Programmation semidéfinie : introduction et applications en optimisation combinatoire

Frédéric Roupin
roupin@lipn.univ-paris13.fr

Préambule

Les problèmes d'optimisation en variables bivalentes ou discrètes sont le modèle mathématique de nombreux problèmes concrets d'optimisation et de décision. Ces problèmes, généralement difficiles, se rencontrent par exemple en conception de réseaux de télécommunication, de circuits intégrés, en productique et en informatique. Il est donc très utile de posséder des méthodes efficaces pour leur résolution (même approchée).

La programmation semidéfinie (SDP) est un cas particulier de la programmation convexe et peut être vue comme une généralisation de la programmation linéaire. La SDP permet d'obtenir des bornes de très bonne qualité pour de nombreux problèmes combinatoires difficiles, et est également à la base d'heuristiques efficaces pour résoudre de manière approchée ces problèmes. Après quelques rappels concernant la programmation linéaire et les matrices symétriques réelles, ce cours présente quelques éléments de la théorie de la dualité en SDP, et plusieurs exemples d'application de la SDP en optimisation combinatoire, où la démarche de modélisation est mise en évidence.

Table des matières

CHAPITRE 1

Rappels d'algèbre linéaire et de programmation linéaire

1.1. S_n^+ : cône convexe fermé des matrices symétriques positives

Dans cette section, nous rappelons quelques propriétés des matrices réelles carrées symétriques, dont l'ensemble est noté S_n, ainsi que quelques résultats concernant S_n^+, le cône des matrices réelles carrées symétriques positives.

1.1.1. Définitions et notations. Soient $x, y \in \mathbb{R}^n$. On notera $x^T y = \sum_{i=1}^n x_i y_i$ le produit scalaire $< x, y >$, et

$$xy^T = \begin{bmatrix} x_1y_1 & \cdots & x_1y_i & \cdots & x_1y_n \\ \vdots & \ddots & & & \vdots \\ x_iy_1 & & x_iy_i & & x_iy_n \\ \vdots & & & \ddots & \vdots \\ x_ny_1 & \cdots & x_ny_i & \cdots & x_ny_n \end{bmatrix}$$ le produit tensoriel $x \otimes y$. Remarquons que $xx^T \in S_n$.

Nous noterons $\mathbf{d}(A)$ la diagonale d'une matrice A de S_n, et $\mathbf{diag}(x)$ la matrice diagonale construite à l'aide d'un vecteur x de $\mathbb{R}^n$, *i.e.* telle que $\mathbf{d}(\mathbf{diag}(x)) = x$. Enfin nous noterons e_n le vecteur de $\mathbb{R}^n$ dont toutes les composantes valent 1.

Soit A et B dans S_n, on pose

$$A \bullet B = Tr\left(A^T B\right) = \sum_{i=1}^n \sum_{j=1}^n A_{ij} B_{ij}$$

Il est facile de vérifier que l'opérateur $\bullet$ définit un produit scalaire dans S_n. A l'aide de $\bullet$, il est possible d'écrire toute forme quadratique $x^T Ax + b^T x + c$ sous la forme $A \bullet xx^T + b^T x + c$.

En effet le terme $xx^T = x \otimes x$ est une matrice qui est le produit tensoriel de x avec lui-même et dont le terme de la $i$ème ligne/$j$ème colonne vaut $x_i x_j$.

Exemple 1.1.1. $x_1^2 + 2x_2^2 - x_1x_2 + x_1 = \begin{bmatrix} 1 & -\frac{1}{2} \\ -\frac{1}{2} & 2 \end{bmatrix} \bullet xx^T + \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ avec $xx^T = \begin{bmatrix} x_1^2 & x_1x_2 \\ x_1x_2 & x_2^2 \end{bmatrix}$

Définition 1.1.2. Une matrice A de S_n est positive (resp. définie positive) si et seulement si pour tout vecteur réel non nul z de dimension n, on a $z^T Az \geq 0$ (resp. $z^T Az > 0$).

Notation 1.1.3. On note : $A \succeq 0$ une matrice positive, $A \succ 0$ une matrice définie positive, S_n^+ l'ensemble des matrices positives, et S_n^{++} l'ensemble des matrices définies positives.

On peut alors définir un ordre partiel sur S_n, noté '$\succeq$' (dit ordre de *Löwner*), défini par : $A \succeq B$ si et seulement si la matrice $A - B$ est positive.

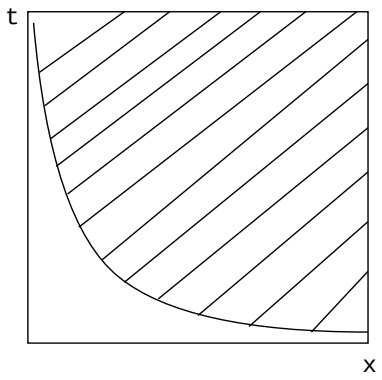
1.1.2. Quelques propriétés de S_n^+. Nous rappelons ici quelques propriétés utiles concernant les matrices positives.

PROPOSITION 1.1.4. *Soit A dans S_n, alors il existe une base de vecteurs orthonormés dans laquelle A est diagonale (la matrice de passage P vérifie donc $PP^T = Id_n$).*

PROPOSITION 1.1.5. *L'ensemble S_n^+ des matrices réelles symétriques positives est un cône fermé convexe dans S_n. De plus, S_n^+ n'est pas un polyèdre pour $n \geq 2$.*

Ce dernier résultat implique que bien que convexe, un ensemble de points vérifiant une inégalité matricielle linéaire ne peut pas en général être représenté par un nombre fini d'inégalités linéaires simples.

EXEMPLE. Soit x et t réels vérifiant $\begin{bmatrix} x & 1 \\ 1 & t \end{bmatrix} \succeq 0 \iff x \geq 0,\ t \geq 0,\ xt \geq 1$ (mineurs symétriques).



La région du plan obtenue est convexe mais n'est pas un polyèdre.

DÉFINITION 1.1.6. On appelle mineur symétrique d'une matrice A de S_n, une sous-matrice de A obtenue en choisissant de 1 à n lignes et les colonnes de mêmes indices.

REMARQUE 1.1.7. Les mineurs principaux d'une matrice A sont des mineurs symétriques. Dans ce cas, on choisit les k premières lignes et colonnes de A ($k \in \{1, ..., n\}$).

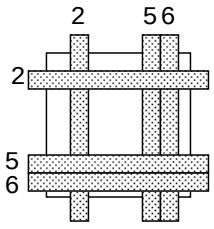


FIGURE 1.1.1. La sous-matrice 3×3 formée à partir des lignes et des colonnes $\{2, 5, 6\}$ est un mineur symétrique.

EXEMPLE. $\begin{bmatrix} 1 & 3 \\ 3 & 6 \end{bmatrix}$ et $\begin{bmatrix} 4 & 5 \\ 5 & 6 \end{bmatrix}$ sont des mineurs symétriques de $\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}$.

Autre cas particulier important : chaque élément de la diagonale d'une matrice M de S_n est un mineur symétrique de M. Donc une matrice positive a nécessairement des éléments diagonaux non négatifs.

PROPOSITION 1.1.8. *Les propositions suivantes sont équivalentes :*

(1) A est positive ($A \in S_n$).

(2) La forme quadratique $x^T Ax$ est positive pour tout vecteur x de $\mathbb{R}^n$.

(3) Théorème de *Fejer* : pour toute matrice $B \succeq 0$, $A \bullet B \geq 0$. Autrement dit, le cône des matrices symétriques positives est son propre polaire :
$S_n^+ = S_n^{+*} = \{A \in S_n : A \bullet B \succeq 0 \,\text{pour tout}\, B \in S_n^+\}$

(4) A peut s'écrire comme la matrice de *Gram* de n vecteurs $v_1, ..., v_n$ de $\mathbb{R}^m$. Donc, pour tout (i, j) on a $a_{ij} = v_i^T v_j$, *i.e.* $A = VV^T$.

(5) **Le déterminant de tout mineur symétrique de A est positif**.

Cette dernière caractérisation est très utile dans la pratique, en particulier pour démontrer qu'une matrice n'est pas positive.

LEMME 1.1.9. *Soit A et B dans S_n. Si $A \succeq 0$ et $B \succeq 0$ alors $A \bullet B \geq 0$ et l'égalité est atteinte si et seulement si $A^T B = AB = 0$.*

DÉMONSTRATION. Ecrivons B sous la forme $B = UDU^T$, où D est diagonale et telle que $UU^T = Id_n$. B est positive donc $D_{ii} \geq 0$ pour $i = 1, \ldots, n$. Soit alors $C = U^T AU$, C est également positive puisque $A \succeq 0$, et donc nécessairement $C_{ii} \geq 0$ pour $i = 1, \ldots, n$. On obtient $A \bullet B = UCU^T \bullet U^T DU = Tr\,(DC) = D \bullet C = \sum_{i=1}^n D_{ii} C_{ii} \geq 0$.

Il nous reste donc à montrer que $DC = 0$ si $D \bullet C = 0$. Supposons que $D \bullet C = 0$. Puisque tous les produits $D_{ii} C_{ii}$ sont positifs, la somme est nulle si et seulement si tous les termes sont nuls. Supposons alors que $(DC)_{ij} \neq 0$ pour un certain couple (i, j), alors $D_{ii} C_{ij} \neq 0$. Donc $D_{ii} > 0$ et nécessairement $C_{ii} = 0$. Mais puisque $C \succeq 0$, les ièmes colonne et ligne de C sont nulles. En effet, les mineurs symétriques 2×2 formés des lignes et colonnes i et k ($k \in \{1, ..., n\}$) sont positifs ou nuls : $C_{ii} C_{kk} - C_{ik}^2 \geq 0 \Rightarrow -C_{ik}^2 \geq 0$. Ce qui est contradictoire puisque $C_{ij} \neq 0$. □

1.1.3. Décomposition de Cholesky.

1.1.3.1. *Enoncé du théorème.*

THÉORÈME 1.1.10. Une matrice A symétrique est définie positive si et seulement si elle est factorisable sous la forme $A = VV^T$ où V est une matrice triangulaire inférieure à diagonale positive. Cette factorisation est unique.

EXEMPLE. $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 5 & 2 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

On peut également utiliser une décomposition de Cholesky *incomplète* si la matrice n'est pas définie. Soit A positive non définie et de rang r. A est également factorisable en VV^T avec $rang(V) = r$. Mais il n'y a plus unicité.

EXEMPLE. On a $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 5 & 2 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ et aussi

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \sqrt{5} & 0 & 0 \\ 0 & \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \sqrt{5} & \frac{2}{\sqrt{5}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{5}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 5 & 2 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1.1.3.2. *Algorithme de la décomposition.* On veut $A = VV^T$ (V triangulaire inférieure).

– Nécessairement $a_{11} = v_{11}^2$, et donc $v_{11} = \sqrt{v_{11}}$.

On peut alors calculer toute la première colonne de V :

$a_{i1} = v_{i1}v_{11}$, et donc $v_{i1} = \frac{a_{i1}}{v_{11}}$.

Si v_{11} est nul alors toute la 1^{ere} ligne (et colonne) de A est nulle (mineurs symétriques) :

$$A = \begin{bmatrix} 0 & a & b & c \\ a & d & \dots & \dots \\ b & \dots & e & \dots \\ c & \dots & \dots & f \end{bmatrix} \succeq 0 \Rightarrow 0.d - a^2 \geq 0 \Rightarrow a = 0$$

On peut prendre $v_{i1} = 0$ dans ce cas.

– On calcule alors v_{22} : $a_{22} = v_{21}^2 + v_{22}^2$, ce qui donne $v_{22} = \sqrt{a_{22} - v_{21}^2}$.

On peut alors calculer la deuxième colonne comme précédemment.

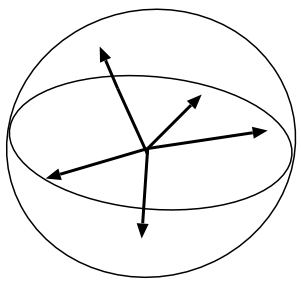
– A l'étape i, on calcule le terme v_{ii} de la façon suivante $a_{ii} = \sum_{j=1}^{i-1} v_{ij}^2 + v_{ii}^2$ d'où $v_{ii} = \sqrt{a_{ii} - \sum_{j=1}^{i-1} v_{ij}^2}$.

On calcule la colonne i de S par $a_{ji} = \sum_{k=1}^{i-1} v_{jk}v_{ik} + v_{ii}v_{ji}$ d'où $v_{ji} = \frac{a_{ji} - \sum_{k=1}^{j-1} v_{jk}v_{ik}}{v_{ii}}$

Ainsi le point 4 de la proposition 1.1.8 peut être complété de la manière suivante :

Toute matrice A de S_n^+ peut s'écrire $A = VV^T$, où V est une matrice de $\mathbb{R}^{m \times n}$ ($m \leq n$). Une matrice de rang maximal peut être obtenue en $O(n^3)$ en utilisant une décomposition de Cholesky.

Si tous les termes diagonaux de A sont égaux à 1, alors pour tout i dans $\{1, ..., n\}$, $a_{ii} = v_i^T v_i = \sum_{j=1}^{n} v_{ij}^2 = 1$, et les vecteurs colonnes de B peuvent être vus comme n vecteurs $v_1, \dots, v_n$ de la sphère unité S^m.



Le terme a_{ij} est alors égal au produit scalaire $v_i^T v_j$, A étant la *matrice de Gram* de $\{v_1, ..., v_n\}$:

$$A = VV^T = \begin{bmatrix} v_{11} & v_{12} & \cdots\cdots\cdots & v_{1n} \\ v_{21} & v_{22} & \cdots\cdots\cdots & v_{2n} \\ \vdots & \vdots & \cdots\cdots\cdots & \vdots \\ v_{mn} & v_{m2} & \cdots\cdots\cdots & v_{mn} \end{bmatrix} \begin{bmatrix} v_{11} & v_{21} & \cdots & v_{n1} \\ v_{12} & v_{22} & \cdots & v_{n2} \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ v_{1m} & v_{2m} & \cdots & v_{nm} \end{bmatrix}$$

PROPOSITION 1.1.11. *Il est équivalent de considérer une matrice positive dont les éléments diagonaux valent tous* 1 *ou un champ de vecteurs de la sphère unité. De plus, on peut passer d'une représentation à l'autre en calculant* VV^T *ou en utilisant une décomposition de Cholesky.*

Pour terminer cette section, rappelons un théorème et deux lemmes qui nous seront utiles lors de l'élaboration de nos relaxations par programmation semidéfinie :

THÉORÈME 1.1.12. *Si* $A \in S_p^{++}$, $C \in S_n$, *et* B *est une matrice réelle* $p \times n$, *alors* $\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \succcurlyeq 0$ *est équivalent à* $C - B^T A^{-1} B \succcurlyeq 0$.

DÉMONSTRATION. Le résultat est immédiat car :

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} = \begin{bmatrix} I & 0 \\ B^T A^{-1} & I \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & C - B^T A^{-1} B \end{bmatrix} \begin{bmatrix} I & A^{-1}B \\ 0 & I \end{bmatrix}$$

□

Un cas particulier important est le suivant :

LEMME 1.1.13. *L'ensemble* $(X, x) \in S_n \times \mathbb{R}^n$ *satisfaisant* $X - xx^T \succeq 0$ *est fermé, convexe, et* $\begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0 \Leftrightarrow X - xx^T \succeq 0$.

LEMME 1.1.14. *Si* $(X, x) \in S_n \times \{0,1\}^n$ *est tel que* $\mathbf{d}(X) = x$ *et* $X - xx^T \succeq 0$, *alors* $X = xx^T$ *(et donc* $X_{ij} \in \{0,1\}$ *pour tout*$(i,j) \in \{1, ..., n\}$*).*

DÉMONSTRATION. Puisque $X \succeq 0$ et $\mathbf{d}(X) = x$, nous avons pour tous $i \neq j$ dans $\{1, ..., n\}$ $\begin{bmatrix} 1 & x_i & x_j \\ x_i & x_i & X_{ij} \\ x_j & X_{ij} & x_j \end{bmatrix} \succeq 0$ (Proposition 1.1.8). Si $x_i = 0$ ou $x_j = 0$ alors $-X_{ij}^2 \leq 0$ et donc $X_{ij} = 0$, et si $x_i = x_j = 1$ alors $\det \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & X_{ij} \\ 1 & X_{ij} & 1 \end{bmatrix} = -(X_{ij} - 1)^2 \geq 0$ et donc $X_{ij} = 1$. Ce qui est équivalent à $X = xx^T$. □

1.2. Programmation linéaire

Considérons un programme linéaire (PL) se présentant sous la forme suivante :

$$(PL) \quad \min_{x \in \mathbb{R}^m} c^T x \text{ sous la contrainte } Ax \geq b \tag{1.2.1}$$

où $A \in \mathbb{R}^{m \times n}$ et $b \in \mathbb{R}^m$. Le dual de (PL) est :

$$(DPL) \quad \max_{y} b^T y \text{ sous les contraintes } A^T y = c \, y \geq 0 \tag{1.2.2}$$

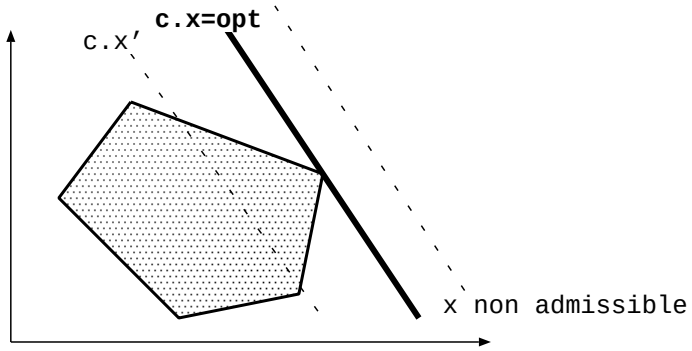


FIGURE 1.2.1. *Exemple de programme linéaire*

Un problème linéaire est évidemment convexe. Géométriquement, la région admissible est un polyèdre ou même un polytope (si elle est bornée comme sur la figure ci-dessus). Le problème est alors de se déplacer dans la direction du vecteur c (ou $-c$) jusqu'à atteindre un optimum (sur la figure il est unique) qui est toujours sur la frontière. Rappelons également que si le PL possède des solutions optimales, il existe toujours un point extrême en faisant partie (un point extrême est un point de la région admissible Π ne pouvant être exprimé comme combinaison convexe stricte d'autres points de Π). Cette dernière propriété est indispensable pour la méthode du simplexe.

LEMME 1.2.1. *(Lemme de Farkas) Un système d'inégalités linéaires $a_1^T x \leq b_1, ..., a_m^T x \leq b_m$ ne possède aucune solution si et seulement si il existe $\lambda_1, ..., \lambda_m$ tels que $\sum_{i=1}^m \lambda_i a_i = 0$ et $\sum_{i=1}^m \lambda_i b_i = -1$.*

Grâce à ce Lemme, on peut montrer le Théorème suivant, dit "**Théorème de la Dualité**", qui est un des résultats fondamentaux en PL :

THÉORÈME 1.2.2. *Si l'un des programmes primal ou dual possède une solution optimale alors l'autre également, et les deux valeurs optimales sont égales.*

Un autre théorème important de la PL est celui des "**Écarts Complémentaires**". Il permet de caractériser les solutions optimales des programmes dual et primal :

THÉORÈME 1.2.3. *Soit x une solution du programme primal et y, une solution du programme dual. x et y sont tous deux optimaux si et seulement si pour chaque indice i tel que $y_i \neq 0$, la $i$ème contrainte du primal est satisfaite avec égalité. Ainsi, on a $y_i \left(Ax - b\right)_i = 0$, pour $i = 1, ..., m$.*

CHAPITRE 2

Programmes semidéfinis

2.1. Région admissible d'un programme semidéfini

2.1.1. Inégalités matricielles linéaires. Grâce à l'ordre de *Löwner* défini au 1.1.1, nous allons pouvoir à présent définir les *inégalités matricielles linéaires*. Cette notion est fondamentale pour la définition des programmes semidéfinis puisqu'elle permet de décrire la région admissible de ces problèmes.

