Résolution de programmes quadratiques en nombres entiers

Amélie Lambert

CEDRIC-ENSIIE

16 février 2010

1. La programmation quadratique convexe
   - Présentation du problème
   - Résolution d'un programme quadratique convexe en nombres entiers
2. La programmation quadratique non convexe
   - Présentation du problème
   - Reformuler (QP) en un programme convexe
   - Rappels sur la programmation semi-définie
   - Calcul de la meilleure convexification par la programmation semie définie
3. Conclusion

Formulation générale

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

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 :

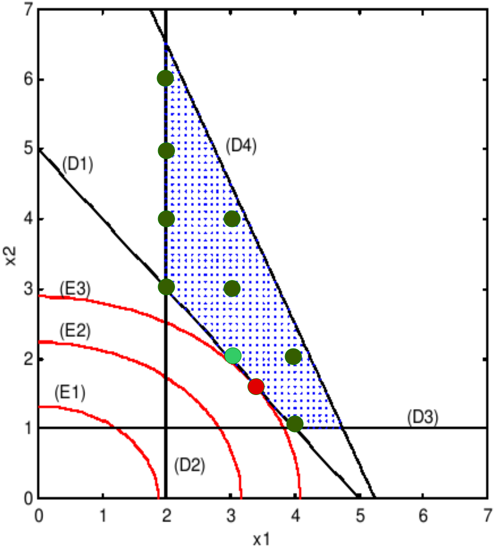
$$
(QP_e) \begin{cases} Min & f(x_1, x_2) = 2x_1^2 + 4x_2^2 \\ s.c & x_1 + x_2 \leq 5 \qquad (D1) \\ & x_1 \geq 2 \qquad (D2) \\ & x_2 \geq 1 \qquad (D3) \\ & 2x_1 + x_2 \leq 10.5 \quad (D4) \\ & x_1, x_2 \quad entier \end{cases}
$$

Trouver le minimum pour la fonction convexe $f(x_1, x_2)$.

A l'intérieur d'un espace définit par des droites $(D1)$, $(D2)$, $(D3)$ et $(D4)$.

Avec deux variables x_1 et x_2 bornées et entières.

Exemple (suite) : Représentation graphique



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

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

Exemple (suite)

Sur ce dessin on voit bien une solution se distinguer

Mais ce n'est pas une solution entière

Pour trouver une solution entière, on se base sur la solution optimale continue, ici $x_1^* = 10/3$, $x_2^* = 5/3$ et $f(x_1^*, x_2^*) = 100/3$.

Et on parcourt un arbre des possibilité des solutions entières à partir de cette solution

Résoudre un programme quadratique convexe en nombres entiers

Résumé : 2 étapes de résolution (le Branch and Bound (B&B))

1. Résoudre le problème en continu (problème polynômial)

Résoudre un programme quadratique convexe en nombres entiers

Résumé : 2 étapes de résolution (le Branch and Bound (B&B))

1. Résoudre le problème en continu (problème polynômial)
2. Parcourir un arbre des solutions entières jusqu'à obtenir une solution optimale entière, à chaque noeud une borne inférieure est donnée par la résolution du problème en continu.

Résoudre un programme quadratique convexe en nombres entiers

Résumé : 2 étapes de résolution (le Branch and Bound (B&B))

1. Résoudre le problème en continu (problème polynômial)
2. Parcourir un arbre des solutions entières jusqu'à obtenir une solution optimale entière, à chaque noeud une borne inférieure est donnée par la résolution du problème en continu.

Donc à chaque noeud on résout un problème polynômial, c'est la taille de l'arbre qui est exponentielle.

Résoudre un programme quadratique convexe en nombres entiers

Résumé : 2 étapes de résolution (le Branch and Bound (B&B))

1. Résoudre le problème en continu (problème polynômial)
2. Parcourir un arbre des solutions entières jusqu'à obtenir une solution optimale entière, à chaque noeud une borne inférieure est donnée par la résolution du problème en continu.

Donc à chaque noeud on résout un problème polynômial, c'est la taille de l'arbre qui est exponentielle.

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

Résoudre un programme quadratique convexe en nombres entiers

Résumé : 2 étapes de résolution (le Branch and Bound (B&B))

1. Résoudre le problème en continu (problème polynômial)
2. Parcourir un arbre des solutions entières jusqu'à obtenir une solution optimale entière, à chaque noeud une borne inférieure est donnée par la résolution du problème en continu.

