Algorithmique avancée
TD Programmation Dynamique

Laurent Lemarchand
Lab-STICC/UBO
Laurent.Lemarchand@univ-brest.fr

TD programmation dynamique
gestion de stock

- Achat et vente de jetskis – stock maximum de 5 jetskis

mois	mars-avril	mai-juin	juillet-aout	sept-oct	nov-déc
ventes	2	3	4	3	2
Prix HT	1.3k€	1.5k€	2.0k€	1.1k€	1.2k€

  - Achats du magasin en début de période
  - Ventes uniquement des unités en stock en début de mois
  - Stock nul au départ et en fin de saison (pas d'entrepot en nov-déc)
- Combien acheter de jetskis au début de chaque période ?

TD programmation dynamique
gestion de stock

- Achat et vente de jetskis – stock maximum de 5 jetskis

mois	mars-avril	mai-juin	juillet-aout	sept-oct	nov-déc
ventes	2	3	4	3	2
Prix HT	1.3k€	1.5k€	2.0k€	1.1k€	1.2k€

  - Achats du magasin en début de période
  - Ventes uniquement des unités en stock en début de mois
  - Stock nul au départ et en fin de saison (pas d'entrepot en jan-fév)
- Combien acheter de jetskis au début de chaque période ?
  - Variables x_i : achats de la période i
    → variables de décision
  - Variables a_i : quantité en stock à la fin de la période i
    → variables d'état du système

TD programmation dynamique
gestion de stock

- Combien peut on acheter de jetskis au début de la première période ? (valeurs possibles de x_1) – quel est le cout associé ?
- Combien de jetskis en stock à la fin de la première période ? (valeurs possibles de a_1) – quel est la relation avec d_1 ?

mois	mars-avril	mai-juin	juillet-aout	sept-oct	nov-déc
ventes	2	3	4	3	2
Prix HT	1.3k€	1.5k€	2.0k€	1.1k€	1.2k€

- Variables x_i : achats de la période i
  → variables de décision
- Variables a_i : quantité en stock à la fin de la période i
  → variables d'état du système

TD programmation dynamique
gestion de stock

- Généraliser pour chaque période
  - La relation entre x_i et a_i
  - La relation entre d_i et a_i
  - La relation entre (x_i, a_i, d_i) et $(x_{i+1}, a_{i+1}, d_{i+1})$
  - Quelle est la fonction de cout du problème
    → donner le programme linéaire (IP) associé

  - Variables x_i : achats de la période i
    → variables de décision
  - Variables a_i : quantité en stock à la fin de la période i
    → variables d'état du système

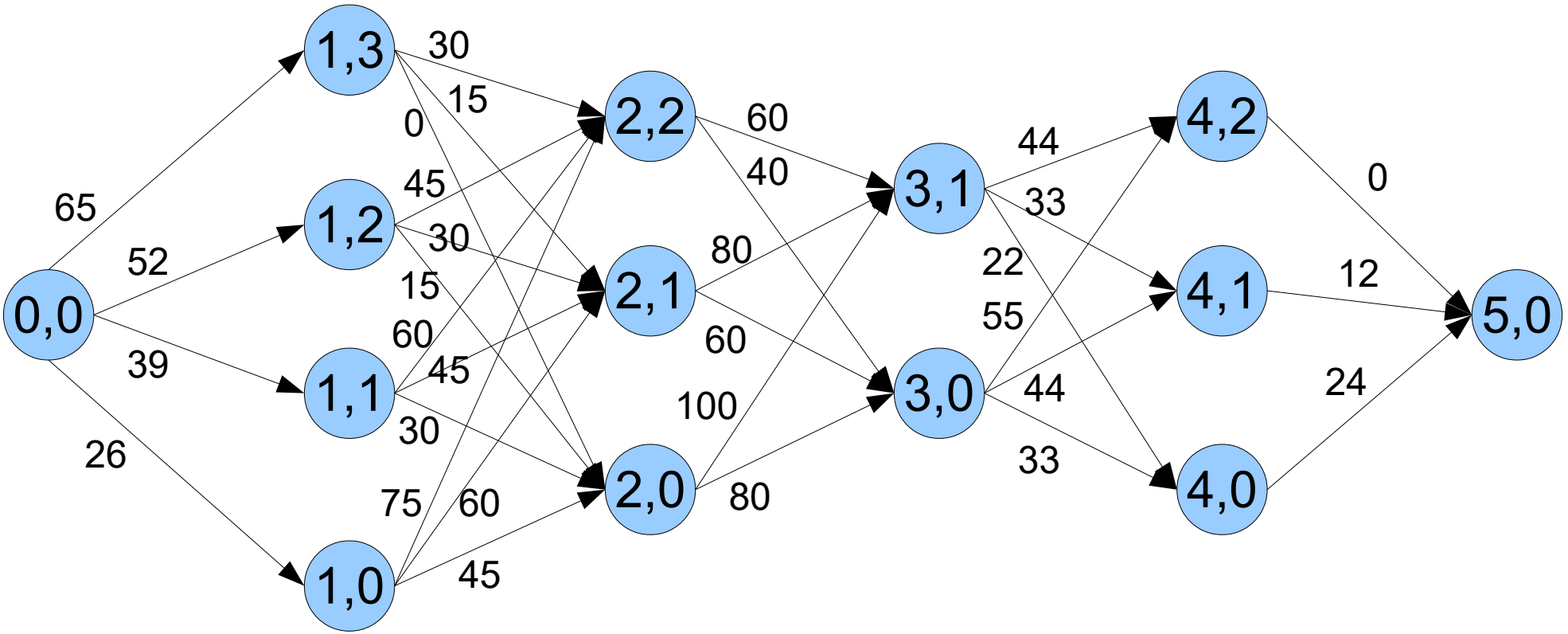
TD programmation dynamique
gestion de stock