DÉFINITION 2.1.1. soit $A_0, A_1, ..., A_m$ des matrices de S_n ($n \geq 1$). On appelle **inégalité matricielle linéaire** une expression de la forme

$$\sum_{i=1}^{m} \lambda_i A_i \succeq A_0$$

où les λ_i ($i \in \{1, ..., m\}$) sont des réels quelconques.

EXEMPLE. imposons à deux réels t_1 et t_2 de vérifier

$$t_1 \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + t_2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \succeq \begin{bmatrix} -1 & 0 & -1 \\ 0 & -2 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

Cette simple inégalité contient a priori beaucoup plus "d'information" qu'une inégalité linéaire classique. En effet, elle est équivalente à l'*infinité* d'inégalités suivantes :

$$\forall z \in \mathbb{R}^n,\ z^T X z \geq 0, \text{ avec } X = \begin{bmatrix} t_2 + 1 & t_1 & 1 \\ t_1 & 2 & 0 \\ 1 & 0 & t_2 \end{bmatrix}.$$

Remarquons que dans une inégalité matricielle linéaire les éléments de la matrice $\sum_{i=1}^{m} \lambda_i A_i - A_0$ sont les fonctions linéaires des λ_i ($i \in \{1, ..., m\}$).

2.1.2. Interprétation des matrices obtenues. Soient $A_0, A_1, ..., A_m$ des matrices de S_n ($n \geq 1$) linéairement indépendantes. Considèrons $\aleph$ l'espace affine défini par le repère $(A_0, A_1, ..., A_m)$. A_0 est donc l'origine et $A_1, ..., A_m$ les vecteurs de base. Soit alors $A \in \aleph$, on peut écrire A à l'aide de ses coordonnées dans $(A_0, A_1, ..., A_m)$:

$$A = \sum_{i=1}^{m} \lambda_i A_i - A_0$$

Si on impose en plus à A d'être positive, nous obtenons une inégalité matricielle linéaire. Ainsi les matrices vérifiant cette inégalité sont exactement les matrices de $S_n^+ \cap \aleph$.

2.2. Définition

Soit le programme mathématique suivant, où on cherche à minimiser une fonction linéaire d'une variable $x \in \mathbb{R}^m$dans une région définie par une inégalité matricielle linéaire.

$$(SDP) \begin{cases} \text{Minimiser} & f(x) = c^T x \\ \text{Sous contrainte} & F(x) = \sum_{i=1}^{m} x_i A_i - A_0 \succeq 0 \end{cases} \tag{2.2.1}$$

où c est un vecteur de $\mathbb{R}^m$, et les constantes A_i ($i \in \{0, ..., m\}$) sont des éléments de S_n supposés être linéairement indépendants. (SDP) est appelé *programme semidéfini.* La fonction F associe donc à tout vecteur x de $\mathbb{R}^m$ une matrice qui est positive lorsque x est admissible.

Il est rapide de vérifier qu'un programme semidéfini est un problème d'optimisation convexe. En effet, la fonction f est linéaire, et si $F(x) \succeq 0$ et $F(y) \succeq 0$ on a $F(\lambda x + (1-\lambda)y) = \lambda F(x) + (1-\lambda)F(y) \succeq 0$, pour tout $0 \leq \lambda \leq 1$.

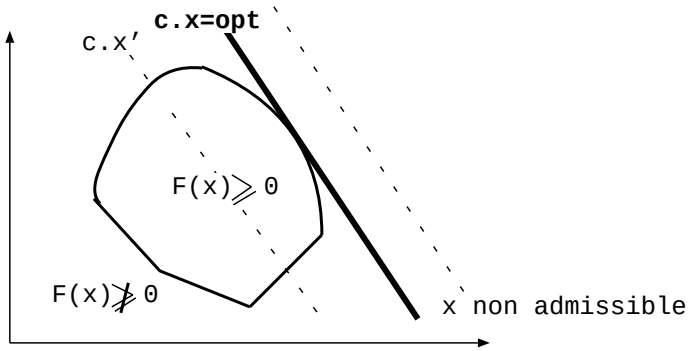


FIGURE 2.2.1. *Exemple de région admissible d'un (SDP) en dimension 2*

La figure 2.2.1 illustre le fait que la frontière de la région admissible n'est pas affine par morceau (comme en programmation linéaire) mais constituée de surfaces algébriques (*i.e.* d'ensembles de points annulant simultanément un certain nombre de polynômes). En effet, les points de cette frontière sont des matrices positives mais nécessairement singulières (puisque par continuité, pour toute matrice définie positive il existe un voisinage sur lequel les matrices sont définies). Or, dire que $F(x)$ est non inversible implique que certains mineurs de $F(x)$ deviennent nuls (et donc certains polynômes en $x_1, ..., x_m$ sont nuls). Ainsi, l'optimum (s'il existe) sera atteint en un point x_{opt} tel que $F(x_{\text{opt}})$ est singulière. En fait, l'intérieur de la région admissible (qui est un convexe fermé) est constitué des points x tels que $F(x)$ est définie positive, et la frontière est l'ensemble des points x tels que $F(x)$ est positive et singulière (voir exercice en travaux dirigés).

La programmation semidéfinie peut être vue comme un programme linéaire *semi-infini* particulier en ce sens que la contrainte $F(x) \succeq 0$ est équivalente à une infinité de contraintes : $\forall z \in \mathbb{R}^n$, $z^T F(x) z \geq 0$. Géométriquement, on peut par exemple imaginer une famille infinie de contraintes linéaires qui "envelopperait" les portions non-linéaires (les surfaces algébriques) de la région admissible.

Toutes ces remarques montrent que la méthode du simplexe est difficilement généralisable aux (SDP). En revanche, il n'y a pas d'obstacle à des extensions de la théorie de la dualité et les méthodes de point intérieur pour les (SDP).

2.3. Comparaison avec la Programmation linéaire

Soit v un vecteur quelconque de $\mathbb{R}^n$. On note $\mathbf{diag}(v)$, la matrice dont la diagonale est égale à v et dont les autres éléments sont nuls. Un vecteur v est positif (toutes ses composantes sont positives) (noté $v \geq 0$) si et seulement si $\mathbf{diag}(v)$ est positive. On donc peut reformuler 1.2.1 comme un (SDP). En effet, posons $F(x) = \mathbf{diag}(Ax - b)$, *i.e.* $A_0 = \mathbf{diag}(b)$, et $A_i = \mathbf{diag}(a_i)$ pour i dans $\{1, ..., m\}$(a_i désigne le $i$ème vecteur colonne de la matrice A).

$$\text{(2.3.1)} \qquad \begin{cases} \text{Minimiser} & c^T x \\ \text{Sous contrainte} & F(x) = \mathbf{diag}(Ax - b) \succeq 0 \end{cases}$$

Ainsi, un (PL) quelconque est un (SDP) où toutes les matrices A_i $(i = 0, ..., m)$ sont diagonales.

Exemple. Soit le programme linéaire suivant :

$$(PL) \begin{cases} \text{Minimiser} & x_1 + x_2 \\ \text{Sous contraintes :} & \\ & 2x_1 + x_2 \leq 0 \\ & 3x_1 - x_2 \geq 0 \end{cases}$$

On peut réécrire ce PL sous la forme du SDP (équivalent) :

$$(SDP)_{PL} \begin{cases} \text{Minimiser} & x_1 + x_2 \\ \text{Sous contraintes :} & \\ & \begin{bmatrix} -2x_1 - x_2 & 0 \\ 0 & 3x_1 - x_2 \end{bmatrix} \succeq 0 \end{cases}$$

Mais la réciproque est fausse. Pour s'en convaincre, considérons le problème convexe (mais non-linéaire) suivant :

$$\text{(2.3.2)} \qquad \begin{cases} \text{Minimiser} & \frac{\left(c^T x\right)^2}{d^T x} \\ \text{Sous contrainte} & Ax + b \geq 0 \end{cases}$$

Supposons (pour simplifier notre analyse) que $d^T x > 0$ si $Ax + b \geq 0$. Ce problème peut alors être reformulé de la façon suivante

$$\text{(2.3.3)} \qquad \begin{cases} \text{Minimiser} & t \\ \text{Sous les contraintes} & Ax + b \geq 0 \\ & \frac{\left(c^T x\right)^2}{d^T x} \leq t \end{cases}$$

Pourquoi introduire cette nouvelle variable t ? Notre but étant d'obtenir un (SDP), il est nécessaire d'optimiser une fonction linéaire, ce qui est maintenant le cas. La variable t est appelée *complément de Schur*. Remarquons à présent que la dernière contrainte peut être remplacée par l'inégalité matricielle $\begin{bmatrix} t & c^T x \\ c^T x & d^T x \end{bmatrix} \succeq 0$. En effet, cette inégalité est équivalente à $t \geq 0$, $d^T x \geq 0$ et $t - \frac{\left(c^T x\right)^2}{d^T x} \geq 0$ (puisque nous avons supposé que $Ax + b \geq 0$ implique $d^T x > 0$) (on considère tous les mineurs symétriques de la matrice). Finalement, nous obtenons le programme semidéfini

suivant

$$(2.3.4)\qquad \begin{cases} \text{Minimiser} & t \\ \text{Sous la contrainte} & \begin{bmatrix} \mathbf{diag}(Ax+b) & 0 & 0 \\ 0 & t & c^T x \\ 0 & c^T x & d^T x \end{bmatrix} \succeq 0 \end{cases}$$

Les (SDP) représentent donc une classe de problèmes convexes strictement plus importante que les (PL).

2.4. Comparaison avec la programmation quadratique convexe

Soit

$$(Q)\begin{cases} \text{Minimiser} & f_0(x) \\ \text{Sous les contraintes} & f_i(x) \leq 0 \quad i=1,...,m \end{cases}$$

où $x \in \mathbb{R}^n$, et pour tout $i \in \{0,...,m\}$, $f_i(x)$ est une forme quadratique convexe. $f_i(x) \leq 0$ peut s'écrire $x^T A_i^T A_i x + c_i^T x + d_i \leq 0$, ce qui est équivalent (Théorème 1.1.12) à $\begin{bmatrix} I_n & A_i x \\ x^T A_i^T & -c_i^T x - d_i \end{bmatrix} \succeq 0$. (Q) peut ainsi s'écrire comme un programme semidéfini (on introduit une variable t pour avoir une fonction objectif linéaire) :

$$\begin{cases} \text{Minimiser} & t \\ \text{s.c.} & \begin{bmatrix} I_n & A_0 x \\ x^T A_0^T & -c_0^T x - d_0 + t \end{bmatrix} \succeq 0 \\ & \begin{bmatrix} I_n & A_i x \\ x^T A_i^T & -c_i^T x - d_i \end{bmatrix} \succeq 0 \;\; i=1,...,m \end{cases}$$

En utilisant le Lemme 1.1.13, on peut également formuler (Q) en introduisant plus de variables mais moins d'inégalités matricielles linéaires :

$$\begin{cases} \text{Minimiser} & t \\ \text{s.c.} & \begin{bmatrix} d_0 - t & \frac{1}{2}c_0^T \\ \frac{1}{2}c_0 & A_0^T A_0 \end{bmatrix} \bullet \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \leq 0 \\ & \begin{bmatrix} d_i & \frac{1}{2}c_i^T \\ \frac{1}{2}c_i & A_i^T A_i \end{bmatrix} \bullet \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \leq 0 \;\; i=1,...,m \\ & \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0 \end{cases}$$

2.5. Exemple d'application : un problème géométrique

Soit le problème suivant : *Soit k ellipsoïdes $E_1,...,E_k$ définis respectivement par $E_i = \{x \mid f_i(x) = x^T A_i x + 2b_i^T x + c_i \leq 0\}$ (i=1,...,k), trouver la plus petite sphère les contenant.*

On peut tester par une inégalité matricielle qu'un ellipsoïde est inclus dans un autre. Pour i et j quelconques dans $\{1,...,k\}$ considérons deux ellipsoïdes E_i et E_j (non vides). On peut montrer que E_i contient E_j si et seulement si il existe $\tau \geq 0$ tel que

$$(2.5.1)\qquad \begin{bmatrix} A_i & b_i \\ b_i^T & c_i \end{bmatrix} \preceq \tau \begin{bmatrix} A_j & b_j \\ b_j^T & c_j \end{bmatrix}$$

La sphère que nous cherchons sera représentée par une équation de la forme $f(x) = x^T x - 2x_c^T x + \gamma \leq 0$, où le vecteur x_c représente le centre de la sphère et $r = \sqrt{x_c^T x_c - \gamma}$ son rayon. Ainsi les contraintes de notre problème sont les suivantes

$$\begin{bmatrix} I & -x_c \\ -x_c^T & \gamma \end{bmatrix} \preceq \tau_i \begin{bmatrix} A_i & b_i \\ b_i^T & c_i \end{bmatrix} \qquad i = 1, ..., k.$$

Rappelons que notre but est de minimiser le rayon de la sphère. Pour cela introduisons une variable t que nous chercherons à minimiser et telle que $r^2 \leq t$. Cette dernière inégalité n'est pas linéaire mais peut s'écrire $\begin{bmatrix} I & x_c \\ x_c^T & t+\gamma \end{bmatrix} \succeq 0$. Remarquons ici encore l'utilisation d'un *complément de Schur* t afin de représenter une contrainte non-linéaire comme une inégalité matricielle linéaire. Nous pouvons enfin écrire notre problème comme un (SDP) :

$$\text{(2.5.2)} \qquad \begin{cases} \text{Minimiser} & t \\ \text{Sous les contraintes} & \begin{bmatrix} I & -x_c \\ -x_c^T & \gamma \end{bmatrix} \preceq \tau_i \begin{bmatrix} A_i & b_i \\ b_i^T & c_i \end{bmatrix} \; ; \; i = 1, ..., k. \\ & \tau_i \geq 0 \; ; \; i = 1, ..., k. \\ & \begin{bmatrix} I & x_c \\ x_c^T & t+\gamma \end{bmatrix} \succeq 0 \end{cases}$$

Toutes ces contraintes peuvent être réunies dans une seule inégalité matricielle $F(t, \gamma, x_c, \tau_1, ..., \tau_k) \succeq A_0$, afin de retrouver la formulation habituelle d'un (SDP). Pour cela on pose :

$$F(t, \gamma, x_c, \tau_1, ..., \tau_k) = \begin{bmatrix} I & -x_c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -x_c^T & \gamma & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & -x_c & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -x_c^T & \gamma & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \tau_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \tau_k & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & x_c \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_c^T & t+\gamma \end{bmatrix}$$

$$\text{et } A_{0=} \begin{bmatrix} -\tau_1 A_1 & -\tau_1 b_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\tau_1 b_1^T & -\tau_1 c_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\tau_k A_k & -\tau_k b_k & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\tau_k b_k^T & -\tau_k c_k & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

CHAPITRE 3

Dualité

3.1. Introduction

Nous allons définir à présent le dual d'un programme semidéfini. En fait, la démarche est très semblable à celle effectuée pour la programmation linéaire. Pour cette dernière la région admissible étant un polyèdre, on obtient un résultat très fort : les valeurs optimales du dual et du primal sont égales lorsqu'elles sont finies (sinon un des problèmes est alors non-réalisable et l'autre non borné). Ici, les résultats concernant la dualité seront plus faibles en ce sens qu'il peut exister un saut de dualité borné entre un (SDP) et son dual. L'absence de saut de dualité peut être garantie au prix d'une hypothèse supplémentaire : l'existence de points "intérieurs" dans les régions admissibles du dual et du primal.

3.2. Dual d'un programme semidéfini

Le dual (DSDP) du programme semidéfini (SDP) (2.2.1) est

$$(DSDP)\begin{cases} \text{Maximiser} & A_0 \bullet Z \\ \text{Sous contraintes} & A_i \bullet Z = c_i\,;\ i = 1, ..., m \\ & Z \succeq 0 \end{cases} \tag{3.2.1}$$

La variable est donc ici une matrice Z appartenant à S_n. Les contraintes sont linéaires en Z. Ce problème revient donc à optimiser une fonction linéaire d'une matrice symétrique positive sous contraintes linéaires. Le cône des matrices symétriques positives étant son propre polaire, $(DSDP)$ est bien le programme dual de (SDP). En effet, ce dernier est $\max_{Z\succeq 0} \min_{x\in\mathbb{R}^n} L(x, Z)$ où $L(x, Z) = c^T x + Z \bullet (A_0 - \sum_{i=1}^m x_i A_i)$ est la fonction de Lagrange associée à (SDP). $L(x, Z)$ admet un minimum $(A_0 \bullet Z)$ à condition que $A_i \bullet Z = c_i\ \forall i \in \{1, \ldots, m\}$ (de façon à annuler tous les termes en x_i) sinon on obtient $-\infty$.

(DSDP) est aussi un programme semidéfini. Pour voir cela, supposons que les matrices A_i $(\forall i \in \{1, \ldots, m\})$ sont linéairement indépendantes (ce qui n'est pas restrictif) et considérons un repère quelconque $(B_0, B_1, \ldots, B_p)$ de l'espace affine

$$\aleph = \{Z \mid Z \in S_n,\ A_i \bullet Z = c_i,\ i = 1, ..., m\}$$

dont les matrices admissibles du dual (DSDP) font partie. Puisque les matrices A_i forment un système libre, on a $p = \frac{n(n+1)}{2} - m$. Posons alors $G(y) = B_{0+} \sum_{i=1}^p y_i B_i$, et définissons le vecteur $d \in \mathbb{R}^p$ par $d_i = -A_0 \bullet B_i$. On a $d^T y = \sum_{i=1}^p (-A_0 \bullet B_i)\, y_i = -A_0 \bullet (G(y) - B_0)$. Nous pouvons alors reformuler (DSDP) comme suit :

$$\begin{cases} \text{Minimiser} & d^T y - A_0 \bullet B_0 \\ \text{Sous contrainte} & G(y) \succeq 0 \end{cases} \tag{3.2.2}$$

qui est bien un programme semidéfini.

3.3. Expression du dual dans le cas d'un programme linéaire

Afin d'illustrer le fait que la définition du dual pour un (SDP) est une extension de celle d'un (PL), nous allons écrire (DSDP) dans le cas particulier d'un (PL).

Comme nous l'avons déjà remarqué précédemment, un (PL) quelconque est un (SDP) où toutes les matrices A_i (i=0,...,m) sont diagonales. Notre dual s'écrit donc ici :

$$\text{(3.3.1)} \qquad \begin{cases} \text{Maximiser} & \mathbf{diag}(b) \bullet Z \\ \text{Sous contraintes} & \mathbf{diag}(a_i) \bullet Z = c_i \,;\ i = 1, ..., m \\ & Z \succeq 0 \end{cases}$$

Nous pouvons simplifier ce programme. En effet, puisque Z est positive, nous pouvons remplacer les éléments non diagonaux de Z par des zéros tout en gardant une matrice positive (ces éléments n'interviennent pas dans l'expression des contraintes). Il est donc possible de se restreindre aux matrices $Z = \mathbf{diag}(z)$ diagonales. Le problème 3.3.1 devient donc

$$\text{(3.3.2)} \qquad \begin{cases} \text{Maximiser} & b^T z \\ \text{Sous contraintes} & a_i^T z = c_i \,;\ i = 1, ..., m \\ & z \geq 0 \end{cases}$$

Nous reconnaissons ici le dual 1.2.2 du programme linéaire 1.2.1.

