Programmation dynamique

OUTILS POUR LA LOGISTIQUE
ECE - 3EME ANNEE
FILIERE *TRANSPORTS ET MOBILITE*

Cédric BENTZ (CNAM)

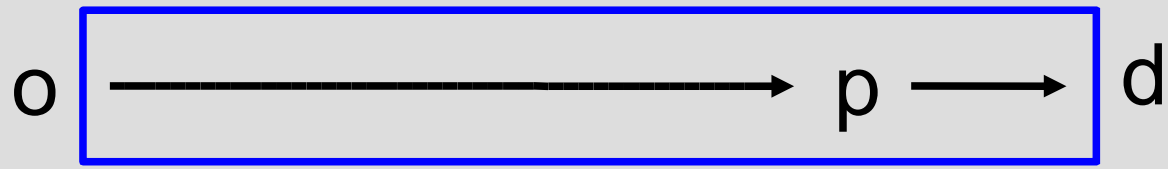
Principe d'optimalité de Bellman

- Vérifié par de nombreux problèmes d'optimisation en lien avec la logistique
- *Enoncé* : si l'on parvient à identifier une partie d'une solution optimale pour une instance donnée d'un tel problème, alors le reste de cette solution doit être optimal sur le reste de l'instance.

R. Bellman (1920-1984)

P.O. de Bellman : l'exemple du plus court chemin (itinéraire optimal)

- Un exemple simple : plus court chemin élémentaire entre deux sommets o (origine) et d (destination)
- Si on sait identifier le prédécesseur p de d dans un tel chemin (noté $pcc_{o-->d}$), alors la partie de $pcc_{o-->d}$ reliant o à p est un plus court chemin de o à p !



$pcc_{o-->d}$

Principe d'optimalité de Bellman : mise en garde et vertus

- Tous les problèmes d'optimisation (en lien avec la logistique ou non) ne respectent pas ce principe !
- Respecter ce principe ne garantit pas nécessairement l'existence d'un algorithme « efficace » (polynomial)...
- Montrer qu'un problème donné respecte ce principe est parfois complexe, et peut nécessiter une analyse très poussée de la structure de ses solutions optimales.
- Le respect de ce principe par un problème donné permet néanmoins, en général, de concevoir un algorithme pour ce problème basé sur la **programmation dynamique**.

Qu'est-ce que la programmation dynamique ?

- L'idée générale d'un algorithme de programmation dynamique est de déterminer la valeur optimale d'une instance d'un problème à partir des valeurs optimales d'instances plus petites du même problème.
- Ces valeurs optimales sont liées entre elles par des **équations de récurrence**, permettant de les calculer en suivant un certain ordre.
- En général, un tel algorithme se compose donc :
  - D'une fonction récursive,
  - De cas terminaux pour la récursion (dont on sait facilement déterminer la valeur),
  - D'un cas permettant d'initialiser la récursion.

Illustration de la programmation dynamique : le sac à dos (1/6)

- Rappel du problème du sac à dos :
  - Données : n objets, chacun muni d'un poids p_i et d'une valeur v_i, et un poids maximum Pmax.
  - Problème : sélectionner un ensemble d'objets de valeur totale maximum et de poids total au plus Pmax.
- Quelques applications parmi d'autres :
  - Sélection de fichiers, chacun ayant un poids en Mo (par exemple, des MP3), sur un support limité en espace de stockage (par exemple, un lecteur MP3), de façon à maximiser un certain « bénéfice ».
  - Stockage de marchandises (chacune ayant une taille / un encombrement associé) dans un container dont la taille est limitée, de façon à maximiser un certain « bénéfice ».

Illustration de la programmation dynamique : le sac à dos (2/6)

- Rappel du PLNE modélisant le sac à dos :

  $\max \Sigma_i v_i x_i$

  $\Sigma_i p_i x_i \leq Pmax$

  $x_i \in \{0,1\}$ pour tout i

- Analyse du problème :
  - On considère les $i \leq n$ premiers objets,
  - On suppose que le poids maximum autorisé est $P \leq Pmax$,
  - Comment calculer la valeur optimale pour cette instance ?
    - Cas 1 : le ième objet est sélectionné dans une solution optimale (impossible si $p_i > P$), auquel cas le poids restant est $P-p_i$.
    - Cas 2 : le ième objet n'est pas sélectionné dans une sol. optim.

