Algorithmique et Optimisation Discrète

Séance V – Résolution de problèmes d'optimisation discrète par Branch and Bound

---

Équipe pédagogique AOD

Ensimag 2ème

Branch and Bound : principe

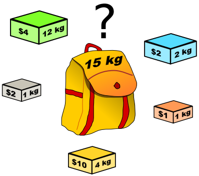
Méthode de résoluton d'un pb d'optimisation par **exploration** :

- Initialement on part d'une solution connue (non optimale a priori)
  $\hookrightarrow$ maj chaque fois qu'on découvre une solution meilleure
- **Branch** : exploration top-down (arborescente) de l'espace des solutions
  $\hookrightarrow$ découper un problème en sous-problèmes plus petits
- **Bound** : inutile d'explorer un sous-arbre qui ne peut pas améliorer la solution courante
- Par rapport à la programmation dynamique :
  - B&B plus général (pas d'optimalité des sous problemes)
  - B&B souvent (très) coûteux si on le laisse tourner...
    mais B&B *anytime* : on peut l'arrêtre quand on veut !
  - Gestion mémoire critique : compromis temps / mémoire-localité (cache)

Branch and Bound pour la PLNE

- **Idée** : Exploiter la solution optimale de la relaxation linéaire $x^* = \text{argmin}\{cx : Ax \leq b\}$, elle n'est peut être pas toujours si mauvaise...
- Si, par chance, toutes les variables sont entières, alors x^* est optimale pour notre problème en nombres entiers
- Sinon, il existe un i tel que $x_i \notin \mathbb{Z}$. On peut décomposer le problème en deux problèmes plus contraints : $\min\{cx : Ax \leq b, x_i \leq \lfloor x_i^* \rfloor\}$ et $\min\{cx : Ax \leq b, x_i \geq \lfloor x_i^* \rfloor + 1\}$
- En résolvant successivement et en décomposant si besoin les sous-problèmes, on construit un arbre d'énumération « intelligent » des solutions du problème.

Exemple : sac à dos multiple

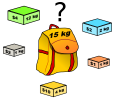


Considérons l'instance sac à dos multiple [1] suivante :

$$
\begin{aligned}
\text{maximiser} \quad & 9x_1 + 2x_2 + 6x_3 + 3x_4 & \\
\text{sujet à} \quad & 7x_1 + 2x_2 + 5x_3 + 4x_4 \leq 10 & (1) \\
& x_1, \ldots, x_4 \in \mathbb{N} & (2)
\end{aligned}
$$

1. *i.e.* on peut utiliser plusieurs fois le même objet

Sac à dos multiple : relaxation linéaire (PL)

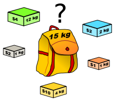
- 

$$
\begin{array}{rll}
\text{Maximiser} & 9x_1 + 2x_2 + 6x_3 + 3x_4 & \\
\text{sujet à} & 7x_1 + 2x_2 + 5x_3 + 4x_4 \leq 10 & (1) \\
 & x_1, \dots, x_4 \geq 0 & (2)
\end{array}
$$

-

objet i	1	2	3	4
utilité u_i	9	2	6	3
volume v_i	7	2	5	4
gain volumique ρ_i	$\frac{9}{7} \simeq 1,28$	$\frac{2}{2} = 1$	$\frac{6}{5} = 1,2$	$\frac{3}{4} = 0,75$

Sac à dos multiple : relaxation linéaire (PL)

- 

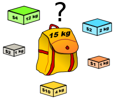
$$\begin{array}{rll} \text{Maximiser} & 9x_1 + 2x_2 + 6x_3 + 3x_4 & \\ \text{sujet à} & 7x_1 + 2x_2 + 5x_3 + 4x_4 \leq 10 & (1) \\ & x_1, \ldots, x_4 \geq 0 & (2) \end{array}$$

-

objet i	1	2	3	4
utilité u_i	9	2	6	3
volume v_i	7	2	5	4
gain volumique ρ_i	$\frac{9}{7} \simeq 1,28$	$\frac{2}{2} = 1$	$\frac{6}{5} = 1,2$	$\frac{3}{4} = 0,75$

- Solution optimale du PL : $\frac{10}{7}$ de l'objet 1 (irréalisable) d'utilité $\frac{90}{7} \simeq 12,8$

Sac à dos multiple : relaxation linéaire (PL)

- 

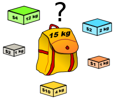
$$
\begin{array}{rll}
\text{Maximiser} & 9x_1 + 2x_2 + 6x_3 + 3x_4 & \\
\text{sujet à} & 7x_1 + 2x_2 + 5x_3 + 4x_4 \leq 10 & (1) \\
 & x_1, \ldots, x_4 \geq 0 & (2)
\end{array}
$$

-

objet i	1	2	3	4
utilité u_i	9	2	6	3
volume v_i	7	2	5	4
gain volumique ρ_i	$\frac{9}{7} \simeq 1,28$	$\frac{2}{2} = 1$	$\frac{6}{5} = 1,2$	$\frac{3}{4} = 0,75$

- Solution optimale du PL : $\frac{10}{7}$ de l'objet 1 (irréalisable) d'utilité $\frac{90}{7} \simeq 12,8$
- Solution glouton (Ali Baba) : [1, 2] d'utilité 11
  Donc à au plus 1 de l'optimal du PLNE !