Donc à chaque noeud on résout un problème polynômial, c'est la taille de l'arbre qui est exponentielle.

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

Cet algorithme est efficace si la solution continue n'est pas trop éloignée de la solution entière.

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

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
- Mais il n'y a pas de logiciels 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
- Mais il n'y a pas de logiciels qui savent le faire, et quasiment pas d'algorithmes efficaces

On propose un algorithme pour résoudre ce type de problèmes

La programmation quadratique non convexe en nombres entiers

Même problème sauf que $\mathbf{f(x)}$ **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 : On ne sait pas trouver le minimum global d'une fonction non convexe en un temps raisonnable

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 : On ne sait pas trouver le minimum global d'une fonction non convexe en un temps raisonnable

La résolution d'un problème non convexe en continu est NP-difficile

Notre problème

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

Notre problème

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

Notre problème

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 la méthode de résolution présentée avant

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 équivalence des solutions si les variables sont 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 y, α, β 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) : $f(x) = x^T Qx + c^T x$, $\forall x \in E$ avec Q qui n'est pas SDP.

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

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

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

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

On a :

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

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

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

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

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, donc 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)

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

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.

Utilisation de la programmation semi-définie

La programmation semi-définie

La programmation semi-définie (SDP) est une extension de la programmation linéaire (LP).

La programmation semi-définie

La programmation semi-définie (SDP) est une extension de la programmation linéaire (LP).

La SDP a été étudiée pour la première fois dans les années 60.

La programmation semi-définie

La programmation semi-définie (SDP) est une extension de la programmation linéaire (LP).

La SDP a été étudiée pour la première fois dans les années 60.

Un problème SDP est un programme dont l'objectif est linéaire en les termes de la matrice variable X et dont les contraintes sont également linéaires en les termes de cette matrice.

Le problème primal SDP

Soit $Q, A^1, \ldots, A^m \in \mathbf{S_n}$, et $b \in \mathrm{R}^m$

Le problème primal SDP

Soit $Q, A^1, \ldots, A^m \in \mathbf{S_n}$, et $b \in \mathbb{R}^m$

Le primal $(PSDP)$ s'écrit sous la forme :

$$(PSDP) \left\{ \begin{array}{llll} Max & \langle Q, X \rangle & & \\ s.c. & \langle A^1, X \rangle & = & b_1 \\ & \vdots & & \vdots \\ & \langle A^m, X \rangle & = & b_m \\ & X \in \mathbf{S_n} & & \\ & X \succeq 0 & & \end{array} \right.$$

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

Le problème dual SDP

En introduisant $\lambda \in \mathrm{R}^m$, les variables duales de $(PSDP)$
Le dual de $(PSDP)$ s'écrit sous la forme suivante :

$$(DSDP) \left\{ \begin{array}{ll} Min & \langle b, \lambda \rangle \\ s.c. & \displaystyle\sum_{k=1}^{m} \lambda_k A^k - Q \succeq 0 \\ & \lambda \in \mathrm{R}^m \end{array} \right.$$

Le problème dual SDP

En introduisant $\lambda \in \mathrm{R}^m$, les variables duales de $(PSDP)$
Le dual de $(PSDP)$ s'écrit sous la forme suivante :

$$(DSDP) \begin{cases} Min & \langle b, \lambda \rangle \\ s.c. & \displaystyle\sum_{k=1}^{m} \lambda_k A^k - Q \succeq 0 \\ & \lambda \in \mathrm{R}^m \end{cases}$$

Notons que $\displaystyle\sum_{k=1}^{m} \lambda_k A^k - Q \in \mathbf{S_n}$

On voit apparaître la dualité comme en (LP).

$$(PSDP)\left\{\begin{array}{llll} Max & \langle Q, X\rangle & & \\ s.c. & \langle A^1, X\rangle & = & b_1 \\ & \vdots & & \vdots \\ & \langle A^m, X\rangle & = & b_m \\ & X \in \mathbf{S_n} & & \\ & X \succeq 0 & & \end{array}\right.$$

$$(DSDP)\left\{\begin{array}{ll} Min & \langle b, \lambda\rangle \\ s.c. & \sum_{k=1}^{m} \lambda_k A^k - Q \succeq 0 \\ & \lambda \in \mathrm{R}^m \end{array}\right.$$

Relation avec le Lagrangien

Comme en programmation linéaire, $(DSPD)$ provient du dual Lagrangien, en effet, $(PSDP)$ est :

$$\max_{X}\{\langle Q, X\rangle : \langle A^k, X\rangle = b_k, k = 1, \ldots, m\,,\, X \succeq 0\}$$