Illustration de la programmation dynamique : le sac à dos (3/6)

- Fonction récursive :

  f(i,P) = valeur totale maximum d'un ensemble d'objets choisis parmi les i premiers, et de poids maximum P.

- Equations de récurrence (pour tout $i \geq 2$ et $P \geq 0$) :

  Si $p_i \leq P$: $f(i,P) = \max(\underbrace{v_i+f(i-1,P-p_i)}_{\text{i choisi}}, \underbrace{f(i-1,P)}_{\text{i non choisi}})$

Preuve ?

1) Il existe deux solutions admissibles de valeur $v_i+f(i-1,P-p_i)$ et $f(i-1,P)$, donc $f(i,P) \geq \max(v_i+f(i-1,P-p_i), f(i-1,P))$

2) L'objet i est choisi ou non, donc $f(i,P) \leq f(i-1,P)$ OU $f(i,P) \leq v_i+f(i-1,P-p_i)$

Illustration de la programmation dynamique : le sac à dos (4/6)

- Autres cas ?
  - Si $p_i > P$ (pour tout $i \geq 2$ et $P \geq 0$) : $f(i,P) = f(i-1,P)$
  - Pour tout $P \geq 0$: $f(1,P) = 0$ si $p_1 > P$, et $f(1,P) = v_1$ sinon

  (Cas terminal pour la récursion.)
- Initialisation de la récurrence ?
  - Calcul de f(n,Pmax)
- Temps d'exécution ?
  - Problème : ne pas calculer plusieurs fois une même valeur !
  - Solution : stocker les valeurs déjà calculées dans un tableau, et transformer le calcul des valeurs d'une fonction récursive en calcul des valeurs d'un tableau.

Illustration de la programmation dynamique : le sac à dos (5/6)

- Implémentation à l'aide d'un tableau t :
  - Pour tout $i \geq 1$ et $P \geq 0$, on note t[i][P] la case du tableau t où est stockée la valeur f(i,P).
  - Calcul de t[1][P] pour $P \geq 0$: O(1) opérations ==> O(Pmax) opérations élémentaires en tout
  - Calcul de t[i][P] pour tous i>1 et $P \geq 0$ (en supposant t[i-1][P'] connu, pour chaque $P' \geq 0$) : O(1) opérations élémentaires ==> O(n*Pmax) opérations en tout
  - Donc complexité globale = O(n*Pmax) opérations : algorithme « rapide » si Pmax « petit ».

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2								
i=1								

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2								
i=1	0	0	0	0	0			

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2								
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2	0							
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2	0	0	0					
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2	0	0	0	55				
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2	0	0	0	55	55			
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2	0	0	0	55	55	100		
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2	0	0	0	55	55	100	100	
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3								
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3	0							
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3	0	18						
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3	0	18	18					
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3	0	18	18	55				
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3	0	18	18	55	73			
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3	0	18	18	55	73	100		
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3	0	18	18	55	73	100	118	
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4								
i=3	0	18	18	55	73	100	118	118
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4	0							
i=3	0	18	18	55	73	100	118	118
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4	0	18	18	55				
i=3	0	18	18	55	73	100	118	118
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4	0	18	18	55	73			
i=3	0	18	18	55	73	100	118	118
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4	0	18	18	55	73	100		
i=3	0	18	18	55	73	100	118	118
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4	0	18	18	55	73	100	118	
i=3	0	18	18	55	73	100	118	118
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4	0	18	18	55	73	100	118	125
i=3	0	18	18	55	73	100	118	118
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

Illustration de la programmation dynamique : le sac à dos (6/6)

- Un exemple à quatre objets (n = 4 et Pmax = 7) :

Objet	1	2	3	4
Poids	5	3	1	4
Valeur	100	55	18	70

	P=0	P=1	P=2	P=3	P=4	P=5	P=6	P=7
i=4	0	18	18	55	73	100	118	125
i=3	0	18	18	55	73	100	118	118
i=2	0	0	0	55	55	100	100	100
i=1	0	0	0	0	0	100	100	100

==> VALEUR OPTIMALE = 125

Calcul de la solution optimale par chaînage arrière

- Retrouver la solution associée à cette valeur ?
  - Si $f(n,Pmax) = v_n + f(n-1,Pmax-p_n)$: cette solution inclut l'objet n (dans le cas contraire, il n'y est pas),
  - On répète ce raisonnement pour tous les entiers i, de n-1 à 2, en repartant à chaque fois de la bonne valeur, $v_i + f(i-1,P-p_i)$ ou $f(i-1,P)$, en fonction des déductions précédentes (objet i inclus ou non)
  - Enfin, on inclut l'objet 1 si, à l'issue de toutes les déductions précédentes, la place résiduelle dans le sac est au moins p_1 (on exclut cet objet sinon)