- Pour la résolution
  - Comment est défini un état du système ?
  - Comment est défini le cout d'un passage d'un état à un autre ?
  - Construire le graphe d'état complet

  - Comment avoir le cout optimal pour un état donné ?
  - Solution = cout optimal de l'état final
  - Appliquer l'algorithme de Bellman pour l'obtenir

TD programmation dynamique
gestion de stock

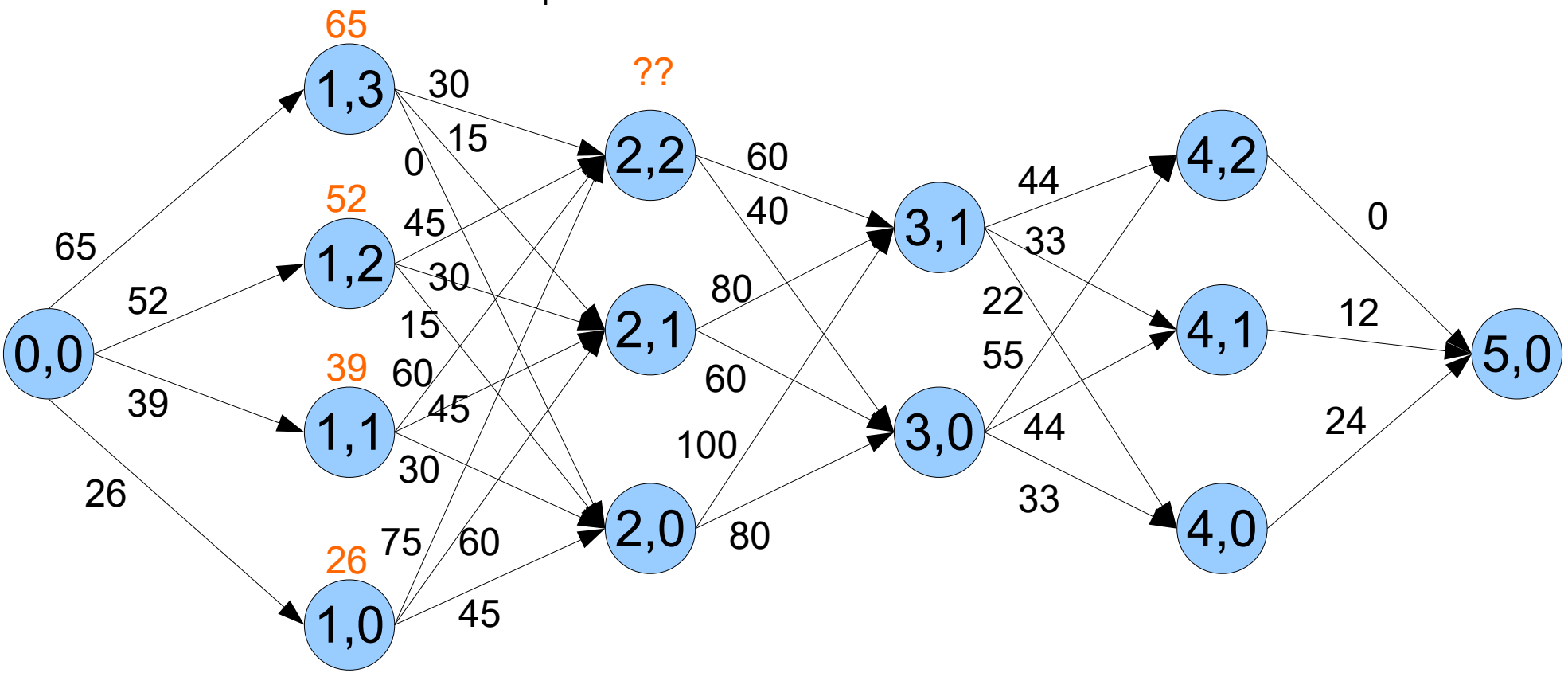
- Sommets états (i, a_i)