Sac à dos multiple : relaxation linéaire (PL)

- 

$$
\begin{array}{rll}
\text{Maximiser} & 9x_1 + 2x_2 + 6x_3 + 3x_4 & \\
\text{sujet à} & 7x_1 + 2x_2 + 5x_3 + 4x_4 \leq 10 & (1) \\
 & x_1, \ldots, x_4 \geq 0 & (2)
\end{array}
$$

-

objet i	1	2	3	4
utilité u_i	9	2	6	3
volume v_i	7	2	5	4
gain volumique ρ_i	$\frac{9}{7} \simeq 1,28$	$\frac{2}{2} = 1$	$\frac{6}{5} = 1,2$	$\frac{3}{4} = 0,75$

- Solution optimale du PL : $\frac{10}{7}$ de l'objet 1 (irréalisable) d'utilité $\frac{90}{7} \simeq 12,8$
- Solution glouton (Ali Baba) : [1, 2] d'utilité 11
  Donc à au plus 1 de l'optimal du PLNE !
- exploration par Branch&bound pour trouver l'optimal...

Sac à dos multiple : relaxation linéaire (dual)

Relaxation linéaire (P) :

$$\begin{aligned}
\text{maximiser} \quad & 9x_1 + 2x_2 + 6x_3 + 3x_4 & \\
\text{sujet à} \quad & 7x_1 + 2x_2 + 5x_3 + 4x_4 \leq 10 & (1) \\
& x_1, \ldots, x_4 \geq 0 & (2)
\end{aligned}$$

Sac à dos multiple : relaxation linéaire (dual)

Relaxation linéaire (P) :

$$
\begin{array}{rll}
\text{maximiser} & 9x_1 + 2x_2 + 6x_3 + 3x_4 & \\
\text{sujet à} & 7x_1 + 2x_2 + 5x_3 + 4x_4 \leq 10 & (1) \\
 & x_1, \ldots, x_4 \geq 0 & (2)
\end{array}
$$

Dual de (P) :

$$
\begin{array}{rll}
\text{minimiser} & 10y & \\
\text{sujet à} & 7y \geq 9 & (1) \\
 & 2y \geq 2 & (2) \\
 & 5y \geq 6 & (3) \\
 & 4y \geq 3 & (4) \\
 & y \geq 0 & (5)
\end{array}
$$

Sac à dos multiple : relaxation linéaire (dual)

Relaxation linéaire (P) :

$$\begin{array}{rll} \text{maximiser} & 9x_1 + 2x_2 + 6x_3 + 3x_4 & \\ \text{sujet à} & 7x_1 + 2x_2 + 5x_3 + 4x_4 \leq 10 & (1) \\ & x_1, \ldots, x_4 \geq 0 & (2) \end{array}$$

Dual de (P) :

$$\begin{array}{rll} \text{minimiser} & 10y & \\ \text{sujet à} & 7y \geq 9 & (1) \\ & 2y \geq 2 & (2) \\ & 5y \geq 6 & (3) \\ & 4y \geq 3 & (4) \\ & y \geq 0 & (5) \end{array}$$

- Solution optimale du dual : $y = \max\{\frac{9}{7}, \frac{2}{2}, \frac{6}{5}, \frac{3}{4}, 0\} = \frac{9}{7}$ valeur $\frac{90}{7}$

Sac à dos multiple : relaxation linéaire (dual)

Relaxation linéaire (P) :

$$\begin{array}{rll} \text{maximiser} & 9x_1 + 2x_2 + 6x_3 + 3x_4 & \\ \text{sujet à} & 7x_1 + 2x_2 + 5x_3 + 4x_4 \leq 10 & (1) \\ & x_1, \ldots, x_4 \geq 0 & (2) \end{array}$$

Dual de (P) :

$$\begin{array}{rll} \text{minimiser} & 10y & \\ \text{sujet à} & 7y \geq 9 & (1) \\ & 2y \geq 2 & (2) \\ & 5y \geq 6 & (3) \\ & 4y \geq 3 & (4) \\ & y \geq 0 & (5) \end{array}$$

- Solution optimale du dual : $y = \max\{\frac{9}{7}, \frac{2}{2}, \frac{6}{5}, \frac{3}{4}, 0\} = \frac{9}{7}$ valeur $\frac{90}{7}$
- Solution optimale du primal : $x_1 = \frac{10}{7}, x_2 = x_3 = x_4 = 0$

Sac à dos : borne PL sur une solution optimale

- Solution optimale : $x_1 = \frac{10}{7}, x_2 = x_3 = x_4 = 0$
- Solution obtenue en remplissant de manière gloutonne avec l'objet de ratio bénéfice / volume le plus important
- **Remarque :** Sac à dos binaire[2] $\rightsquigarrow$ choisir par ordre décroissant de poids volumétrique jusqu'à remplir complètement le sac (seul le dernier objet est choisi de manière fractionnaire).

