République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira de Béjaïa
Faculté des Sciences Exactes
Département de Recherche Opérationnelle

Mémoire de Magister

En

Mathématiques Appliquées

Option :

Modélisation Mathématique et Techniques de Décision

Thème :

Résolution d'un problème de programmation bi-niveaux linéaire par la méthode D.C.

Présenté par :

M^{elle} Aicha ANZI

Devant le jury composé de :

Président	M^{r}	Djamil	AISSANI	Professeur	Université de Béjaia
Rapporteur	M^{r}	M.ohammed Said	RADJEF	Professeur	Université de Béjaia
Examinateur	M^{r}	Mohand Ouamer	BIBI	Professeur	Université de Béjaia
Examinateur	M^{r}	Mohand	OUANES	M. C.	Université de Tizi-Ouzou

Béjaia, 2009.

À ***ma mère et mon père***

À ***mes sœurs et mon frère***

À ***Nassim et sa famille***

Remerciements

D'abord je remercie mon Créateur le tout Puissant Dieu qui m'a donné force, santé et endurance.

Je tiens à présenter mes remerciements les plus sincères à Mr. Mohammed Said RADJEF, mon directeur de thèse, pour sa confiance, ses conseils et surtout pour l'intérêt particulier qu'il a accordé à ce travail.

Je remercie Monsieur Djamil AISSANI, professeur à l'université de Béjaia et président du laboratoire LAMOS (Béjaia), qui m'a fait l'honneur de présider ce jury.

J'adresse mes vifs remerciements à Messieurs Mohand Ouamer BIBI, professeur à l'université de Béjaia, et Mohand OUANES, maître de conférence à l'université de Tizi-Ouzou, pour avoir acceptés de juger ce travail.

Je remercie plus particulièrement Nassim pour sa précieuse aide, ses conseils, ses encouragements et surtout pour le soutien qu'il m'a apporté tout au long de ce travail.

Je remercie beaucoup mes parents qui m'ont toujours aidé et encouragé à réaliser ce travail.

Enfin je remercie tous ceux qui m'ont soutenu de près ou de loin, sans oublier Mme. M. CHOUCHAA, la secrétaire du département de Recherche Opérationnelle, et Mr. H. SLIMANI, le chef de département de Recherche Opérationnelle, pour leur serviabilité.

Table des matières

Introduction générale

Souvent, les processus décisionnels possèdent une structure hiérarchique où le contrôle des variables de décision est partitionné entre les preneurs de décision. Dans ces systèmes, la décision prise par chaque acteur affecte les décisions des autres ainsi que leurs fonctions objectifs. Par exemple, dans une société décentralisée, la direction générale prend une décision concernant le budget. Et donc, chaque division détermine un plan de production avec une connaissance parfaite du budget.

Ces processus sont souvent modélisés et résolus par la programmation bi-niveaux. Cette classe de programmes constitue une branche de la programmation mathématique dans laquelle les contraintes sont déterminées, en partie, par un autre problème d'optimisation.

La programmation bi-niveaux est motivée par la théorie des jeux statiques et non coopératifs de Stackelberg appliquée aux problèmes économiques. Dans ces problèmes, le niveau supérieur est appelé Leader et le niveau inférieur Suiveur. Le contrôle des variables de décision est partitionné entre les preneurs de décision qui cherchent à optimiser leurs fonctions objectifs individuelles. Le Leader prend, le premier, sa décision dans l'objectif d'optimiser sa fonction de gains. Le Suiveur observe la décision du Leader et construit sa décision. Comme les ensembles des décisions sont interdépendants, la décision du Leader affecte l'ensemble des décisions et les gains du Suiveur et vice versa.

Les problèmes qui peuvent être modélisés sous forme d'un programme bi-niveaux sont nombreux et on rencontre des applications en économie [50], en transport urbain [34], en contrôle de pollution [5], en production [61], ...etc (voir aussi [6, 15, 43, 36]), sans oublier le domaine de la guerre qui est considéré parmi les premières applications de cette classe de problèmes. En effet, malgré que Candler et Norton (1977)(cité dans [36]) furent les premiers à utiliser la terminologie *programmation à deux niveaux ou à plusieurs niveaux* dans un rapport de la Banque Mondiale, les toutes premières formulations liées au problème de programmation bi-niveaux sont apparues dans l'œuvre des auteurs J. Bracken et J. McGill (1973) [27] consacrée à ce dernier domaine, et avec

l'appellation "*programmes mathématiques avec des problèmes d'optimisation dans les contraintes*".

La programmation bi-niveaux linéaire est l'un des modèles de base de l'optimisation bi-niveaux, où les fonctions objectifs et les contraintes du Leader et du Suiveur sont linéaires. Dans ce cas, il a été prouvé que la solution est atteinte en un certain point extrême de l'ensemble des contraintes qui est un polyèdre [15]. Malgré sa simplicité apparente, le problème de programmation bi-niveaux linéaire est un problème compliqué et difficile à résoudre. Beaucoup d'approches ont été proposées dans la littérature pour sa résolution [15, 43]. Les plus populaires sont basées sur sa reformulation en un programme mathématique qui est un programme linéaire avec une contrainte de complémentarité non linéaire. C'est cette contrainte de complémentarité qui constitue la difficulté majeure dans cette approche.

La programmation DC (Difference of Convex functions), quant à elle, traite, à l'aide de l'algorithme DCA (DC Algorithm), les problèmes d'optimisation non convexe et non différentiable où la fonction peut être représentée sous forme de différence de deux fonctions convexes.

La programmation DC et l'algorithme DCA ont été introduits par P.D.Tao en 1986 comme une extension naturelle et logique des travaux de P.D. Tao depuis 1974 concernant la programmation concave. Ils ont été intensivement développés par P.D. Tao et L.T. Hoai An depuis 1993 pour devenir maintenant classiques et de plus en plus populaires. DCA est une méthode de descente sans recherche linéaire pour la résolution d'un programme DC général, c'est-à-dire la minimisation de la différence de fonctions convexes. Plus précisément, c'est une méthode primale-duale de sous-gradient basée sur l'optimalité locale et la dualité en optimisation DC appliqué non pas à la fonction DC elle même, mais plutôt à ses composantes DC.

L'algorithme DCA a été appliqué avec succès pour un grand nombre de problèmes d'optimisation non convexe dans différents domaines de la science appliquée [79, 83]. La globalité des solutions calculées par DCA n'est pas toujours garantie. Cependant, il a été remarqué, qu'avec un bon choix du point initial, il converge souvent vers la solution globale.

Notre travail consiste en la résolution d'un problème de programmation bi-niveaux linéaire en utilisant les techniques de programmation DC et l'algorithme DCA.

Ce mémoire est organisé comme suit :

Le premier chapitre est consacré à la programmation bi-niveaux. Nous donnons la formulation mathématique d'un problème de programmation bi-niveaux, et en particulier le cas linéaire pour lequel nous présentons quelques propriétés de base. Nous abordons également la question d'existence de solutions et nous citons les principales méthodes de résolution de ce problème.

Dans le deuxième chapitre nous rappelons brièvement quelques notions d'analyse convexe. Ensuite, nous introduisons la méthode DC et exposons la théorie relative à cette méthode. Nous décrivons également l'algorithme DCA et sa construction.

Dans le troisième chapitre nous donnons un bref apercu sur les différentes méthodes de pénalité.

Le quatrième et dernier chapitre comporte l'application de la programmation DC pour la résolution du problème de programmation bi-niveaux linéaire, l'élaboration de l'algorithme DCA et les tests numériques. On y trouve également les résultats numériques d'un autre algorithme de résolution des problèmes de programmation bi-niveaux linéaire ainsi qu'une comparaison. On clôture ce mémoire par une conclusion.

