Programmation quadratique en nombres entiers : approche par reformulation quadratique convexe

Amélie Lambert

CEDRIC-ENSIIE

JFRO - 19 mars 2010

Formulation générale

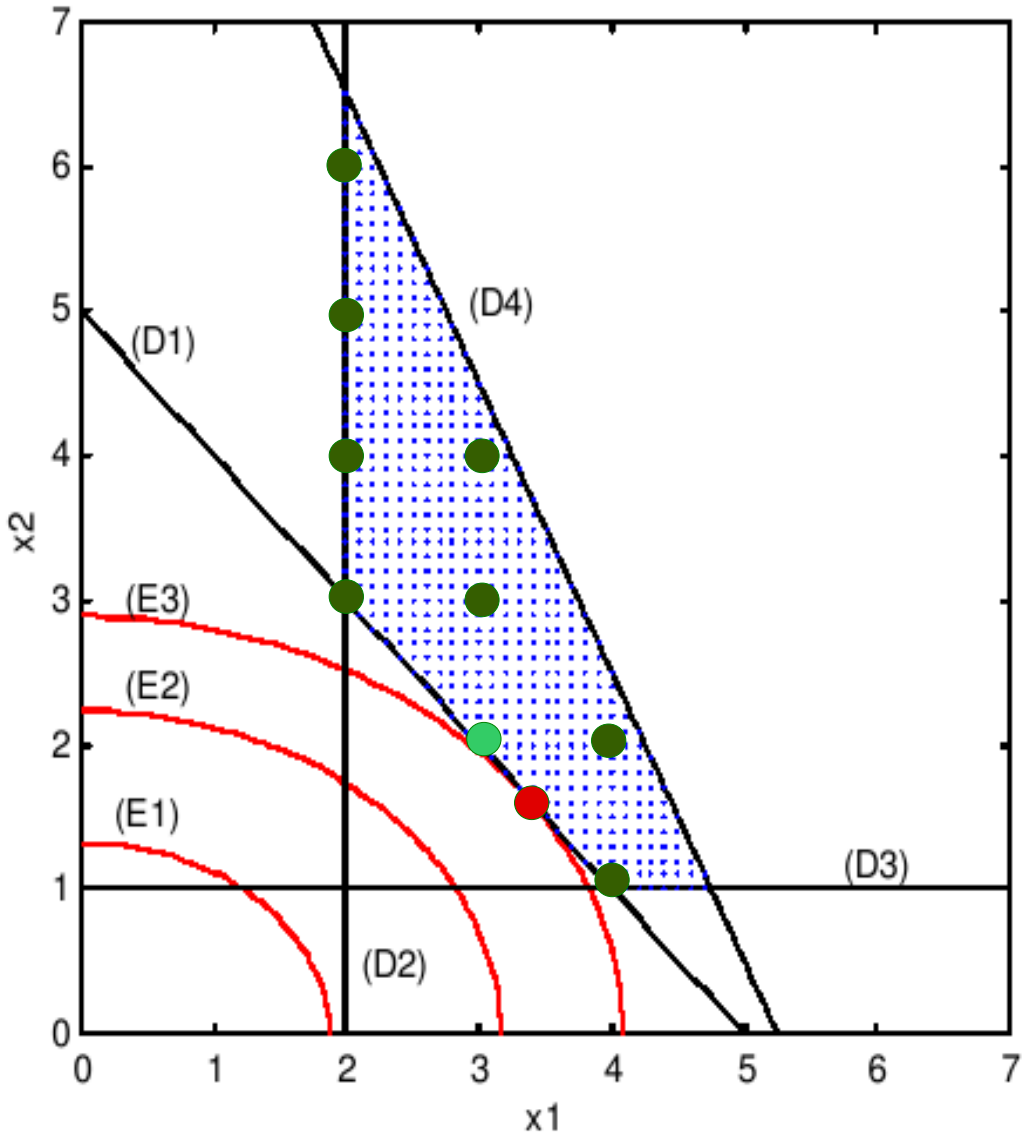
$$(QP)\begin{cases} Min & f(x) = x^T Qx + c^T x & \\ s.c. & Ax = b & \text{m égalités} \\ & Dx \leq e & \text{p inégalités} \\ & 0 \leq x_i \leq u_i & \text{n variables} \\ & x_i \text{ entier} & \end{cases}$$

Formulation générale

$$(QP)\begin{cases} Min & f(x) = x^T Qx + c^T x & \\ s.c. & Ax = b & \text{m égalités} \\ & Dx \leq e & \text{p inégalités} \\ & 0 \leq x_i \leq u_i & \text{n variables} \\ & x_i \text{ entier} & \end{cases}$$

$f(x)$ est convexe ssi Q est semi-définie positive (SDP)