3.4. Saut de dualité et points intérieurs

3.4.1. Théorèmes de dualité. Comme nous l'avons évoqué dans l'introduction, la théorie de la dualité pour les (SDP) ne fournit pas un résultat aussi fort qu'en PL (absence de saut de dualité lorsque les problèmes sont réalisables). Toutefois, sous des hypothèses un peu plus fortes, on peut obtenir le même résultat (pas de saut) pour les (SDP). Pour cela, nous avons besoin d'introduire une nouvelle notion.

Définition 3.4.1. On appelle point intérieur du primal (SDP) (resp. du dual (DSDP)) tout vecteur x (resp. matrice Z) tel que $F(x) \succ 0$ ($F(x)$ est inversible) (resp. $Z \succ 0$ et $A_i \bullet Z = c_i$, i=0,...,m).

De tels points sont donc bien "géométriquement" à l'intérieur des régions admissibles. Lorsque de tels points existent, nous dirons que le problème (primal ou dual) est *strictement réalisable.*

Etudions à présent le saut de dualité entre (SDP) et (DSDP). Une propriété fondamentale d'un dual est de fournir des bornes pour la valeur optimale du primal (et réciproquement). Nous allons démontrer que

Proposition 3.4.2. *(Théorème de dualité faible) Pour toutes solutions admissibles x et Z respectivement de (SDP) et (DSDP), on a $c^T x \geq A_0 \bullet Z$.*

Démonstration. On a $c^T x - A_0 \bullet Z = \sum_{i=1}^{m} (A_i \bullet Z)\, x_i - A_0 \bullet Z = F(x) \bullet Z$. Or nous savons que les matrices $F(x)$ et Z sont positives, donc leur produit scalaire l'est également (voir le chapitre sur les rappels). Ce qui achève la preuve. □

Ainsi, la valeur du dual en tout point Z est toujours inférieure ou égale à la valeur du primal en tout point x. Le saut de dualité associé au couple (x, Z) étant égal à $\eta = F(x) \bullet Z$.

Soit alors les valeurs optimales de (SDP) et de (DSDP) ainsi que les ensembles respectifs de points optimaux (qui peuvent être vides), on pose :

$$p^* = \inf \left\{ c^T x \mid F(x) \succeq 0 \right\}$$

$$d^* = \sup \left\{ A_0 \bullet Z \mid Z = Z^T,\ Z \succeq 0,\ A_i \bullet Z = c_i,\ i = 1, ..., m \right\}$$

$$X_{\text{opt}} = \left\{ x \mid F(x) \succeq 0 \,\text{et}\, c^T x = p^* \right\}$$

$$Z_{\text{opt}} = \{ Z \mid Z \succeq 0,\ A_i \bullet Z = c_i,\ i = 1, ..., m, \,\text{et}\, A_0 \bullet Z = d^* \}$$

PROPOSITION 3.4.3. ***(Théorème de dualité forte) [Nesterov, Nemirovsky]*** *On a $d^* = p^*$ si au moins l'une des conditions suivantes est satisfaite :*

- *Le problème primal est strictement réalisable (il existe des points intérieurs : $\exists x \,/\, F(x) \succ 0$).*
- *Le problème dual est strictement réalisable ($\exists Z \,/\, Z = Z^T \succ 0,\ A_i \bullet Z = c_i,\ i = 1, ..., m$).*

Si les deux conditions sont remplies, alors les ensembles optimaux X_{opt} et Z_{opt} sont non vides.

3.4.2. *Condition des écarts complémentaires.* S'il existe des points optimaux alors on a $c^T x = A_0 \bullet Z = p^* = d^*$, et donc $\eta = F(x) \bullet Z = 0$ ce qui équivaut à $ZF(x) = 0$ (Z et $F(x)$ sont symétriques positives). La condition $ZF(x) = 0$ est appelée *condition des écarts complémentaires.*

En effet, pour un PL, en posant $Z = \mathbf{diag}(z)$ et $F(x) = \mathbf{diag}(Ax - b)$, notre condition devient $z_i \left(Ax - b\right)_i = 0$ pour i=1,...,n (les zéros dans z et $Ax - b$ sont "complémentaires").

3.4.3. Conditions d'optimalité pour un (SDP). Nous pouvons déduire de ce qui précède un ensemble de conditions d'optimalité pour un (SDP) (sous réserve de l'hypothèse de stricte réalisabilité) :

$$\begin{array}{l} F(x) \succeq 0 \\ Z \succeq 0,\ A_i \bullet Z = c_i,\ i = 1, ..., m \\ ZF(x) = 0 \end{array} \tag{3.4.1}$$

CHAPITRE 4

Exemples d'application en Optimisation Combinatoire

4.1. Introduction

Dans ce chapitre nous allons présenter plusieurs applications de la programmation semidéfinie pour la résolution approchée de problèmes combinatoires classiques. La SDP, tout comme la PL, peut être utilisée pour obtenir des bornes afin de mesurer la qualité de solutions fournies par des heuristiques ou dans un Branch&Bound. Il existe également des méthodes de "primalisation" qui permettent d'utiliser la solution optimale obtenue de la relaxation pour construire une solution approchée admissible pour le problème combinatoire initial. Ces techniques permettent également souvent de borner le saut du à la relaxation dans le pire cas. L'exemple réussi le plus fameux d'une telle démarche est la résolution approchée de MAX-CUT.

4.2. Modélisations en variables bivalentes à valeurs dans $\{0,1\}$ ou en $\{-1,1\}$

Avant de présenter plusieurs relaxations de problèmes combinatoires par programmation semidéfinie, nous allons distinguer deux types de modélisation. En effet, dans la littérature, les relaxations SDP sont élaborées tantôt à partir de programmes en variables à valeurs dans $\{0,1\}$, tantôt en variables à valeurs dans $\{-1,1\}$. Nous verrons que bien qu'équivalentes relativement à la qualité des relaxations obtenues, ces deux possibilités conduisent à l'élaboration de programmes semidéfinis différents tant sur les "recettes" de primalisation que sur leur interprétation géométrique.

4.2.1. Un premier exemple : QP. Prenons comme premier exemple le cas simple d'un problème quadratique où interviennent n variables bivalentes et sans autre contrainte (la diagonale de C peut être prise nulle puisque $x_i^2 = x_i$ pour tout i) :

$$(QP)_{\{0,1\}} \begin{cases} \text{Max } f(x) = \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n C_{ij}x_ix_j + \sum_{i=1}^n b_ix_i \\ \qquad = \frac{1}{2}C \bullet xx^T + b^Tx \\ \text{s.c.} \quad x_i \in \{0,1\} \ ; \ i = 1,...,n \end{cases}$$

On peut appliquer le changement de variables $y_i = 2x_i - 1$ $(i \in \{1,...,n,\})$ et travailler avec des variables en $\{-1,1\}$ pour obtenir une formulation équivalente

$$(QP)_{\{-1,1\}} \begin{cases} \text{Max } f(y) = \frac{1}{8}\sum_{i=1}^n \sum_{j=1}^n C_{ij}\,(1+y_i)\,(1+y_j) + \frac{1}{2}\sum_{i=1}^n b_i\,(1+y_i) \\ \qquad = \frac{1}{8}\left[C \bullet yy^T + \sum_{i=1}^n \left(4b_i + 2\sum_{j=1}^n C_{ij}\right) y_i\right] + \left(\frac{1}{4}C_{tot} + \frac{1}{2}\sum_{i=1}^n b_i\right) \\ \text{s.c.} \quad y_i \in \{-1,1\} \ ; \ i = 1,...,n \end{cases}$$

puisque $x_i = \frac{1+y_i}{2}$ $(i \in \{1,...,n,\})$. Nous allons considérer deux formes de relaxations SDP correspondant à chacune des formulations ci-dessus.

4.2.1.1. *Démarche géométrique : modélisation* $\{-1,1\}$. Remplaçons chacune des variables y_i $(i \in \{1,...,n,\})$ par un vecteur v_i de dimension n appartenant à la sphère unité. Initialement, nos variables ne pouvaient prendre que deux valeurs, à présent elles peuvent parcourir une région

beaucoup plus vaste mais qui contient néanmoins pour tout i dans $\{1,...,n\}$ les deux points initialement admissibles : $(1,0,....,0)$ et $(-1,0,0,...,0)$ (voir figure). Nous procédons donc bien à une relaxation du problème. On peut interpréter cette dernière comme une relaxation sur la "dimension". En effet, initialement les vecteurs étaient inclus dans un espace de dimension 1, alors qu'à présent ils peuvent engendrer un espace de dimension au plus n (leur nombre).

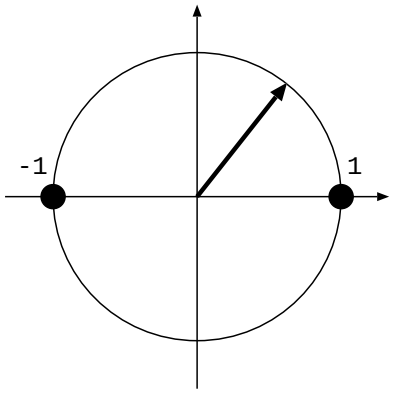


FIGURE 4.2.1. *Relaxation du problème combinatoire : on passe ici en dimension 2.*

A présent, nous devons réécrire notre fonction f en utilisant un produit dont la restriction à $\mathbb{R}$ soit un simple produit. Le produit scalaire habituel convient tout à fait. Reste le problème des termes linéaires $b'^T y = \sum_{i=1}^n b'_i y_i$ puisqu'ici nous ne pouvons pas simplement remplacer y_i par un vecteur v_i ! Pour régler ce problème nous ajoutons un vecteur supplémentaire v_0 de la sphère unité, et nous remplaçons $\sum_{i=1}^n b'_i y_i$ par $\sum_{i=1}^n b'_i v_i^T v_0$. Cela revient à "quadratiser" totalement la fonction.

Ecrivons à présent le programme obtenu :

$$(4.2.1) \qquad (P)_{QP\{-1,1\}} \begin{cases} \text{Max } F(v_0,...,v_n) = \frac{1}{8}\sum_{i=1}^n \sum_{j=1}^n C_{ij} v_i^T v_j \\ \qquad\qquad + \frac{1}{8}\sum_{i=1}^n \left(4b_i + 2\sum_{j=1}^n C_{ij}\right) v_i^T v_0 + \left(\frac{1}{4}C_{tot} + \frac{1}{2}\sum_{i=1}^n b_i\right) \\ \text{s.c. : } v_i \in S^n \; i = 0,...,n \end{cases}$$

Il s'agit bien d'une relaxation de $(QP)_{\{-1,1\}}$: la région admissible de $(P)_{QP\{-1,1\}}$ contient celle de $(QP)_{\{-1,1\}}$ et les valeurs de $F(v_0,...,v_n)$ coïncident avec celles de $f(y)$ pour les solutions admissibles de $(QP)_{\{-1,1\}}$.

Le théorème de Cholesky implique que se donner un champ de vecteurs unitaires est équivalent à se donner une matrice positive dont les éléments diagonaux valent tous 1. Nous pouvons donc effectuer un dernier changement de variables en remplaçant les vecteurs $v_0,...,v_n$ par leur matrice de Gram $Y = VV^T$(on a $Y_{ij} = v_i^T v_j$ pour tout couple (i,j)). Nous prouvons ainsi que $(P)_{QP\{-1,1\}}$ peut s'écrire comme un SDP :

$$(SDP)_{QP\{-1,1\}} \begin{cases} \text{Max } \frac{1}{8}C' \bullet Y + \left(\frac{1}{4}C_{tot} + \frac{1}{2}\sum_{i=1}^n b_i\right) \\ \text{s.c.} \quad \mathbf{d}(Y) = e_{n+1} \text{ i.e } Y_{ii} = 1\,;\, i = 0,...,n \\ \qquad Y \in S_{n+1}^+ \end{cases}$$

où $C' = \begin{bmatrix} 0 & 2b_1 + \sum_{j=1}^n C_{1j} & \cdots & 2b_i + \sum_{j=1}^n C_{ij} & \cdots & 2b_n + \sum_{j=1}^n C_{nj} \\ 2b_1 + \sum_{j=1}^n C_{1j} & & & & & \\ \vdots & & & & & \\ 2b_i + \sum_{j=1}^n C_{ij} & & & C & & \\ \vdots & & & & & \\ 2b_n + \sum_{j=1}^n C_{nj} & & & & & \end{bmatrix} \in S_{n+1}$.

4.2.2. Deuxième démarche : modélisation $\{0,1\}$. Ici, l'idée consiste à "convexifier" notre problème en :

(1) Relâchant les contraintes d'intégrité sur les variables x_i ($i \in \{1,...,n\}$) qui deviennent donc continues.

(2) Remplaçant la matrice xx^T par une matrice X telle que $X \succeq xx^T$.
Initialement on avait $X = xx^T$, il s'agit bien d'une relaxation puisque la matrice $X - xx^T = 0$ est positive.
Le Lemme 1.1.13 nous permet d'écrire cette contrainte de manière équivalente $\begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0$.

(3) Bornant les variables x_i entre 0 et 1 en imposant $\mathbf{d}(X) = x$, i.e. la diagonale de X sera égale à x. Explication : puisque $\begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0$ tous ses mineurs symétriques ont un déterminant positif, et en particulier les n mineurs 2×2 contenant les $i$ème et dernière lignes/colonnes. Ceci conduit pour tout i dans $\{1,...,n\}$ à $det \begin{bmatrix} 1 & x_i \\ x_i & x_i \end{bmatrix} \geq 0$ *i.e.* $1 \geq x_i \geq x_i^2 \geq 0$.

Nous obtenons alors directement le programme semidéfini suivant

$$(SDP)_{QP\{0,1\}} \begin{cases} \text{Max } \frac{1}{2} C \bullet X + b^T x \\ \text{s.c.} \quad \mathbf{d}(X) = x \\ \qquad \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0 \end{cases}$$

4.2.3. Lien avec l'approche lagrangienne. Notre problème de la maximisation d'une fonction quadratique pseudo-booléenne sans contrainte peut également se formuler comme (il suffira de prendre l'opposé de la valeur obtenue) :

$$(QP) \begin{cases} \text{Min} \quad f(x) = -x^T C x - b^T x \\ \text{s.c.} \quad x_i^2 = x_i \qquad \forall i \in \{1,...,n\} \end{cases}$$

Ecrivons la fonction duale de (QP) : $\theta(\mu) = \min_x x^T \left(-C + \mathbf{diag}(\mu)\right) x + (-b - \mu)^T x$. Le programme dual est : $\theta = \max_{\mu \in \mathbb{R}^n} \theta(\mu)$.

PROPOSITION 4.2.1. *Soit une forme quadratique quelconque $q(x) = x^T Q x + f^T x + c$ définie sur $\mathbb{R}^n$. Une condition nécessaire et suffisante pour que q admette une borne inférieure finie est $Q \succeq 0$ et $\exists f'/f = Qf'$ ($f \in Im(Q)$).*

A l'aide de ce résultat, on peut en déduire que le programme dual de (QP) peut s'exprimer comme suit :

$$\begin{cases} \text{Max}_{\mu} & \left[\text{Min}_{x\in\mathbb{R}}\, x^T\left(-C+\mathbf{diag}\,(\mu)\right)x+\left(-b-\mu\right)^T x\right] \\ \text{s.c.} & -C+\mathbf{diag}\,(\mu)\succeq 0 \\ & -b-\mu\in Im(-C+\mathbf{diag}\,(\mu)) \end{cases}$$

$$\text{qui est équivalent à} \begin{cases} \text{Max} & r \\ \text{s.c.} & \begin{bmatrix} -r & \frac{1}{2}\left(-b-\mu\right)^T \\ \frac{1}{2}\left(-b-\mu\right) & -C+\mathbf{diag}\,(\mu) \end{bmatrix} \succeq 0 \end{cases}$$

$$\text{Le dual de ce programme semidéfini est} \begin{cases} \text{Min} & -\begin{bmatrix} 0 & \frac{1}{2}b \\ \frac{1}{2}b & C \end{bmatrix} \bullet X \\ \text{s.c.} & X_{11}=1 \\ & X_{ii}=X_{1i} \qquad\qquad \forall i\in\{1,...,n\} \\ & X\succeq 0 \end{cases}$$

En bidualisant (QP) nous avons donc retrouvé le programme semidéfini $(SDP)_{QP\{0,1\}}$ (il suffit de prendre l'opposé et de maximiser au lieu de minimiser) dans lequel les variables $(X_{11},..,X_{1n})$ vérifient $0\leq X_{1i}\leq 1$ pour tout i dans $\{1,...,n\}$ et peuvent être associées aux variables x_i initiales.

Le problème de l'équivalence des deux relaxations SDP obtenues respectivement des modèles $\{0,1\}$ et $\{-1,1\}$ se pose alors : les bornes sont-elles les mêmes ? Dans l'affirmative, peut-on associer une solution optimale du premier SDP à une solution optimale du second ? Ou mieux encore : existe-t-il une transformation entre les deux formulations qui permette d'associer à toute solution admissible de l'un une solution admissible de l'autre ? La réponse est oui.

4.2.4. Changement de variables dans le cas général. Soit Q une matrice inversible (pas forcément positive) de S_n. Soit l'application ϕ définie de S_n dans lui-même par $\phi(M)=QMQ^T$. On peut montrer que ϕ est un automorphisme (application linéaire bijective) de S_n qui laisse stables S_n^+et S_n^{++}.

Considérons à présent un programme semidéfini quelconque :

$$(SDP)\begin{cases} \text{Max } C\bullet X \\ \text{s.c.} \quad A_i\bullet X=(\,ou\ \leq)b_i\,;\ i=1,...,k \\ \qquad X\succeq 0 \end{cases}$$

et supposons que nous souhaitions effectuer le changement de variable $W=QXQ^T$ (on effectue en fait un changement de base). On pose $\widetilde{C}=\left(Q^{-1}\right)^T CQ^{-1}$, et $\widetilde{A}_i=\left(Q^{-1}\right)^T A_iQ^{-1}$ pour $i\in\{1,...,k\}$, et on définit le programme semidéfini suivant

$$(SDP)_Q\begin{cases} \text{Max } \widetilde{C}\bullet W \\ \text{s.c.} \quad \widetilde{A}_i\bullet W=(ou\ \leq)\,b_i\,;\ i=1,...,k \\ \qquad W\succeq 0 \end{cases}$$

On démontre le lemme suivant :

LEMME 4.2.2. *X est une solution admissible de (SDP) si et seulement si $W=QXQ^T$ est une solution admissible de $(SDP)_Q$. De plus, on a $C\bullet X=\widetilde{C}\bullet W$.*

DÉMONSTRATION. Supposons W admissible de $(SDP)_Q$. On pose $X = \left(Q^{-1}\right)^T XQ^{-1}$. X est bien positive puisque W l'est (S_n^+ est stable par ϕ). On a pour tout $i \in \{1, ..., k\}$

$$\begin{aligned} \widetilde{A}_i \bullet W &= Tr\left(\left(Q^{-1}\right)^T A_i Q^{-1} QXQ^T\right) \\ &= Tr\left(\left(Q^{-1}\right)^T A_i XQ^T\right) \\ &= Tr\left(A_i X\right) \\ &= A_i \bullet X \\ &= (ou \leq) \quad b_i \end{aligned}$$

□

Le calcul est le même pour montrer que $C \bullet X = \widetilde{C} \bullet W$. La réciproque est immédiate en prenant Q^{-1} à la place de Q et en refaisant le même calcul.

