  $w_i$  pour l'entreprise qui peut la facturer au client. On cherche donc à ordonner les tâches sur les deux machines de sorte que la durée des tâches sur chacune des machines soit inférieure à la durée ouvrable du mois  $M$  en maximisant le profit de l'atelier.

Question 1 : Modéliser ce problème à l'aide d'un programme mathématique.

On définit  $F_k(A, B)$  le profit maximal d'un ordonnancement des tâches  $k$  à  $n$  pour lequel la durée des tâches sur la machine 1 est inférieure ou égale à  $A$ , celle sur la machine 2 est inférieure ou égale à  $B$ .

Question 2 : Décrire le schéma de programmation dynamique induit par cette définition (phases, états, décisions, transitions, etc)

Question 3 : Donner la valeur de  $F_n(A, B)$  en fonction des données du problème et de  $A, B$

Question 4 : Indiquer l'équation de récurrence satisfait par  $F_k(A, B)$ .

Question 5 : Ecrire l'algorithme qui découle de ces égalités permettant de trouver un ordonnancement optimal. On précisera sa complexité.

$$f_k(A, B) = \begin{cases} \max(w_k + f_{k+1}(A - a_k, B), w_k + f_{k+1}(A, B - b_k), f_{k+1}(A, B)) & \text{si } a_k \leq A \text{ et } b_k \leq B \\ \max(w_k + f_{k+1}(A - a_k, B), f_{k+1}(A, B)) & \text{si } a_k > A \text{ et } b_k \leq B \\ \max(w_k + f_{k+1}(A, B - b_k), f_{k+1}(A, B)) & \text{si } a_k \leq A \text{ et } b_k > B \\ f_{k+1}(A, B) & \text{sinon} \end{cases}$$

Valeurs optimales :  
C'est en fait pour la dernière colonne.

$$f_n(A, B) = \begin{cases} w_n & \text{si } a_n \leq A \text{ et } b_n \leq B \\ 0 & \text{sinon} \end{cases} \rightarrow \begin{array}{l} \text{si la dernière tâche} \\ \text{peut entrer dans machine} \\ \text{and...} \end{array}$$

Complexité : pour les états  $\mathbb{N}^2$  car les états c'est un couple de temps  $(A, B)$  allant de  $0 \in \mathbb{N}$  chaque tâche.

phase.  
état  
décisions  
transition  
coût immédiat.  
step 1 : phases

phase =  
horizon =

### Exercice 1.3 Gestion de stock

Une usine automobile cherche à planifier sa production sur les  $H$  prochaines semaines. Elle dispose de ses prévisions, qui donnent pour chaque semaine  $i$  un nombre de véhicules  $D_i$  à livrer en fin de semaine.

Selon les aléas de la production (notamment congés, coût des transports, etc), le coût de production des véhicules varie selon les semaines. On note  $c_i$  le coût de production d'un véhicule durant la semaine  $i$ . Produire  $x$  véhicules pendant la semaine  $i$  coûte donc  $c_i x$ . L'usine a, de plus, une capacité de production fixe par semaine, notée  $Q$  (c'est à dire ne peut produire plus de  $Q$  véhicules par semaine).

Ces deux éléments peuvent conduire à produire des véhicules en avance pour une commande d'une semaine ultérieure. Les véhicules doivent donc être stockés. Chaque véhicule stocké à la fin de la semaine  $i$  a un coût unitaire de stockage noté  $t_i$ . (stocker  $x$  véhicules en fin de semaine  $i$  coûte donc  $t_i x$ ).

La production se déroule donc selon le schéma suivant : chaque semaine  $i$ , le stock restant de la semaine précédente est complété avec la production de la semaine, le client est livré en fin de semaine, et il peut éventuellement rester un stock pour la semaine suivante. L'entreprise cherche à calculer un plan de production, c'est à dire savoir combien de véhicules doivent être produits chaque semaine, de sorte que le coût total de production et de stockage soit minimum et que tous les clients soient livrés. Modéliser ce problème.

## contraintes:

- capacité de prod,  $Q_i$ , de l'usine  $i$  pour la semaine  $i$ :  $x_{ij} \in Q_i$  pour tout  $j \in \{1, \dots, n\}$ ,  $x_{ij} \in \mathbb{N}$ .
- stock restant de la semaine  $i$  après l'usage des stock  $s_i$   $S_i = S_{i-1} + x_{ij} - D_j$
- limite  $D_j$  pour la semaine  $i$ :  $S_i > 0 \rightarrow$  si il reste du stock, au final le stock est à zéro, ça veut dire on a assez produit si  $s_i \geq D_j$  pour aller de l'autre produit

## objectifs

$$\text{Min} \sum_{i=1}^n c_i x_{ij} + b_i s_i$$