Exemple (suite) : Représentation graphique



- $(D1) : x_1 + x_2 \leq 5$
- $(D2) : x_1 \geq 2$
- $(D3) : x_2 \geq 1$
- $(D4) : 2x_1 + x_2 \leq 10.5$

- $(E1) : 2x_1^2 + 4x_2^2 = 7$
- $(E2) : 2x_1^2 + 4x_2^2 = 20$
- $(E3) : 2x_1^2 + 4x_2^2 = 100/3$

Résolution d'un programme quadratique convexe en nombres entiers

Algorithme de Branch and Bound (B&B) fondé sur la relaxation continue

Résolution d'un programme quadratique convexe en nombres entiers

Algorithme de Branch and Bound (B&B) fondé sur la relaxation continue

A chaque noeud on résout la relaxation continue d'un problème convexe qui est polynômiale, mais la taille de l'arbre est exponentielle.

Résolution d'un programme quadratique convexe en nombres entiers

Algorithme de Branch and Bound (B&B) fondé sur la relaxation continue

A chaque noeud on résout la relaxation continue d'un problème convexe qui est polynômiale, mais la taille de l'arbre est exponentielle.

C'est l'algorithme utilisé dans les solveurs standards (Cplex, Xpress)

Résolution d'un programme quadratique convexe en nombres entiers

Algorithme de Branch and Bound (B&B) fondé sur la relaxation continue

A chaque noeud on résout la relaxation continue d'un problème convexe qui est polynômiale, mais la taille de l'arbre est exponentielle.

C'est l'algorithme utilisé dans les solveurs standards (Cplex, Xpress)

Cet algorithme est efficace si la valeur de la solution continue n'est pas trop éloignée de la valeur de la solution entière, i.e. plus la borne est fine, plus l'arbre est petit.

Performances du B&B

Performances du B&B

Dans le cas où la fonction objectif est **convexe** :

Performances du B&B

Dans le cas où la fonction objectif est **convexe** :

- Résolution en continue : de l'ordre de quelques **milliers** de variables et contraintes.

Performances du B&B

Dans le cas où la fonction objectif est **convexe** :

- Résolution en continue : de l'ordre de quelques **milliers** de variables et contraintes.
- Résolution en nombres entiers : de l'ordre de quelques **centaines** de variables et contraintes.

Performances du B&B

Dans le cas où la fonction objectif est **convexe** :

- Résolution en continue : de l'ordre de quelques **milliers** de variables et contraintes.
- Résolution en nombres entiers : de l'ordre de quelques **centaines** de variables et contraintes.

Dans le cas où la fonction objectif **n'est pas convexe** :

Performances du B&B

Dans le cas où la fonction objectif est **convexe** :

- Résolution en continue : de l'ordre de quelques **milliers** de variables et contraintes.
- Résolution en nombres entiers : de l'ordre de quelques **centaines** de variables et contraintes.

Dans le cas où la fonction objectif **n'est pas convexe** :

- Résolution en nombres entiers : de l'ordre de quelques **dizaines** de variables et contraintes (algorithmes d'approximation d'enveloppe convexe).

Performances du B&B

Dans le cas où la fonction objectif est **convexe** :

- Résolution en continue : de l'ordre de quelques **milliers** de variables et contraintes.
- Résolution en nombres entiers : de l'ordre de quelques **centaines** de variables et contraintes.

Dans le cas où la fonction objectif **n'est pas convexe** :

- Résolution en nombres entiers : de l'ordre de quelques **dizaines** de variables et contraintes (algorithmes d'approximation d'enveloppe convexe).
- Mais il n'y a pas de solveurs commerciaux qui savent le faire, et quasiment pas d'algorithmes efficaces

Performances du B&B

Dans le cas où la fonction objectif est **convexe** :

- Résolution en continue : de l'ordre de quelques **milliers** de variables et contraintes.
- Résolution en nombres entiers : de l'ordre de quelques **centaines** de variables et contraintes.

Dans le cas où la fonction objectif **n'est pas convexe** :

- Résolution en nombres entiers : de l'ordre de quelques **dizaines** de variables et contraintes (algorithmes d'approximation d'enveloppe convexe).
- Mais il n'y a pas de solveurs commerciaux qui savent le faire, et quasiment pas d'algorithmes efficaces

Nous proposons un algorithme pour résoudre ce type de problèmes

La programmation quadratique non convexe en nombres entiers

Même problème sauf que **f(x) n'est pas convexe**

La programmation quadratique non convexe en nombres entiers

Même problème sauf que **$\mathbf{f(x)}$ n'est pas convexe**

i.e. Q n'est plus semi-définie positive

La programmation quadratique non convexe en nombres entiers

Même problème sauf que **f(x) n'est pas convexe**

i.e. Q n'est plus semi-définie positive

Problème : La résolution en continu est NP-difficile

$\Longrightarrow$ On ne peut pas résoudre (QP) par un algorithme de B&B, car le travail à faire à chaque noeud serait trop important