1

Programmation à deux niveaux

Introduction

Les problèmes de programmation à deux niveaux sont des problèmes d'optimisation dont les contraintes sont déterminées, en partie, par un autre problème d'optimisation. En d'autres termes, ce sont des programmes mathématiques hiérarchiques avec deux niveaux de décision.

Dans cette partie, nous donnons la formulation mathématique d'un problème de programmation bi-niveaux (PB) dans sa forme générale. Ensuite nous focalisons l'attention sur le cas linéaire en donnant la définition d'un (PBL) (programme bi-niveaux linéaire). Puis, nous présentons quelques propriétés de base de la programmation bi-niveaux linéaire que nous illustrons avec des exemples. Nous abordons également la question d'existence des solutions du (PBL) ainsi que leurs caractérisations. Finalement, nous rappelons les principales approches algorithmiques existantes pour la résolution du (PBL).

1.1 Formulation d'un PB

Dans sa forme générale, un problème de programmation bi-niveaux est formulé comme suit :

$$(PB)\begin{cases}\max\limits_{x} F(x,y)\\ \quad s.c. \quad G(x,y)\leq 0\\ \qquad\quad \max\limits_{y} f(x,y)\\ \qquad\qquad s.c. \quad g(x,y)\leq 0,\end{cases} \tag{1.1}$$

où les variables sont divisées en deux classes, les variables $x \in \mathbb{R}^{n_1}$ du niveau supérieur et les variables $y \in \mathbb{R}^{n_2}$ du niveau inférieur. De même, les fonctions $F : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \longrightarrow \mathbb{R}$ et $f : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \longrightarrow \mathbb{R}$ sont respectivement les fonctions objectifs du niveau supérieur et niveau inférieur, et les fonctions vectorielles $G : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \longrightarrow \mathbb{R}^{m_1}$ et $g : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \longrightarrow \mathbb{R}^{m_2}$ sont les contraintes du niveau supérieur et niveau inférieur respectivement.

Dans le problème (1.1), en ajoutant l'hypothèse de linéarité des fonctions $F(x,y)$, $f(x,y)$, $G(x,y)$ et $g(x,y)$, on obtient un programme linéaire qu'on appelle *Programme bi-niveaux linéaire* et qu'on notera (PBL) .

1.2 Programmation Bi-niveaux Linéaire

La programmation bi-niveaux linéaire est l'un des modèles de base de l'optimisation bi-niveaux, où les fonctions objectifs et les contraintes du Leader et du Suiveur sont linéaires. Ce modèle a été intensivement étudié dans la littérature par plusieurs auteurs. Dans cette section, nous allons voir, à travers des exemples, les propriétés de base des problèmes de programmation bi-niveaux linéaire.

1.2.1 Formulation mathématique

Un problème de programmation bi-niveaux linéaire (PBL) peut être écrit sous sa forme générale :

$$(PBL)\begin{cases}\max\limits_{x}\ F(x,y)=c_1^t x+d_1^t y,\\ \quad s.c.\ \ A_1x+B_1y\leq b_1,\\ \qquad x\geq 0;\\ \qquad \max\limits_{y}\ f(x,y)=c_2^t x+d_2^t y,\\ \qquad\quad s.c.\ \ A_2x+B_2y\leq b_2,\\ \qquad\qquad y\geq 0,\end{cases} \tag{1.2}$$

où $F : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \longrightarrow \mathbb{R}$; $f : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \longrightarrow \mathbb{R}$ sont les fonctions objectifs du Leader et du Suiveur respectivement ; $c_1, c_2 \in \mathbb{R}^{n_1}$; $d_1, d_2 \in \mathbb{R}^{n_2}$; $b_1 \in \mathbb{R}^{m_1}$,

$b_2 \in \mathbb{R}^{m_2}$; A_1-$m_1 \times n_1$-matrice, B_1-$m_1 \times n_2$-matrice, A_2-$m_2 \times n_1$-matrice et B_2-$m_2 \times n_2$-matrice.

1.2.2 Exemple de problème bi-niveaux linéaire

Problème de contrôle de ressources : [26]

Le problème bi-niveaux linéaire de contrôle de ressources est formulé de la manière suivante :

$$\begin{cases} \max_x F(x,y) = c_{11}^t x + c_{12}^t y \\ \max_y f(x,y) = c_{21}^t x + d_{22}^t y \\ A_1 x + A_2 y \leq b \end{cases}$$

Ce modèle a été proposé pour analyser la répartition d'un budget fédéral entre plusieurs états (Cassidy et al. cité dans [26]). Le vecteur x représente la répartition du budget de chaque état. Au deuxième niveau, chaque état peut choisir de manière indépendante le financement de projets en respectant son budget. Le niveau de financement est représenté par le vecteur y.

1.2.3 Définition du (PBL)

Considérons le problème (1.2). Nous avons les définitions suivantes.

Définition 1.1. [15]

a) Le domaine S des contraintes du (PBL) est défini par :

$$S = \{(x,y) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} : A_1 x + B_1 y \leq b_1,\ A_2 x + B_2 y \leq b_2,\ x \geq 0,\ y \geq 0\}.$$

b) L'ensemble des solutions réalisables du Suiveur pour un x fixé est noté par $S(x)$ et défini par :

$$S(x) = \{y \in \mathbb{R}^{n_2} : B_2 y \leq b_2 - A_2 x, y \geq 0\}.$$

c) La projection de S sur l'ensemble des décisions du Leader :

$$P(X) = \{x \in \mathbb{R}^{n_1} : \exists y \in \mathbb{R}^{n_2},\ A_1 x + B_1 y \leq b_1,\ A_2 x + B_2 y \leq b_2,\ x \geq 0,\ y \geq 0\}.$$

d) L'ensemble des réactions rationnelles du Suiveur pour $x \in P(X)$, est donné par :

$$R(x) = \{y \in \mathbb{R}^{n_2} : y = \arg\max[f(x,\hat{y}) : \hat{y} \in S(x)]\},$$

avec

$$\arg\max[f(x,\hat{y}) : \hat{y} \in S(x)] = \{y \in S(x) : f(x,y) \geq f(x,\hat{y}),\ \forall \hat{y} \in S(x)\}\,.$$

e) La région induite (ou ensemble induit) :

$$RI = \{(x,y) \in S,\ y \in R(x)\}\,.$$

■

Le Leader peut obtenir différentes réactions rationnelles du Suiveur en attribuant différentes valeurs à x. L'union de toutes les valeurs possibles, x, que le Leader peut sélectionner ainsi que les réactions rationnelles correspondantes $y \in R(x)$ forment la région induite RI définie précédemment.

1.2.4 Nouvelle définition du (PBL)

La majorité des résultats obtenus dans la littérature concernent la version particulière du (PBL) sans contraintes du Leader (i.e. sans les contraintes $A_1x + B_1y \leq b_1,\ x \geq 0$). Il a été remarqué que la présence de ces contraintes, appelées parfois contraintes couplantes [47], ou contraintes de connexion [44], pourrait influencer considérablement la structure de l'ensemble réalisable du (PBL) (région induite). Elle pourrait le rendre déconnecté et parfois même vide.

Dans la pratique, la position des contraintes n'est pas arbitraire dans un (PBL) et dépend fortement de l'application considérée. Les problèmes avec des contraintes du niveau supérieur peuvent surgir par exemple si le Leader n'est pas disposé à accepter certaines décisions optimales du Suiveur. De plus, il faut noter que le déplacement de telles contraintes vers le niveau inférieur peut changer complètement le modèle.