2. *i.e.* un seul exemplaire de chaque objet

Sac à dos binaire par Branch and Bound

Comment résoudre l'instance de sac à dos **binaire** suivante ?

$$
\begin{aligned}
\text{maximiser} \quad & 9x_1 + 2x_2 + 6x_3 + 3x_4 & \\
\text{sujet à} \quad & 7x_1 + 2x_2 + 5x_3 + 4x_4 \leq 10 & (1)\\
& x_1, \ldots, x_4 \in \{0, 1\} & (2)
\end{aligned}
$$

Sac à dos binaire par Branch and Bound

Comment résoudre l'instance de sac à dos **binaire** suivante ?

$$
\begin{aligned}
\text{maximiser} \quad & 9x_1 + 2x_2 + 6x_3 + 3x_4 & \\
\text{sujet à} \quad & 7x_1 + 2x_2 + 5x_3 + 4x_4 \leq 10 & (1)\\
& x_1, \ldots, x_4 \in \{0, 1\} & (2)
\end{aligned}
$$

Nous allons utiliser la méthode du **Branch and Bound** s'appuyant sur la **relaxation linéaire**.

Sac à dos binaire et Branch and Bound

$$
\begin{aligned}
\text{maximiser} \quad & 9x_1 + 2x_2 + 6x_3 + 3x_4 \\
\text{sujet à} \quad & 7x_1 + 2x_2 + 5x_3 + 4x_4 \leq 10 && (1) \\
& x_1, \ldots, x_4 \in \{0, 1\} && (2)
\end{aligned}
$$

$$
\begin{aligned}
\text{maximiser} \quad & 9x_1 + 2x_2 + 6x_3 + 3x_4 & \\
\text{sujet à} \quad & 7x_1 + 2x_2 + 5x_3 + 4x_4 \leq 10 & (1) \\
& x_1, \ldots, x_4 \in \{0, 1\} & (2)
\end{aligned}
$$

Solution optimale de la relaxation linéaire : $(1, 0, 3/5, 0)$, $z = 12.6$.

$$\begin{aligned} \text{maximiser} \quad & 9x_1 + 2x_2 + 6x_3 + 3x_4 & \\ \text{sujet à} \quad & 7x_1 + 2x_2 + 5x_3 + 4x_4 \leq 10 & (1) \\ & x_1, \ldots, x_4 \in \{0, 1\} & (2) \end{aligned}$$

Solution optimale de la relaxation linéaire : $(1, 0, 3/5, 0)$, $z = 12.6$.

x_3 non entier $\Rightarrow$ nous allons **brancher** sur $x_3 = 0$ ou $x_3 = 1$.