La programmation quadratique non convexe en nombres entiers

Même problème sauf que **$\mathbf{f(x})$ n'est pas convexe**

i.e. Q n'est plus semi-définie positive

Problème : La résolution en continu est NP-difficile

$\Longrightarrow$ On ne peut pas résoudre (QP) par un algorithme de B&B, car le travail à faire à chaque noeud serait trop important

Idée : transformer (QP) pour pouvoir le résoudre

La programmation quadratique non convexe en nombres entiers

Même problème sauf que **$\mathbf{f(x})$ n'est pas convexe**

i.e. Q n'est plus semi-définie positive

Problème : La résolution en continu est NP-difficile

$\Longrightarrow$ On ne peut pas résoudre (QP) par un algorithme de B&B, car le travail à faire à chaque noeud serait trop important

Idée : transformer (QP) pour pouvoir le résoudre

On choisit ici de le reformuler en un problème convexe, puis d'utiliser un solveur efficace

La convexification IQCR (Integer Quadratic Convex Reformulation) (Billionnet-Elloumi-Lambert -2009)

But : rendre la fonction objectif de ce problème convexe

La convexification IQCR (Integer Quadratic Convex Reformulation) (Billionnet-Elloumi-Lambert -2009)

But : rendre la fonction objectif de ce problème convexe

Méthode : Trouver un certain problème (QP') qui a une fonction objectif convexe et qui a la même valeur optimale que (QP)

La convexification IQCR (Integer Quadratic Convex Reformulation) (Billionnet-Elloumi-Lambert -2009)

But : rendre la fonction objectif de ce problème convexe

Méthode : Trouver un certain problème (QP') qui a une fonction objectif convexe et qui a la même valeur optimale que (QP)

Cela revient à perturber la matrice Q pour la rendre SDP, tout en gardant l'équivalence pour toutes les solutions entières.

Transformation de $f(x)$: nouvelle matrice

Dans (QP) : $f(x) = x^T Qx + c^T x$, $\forall x \in E$ avec Q qui n'est pas SDP.

Transformation de $f(x)$: nouvelle matrice

Dans (QP) : $f(x) = x^T Qx + c^T x$, $\forall x \in E$ avec Q qui n'est pas SDP.

Dans (QP') : soit α et β, on introduit les variables y et on définit $\forall (x, y) \in F$:

$$\begin{aligned} f_{\alpha,\beta}(x,y) &= x^T Qx + c^T x + \alpha(Ax - b)^2 + \beta(x^T x - y) \\ &= x^T Q_{\alpha,\beta} x + c^T_{\alpha,\beta} x + b^T b - \langle \beta, y \rangle \end{aligned}$$

(notation : $\forall A, B \in \mathbf{S}_n$, $\langle A, B \rangle = \sum_{i=1}^{n} \sum_{i=1}^{n} a_{ij} b_{ij}$)

Transformation de $f(x)$: nouvelle matrice

Dans (QP) : $f(x) = x^T Q x + c^T x$, $\forall x \in E$ avec Q qui n'est pas SDP.

Dans (QP') : soit α et β, on introduit les variables y et on définit $\forall (x, y) \in F$:

$$\begin{aligned} f_{\alpha,\beta}(x,y) &= x^T Q x + c^T x + \alpha(Ax - b)^2 + \beta(x^T x - y) \\ &= x^T Q_{\alpha,\beta} x + c^T_{\alpha,\beta} x + b^T b - \langle \beta, y \rangle \end{aligned}$$

(notation : $\forall A, B \in \mathbf{S}_n$, $\langle A, B \rangle = \sum_{i=1}^{n} \sum_{i=1}^{n} a_{ij} b_{ij}$)

On a une nouvelle matrice $Q_{\alpha,\beta}$ qui dépend de α et β et on peut choisir α et β tels que $Q_{\alpha,\beta}$ soit SDP.

Transformation de $f(x)$: équivalence des solutions

On a :

1. $\forall x \in E : f(x) = x^T Q x + c^T x$
2. $\forall (x, y) \in F : f_{\alpha,\beta}(x, y) = x^T Q x + c^T x + \alpha(Ax - b)^2 + \beta(x^T x - y)$

Transformation de $f(x)$: équivalence des solutions

On a :

1. $\forall x \in E : f(x) = x^T Q x + c^T x$
2. $\forall (x, y) \in F : f_{\alpha,\beta}(x, y) = x^T Q x + c^T x + \alpha(Ax - b)^2 + \beta(x^T x - y)$

On veut $\forall (x, y) \in F$, $f_{\alpha,\beta}(x, y) = f(x)$ $\forall x \in E$:

1. $(Ax - b) = 0$: toujours vrai

Transformation de $f(x)$: équivalence des solutions

On a :

1. $\forall x \in E : f(x) = x^T Q x + c^T x$
2. $\forall (x, y) \in F : f_{\alpha,\beta}(x, y) = x^T Q x + c^T x + \alpha(Ax - b)^2 + \beta(x^T x - y)$

On veut $\forall (x, y) \in F$, $f_{\alpha,\beta}(x, y) = f(x)$ $\forall x \in E$:

1. $(Ax - b) = 0$: toujours vrai
2. $(x^T x - y) = 0$: vrai si chaque produit $x_i x_j = y_{ij}$

La contrainte $x_i x_j = y_{ij}$ n'est pas convexe. Nous forçons l'égalité $x_i x_j = y_{ij}$ à l'aide d'un ensemble d'inégalités linéaires et de nouvelles variables (noté S).