Un 2ème algo. de programmation dynamique pour le sac à dos

- On peut aussi définir une autre fonction récursive :
  - g(i,V) = poids total min. d'un ensemble d'objets choisis parmi les i premiers, et de valeur $\geq$ V.
  - On a alors : g(i,V) = min(p_i + g(i-1,V-v_i), g(i-1,V)) (pour tout i>1 et V tel que $v_i \leq V$)
  - Il reste à calculer max{V/g(n,V)≤Pmax} (où $V \leq \Sigma_i v_i$)
  - L'algorithme de programmation dynamique obtenu a une complexité en $O(n \Sigma_i v_i) = O(n^2 \max_i v_i)$
  - Ici, l'algorithme est rapide si $\max_i v_i$ est « petit ».

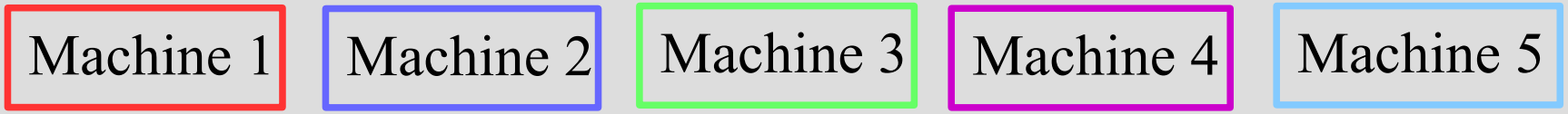
Un problème d'ordonnancement

- Soit à présent un problème d'ordonnancement de tâches (éléments de production) sur une machine
- Pour chaque tâche, on connaît ses dates de début et de fin d'exécution sur la machine
- Evidemment, deux tâches qui se « chevauchent » ne peuvent être exécutées sur la machine : elles sont **incompatibles** (elles sont compatibles sinon)
- A chaque tâche est associée un profit (financier ou non), et on souhaite donc choisir un ensemble de tâches compatibles de profit total maximum

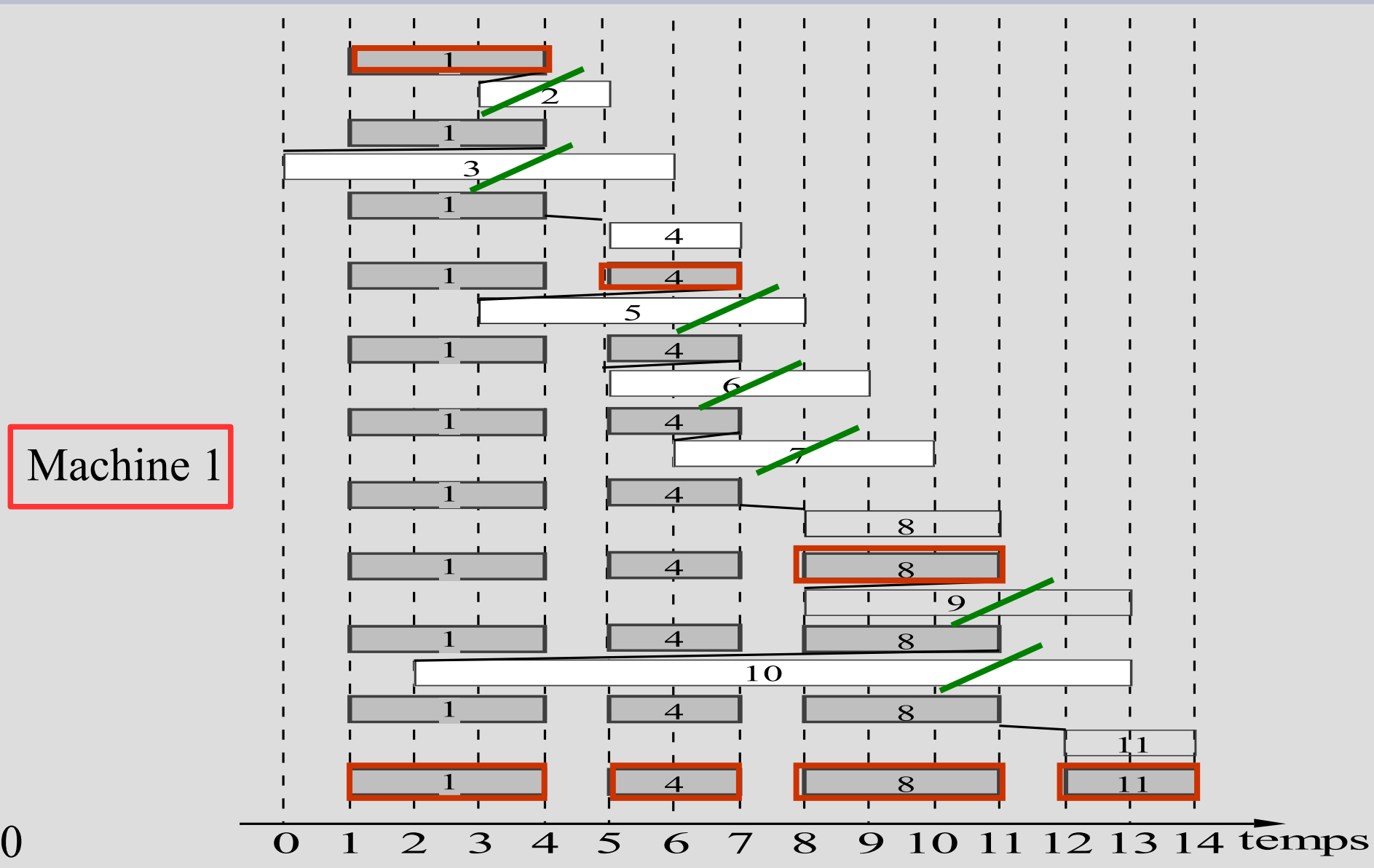
Un exemple d'affectation de tâches à des machines