4.2.5. Changement de variables pour le cas $\{-1,1\}$ et $\{0,1\}$. Reprenons notre exemple : le cas d'un problème où la fonction est quadratique et où interviennent n variables bivalentes et sans autre contrainte. Pour appliquer le lemme 4.2.2 au cas présent, il nous faut déterminer l'expression de la matrice Q. Remarquons qu'initialement on possède la relation $x = \frac{1}{2}(y + e_n)$ entre les variables des modèles $(QP)_{\{0,1\}}$ et $(QP)_{\{-1,1\}}$ (x et y sont de même dimension n). Toutefois, dans les relaxations $(SDP)_{QP\{0,1\}}$ et $(SDP)_{QP\{-1,1\}}$ nous avons ajouté une dimension aux matrices X et Y pour pouvoir traiter les termes linéaires présents dans la fonction objectif. C'est pourquoi nous ajoutons une première composante toujours égale à 1 aux deux vecteurs afin d'homogénéiser les calculs qui vont suivre. On peut écrire matriciellement

$$\begin{pmatrix} 1 \\ x \end{pmatrix} = Q \begin{pmatrix} 1 \\ y \end{pmatrix}$$

où $Q = \begin{bmatrix} 1 & 0 \\ \frac{1}{2}e_n & \frac{1}{2}I_n \end{bmatrix}$, et donc $Q^{-1} = \begin{bmatrix} 1 & 0 \\ -e_n & 2I_n \end{bmatrix}$. On peut alors appliquer les résultats du 4.2.4 puisque ϕ est bien ici un automorphisme. Appliquons notre transformation sur $(SDP)_{QP\{-1,1\}}$ pour vérifier que nous obtenons bien $(SDP)_{QP\{0,1\}}$.

On a $B = \left(Q^{-1}\right)^T AQ^{-1}$, et pour chaque contrainte $Y_{ii} = E_i \bullet Y = 1$ $i \in \{0,...,n\}$, on peut écrire que :

$$\left(Q^{-1}\right)^T E_i Q^{-1} = \begin{bmatrix} 1 & -e_n \\ 0 & 2I_n \end{bmatrix} \begin{bmatrix} 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & & & & & \vdots \\ \vdots & & 0 & & & & \vdots \\ \vdots & & & 1 & & & \vdots \\ \vdots & & & & 0 & & \vdots \\ \vdots & & & & & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -e_n & 2I_n \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & \cdots & 0 & -1 & 0 & \cdots & 0 \\ 0 & 0 & & & & & & \vdots \\ \vdots & & \ddots & & & & & \vdots \\ \vdots & & & 0 & & & & \vdots \\ \vdots & & & & 2 & & & \vdots \\ \vdots & & & & & 0 & & \vdots \\ \vdots & & & & & & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -e_n & 2I_n \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & \cdots & 0 & -2 & 0 & \cdots & 0 \\ 0 & 0 & & & & & & \vdots \\ \vdots & & \ddots & & & & & \vdots \\ 0 & & & 0 & & & & \vdots \\ -2 & & & & 4 & & & \vdots \\ 0 & & & & & 0 & & \vdots \\ \vdots & & & & & & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \end{bmatrix}$$

Nous obtenons donc la contrainte correspondante pour la matrice X :

$\left(Q^{-1}\right)^T E_i Q^{-1} \bullet X = 1$ *i.e.* $X_{00} - 4X_{i0} + 4X_{ii} = 1$. Puisque $X_{00} = 1$, on retrouve bien nos contraintes $X_{ii} = X_{0i}$ pour tout $i \in \{1,...,n\}$.

Intéressons nous à présent à la fonction objectif de $(SDP)_{QP\{-1,1\}}$. Sous forme matricielle, on a $f(Y) = \left(\frac{1}{4}C_{tot} + \frac{1}{2}\sum_{i=1}^{n} b_i\right) + \frac{1}{8}C' \bullet Y$

Appliquons l'opérateur ϕ^{-1} à

$$C' = \begin{bmatrix} 0 & 2b_1 + \sum_{j=1}^n C_{1j} & \cdots & 2b_i + \sum_{j=1}^n C_{ij} & \cdots & 2b_n + \sum_{j=1}^n C_{nj} \\ 2b_1 + \sum_{j=1}^n C_{1j} & & & & & \\ \vdots & & & & & \\ 2b_i + \sum_{j=1}^n C_{ij} & & & C & & \\ \vdots & & & & & \\ 2b_n + \sum_{j=1}^n C_{nj} & & & & & \end{bmatrix}$$

pour déterminer l'expression de la fonction objectif pour la relaxation SDP en $\{0,1\}$:

$$\begin{aligned} \phi^{-1}(C) &= \left(Q^{-1}\right)^T C' Q^{-1} \\ &= \begin{bmatrix} 1 & -e_n \\ 0 & 2I_n \end{bmatrix} C' \begin{bmatrix} 1 & 0 \\ -e_n & 2I_n \end{bmatrix} \\ &= \begin{bmatrix} -2\sum_{i=1}^n b_i - 2C_{tot} & 2b_1 & \cdots & 2b_i & \cdots & 2b_n \\ 4b_1 + 2\sum_{j=1}^n C_{1j} & & & & & \\ \vdots & & & & & \\ 4b_i + 2\sum_{j=1}^n C_{ij} & & & 2C & & \\ \vdots & & & & & \\ 4b_n + 2\sum_{j=1}^n C_{nj} & & & & & \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -e_n & 2I_n \end{bmatrix} \\ &= \begin{bmatrix} -4\sum_{i=1}^n b_i - 2C_{tot} & 4b_1 & \cdots & 4b_i & \cdots & 4b_n \\ 4b_1 & & & & & \\ \vdots & & & & & \\ 4b_i & & & 4C & & \\ \vdots & & & & & \\ 4b_n & & & & & \end{bmatrix} \end{aligned}$$

Nous obtenons donc comme transformée de notre fonction objectif :

$$\begin{aligned} \frac{1}{2}\sum_{i=1}^n b_i + \frac{1}{4}C_{tot} + \frac{1}{8}\phi^{-1}(C') \bullet X &= \frac{1}{2}\sum_{i=1}^n b_i + \frac{1}{4}C_{tot} + \frac{1}{8}\left(-4\sum_{i=1}^n b_i - 2C_{tot} + 4\sum_{i=1}^n\sum_{j=1}^n C_{ij}X_{ij} + 8b^T x\right) \\ &= \frac{1}{2}C \bullet X + b^T x \end{aligned}$$

Les deux SDP sont équivalents.

4.2.6. Lien avec la linéarisation classique des programmes quadratiques. Considérons la relaxation linéaire classique du problème $(QP)_{\{0,1\}}$ que nous écrivons sous forme matricielle :

$$(LP)_{\{0,1\}} \begin{cases} \text{Max } \frac{1}{2}C \bullet X + b^T x \\ \text{s.c.} \quad 0 \leq x_i \leq 1 \ i \in \{1,...,n\} \\ \qquad 0 \leq X_{ij} \leq x_i \ i \neq j \in \{1,...,n\} \\ \qquad 0 \leq X_{ij} \leq x_j \ i \neq j \in \{1,...,n\} \\ \qquad x_i + x_j \leq 1 + X_{ij} \ i \neq j \in \{1,...,n\} \end{cases}$$

Usuellement, on introduit uniquement les variables de linéarisation X_{ij} pour $i<j$. Il est clair que notre réécriture ne change en rien les propriétés de cette linéarisation. Considérons à présent notre relaxation $(SDP)_{QP\{-1,1\}}$ que nous allons améliorer en ajoutant certaines inégalités valides. Il est connu (et rapide à vérifier) que les inégalités triangulaires suivantes sont des inégalités valides (en fait ce sont même des facettes du polytope associé) pour le modèle $(QP)_{\{-1,1\}}$:

$$Y_{ij}+Y_{ik}+Y_{jk} \geq -1 \tag{4.2.2}$$

$$Y_{ij}-Y_{ik}-Y_{jk} \geq -1 \tag{4.2.3}$$

$$-Y_{ij}+Y_{ik}-Y_{jk} \geq -1 \tag{4.2.4}$$

$$-Y_{ij}-Y_{ik}+Y_{jk} \geq -1 \tag{4.2.5}$$

où i,j,k sont pris distincts dans $\{1,...,n\}$. Nous allons mettre certaines de ces inégalités sous la forme $\widetilde{b}\widetilde{b}^T \bullet Y \geq 1$ où $\widetilde{b}=\begin{pmatrix} b_0 \\ b \end{pmatrix} \in \mathbb{Z}^{n+1}$. Ce type d'inégalités est appelé hypermétrique. L'intérêt d'une telle formulation est qu'elle permet un calcul aisé de la transformée par ϕ^{-1} de la matrice $\widetilde{b}\widetilde{b}^T$. Considérons pour commencer le cas particulier où $b_0=b_i=b_j=1$, et $b_k=0$ pour tous les autres indices k dans $\{1,...,n\}$. On a ici

$$\begin{aligned} \widetilde{b}\widetilde{b}^T \bullet Y \geq 1 &\Leftrightarrow Y_{00}+2\left(Y_{0i}+Y_{0j}\right)+Y_{ii}+Y_{jj}+2Y_{ij} \geq 1 \\ &\Leftrightarrow 2\left(Y_{0i}+Y_{0j}\right)+2Y_{ij} \geq -2 \\ &\Leftrightarrow Y_{0i}+Y_{0j}+Y_{ij} \geq -1 \end{aligned}$$

Le passage de la première à la deuxième ligne est justifié par les égalités $Y_{ii}=1$ pour $i \in \{0,...,n\}$. On retrouve ici la première inégalité triangulaire en prenant $k=0$. Prenons à présent comme deuxième inégalité de la forme $\widetilde{b}\widetilde{b}^T \bullet Y \geq 1$ le cas où $b_0=1$, $b_i=b_j=-1$. On a

$$\begin{aligned} \widetilde{b}\widetilde{b}^T \bullet Y \geq 1 &\Leftrightarrow Y_{00}-2\left(Y_{i0}+Y_{j0}\right)+Y_{ii}+Y_{jj}+2Y_{ij} \geq 1 \\ &\Leftrightarrow Y_{ij}-Y_{i0}-Y_{j0} \geq -1 \end{aligned}$$

On obtient donc encore une inégalité triangulaire (la deuxième) avec $k=0$. Prenons comme troisième inégalité de la forme $\widetilde{b}\widetilde{b}^T \bullet Y \geq 1$ le cas où $b_0=b_i=1$, $b_j=-1$. On a

$$\begin{aligned} \widetilde{b}\widetilde{b}^T \bullet Y \geq 1 &\Leftrightarrow Y_{00}+2Y_{i0}-2Y_{j0}+Y_{ii}+Y_{jj}-2Y_{ij} \geq 1 \\ &\Leftrightarrow Y_{i0}-Y_{j0}-Y_{ij} \geq -1 \end{aligned}$$

On obtient donc encore une inégalité triangulaire (la troisième) avec $k=0$.

Prenons comme quatrième inégalité de la forme $\widetilde{b}\widetilde{b}^T \bullet Y \geq 1$ le cas où $b_0=b_j=1$, $b_i=-1$. On a

$$\begin{aligned} \widetilde{b}\widetilde{b}^T \bullet Y \geq 1 &\Leftrightarrow Y_{00}-2Y_{i0}+2Y_{j0}+Y_{ii}+Y_{jj}-2Y_{ij} \geq 1 \\ &\Leftrightarrow -Y_{i0}+Y_{j0}-Y_{ij} \geq -1 \end{aligned}$$

On obtient donc encore une inégalité triangulaire (la quatrième) avec $k = 0$. Ajoutons ces familles d'inégalités à $(SDP)_{QP\{-1,1\}}$ pour obtenir

$$(SDP_2)_{QP\{-1,1\}} \begin{cases} \text{Max } \frac{1}{8}C' \bullet Y + \left(\frac{1}{4}C_{tot} + \frac{1}{2}\sum_{i=1}^{n} b_i\right) \\ \text{s.c.} \quad Y_{0i} + Y_{0j} + Y_{ij} \geq -1 \; i \neq j \in \{1,...,n\} \\ \qquad Y_{ij} - Y_{i0} - Y_{j0} \geq -1 \; i \neq j \in \{1,...,n\} \\ \qquad Y_{i0} - Y_{j0} - Y_{ij} \geq -1 \; i \neq j \in \{1,...,n\} \\ \qquad -Y_{i0} + Y_{j0} - Y_{ij} \geq -1 \; i \neq j \in \{1,...,n\} \\ \qquad \mathbf{d}(Y) = e_{n+1} \text{ i.e } Y_{ii} = 1 \, ; \, i = 0,...,n \\ \qquad Y \in S_{n+1}^{+} \end{cases}$$

Traduisons ce nouveau SDP pour obtenir un SDP en $\{0,1\}$: $(SDP_2)_{\{0,1\}}$. Il nous faut uniquement traduire les inégalités triangulaires que nous venons d'ajouter en utilisant notre automorphisme ϕ. Pour une inégalité hypermétrique quelconque $\widetilde{b}\widetilde{b}^T \bullet Y \geq 1$, on obtient :

$$\left(Q^{-1}\right)^T \widetilde{b}\widetilde{b}^T Q^{-1} = \begin{bmatrix} b_0 - e_n^T b \\ 2b \end{bmatrix} \begin{bmatrix} b_0 - e_n^T b & 2b^T \end{bmatrix}$$

La contrainte obtenue sera donc :

$$\begin{bmatrix} b_0 - e_n^T b \\ 2b \end{bmatrix} \begin{bmatrix} b_0 - e_n^T b \\ 2b \end{bmatrix}^T \bullet \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \geq 1$$

Pour les inégalités $Y_{0i} + Y_{0j} + Y_{ij} \geq -1 \; i \neq j \in \{1,...,n\}$
on a $\left[b_0 - e_n^T b, \, 2b^T\right] = [-1, 0, ..., 0, 2, 0, ..., 0, 2, 0, ..., 0]$. D'où

$$1 - 4X_{0i} - 4X_{0j} + 4X_{ii} + 4X_{jj} + 8X_{ij} \geq 1$$

Remarquons que nous avons imposé $\mathbf{d}(X) = x$ et donc que $X_{0i} = X_{ii}$ pour tout $i \in \{1,...,n\}$, on obtient

$$X_{ij} \geq 0$$

Pour les inégalités $Y_{ij} - Y_{i0} - Y_{j0} \geq -1 \; i \neq j \in \{1,...,n\}$, on a

$$\left[b_0 - e_n^T b, \, 2b^T\right] = [3, 0, ..., 0, -2, 0, ..., 0, -2, 0, ..., 0]$$

L'inégalité correspondante pour $(SDP_2)_{\{0,1\}}$ est :

$$\begin{aligned} 9 - 12X_{0i} - 12X_{0j} + 4X_{ii} + 4X_{jj} + 8X_{ij} &\geq 1 \\ X_{ii} + X_{jj} &\leq 1 + X_{ij} \end{aligned}$$

Pour les inégalités $Y_{i0} - Y_{j0} - Y_{ij} \geq -1 \; i \neq j \in \{1,...,n\}$, on a

$$\left[b_0 - e_n^T b, \, 2b^T\right] = [1, 0, ..., 0, 2, 0, ..., 0, -2, 0, ..., 0]$$

L'inégalité correspondante pour $(SDP_2)_{\{0,1\}}$est :

$$\begin{aligned} 1 + 4X_{0i} - 4X_{0j} + 4X_{ii} + 4X_{jj} - 8X_{ij} &\geq 1 \\ X_{ii} &\geq X_{ij} \end{aligned}$$

Enfin, pour les inégalités $-Y_{i0} + Y_{j0} - Y_{ij} \geq -1\, i \neq j \in \{1, ..., n\}$, on a

$$\left[b_0 - e_n^T b,\, 2b^T\right] = [1, 0, ..., 0, -2, 0, ..., 0, 2, 0, ..., 0]$$

L'inégalité correspondante pour $(SDP_2)_{\{0,1\}}$est :

$$\begin{aligned} 1 - 4X_{0i} + 4X_{0j} + 4X_{ii} + 4X_{jj} - 8X_{ij} &\geq 1 \\ X_{jj} &\geq X_{ij} \end{aligned}$$

Ainsi on obtient

$$(SDP_2)_{\{0,1\}} \begin{cases} \text{Max } \frac{1}{2} W \bullet X + b^T x \\ \text{s.c.} \quad \mathbf{d}(X) = x \\ \qquad \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0 \\ \qquad 0 \leq X_{ij}\ i \neq j \in \{1, ..., n\} \\ \qquad X_{ij} \leq X_{ii}\ i \neq j \in \{1, ..., n\} \\ \qquad X_{ij} \leq X_{jj}\ i \neq j \in \{1, ..., n\} \\ \qquad X_{ii} + X_{jj} \leq 1 + X_{ij}\ i \neq j \in \{1, ..., n\} \end{cases}$$

Remarquons que les inégalités $0 \leq X_{ii} \leq 1\ i \in \{1, ..., n\}$ sont impliquées par $\begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0$ puisque $1 \geq X_{ii} \geq X_{ii}^2 \geq 0$. Ceci montre que la relaxation $(SDP_2)_{\{0,1\}}$ est meilleure que $(LP)_{\{0,1\}}$ (elle contient en plus les contraintes de positivité).