Transformation de $f(x)$: équivalence des solutions

On a :

1. $\forall x \in E : f(x) = x^T Q x + c^T x$
2. $\forall (x, y) \in F : f_{\alpha,\beta}(x, y) = x^T Q x + c^T x + \alpha(Ax - b)^2 + \beta(x^T x - y)$

On veut $\forall (x, y) \in F$, $f_{\alpha,\beta}(x, y) = f(x)$ $\forall x \in E$:

1. $(Ax - b) = 0$: toujours vrai
2. $(x^T x - y) = 0$: vrai si chaque produit $x_i x_j = y_{ij}$

La contrainte $x_i x_j = y_{ij}$ n'est pas convexe. Nous forçons l'égalité $x_i x_j = y_{ij}$ à l'aide d'un ensemble d'inégalités linéaires et de nouvelles variables (noté S).

$f(x)$ et $f_{\alpha,\beta}(x, y)$ sont équivalentes sur leurs espaces de contraintes respectifs (E et F).

Transformation de $f(x)$: équivalence des solutions

On a :

1. $\forall x \in E : f(x) = x^T Q x + c^T x$
2. $\forall (x, y) \in F : f_{\alpha,\beta}(x, y) = x^T Q x + c^T x + \alpha(Ax - b)^2 + \beta(x^T x - y)$

On veut $\forall (x, y) \in F$, $f_{\alpha,\beta}(x, y) = f(x)$ $\forall x \in E$:

1. $(Ax - b) = 0$: toujours vrai
2. $(x^T x - y) = 0$: vrai si chaque produit $x_i x_j = y_{ij}$

La contrainte $x_i x_j = y_{ij}$ n'est pas convexe. Nous forçons l'égalité $x_i x_j = y_{ij}$ à l'aide d'un ensemble d'inégalités linéaires et de nouvelles variables (noté S).

$f(x)$ et $f_{\alpha,\beta}(x, y)$ sont équivalentes sur leurs espaces de contraintes respectifs (E et F).

On obtient donc un programme quadratique et convexe.

Nouveau problème quadratique convexe et équivalent à (QP)

$$(QP') \begin{cases} Min & f_{\alpha,\beta}(x,y) = x^T Q_{\alpha,\beta} x + c_{\alpha,\beta}^T x + b^T b - \langle \beta, y \rangle \\ s.c. & Ax = b \\ & Dx \leq e \\ & 0 \leq x \leq u \\ & x \text{ entier} \\ & (x,y) \in S \ (\text{ i.e. } y_{ij} = x_i x_j \ ) \end{cases}$$

Nouveau problème quadratique convexe et équivalent à (QP)

$$(QP') \begin{cases} Min & f_{\alpha,\beta}(x,y) = x^T Q_{\alpha,\beta} x + c_{\alpha,\beta}^T x + b^T b - \langle \beta, y \rangle \\ s.c. & Ax = b \\ & Dx \leq e \\ & 0 \leq x \leq u \\ & x \text{ entier} \\ & (x,y) \in S \ (\text{ i.e. } y_{ij} = x_i x_j \ ) \end{cases}$$

Calculer α et β tels que :

1. $Q_{\alpha,\beta}$ soit SDP
2. la solution optimale continue de (QP') soit aussi proche que possible de sa solution optimale entière.

Notre problème maintenant

$$(CP) = \max_{\alpha,\beta,Q_{\alpha,\beta} \succeq 0} \min_{(x,y)\in F} v(\overline{QP}')$$

Notre problème maintenant

$(CP) = \max_{\alpha,\beta,Q_{\alpha,\beta}\succeq 0}$	$\min_{(x,y)\in F} v(\overline{QP}')$
Maximiser sur α et β tel que notre nouvelle matrice $Q_{\alpha,\beta}$ soit SDP	la valeur de la relaxation continue du problème reformulé

Une relaxation SDP de (QP)

Soit $(PSDP)$ une relaxation SDP de (QP) :

$$(PSDP)\begin{cases} Min & f(x,X) = \langle Q, X\rangle + c^T x \\ s.c. & Ax = b \\ & Dx \leq e \\ & (x,X) \in S \\ & \langle A^T A, X\rangle - (2A^T b)^T x = -b^T b \\ & \begin{pmatrix} 1 & x \\ x^T & X \end{pmatrix} \succeq 0 \\ & X \succeq 0, x \in \mathrm{R} \end{cases}$$

