6. Programmation Linéaire en Nombres Entiers et Branch & Bound

15 Novembre 2023

Problème combinatoire générique

$$(P):\ \min_{x\in\mathcal{X}} f(x)$$

$\mathcal{X}$ ensemble fini, trop grand pour énumérer toutes les solutions en temps raisonnable

Description Branch-and-Bound

On dispose d'une "borne inférieure" $\lambda : \mathcal{P}(X) \to \mathbb{R}$ qui a toute partie Y de X, associe

$$\lambda(Y) \leq \min_{x \in Y} f(x).$$

On suppose que λ se calcule "facilement" (par exemple, en temps polynomial).

L'algorithme maintient :

- un ensemble $\mathcal{Y}$ de parties de X telle que $\bigcup_{Y \in \mathcal{Y}} \mathcal{Y}$ contient un minimum de f sur X
- la meilleure solution courante $\tilde{x}$.

Idée principale. On peut se débarrasser de $Y \in \mathcal{Y}$ si $\lambda(Y) \geq f(\tilde{x})$.

Itération

Initialisation : $\mathcal{Y} = \{X\}$.

On part d'une solution réalisable quelconque $\tilde{x}$. Une itération est

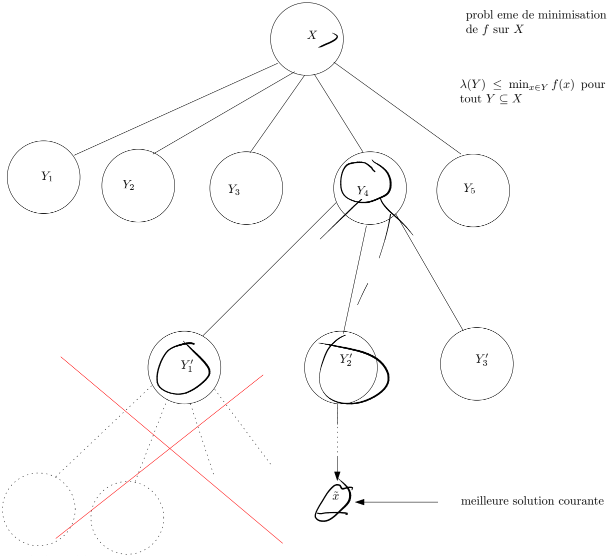
1. Choisir une partie Y de $\mathcal{Y}$.
2. Partitionner Y en parties $Y_1', Y_2', \ldots, Y_s'$ ($s = 2$ en général)
3. Supprimer Y de $\mathcal{Y}$.
4. Faire pour $i = 1, \ldots, s$:
   - si l'on trouve directement $y \in Y_i'$ minimisant $f(y)$ et si $f(y) < f(\tilde{x})$, poser $\boxed{\tilde{x} := y}$ $\min_{y \in Y_i'} f(y)$
   - si $\lambda(Y_i') < f(\tilde{x})$, poser $\mathcal{Y} := \mathcal{Y} \cup \{Y_i'\}$.
   - sinon $\lambda(Y_i') \geq f(\tilde{x})$ donc pas de solution meilleure que $\tilde{x}$ dans Y_i'

Calcul d'une solution réalisable

On trouve directement $y \in Y_i'$ minimisant $f(y)$ par exemple

- quand Y_i' est un singleton, ou
- quand par chance le calcul de $\lambda(Y_i')$ fournit une solution dans Y_i' (cas de la PLNE et de la relaxation linéaire ci-après).

Schéma en arbre



si $\lambda(Y_1') \geq f(\tilde{x})$, il ne sert a rien d'explorer cette région.

Performances

- Nombre fini d'itération, mais **non polynomial**.
- Les performances seront d'autant plus grandes que la borne est de bonne qualité : pour un problème de minimisation, plus elle grande, plus elle permettra de "couper".
- La façon de choisir et partitionner Y est également d'une grande importance $\rightarrow$ **stratégie de branchement**.

Exemple : programmation linéaire en nombres entiers

La **programmation linéaire en nombre entiers** (PLNE) se modélise, avec $A \in \mathbb{R}^{m \times n}$, et $b \in \mathbb{R}^m$ et $c \in \mathbb{R}^n$

$$(PLNE) \quad \begin{array}{ll} \min\limits_{x} & c^\top x \\ \text{s.c.} & Ax \leq b \\ & x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}. \end{array}$$

(Rappel : PL$\in \mathcal{P}$.)