- Exemple avec 11 tâches (d_i = début, f_i = fin) :

i	1	2	3	4	5	6	7	8	9	10	11
di	1	3	0	5	3	5	6	8	8	2	12
fi	4	5	6	7	8	9	10	11	12	13	14



Un exemple d'affectation de tâches à UNE machine



Comment résoudre ce problème d'ordonnancement ?

- Dans le cas où toutes les tâches ont le même profit, il existe un algorithme glouton simple
- Dans le cas général, une stratégie de résolution consiste à utiliser la programmation dynamique
- On note n le nombre total de tâches, p_i le profit de la ième tâche, pour i=1,...,n, et on numérote les tâches de 1 à n par ordre croissant des f_i
- Contrairement au sac à dos, on utilise un tableau à 1 seule dimension (fonction récursive à 1 variable)

Programmation dynamique pour l'exécution de tâches sur 1 machine

- On définit h(i) = profit total maximum que peut générer un ensemble de tâches compatibles, si la dernière choisie dans cet ensemble est la ième
- On exprime la valeur de h(i) en fonction des valeurs des h(j) pour j<i, à l'aide de l'équation de récurrence suivante (qui est valide) :

$$h(i) = p_i + \max_{j<i \text{ tels que fin de } j \leq \text{début de } i} h(j)$$

(S'il n'y a aucun j tel que $f_j \leq d_i$, alors par convention on aura : $\max_{j<i \text{ tels que fin de } j \leq \text{début de } i} h(j) = 0$.)

Complexité de l'algorithme

- La valeur optimale (initialisation) est alors obtenue en prenant le maximum sur tous les $i \leq n$ des h(i)
- Si le calcul des valeurs de h est implémenté à l'aide d'un tableau à 1 dimension (lors du calcul de h(i), les valeurs h(j) pour j<i sont déjà connues), on obtient les complexités suivantes :
  - Il y a O(n) valeurs h(i) à calculer,
  - Le calcul de chaque h(i) se fait en temps O(n),
  - La complexité globale est donc $O(n^2)$, et ne dépend donc que du nombre de tâches (polynomialement).

Bilan sur la programmation dynamique

- Méthode précieuse de conception d'algorithmes
- Implémentation : récursivité vs tableaux
- De nombreux algorithmes, pour des problèmes très variés, peuvent être conçus ainsi :
  - Un algorithme en $O(n\,2^n)$ pour le problème de tournée appelé VC (Voyageur de Commerce),
  - Un algorithme efficace pour le problème de base en dimensionnement de lots,
  - Etc.