Une relaxation SDP de (QP)

Soit $(PSDP)$ une relaxation SDP de (QP) :

$$(PSDP)\begin{cases} Min & f(x,X) = \langle Q, X\rangle + c^T x \\ s.c. & Ax = b \\ & Dx \leq e \\ & (x,X) \in S \\ & \langle A^T A, X\rangle - (2A^T b)^T x = -b^T b \\ & \begin{pmatrix} 1 & x \\ x^T & X \end{pmatrix} \succeq 0 \\ & X \succeq 0, x \in \mathrm{R} \end{cases}$$

1. On a introduit une variable matricielle X représentant l'ensemble des produits $x^T x$, (ou de façon équivalente les variables y).

Une relaxation SDP de (QP)

Soit $(PSDP)$ une relaxation SDP de (QP) :

$$(PSDP)\begin{cases} Min & f(x,X) = \langle Q, X\rangle + c^T x \\ s.c. & Ax = b \\ & Dx \leq e \\ & (x,X) \in S \\ & \langle A^T A, X\rangle - (2A^T b)^T x = -b^T b \\ & \begin{pmatrix} 1 & x \\ x^T & X \end{pmatrix} \succeq 0 \\ & X \succeq 0, x \in \mathrm{R} \end{cases}$$

1. On a introduit une variable matricielle X représentant l'ensemble des produits $x^T x$, (ou de façon équivalente les variables y).

2. $\begin{pmatrix} 1 & x \\ x^T & X \end{pmatrix} \succeq 0 \Longleftrightarrow X - xx^T \succeq 0$ (relaxation de $X = xx^T$ qui n'est pas convexe).

Calcul du dual de (*PSDP*)

$$(PSDP)\begin{cases} Min & f(x,X) = \langle Q, X\rangle + c^T x & \\ s.c. & Ax = b & \leftarrow \rho \\ & Dx \leq e & \leftarrow \sigma \\ & (x,X) \in S & \leftarrow \beta \\ & \langle A^T A, X\rangle - (2A^T b)^T x = -b^T b & \leftarrow \alpha \\ & \begin{pmatrix} 1 & x \\ x^T & X \end{pmatrix} \succeq 0 & \\ & X \succeq 0, x \in \mathrm{R} & \end{cases}$$

Son dual

$$(DSDP)\begin{cases} Max & g(\rho,\sigma,\beta,\alpha) = -\rho^T b - \sigma^T e + h_1(\beta) + \alpha b^T b \\ s.c. & Q + \alpha A^T A + \beta \succeq 0 \\ & c - 2\alpha A^T b + A^T \rho + D^T \sigma + h_2(\beta) \geq 0 \\ & \rho \in \mathrm{R}^m, \sigma \in \mathrm{R}^p, \beta \in \mathbf{S}_n, \alpha \in \mathrm{R} \end{cases}$$

Son dual

$$(DSDP)\begin{cases} Max & g(\rho,\sigma,\beta,\alpha) = -\rho^T b - \sigma^T e + h_1(\beta) + \alpha b^T b \\ s.c. & Q + \alpha A^T A + \beta \succeq 0 \\ & c - 2\alpha A^T b + A^T \rho + D^T \sigma + h_2(\beta) \geq 0 \\ & \rho \in \mathrm{R}^m, \sigma \in \mathrm{R}^p, \beta \in \mathbf{S}_n, \alpha \in \mathrm{R} \end{cases}$$

$$(DSDP)\begin{cases} Max & g(\rho,\sigma,\beta,\alpha) \\ s.c. & Q_{\alpha,\beta} \succeq 0 \\ & c - 2\alpha A^T b + A^T \rho + D^T \sigma + h_2(\beta) \geq 0 \\ & \rho \in \mathrm{R}^m, \sigma \in \mathrm{R}^p, \beta \in \mathbf{S}_n, \alpha \in \mathrm{R} \end{cases}$$

Calcul des paramètres α et β par la programmation semie définie

$$(DSDP)\begin{cases} Max & g(\rho,\sigma,\beta,\alpha) = -\rho^T b - \sigma^T e + h_1(\beta) + \alpha b^T b \\ s.c. & Q + \alpha A^T A + \beta \succeq 0 \\ & c - 2\alpha A^T b + A^T \rho + D^T \sigma + h_2(\beta) \geq 0 \\ & \rho \in \mathrm{R}^m, \sigma \in \mathrm{R}^p, \beta \in \mathbf{S}_n, \alpha \in \mathrm{R} \end{cases}$$