Sac à dos : arbre de parcours

$(1, 0, \frac{3}{5}, 0)$

$z = 12.6$

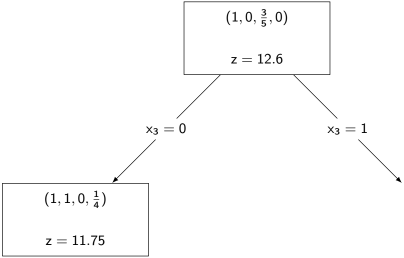
Sac à dos : arbre de parcours

$(1, 0, \frac{3}{5}, 0)$

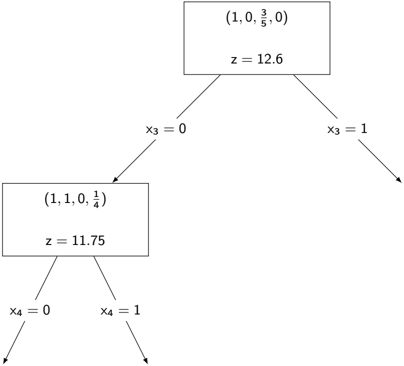
$z = 12.6$

$x_3 = 0$ $\quad$ $x_3 = 1$

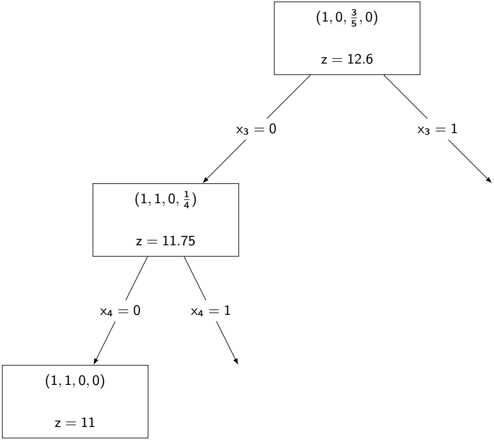
Sac à dos : arbre de parcours



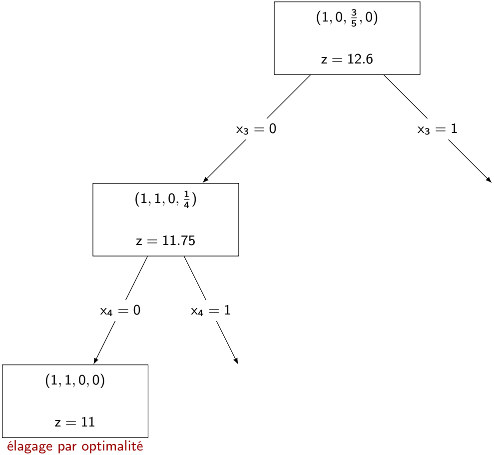
Sac à dos : arbre de parcours



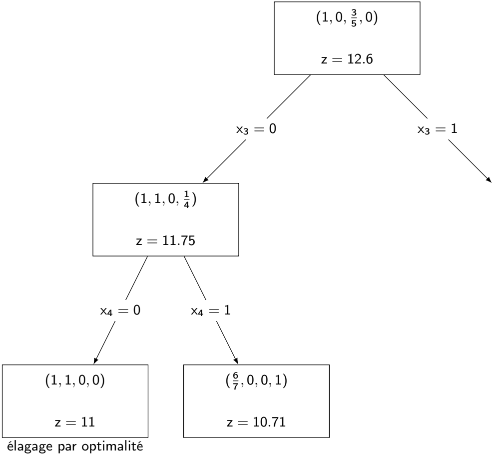
Sac à dos : arbre de parcours



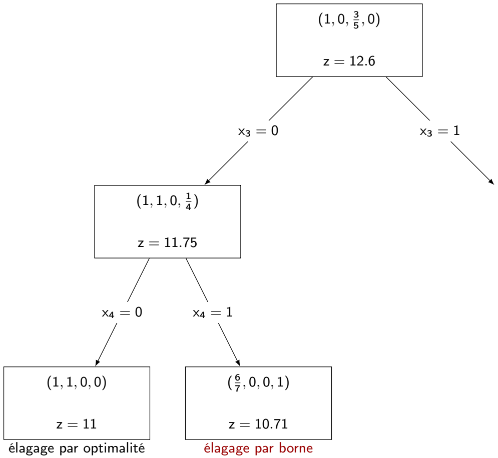
Sac à dos : arbre de parcours



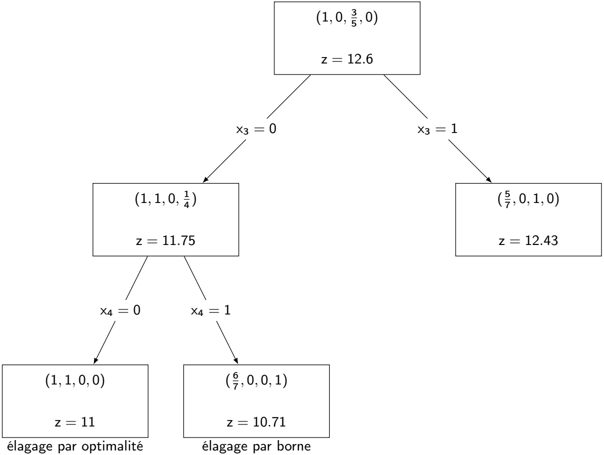
Sac à dos : arbre de parcours



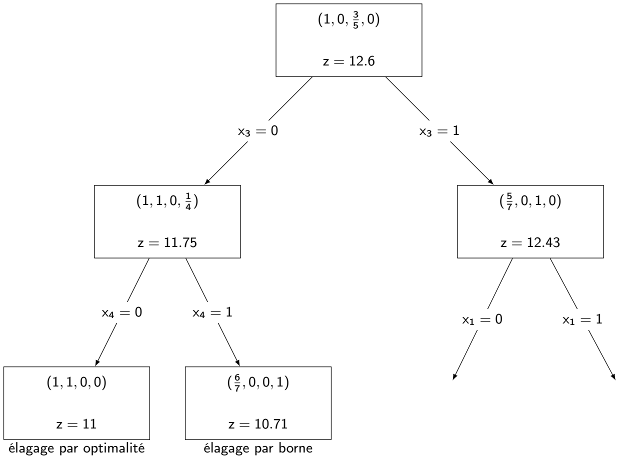
Sac à dos : arbre de parcours



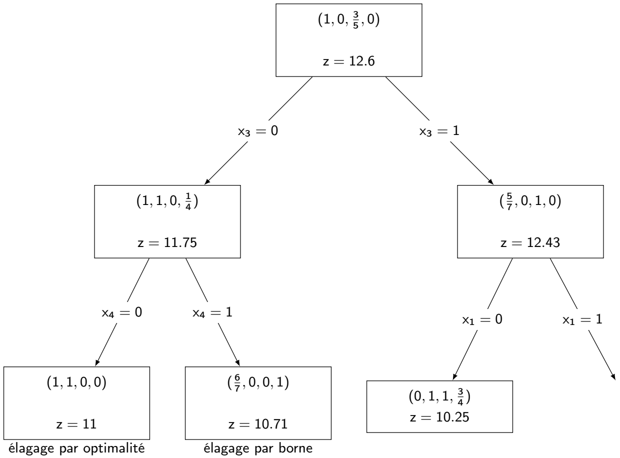
Sac à dos : arbre de parcours



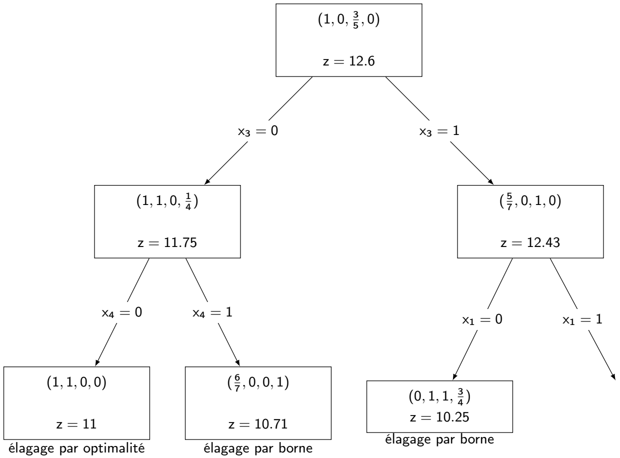
Sac à dos : arbre de parcours



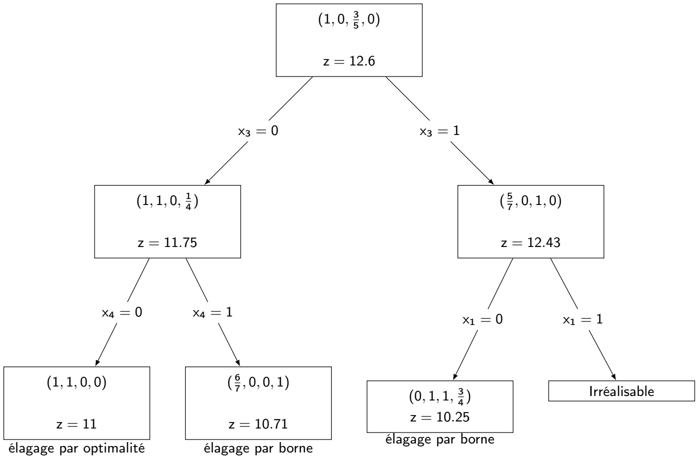
Sac à dos : arbre de parcours



Sac à dos : arbre de parcours



Sac à dos : arbre de parcours



Que retenir de cet exemple ?

- Il faut faire certains **choix** :
  - procédure de choix du noeud à explorer à définir (ex : meilleure borne)
  - procédure de « branchement » à définir (ex : la première variable fractionnaire [3])
- **Elagage** (*Bound*) : ici PL, mais d'autres choix possibles
- **Solution initiale** : si bonne solution (par heuristique par exemple), algorithme plus efficace

3. NB : il n'y en a qu'une à chaque fois ici