Dans [73], les auteurs traitent le (PBL) avec contraintes du Leader, et proposent une nouvelle définition alternative à la définition 1.1. La différence entre les deux définitions est que la nouvelle définition implique que le Suiveur est tenu de respecter les contraintes du Leader, ce qui n'est pas le cas pour le (PBL) . Les auteurs motivent leur nouvelle définition en donnant une instance du (PBL) pour laquelle aucune solution ne peut être trouvée en utilisant la définition 1.1, même si $S \neq \emptyset$. Ils montrent que cette instance possède une solution et que le fait de ne pas trouver cette solution est une défaillance de la définition classique du (PBL) .

Par la suite, S. Dempe et A.G. Mersha [44] et C. Audet et al [47] ont analysé cette nouvelle définition et ont démontré qu'elle est simplement équivalente à

transférer les contraintes du niveau supérieur au niveau inférieur ce qui n'est pas permis comme nous l'avons déjà mentionné. Ils ont montré que la solution optimale du (PBL) original n'est plus optimale après déplacement de ces contraintes (voir les exemples 1.2 et 1.3).

Remarque 1. La définition considérée dans ce document est la définition classique donnée par définition 1.1. Pour plus de détails sur la nouvelle définition, le lecteur peut consulter [51, 74, 71, 72, 73].

■

Exemple 1.1. Considérons l'exemple suivant :

$$\begin{cases} \max\limits_{x} \; F(x,y) = -x + 4y \\ \quad s.c. \quad \max\limits_{y} \; f(x,y) = -y \\ \qquad\qquad -x - y \leq -3 \\ \qquad\qquad -2x + y \leq 0 \\ \qquad\qquad 2x + y \leq 12 \\ \qquad\qquad -3x + 2y \geq -4 \end{cases}$$

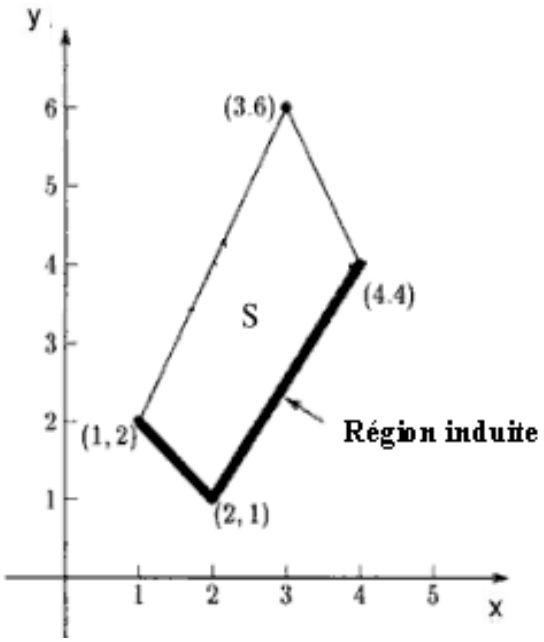

FIG. 1.1 – Géométrie du (PBL)

Suivant la définition 1.1, nous avons :
• $P(X) = \{x : 1 \leq x \leq 4\}$.

• L'ensemble des solutions réalisables $S(x)$ du Suiveur pour un $x \in P(X)$ fixé est représenté par une ligne verticale contenue dans le polyèdre S au point particulier x.
• Ensemble des réactions rationnelles du Suiveur : puisque la fonction objectif du Suiveur est de minimiser y, donc $R(x)$ est le plus petit point réalisable y sur la ligne verticale. Il est donné par

$$R(x) = \begin{cases} -x + 3, & \text{si } 1 \leq x \leq 2, \\ (3x - 4)/2, & \text{si } 2 \leq x \leq 4. \end{cases}$$

• La région induite *RI* est la portion en gras du périmètre de S.

1.2.5 Complexité du (PBL)

Le problème de programmation bi-niveaux linéaire est un problème compliqué et difficile à résoudre. Malgré que les fonctions objectifs et les contraintes du niveau inférieur et du niveau supérieur sont toutes linéaires, le (PBL) est ni continu partout, ni convexe. Bialas et Karwan (cité dans [90]) ont prouvé la non convexité du problème du niveau supérieur avec un exemple pratique. Jeroslow (cité dans [36]), Bard, Ben-Ayed et Blair [20] ont prouvé que le (PBL) est un problème NP-difficile. En outre, Hansen, Jaumard et Savard ont établi que le (PBL) est fortement NP-difficile (cité dans [36]). Plus tard, Vicente, Savard et Judice ont montré qu'établir l'optimalité stricte ou locale d'une solution est également NP-difficile (cité dans [36]).

1.2.6 Propriétés du (PBL)

Dans la définition 1.1 du (PBL), l'ensemble des contraintes représente toutes les combinaisons de choix que le Leader et le Suiveur peuvent faire. L'objet le plus important dans cette définition est la région induite. Elle représente l'ensemble réalisable du (PBL) sur lequel le Leader peut optimiser sa fonction objectif et est composée de l'union de faces de l'ensemble S [43]. Cet ensemble est polyédral, souvent non convexe et, en présence des contraintes du niveau supérieur, il peut être discontinu et même vide.
Afin d'illustrer les propriétés citées ci-dessus, nous proposons les exemples suivants :

Exemple 1.2. [44] Considérons le problème suivant :

$$(P1)\begin{cases} \max\limits_{x} x + 2y \\ \quad \max\limits_{y} y \\ \quad\quad s.c. \quad -3x + y \leq -3 \\ \quad\quad\quad\quad 3x + y \leq 30 \\ \quad\quad\quad\quad 2x - 3y \geq -12 \\ \quad\quad\quad\quad x + y \leq 14 \end{cases}$$

La région des contraintes de ce problème est représentée dans la figure 1.2. On remarque que cet ensemble est convexe. La solution de cet exemple est le point B=(6,8)

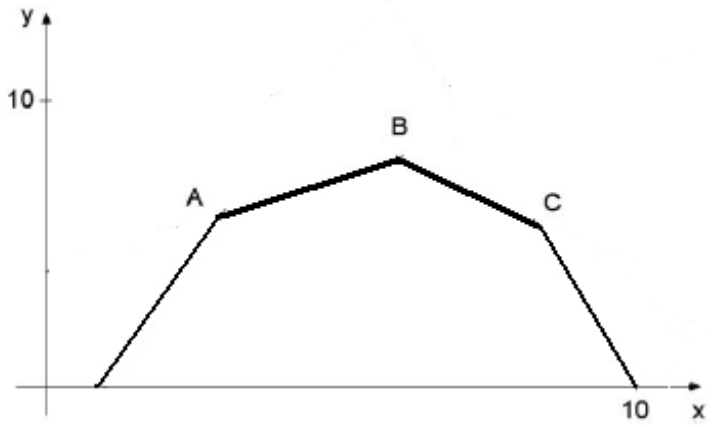

FIG. 1.2 – Représentation graphique de l'exemple 2

La région induite est représentée par des traits en gras. On voit bien qu'elle est non convexe mais continue (connectée). Dans le cas de présence des contraintes du niveau supérieur (le Leader), cette région est souvent discontinue (voir exemple 1.3).