Calcul des paramètres α et β par la programmation semie définie

$$(DSDP)\begin{cases} Max & g(\rho,\sigma,\beta,\alpha) = -\rho^T b - \sigma^T e + h_1(\beta) + \alpha b^T b \\ s.c. & Q + \alpha A^T A + \beta \succeq 0 \\ & c - 2\alpha A^T b + A^T \rho + D^T \sigma + h_2(\beta) \geq 0 \\ & \rho \in \mathrm{R}^m, \sigma \in \mathrm{R}^p, \beta \in \mathbf{S}_n, \alpha \in \mathrm{R} \end{cases}$$

1. Les solutions optimales de $(DSDP)$, α^* et β^* rendent notre matrice Q_{α^*,β^*} semi définie positive.

Calcul des paramètres α et β par la programmation semie définie

$$(DSDP)\begin{cases} Max & g(\rho,\sigma,\beta,\alpha) = -\rho^T b - \sigma^T e + h_1(\beta) + \alpha b^T b \\ s.c. & Q + \alpha A^T A + \beta \succeq 0 \\ & c - 2\alpha A^T b + A^T \rho + D^T \sigma + h_2(\beta) \geq 0 \\ & \rho \in \mathrm{R}^m, \sigma \in \mathrm{R}^p, \beta \in \mathbf{S}_n, \alpha \in \mathrm{R} \end{cases}$$

1. Les solutions optimales de $(DSDP)$, α^* et β^* rendent notre matrice Q_{α^*,β^*} semi définie positive.

2. Ces α^* et β^* sont les meilleurs dans notre schéma pour que la valeur de la solution continue de notre programme reformulé soit la plus proche possible de la valeur de sa solution entière. (Dualité lagrangienne)

Différence entre le 0-1 et l'entier :

Linéarisation des produits $x_i x_j$ non triviale pour l'entier.

Différence entre le 0-1 et l'entier :

Linéarisation des produits $x_i x_j$ non triviale pour l'entier.

Solution 1 :

1. Décomposer en puissance de 2 chaque variable entière.
2. Appliquer une méthode existante : QCR au nouveau problème 0-1
   $\Longrightarrow$ **Problème reformulé trop grand**

Différence entre le 0-1 et l'entier :

Linéarisation des produits $x_i x_j$ non triviale pour l'entier.

Solution 1 :

1. Décomposer en puissance de 2 chaque variable entière.
2. Appliquer une méthode existante : `QCR` au nouveau problème 0-1
   $\Longrightarrow$ **Problème reformulé trop grand**

Solution 2 : `IQCR`

1. Calcul des coefficients de convexification dans l'espace des variables entières x.
2. Linéariser de façon non triviale.
   $\Longrightarrow$ **Beaucoup plus efficace à la fois en terme de taille et de borne**

Linéarisation de $y_{ij} = x_i x_j$: BIL (Binary Integer Linearization) (Billionnet, Elloumi, Lambert - 2008)

1. Décompostion binaire de chaque variable x_j

$$x_j = \sum_{k=0}^{\lfloor log(u_j) \rfloor} 2^k t_{jk} \text{ avec } t_{jk} \in \{0,1\}$$

Linéarisation de $y_{ij} = x_i x_j$: BIL (Binary Integer Linearization) (Billionnet, Elloumi, Lambert - 2008)

1. Décompostion binaire de chaque variable x_j

$$x_j = \sum_{k=0}^{\lfloor log(u_j) \rfloor} 2^k t_{jk} \text{ avec } t_{jk} \in \{0,1\}$$

2. $x_i x_j = x_i(\sum_{k=0}^{\lfloor log(u_j) \rfloor} 2^k t_{jk}) = \sum_{k=0}^{\lfloor log(u_j) \rfloor} 2^k x_i t_{jk}$,

Linéarisation de $y_{ij} = x_i x_j$: BIL (Binary Integer Linearization) (Billionnet, Elloumi, Lambert - 2008)

1. Décompostion binaire de chaque variable x_j

$$x_j = \sum_{k=0}^{\lfloor log(u_j) \rfloor} 2^k t_{jk} \text{ avec } t_{jk} \in \{0,1\}$$

2. $x_i x_j = x_i(\sum_{k=0}^{\lfloor log(u_j) \rfloor} 2^k t_{jk}) = \sum_{k=0}^{\lfloor log(u_j) \rfloor} 2^k x_i t_{jk}$,

3. Linéarisation : $z_{ijk} = x_i t_{jk}$.

$$\begin{cases} z_{ijk} \leq t_{jk} u_i \\ z_{ijk} \leq x_i \\ z_{ijk} \geq x_i - u_i(1 - t_{jk}) \\ z_{ijk} \geq 0 \end{cases} \iff x_i t_{jk} = z_{ijk}$$

Une extension de IQCR

Extension de IQCR aux programmes quadratiques en variables mixtes entières : MIQCR (Mixed Integer Quadratic Convex Reformulation)

- **Difficulté :** On ne sait pas linéariser le produit de deux variables réelles.