Que retenir de cet exemple ?

- Il faut faire certains **choix** :
  - procédure de choix du noeud à explorer à définir (ex : meilleure borne)
  - procédure de « branchement » à définir (ex : la première variable fractionnaire[3])
- **Elagage** (*Bound*) : ici PL, mais d'autres choix possibles
- **Solution initiale** : si bonne solution (par heuristique par exemple), algorithme plus efficace

Formalisons tout cela...

3. NB : il n'y en a qu'une à chaque fois ici

Arbre d'énumération (pour un max)

- Soit un problème d'optimisation discrète $\max_{S \in \mathcal{X}} f(S)$ avec $|\mathcal{X}|$ fini.
- **Idée (Branch) :** on **décompose** l'espace des solutions en sous-espaces plus petits :
  $\mathcal{X}_1 \cup \cdots \cup \mathcal{X}_K = \mathcal{X}$ avec $\emptyset \neq \mathcal{X}_k \subset \mathcal{X}$ pour tout $k = 1, \ldots, K$
  - Remarque : les $\mathcal{X}_k$ sont en général disjoints).
- On a en particulier $z = \max_{k=1,\ldots,K} \min_{S \in X_k} f(S)$.
- On répète cete opération de décompositiion (Branch) sur les ensembles $\mathcal{X}_k$ tant que c'est possible (jusqu'à obtenir des singletons)
  $\hookrightarrow$ on obtient un arbre d'énumération des solutions.
- un sous-arbre d'énumération est appelé **b**ranche

Exemple sur le sac à dos

$$
\begin{aligned}
\text{maximiser} \quad & x_1 + x_2 + x_3 \\
\text{sujet à} \quad & x_1 + 2x_2 + 2x_3 \leq 3 && (1) \\
& x_1, x_2, x_3 \in \{0, 1\} && (2)
\end{aligned}
$$

