La programmation mathématique en nombres entiers

Algorithmes de coupes

Stanislas Francfort

Orange Labs

5 Juin 2017

Algorithmes de coupes

- Comment améliorer une relaxation ?
- Deux cas avec grand nombres de contraintes :
  - Formulation avec un grand nombre de contraintes
  - Ajout de contraintes pour améliorer une relaxation

Algorithmes de coupes

- Nous pouvons générer *des nouvelles* contraintes automatiquement
- Nous avons des procédures pour les ajouter à un PLNE
- (Remarque : différent de l'approche polyédrale qui consiste à ajouter *les meilleures* contraintes)

Coupes et séparation

$$(P) Max\{c^T x | Ax \leq b\}$$

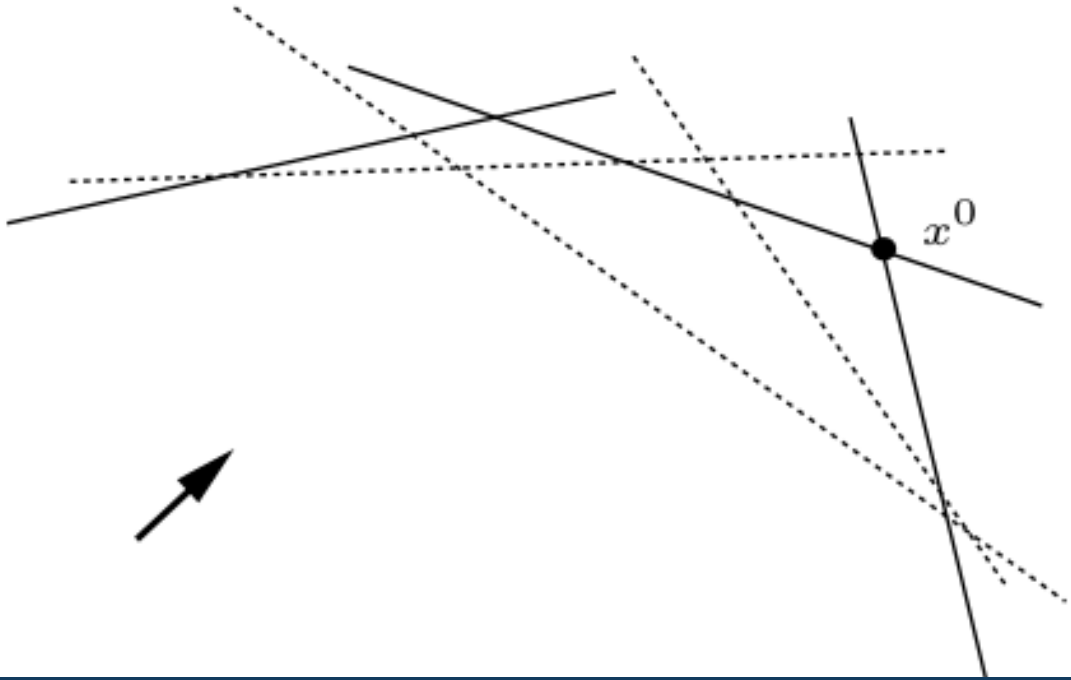
- Avec n variables
- mais avec énormément (exponentiel ?) de contraintes
- on dit que le programme est *non compact* sur les contraintes.

Coupes et séparation

- Ca ne peut pas être une instance d'un algorithme de branchement et évaluation
- il ne peut pas être utilisé directement dans les solveurs entiers
- **Mais** : un cadre algorithmique efficace, les *algorithmes de coupes*.

Coupes et séparation

- Pour débuter :
- Choisissons un sous-ensemble $A_0 x \leq b_0$ de contraintes de $Ax \leq b$
- tel qu'il existe une solution finie x^0 au problème $(P0)$ $Max\{c^T x | A_0 x \leq b_0\}$
- x^0 est un point extrême du polyèdre défini par $A_0 x \leq b_0$



Coupes et séparation

- Mais x^0 peut ne pas satisfaire certaines contraintes de $Ax \leq b$
- On dit que x^0 *viole* certaines contraintes
- dans ce cas, x^0 n'est pas solution de $Ax \leq b$