4.3. Construire une relaxation semidéfinie à partir d'une relaxation PL

4.3.1. Introduction. Nous disposons de deux approches ($\{0, 1\}$ et $\{-1, 1\}$) pour construire nos relaxations SDP à partir d'un problème combinatoire se présentant comme l'optimisation d'une fonction quadratique (ou simplement linéaire) soumise à des contraintes quadratiques et/ou linéaires. Mieux encore, on peut constater que quelle que soit la relaxation par programmation linéaire considérée, il sera toujours possible d'ajouter la contrainte de positivité sur la matrice constituée des variables initiales et celles ajoutées pour linéariser le programme, et ainsi de renforcer la relaxation. Avant d'appliquer ces techniques sur plusieurs problèmes combinatoires classiques, nous allons voir qu'il est possible de procéder à des traitements encore plus efficaces, afin d'obtenir des relaxations semidéfinies encore meilleures. L'idée est de considérer l'existant (les relaxations obtenues par programmation linéaire) pour obtenir une ou des relaxations SDP améliorées. Dans ce qui suit, nous allons considérer un problème combinatoire en variables $0-1$ quadratique ou linéaire. Rappelons que grâce aux résultats de la section précédente, ces résultats pourront s'appliquer en fait à tout programme quadratique ou linéaire à variables bivalentes. Soit un problème pouvant être formulé comme suit :

$$(P\{0,1\})\begin{cases} \text{Min} & A_0 \bullet X + b^T x & \\ \text{s.c. :} & A_i \bullet X + c_i^T x = (\text{ou} \leq)\, d_i & i \in \{1,...,m_1\} \\ & c_i^T x = d_i & i \in \{m_1+1,...,m_2\} \\ & d_i \leq c_i^T x \leq d_i' & i \in \{m_2+1,...,m\} \\ & X = xx^T & \\ & x \in \{0,1\}^n & \end{cases}$$

où $1 \leq m_1 \leq m_2 \leq m$. Dans le cas général, certains d_i ou d_i' , $i \in \{m_2,...,m\}$, peuvent être absents. De plus, certaines matrices A_i $i \in \{0,...,m\}$ peuvent être nulles, et si cela est le cas pour chacune d'elles (pour tout i dans $\{0,...,m\}$) alors $(P\{0,1\})$ est un programme linéaire en $0-1$. Appliquons les résultats vus dans la section précédente, afin d'obtenir une relaxation semidéfinie (fondée sur un modèle en $\{0,1\}$) :

$$(SDP_0)\begin{cases} \text{Min} & A_0 \bullet X + b^T x & \\ \text{s.c. :} & A_i \bullet X + c_i^T x = (\text{ou} \leq)\, d_i & i \in \{1,...,m_1\} \\ & c_i^T x = d_i & i \in \{m_1+1,...,m_2\} \\ & d_i \leq c_i^T x \leq d_i' & i \in \{m_2+1,...,m\} \\ & X \succeq xx^T \Leftrightarrow \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0 & \\ & \mathbf{d}(X) = x & \end{cases}$$

La relaxation standard par programmation linéaire est :

$$(P_L)\begin{cases} \text{Min} & A_0 \bullet X + b^T x & \\ \text{s.c.} & & \\ (Q) & A_i \bullet X + c_i^T x = (\text{ou} \leq)\, d_i & i \in \{1,...,m_1\} \\ (LE) & c_i^T x = d_i & i \in \{m_1+1,...,m_2\} \\ (LI) & d_i \leq c_i^T x \leq d_i' & i \in \{m_2+1,...,m\} \\ & 0 \leq x_i \leq 1 & i \in \{1,...,n\} \\ (1) & X_{ij} \geq 0 & i < j \in \{1,...,n\} \\ (2) & X_{ij} \leq x_i\,;\, X_{ij} \leq x_j & i < j \in \{1,...,n\} \\ (3) & x_i + x_j \leq 1 + X_{ij} & i < j \in \{1,...,n\} \\ & X \in S_n & \end{cases}$$

On peut vérifier que les régions admissibles de ces deux programmes sont non comparables : aucune d'entre elles n'est inclue dans l'autre. C'est pourquoi il est naturel de vouloir construire une nouvelle relaxation SDP en partant de (P_L). Nous allons donc considérer chaque type de contrainte présente dans (P_L), et lui associer une ou plusieurs contraintes. Ainsi, nous obtiendrons un SDP qui sera par construction une meilleure relaxation que (P_L), et surtout qui tiendra mieux compte des relaxations existantes.

4.3.2. Traitements plus efficaces pour les contraintes linéaires simples.

Egalités linéaires

Considérons une contrainte se présentant comme une égalité linéaire $c^T x = d$. Nous pouvons supposer que $d \geq 0$ (sinon on change les signes de chacun des membres). Dans (SDP_0) nous avons simplement gardé la contrainte $c^T x = d$, mais nous pouvons faire mieux.

PROPOSITION 4.3.1. *$cc^T \bullet xx^T = d^2$ est une contrainte valide pour $(P\{0,1\})$. Si $(X,x) \in S_n \times R^n$ est tel que $cc^T \bullet X = d^2$, $c^T x = d$ et $X - xx^T \succeq 0$ alors on a $cc^T \left(X - xx^T\right) = 0$.*

DÉMONSTRATION. Premièrement, $cc^T \bullet xx^T = d^2$ est le carré de $c^T x = d$, et donc est une contrainte valide pour $(P\{0,1\})$. Deuxièmement, considérons $(X,x) \in S_n \times R^n$ tel que $cc^T \bullet X = d^2$, $c^T x = d$ et $X - xx^T \succeq 0$. Nous pouvons écrire que $cc^T \bullet \left(X - xx^T\right) + cc^T \bullet xx^T = d^2$. La contrainte $c^T x = d$ implique $cc^T \bullet xx^T = d^2$, et donc nous obtenons $cc^T \bullet \left(X - xx^T\right) = 0$. Grâce au Lemme 1.1.9, il est équivalent d'écrire $cc^T \left(X - xx^T\right) = 0$. □

Remarquons que que si $x \in \{0,1\}^n$ est tel que $\mathbf{d}(X) = x$ et $X - xx^T \succeq 0$ alors $X = xx^T$ (Lemme 1.1.14). Donc dans ce cas, $cc^T \bullet X = d^2$ est équivalent à $c^T x = d$ (puisque $d \geq 0$). Mais la contrainte quadratique $cc^T \left(X - xx^T\right) = 0$ n'est pas satisfaite par toute les solutions admissibles et non-entières de (P_L). Ainsi, ceci conduit à une meilleure relaxation SDP que de garder simplement $c^T x = d$.

PROPOSITION 4.3.2. *On a $\left\{(X,x) \in S_n \times R^n \,;\, X - xx^T \succeq 0,\, cc^T \bullet X = d^2,\, c^T x = d\right\} = \left\{(X,x) \in S_n \times R^n \,;\, X - xx^T \succeq 0,\, cc^T \bullet X - 2dc^T x + d^2 = 0\right\}$.*

DÉMONSTRATION. Premièrement, remarquons que la contrainte $cc^T \bullet xx^T - 2dc^T x + d^2 = 0$ peut être obtenue en mettant au carré $c^T x - d = 0$, et donc c'est bien une égalité valide pour le programme $(P\{0,1\})$. Soit $(X,x) \in S_n \times R^n$ tel que $X - xx^T \succeq 0$. Si $cc^T \bullet X = d^2$ et $c^T x = d$, alors $cc^T \bullet X - 2dc^T x + d^2 = cc^T \bullet \left(X - xx^T\right) + \left(c^T x - d\right)^2 = cc^T \bullet \left(X - xx^T\right) = 0$ (Proposition 4.3.1). Réciproquement, si $cc^T \bullet X - 2dc^T x + d^2 = 0$ alors $cc^T \bullet \left(X - xx^T\right) + \left(c^T x - d\right)^2 = 0$ implique $c^T x = d$ (puisque $cc^T \bullet \left(X - xx^T\right) \geq 0$, Lemme 1.1.9), et donc $cc^T \bullet X - 2dc^T x + d^2 = cc^T \bullet X - 2d^2 + d^2 = 0$. □

Nous pouvons également multiplier chacun des côtés de l'égalité $c^T x = d$ par x_i pour tout i dans $\{1,...,n\}$ pour obtenir les contraintes "*produit*" $\sum_{j=1}^n c_j X_{ij} = dx_i \,\forall i \in \{1,...,n\}$.

PROPOSITION 4.3.3. *les contraintes $\sum_{j=1}^n c_j X_{ij} = dx_i \,\forall i \in \{1,...,n\}$ sont des inégalités valides pour $(P\{0,1\})$. Si $(X,x) \in S_n \times R^n$ est tel que $\sum_{j=1}^n c_j X_{ij} = dx_i \,\forall i \in \{1,...,n\}$ et $c^T x = d$ alors on a $cc^T \bullet X = d^2$.*

Pour montrer la première implication, il suffit de multiplier chacune des égalités $\sum_{j=1}^n c_j X_{ij} = dx_i \,\forall i \in \{1,...,n\}$ respectivement par c_i, et on les somme pour obtenir finalement $\sum_{i=1}^n \sum_{j=1}^n c_i c_j X_{ij} = d\sum_{i=1}^n c_i x_i$ i.e. $cc^T \bullet X = dc^T x$ et donc $cc^T \bullet X = d^2$ (puisque $c^T x = d$).

La réciproque est vraie si $X \succeq xx^T$. Ceci est moins évident intuitivement puisque le nombre de contraintes vérifiées passe de $O(1)$ à $O(n)$. La preuve utilise le Lemme 1.1.9.

PROPOSITION 4.3.4. *Soit $(X,x) \in S_n \times \mathbb{R}$ vérifiant $cc^T \bullet X = d^2$, $c^T x = d$ et $\begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0$, alors (X,x) vérifie $\sum_{j=1}^n c_j X_{ij} = dx_i \,\forall i \in \{1,...,n\}$*

DÉMONSTRATION. On a $cc^T \bullet X = d^2$, donc $cc^T \bullet \left(X - xx^T\right) + \left(c^T x\right)^2 = d^2$ ce qui implique $cc^T \bullet \left(X - xx^T\right) = 0$ (puisque $c^T x = d$). Or $cc^T \succeq 0$ et $X - xx^T \succeq 0$, donc leur produit scalaire est nul si est seulement si leur produit (matriciel) est nul, ce qui s'écrit :

$$\begin{bmatrix} c_1^2 & \cdots & c_1c_j & \cdots & c_1c_n \\ \vdots & \ddots & & & \vdots \\ c_kc_1 & & c_j^2 & & c_kc_n \\ \vdots & & & \ddots & \vdots \\ c_nc_1 & \cdots & c_nc_j & \cdots & c_n^2 \end{bmatrix} \begin{bmatrix} X_{11}-x_1^2 & \cdots & X_{1j}-x_1x_j & \cdots & X_{n1}-x_1x_n \\ \vdots & \ddots & & & \vdots \\ X_{1k}-x_1x_k & & X_{jj}-x_j^2 & & X_{nk}-x_kx_n \\ \vdots & & & \ddots & \vdots \\ X_{1n}-x_nx_1 & \cdots & X_{nj}-x_nx_j & \cdots & X_{nn}-x_n^2 \end{bmatrix} = 0$$

On donc pour tout k et j dans $\{1,...,n\}$: $c_k\left(\sum_{i=1}^n c_i\left(X_{ij}-x_ix_j\right)\right)=0$. Si le vecteur c est nul, le résultat est trivial, on peut donc supposer qu'il existe k_0 dans $\{1,...,n\}$ tel que $c_{k0}\neq 0$. Ce qui nous donne pour tout j dans $\{1,...,n\}$ $\sum_{i=1}^n c_iX_{ij}=x_j\sum_{i=1}^n c_ix_i=dx_j$ puisque $c^Tx=d$. □

En considérant ces résultats, nous pouvons proposer d'utiliser une des deux règles suivantes pour obtenir notre SDP "amélioré" :

(P_L)	$(SDP\{0,1\})$
$c^Tx=d$	Règle **LE1**
	$\begin{cases} cc^T\bullet X=d^2 \\ c^Tx=d \end{cases}$ $\Leftrightarrow cc^T\bullet X-2dc^Tx+d^2=0$
	Règle **LE2** $\sum_{j=1}^n c_jX_{ij}=dx_i\,\forall i\in\{1,...,n\}$ $c^Tx=d$

TABLE 1. *Règle **LE** pour les égalités linéaires*

Grâce aux propositions précédentes, on peut constater que les règles LE1 et LE2 conduiront à des relaxations semidéfinies de $P\left(\{0,1\}\right)$ équivalentes, mais en fonction du solveur utilisé, il sera plus judicieux de choisir l'une ou l'autre des relaxations.

On peut également montrer que dans le cas de multiples égalités (écrites sous la forme d'un système linéaire de p lignes $Ax=b$), on peut "agréger" le tout en une seule contrainte dans le programme semidéfini : $A^TA\bullet X-2b^TAx+b^Tb=0$, au lieu d'utiliser les règles LE1 ou LE2 sur chaque égalité.

Remarquons tout d'abord que pour (X,x) vérifiant cette contrainte et $X-xx^T\succeq 0$, on a $Ax=b$. En effet, $A^TA\succeq 0$ et $X-xx^T\succeq 0$ impliquent $A^TA\bullet\left(X-xx^T\right)\geq 0$, et donc $A^TA\bullet\left(X-xx^T\right)+(Ax-b)^2=0$ implique $Ax=b$. De plus on prouve que :

PROPOSITION 4.3.5. *Les régions* $R_1=\left\{(X,x):\ X-xx^T\succeq 0,\ A^tA\bullet X-2b^TAx+b^2=0\right\}$ *et* $R_2=\left\{(X,x):\ Ax=b,\ X-xx^T\succeq 0,\ \sum_{k=1}^n A_{jk}X_{ki}-b_jx_i=0\forall i\in\{1,...,n\}\,\forall j\in\{1,...,p\}\right\}$ *sont égales.*

DÉMONSTRATION. Soit (X,x) dans R_2. Pour chaque j, multiplions la contrainte $\sum_k A_{jk}X_{ki}-b_jx_i$ par A_{ji}, puis nous sommons le tout sur j et i. On obtient $A^TA\bullet X-b^TAx=0$. Puisque $Ax=b$, on a $A^TA\bullet X-2b^TAx+b^2=0$, donc (X,x) est dans R_1. Réciproquement, si (X,x) est dans R_1 alors $A^TA\bullet\left(X-xx^T\right)+(Ax-b)^2=0$. Donc on a $A^TA\bullet\left(X-xx^T\right)=0$ ce qui implique

$A^T A\left(X - xx^T\right) = 0$ puisque $A^T A$ and $X - xx^T$ sont positives. Ainsi, on a pour tout (r,i) dans $\{1,...,n\}^2$ $\sum_{k=1}^{n}\sum_{j=1}^{p} A_{jr}A_{jk}X_{ki} - x_i\sum_{k=1}^{n}\sum_{j=1}^{p} A_{jr}A_{jk}x_k = 0$. Puisque $\sum_{k=1}^{n}\sum_{j=1}^{p} A_{jr}A_{jk}x_k = \sum_{j=1}^{p} A_{jr}\sum_{k=1}^{n} A_{jk}x_k = \sum_{j=1}^{p} A_{jr}a_j^T x$, (où a_j est la $j$ème ligne de A) et $a_j^T x = b_j$ pour tout $j \in \{1,...,p\}$, on a pour tout (r,i) dans $\{1,...,n\}^2$ $\sum_{j=1}^{p} A_{jr}\left(\sum_{k=1}^{n} A_{jk}X_{ki} - b_j x_i\right) = 0$. Remarquons alors que $\sum_{k=1}^{n} A_{jk}X_{ki} - b_j x_i$ ne dépend pas de r, et donc nous avons ici une combinaison linéaire des p lignes de A. Puisqu'on peut supposer que le rang de A est p (sinon on supprime les contraintes redondantes), nous obtenons $\sum_{k=1}^{n} A_{jk}X_{ki} - b_j x_i = 0$ pour chaque j dans $\{1,...,p\}$ et i dans $\{1,...,n\}$. Donc (X,x) est dans R_2. □

Ici encore, suivant l'outil utilisé pour résoudre les programmes semidéfinis obtenus, il est plus efficace d'utiliser l'un ou l'autre des SDP (qui sont équivalents).

Inégalités linéaires.
Ici, nous supposons avoir à traiter une contrainte $d' \leq c^T x \leq d$. Ce n'est pas restrictif puisque si d ou d' sont absents, nous pouvons utiliser à la place $\sum_{i \text{ tel que. } c_i>0} c_i$ pour d et $\sum_{i \text{ tel que } c_i<0} c_i$ pour d' (chaque x_i est compris entre 0 et 1).

PROPOSITION 4.3.6. *Si (X,x) est admissible pour $(P\{0,1\})$ alors $cc^T \bullet X - (d+d')\,c^T x + dd' \leq 0$ est équivalent à $d' \leq c^T x \leq d$. Si $(X,x) \in S_n \times R^n$ n'est pas entier et est tel que $cc^T \bullet X - (d+d')\,c^T x + dd' \leq 0$ et $X - xx^T \succeq 0$, alors soit $cc^T\left(X - xx^T\right) = 0$ ou soit $d' < c^T x < d$ ou soit $c^T x \notin [d',d]$.*

DÉMONSTRATION. Premièrement, si $\left(X = xx^T, x\right)$ est admissible pour $(P\{0,1\})$ alors les deux racines de $cc^T \bullet X - (d+d')\,c^T x + dd'$ sont d et d' (et $\mathbf{d}(cc^T) \geq 0$). Deuxièmement, soit $(X,x) \in S_n \times R^n$ non-entier et tel que $cc^T \bullet X - (d+d')\,c^T x + dd' \leq 0$ et $X - xx^T \succeq 0$. On a $cc^T \bullet \left(X - xx^T\right) + \left(c^T x\right)^2 - (d+d')\,c^T x + dd' \leq 0$. Grâce au Lemme 1.1.9, nous avons $cc^T \bullet \left(X - xx^T\right) \geq 0$, et si (X,x) est tel que $cc^T \bullet \left(X - xx^T\right) = 0$ alors $cc^T\left(X - xx^T\right) = 0$. Dans l'autre cas, nous avons $cc^T \bullet \left(X - xx^T\right) > 0$ (x est donc non-entier). Si les deux racines $\frac{d'+d \pm \sqrt{(d'-d)^2 - 4cc^T \bullet (X - xx^T)}}{2}$ de $cc^T \bullet \left(X - xx^T\right) + \left(c^T x\right)^2 - (d+d')\,c^T x + dd'$ existent on obtient $d' < c^T x < d$, et si $(d'-d)^2 - 4cc^T \bullet \left(X - xx^T\right) < 0$ alors $c^T x \notin [d',d]$. □

PROPOSITION 4.3.7. *Si (X,x) est admissible pour $(P\{0,1\})$ alors pour tout i dans $\{1,...,n\}$ les contraintes*
$d'x_i \leq \sum_{j=1}^{n} c_j X_{ij} \leq dx_i$ et $d'(1-x_i) \leq \sum_{j=1}^{n} c_j(x_j - X_{ij}) \leq d(1-x_i)$ sont valides pour $(P\{0,1\})$. Si $(X,x) \in S_n \times R^n$ est tel que $X - xx^T \succeq 0$ et s'il existe i_0 dans $\{1,...,n\}$ tel que
$d'x_{i_0} \leq \sum_{j=1}^{n} c_j X_{i_0 j} \leq dx_{i_0}$ et $d'(1-x_{i_0}) \leq \sum_{j=1}^{n} c_j(x_j - X_{i_0 j}) \leq d(1-x_{i_0})$ alors
$d' \leq c^T x \leq d$.