Exemple sur le sac à dos

$$
\begin{array}{rll}
\text{maximiser} & x_1 + x_2 + x_3 & \\
\text{sujet à} & x_1 + 2x_2 + 2x_3 \leq 3 & (1) \\
& x_1, x_2, x_3 \in \{0, 1\} & (2)
\end{array}
$$

Solutions réalisables :

$$
\left\{ 0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, 1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, 2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, 4 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, 5 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, 6 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \right\}
$$

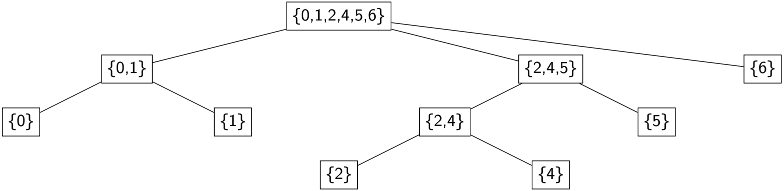
Arbre d'énumération

$$\left\{0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, 1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, 2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, 4 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, 5 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, 6 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}\right\}$$

Arbre d'énumération

$$\left\{0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, 1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, 2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, 4 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, 5 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, 6 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}\right\}$$

- Un exemple d'arbre d'énumération possible



- Il y en a évidemment beaucoup d'autres possibles : on pourrait par exemple brancher sur la valeur de x_1 puis sur x_2 puis sur x_3

Arbre d'énumération et Branch and Bound

Arbre d'énumération $\Longrightarrow$ stratégie de branchement. Par exemple :

- résoudre la relaxation linéaire (pour le sous-problème)
- si $\exists$ une variable non entière, brancher dessus
- sinon brancher sur la première variable non encore explorée

Arbre d'énumération et Branch and Bound

Arbre d'énumération $\Longrightarrow$ stratégie de branchement. Par exemple :

- résoudre la relaxation linéaire (pour le sous-problème)
- si $\exists$ une variable non entière, brancher dessus
- sinon brancher sur la première variable non encore explorée

Elagage (Bound) : si pas de solution **pertinente** dans le sous-arbre

Trois cas d'éalagage :

- par **optimalité** : on a déjà trouvé la solution optimale de la branche (*e.g.* le PL a une solution entière)
- par **réalisabilité** : aucune solution réalisable dans cette branche ($\mathcal{X}_k = \emptyset$)
- par **borne** (*bound*) : pas de meilleure solution dans cette branche
  - on calcule une **évaluation optimiste** du sous-arbre :
    - si max : majorant de toutes les valeurs de la bracnhe
    - si min : minorant de toutes les valeurs de la bracnhe
  - si cette évaluation optimiste est moins bonne que la meilleure soluion rencontrée jusqu'ici $\Longrightarrow$ élagage

B&B : algo générique ($\min f$)

Soit $\bar{z}$ la valeur d'une solution réalisable initiale;
Soit $\mathcal{E} = \{\mathcal{X}\}$;
Soit $g : \{\mathcal{P}(X)\} \mapsto \mathbb{R}$ une fonction d'évaluation optimiste (minorant) :
  $g(X) \leq \min_{s \in X} f(s)$;
répéter
  Choisir $P \in \mathcal{E}$ et le supprimer de $\mathcal{E}$: $\mathcal{E} = \mathcal{E} \setminus \{P\}$;
  Évaluer de manière optimiste : $\underline{z} = g(P) \leq \min_{s \in P} f(s)$;
  si on peut montrer $\underline{z} = \min_{s \in P} f(s)$ **alors**
    mettre à jour la solution courante $\bar{z}$ si besoin;
  sinon si $\underline{z} < \bar{z}$ **alors**
    Décomposer P en k sous-problèmes $P_1 \sqcup \cdots \sqcup P_k = P$;
    $\mathcal{E} = \mathcal{E} \cup \{P_1, \ldots, P_k\}$;
jusqu'à $\mathcal{E} = \emptyset$;

- Pour autoriser des branchements conduisant à des sous-problèmes irréalisables, on étend la définition de g et $\min_{s \in P} f(s)$ à $\mathbb{R} \cup \{+\infty\}$
- On doit instancier 3 éléments : choisir, évaluer, décomposer
- *NB : on ne garde pas la structure de l'arbre mais les « noeuds actifs » dans $\mathcal{E}$*

B&B : écart garanti + anytime $(\min f)$

- À tout instant :
  1. $\forall P$ dont aucun fils n'est dans $\mathcal{E}$, $\forall s \in P : f(s) \geq \bar{z}$;
  2. soit $\underline{m}(\mathcal{E}) = \min\left(\{g(P)\}_{P \in \mathcal{E}} \cup \{\bar{z}\}\right)$. Alors $\boxed{\underline{m}(\mathcal{E}) \leq \min_s f(s)}$
     - NB coût de calcul de $\underline{m}(\mathcal{E}) = \#\mathcal{E}$ appels de g (amorti !).
- D'où l'encadrement : $\boxed{\underline{m}(\mathcal{E}) \leq \min_s f(s) \leq f(\bar{s}) = \bar{z}}$.
  La valeur $\bar{z}$ de la solution courante $\bar{s}$ est à au plus $(\bar{z} - \underline{m}(\mathcal{E}))$ de l'optimum.
  $\hookrightarrow$ garantie d'approximation ! ! ! (***approximation ratio***)
  $\hookrightarrow$ à tout instant ! ! ! (***anytime***)
- En pratique : on peut arrêter l'algorithme n'importe quand
  ou dès que la solution z est à moins de 5% de l'optimum, par exemple.

