Les méthodes de recherche arborescente par séparation et évaluation

16 septembre 2020

Plan

Généralités et Définitions

Définition de l'arborescence : concept de séparation

Concept d'évaluation

Mise en oeuvre pratique : Application à la programmation linéaire en nombres entiers

Sommaire

Généralités et Définitions

Définition de l'arborescence : concept de séparation

Concept d'évaluation

Mise en oeuvre pratique : Application à la programmation linéaire en nombres entiers

Généralités

- Land et Doig (1960), Bertier et Roy (1964)
- Problème d'optimisation combinatoire (P) :
  Etant donné un ensemble fini S et une application $f : S \to \mathbb{R}$ déterminer $s^* \in S$ tel que

$$(P) \qquad f(s^*) = \max_{s \in S} f(s)$$

- On note $v(P)(= f(s^*))$, la valeur de (P)

- Idée la plus simple : Énumération exhaustive des éléments de S (Énumération finie)
- Si $|S|$ grand (ex : $|S| = 2^n$ avec $n \geq 60$) $\Rightarrow$ temps de calcul prohibitif
- $n = 60$ et examen d'un élément en $10^{-9}s \Rightarrow$ calcul dure plus de 30 ans

- On cherche à résoudre (P) via une énumération *implicite* des éléments de S
- Principe :
  1. Séparation de S en sous-ensembles
  2. Évaluation des sous-ensembles de S
  3. Utilisation d'une représentation de S sous forme d'une arborescence.

Sommaire

Généralités et Définitions

Définition de l'arborescence : concept de séparation

Concept d'évaluation

Mise en oeuvre pratique : Application à la programmation linéaire en nombres entiers

Concept de séparation

- Sommets de l'arborescence correspondent à des sous-ensembles de S

Définition 1

Un sous-ensemble S_j de S est dit SÉPARÉ en les sous-ensembles $S_{j_1}, S_{j_2}, \ldots, S_{j_{k_j}}$ si

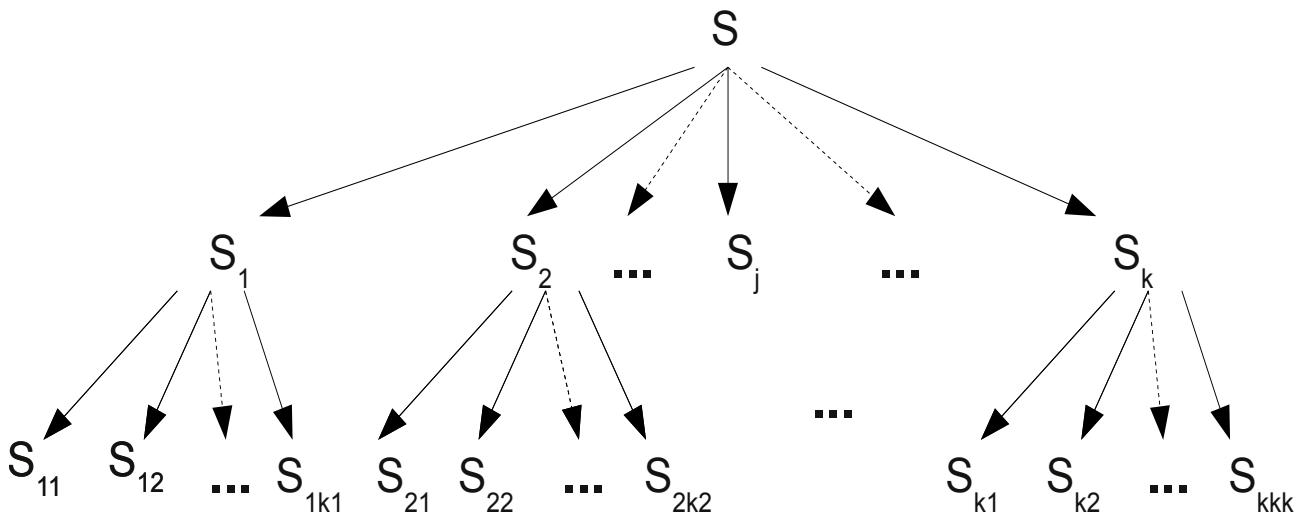
- $S_{j_i} \subset S_j \quad \forall i \in \{1, 2, \ldots, k_j\}$
- $S_{j_1} \cup S_{j_2} \cup \ldots \cup S_{j_k} = S_j$