DÉMONSTRATION. Premièrement, l'ensemble des ces $4n$ contraintes peut être obtenu en multipliant les inégalités $d' \leq c^T x \leq d$ par x_i ou par $(1-x_i)$ pour chaque i dans $\{1,...,n\}$. Donc ces contraintes sont valides pour $(P\{0,1\})$. La seconde partie de la proposition est obtenue simplement en sommant les deux inégalités. □

PROPOSITION 4.3.8. *Si $c_j \geq 0$ pour tout $j \in \{1,...,n\}$, $d' \leq 0$, et $(X,x) \in S_n \times R^n$ est tel que $X - xx^T \succeq 0$, alors : $\sum_{j=1}^{n} c_j X_{ij} \leq dx_i$ pour tout i dans $\{1,...,n\}$ et $\sum_{j=1}^{n} c_j(x_j - X_{i_0 j}) \leq d(1-x_{i_0})$ pour au moins un indice i_0 dans $\{1,...,n\}$ impliquent $cc^T \bullet X - (d'+d)\,c^T x + dd' \leq 0$.*

DÉMONSTRATION. Premièrement, $\sum_{j=1}^{n} c_j (x_j - X_{i_0 j}) \leq d(1 - x_{i_0})$ et $\sum_{j=1}^{n} c_j X_{i_0 j} \leq d x_{i_0}$ impliquent $c^T x \leq d$. Deuxièmement, nous pouvons multiplier $\sum_{j=1}^{n} c_j X_{ij} \leq d x_i$ par $c_i \geq 0$ pour chaque i dans $\{1, ..., n\}$ pour obtenir $\sum_{j=1}^{n} c_i c_j X_{ij} \leq d c_i x_i$, puis nous sommons ces n inégalités sur i pour obtenir $cc^T \bullet X - dc^T x \leq 0$. Par conséquent, nous obtenons $cc^T \bullet X - (d' + d) c^T x + dd' \leq d' \left(d - c^T x\right) \leq 0$. □

Remarquons que dans la Proposition 4.3.8, l'hypothèse "il existe i_0 dans $\{1, ..., n\}$ tel que $\sum_{j=1}^{n} c_j (x_j - X_{i_0 j}) \leq d(1 - x_{i_0})$" peut être remplacée par $c^T x \leq d$. Considérant ces résultats, nous proposons les règles suivantes pour traiter le cas des inégalités linéaires :

(P_L)	$(SDP\{0,1\})$
$d' \leq c^T x \leq d$	Règle **LI1** $cc^T \bullet X - (d + d') c^T x + dd' \leq 0$
	Règle **LI2** $\sum_{j=1}^{n} c_j X_{ij} \leq d x_i \quad i = 1, ..., n$ $d' x_i \leq \sum_{j=1}^{n} c_j X_{ij} \quad i = 1, ..., n$ $d'(1 - x_i) \leq \sum_{j=1}^{n} c_j (x_j - X_{ij}) \quad i = 1, ..., n$ $\sum_{j=1}^{n} c_j (x_j - X_{ij}) \leq d(1 - x_i) \quad i = 1, ..., n$

TABLE 2. *Règles **LI1** et **LI2** pour les inégalités linéaires*

4.3.3. Algorithme de construction d'une relaxation SDP à partir d'une relaxation PL quelconque. En utilisant les règles et résultats précédents, on peut proposer l'algorithme suivant pour construire facilement une relaxation SDP connaissant une relaxation linéaire (P_L) donnée pour $P(\{0,1\})$:

(1) Choisir une relaxation linéaire (P_L) pour le problème combinatoire $(P\{0,1\})$. Rappelons que X contient les variables de linéarisation présentes dans (P_L).

(2) Construire la relaxation semidéfinie comme suit :

   (a) Copier la fonction à optimiser.

   (b) Remplacer les contraintes $x \in [0,1]^n$ par $X - xx^T \succeq 0$ et $\mathbf{d}(X) = x$.

   (c) Copier les contraintes contenant des variables de linéarisation (*i.e.* X_{ij} $i, j \in \{1, ..., n\}$, $i \neq j$).

   (d) Appliquer la règle **LE1** ou la règle **LE2** aux contraintes d'égalités contenant niquement x.

   (e) Appliquer la règle **LI1** ou la règle **LI2** aux contraintes d'inégalités contenant uniquement x.

   (f) Retirer éventuellement les contraintes redondantes (i.e. impliquées par d'autres).

La relaxation semidéfinie obtenue sera toujours ***meilleure ou équivalente*** à la relaxation linéaire utilisée comme point de départ.

4.4. Résolution approchée de Max-cut

4.4.1. Définition et Introduction. Soit le problème combinatoire suivant :

MAX-CUT : Soit un graphe $G = (V, X)$ possédant n sommets et dont les arêtes $[v_i, v_j]$ sont valuées positivement par une matrice $W = (w_{ij})$. Trouver une partition des sommets de V en deux sous-ensembles (V_1, V_2) telle que la somme des poids des arêtes ayant leurs extrémités dans des paquets différents soit maximale.

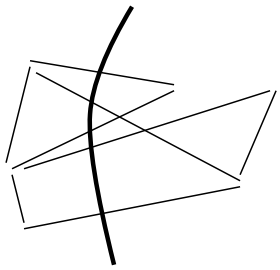


FIGURE 4.4.1. *La coupe vaut ici 5*

Ce problème a de très nombreuses applications. Malheureusement, il est NP-difficile, ce qui signifie qu'il n'existe très probablement pas d'algorithme de complexité polynomiale en la taille du problème (ici le nombre de sommets) permettant de le résoudre (sauf si $P = NP$). Néanmoins, de nombreuses méthodes permettent d'obtenir des solutions approchées (ou même optimales) pour des instances assez grandes (cf. le cours de Recherche Opérationnelle). En particulier, la programmation linéaire est un outil puissant pour obtenir des bornes et des solutions approchées de MAX-CUT.

Toutefois, il a été prouvé que dans le pire cas, ces relaxations issues de la (PL) pouvaient seulement fournir des bornes à 50% de la valeur optimale. Dans cette section, nous allons présenter succinctement un ensemble de résultats remarquables et récents (1995) obtenus par Michel X. Goemans et David P. Williamson. En utilisant la programmation semidéfinie, ces deux auteurs ont démontré qu'il est possible d'obtenir des solutions admissibles de MAX-CUT dont la valeur est à moins de 14% (au pire cas) de la valeur optimale. En fait, la pratique a montré que l'erreur moyenne se situait à moins de 3%.

4.4.2. Modélisation du problème. Avant de présenter l'algorithme évoqué précédemment, il nous faut modéliser notre problème. Pour cela, nous allons considérer des variables dites de "décision" notées y_i (i=1,...,n). Ces variables sont à valeurs dans $\{-1, 1\}$, et ont la signification suivante : si le sommet v_i est placé dans l'ensemble V_1 alors $y_i = -1$, sinon ($v_i \in V_2$) $y_i = 1$. Grâce à ces variables, notre problème peut être formulé comme suit :

$$\begin{cases} \text{Maximiser } \frac{1}{2} \sum_{i<j} w_{ij} \left(1 - y_i y_j\right) \\ \text{Sous les contraintes } : \ y_i \in \{-1, 1\} \end{cases} \tag{4.4.1}$$

Considérons alors la relaxation suivante

$$\begin{cases} \text{Maximiser } \frac{1}{2} \sum_{i<j} w_{ij} \left(1 - v_i^T v_j\right) \\ \text{Sous les contraintes } : \ v_i \in S^n \end{cases} \tag{4.4.2}$$

où S^n représente la sphère unité en dimension n. Remarquons que le programme 4.4.2 peut être vu comme un (SDP). En effet, il est possible de le formuler comme suit

$$\text{(4.4.3)} \qquad \begin{cases} \text{Maximiser } \frac{1}{2}\sum_{i<j} w_{ij}\,(1-Y_{ij}) = \frac{1}{2}W_{\text{tot}} - \frac{1}{4}W \bullet Y \\ \text{Sous les contraintes : } Y_{ii} = 1\,;\, i = 1,...,n \\ \qquad\qquad\qquad Y \succeq 0 \end{cases}$$

La matrice $Y = VV^T$ est la matrice de Gram des vecteurs $v_1, ..., v_n$. Réciproquement, à toute matrice satisfaisant les contraintes du (SDP) ci-dessus, on peut associer un champ de vecteurs de la sphère unité. La relaxation peut donc également être vue comme le passage d'une matrice de rang 1 à une matrice de rang inférieur à n. En fait, ce SDP correspond à la forme duale donnée au chapitre 3.

Remarquons qu'ici, l'absence de termes linéaires dans la fonction objectif nous permet d'éviter l'ajout d'une dimension à la matrice Y.

Pour écrire le dual de ce SDP, il suffit d'identifier les différentes constantes du problème. Ici, on a $c = [1, ..., 1]$, $A_0 = -W$, et A_i $(i \in \{1, ..., n\})$ est la matrice de S_n contenant un seul élément non nul : $A_{ii} = 1$. La formulation du dual est donc très simple :

$$\text{(4.4.4)} \qquad \begin{cases} \text{Minimiser } \frac{1}{2}W_{\text{tot}} + \frac{1}{4}\sum_{i=1}^{n} x_i \\ \text{Sous la contrainte : } \mathbf{diag}(x) + W \succeq 0 \end{cases}$$

On peut vérifier que ces deux problèmes sont strictement réalisables, et donc que $p^* = d^*$. Il est ainsi possible d'utiliser les méthodes de résolution vues au chapitre précédent. Nous avons donc obtenu une borne supérieure de notre problème initial que nous pourrions par exemple utiliser pour valider une heuristique ou dans une énumération "intelligente" (Branch&Bound).

4.4.3. Primalisation : construction d'une solution admissible pour le problème initial (avec garanties de performance). Dans cette section, nous allons montrer comment utiliser le champ de vecteurs obtenus en résolvant 4.4.3 (dont la matrice de Gram est Y) pour obtenir une solution admissible pour le problème combinatoire initial.

Soit l'algorithme Ω suivant :

1. Prendre un vecteur e_0 uniformément distribué sur S^n.

2. Pour chaque vecteur v_i $(i = 1, ..., n)$, si $v_i.e_0 \geq 0$ alors poser $y_i = 1$ sinon poser $y_i = -1$.

L'espérance de la solution y ainsi obtenue vaut $E[W] = \frac{1}{\pi}\sum_{i<j} w_{ij} \arccos(v_i.v_j)$

L'algorithme Ω revient à choisir un hyperplan passant par le centre de S^n (de manière aléatoire), et à placer dans le même paquet les sommets dont les vecteurs respectifs sont "du même côté" de l'hyperplan. Par linéarité de l'espérance on peut écrire que $E[W] = \sum_{i<j} w_{ij}\Pr[sign(v_i.e_0) \neq sign(v_j.e_0)]$. Remarquons que la probabilité que l'hyperplan sépare deux vecteurs est directement proportionnelle à l'angle séparant ces deux vecteurs, c'est-à-dire $\theta_{ij} = \arccos(v_i.v_j)$. Elle est égale à 0 si les deux vecteurs sont égaux, et à 1 si les vecteurs sont opposés. Ainsi on a $\Pr[sign(v_i.e_0) \neq sign(v_j.e_0)] = \frac{1}{\pi}\theta_{ij}$.

On a $E[W] \geq \alpha\frac{1}{2}\sum_{i<j} w_{ij}\,(1 - v_i.v_j)$,
où $\alpha = \min_{0\leq\theta\leq\pi} \frac{2}{\pi}\frac{\theta}{1-cos\theta} > 0,878$.

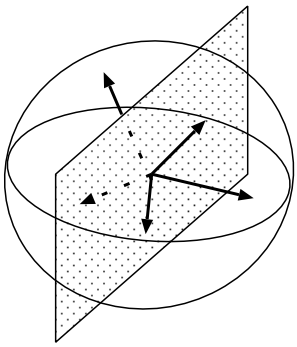


FIGURE 4.4.2. *L'hyperplan permet d'affecter les sommets selon la position des vecteurs*

Une simple étude de fonction permet de vérifier que pour $-1 \leq y \leq 1$ on a $\frac{\arccos(y)}{\pi} \geq \alpha \frac{1}{2}(1-y)$. En posant $y = v_i.v_j$, on obtient le résultat annoncé.

Les deux propositions précédentes nous permettent d'affirmer que la solution admissible de MAX-CUT construite par l'algorithme Ω est à moins de 14% de la solution optimale. En fait, la pratique a montré que l'erreur moyenne est inférieure à 3%. De plus, cette différence contient le saut dû à la relaxation, c'est pourquoi on peut affirmer que les solutions obtenues sont d'excellente qualité.

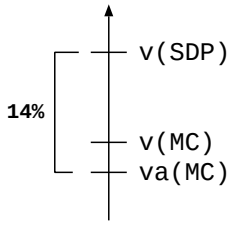


FIGURE 4.4.3. La solution construite de valeur $va(MC)$ est $0,878$-approchée, puisque $v(SDP) \geq v(MC)$

Dans la figure ci-dessus, $v(MC)$ désigne la valeur optimale de notre problème, $v(SDP)$ la valeur optimale de la relaxation semidéfinie, et $va(MC)$ la valeur de la solution admissible construite par la "primalisation" Ω .

4.4.4. Conclusion. On peut bien sûr obtenir une relaxation SDP équivalent à celle présentée précédemment en utilisant un modèle initial en $\{0,1\}$. En fait, le problème MAX-CUT est un cas particulier du problème QP présenté dans la Section 4.2.1. On peut reprendre tous les résultats établis précédemment pour ce problème. Néanmoins, on peut remarquer que la technique de primalisation utilisée pour obtenir l'algorithme $0,878$-approché utilise le caractère "géométrique" de l'approche $\{-1,1\}$.

4.5. Résolution approchée de Vertex-cover

4.5.1. Introduction. Soit $G = (V, E)$ un graphe non orienté. Un recouvrement de G par les sommets ou "vertex-cover" est un sous-ensemble S de sommets de G tel que pour chaque arête e dans E, au moins une des extrémités de e est dans S. Ainsi, un vertex-cover est le complément

d'un stable dans G (un stable est un sous-graphe de G ne possédant aucune arête). Pour un graphe dont les sommets i sont valués par un poids w_i, trouver le vertex-cover de poids minimal est un problème combinatoire NP-difficile classique. Il existe de très nombreuses applications de ce problème : par exemple en conception de réseaux de télécommunication. Il est donc très intéressant de pouvoir le résoudre efficacement (même de manière approchée).

4.5.2. Formulation du problème comme un programme linéaire en nombres entiers (PLNE). Considérons les variables de décision x_i $(i = 1, ..., n)$ (n est le nombre de sommets de G). Nous poserons $x_i = 1$ si le sommet i est dans S et $x_i = 0$ sinon. On peut alors formuler notre problème comme suit :

$$(VC) \begin{cases} \text{Minimiser } \sum_{i=1}^{n} w_i x_i \\ \text{Sous les contraintes } : \; x_i + x_j \geq 1 \; (i,j) \in E \\ \qquad\qquad\qquad\qquad x_i \in \{0,1\} \;\; i = 1, ..., n \end{cases} \tag{4.5.1}$$

4.5.3. Une relaxation par PL. Une première relaxation possible peut être le relâchement des contraintes d'intégrité des variables x_i. On obtient alors le programme linéaire continu suivant que l'on sait résoudre efficacement par la méthode du simplexe ou par une méthode de point intérieur :

$$(VC)_L \begin{cases} \text{Minimiser } \sum_{i=1}^{n} w_i x_i \\ \text{Sous les contraintes } : \; x_i + x_j \geq 1 \; (i,j) \in E \\ \qquad\qquad\qquad\qquad 0 \leq x_i \leq 1 \; i = 1, ..., n \end{cases} \tag{4.5.2}$$

On peut démontrer que cette relaxation fournit au pire cas une solution $lp(G)$ au moins égale à la moitié de la valeur optimale de (VC) (en fixant à 1 les variables de la solution optimale de $(VC)_L$ supérieures ou égales à $\frac{1}{2}$ et à 0 les autres). Ce résultat n'est pas très bon, mais en moyenne et en pratique on peut espérer obtenir de meilleures bornes. Toutefois, l'utilisation d'une telle borne dans un "Branch&Bound" reste limitée à des instances de taille modérée.

4.5.4. Première utilisation de la SDP : approche $\{-1, 1\}$. Afin d'obtenir une relaxation plus fine, nous pouvons à présent utiliser la programmation semidéfinie. Pour cela, nous allons considérer une formulation en programme quadratique de notre problème. Soit une variable y_0 à valeur dans $\{-1, 1\}$et les variables de décision y_i $(i = 1, ..., n)$ telles que $y_i = y_0$ si le sommet i est dans S, et $y_i = -y_0$ sinon. On peut alors reformuler (VC) de la manière suivante :

$$(VC)_Q \begin{cases} \text{Minimiser } \frac{1}{2} \sum_{i=1}^{n} w_i \, (1 + y_0 y_i) \\ \text{Sous les contraintes } : \; (y_0 - y_i)^T (y_0 - y_j) = 0 \, ; \, (i,j) \in E \\ \qquad\qquad\qquad\qquad y_i \in \{-1, 1\} \; ; \, i = 0, ..., n \end{cases} \tag{4.5.3}$$

Relâchons alors ce programme quadratique en $\{-1, 1\}$ en autorisant les variables y_i $(i = 0, ..., n)$ à être des vecteurs de $\mathbb{R}^{n+1}$ de norme 1 (donc de la sphère unité). Il s'agit donc d'une relaxation sur la dimension des variables, puisqu'initialement les y_i appartenaient à la sphère unité de $\mathbb{R}$ c'est-à-dire $\{-1, 1\}$. Notre relaxation peut s'écrire :

$$(4.5.4) \qquad (SD) \begin{cases} \text{Minimiser } \frac{1}{2}\sum_{i=1}^{n} w_i\,(1+y_0y_i) \\ \text{Sous les contraintes } : \ (y_0-y_i)^T(y_0-y_j)=0\,;\,(i,j)\in E \\ \qquad\qquad\qquad\qquad\quad y_i^2=1\,;\,i=0,...,n \\ \qquad\qquad\qquad\qquad\quad y\in\mathbb{R}^{n+1} \end{cases}$$

Les contraintes $(y_0-y_i).(y_0-y_i)=0$ peuvent être interprétées géométriquement en disant que le point $\frac{1}{2}(y_i+y_j)$ doit être dans la sphère de centrée en $\frac{1}{2}y_0$ et de rayon $\frac{1}{2}$, i.e. que $\left(\frac{y_i+y_j-y_0}{2}\right)^2=\frac{1}{4}$. Cette relaxation peut être vue comme un programme semidéfini. En effet, (SD) peut être formulée comme suit :

$$(4.5.5) \qquad (SDP1)_{VC} \begin{cases} \text{Minimiser } \frac{1}{2}\sum_{i=1}^{n} w_i\,(1+Y_{0i}) \\ \text{Sous les contraintes } : \ Y_{00}+Y_{ij}-Y_{0i}-Y_{0j}=0\,;\,\forall(i,j)\in E \\ \qquad\qquad\qquad\qquad\quad Y_{ii}=1\,;\,i=0,...,n \\ \qquad\qquad\qquad\qquad\quad Y\succeq 0 \end{cases}$$

En effet, rappelons qu'une matrice de positive symétrique de dimension $n+1$ dont tous les éléments de la diagonale valent 1 est la matrice de Gram d'un champ de vecteurs de la sphère unité S^{n+1} (et réciproquement).

4.5.5. Une deuxième relaxation SDP : approche $\{0,1\}$. Nous allons construire une autre relaxation SDP en appliquant les règles de la Section 4.3 en partant de $(VC)_L$:

$$(4.5.6) \qquad (SDP2)_{VC} \begin{cases} \text{Minimiser} & w^Tx \\ \text{Sous les contraintes } : & [A]\ X_{ij}+1=x_i+x_j \quad \forall(i,j)\in E \\ & [B]\ X_{ij}+1\geq x_i+x_j \quad \forall(i,j)\in V^2-E \\ & \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix}\succeq 0 \end{cases}$$

Pour appliquer la règle LI1 (Table 2) , nous avons besoin de disposer d'une borne pour chaque contrainte $x_i+x_j\geq 1$, il est facile de voir que 2 convient. Ainsi, on obtient la contrainte $2X_{ij}+X_{ii}+X_{jj}-3\,(x_i+x_j)+2\leq 0$, *i.e.* $X_{ij}+1\leq x_i+x_j$. Mais nous savons également que l'inégalité triangulaire $X_{ij}+1\geq x_i+x_j$ (utile pour linéariser un programme quadratique) est valable pour tout couple $(i,j)\in V^2$. C'est pourquoi on obtient la contrainte $X_{ij}+1=x_i+x_j\ \forall(i,j)\in E$ dans $(SDP2)_{VC}$. Nous laissons le soin au lecteur de vérifier que ces contraintes d'égalité sont les transformées (voir Section 4.2.5) des contraintes $Y_{00}+Y_{ij}-Y_{0i}-Y_{0j}=0\,;\,\forall(i,j)\in E$ dans $(SDP1)_{VC}$. $(SDP2)_{VC}$ sans les contraintes $[B]$ est donc équivalent à $(SDP1)_{VC}$, car leurs fonctions sont également équivalentes en utilisant le même changement de variable. Pour le problème que nous allons à présent traiter (K-CLUSTER) nous détaillerons plus les calculs prouvant l'équivalence des deux approches.