Une extension de IQCR

Extension de IQCR aux programmes quadratiques en variables mixtes entières : MIQCR (Mixed Integer Quadratic Convex Reformulation)

- **Difficulté :** On ne sait pas linéariser le produit de deux variables réelles.
- **Hypothèse :** La sous fonction des produits de variables réelles est convexe.

Une extension de IQCR

Extension de IQCR aux programmes quadratiques en variables mixtes entières : MIQCR (Mixed Integer Quadratic Convex Reformulation)

- **Difficulté :** On ne sait pas linéariser le produit de deux variables réelles.
- **Hypothèse :** La sous fonction des produits de variables réelles est convexe.
- **Méthode :** La perturbation $\beta_{ij}(x_i x_j - y_{ij})$ n'est pas considérée si x_i **et** x_j sont **réels**.

Conséquence de MIQCR

Amélioration : Perturbation de $f(x)$ avec les contraintes d'inégalité

1. Transformer les p contraintes d'inéglité en contraintes d'égalité en utilisant p variables d'écart réelles.
2. Appliquer MIQCR

$\Longrightarrow$ Perturbation de la fonction objectif par les contraintes d'inégalité.

Conclusion

- IQCR : méthode de résolution **exacte** de **programmes quadratiques en variables entières** soumis à des contraintes linéaires.

Conclusion

- IQCR : méthode de résolution **exacte** de **programmes quadratiques en variables entières** soumis à des contraintes linéaires.
- IQCR peut être étendue **efficacement** aux problèmes en variables **mixtes entières** (MIQCR).

Conclusion

- IQCR : méthode de résolution **exacte** de **programmes quadratiques en variables entières** soumis à des contraintes linéaires.
- IQCR peut être étendue **efficacement** aux problèmes en variables **mixtes entières** (MIQCR).
- Pour la **programmation 0-1**, **IQCR améliore** les convexifications existantes :

Conclusion

- IQCR : méthode de résolution **exacte** de **programmes quadratiques en variables entières** soumis à des contraintes linéaires.
- IQCR peut être étendue **efficacement** aux problèmes en variables **mixtes entières** (MIQCR).
- Pour la **programmation 0-1**, **IQCR améliore** les convexifications existantes :
  i) **meilleure borne**

Conclusion

- IQCR : méthode de résolution **exacte** de **programmes quadratiques en variables entières** soumis à des contraintes linéaires.
- IQCR peut être étendue **efficacement** aux problèmes en variables **mixtes entières** (MIQCR).
- Pour la **programmation 0-1**, **IQCR améliore** les convexifications existantes :
  i) **meilleure borne**
  ii) **pertubation** de la fonction objectif avec les **contraintes d'inégalité**

Conclusion

- `IQCR` : méthode de résolution **exacte** de **programmes quadratiques en variables entières** soumis à des contraintes linéaires.
- `IQCR` peut être étendue **efficacement** aux problèmes en variables **mixtes entières** (`MIQCR`).
- Pour la **programmation 0-1**, **IQCR améliore** les convexifications existantes :
  i) **meilleure borne**
  ii) **pertubation** de la fonction objectif avec les **contraintes d'inégalité**
  iii) **extension au mixte 0-1**

Conclusion

- IQCR : méthode de résolution **exacte** de **programmes quadratiques en variables entières** soumis à des contraintes linéaires.
- IQCR peut être étendue **efficacement** aux problèmes en variables **mixtes entières** (MIQCR).
- Pour la **programmation 0-1**, **IQCR améliore** les convexifications existantes :
  i) **meilleure borne**
  ii) **pertubation** de la fonction objectif avec les **contraintes d'inégalité**
  iii) **extension au mixte 0-1**
- On arrive à résoudre des problèmes de plusieurs dizaines de contraintes et variables (*IIQP*, taille 40, bornes supérieures à 70, gap moyen : 0.1%, tps moyen : 713 s)

Conclusion

- `IQCR` : méthode de résolution **exacte** de **programmes quadratiques en variables entières** soumis à des contraintes linéaires.
- `IQCR` peut être étendue **efficacement** aux problèmes en variables **mixtes entières** (`MIQCR`).
- Pour la **programmation 0-1**, **IQCR améliore** les convexifications existantes :
  i) **meilleure borne**
  ii) **pertubation** de la fonction objectif avec les **contraintes d'inégalité**
  iii) **extension au mixte 0-1**
- On arrive à résoudre des problèmes de plusieurs dizaines de contraintes et variables (*IIQP*, taille 40, bornes supérieures à 70, gap moyen : 0.1%, tps moyen : 713 s)
- Logiciel `SIQP` : `http ://cedric.cnam.fr/siqp/siqp.php`