NB :

- *Les S_{j_i} sont en général disjoints deux à deux*
- *La collection de sous-ensembles $S_{j_1}, S_{j_2}, \ldots, S_{j_k}$ est finie*
- *Certains S_{j_i} peuvent être vides*

Définition de l'arborescence

- Un sommet S_j est relié à chacun des S_{j_i} par un arc (S_j, S_{j_i})
- Racine de l'arborescence : S
- Propriété de l'arborescence : s'il y a un chemin de S_j à S_i alors $S_i \subset S_j$



Exploration de l'arborescence

- Un sommet S_j est abandonné s'il n'y a pas d'intérêt à le séparer ; il est actif dans le cas contraire
- un sommet abandonné est appelé feuille, sommet pendant ou sommet terminal
- L'exploration de l'arborescence débute à partir d'un sommet actif. On cherche alors à le séparer ou à l'abandonner. Si le sommet traité est abandonné, on retourne (*un backtracking*) à son unique prédécesseur dans l'arborescence et l'on continue jusqu'à trouver un sommet ayant au moins un successeur actif ; lorsqu'il n'y a plus de sommets actifs, l'énumération des solutions est terminée.

Sommaire

Généralités et Définitions

Définition de l'arborescence : concept de séparation

Concept d'évaluation

Mise en oeuvre pratique : Application à la programmation linéaire en nombres entiers

Évaluation des sommets de l'arborescence (1/3)

- Comment déterminer s'il faut séparer ou abandonner un sommet associé à un sous-ensemble S_j de solutions de S ?
  $\Rightarrow$ concept d'évaluation d'un sous-ensemble S_j à l'aide de bornes.
- On suppose que pour chaque sous-ensemble S_j de S on puisse déterminer , par calcul, une évaluation par excès, soit un majorant $\overline{v}(P_j)$ de $v(P_j)$, la valeur du sous-problème (P_j) engendré par S_j :

$$(P_j) \qquad v(P_j) = \max_{s \in S_j} f(s)$$

Évaluation des sommets de l'arborescence (2/3)

Définition 2

- *Étant donné (P_j), EVALUER un sous-ensemble S_j de S revient à savoir déterminer un majorant $\overline{v}(P_j)$ tel que*

$$\overline{v}(P_j) \geq f(s) \quad \forall s \in S_j$$

NB : Une évaluation $\overline{v}(P_j)$ est dite exacte s'il existe $s_j{}^* \in S_j$ tel que $\overline{v}(P_j) = f(s_j{}^*)$. Dans ce cas, $s_j{}^*$ est une solution optimale de (P_j).

Évaluation des sommets de l'arborescence (3/3)

Soit $\hat{s}$ une solution de S de valeur $f(\hat{s})$. Elle fournit un minorant de $v(P)$, notée $\underline{v}(P)$ (*éventuellement, si on n'en a pas trouvé, on conviendra de poser* $\underline{v}(P) = -\infty$)

Définition 3

- *Un sommet* S_j *est abandonné si*
  1. S_j *est vide*
  2. $\overline{v}(P_j)$ *est une évaluation exacte* $\Rightarrow$ *on a obtenu une solution optimale de* (P_j)
  3. $\overline{v}(P_j) \leq \underline{v}(P) \Rightarrow$ *dans l'ensemble de solutions* S_j*, il ne peut exister de solution meilleure que* $\hat{s}$

Rq : La notion d'évaluation permet l'élagage de l'arborescence et ainsi d'éviter sa construction complète, soit l'énumération de tous les successeurs directs ou indirects d'un sommet S_j *de l'arborescence*

Propriété

Soit $\mathcal{F} = \{S_1, S_2, \ldots, S_j, \ldots, S_k\}$ *une famille de sous-ensembles de* S *dont l'union est égale à* S*.*

- **Si** *tous les ensembles de la famille de* $\mathcal{F}$ *ont une évaluation exacte et* **si** S_j *est un ensemble de* $\mathcal{F}$ *d'évaluation maximale* **alors**
  s^* *telle que* $f(s^*) = v(P_j)$ *est une solution optimale de* (P)*.*

Schéma général des méthodes par séparation et évaluation

Notations :