Relation avec le Lagrangien

Comme en programmation linéaire, $(DSPD)$ provient du dual Lagrangien, en effet, $(PSDP)$ est :

$$\max_X \{\langle Q, X\rangle : \langle A^k, X\rangle = b_k, k = 1, \ldots, m\,,\, X \succeq 0\}$$

Son Lagrangien est :

$$\mathcal{L}(X, \lambda) = \langle Q, X\rangle - \sum_{k=1}^{m}(\langle A^k, X\rangle - b_k)\lambda_k$$

$$\mathcal{L}(X, \lambda) = \langle b, \lambda\rangle - \langle \sum_{k=1}^{m}\lambda_k A^k - Q, X\rangle$$

Dont la fonction duale est :

$$L(\lambda) = \max_{X \succeq 0} \mathcal{L}(X, \lambda) = \langle b, \lambda \rangle - \langle \sum_{k=1}^{m} \lambda_k A^k - Q, X \rangle$$

Dont la fonction duale est :

$$L(\lambda) = \max_{X \succeq 0} \mathcal{L}(X, \lambda) = \langle b, \lambda \rangle - \langle \sum_{k=1}^{m} \lambda_k A^k - Q, X \rangle$$

et son dual Lagrangien est :

$$(DL) = \min_{\lambda \in \mathrm{R}^m} \max_{X \succeq 0} \mathcal{L}(X, \lambda) = \langle b, \lambda \rangle - \langle \sum_{k=1}^{m} \lambda_k A^k - Q, X \rangle$$

Or il est connu que :

$$\max_{X \succeq 0} \mathcal{L}(X, \lambda) = \begin{cases} \langle b, \lambda \rangle \text{ ssi } (\sum_{k=1}^{m} \lambda_k A^k - Q) \succeq 0 \\ \infty \text{ sinon} \end{cases}$$

Or il est connu que :

$$\max_{X \succeq 0} \mathcal{L}(X, \lambda) = \begin{cases} \langle b, \lambda \rangle \text{ ssi } (\sum_{k=1}^{m} \lambda_k A^k - Q) \succeq 0 \\ \infty \text{ sinon} \end{cases}$$

donc :

$$(DL) = \min_{\lambda \in \mathrm{R}^m} \{\langle b, \lambda \rangle : (\sum_{k=1}^{m} \lambda_k A^k - Q) \succeq 0\}$$

qui est exactement $(DSDP)$

Admissibilité

primale : Une matrice $X \succeq 0$ est réalisable pour $(PSDP)$ ssi :

$$\langle A^k, X \rangle = b_k \;\; k = 1, \ldots, m.$$

Admissibilité

primale : Une matrice $X \succeq 0$ est réalisable pour $(PSDP)$ ssi :

$$\langle A^k, X \rangle = b_k \;\; k = 1, \ldots, m.$$

duale : Le vecteur $\lambda \in \mathrm{R}^m$ est réalisable pour $(DSDP)$ ssi

$$\sum_{k=1}^{m} \lambda_k A^k - Q \succeq 0.$$

Admissibilité stricte

primale : Une matrice $X \succ 0$ est strictement réalisable pour $(PSDP)$ ssi

$$\langle A^k, X \rangle = b_k \;\; k = 1, \ldots, m.$$

Admissibilité stricte

primale : Une matrice $X \succ 0$ est strictement réalisable pour $(PSDP)$ ssi

$$\langle A^k, X \rangle = b_k \;\; k = 1, \ldots, m.$$

duale : Le vecteur $\lambda \in \mathrm{R}^m$ est strictement réalisable pour $(DSDP)$ ssi

$$\sum_{k=1}^{m} \lambda_k A^k - Q \succ 0.$$

Saut de dualité

Etant donné X et λ solutions réalisables de $(PSDP)$ et $(DSDP)$, la différence entre les valeurs des objectifs de $(PSDP)$ et $(DSDP)$ est appelée le saut de dualité. Il est donné par :

$$\langle b, \lambda \rangle - \langle Q, X \rangle$$

Dualité faible et forte

Dualité faible : Etant donné X et λ solutions réalisables de $(PSDP)$ et $(DSDP)$ alors

$$\langle Q, X \rangle \leq \langle b, \lambda \rangle$$

Dualité faible et forte

Dualité faible : Etant donné X et λ solutions réalisables de $(PSDP)$ et $(DSDP)$ alors

$$\langle Q, X\rangle \leq \langle b, \lambda\rangle$$

Dualité forte : Si pour X et λ solutions admissibles de $(PSDP)$ et $(DSDP)$, le saut de dualité vaut 0, alors il y a dualité forte et X et λ sont les valeurs optimales de $(PSDP)$ et $(DSDP)$.