- $D(k, a_k) = \min_{\{ak-1\}} \{ D(k-1, a_{k-1}) + p_k(a_k - a_{k-1} + d_k) \}$

TD programmation dynamique
gestion de stock

- Sommets états (i, a_i)



- $D(k, a_k) = \min_{\{ak-1\}} \{ D(k-1, a_{k-1}) + p_k(a_k - a_{k-1} + d_k) \}$

TD Branch and Bound
sac à dos

- Définition **KP**

0-1 Knapsack problem

The *0-1*, or *Binary*, *Knapsack Problem* (KP) is: given a set of n *items* and a *knapsack*, with

p_j = *profit* of item j,

w_j = *weight* of item j,

c = *capacity* of the knapsack,

données

select a subset of the items so as to

IP

$$\text{maximize} \quad z = \sum_{j=1}^{n} p_j x_j \tag{2.1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} w_j x_j \leq c, \tag{2.2}$$

$$x_j = 0 \text{ or } 1, \qquad j \in N = \{1, \ldots, n\}, \tag{2.3}$$

where

$$x_j = \begin{cases} 1 & \text{if item } j \text{ is selected;} \\ 0 & \text{otherwise.} \end{cases}$$

variables

TD Branch and Bound
sac à dos

- **Relaxation C(KP)** :

  solution dans [0..1] ($\mathbb{R}$)

  au lieu de {0,1} ($\mathbb{N}$)

  $z^*(\mathbb{R}) \geq z^*(\mathbb{N})$ car $\mathbb{N} \subset \mathbb{R}$

$$\text{maximize} \quad \sum_{j=1}^{n} p_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} w_j x_j \leq c,$$

$$0 \leq x_j \leq 1, \qquad j = 1, \ldots, n.$$

  → meilleure borne supérieure que $z^*(\mathbb{R})$

  → résolution directe possible sans passer par la PL

TD Branch and Bound
sac à dos

- Résolution de **C(KP)**

tri des items par profit/kg

Suppose that the items, ordered according to (2.7), are consecutively inserted into the knapsack until the first item, s, is found which does not fit. We call it the *critical item*, i.e.

$$p_j / w_j >= p_{j+1} / w_{j+1} \qquad (2.7)$$

$$s = \min \left\{ j : \sum_{i=1}^{j} w_i > c \right\}, \qquad (2.8)$$

and note that, because of assumptions (2.4)–(2.6), we have $1 < s \leq n$. Then $C(KP)$ can be solved through a property established by Dantzig (1957), which can be formally stated as follows.

Theorem 2.1 *The optimal solution $\bar{x}$ of $C(KP)$ is*

solution optimale

$$\bar{x}_j = 1 \qquad \text{for } j = 1, \ldots, s-1,$$

$$\bar{x}_j = 0 \qquad \text{for } j = s+1, \ldots, n,$$

$$\bar{x}_s = \frac{\bar{c}}{w_s},$$

where

$$\bar{c} = c - \sum_{j=1}^{s-1} w_j. \qquad (2.9)$$

TD Branch and Bound
sac à dos

Borne supérieure $\mathbf{U_1}$

valeur de solution optimale pour **C(KP)**

borne supérieure pour **KP**

$z^*(KP) \leq \mathbf{U_1}$

Given a sufficiently small $\varepsilon > 0$, we could increase the value of x_k^* by ε and decrease that of x_q^* by $\varepsilon w_k / w_q$, thus augmenting the value of the objective function of $\varepsilon(p_k - p_q w_k / w_q)$ (> 0, since $p_k / w_k > p_q / w_q$), which is a contradiction. In the same way we can prove that $x_k^* > 0$ for $k > s$ is impossible. Hence $\bar{x}_s = \bar{c}/w_s$ from maximality. □