Théorème. PLNE est $\mathcal{NP}$-difficile.

Application du branch-and-bound pour la PLNE

Une borne inférieure à la valeur optimale v_{plne} d'un PLNE s'obtient en relâchant la contrainte d'intégrité :

$$(PL) \quad \begin{array}{ll} \min\limits_{x} & c^\top x \\ \text{s.c.} & Ax \leq b \\ & x \in \mathbb{R}^n. \end{array}$$

et notons v_{pl} sa valeur optimale.

On a :

$$v_{\mathsf{plne}} \geq v_{\mathsf{pl}}.$$

Exemple

$\mathcal{Y} = \{X\}$

$\lambda(X) = 6$

$$\left|\begin{array}{ll} \text{Min} & x_1 + 5x_2 \\ \text{s.c.} & \left.\begin{array}{l} 4x_1 - 2x_2 \leq 13 \\ 2x_1 + 8x_2 \geq 11 \\ x_1, x_2 \in \mathbb{Z}. \end{array}\right\} X \end{array}\right.$$

On commence par résoudre ce programme en ne demandant que $x_1, x_2 \in \mathbb{R}$ (on relâche la contrainte d'intégrité).

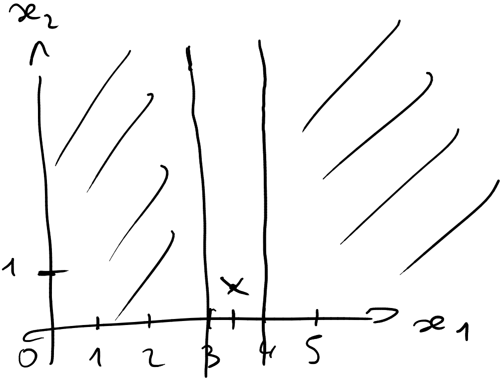
On obtient : $x_1 = 3.5$, $x_2 = 0.5$, $v = 6$

$X = Y_1 \cup Y_2$

On "branche sur x_1" :

$$\begin{array}{ll} \text{Min} & x_1 + 5x_2 \\ \text{s.c.} & \left.\begin{array}{l} 4x_1 - 2x_2 \leq 13 \\ 2x_1 + 8x_2 \geq 11 \\ \boxed{x_1 \leq 3} \\ x_1, x_2 \in \mathbb{Z}. \end{array}\right\} Y_1 \end{array}$$

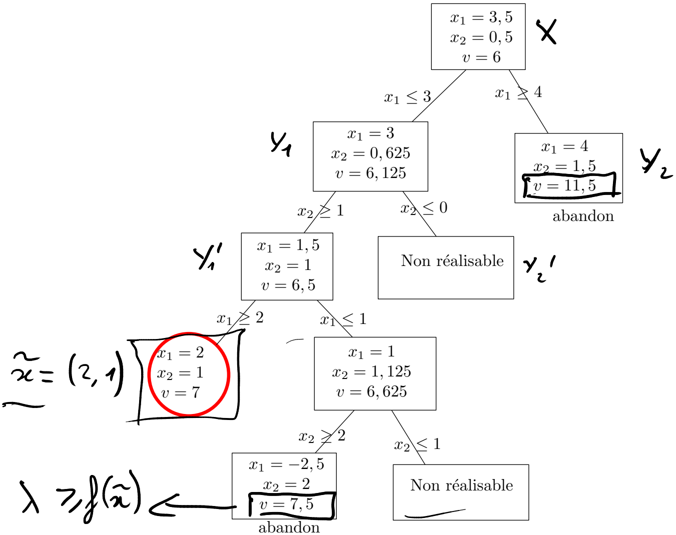
$$\begin{array}{ll} \text{Min} & x_1 + 5x_2 \\ \text{s.c.} & \left.\begin{array}{l} 4x_1 - 2x_2 \leq 13 \\ 2x_1 + 8x_2 \geq 11 \\ \boxed{x_1 \geq 4} \\ x_1, x_2 \in \mathbb{Z}. \end{array}\right\} Y_2 \end{array}$$



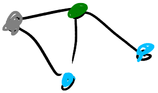
x_2
1
x_1
0
1
2
3
4
5

Tout l'algorithme

En commençant toujours par la branche de gauche.