Et on a :

$$\langle b, \lambda\rangle - \langle Q, X\rangle = 0 \Longleftrightarrow \langle X, \sum_{k=1}^{m} \lambda_k A^k - Q\rangle = 0$$

Dualité faible et forte

Dualité faible : Etant donné X et λ solutions réalisables de $(PSDP)$ et $(DSDP)$ alors

$$\langle Q, X\rangle \leq \langle b, \lambda\rangle$$

Dualité forte : Si pour X et λ solutions admissibles de $(PSDP)$ et $(DSDP)$, le saut de dualité vaut 0, alors il y a dualité forte et X et λ sont les valeurs optimales de $(PSDP)$ et $(DSDP)$.

Et on a :

$$\langle b, \lambda\rangle - \langle Q, X\rangle = 0 \Longleftrightarrow \langle X, \sum_{k=1}^{m} \lambda_k A^k - Q\rangle = 0$$

Cependant, il n'y a pas nécessairement dualité forte en SDP.

Contraintes de qualification de Slater

$(PSDP)$ satisfait les conditions de Slater si il existe $X \succ 0$ tel que :

$$\langle A^k, X \rangle = b_k \ , \ k = 1, \ldots, m$$

$(DSDP)$ satisfait les conditions de Slater si il existe $\lambda \in \mathrm{R}^m$ tel que :

$$\sum_{k=1}^{m} \lambda_k A^k - Q \succ 0.$$

Théorème

Soit $p^* =$ la valeur optimale de $(PSDP)$ et d^* la valeur optimale de $(DSDP)$

i) si $(PSDP)$ satisfait les conditions de Slater avec p^* *fini*, alors $p^* = d^*$ et cette valeur est atteinte pour $(DSDP)$

ii) si $(DSDP)$ satisfait les conditions de Slater avec d^* *fini*, alors $p^* = d^*$ et cette valeur est atteinte pour $(PSDP)$

iii) si $(PSDP)$ et $(DSDP)$ satisfont les conditions de Slater alors $p^* = d^*$ et cette valeur est atteinte pour les deux problèmes.

Conditions d'optimalité

Pour les problèmes à dualité forte, nous avons les conditions nécessaires et suffisantes d'optimalité suivantes :

$$\begin{cases} \langle A^k, X\rangle = b_k \; \forall k,\; X \succeq 0 \text{ (admissibilité du primal)} \\ \sum\limits_{k=1}^{m} \lambda_k A^k - Q \succeq 0,\;\; \lambda \in \mathrm{R}^m \text{ (admissibilité du dual)} \\ (\sum\limits_{k=1}^{m} \lambda_k A^k - Q)X = \mathbf{0} \text{ (complémentarité de } X \text{ et } (\sum\limits_{k=1}^{m} \lambda_k A_k - Q)) \end{cases}$$

Principaux logiciels et performances

- CSDP (Brian Borchers -1999)
  Efficace sur la programmation en nombres entiers, mais avec un nombre limité de contraintes et variables (quelques milliers)

- Sbmethod (Helmberg -2000)
  Efficace sur la programmation 0-1 et pour un grand nombre de contraintes (plusieurs milliers)

Calculer α et β tels que :

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.

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.

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

Réécrivons (QP)

$$(QP)\left\{\begin{array}{ll} Min & f(x) = x^T Qx + c^T x \\ s.c. & Ax = b \\ & Dx \leq e \\ & 0 \leq x \leq u \\ & x \text{ entier} \end{array}\right. \iff (QP'')\left\{\begin{array}{ll} Min & f(x) = x^T Qx + c^T x \\ s.c. & Ax = b \\ & Dx \leq e \\ & 0 \leq x \leq u \\ & x \text{ entier} \\ & (x, y) \in S \ (\text{ i.e. } y_{ij} = x_i x_j \ ) \\ & (Ax - b)^2 = 0 \end{array}\right.$$

$$\iff (QP'')\left\{\begin{array}{ll} Min & f(x) = \langle Q, y\rangle + c^T x \\ s.c. & Ax = b \\ & Dx \leq e \\ & 0 \leq x \leq u \\ & x \text{ entier} \\ & (x, y) \in S \ (\text{ i.e. } y_{ij} = x_i x_j \ ) \\ & \langle A^T A, y\rangle - (2A^T b)^T x = -b^T b \end{array}\right.$$