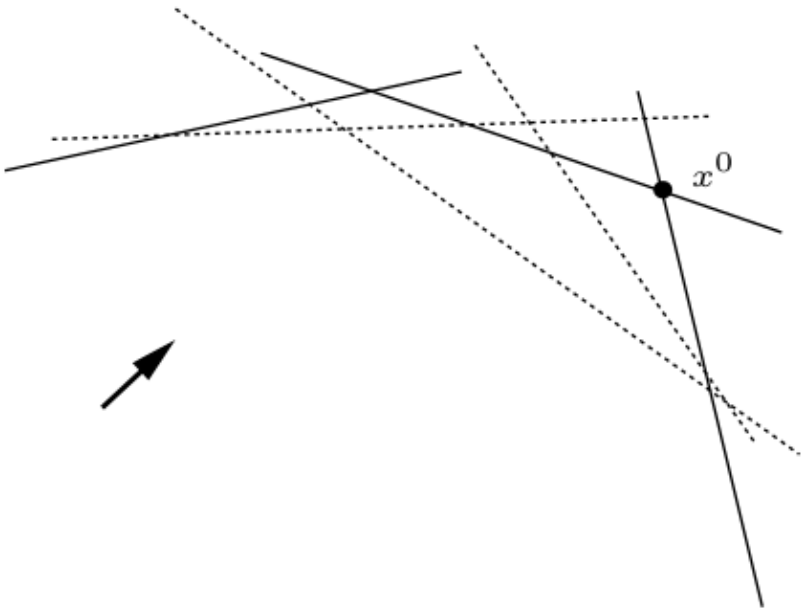
Problème de séparation : étant donné x, déterminer si x satisfait toutes les inégalités de $Ax \leq b$ et sinon : trouver une inégalité violée par x.

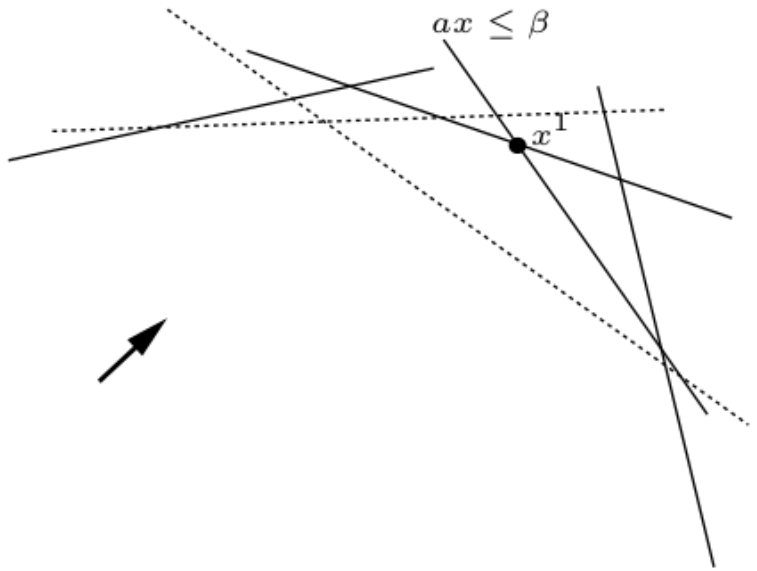
Coupes et séparation

- Un *algorithme de séparation* résoud ce problème
- Si on sait déterminer une inégalité $\alpha x \leq \beta$ qui est violée par le point extrême x^0
- Alors on ajoute alors cette contrainte aux inégalités du système courant
- On appelle alors une telle inégalité *une coupe*
- car elle "coupe" le point indésirable de l'espace des solutions possibles.

Coupes et séparation

- La contrainte a séparé le morceau d'espace indésirable





Coupes et séparation

- On pose :

$$(A_1 x \leq b_1) = (A_0 x \leq b_0) \cup (\alpha x \leq \beta)$$

- on repète le même algorithme de séparation pour le nouveau point extrême x^1
- Remarque : ceci ne nécessite que la résolution d'un PL, et d'un algorithme de séparation, et non d'un PLNE

Coupes et séparation

- On réitère l'algorithme de séparation tant que l'on peut ajouter des contraintes
- Ceci s'appelle : *méthode de coupes* (*cutting plane algorithm*)
- A la fin d'une méthode de coupes, on obtient donc nécessairement un point extrême $x^\star$
- $x^\star$ est point extrême du système courant et qui n'est pas violé par une contrainte de $Ax \leq b$
- $x^\star$ est donc une solution optimale

Coupes et séparation

- Questions importantes :
  - Quel algorithme de séparation ?
  - La méthode de coupe se termine-t-elle ?
  - Quelle complexité pour la méthode de coupe ?