Coloration et PLNE



$G = (V, E)$

$K = |V|$

Modéliser le problème de coloration sur un graphe $G = (V, E)$ par un programme linéaire en nombres entiers.

- variables : $\forall v \in V, \forall i \in [K]$, $x_{vi} = \mathbb{1}\{v \text{ colorier avec couleur } i\} \in \{0,1\}$

$y_i = \mathbb{1}\{\text{couleur } i \text{ utilisée}\} \in \{0,1\} \quad \forall i \in [K]$

- Contraintes : $\sum_{i \in [K]} x_{vi} = 1, \forall v$ → une seule couleur par sommet

$\forall (u,v) \in E, \forall i \quad x_{ui} + x_{vi} \leq 1$

y_i

- Objectif : $\min_{x,y} \sum y_i$

- $x_{vi} \leq y_i \quad \forall v \; \forall i$

Coloration et PLNE

Modéliser le problème de coloration sur un graphe $G = (V, E)$ par un programme linéaire en nombres entiers.

Solution : soit K une borne supérieure au nombre chromatique (si on le choisit trop petit, le PLNE est infaisable).

$$
\begin{array}{lll}
\min\limits_{x,y} & \displaystyle\sum_{i=1}^{K} y_i & \\
& x_{ui} + x_{vi} \leq y_i & \forall (u,v) \in A, \forall i \in \{1, \ldots, K\} \\
& \displaystyle\sum_{i=1}^{K} x_{vi} = 1 & \forall v \in V \\
& x_v \in \{0,1\} & \forall v \in V \\
& \boxed{y_i \in \mathbb{R}} & \forall i \in \{1, \ldots, K\}
\end{array}
$$

Flots et PLNE

Exercice

Modéliser le problème de s-t flot maximum sur un graphe orienté par un programme linéaire en nombres entiers. $G = (V, A)$. On notera u_a la capacité d'un arc.

Flots et PLNE

Exercice

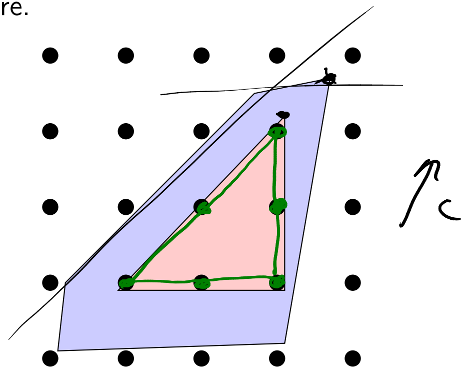
Modéliser le problème de s-t flot maximum sur un graphe orienté par un programme linéaire en nombres entiers. $G = (V, A)$. On notera u_a la capacité d'un arc.

Solution

$$\begin{array}{lll}
\max_x & \sum_{a \in \delta^+(s)} x_a - \sum_{a \in \delta^-(s)} x_a & \\
\text{s.t.} & \sum_{a \in \delta^-(v)} x_a = \sum_{a \in \delta^+(v)} x_a & \forall v \in V \backslash s, t \\
& x_a \leq u_a & \forall a \in A \\
& x_a \in \mathbb{Z}_+, & \forall a \in A
\end{array}$$

Bien choisir sa formulation

Trouver un équilibre entre qualité de la relaxation et taille du programme linéaire.



L'enveloppe convexe des points entiers à l'intérieur d'un PLNE est un polyhèdre, avec généralement un nombre exponentiel de facettes.

Matrice totalement unimodulaire

Une matrice A à coefficients entiers est *totalement unimodulaire* si toute sous matrice carrée admet un déterminant égal à -1, 0, ou +1.

Théorème

Si A est totalement unimodulaire si et seulement si les sommets du polyhedre $\{x, Ax \leq b\}$ sont entiers pour tout vecteur b entier.

Lemma

Soit A une matrice à coefficients dans $\{+1, 0, -1\}$. Si toute colonne contient au plus un $+1$ et au plus un -1, alors A est totalement unimodulaire.