The optimal solution value of $C(KP)$ follows:

$$z(C(KP)) = \sum_{j=1}^{s-1} p_j + \bar{c}\frac{p_s}{w_s}.$$

Because of the integrality of p_j and x_j, a valid upper bound on $z(KP)$ is thus

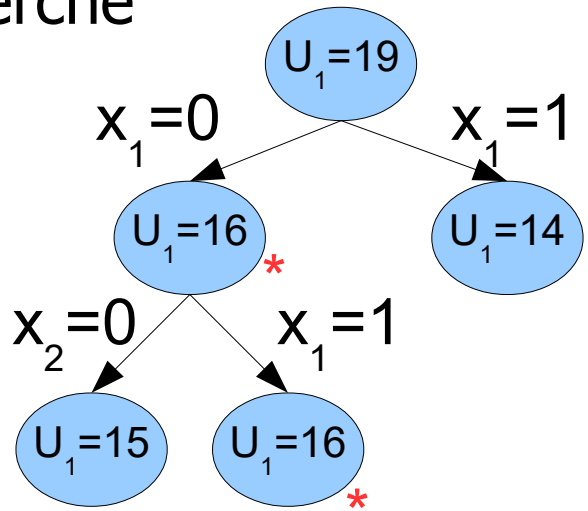
$$U_1 = \lfloor z(C(KP)) \rfloor = \sum_{j=1}^{s-1} p_j + \left\lfloor \bar{c}\frac{p_s}{w_s} \right\rfloor, \qquad (2.10)$$

where $\lfloor a \rfloor$ denotes the largest integer not greater than a.

TD Branch and Bound
sac à dos

Pour un Branch & Bound sur KP

- Borne sup U_1
- Règle de séparation et parcours de l'arbre de recherche



2.5 BRANCH-AND-BOUND ALGORITHMS

The first branch-and-bound approach to the exact solution of KP was presented by Kolesar (1967). His algorithm consists of a *highest-first* binary branching scheme which: (a) at each node, selects the not-yet-fixed item j having the maximum profit per unit weight, and generates two descendent nodes by fixing x_j, respectively, to 1 and 0; (b) continues the search from the feasible node for which the value of upper bound U_1 is a maximum.