Coupes et séparation

Theorem (Grôtschel, Lovasz et Schrijver (1981))

Une méthode de coupes sur un système $Ax \leq b$ de contraintes est polynomial si et seulement si l'algorithme de séparation associé à $Ax \leq b$ est polynomial.

- Résultat **fondamental** qui permet de manipuler des formulations exponentielles pour la PLNE !
- Il indique ainsi qu'optimiser est équivalent a séparer.

Coupes et séparation

- Conséquence du théorème :
- comme on ajoute une inégalité un nombre polynomial de fois
- $\Rightarrow$ le système courant a au plus une taille polynomiale ! Ainsi, au lieu de manipuler un système exponentiel, un simple système polynomial est suffisant.
- non surprenant, car pour définir un point extrême (solution) il suffit de n contraintes (linéairement indépendantes) satisfaites a l'égalité
- le reste peut être "écarté" du PL correspondant

- **Comment trouver ces n contraintes ?**

Contraintes de coupes pour le TSP

Formulation pour le voyageur de commerce symétrique

$$Min\ c(e)x(e) \tag{1}$$

$$\sum_{e \in \delta(v)} x(e) = 2,\ \forall v \in V \tag{2}$$

$$\sum_{e \in \delta(W)} x(e) \geq 2,\ \forall W \subsetneq V,\ W \neq \emptyset \tag{3}$$

$$x(e) \geq 0,\ \forall E \in E \tag{4}$$

$$x(e) \in \{0,1\},\ \forall E \in E \tag{5}$$

- Les contraintes (3) sont en nombre exponentiel
- On peut construire un algorithme de coupes
- en prenant comme ensemble initial les contraintes (2) et les contraintes triviales.

Contraintes de coupes pour le TSP

- But : résoudre la relaxation linéaire de cette formulation PLNE
- Notons $x^\star$ une solution optimale du PL limité aux contraintes initiales
- Problème de séparation :
  - déterminer une contrainte (3) violée par $x^\star$ si elle existe et dire sinon qu'il n'en n'existe pas.
  - quel est cet algorithme de séparation ?

Contraintes de coupes pour le TSP

- Déterminer une contrainte de coupe (3) violée par $x^\star$
- C'est à dire : déterminer une coupe min dans le graphe G
- en utilisant pour capacité le vecteur $x^\star$ associée aux arêtes de G
- En effet, si l'on dispose d'une coupe $\delta(W^\star)$ avec :
  - $W^\star \subsetneq V$
  - $W^\star \neq \emptyset$
  - tel que $x^\star(\delta(W^\star)) = \sum_{e \in \delta(W)} x^\star(e)$ soit **minimum**
- Alors on a deux cas

Contraintes de coupes pour le TSP

- Les deux cas :
  - soit $x^\star(\delta(W^\star)) < 2$ dans ce cas, on a déterminé une contrainte $\sum e \in \delta(W)x(e) \geq 2$ violée
  - soit $x^\star(\delta(W^\star)) \geq 2$ dans ce cas, toutes les contraintes (3) sont non violées.
- Comme on sait déterminer une coupe-min avec des capacités positives en temps polynomial
- Alors on sait donc résoudre en temps polynomial cette formulation exponentielle
- Cependant la solution obtenue sera fractionnaire
- On peut améliorer encore, avec par exemple les *inégalités de peigne [Chvatal 1994]*

Contraintes de clique pour le stable

- Problème du stable : contraintes de cliques :
- (Contient un nombre exponentiel de cliques)

$$Max \sum_{u \in V} c(u)x(u) \tag{6}$$

$$\sum_{u \in K} x(u) \leq 1, \ \forall K \text{ clique de } G \tag{7}$$

$$x(u) \in \{0,1\}, \ \forall u \in V \tag{8}$$

- But : résoudre la relaxation linéaire de cette formulation PLNE
- Notons $x^\star$ une solution optimale du PL limité aux contraintes triviales
- Problème de séparation :
  - déterminer une contrainte (7) violée par $x^\star$ si elle existe et dire sinon qu'il n'en n'existe pas.
  - quel est cet algorithme de séparation ?

Contraintes de clique pour le stable

- problème de séparation : déterminer une contrainte $\sum_{u \in K} x(u) \leq 1, \ \forall K$ clique de G violée par $x^\star$
- Problème de séparation : rechercher une clique K de plus grand poids positifs associés aux sommets dans le graphe G
- Or ce problème est NP-complet
- $\Rightarrow$ on ne peut pas construire un algorithme de coupes pour cette formulation...