- s^* : meilleure solution de S connue. On note $\underline{v}(P)$ sa valeur,
- $\mathcal{F}$: famille de sous-ensembles **actifs** de S,
- $\overline{v}(P_j)$: majorant de (P_j), le problème d'optimisation associé à l'ensemble S_j

Algorithme séparation et évaluation

Initialisation : s^* *indéterminé ;* $\underline{v}(P) \leftarrow -\infty$ *;* $\mathcal{F} \leftarrow S$
Tant que $\mathcal{F} \neq \emptyset$ **Faire**

- *CHOIX : Choisir* $S_j \subset \mathcal{F}$ **actif**
- **Si** S_j *n'a pas été évalué* **Alors**
  - *ÉVALUATION : déterminer* $\overline{v}(P_j)$
  - **Si** $\overline{v}(P_j) \leq \underline{v}(P)$ **alors** $\mathcal{F} \leftarrow \mathcal{F} - \{S_j\}$
  - **Sinon**
  - *Si* $\exists\ \hat{s} \in S_j$ *t.q.* $f(\hat{s}) = \overline{v}(P_j)$ **alors**
    1. $s^* \leftarrow \hat{s}$ *;* $\underline{v}(P) \leftarrow \overline{v}(P_j)$
    2. $\mathcal{F} \leftarrow \mathcal{F} - \{S_j\}$
    3. **Pour tout** $S_i \in \mathcal{F}$ *t.q.* $\overline{v}(P_i) \leq \underline{v}(P)$ **Faire** $\mathcal{F} \leftarrow \mathcal{F} - \{S_i\}$
  - **FinSi**
- **Sinon** *SÉPARATION : création des* $S_{j_1}, S_{j_2}, \ldots, S_{j_k}$
  $\mathcal{F} \leftarrow \mathcal{F} - \{S_j\} + \{S_{j_1}\} + \{S_{j_2}\} + \ldots + \{S_{j_k}\}$
- **FinSi**

Fin Tant que

Sommaire

Généralités et Définitions

Définition de l'arborescence : concept de séparation

Concept d'évaluation

Mise en oeuvre pratique : Application à la programmation linéaire en nombres entiers

Différents degrés de liberté :

- Stratégie de parcours de l'arborescence : **CHOIX** du sommet à traiter à chaque étape
- L'**ÉVALUATION** : détermination du majorant de $v(P_j)$, la valeur du sous-problème associé au sous-ensemble de solutions S_j
- La procédure de **SÉPARATION** du sommet choisi

Soit le programme linéaire en nombres entiers (P) suivant :

$$(P)\begin{cases} \max & z = cx \\ \text{s.c} & Ax = b \\ & x \geq 0 \quad \textit{entier} \end{cases}$$

Posons $S = \{x | Ax = b, x \geq 0 \quad \textit{entier}\}$

SÉPARATION D'UN SOMMET

- Dans le cas de la programmation linéaire en nombres entiers, l'arbre construit est le plus souvent binaire.
- Un ensemble S_j est séparé en 2 sous-ensembles S_{j1} et S_{j2} soumis à des conditions complémentaires de sorte que :

$$S_j = S_{j1} \cup S_{j2}$$

$$\text{avec } S_{j1} = \{S_j | \textit{ condition}\} \text{ et } S_{j2} = \{S_j | \textit{ condition inverse}\}$$

À chaque sommet S_j est donc associé un PL en nombres entiers (P_j) dont le système d'équations à satisfaire est le système initial de (P) augmenté de nouvelles contraintes.

Application au problème du sac à dos

2 exemples traités :

$$\begin{cases} \max z = 3x_1 + 4x_2 + 2x_3 + 4x_4 + x_5 + 2x_6 \\ \text{s.c.} \quad x_1 + 2x_2 + x_3 + 4x_4 + 2x_5 + 4x_6 \leq 6 \\ \qquad x_1, x_2, x_3, x_4, x_5, x_6 \in \{0,1\} \end{cases}$$

$$\begin{cases} \max z = 8x_1 + 18x_2 + 20x_3 + 11x_4 \\ \text{s.c.} \quad 3x_1 + 7x_2 + 9x_3 + 6x_4 \leq 17 \\ \qquad x_1, x_2, x_3, x_4 \in \{0,1\} \end{cases}$$