4.6. Résolution approchée de k-cluster

4.6.1. Introduction. Considérons un graphe orienté $G=(V,E)$ de n sommets $(v_1,...,v_n)$, dont chaque arête $[v_i,v_j]$ est valuée par un poids w_{ij}. Le problème K-CLUSTER consiste à trouver un sous-ensemble V' de k sommets (k appartient donc à $\{2,...,n-1\}$) qui maximise la somme

des poids des arêtes qui ont leurs deux extrémités dans V'. Ce problème contient en particulier le célèbre problème de la recherche de la clique de taille maximale dans un graphe (graphe complet contenant un nombre maximal de sommets). Ce problème combinatoire se retrouve dans de nombreux domaines : pagination de la mémoire, conception de réseaux de télécommunication, conception de circuits intégrés.

4.6.2. Formulation en $\{-1,1\}$. Ici encore, on peut représenter notre problème comme un programme quadratique en variables à valeurs dans $\{-1,1\}$ comme suit :

$$(KC)\begin{cases} \text{Maximiser} & f(x) = \frac{1}{8}\sum_{i=1}^{n}\sum_{j\neq i} W_{ij}(1+y_0y_i)(1+y_0y_j) \\ \text{Sous les contraintes :} & \sum_{i=1}^{n} y_0y_i = 2k-n \\ & y \in \{-1,1\}^{n+1} \end{cases}$$

où $y_0y_i = 1$ si et seulement si le sommet v_i est placé dans V'. On peut vérifier que le terme $\frac{1}{4}(1+y_0y_i)(1+y_0y_j)$ vaut 1 lorsque v_i et v_j sont placés dans V', et 0 sinon. La contrainte $\sum_{i=1}^{n} y_0y_i = 2k-n$ impose que k variables valent 1 et $n-k$ valent -1 (ce qui traduit bien que $|V'| = k$).

4.6.3. Relaxation semidéfinie $\{-1,1\}$. Comme pour les problèmes précédents, relâchons à présent (P) en laissant les variables y_i devenir des vecteurs de la sphère unité S^{n+1}. Chaque produit y_0y_i devient alors le produit scalaire $u_i^T u_j$. Ce qui nous donne le programme suivant :

$$(\underline{KC})\begin{cases} \text{Maximiser} & f(x) = \frac{1}{8}\sum_{i=1}^{n}\sum_{j\neq i} W_{ij}(1+u_0^T u_i)(1+u_0^T u_j) \\ \text{Sous les contraintes :} & \sum_{i=1}^{n} u_0^T u_i = 2k-n \\ & u_i \in S^{n+1} \end{cases}$$

Ce problème peut être vu comme un programme semidéfini en remplaçant chaque terme $u_i.u_j$ par Y_{ij} terme d'une matrice positive symétrique dont les éléments diagonaux $Y_{ii} = u_i^2$ valent tous 1.

$$(KC\,SDP)_{\{-1,1\}}\begin{cases} \text{Maximiser} & f(x) = \frac{1}{8}\sum_{i=1}^{n}\sum_{j\neq i} W_{ij}(1+Y_{0i})(1+Y_{0j}) \\ \text{Sous les contraintes :} & \sum_{i=1}^{n} Y_{0i} = 2k-n \\ & Y_{ii} = 1 \;\; i = 0,...,n \\ & Y \succeq 0 \end{cases}$$

Il est possible de montrer que les bornes obtenues par cette relaxation dans le pire des cas sont bien meilleures que celles obtenues par une simple relaxation par PL.

4.6.4. Primalisation des solutions de $(KC\,SDP)_{\{-1,1\}}$. Nous pouvons tenter d'appliquer la même méthode que celle employée pour le problème MAX-CUT :

(1) Résoudre $(KC\,SDP)_{\{-1,1\}}$, et obtenir un champ de vecteurs $v_0,...,v_n$ par une décomposition de Cholesky de la solution Y^*obtenue.

(2) Choisir aléatoirement un vecteur e_0 de S^{n+1} et affecter les sommets en fonction de leur position relative à l'hyperplan défini par e_0.

Mais ici, nous n'avons a priori aucune certitude que la contrainte initiale $\sum_{i=1}^{n} y_0y_i = 2k-n$ sera bien respectée. Néanmoins, on peut obtenir une solution admissible du problème initial en considérant les trois cas suivants :

(1) Si la contrainte est respectée, on ne fait évidemment rien.

(2) Si on a $\sum_{i=1}^{n} y_0 y_i < 2k - n$, alors il suffit d'ajouter le nombre de sommets manquants (même au hasard), car la valeur de la solution ainsi obtenue sera forcément améliorée.

(3) Si on a $\sum_{i=1}^{n} y_0 y_i > 2k - n$, alors il faut enlever des sommets de V'. Cela peut être réalisé par l'heuristique "gourmande" suivante : choisir le sommet dont la somme des arêtes l'ayant pour extrémité est minimale, l'enlever de V', recommencer la même procédure sur le graphe réduit. On peut démontrer que cette méthode ne diminue pas trop la valeur de la solution initiale (obtention d'une garantie de performance).

On constate ici que pour des problèmes ayant des contraintes plus nombreuses et complexes, la "primalisation" est plus délicate. Néanmoins la SDP fournira toujours au moins une borne.

4.6.5. Relaxation SDP $\{0,1\}$. Une autre modélisation possible de notre problème est

$$(KC)_{\{0,1\}} \begin{cases} \text{Maximiser} & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}x_i x_j \\ \text{Sous contraintes} & \sum_{i=1}^{n} x_i = k \\ & x \in \{0,1\}^n \end{cases}$$

Mettons sous forme matricielle ce programme : notons X la matrice xx^T, W la matrice des poids et e_n le vecteur de dimension n composé uniquement de '1'. Le problème devient alors :

$$(KC)_{\{0,1\}} \begin{cases} \text{Max} & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}X_{ij} = \frac{1}{2}W \bullet X \\ \text{s.c. :} & \sum_{i=1}^{n} x_i = k \\ & X = xx^T \\ & x \in \{0,1\}^n \end{cases}$$

La relaxation SDP est alors directement obtenue par

$$(KC\,SDP)_{\{0,1\}} \begin{cases} \text{Maximiser} & \frac{1}{2}W \bullet X \\ \text{Sous contraintes} & e_n^T x = k \\ & \mathbf{d}(X) = x \\ & X' = \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0 \end{cases}$$

4.6.6. Lien entre les deux relaxations SDP. Les relaxations $(KC\,SDP)_{\{0,1\}}$ et $(KC\,SDP)_{\{-1,1\}}$ sont-elles équivalentes ? Nous allons répondre à cette question en utilisant notre automorphisme ϕ (voir Section 4.2.5).

4.6.6.1. *Traitement de la contrainte d'égalité.* Dans la modélisation SDP en $\{-1,1\}$ du problème k-cluster, on a la contrainte $\sum_{i=1}^{n} Y_{0i} = 2k - n$ qui peut s'écrire $A \bullet Y = 4k - 2n$ avec $A = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & & & \\ \vdots & & 0 & \\ 1 & & & \end{bmatrix}$.

Appliquons l'automorphisme ϕ^{-1} à A pour calculer la contrainte correspondante dans la relaxation SDP en $\{0,1\}$:

$$\begin{aligned}
\phi^{-1}(A) = \left(Q^{-1}\right)^T A Q^{-1} &= \begin{bmatrix} 1 & -e_n \\ 0 & 2I_n \end{bmatrix} \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & & & \\ \vdots & & 0 & \\ 1 & & & \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -e_n & 2I_n \end{bmatrix} \\
&= \begin{bmatrix} -n & 1 & \cdots & 1 \\ 2 & & & \\ \vdots & & 0 & \\ 2 & & & \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -e_n & 2I_n \end{bmatrix} \\
&= \begin{bmatrix} -2n & 2 & \cdots & 2 \\ 2 & & & \\ \vdots & & 0 & \\ 2 & & & \end{bmatrix}
\end{aligned}$$

La contrainte recherchée est donc $\phi^{-1}(A) \bullet \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} = -2nX'_{00} + 4\sum_{i=1}^n X'_{i0} = 4k - 2n.$ Rappelons que $X_{00} = 1$ et que $\mathbf{d}(X) = x$. On obtient donc bien $\sum_{i=1}^n X_{ii} = k$

4.6.6.2. *Traitement de la fonction objectif.* La fonction économique de $(KC\,SDP)_{\{-1,1\}}$ est un cas particulier de la fonction objectif de $(SDP)_{QP\{-1,1\}}$ où $b = 0$ et où les éléments de C sont positifs. Le calcul est donc simplifié. Ici $f(Y) = \frac{1}{8}\sum_{i=1}^n \sum_{j=1}^n W_{ij}\,(1 + Y_{0i} + Y_{0j} + Y_{ij})$ peut s'écrire $f(x) = \frac{1}{4}W_{tot} + \frac{1}{8}\sum_{i=1}^n \sum_{j=1}^n W_{ij}\,(1 + Y_{0i} + Y_{0j} + Y_{ij})$. Sous forme matricielle, on obtient $f(x) = \frac{1}{4}W_{tot} + \frac{1}{8}D \bullet Y$

où $D = \begin{bmatrix} 0 & \sum_{j=1}^n W_{1j} & \cdots & \sum_{j=1}^n W_{ij} & \cdots & \sum_{j=1}^n W_{nj} \\ \sum_{j=1}^n W_{1j} & & & & & \\ \vdots & & & & & \\ \sum_{j=1}^n W_{ij} & & & W & & \\ \vdots & & & & & \\ \sum_{j=1}^n W_{nj} & & & & & \end{bmatrix}$.

Appliquons l'opérateur ϕ^{-1} à D pour déterminer l'expression de la fonction objectif pour le modèle en $\{0,1\}$:

$$
\begin{aligned}
\phi^{-1}(D) &= \left(Q^{-1}\right)^T D Q^{-1} \\
&= \begin{bmatrix} 1 & -e_n \\ 0 & 2I_n \end{bmatrix} \begin{bmatrix} 0 & \sum_{j=1}^n W_{1j} & \cdots & \sum_{j=1}^n W_{ij} & \cdots & \sum_{j=1}^n W_{nj} \\ \sum_{j=1}^n W_{1j} & & & & & \\ \vdots & & & & & \\ \sum_{j=1}^n W_{ij} & & & W & & \\ \vdots & & & & & \\ \sum_{j=1}^n W_{nj} & & & & & \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -e_n & 2I_n \end{bmatrix} \\
&= \begin{bmatrix} -\sum_{i=1}^n \sum_{j=1}^n W_{ij} & 0 & \cdots & 0 & \cdots & 0 \\ 2\sum_{j=1}^n w_{1j} & & & & & \\ \vdots & & & & & \\ 2\sum_{j=1}^n w_{ij} & & & 2W & & \\ \vdots & & & & & \\ 2\sum_{j=1}^n w_{nj} & & & & & \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -e_n & 2I_n \end{bmatrix} \\
&= \begin{bmatrix} -2W_{tot} & 0 & \cdots & 0 & \cdots & 0 \\ 0 & & & & & \\ \vdots & & & & & \\ 0 & & & 4W & & \\ \vdots & & & & & \\ 0 & & & & & \end{bmatrix}
\end{aligned}
$$

Nous obtenons donc comme transformée de notre fonction objectif :

$$
\begin{aligned}
\frac{1}{4}W_{tot} + \frac{1}{8}\phi^{-1}(C) \bullet X' &= \frac{1}{4}W_{tot} + \frac{1}{8}\left(-2W_{tot} + 4\sum_{i=1}^n \sum_{j=1}^n W_{ij}X_{ij}\right) \\
&= \frac{1}{2}W \bullet X
\end{aligned}
$$

Nous retrouvons bien l'expression de la fonction objectif du programme $(KC\,SDP)_{\{0,1\}}$.

4.6.7. Amélioration des relaxations précédentes. On peut obtenir une meilleure relaxation SDP que celles précédemment présentées en partant d'une relaxation PL existante. Pour cela, nous considérons la relaxation PL :

$$
(PL)_{KC} \begin{cases} \text{Max} & \frac{1}{2}W \bullet X & \\ \text{s.c. :} & & \\ (a_3) & \sum_{i=1}^n x_i = e_n^T x = k & \\ (b_3) & \sum_{i<j} X_{ij} + \sum_{j>i} X_{ij} = (k-1)x_i & i \in \{1,...,n\} \\ (c_3) & X_{ij} \geq 0 & i < j \notin E \\ (d_3) & X_{ij} \leq x_i\,;\, X_{ij} \leq x_j & i < j \notin E \\ & 0 \leq x_i \leq 1 & i \in \{1,...,n\} \end{cases}
$$

et nous appliquons les règles données à la Section 4.3. En ce qui concerne les contraintes impliquant des termes X_{ij}, nous nous contentons de les recopier. Ainsi, nous obtenons le SDP

suivant :

$$(SDP)_{KC}\begin{cases} \text{Max} & \frac{1}{2}W \bullet X & \\ \text{s.t.} & & \\ (a'_3) & e_n^T x = k\,,\, e_n e_n^T \bullet X = k^2 & \\ (b'_3) & \sum_{i=1}^n X_{ij} = k x_j & \forall j \in \{1,...,n\} \\ (c'_3) & X_{ij} \geq 0 & i < j \in \{1,...,n\} \\ (d'_3) & X_{ij} \leq x_i\,;\, X_{ij} \leq x_j & i < j \in \{1,...,n\} \\ & \mathbf{d}(X) = x & \\ & \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0 & \end{cases}$$

On peut constater que ce SDP est meilleur que $(KC\,SDP)_{\{0,1\}}$ et $(KC\,SDP)_{\{-1,1\}}$ (qui sont équivalents), puisqu'il contient toutes les contraintes de $(KC\,SDP)_{\{0,1\}}$ plus d'autres. Ceci illustre l'intérêt qu'il y a à partir d'une relaxation PL existante pour élaborer des relaxations SDP.

4.7. Bornes SDP pour le Knapsack (sac-à-dos) quadratique

4.7.1. Introduction. Considérons le problème quadratique suivant dit du sac-à-dos (knapsack) :

$$(KQ)\begin{cases} \text{Maximiser} & y^T C y \\ \text{s.c. :} & a^T y \leq b \\ & y \in \{0,1\}^n \end{cases}$$

Ce problème a de très nombreuses applications pratiques.

4.7.2. Première relaxation SDP. On peut appliquer simplement le modèle $\{0,1\}$ (voir Section 4.2.2.) pour obtenir

$$(SDP_1)\begin{cases} \text{Maximiser} & C \bullet Y = Tr\,(CY) \\ \text{s.c. :} & \mathbf{diag}(a) \bullet Y = a^T y \leq b \\ & Y - yy^T \succeq 0 \end{cases}$$

4.7.3. Une deuxième relaxation SDP de (KQ). Nous appliquons ici directement à la relaxation PL standard de (KQ) la recette vue à la Section 4.3 :

$$(SDP_2)\begin{cases} \text{Maximiser} & C \bullet Y \\ \text{s.c. :} & aa^T \bullet Y \leq b a^T y \\ & Y - yy^T \succeq 0 \end{cases}$$

Pour cela, nous avons remarqué que $0 \leq a^T y \leq b$ et appliqué la règle LI1 (voir la Table 2). Les résultats de la Section ona sait que (SDP_2) est une meilleure relaxation que (SDP_1).

4.7.4. Une troisième relaxation SDP de (KQ). Nous pouvons encore faire mieux en utilisant à présent la règle LI2 (voir Section 4.3.2). On multiplie donc chaque inégalité $a^T y \leq b$ par y_i ou $(1 - y_i)$ pour la rendre quadratique. Pour i fixé, on obtient :

$$\textstyle\sum_{j=1}^n a_j y_{ij} \leq b y_i \text{ et } \sum_{j=1}^n a_j\,(y_j - y_{ij}) \leq b\,(1 - y_i)$$

Remarquons qu'en sommant ces deux inégalités, on obtient : $a^T y \leq b$, donc la relaxation SDP obtenue est une meilleure relaxation que (SDP_1). Pour des raisons pratiques de taille, on peut se

contenter de conserver une seule contrainte obtenue en multipliant par $1-y_i$ (donc en prenant un i particulier). Par exemple, on peut considérer le SDP suivant (nous avons pris $i=1$) :

$$(SDP_3)\begin{cases} \text{Max} & C \bullet Y = Tr\left(CY\right) \\ \text{s.c. :} & \sum_{j=1}^{n} a_j y_{ij} \leq b y_i \;\; i=1,...,n \\ & \sum_{j=1}^{n} a_j \left(y_j - y_{1j}\right) \leq b\left(1-y_1\right) \\ & Y - yy^T \succeq 0 \end{cases}$$

Cette dernière relaxation fournie des bornes d'une très grande qualité (moins de 1% de l'optimum en moyenne).

4.8. Borne SDP pour le problème Max-2SAT

4.8.1. Introduction. Soit un ensemble de variables booléennes $X = \{x_1, ..., x_n\}$ et un ensemble de clauses $X = \{C_1, ..., C_m\}$ comportant au plus deux littéraux. On cherche à fixer les variables à "vrai" ou "faux" de façon à maximiser le nombre de clauses satisfaites.

EXEMPLE. $X = \{x_1, x_2\}$, $C_1 = x_1 \vee \overline{x}_2$, $C_2 = x_2$, $C_3 = \overline{x}_1 \vee \overline{x}_2$. On fixe x_1 à "vrai" et x_2 à faux : C_1 et C_3 sont satisfaites, pas C_2.

4.8.2. Relaxation SDP du problème. Notre objectif est d'écrire Max-2SAT sous la forme :

$$\begin{cases} \text{Max } \sum_{i<j} \left[a_{ij}\left(1-y_i y_j\right) + b_{ij}\left(1+y_i y_j\right)\right] \\ \text{Sous les contraintes : } y_i \in \{-1, 1\} \end{cases}$$

Pour cela on ajoute une variable supplémentaire y_0 afin d'exprimer les valeurs des différents types de clauses. On peut alors écrire :

$$\begin{aligned} v\left(x_i \vee x_j\right) &= 1 - v\left(\overline{x}_i \wedge \overline{x}_j\right) \\ &= 1 - v\left(\overline{x}_i\right) v\left(\overline{x}_j\right) \\ &= 1 - \frac{1-y_0 y_i}{2}\frac{1-y_0 y_j}{2} \\ &= \frac{1}{4}\left(3 + y_0 y_i + y_0 y_j - y_0^2 y_i y_j\right) \\ &= \frac{1+y_0 y_i}{4} + \frac{1+y_0 y_j}{4} - \frac{1+y_i y_j}{4} \end{aligned}$$

Pour les autres types de clauses on fait le même type de calcul en remplaçant simplement y_i par $-y_i$ si on a $\overline{x}_i$ à la place de x_i. Ici, on utilise la même primalisation que pour le problème MAX-CUT. On peut alors démontrer que dans le pire des cas, on obtient une solution à moins de 9% de la solution optimale. En pratique, la moyenne d'erreur est bien inférieure : moins de 1%.