Exemple

4 variables, 1 contrainte d'égalité et 1 contrainte d'inégalité :

$$(QP_e)\begin{cases} Min \quad f(x) = x^T \begin{pmatrix} 5 & -7 & -6 & -1 \\ -7 & 3 & -8 & -18 \\ -6 & -8 & -17 & 10 \\ -1 & -18 & 10 & 3 \end{pmatrix} x + \begin{pmatrix} -5 \\ -11 \\ 4 \\ 1 \end{pmatrix}^T x \\ s.c. \quad 3x_1 + 19x_2 + 18x_3 + 11x_4 = 255 \\ \qquad 11x_1 + 13x_2 + 8x_3 + x_4 \leq 165 \\ \qquad 0 \leq x_i \leq 10 \qquad\qquad i \in \{1, \ldots, 4\} \\ \qquad x_i \text{ entier} \qquad\qquad i \in \{1, \ldots, 4\} \end{cases}$$

valeur optimale : -2552.

Exemple (suite) : IQCR

Avec la méthode IQCR, nous perturbons la matrice Q de la manière suivante :

$$\begin{pmatrix} 5+\beta_{11}+9\alpha & -7+\beta_{12}+57\alpha & -6+\beta_{13}+54\alpha & -1+\beta_{14}+33\alpha \\ -7+\beta_{12}+57\alpha & 3+\beta_{22}+361\alpha & -8+\beta_{23}+342\alpha & -18+\beta_{24}+209\alpha \\ -6+\beta_{13}+54\alpha & -8+\beta_{23}+342\alpha & -17+\beta_{33}+324\alpha & 10+\beta_{34}+198\alpha \\ -1+\beta_{14}+33\alpha & -18+\beta_{24}+209\alpha & 10+\beta_{34}+198\alpha & 3+\beta_{44}+121\alpha \end{pmatrix}$$

Avec $\alpha = 2090.76$, et $\beta_{11} = \beta_{12} = \beta_{13} = \beta_{14} = \beta_{22} = \beta_{23} = \beta_{24} = 0$, $\beta_{33} = 24.45$, $\beta_{34} = -6.80$, $\beta_{44} = 4.92$.

Exemple (suite) : IQCR

optimum	-2552

	IQCR
nb var	100
nb cont	246
tps (s)	0.004
relax cont	-2808.77
gap	10.06 %

Exemple (suite)

4 variables entières, 1 variable réelle, et 2 contraintes d'égalité.

$$
(QP'_e)\begin{cases}
Min \quad f(x) = x^T \begin{pmatrix} 5 & -7 & -6 & -1 & 0 \\ -7 & 3 & -8 & -18 & 0 \\ -6 & -8 & -17 & 10 & 0 \\ -1 & -18 & 10 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} x + \begin{pmatrix} -5 \\ -11 \\ 4 \\ 1 \\ 0 \end{pmatrix}^T x \\
s.t. \quad 3x_1 + 19x_2 + 18x_3 + 11x_4 = 255 \\
\qquad 11x_1 + 13x_2 + 8x_3 + x_4 + s = 165 \\
\qquad 0 \leq x_i \leq 10 \\
\qquad 0 \leq s \leq 165 \\
\qquad x_i \text{ integer} \\
\qquad s \text{ réel}
\end{cases}
$$

Exemple (suite) : MIQCR

Avec la méthode MIQCR, nous perturbons la matrice Q de la manière suivante :

$$\begin{pmatrix} 5+\beta_{11}+9\alpha+121\alpha' & -7+\beta_{12}+57\alpha+143\alpha' & \dots & \beta_{15}+11\alpha' \\ -7+\beta_{12}+57\alpha+143\alpha' & 3+\beta_{22}+361\alpha+169\alpha' & \dots & \beta_{25}+13\alpha' \\ -6+\beta_{13}+54\alpha+88\alpha' & -8+\beta_{23}+342\alpha+104\alpha' & \dots & \beta_{35}+8\alpha' \\ -1+\beta_{14}+33\alpha+11\alpha' & -18+\beta_{24}+209\alpha+13\alpha' & \dots & \beta_{45}+\alpha' \\ \beta_{15}+11\alpha' & \beta_{25}+13\alpha' & \dots & \alpha' \end{pmatrix}$$

Avec $\alpha = 116.91$, $\alpha' = 249.74$, et $\beta_{11} = \beta_{12} = 0$, $\beta_{13} = 0.04$, $\beta_{14} = -0.04$, $\beta_{14} = -0.38$, $\beta_{23} = -0.09$, $\beta_{24} = 0.09$, $\beta_{25} = 0.44$, $\beta_{33} = 27.23$, $\beta_{34} = -4.72$, $\beta_{35} = -2.18$, $\beta_{44} = 2.60$, $\beta_{45} = 1.63$.

Exemple (suite) : IQCR et MIQCR

optimum	-2552	
	IQCR	MIQCR
nb var	100	121
nb cont	246	302
tps (s)	0.004	0.04
relax cont	-2808.77	-2776.07
gap	10.06 %	8.9%