Exemple 1.3. [44] Considérons le (PBL) de l'exemple 1.2 où les deux dernières

contraintes du Suiveur sont déplacées vers le niveau supérieur ;

$$(P2)\begin{cases} \max\limits_{x} x + 2y \\ \quad s.c. \quad 2x - 3y \geq -12 \\ \qquad\quad x + y \leq 14 \\ \qquad\quad \max\limits_{y} \; y \\ \qquad\qquad s.c. \quad -3x + y \leq -3 \\ \qquad\qquad\qquad\quad 3x + y \leq 30 \end{cases}$$

La région des contraintes ainsi que la région induite (en gras) sont représentées dans la figure 1.3

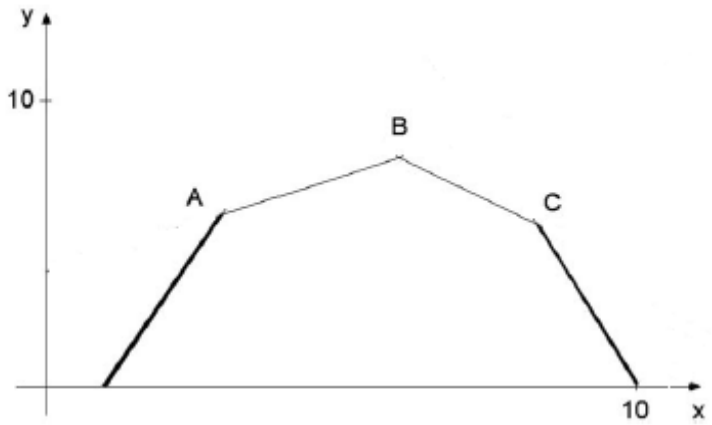

FIG. 1.3 – Représentation graphique de l'exemple 3

On voit bien d'après le graphe de l'exemple 1.3 que la région induite est discontinue. Ceci est dû à la présence des contraintes du niveau supérieur. La solution globale de cet exemple est le point C=(8,6) et A est une solution locale.

Comme nous l'avons déjà mentionné, la région induite peut être vide dans le cas de présence des contraintes du niveau supérieur. Regardons l'exemple suivant [73].

Exemple 1.4.

$$(P3)\begin{cases} \max\limits_{x} \; F(x,y) = -x + 4y \\ \quad s.c. \quad -x - y \leq -3 \\ \qquad\quad -3x + 2y \geq -4 \\ \qquad\quad \max\limits_{y} \; f(x,y) = -x - y \\ \qquad\quad s.c \quad -2x + y \leq 0 \\ \qquad\qquad\quad 2x + y \leq 12 \end{cases}$$

Le graphe de cet exemple donné dans la figure 1.4 nous montre que la région induite est vide : nous avons la région des contraintes S, et pour un x fixé l'ensemble réalisable du Suiveur est représenté par une ligne vertical ; par exemple pour $x = 2$, $S(2)$ est représenté par la ligne en gras dans la figure 1.4(a). Puisque l'objectif du Suiveur est de minimiser $f(x, y) = x + y$, l'ensemble des réactions rationnelles de ce dernier est l'ensemble $R(x) = \{0\}$ (voir figure 1.4(b)). En utilisant la définition 1.1(e), nous avons bien $RI = \emptyset$.

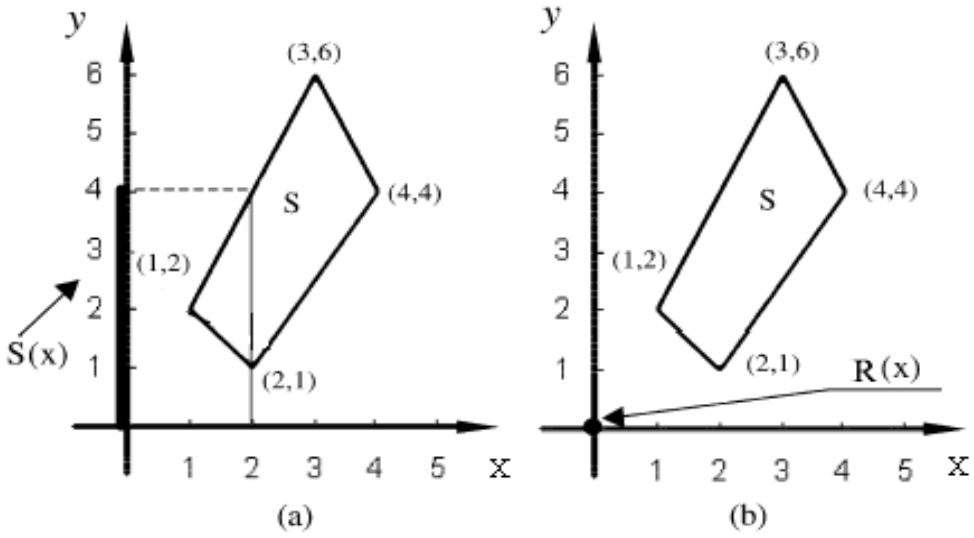

FIG. 1.4 – Représentation graphique de l'exemple 4

Solutions optimales et multiplicité des solutions

Considérons le (PBL) et supposons que l'ensemble des réactions rationnelles du Suiveur est constitué d'un seul point (au plus) pour tout $x \in P(X)$, i.e. $|R(x)| \leq 1, \ \forall\, x \in P(X)$.

Définition 1.2. [44]
Un point $(x^*, y^*) \in RI$ est optimal pour le (PBL) si :

$$c_1^t x^* + d_1^t y^* \geq c_1^t x + d_1^t y, \quad \forall (x, y) \in RI.$$

■

Cette définition d'une solution optimale du (PBL) est valable uniquement dans les conditions citées ci-dessus. Par contre, dans le cas où l'ensemble $R(x)$ est constitué de plus d'un élément (multiplicité de solutions optimales du Suiveur) pour un $x \in P(X)$ fixé, le raisonnement est différent. En effet, dans ce cas il existe dans la littérature deux approches principales pour formuler la solution du (PBL) : l'approche pessimiste (ou forte) et l'approche optimiste (ou faible).

1. **Approche pessimiste** : cette approche est appropriée pour le cas où la coopération entre le Leader et le Suiveur n'est pas autorisée et que le Leader ne peut influencer le choix du Suiveur. Dans ce cas, il est rationnel que le Leader se protège en limitant le dommage résultant d'une sélection indésirable du Suiveur tout en respectant son objectif. Donc, son problème sera formulé de la manière suivante [64] :

$$\begin{cases} \max\limits_{x} \min\limits_{y} & F(x,y) \\ s.c. & (x,y) \in S \\ & y \in R(x) \end{cases}$$

2. **Approche optimiste** : dans ce cas, le Leader peut supposer la coopération du Suiveur, dans le sens où ce dernier va choisir à chaque fois une solution qui est la meilleure du point de vue du Leader. Ce cas se formule de la même manière que le (PBL), i.e.

$$\begin{cases} \max\limits_{x,y} & F(x,y) \\ s.c. & (x,y) \in S \\ & y \in R(x) \end{cases}$$

En se mettant dans les conditions de la formulation optimiste du (PBL) , le Suiveur va choisir

$$y \in \arg\max\{c_1^t x + d_1^t y : y \in R(x)\},$$

ce qui signifie la meilleure solution dans $R(x)$ du point de vue du Leader. Le (PBL) peut donc être écrit d'une manière équivalente sous forme d'un programme mathématique standard :

$$\max\{F(x,y) : (x,y) \in RI\}, \qquad (PBL)'$$

et dans ce cas la définition 1.2 pour une solution optimale du (PBL) reste valable.

Cette définition peut être formulée autrement en utilisant la définition suivante :

Définition 1.3. [47]
Considérons un (PBL) et soit S son ensemble des contraintes. Alors

1. (x,y) est un point réalisable si $(x,y) \in S$
2. (x,y) est un point admissible si (x,y) est un point réalisable et $y \in S(x)$

■

Une solution optimale du (PBL) sera alors définie comme suit :

Définition 1.4. [47]
Un point (x^*, y^*) est une solution optimale du (PBL) si (x^*, y^*) est admissible et pour tout point admissible (x, y) on a : $F(x^*, y^*) \geq F(x, y)$.

■

Remarque 2. Il faut noter qu'en l'absence des contraintes du niveau supérieur, toute solution rationnelle est aussi solution admissible. Dans le cas de présence de ces contraintes, l'inverse n'est pas toujours vrai.