B&B : Semi-instanciation de l'algo générique (MAX)

```
s := ... solution initiale... (par ex. heuristique) ;
z := f(s) ;
Collection <Nœud>E := { racine de l'arbre d'exploration } ;
répéter
    P := E.pop() ;
    si P.est_une_solution() et P.val() > z alors
        s := P; z := s.val() ;
    sinon si P.bound() > z alors
        P.branch(E) ;
jusqu'à E = ∅ ou |m̄(E) − z| < 5%;
```

- L'algorithme générique s'applique à **tout** problème d'optimisation discrète
- NB : ici, $\bar{m}$ est le max des nœuds restants à explorer

Exemple : sac à dos n-aire ($\max f$)

- Pré-tri des n objets par gain volumique décroissant (cf PLNE) :

$$\frac{c_1}{a_1} \geq \ldots \geq \frac{c_n}{a_n}$$

Exemple : sac à dos n-aire ($\max f$)

- Pré-tri des n objets par gain volumique décroissant (cf PLNE) :

$$\frac{c_1}{a_1} \geq \ldots \geq \frac{c_n}{a_n}$$

- $\mathcal{P}$ = type `Nœud` = représentation d'un sous-problème

```
class Nœud { int k ; int x[n] ; int c ; int r ; };
```

Exemple : sac à dos n-aire ($\max f$)

- Pré-tri des n objets par gain volumique décroissant (cf PLNE) :

$$\frac{c_1}{a_1} \geq \ldots \geq \frac{c_n}{a_n}$$

- $\mathcal{P}$ = type `Nœud` = représentation d'un sous-problème

```
class Nœud { int k ; int x[n] ; int c ; int r ; };
```

- Ici : branchement sur les variables $x_1, ..., x_n$ dans cet ordre

Exemple : sac à dos n-aire ($\max f$)

- Pré-tri des n objets par gain volumique décroissant (cf PLNE) :

$$\frac{c_1}{a_1} \geq \ldots \geq \frac{c_n}{a_n}$$

- $\mathcal{P}$ = type `Nœud` = représentation d'un sous-problème

```
class Nœud { int k ; int x[n] ; int c ; int r ; };
```

- Ici : branchement sur les variables $x_1, ..., x_n$ dans cet ordre
- Au niveau k, $x_1, .., x_k$ fixées avec gain c et volume libre restant r.

Exemple : sac à dos n-aire ($\max f$)

- Pré-tri des n objets par gain volumique décroissant (cf PLNE) :

$$\frac{c_1}{a_1} \geq \ldots \geq \frac{c_n}{a_n}$$

- $\mathcal{P}$ = type `Nœud` = représentation d'un sous-problème

```
class Nœud { int k ; int x[n] ; int c ; int r ; };
```

- Ici : branchement sur les variables $x_1, ..., x_n$ dans cet ordre
- Au niveau k, $x_1, .., x_k$ fixées avec gain c et volume libre restant r.
- L'attribut k représente le niveau.

Attention : $x_k = x[k-1]$

Exemple : sac à dos n-aire ($\max f$)

- **init** : { racine(s) de l'arbre d'exploration }, ici :

```
Collection <Nœud> E;
Nœud n=new Nœud(0);
n.c=0; n.r=b;
E.add(n);
```

- z= valeur de la meilleure solution réalisable s rencontrée
  eval : calcul de s par une heuristique et $z = f(s)$. Ici : Ali Baba (PLNE)

```
Nœud s  = new Nœud(n) ;
s.r = b; s.c = 0;
for (i=0; (i < n) && (s.r >= 0); ++i) {
  s.x[i] = Math.floor(s.r / a[i]);
  s.r -= s.x[i] * a[i];
  s.c += s.x[i] * c[i];
}
z = s.c;
```

- **branch(Collection E)** = branchement

```
for (int i = 0; i <= Math.floor(r / a[k]); i++) {
  Nœud n = new Nœud(k+1);
  for (int j = 0; j < k; j++) n.x[j] = x[j];
  n.x[k] = i ;
  n.r = r - a[k] * n.x[k]
  n.c = c + c[k] * n.x[k]
  E.add(n);
}
```

Exemple : sac à dos n-aire ($\max f$)

- **bound** = évaluation optimiste g :

```
if (k < n ) {
  return c + (c[k] * r / a[k]);
}
```