Contraintes de clique pour le stable

$$
\begin{aligned}
Max \sum_{u \in V} & c(u)x(u) \\
& x(u) + x(v) \leq 1, \ \forall uv \in E \\
& \sum_{u \in K} x(u) \leq 1, \ \forall K \text{ clique de } G, \ |K| \geq 3 \\
& x(u) \in \{0,1\}, \ \forall u \in V
\end{aligned}
$$

- Seules les contraintes d'arêtes sont nécessaires pour cette formulation
- Elles peuvent donc être énumérées et utilisées tout au long de l'algorithme de coupes

Contraintes de clique pour le stable

- $\sum_{u \in K} x(u) \leq 1, \ \forall K$ clique de $G, \ |K| \geq 3$
- Les contraintes associées à des cliques de tailles au moins 3 seront utilisées dans un algorithme de coupes heuristiques
- c'est-a-dire qu'au lieu de déterminer s'il existe ou non une contrainte de cliques violées, on recherche heuristiquement une contrainte de cliques violée.

Contraintes de clique pour le stable

- Rechercher heuristiquement une contrainte de cliques violée :
- en utilisant par exemple un algorithme glouton :
  - pour une solution $x^\star$ du problème relaxée
  - on recherche un sommet de grand poids
  - puis on essaye d'ajouter un sommet de grand poids formant une clique avec le sommet précédent
  - répéter l'opération
- Algorithme non exact
- mais permet d'obtenir des contraintes de cliques en temps polynomial
- on peut donc le réitérer dans un processus de génération de contraintes.

Contraintes de clique pour le stable

- Limitation :
- ne permet pas de résoudre la relaxation linéaire du PLNE du stable donné précédemment
- Mais, on obtient à la fin une bien meilleure valeur de relaxation linéaire que la formulation limitée aux contraintes dites "aux arêtes".

Algorithmes de coupes et branchements

Theorem

- *Si on connaît un système linéaire définissant un polyèdre entier*
- *ET que l'on connait un algorithme de séparation polynomial pour chacune de ces contraintes*
- *Alors on obtient une méthode de coupes qui permet de résoudre un PLNE en temps polynomial*

Algorithmes de coupes et branchements

- Le problème de séparation sur certaines classes d'inégalités valides peut être NP-difficile.
- Dans ce cas, on ne peut disposer que de techniques de séparation approchées
- Ainsi, une méthode de coupes seule peut ne fournir que des solutions fractionnaires (non optimales)

Algorithmes de coupes et branchements

- Si la méthode de coupes ne donne que des solutions fractionnaires :
- on exécute une étape de branchement
- Par exemple en choisissant une variable fractionnaire x_i dans la solution et à considérer deux sous-problèmes du problème courant en fixant x_i à 0 pour l'un et à 1 pour l'autre
- On applique alors la méthode de coupes pour les deux sous-problèmes.
- La solution optimale du problème sera donc la meilleure entre les deux

Algorithmes de coupes et branchements

- Une fois les deux problèmes résolus :
- La phase de branchement est répétée jusqu'a l'obtention d'une solution entière optimale
- La combinaison de *branchements* et de *coupes* s'appelle méthode de *coupes et branchements (Branch and Cut method)*
- Méthode très efficace pour certains problèmes d'optimisation combinatoire réputés difficiles
- Efficace sur : voyageur de commerce ou coupe maximum.

Algorithmes de coupes et branchements

- La mise en oeuvre d'un algorithme de coupe et branchement est assez complexe
- elle nécessite de bien gérer un algorithme de branchement et évaluation et plusieurs algorithmes de séparation
- Il faut également gérer les contraintes en très grand nombre
- et un solveur linéaire...

Résolution d'un PLNE compact

- Il est utile d'avoir des classes d'inégalités valides (et non redondantes) pour un PLNE
- Les choisir de manière à couper des points extrêmes fractionnaires qui apparaissent dans la relaxation étudiée du polyèdre
- On les détermine soit par intuition,
- soit par extrapolation de cas simples,
- soit en les dérivant par des procédés tels que celui de *Chvatal-Gomory*,...