■

1.2.7 Existence et caractérisation des solutions du (PBL)

Beaucoup d'auteurs ont cherché à caractériser les solutions optimales d'un (PBL) ainsi que les conditions de leur existence. Candler et Townsley [30], Bard et Falk [17] ont établi l'existence d'une solution optimale en un point extrême du domaine réalisable S ; Bialas et Karwan [25, 26] ont démontré ce résultat sous l'hypothèse que S est borné. G. Savard (1989)(cité dans [12]) a également montré que dans le cas où la région induite est non vide, il existe au moins une solution optimale pour le (PBL) atteinte en un point extrême de l'ensemble S et que ce résultat est valable même dans le cas de présence des contraintes du niveau supérieur.

Pour chercher les conditions d'existence des solutions pour le (PBL) et pouvoir caractériser ces solutions, nous allons regarder un peu la géométrie de son ensemble réalisable (région induite RI).

Nous considérons la formulation du (PBL) sous forme d'un programme standard, $(PBL)'$, et nous posons les hypothèses suivantes.

H 1. S est un ensemble non vide et compact.

H 2. $R(x)$ est réduit à un singleton.

Dans le théorème suivant, il est montré que quand le problème est écrit sous forme d'un programme mathématique standard, la région induite résultante est composée de faces connectées de S et qu'une solution se produit en un sommet.

Théorème 1.1. *[15]*
La région induite peut être écrite d'une manière équivalente comme une contrainte d'égalité linéaire composée des hyperplans de support de S.

■

Preuve : Ecrivons d'abord la région induite sous la forme :

$$RI = \{(x, y) \in S : d_2^t y = \max[d_2^t \hat{y} : B_2\hat{y} \leq b_2 - A_2x, \hat{y} \geq 0]\}.$$

Définissons maintenant

$$Q(x) = \max\{d_2^t y : B_2^t y \leq b_2 - A_2x, y \geq 0\}. \tag{1.3}$$

Pour chaque $x \in P(X)$, l'ensemble réalisable du $(PBL)'$ est non vide et compact. Ainsi, $Q(x)$ qui est un programme linéaire paramétré en x, possède toujours une solution. En utilisant la dualité, on a :

$$\min\{u^t(b_2 - A_2x) : B_2^t u \geq d_2, u \geq 0\}, \tag{1.4}$$

qui a la même valeur optimale que (1.3) au point u^*.
Soit $u^1, ..., u^s$ les points extrêmes de l'ensemble des contraintes de (1.4) donné par $U = \{u : B_2u \geq d_2, u \geq 0\}$. Sachant qu'une solution de (1.4) se produit en un point extrême de U, on a le problème équivalent :

$$\min\{u^j(b_2 - A_2x) : u^j \in \{u^1, ..., u^s\}\},$$

ce qui montre que $Q(x)$ est une fonction linéaire. Si on écrit RI sous la forme

$$RI = \{(x, y) \in S : d_2^t y - Q(x) = 0\}, \tag{1.5}$$

on aura le résultat du théorème.

■

La fonction $Q(x)$, définie par (1.3), est convexe et continue. De plus, on veut maximiser, sous RI, une fonction linéaire $F = c_1^t x + d_1^t y$, qui est bornée supérieurement sur S par la valeur $\max\{c_1^t x + d_1^t y : (x, y) \in S\}$. Nous avons donc le résultat suivant :

Corollaire 1.1. *[15]*
Une solution du (PBL) est atteinte en un point extrême de RI.

■

Nous avons constaté dans la section 1.2.6 que malgré la linéarité des contraintes et des fonctions objectifs des deux niveaux dans les problèmes de programmation bi-niveaux linéaires, l'ensemble réalisable RI peut ne pas être convexe. Néanmoins, cet ensemble possède quelques propriétés des ensembles convexes [25]. Nous avons le théorème suivant.

Théorème 1.2. *[25]*
Supposons que S est borné. Soient $(x_1, y_1), \ldots, (x_r, y_r)$, r points de S et $\lambda_1, \ldots, \lambda_r$ des scalaires non négatifs, avec $\sum_{i=1}^{r} \lambda_i = 1$ tels que $\sum_{i=1}^{r} \lambda_i (x_i, y_i) \in RI$. Alors, $(x_i, y_i) \in RI$, pour tout $i \in \{1, 2, \ldots, r\}$ tel que $\lambda_i > 0$.

■

Dans ce théorème, il est établi que tout point dans S qui contribue strictement dans une combinaison convexe de points de S pour former un point de RI doit être dans RI.

Corollaire 1.2. *[25]*
Si (x, y) est un point extrême de RI, alors (x, y) est un point extrême de S.

■

Preuve (par contradiction)
Soit (x, y) un point extrême de RI et supposons que (x, y) n'est pas un point extrême de S. Alors, il existe des sommets $(x_1, y_1), \ldots, (x_r, y_r) \in S$ et $\lambda_1, \ldots, \lambda_r > 0$ avec $\sum_{i=1}^{r} \lambda_i = 1$ tels que $(x, y) = \sum_{i=1}^{r} \lambda_i (x_i, y_i)$.
D'après le théorème 1.2, $(x_1, y_1), \ldots, (x_r, y_r) \in RI$, donc (x, y) ne peut pas être un sommet de RI, ce qui contredit notre hypothèse et démontre le corollaire.

■

Corollaire 1.3. *[25]*
Une solution optimale (x^*, y^*) du (PBL) se produit en un sommet de S.

■

Preuve D'après le corollaire 1.1, une solution du (PBL), si elle existe, se produit en un point extrême de RI. En utilisant le corollaire 1.2, ce point est un sommet de S.

■

1.3 Résolution de (PBL)

Pour la résolution du (PBL), il serait nécessaire de transformer ce dernier en un programme mathématique standard pour avoir une représentation explicite de son ensemble réalisable, RI. L'approche la plus populaire pour cette transformation est le remplacement du problème du niveau inférieur par ses conditions d'optimalité de Karush-Khun-Tucker et l'insertion du système résultant dans le problème du niveau supérieur.

1.3.1 Reformulation du (PBL)

Soit le problème de programmation bi-niveaux linéaire (PBL) et soit $u \in \mathbb{R}^{m_2}$, $v \in \mathbb{R}^{n_2}$ les variables duales associées aux contraintes $A_2x + B_2y \leq b_2$ et à $y \geq 0$ respectivement. Nous avons la proposition suivante.

Proposition 1.1. *[15]*
Une condition nécessaire pour que (x^*, y^*) soit solution du (PBL) est qu'il existe u^* et v^* tels que (x^*, y^*, u^*, v^*) résout le système :

$$(PBL)_{KKT} \begin{cases} \max\limits_{x,y} \; F(x,y) = c_1^t x + d_1^t y \\ \qquad A_1x + B_1y \leq b_1 \\ \qquad A_2x + B_2y + w = b_2 \\ \qquad B_2^t u - v = d_2 \\ \qquad v^t y + u^t w = 0 \\ \qquad x \geq 0,\ y \geq 0,\ u \geq 0,\ v \geq 0,\ w \geq 0 \end{cases}$$

■

Cette formulation du (PBL) va jouer par la suite un rôle important dans le développement d'algorithmes de résolution du (PBL) . Le problème d'optimisation $(PBL)_{KKT}$ est linéaire à l'exception des contraintes de complémentarité $(v^t y + u^t w = 0)$.