- **est_une_solution** = tester si on a fixé toutes les variables :

```
  return k == n ;
```

ou mieux

```
  return  (k==n) || (  r < min_a[k]  ) ;
```

avec $min\_a[k] = \min\{a[i], i = k..n-1\}$ (précalculé en $O(n)$) ;

- **val** = évaluer f :

```
  return c ;
```

L'arborescence critique

- Si la solution initiale est optimale $\Longrightarrow$ solution courante jamais mise à jour
- **Définition :** L'arborescence critique est l'ensemble des nœuds explorés lorsque la solution initiale est optimale.
- Propriétés :
  - En général de grande taille, elle sert à prouver l'optimalité
  - Elle est indépendante du choix effectué pour implémenter la collection E (car indépendante de l'ordre de parcours)
- En pratique : l'algorithme Branch&Bound peut être utilisé pour prouver l'optimalité (ou la distance à l'optimum) d'une solution initiale calculée par un algorihme meta-heuristique.

Branch and Bound : remarques

- La performance de l'algorithme de B&B dépend fortement :
  - de la qualité de la solution initiale
  - de la qualité de la décoposition en sous-problèmes (**branch**)
    $\hookrightarrow$ arborescence et ordre de parcours de E
  - de la qualité de l'évaluation optimiste g (**bound**) : relaxation
- Pour le parcours de l'arborescence, le choix de la collection (file, pile ou autre) a un impact fort sur la performance (temps et mémoire).
  cf exercice TD5...

Conclusion : ce qu'on a vu aujourd'hui

Branch&Bound :

- exploration de l'espace des solutions avec une collection
- **Branch** : extravction d'un noued et ajout de ses fils
  $\hookrightarrow$ mise à jour de la solution courante
- **Bound** : suppression d'un sous-arbre qui n'améliore pas la solution courante
  Pour la PLNE, la résolution rapide du PL donne l'évaluation optimiste !
- Par rapport à la programmation dynamique :
  - B&B plus général (pas d'optimalité des sous problemes)
  - B&B *anytime* avec approcimation garantie
  - Compromis temps / mémoire-localité (cache)
    File de priorité jusqu'à une taille maximale puis pile (LIFO) !
  - Arborescence critique

Conclusion cours AOD

- Importance de la localité des données $\hookrightarrow$ défauts de cache
  - l'ordre des boucles dépend des données (voire parcours par blocs)
  - Programme cache oblivious par "blocs" (récursif avec seuil)
- Résolution de problèmes d'optimisation discrets
  - Programmation dynamique : caractérisation récursive et memorisation itérative
    cache oblivious : programme récursif
  - Branch&Bound : exploration arborescente avec boucle sur ube collections
    impact de la mémoire : pile pour limite l'explosion et assurer la localité
- Le "style" de programmation d'un algorithme a un impact sur son coût mémoire !
  Cascade récursive -> itérative

Exemple du TSP : Relaxation combinatoire

- **Possibilité d'énumération :** On part du sommet 1 et on énumère toutes les possibilités de successeur dans le tour.

Exemple du TSP : Relaxation combinatoire

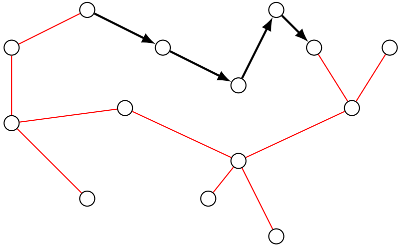
- **Possibilité d'énumération :** On part du sommet 1 et on énumère toutes les possibilités de successeur dans le tour.
- Borne de qualité ?

Exemple du TSP : Relaxation combinatoire

- **Possibilité d'énumération :** On part du sommet 1 et on énumère toutes les possibilités de successeur dans le tour.
- Borne de qualité ?
  - **Solution 1 :** Relaxation linéaire en fixant les arêtes sélectionnées à 1

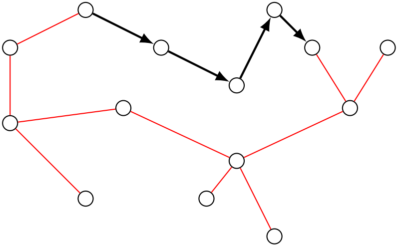
Exemple du TSP : Relaxation combinatoire

- **Possibilité d'énumération :** On part du sommet 1 et on énumère toutes les possibilités de successeur dans le tour.
- Borne de qualité ?
  - **Solution 1 :** Relaxation linéaire en fixant les arêtes sélectionnées à 1
  - **Solution 2 :** Coût de connexion par un arbre de tous les sommets restants aux deux extrémité de la sous-séquence.



Exemple du TSP : Relaxation combinatoire

- **Possibilité d'énumération :** On part du sommet 1 et on énumère toutes les possibilités de successeur dans le tour.
- Borne de qualité ?
  - **Solution 1 :** Relaxation linéaire en fixant les arêtes sélectionnées à 1
  - **Solution 2 :** Coût de connexion par un arbre de tous les sommets restants aux deux extrémité de la sous-séquence.



- NB : Nombreuses heuristiques pour initialiser (glouton + k-opt, ...)