Exercice : montrer qu'une matrice du problème de s-t flot maximum est totalement unimodulaire.

$$\sum_{a\in\delta^-(v)} x_a - \sum_{a\in\delta^+(v)} x_a = 0 \quad \forall v \notin s,t \qquad |V|-2$$

$$x_a \le u_a \quad \forall a \qquad + |A|$$

$$\overbrace{\qquad\qquad}^{|A|}$$

$$A_{a=(u,v)}$$

$$A = \begin{array}{cc} & a=(u,v) \\ \begin{array}{c} \\ u \\ V \\ v \\ \\ \end{array} & \left[\begin{array}{c} 1 \\ 0 \\ 0 \\ -1 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ 0 \end{array}\right] \\ A & \left[\begin{array}{ccc} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{array}\right] \end{array}$$

$$b = \begin{pmatrix} 0 \\ u \end{pmatrix}$$

Remarques sur la PLNE

Pour la PLNE, on choisit souvent de brancher sur la variable la "moins" entière, en partant du nœud de l'arbre pour lequel la relaxation est la meilleure.

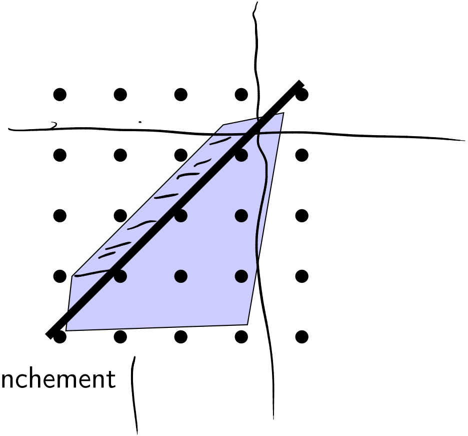
En pratique : **ne jamais reprogrammer un branch-and-bound pour la PLNE**. Il existe de nombreux solvers – libres ou payants – extrêmement performants.

Exemple de solver libre : HiGHS, GLPK, SCIP.
Exemple de solveur commerciaux : Gurobi, CPLEX, Xpress

Ne jamais réimplémenter un solveur de PLNE

Sur un même ordinateur, la version actuelle de CPLEX est 580 000 fois plus rapide que la version initiale de 1991.

- Programmation linéaire (maintenir la factorisation de la matrice)
- Coupes valides



- Stratégies de branchement

Inégalité valide

Soit avec $A \in \mathbb{R}^{m \times n}$, et $b \in \mathbb{R}^m$ et $c \in \mathbb{R}^n$

$$\begin{aligned} \min_x \quad & c^\top \mathbf{x} \\ \text{s.c.} \quad & A\mathbf{x} \leq b \\ & \mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}. \end{aligned}$$

Definition

Soit $\alpha \in \mathbb{R}^n$ et $\beta \in \mathbb{R}$, l'inégalité $\alpha^\top \mathbf{x} \leq \beta$ est valide ssi elle est satisfaite par toute solution réalisable dans $\mathbb{Z}^p \times \mathbb{R}^{n-p}$ mais pas par toute solution du relâché linéaire.

$\rightarrow$ Permet d'améliorer la qualité du relâché linéaire.

Branch and cut

Version améliorée du Branch and bound : étape additionnelle qui consiste (pas systématique) à renforcer la relaxation linéaire.

Comment ? Une famille $\mathcal{F}$ d'inégalités valides $(\alpha_f, \beta_f)_{f\in\mathcal{F}}$ est considérée. En fixant $\mathbf{x}$ la solution courante, on définit l'inégalité valide la plus violée f^* par le **problème de séparation** :

$$\min_{f\in\mathcal{F}} \beta_f - \alpha_f^\top \mathbf{x}$$

Dans le cas où $\beta_{f^*} - \alpha_{f^*}^\top \mathbf{x} < 0$, on ajoute f^* au PL. La solution courante ne la satisfait pas, on améliore donc la relaxation.

Remarque En pratique on considère souvent $\mathcal{F}$ de taille exponentielle $\rightarrow$ on a besoin d'un problème de séparation facile.

Comment aborder un problème de RO ?

1. S'il peut se modéliser sous la forme d'un problème facile connu $\Longrightarrow$ algorithme exact
2. Formulation PLNE tractable $\Longrightarrow$ PLNE avec solveur industriel
3. Métaheuristique