- Ces idées algorithmiques permettent aujourd'hui de résoudre efficacement des PLNE de bonnes tailles (quelques milliers de lignes/contraintes)
- Surtout dans les cas où le PLNE a une structure particulière

Contraintes valides et renforcement

- Soit le PLNE (P) $Max\ \{c^T x | Ax \leq b,\ x \in \mathbb{N}^n\}$
- avec un nombre compact de contraintes et de variables
- on veut ajouter des contraintes supplémentaires.
- Soit le polyèdre $P = \{x \in \mathbb{R}^n | Ax \leq b\}$

Définition Une inégalité (ou contrainte) $\alpha x \leq \beta$ est dite *valide* pour le polyèdre P si elle est vérifiée par tous les points de P i.e. $P \subseteq \{x \in \mathbb{R}^n | \alpha x \leq \beta\}$.

Contraintes valides et renforcement

- Si on ajoute une inégalité valide violée à un PLNE
- Alors cela permet de couper un point extrême indésirable
- Si cet ajout est réalisé dans le cadre d'un algorithme de coupes
- Alors cela permet même d'écarter des points optimaux qui ne sont pas des solutions du PLNE.
- De plus, ces contraintes permettent d'obtenir une relaxation plus intéressantes
- On parle alors de *renforcement* de la formulation.

Somme de Chvatal : exemple du problème du stable max

- La *somme de Chvatal-Gomory* permet d'obtenir de telles inégalités valides de manière automatique
- Principe :
  - additionner plusieurs contraintes
  - et diviser les coefficients de la contrainte obtenue
  - en utilisant le fait que certaines sommes de termes sont entières, on peut en déduire une nouvelle contrainte.

Somme de Chvatal : exemple du problème du stable max

- Exemple : polyèdre associé a la relaxation linéaire de la formulation aux arêtes du stable.
- Soit $G = (V, E)$ un graphe non orienté. Soit A la matrice d'incidence aux arêtes de G.
- Le polyèdre $(\mathcal{D}) = \{x | xA \leq Id,\ x \geq 0\}$ a toujours des sommets à coordonnées non entières si G n'est pas biparti
  - G non biparti $\Rightarrow$ contient un cycle impair C
  - Soient $(u_1, \ldots, u_k)$ les sommets de C, et soit $z \in \mathbb{R}^n$ dont les composantes sont nulles si $u \notin \{u_1, \ldots, u_k\}$ et $\frac{1}{2}$ sinon
  - z vérifie bien à l'égalité les k contraintes associées aux arêtes de G et les $n - k$ contraintes triviales associées aux sommets de $V \backslash \{u_1, \ldots, u_k\}$
  - Comme les vecteurs des coefficients de ces contraintes sont linéairement indépendantes. z est un point extrême fractionnaire de $P(G)$.

Somme de Chvatal : exemple du problème du stable max

- On veut produire des contraintes qui coupent ce type de points extrêmes
- Supposons que G contienne un cycle impair de C
- On somme les k contraintes associées aux k arêtes du cycle
- On obtient ainsi :

$$x(u_i) + x(u_i + 1) \leq 1, \ \forall i \in \{1, \ldots, k-1\}$$
$$x(u_1) + x(u_k) \leq 1$$

- En sommant le tout, on obtient : $2 \sum_{u \in V(C)} x(u) \leq k$, d'où $\sum_{u \in V(C)} x(u) \leq \frac{k}{2}$
- Comme le côté gauche est entier, alors le côté droit doit l'être aussi
- et comme k impair, on obtient : $\sum_{u \in V(C)} x(u) \leq \frac{k-1}{2}$

Somme de Chvatal : exemple du problème du stable max

- Lorsque k est impair, on a donc les contraintes dites de *cycles impairs* pour le problème du stable :

$$\sum_{v \in V(C)} x(v) \leq \frac{|C| - 1}{2}, \ \forall C \text{ cycle impair}$$

- On remarque qu'alors ces contraintes coupent le point fractionnaire z.

Méthode de Chvatal-Gomory et Lovasz-Schrijver.

- Deux méthodes génériques :
  - Chvatal-Gomory
  - Lovasz-Schrivjer
- *Méthode duale fractionnaire* : utiliser ces méthodes pour créer un algorithme de coupes génériques et adaptables à tout PLNE
- Cette méthode converge vers une solution optimale mais exessivement lentement
- En revanche, les inégalités produites par cette méthode permettent d'accélérer fortement les algorithmes de coupes et branchements génériques.