1.3.2 Méthodes de résolution du (PBL)

Pour la résolution du (PBL) , de nombreux algorithmes ont été mis au point et particulièrement pour le cas linéaire (voir [36]). Ces algorithmes peuvent être classés en trois principales catégories.

a) Méthodes basées sur la reformulation KKT : les méthodes développées dans cette catégorie sont nombreuses et requièrent la transformation du problème en un problème d'optimisation à un seul niveau en utilisant les conditions KKT. Donc au lieu de travailler avec le problème (PBL) on travaille avec le problème $(PBL)_{KKT}$. A cause de la présence des contraintes de complémentarité dans le système KKT, le problème résultant est non linéaire. Les approches développées dans cette catégorie peuvent être résumées en :

- *Approche basée sur la programmation entière mixte* : J. Fortuny-Amat et B. McCarl [50] ont proposé, pour la résolution du problème $(PBL)_{KKT}$ de convertir les contraintes de complémentarité en contraintes linéaires en ajoutant des variables binaires.

– *Approche du pivot de complémentarité* : le premier qui a développé un algorithme de pivot de complémentarité était Bialas et al.(voir [22]). A chaque itération, l'algorithme du pivot de complémentarité (PCP) calcule un point réalisable (x, y) pour le problème original de telle façon que la fonction objectif du Leader prenne une valeur au plus égale à un nombre α. Ce paramètre est mis à jour à chaque itération et le processus s'arrête lorsque aucune solution réalisable ne peut être trouvée. Júdice et Faustino ont introduit un problème appelé *problème de complémentarité linéaire séquentielle* pour la résolution des problèmes de programmation bi-niveaux linéaires-quadratiques. Leur approche peut être vue comme une combinaison entre la technique de Branch and Bound et la méthode d'énumération de points extrêmes (voir [56, 57, 22]).

– *Approche basée sur les techniques Branch and Bound* : l'idée de cette approche est de supprimer le terme de complémentarité et résoudre le programme linéaire résultant. A un noeud donné de l'arbre Branch and Bound qui ne vérifie pas les contraintes de complémentarité, la séparation se fait de la manière suivante : deux noeuds seront créés, l'un avec $(v,\ u) = 0_{\mathbb{R}^{n_2+m_2}}$ comme une contrainte additionnelle et l'autre avec $(y,\ w) = 0_{\mathbb{R}^{n_2+m_2}}$. Bard et Falk [17] et Fortuny-Amat et McCarl [50] ont développé des algorithme basés sur la technique Branch and Bound pour le cas linéaire. Cette approche a été adaptée par Bard et Moore [18] pour le cas linéaire-quadratique et par Bard [14] et Edmunds et Bard pour le cas quadratique [16].

– *Approche basée sur les fonctions de pénalité* : après transformation du problème original en un problème à un seul niveau, une pénalité exacte est utilisée. Dans Anandalingam et White [7][8], la fonction pénalité utilisée est la différence entre les valeurs des fonctions objectifs primale et duale du Suiveur. Campêlo et Scheimberg [29] ont utilisé également une pénalisation, mais cette fois-ci des contraintes de complémentarité.

b) Méthodes basées sur l'énumération de points extrêmes : ces méthodes sont développées pour le cas linéaire. Une propriété importante des problèmes de programmation bi-niveaux linéaires est que les points extrêmes de la région réalisable du (PBL) sont les points extrêmes de la région des contraintes S et que la solution optimale du problème est l'un de ces points.

En se basant sur cette idée, beaucoup d'auteurs ont développé des algorithmes pour la résolution du (PBL) . Bialas et Karwan [25] ont proposé le $\mathrm{K}^{ième}$ meilleur algorithme. Sous la condition que la région

induite RI est bornée et que l'ensemble des réactions rationnelles $R(x)$ est réduit à un singleton, cet algorithme trouve la solution globale du problème. Les mêmes auteurs [26] ont proposé une autre méthode mais qui aboutit à une solution locale.

Bard [13] a développé un algorithme de recherche grille. Cet algorithme calcule une borne inférieure et une borne supérieure de la valeur optimale cherchée. Hansen et al.[12] ont présenté un algorithme de type Branch and Bound qui n'est pas basé sur les conditions KKT du niveau inférieur, mais qui cherche à déterminer les contraintes satisfaites ainsi que les variables du suiveur égales à zero au point optimal.

c) Méthodes basées sur les heuristiques : dans cette catégorie on peut trouver des algorithmes basés sur les algorithmes génétiques [90, 53], des algorithmes basés sur les réseaux de neurones [58], des algorithmes utilisant le recuit simulé.

d) Autres méthodes : d'autres méthodes ont été proposées par plusieurs auteurs dans le but de résoudre le cas non linéaire sous différentes hypothèses. On peut citer :

- *Méthodes de descente* : proposées par Vicente et al.[87] pour le cas quadratique convexe, i.e., les problèmes où les deux fonctions objectifs sont quadratiques et où les contraintes sont linéaires. Falk and Liu [48] ont également proposé un algorithme de descente appelé *algorithme du Leader prédominant* vu le rôle joué par le Leader dans le processus de prise de décision.

- *Méthodes de fonctions de pénalité* : ces méthodes sont généralement limitées au calcul des points stationnaires et optimums locaux. Les premiers auteurs ayant développé ce genre de méthodes sont Aiyoshi et Shimizu [1, 2]. Leur approche consistait en le remplacement du problème du niveau inférieur par un problème pénalisé en introduisant un paramètre de pénalité. Ishizuka et Aiyoshi [55] ont proposé une méthode à double pénalité où cette fois-ci les deux fonctions objectifs (les fonctions du Leader et du Suiveur) sont pénalisées.

- *Méthodes de région de confiance* : dans ces méthodes itératives, l'idée est l'approximation, à chaque itération, du problème original par un modèle de même type [35](approximations linéaires de la fonction objectif du Leader ainsi que ses contraintes, et des approximations quadratiques de

la fonction objectif du Suiveur). Un algorithme de région de confiance a été développé pour les problèmes sans contraintes du Leader et où le programme du Suiveur est convexe avec des contraintes linéaires (voir Liu et al.[59]).

2
Programmation DC

Introduction

Ce chapitre est divisé en deux parties. Dans la première partie, nous allons donner brièvement quelques notions d'analyse convexe dont nous aurons besoin dans ce document. La deuxième partie est consacrée à la méthode DC et la présentation de l'algorithme DCA.

2.1 Eléments d'analyse convexe

Dans la suite, nous considérons l'ensemble X comme étant l'espace euclidien $\mathbb{R}^n$ muni du produit scalaire usuel $\langle ., . \rangle$ et de la norme euclidienne $\|x\| = \sqrt{\langle ., . \rangle}$, et soit C un sous ensemble de X. Y désignera l'espace dual de X que l'on pourra identifier à X.
On notera $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ muni de la structure algébrique déduite de celle de $\mathbb{R}$, avec la convention $+\infty - (+\infty) = +\infty$.

2.1.1 Ensembles convexes

Définition 2.1. (*segment de ligne*)
Un *segment de ligne* entre deux points $x, y \in \mathbb{R}^n$ est un ensemble défini par :

$$[x, y] = \{\lambda x + (1 - \lambda)y; \lambda \in [0, 1]\}.$$

Définition 2.2. (*ensemble affine*)
Un ensemble C est dit affine si toute ligne entre deux points $x, y \in C$ est incluse dans C, i.e. si

$$\forall\ x,\ y \in C, \forall\ \lambda \in \mathbb{R},\ \lambda x + (1-\lambda)y \in C.$$

Un ensemble affine est appelé également *variété affine* ou *variété linéaire*.

Les ensembles $\emptyset$ et $\mathbb{R}^n$ sont des ensembles affines.

L'ensemble de toutes les combinaisons affines des points d'un ensemble affine C est appelé *enveloppe affine* de C et est donné par :

$$aff(C) = \Big\{ \sum_{i=1}^{k} \lambda_i x_i, \lambda_i \in \mathbb{R},\ x_i \in C\ \forall i = \overline{1,k} \text{ et } \sum_{i=1}^{k} \lambda_i = 1 \Big\};$$

$aff(C)$ est le plus petit ensemble affine qui contient C.