*

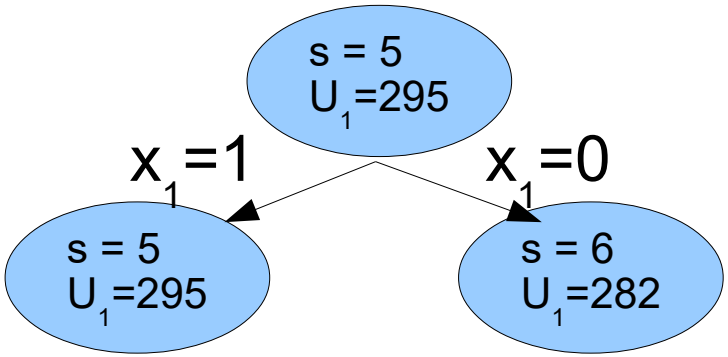
$p_1/w_1 \geq p_2/w_2 \geq \ldots$

TD Branch and Bound
sac à dos

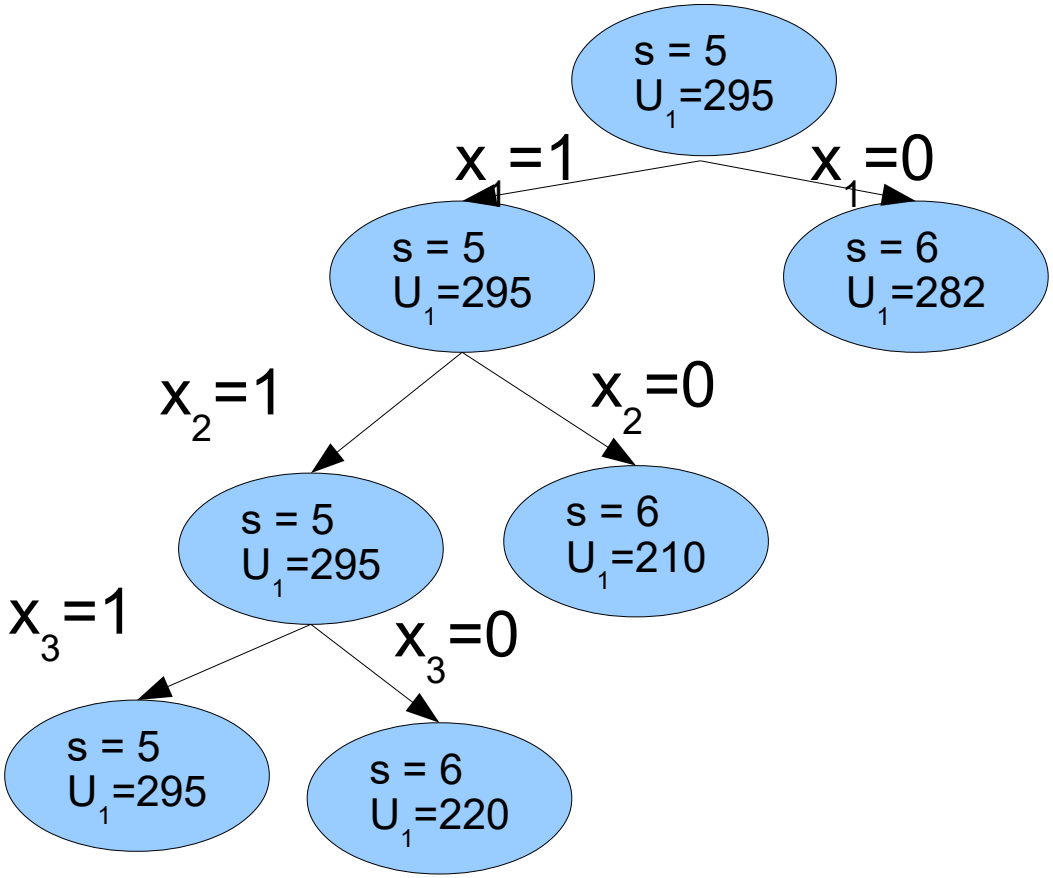
Appliquer au problème suivant, avec n=8 et c=102:

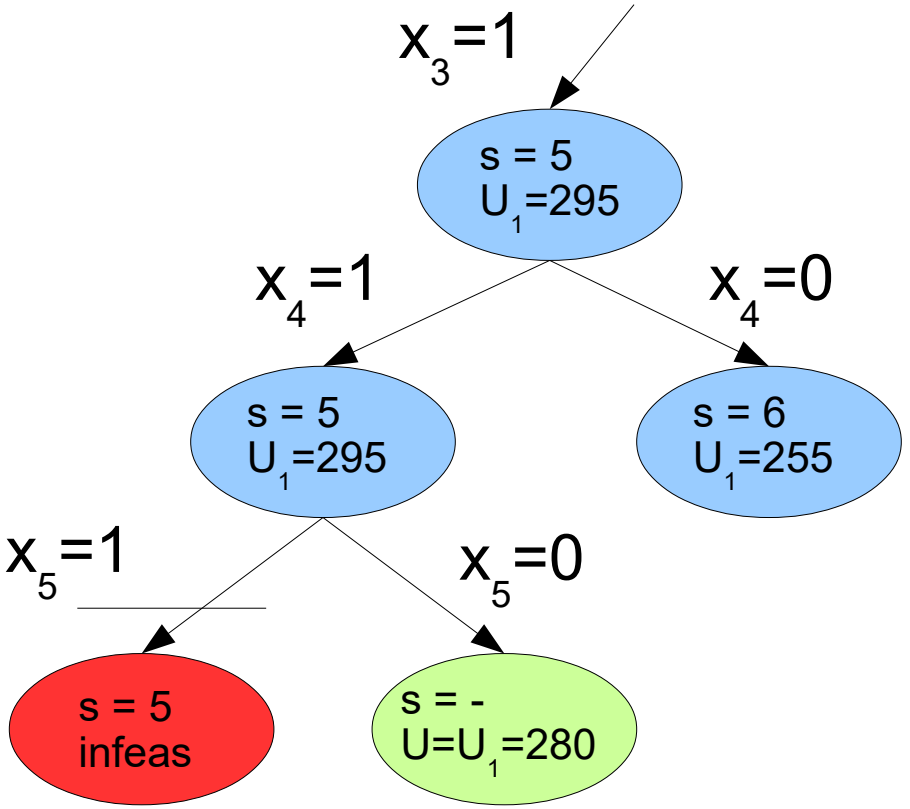
- Poids : 2 20 20 30 40 30 60 10
- Profits : 15 100 90 60 40 15 10 1

- Triés par profit unitaire décroissant : 7.5, 5, 4.5, 2, 1, 0.5, 0.17, 0.10



TD Branch and Bound
sac à dos





(1 1 1 1 0 1)
réalisable

TD Branch and Bound
sac à dos