Autres méthodes

- La méthode duale fractionnaire de Chvatal-Gomory est malheureusement peu efficace si elle est utilisée seule
- Le principe a été étendu et permet de construire d'autres "renforcement" d'un PLNE quelconque :
  - utilisation des structures sous-jacentes de la matrice : par exemple des structures de stable
  - utilisation des structures particulières de certaines contraintes : par exemple de sac à dos
  - Lovasz-Shrivjer : déduire des contraintes en les "multipliant" entre elles. Permet *Lift-and-Project* initiée par *Balas*
  - *méthode disjonctives* qui génèrent aujourd'hui les coupes les plus efficaces (*projet Bonemine de Cornuejols, Margot, Bonami*).
- $\Rightarrow$ cela a permis des améliorations substancielles des solveurs PLNE génériques
- **Mais** pas de résoudre des problèmes OC difficile ! Étude dédiée nécessaire

Lift and Project

- Méthode *Lift-and-Project* (*Lovasz-Schrijver*)
- S'élever dans un espace de dimension supérieur
- et redescendre dans un espace de dimension inférieur
- permet de générer des inégalités valides

Lift and Project

- Soit $P = \{x \in \mathbb{Z}^n | Ax \leq b\}$
- **Lift** : Multiplier $Ax \leq b$ par x_j et par $(1 - x_j)$
  - $(Ax)x_j \leq bx_j$
  - $(Ax)(1 - x_j) \leq b(1 - x_j)$
- substituer $y_{ij} = x_i x_j$ pour $i = 1, \ldots, n$, et $i \neq j$, et x_j^2 par x_j
- Soit $L_j(P)$ le polyèdre obtenu
- **Project** : Projeter $L_j(P)$ de nouveau sur la variable x en éliminant les variables y
- On note P_j le polyèdre obtenu $(P_j = (L_j(P))_x)$

Lift and Project

Theorem

$P_j = conv(P \cap \{x \in \mathbb{R}^n | x_j \in \{0, 1\}\})$

Exemple

$P = \{(x_1, x_2 | Ax \leq b\}$ avec $Ax \leq b$ exprimé ainsi :

$$\begin{aligned} 2x_1 - x_2 &\geq 0 \\ 2x_1 + x_2 &\leq 2 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

Exemple

Lift pour la variable x_1 nous donne :

$$
\begin{aligned}
2x_1^2 - x_2x_1 &\geq 0 \\
2x_1(1-x_1) - x_2(1-x_1) &\geq 0 \\
2x_1^2 + x_2x_1 &\leq 2x_1 \\
2x_1(1-x_1) + x_2(1-x_1) &\leq 2(1-x_1) \\
x_1^2 &\geq 0 \\
x_1(1-x_1) &\geq 0 \\
x_2x_1 &\geq 0 \\
x_2(1-x_1) &\geq 0
\end{aligned}
$$

Exemple

Le changement de variable $y = x_1 x_2$ et $x_1^2 = x_1$ nous donne :

$$\begin{aligned} 2x_1 - y &\geq 0 \\ -x_2 + y &\geq 0 \\ y &\leq 0 \\ x_2 - y &\leq 2 - 2x_1 \\ x_1 &\geq 0 \\ 0 &\geq 0 \\ y &\geq 0 \\ x_2 - y &\geq 0 \end{aligned}$$

Ce qui implique que $y = 0$

Exemple

Project : avec $y = 0$, on obtient :

$$\begin{aligned} 2x_1 &\geq 0 \\ -x_2 &\geq 0 \\ x_2 &\leq 2 - 2x_1 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

Et on obtient donc :

$$\begin{aligned} P_1 &= \{(x_1, x_2) \mid 0 \leq x_1 \leq 1,\ x_2 = 0\} \\ &= conv(P \cap \{(x_1, x_2) \mid x_1 \in \{0, 1\}\}) \end{aligned}$$

Lift and Project

Si on note $P_{i_1,i_2,\ldots,i_t} = ((P_{i_1})_{i_2\ldots})_{i_t}$, alors on a :

Theorem

$P_{i_1,i_2,\ldots,i_t} = conv(P \cap \{x \in \mathbb{R}^n | x_i \in \{0,1\},\ i \in \{i_1,\ldots,i_t\}\})$
et
$P_{1,2,\ldots,n} = P_{binaire}$

Lift and Project

- La méthode *Lift-and-Project* permet de générer des inégalités valides pour des programmes en $\{0, 1\}$
- Avantage théorique : construire les inégalités décrivant le polytope binaire
- avantage pratique : une variante permet de construire une méthode de Branchement et Coupe (Branch and Cut)
- Cette variante est efficace car elle s'interface bien avec la matrice du simplexe (*Balas, Ceria, Cornuéjols*)