Définition 2.3. (*ensemble convexe*)
Un ensemble C est dit convexe s'il contient le segment $[x,\ y]$ liant chaque paire $x,\ y$ de ses points (voir figure 2.1), i.e.

$$\forall\ x,\ y \in C, \forall\ \lambda \in [0,1],\ \lambda x + (1-\lambda)y \in C.$$

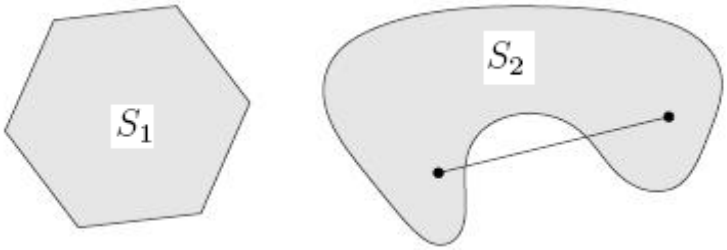

FIG. 2.1 – S_1 : ensemble convexe, S_2 : ensemble non convexe

Tous les ensembles affines (y compris $\emptyset$ et $\mathbb{R}^n$) sont convexes.

L'*enveloppe convexe* de C est l'ensemble de toutes les combinaisons convexes des points dans C. Elle est donnée par :

$$ConvC = \Big\{ \sum_{i=1}^{k} \lambda_i x_i, \lambda_i \geq 0,\ x_i \in C,\ \forall i = \overline{1,k} \text{ et } \sum_{i=1}^{k} \lambda_i = 1 \Big\}.$$

L'*intérieur relatif* d'un ensemble C est défini comme son intérieur relatif à $aff(C)$. Il est donné par :

$$ir(C) = \{x \in C : \exists r > 0 \mid B(x,r) \cap aff(C) \subset C\},$$

où $B(x,r)$ est la boule de centre x et de rayon r.

Définition 2.4. (*Ensemble polyédral*)
Un ensemble C est dit *polyédral*, s'il peut être exprimé comme une intersection d'une famille finie de demi-espaces, i.e. comme un ensemble de solutions d'un système fini d'inégalités :

$$\langle a_i, x\rangle \leq b_i, \quad i = \overline{1,m}.$$

2.1.2 Fonctions convexes

Définition 2.5. (*fonction affine*)
Une fonction $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ est dite *affine* si elle est la somme d'une fonction linéaire et une constante, i.e. elle possède la forme :

$$f(x) = ax + b, \text{ avec } a \in \mathbb{R}^{n\times m} \text{ et } b \in \mathbb{R}^m.$$

Définition 2.6. (*fonction convexe*)
Une fonction $f : C \longrightarrow \mathbb{R} \cup \{+\infty\}$ est dite convexe, si :
- C est un ensemble convexe.
- Pour tout $x,\ y \in C$ et $\lambda \in [0,1]$, on a :

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y). \tag{2.1}$$

Géométriquement, la fonction $f : C \longrightarrow \mathbb{R} \cup \{+\infty\}$ est convexe si le segment entre les points $(x, f(x))$ et $(y, f(y))$ se trouve au dessus du graphe de f (voir figure 2.2).
Une fonction f, définie sur un domaine convexe, est dite concave, si $(-f)$ est convexe.

Remarque 3. Soit f et g deux fonctions convexes, alors les fonctions

- $f + g$;
- $\max\{f, g\}$;
- λf avec $\lambda > 0$;

sont des fonctions convexes, mais la fonction $(f - g)$ n'est pas nécessairement convexe.

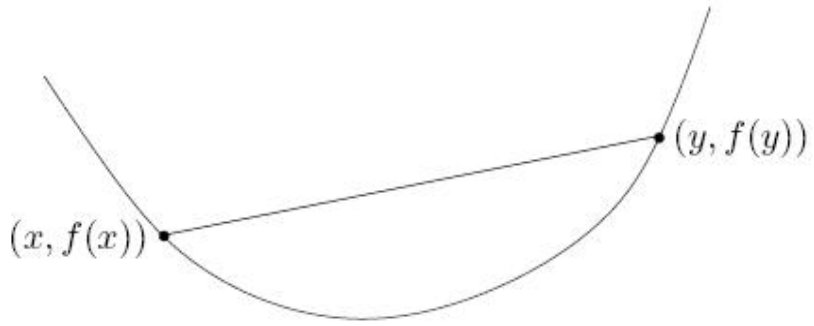

FIG. 2.2 – Fonction convexe

Le *domaine effectif* d'une fonction convexe $f : C \longrightarrow \mathbb{R} \cup \{+\infty\}$ définie sur un ensemble convexe C est l'ensemble des $x \in C$ où $f(x)$ est finie. Il est donné par :

$$domf = \big\{x \in C : f(x) < +\infty\big\}.$$

L'*épigraphe* d'une fonction convexe est un sous ensemble de $\mathbb{R}^{n+1}$ défini comme suit :

$$epif = \big\{(x,t) \in C \times \mathbb{R} \mid x \in domf,\ f(x) \leq t\big\};$$

l'épigraphe de f est l'ensemble de points qui se situent au dessus de son graphe. Une fonction est dite convexe, si son épigraphe est convexe.

Définition 2.7. (*fonction convexe propre*)
Une fonction convexe est dite propre, si elle ne prend nulle part la valeur $-\infty$ et n'est pas identiquement égale à $+\infty$.
En utilisant l'épigraphe, on dit que f est propre, si son épigraphe est non vide.

Définition 2.8. (*fonction indicatrice*)
La fonction indicatrice d'un ensemble C est définie par :

$$\chi_C(x) = \begin{cases} 0, & si \ \ x \in \mathcal{C}, \\ +\infty, & sinon. \end{cases}$$

Elle est convexe, si l'ensemble C est convexe.

Définition 2.9. (*fonction semi-continue inférieurement*)
Une fonction $f : C \longrightarrow \mathbb{R} \cup \{+\infty\}$ est dite semi-continue inférieurement (s.c.i) en $x \in C$, si pour toute suite de points $\{x_k\}$ convergeant vers x, on a :

$$f(x) \leq \lim_{k \to \infty} \inf f(x_k).$$

Nous avons le résultat suivant.

Proposition 2.1. *[46]*
Une fonction $f : C \longrightarrow \mathbb{R} \cup \{+\infty\}$ est semi-continue inférieurement si et seulement si son épigraphe est fermé.

■

Définition 2.10. (*conjuguée d'une fonction*)
On appelle fonction *conjuguée* définie sur Y d'une fonction $f : X \longrightarrow \mathbb{R} \cup \{+\infty\}$ définie sur X, la fonction $f^* : Y \longrightarrow \mathbb{R} \cup \{+\infty\}$ $\forall y \in Y$. définie par

$$f^*(y) = \sup\{\langle x, y\rangle - f(x), x \in X\}, \ \forall y \in Y.$$

La fonction $f^{**} = (f^*)^* : X \longrightarrow \mathbb{R} \cup \{+\infty\}$ est appelée la *biconjuguée* de f, elle est donnée par

$$f^{**}(x) = \sup\{\langle x, y\rangle - f^*(y), y \in Y\}.$$

Soit f une fonction convexe et f^* sa fonction conjuguée. La relation

$$\langle x, y\rangle \leq f(x) + f^*(y), \quad \forall\ (x, y) \in X \times Y$$

est appelée *inégalité de Fenchel*.

Propriété 2.1. [3] On a les propriétés suivantes :

1. f^* est toujours convexe.
2. S'il existe $x \in X$ tel que $f(x) = -\infty$, alors $f^*(y) = +\infty, \quad \forall y \in Y$.
3. On a $f^{**} \leq f$.