Une relaxation SDP de (QP)

On construit (SDP) en introduisant une variable matricielle X représentant l'ensemble des produits $x^T x$, (ou de façon équivalente les variables y), après avoir relâché les contraintes d'intégrité de (QP'') :

Une relaxation SDP de (QP)

On construit (SDP) en introduisant une variable matricielle X représentant l'ensemble des produits $x^T x$, (ou de façon équivalente les variables y), après avoir relâché les contraintes d'intégrité de (QP'') :

$$(\overline{QP}'') \left\{ \begin{array}{ll} Min & f(x) = \langle Q, y \rangle + c^T x \\ s.c. & Ax = b \\ & Dx \leq e \\ & 0 \leq x \leq u \\ & x \text{ réel} \\ & (x, y) \in S \ (\text{ i.e. } y_{ij} = x_i x_j \ ) \\ & \langle A^T A, y \rangle - (2A^T b)^T x = -b^T b \end{array} \right.$$

Une relaxation SDP de (QP)

On construit (SDP) en introduisant une variable matricielle X représentant l'ensemble des produits $x^T x$, (ou de façon équivalente les variables y), après avoir relâché les contraintes d'intégrité de (QP'') :

$$
(\overline{QP}'') \left\{
\begin{array}{ll}
Min & f(x) = \langle Q, y \rangle + c^T x \\
s.c. & Ax = b \\
 & Dx \leq e \\
 & 0 \leq x \leq u \\
 & x \text{ réel} \\
 & (x, y) \in S \ (\text{ i.e. } y_{ij} = x_i x_j \ ) \\
 & \langle A^T A, y \rangle - (2A^T b)^T x = -b^T b
\end{array}
\right.
$$

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

Une relaxation SDP de (QP)

$\begin{pmatrix} 1 & x \\ x^T & X \end{pmatrix} \succeq 0$ est équivalent à $X - xx^T \succeq 0$, qui est la relaxation de $X = xx^T$ n'est pas convexe.

Calcul du dual de $(PSDP)$

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

Son dual

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

Son dual

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

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

Calcul des paramètres α et β par la programmation semie définie

Calcul des paramètres α et β par la programmation semie définie

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

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 solution continue de notre programme reformulé soit la plus proche possible de sa solution entière. (Dualité lagrangienne)

Conclusion

- Algorithme de résolution des programmes quadratiques convexes en nombres entiers utilisé dans les solveurs (B & B).

Conclusion

- Algorithme de résolution des programmes quadratiques convexes en nombres entiers utilisé dans les solveurs (B & B).
- Rappel sur la programmation SDP.

Conclusion

- Algorithme de résolution des programmes quadratiques convexes en nombres entiers utilisé dans les solveurs (B & B).
- Rappel sur la programmation SDP.
- Utilisation de la programmation SDP pour la conception d'un algorithme de résolution de programmes quadratiques non convexes en nombres entiers

Conclusion

- Algorithme de résolution des programmes quadratiques convexes en nombres entiers utilisé dans les solveurs (B & B).
- Rappel sur la programmation SDP.
- Utilisation de la programmation SDP pour la conception d'un algorithme de résolution de programmes quadratiques non convexes en nombres entiers
    1. Résoudre $(DSDP)$ pour calculer α^* et β^* qui rendent convexe (QP')

Conclusion

- Algorithme de résolution des programmes quadratiques convexes en nombres entiers utilisé dans les solveurs (B & B).
- Rappel sur la programmation SDP.
- Utilisation de la programmation SDP pour la conception d'un algorithme de résolution de programmes quadratiques non convexes en nombres entiers
  1. Résoudre $(DSDP)$ pour calculer α^* et β^* qui rendent convexe (QP')
  2. Résoudre notre problème reformulé par un solveur standard

Conclusion

- Algorithme de résolution des programmes quadratiques convexes en nombres entiers utilisé dans les solveurs (B & B).
- Rappel sur la programmation SDP.
- Utilisation de la programmation SDP pour la conception d'un algorithme de résolution de programmes quadratiques non convexes en nombres entiers
  1. Résoudre $(DSDP)$ pour calculer α^* et β^* qui rendent convexe (QP')
  2. Résoudre notre problème reformulé par un solveur standard

On arrive à résoudre des problèmes de plusieurs dizaines de contraintes et variables