4.9. Borne SDP pour le nombre chromatique d'un graphe

4.9.1. Formulation du problème en PLNE. Le calcul du nombre chromatique d'un graphe est un problème combinatoire classique et très difficile à résoudre. Etant donné un graphe $G = (V, E)$ non orienté et non valué, on cherche à "colorier" les sommets de G de façon à ce que deux sommets adjacents n'aient pas la même couleur. Le problème est bien entendu d'utiliser un nombre minimal de couleurs.

Soit les variables de décision y_k $(k = 1, ..., n)$ définies de la manière suivante : $y_k = 1$ si on utilise la k^{ieme} couleur. Ainsi si k_0 couleurs sont utilisées alors les variables $y_{k_0+1}, ...y_n$ seront

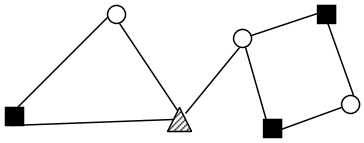


FIGURE 4.9.1. *On a besoin ici de trois couleurs*

de valeur nulle. On définit également les variables booléennes x_{ik} : $x_{ik} = 1$ si et seulement si le sommet v_i est colorié avec la couleur k. Nous devons alors traduire les contraintes suivantes :

- Chaque sommet doit être colorié avec une seule couleur.
- Les extrémités d'une arête doivent être coloriées de deux couleurs distinctes.
- Pour chaque couleur k, $y_k = \max_{1 \leq i \leq n} x_{ik}$.

On obtient ainsi une formulation de notre problème comme programme linéaire en nombres entiers :

$$(CH) \begin{cases} \text{Minimiser} & \sum_{k=1}^{n} y_k \\ \text{Sous les contraintes :} & \sum_{k=1}^{n} x_{ik} = 1 \ ; \ i = 1, ..., n \\ & x_{ik} + x_{jk} \leq 1 \ ; \ (i,j) \in E \, ; \, k = 1, ..., n \\ & x_{ik} \leq y_k \ ; \, k = 1, ..., n \, ; \, i = 1, ..., n \\ & y \in \{0,1\}^n \ ; \, x \in \{0,1\}^{n^2} \end{cases}$$

Remarquons que puisque nous cherchons à minimiser $\sum_{k=1}^{n} y_k$, les contraintes $x_{ik} \leq y_k$ impliquent bien qu'à l'optimum on aura $y_k = \max_{1 \leq i \leq n} x_{ik}$.

4.9.2. Relaxation continue du PLNE. Une première approche consiste à relâcher les contraintes d'intégrité du programme (CH) afin d'obtenir un programme linéaire continu $(\underline{CH})$. Mais malheureusement, la borne obtenue est très mauvaise. En effet, la solution $x_{ik} = y_k = \frac{1}{n} (i = 1, ..., n \, ; \, k = 1, ..., n)$ admissible de $(\underline{CH})$ nous donne une valeur égale à $\sum_{k=1}^{n} y_k = 1$ (ce qui n'est pas très utile ...). Bien sûr, il est possible d'améliorer cette borne en ajoutant d'autres inégalités valides pour le programme (CH), mais ce résultat montre que la PL ne va probablement pas nous fournir des bornes exploitables.

4.9.3. Relaxation par un SDP. Nous allons considérer à présent une formulation de notre problème en un programme en nombres entiers particulier, afin de pouvoir le relâcher plus facilement en SDP par la suite. Considérons le programme suivant :

$$(CH2) \begin{cases} \text{Minimiser} & k \\ \text{Sous les contraintes :} & Y_{ij} = -\frac{1}{k-1} \, ; \, \forall (i,j) \in E \\ & Y_{ii} = 1 \, ; \, i = 1, ..., n \\ & Y_{ij} \in \left\{ -\frac{1}{k-1}, 1 \right\} \, ; \, i, j = 1, ..., n \\ & Y \succeq 0 \\ & k \in \mathbb{N}^* \end{cases}$$

On peut qualifier $(CH2)$ de "SDP en nombres entiers". Il est possible de démontrer que $(CH2)$ est une formulation du problème du nombre chromatique d'un graphe, en utilisant les deux lemmes suivants :

Soit un entier $1 < k \leq n$, alors il existe une famille de k vecteurs de S^n tels que $u_i^T u_j \leq -\frac{1}{k-1}$ $(i, j = 1, ..., k \, ; \, i \neq j)$. De plus, $-\frac{1}{k-1}$ est la valeur minimale pour laquelle il existe une telle famille.

Soit un entier $1 < k \leq n$, et $u_1, ..., u_n$ une famille de vecteurs de S^n telle que $u_i^T u_j \in \left\{-\frac{1}{k-1}, 1\right\}$ $(i, j = 1, ..., n)$, alors il y a au plus k vecteurs distincts parmi les n.

Pour visualiser ces résultats, on peut considérer les cas particuliers en dimensions 1, 2 et 3 :

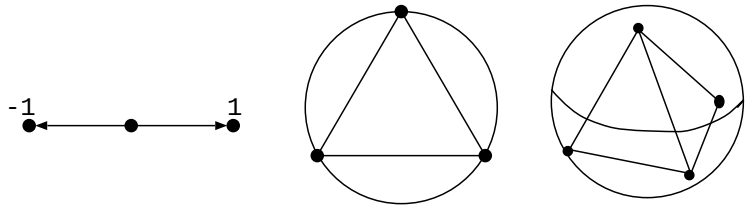


FIGURE 4.9.2. *En dimension 3, on obtient un tétraèdre régulier inscrit dans S^3.*

Une relaxation possible en SDP de $(CH2)$ est :

$$(CHSDP) \begin{cases} \text{Minimiser} & k \\ \text{Sous les contraintes :} & Y_{ij} = -\frac{1}{k-1} \,;\, \forall (i,j) \in E \\ & Y_{ii} = 1 \,;\, i = 1, ..., n \\ & -1 \leq Y_{ij} \leq 1 \,;\, i, j = 1, ..., n \\ & Y \succeq 0 \\ & k \in \mathbb{R}^{+*} \end{cases}$$

Il est possible d'obtenir des garanties de performance au pire cas à partir de ce SDP. Même si ces dernières ne sont pas aussi bonnes que pour MAX-CUT par exemple, elles permettent d'obtenir d'assez bonnes solutions admissibles, notamment en "primalisant" la solution optimale du SDP.

4.9.4. Primalisation. On peut tenter de construire une k'-coloration pour le graphe initial en découpant par exemple la sphère unité S^n en k' zones et en affectant une même couleur aux vecteurs se trouvant dans la même zone. Toutefois, cette technique ne permet pas de garantir une vraie coloration. En effet, deux sommets adjacents peuvent être coloriés de la même couleur. On parle alors de *semi-coloration*. Cette technique peut être répétée jusqu'à obtention d'une solution admissible (en considérant le sous-graphe constitué des arêtes dont les extrémités sont de la même couleur). On peut malgré tout obtenir des garanties de performance, en calculant la probabilité que deux vecteurs unitaires dont le produit scalaire vaut $-\frac{1}{k-1}$ soient coloriés d'une même couleur. Ici encore, on peut constater que la primalisation est délicate car le nombre de contraintes initiales du problème combinatoire rend la construction d'une solution admissible plus complexe.

CHAPITRE 5

Exercices

5.1. Propriétés de la région admissible $X = \{x \mid F(x) \succeq 0\}$

Montrer que X est fermé dans $\mathbb{R}^n$

Montrer que $\partial X = \{x \mid F(x) \succeq 0, \det F(x) = 0\}$ et que $\overset{\circ}{X} = \{x \mid F(x) \succ 0\}$

5.2. Dualité

Soit le problème

$$\begin{cases} \text{Minimiser} & t \\ \text{Sous contrainte} & \begin{bmatrix} x & 1 \\ 1 & t \end{bmatrix} \succeq 0 \end{cases}$$

(1) Calculer p^*. Que vaut X_{opt} ?

(2) Ecrire le dual, calculer d^*, et donner Z_{opt}.

(3) Peut-on expliquer ces résultats en appliquant le théorème de Nesterov-Nemirovsky ?

Soit le programme semidéfini suivant

$$\begin{cases} \text{Minimiser} & x_1 \\ \text{Sous contrainte} & \begin{bmatrix} 0 & x_1 & 0 \\ x_1 & x_2 & 0 \\ 0 & 0 & x_1+1 \end{bmatrix} \succeq 0 \end{cases}$$

(1) Quelle est la région admissible du problème ?

(2) En déduire la valeur de p^*($p^* = \inf \left\{ c^T x \mid F(x) \succeq 0 \right\}$).

(3) Ecrire le dual, et expliciter sa région admissible. En déduire d^*.
($d^* = \sup \left\{ A_0 \bullet Z \mid Z = Z^T,\ Z \succeq 0,\ A_i \bullet Z = c_i,\ i = 1, ..., m \right\}$).

(4) Justifier l'existence du saut de dualité.

Soit le programme semidéfini suivant (rappel : la matrice X est symétrique) :

$$(\Pi) \begin{cases} \text{Minimiser} & x_{11} + 2x_{12} + 4x_{22} \\ \text{Sous les contraintes} & x_{33} = 0 \\ & x_{21} - x_{22} = 0 \\ & X \succeq 0 \end{cases}$$

(1) Trouver la région admissible de (Π).

(2) Ecrire le programme dual de (Π) et déterminer sa région admissible.

(3) Sans calculer d^* et p^*, déterminer s'il existe ou non un saut de dualité (si c'est possible, sinon justifier le fait que les calculs de d^* et de p^* sont nécessaires).

(4) Calculer effectivement d^* et p^*, et donner des solutions optimales de (II) et de son dual s'il en existe.

5.3. Conditions des écarts complémentaires I

Soit

$$\begin{cases} \text{Minimiser} & x_2 \\ \text{Sous contrainte} & \begin{bmatrix} x_2 & x_1 & 0 \\ x_1 & 1 & 0 \\ 0 & 0 & x_1 - 1 \end{bmatrix} \succeq 0 \end{cases}$$

(1) Ecrire le dual de ce SDP.

(2) Tracer la région admissible du primal.

(3) Résoudre le primal.

(4) Résoudre le dual en appliquant la condition des écarts complémentaire.

(5) Appliquer le théorème de dualité forte.

5.4. Conditions des écarts complémentaires II

On considère le programme semidéfini suivant, où Y est une matrice 3×3 symétrique réelle :

$$(P) \begin{cases} \text{Maximiser} & -Y_{22} \\ \text{s.c.} & Y_{12} = 1 \\ & Y_{23} = 4 \\ & Y_{13} - Y_{11} = 0 \\ & Y_{33} - Y_{22} = 0 \\ & Y \succeq 0 \end{cases}$$

(1) Montrer qu'il n'existe pas de saut de dualité entre (P) et son dual.

(2) Ecrire (DP), le dual de (P). Montrer que (P) et (DP) admettent des solutions optimales.

(3) Montrer que la valeur optimale de (P) vaut au plus -4. En supposant que cette valeur est atteinte, montrer qu'on a nécessairement $Y_{11} = 1$.

(4) En utilisant les conditions des écarts complémentaires, en déduire une solution optimale de (DP).

5.5. Programmation Quadratique I

Soit le programme semidéfini suivant :

$$(SDP) \begin{cases} \min_{z \in \Re} & z \\ \text{s.c.} & \begin{bmatrix} 3z & -z-1 \\ -z-1 & 3z \end{bmatrix} \succeq 0 \end{cases}$$

(1) Résoudre (SDP), i.e. donner sa valeur optimale ainsi que l'ensemble des solutions optimales.

(2) Ecrire $(DSDP)$, le programme dual de (SDP).

(3) Montrer qu'il n'y a pas de saut de dualité entre $(DSDP)$ et (SDP), et résoudre $(DSDP)$ en utilisant les conditions des écarts complémentaires.

(4) Soit le programme quadratique :

$$(P)\begin{cases}\max & 2x_1x_2\\ \text{s.c.} & 3x_1^2+3x_2^2-2x_1x_2=1\\ & (x_1,x_2)\in\Re^2\end{cases}$$

Etablir que la valeur optimale de $(DSDP)$ est supérieure ou égale à celle de (P), en montrant que $(DSDP)$ est une relaxation de (P).

(5) Donner le rang de la solution optimale de $(DSDP)$ trouvée à la question 3. En déduire une solution optimale de (P).

5.6. Programmation Quadratique II

Soient les deux programmes mathématiques suivants, où x est dans $\Re^2$ et X est une matrice symétrique réelle 2×2 :

$$(P)\begin{cases}\text{Maximiser} & x^TAx=2x_1x_2+x_1^2-x_2^2\\ \text{s.c.} & x_1^2+x_2^2=1\end{cases}\qquad (SDP)\begin{cases}\text{Maximiser} & 2X_{12}+X_{11}-X_{22}\\ \text{s.c.} & X_{11}+X_{22}=1\\ & X\succeq 0\end{cases}$$

(1) Ecrire $(DSDP)$, le programme dual de (SDP), et montrer qu'il n'existe pas de saut de dualité entre (SDP) et $(DSDP)$.

(2) Résoudre $(DSDP)$. Que représente la valeur optimale de $(DSDP)$ pour la matrice A ?

(3) Montrer que (P) est équivalent à $(DSDP)$.

(4) En déduire que (P) et (SDP) sont équivalents.

5.7. Résolution approchée d'un système linéaire

On souhaite résoudre de manière approchée un système linéaire $Ax=b$ où $A\in\mathbb{R}^{p\times n}$ et $b\in\mathbb{R}^p$ (on suppose que ce dernier n'admet pas de solution exacte). Pour cela, on choisit de minimiser la quantité

$$\max_{i\in\{1,\dots,p\}} \left|A_i^Tx-b_i\right|$$

où A_i représente la ième ligne ($i\in\{1,...,p\}$) de la matrice A.

(1) Modéliser ce problème comme un PL.

(2) Dans certaines applications les quantités b_i sont exprimées à l'aide d'une échelle logarithmique (car elles représentent par exemple une puissance ou une intensité). On souhaite alors minimiser la quantité

$$\max_{i\in\{1,...,p\}} \left|log\left(A_i^Tx\right)-log\left(b_i\right)\right|$$

modéliser ce nouveau problème comme un SDP.

5.8. Programmation linéaire et semidéfinie

Soit le programme linéaire :

$$(PL)\begin{cases} \text{Minimiser} & x_1 + 4x_2 \\ \text{Sous les contraintes} & x_1 - x_2 + x_3 = 2 \\ & 3x_1 - 5x_3 = 6 \\ & x_1 \geq 0\,,\ x_2 \geq 0\,,\ x_3 \geq 0 \end{cases}$$

et le programme semidéfini :

$$(SDP)\begin{cases} \text{Minimiser} & x_1 + 4x_2 \\ \text{Sous les contraintes} & \left(e_1 e_1^T + e_2 e_2^T\right) \bullet X - \left(4e_1 + 12e_2\right)^T x + 40 = 0 \\ & \begin{bmatrix} x_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & x_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_1 & x_2 & x_3 \\ 0 & 0 & 0 & x_1 & X_{11} & X_{12} & X_{13} \\ 0 & 0 & 0 & x_2 & X_{12} & X_{22} & X_{23} \\ 0 & 0 & 0 & x_3 & X_{13} & X_{23} & X_{33} \end{bmatrix} \succeq 0 \end{cases}$$

où $e_1 = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}$, $e_2 = \begin{pmatrix} 3 \\ 0 \\ -5 \end{pmatrix}$. Montrer que (SDP) et (PL) sont équivalents.

5.9. Résolution d'un problème de coupe dans un graphe

Soit $G = (V, E)$ un graphe non orienté comportant n sommets, dont chaque arête (v_i, v_j) est valuée par un réel quelconque W_{ij}, et $1 \leq k \leq n-1$ un entier. On cherche dans G la coupe (i.e. une partition de V en V_1 et V_2) de valeur maximale telle que le nombre de sommets dans V_1 soit exactement égal à k (donc le nombre de sommets dans V_2 est égal à $n-k$). Rappel : la valeur d'une coupe est la somme des valuations des arêtes ayant une extrémité dans V_1 et l'autre dans V_2.

(1) Modéliser ce problème comme un programme quadratique (Q) en variables à valeurs dans $\{0, 1\}$.

(2) Ecrire la relaxation semidéfinie basique (SDP_0) de (Q).

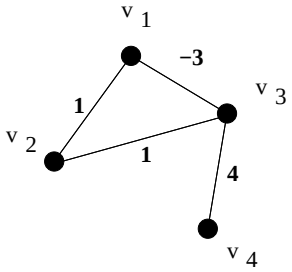
(3) Proposer une relaxation (P_L) de (Q) par programmation linéaire continue.

(4) Construire une relaxation semidéfinie (SDP) à partir de (P_L). Justifier qu'elle est meilleure que (SDP_0).

(5) Application

(a) Ecrire (SDP_0) et son dual $(DSDP_0)$ pour le graphe de la figure ci-dessus en prenant $k = 2$.

(b) Montrer qu'il n'existe pas de saut de dualité entre (SDP_0) et $(DSDP_0)$.



(c) Montrer que $x^* = (1, 0, 1, 0)$ est une solution optimale pour (Q) en prouvant que $\left(x^*, x^* x^{*T}\right)$ est optimale pour (SDP_0).

5.10. Nombre de Lovász d'un graphe

Soit un graphe $G = (V, E)$ possédant n sommets. Un stable est un sous-ensemble S de V tel qu'il n'existe aucune arête entre les éléments de S (on dit aussi que les sommets sont tous non-adjacents). La cardinalité maximale d'un ensemble stable dans G est appelée le nombre de stabilité de G, et notée $\alpha(G)$. on définit l'ensemble $P = \{A \in S_n : a_{ij} = 1 \text{ si } (i,j) \notin E \text{ ou } (i = j)\}$ (on rappelle que S_n désigne l'ensemble des matrices symétriques réelles $n \times n$). On définit également pour tout $1 \leq k \leq n$, la matrice J_k de dimension k uniquement constituée de 1.

(1) Montrer qu'il existe un stable de taille k dans G si et seulement si la matrice J_k est un mineur symétrique de toute matrice A de P (les lignes et colonnes choisies pour constituer J_k ne dépendant pas de A).

(2) Montrer que la plus grande valeur propre $\lambda_{max}(J_k)$ de J_k vaut k.

(3) Montrer que la plus grande valeur propre $\lambda_{max}(A)$ de A quelconque dans P est supérieure ou égale à $\lambda_{max}(J_k)$.

(4) Montrer que $\lambda_{max}(A) = \min\{t : tI - A \succeq 0\}$.

(5) Pour tout couple (i, j) tel que l'arête $[v_i, v_j]$ est dans E on considère E^{ij} la matrice de S_n possédant seulement deux éléments non nuls et égaux à 1 : E^{ij}_{ij} et E^{ij}_{ji}. Montrer que toute matrice A de P peut s'écrire : $A = J_n + \sum_{(i,j)\in E} x_{ij} E^{ij}$, où les x_{ij} sont des réels quelconques.

(6) Déduire des résultats précédents un programme semidéfini qui soit une relaxation du problème du stable de taille maximale dans un graphe.

(7) Ecrire le dual de ce SDP.