Pour la propriété 3, si de plus $f \in \Gamma_0(X)$ (ensemble des fonctions convexes, propres et s.c.i), alors $f^{**} = f$.
Soit χ_C la fonction indicatrice de l'ensemble C, sa conjuguée est donnée par :

$$\chi_C^*(y) = \sup_{x \in C} \langle y, x\rangle$$

qui est elle même la fonction d'*appui* de l'ensemble C.

Définition 2.11. (*fonction convexe polyédrale*)
Une fonction f est dite convexe polyédrale, si son épigraphe est un ensemble polyédral. En d'autres termes, f est convexe polyédrale si et seulement si elle peut être exprimée sous la forme :

$$f(x) = \max\left\{\langle a_i, x\rangle - b_i, \ i = \overline{1, m}\right\} + \chi_C(x) \tag{2.2}$$

avec $a_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$ pour $i = \overline{1, m}$ et C est un sous ensemble non vide et convexe de $\mathbb{R}^n$ et χ_C est la fonction indicatrice de l'ensemble C.

Remarque 4. Comme exemples de fonctions convexes polyédrales, on a les *fonctions affines* et la *fonction indicatrice*d'un ensemble convexe.

Théorème 2.1. *[68]*
La conjuguée d'une fonction convexe polyédrale est polyédrale.

■

2.1.3 Sous-différentiabilité

Définition 2.12. (*sous-gradient*)
Un vecteur $y_0 \in Y$ est dit sous-gradient de la fonction convexe f au point $x_0 \in X$ si :

$$f(x) \geq f(x_0) + \langle y_0, x - x_0 \rangle, \quad \forall x \in X. \tag{2.3}$$

La relation (2.3) possède une interprétation géométrique simple : elle signifie que le graphe de la fonction (affine) $\varphi(x) = f(x_0) + \langle y_0, x - x_0 \rangle$ est un hyperplan de support *non vertical* à l'ensemble convexe $epi f$ au point $(x_0, f(x_0))$. Pour une fonction différentiable, cet hyperplan est vertical au point x_0 (voir figure 2.3 et figure 2.4).

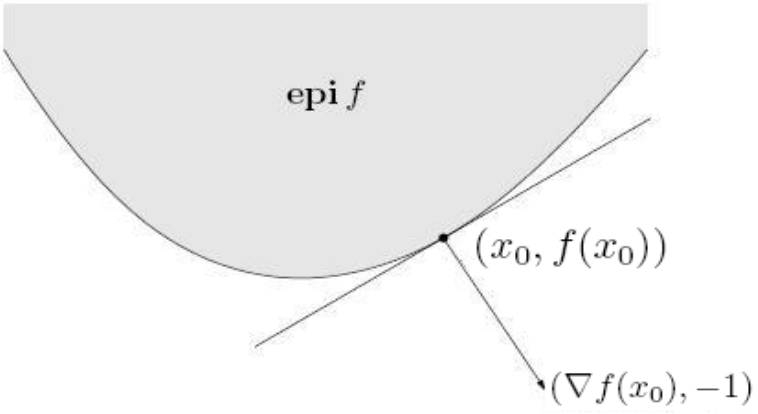

FIG. 2.3 – Fonction différentiable

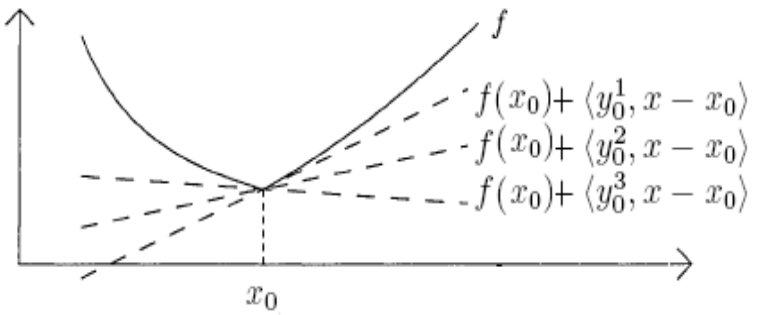

FIG. 2.4 – Fonction non différentiable : y_0^1, y_0^1 et y_0^1 sous-gradients de f au point x_0

Définition 2.13. (*sous-différentiel*)
Le sous-différentiel de f au point x_0, noté $\partial f(x_0)$, est l'ensemble de tous les sous-gradients de f en ce point, et il est donné par :

$$\partial f(x_0) = \big\{y_0 \in Y, f(x) \geq f(x_0) + \langle y_0, x - x_0\rangle,\ \forall x \in X\big\}.$$

L'ensemble $\partial f(x_0)$ est un ensemble fermé et convexe et si $\partial f(x_0) \neq \emptyset$, alors la fonction f sera dite *sous-différentiable* au point x_0.

Si $\partial f(x_0)$ est réduit à un singleton, i.e. $\partial f(x_0) = \{y_0\}$, alors f est *différentiable* en x_0 et $y_0 = \nabla f(x_0)$.

Soit $\epsilon > 0$, alors l'ensemble

$$\partial_\epsilon f(x_0) = \big\{y_0 \in Y | f(x) \geq f(x_0) + \langle y_0, x - x_0\rangle - \epsilon,\ \forall x \in X\big\}.$$

est appelé ϵ-*sous-différentiel* de f au point x_0.
Nous avons le lemme suivant.

Lemme 2.1. [82]
Soit f une fonction convexe et propre et $x_0 \in int\ domf$. Nous avons :

1. $y_0 \in \partial f(x_0) \Longleftrightarrow f^*(y_0) = \langle x_0, y_0\rangle - f(x_0)$.
2. $y_0 \in \partial_\epsilon f(x_0) \Longleftrightarrow f(x_0) + f^*(y_0) - \langle x_0, y_0\rangle \leq \epsilon$.
3. Si f est s.c.i, alors $\partial f(x_0) \neq \emptyset$.
4. Si f est convexe et s.c.i, alors $f^{**}(x) = f(x),\ \ \forall x \in int\ domf$.

■

Calcul sous-différentiel

Soient f, $g : C \longrightarrow \mathbb{R} \cup \{+\infty\}$ deux fonctions et $\lambda > 0$.

De la définition du sous-différentiel, nous avons les règles suivantes :

$$\begin{aligned} \partial(\lambda f(x)) &= \lambda \partial f(x); \\ \partial f(x) + \partial g(x) &\subset \partial(f+g)(x). \end{aligned}$$

En général $\partial(f+g)(x) \neq \partial f(x) + \partial g(x)$. La proposition suivante nous donne une condition suffisante pour l'égalité.

Proposition 2.2. *[92]*
Soient f, $g : C \longrightarrow \mathbb{R} \cup \{+\infty\}$ deux fonctions convexes. S'il existe $x_0 \in \partial f(x) \cap \partial g(x)$ où f est continue, alors pour tout $x \in C$:

$$\partial(f+g)(x) = \partial f(x) + \partial g(x). \tag{2.4}$$

■

Remarque 5.

1. Dans le cas où les fonctions f et g sont convexes et continues, l'égalité dans (2.4) reste toujours vraie [32].
2. Si f et g sont deux fonctions différentiables, alors le sous-différentiel de la fonction $\min\{f(x), g(x)\}$ est donnée par [43] :

$$\partial \min\{f(x), g(x)\} \begin{cases} = \{f'(x)\} & \text{si } f(x) < g(x) \\ \subseteq Conv\{f'(x), g'(x)\} & \text{si } f(x) = g(x) \\ = \{g'(x)\} & \text{si } f(x) > g(x) \end{cases}$$

2.2 Méthode DC

Dans cette partie, on considère $\Gamma_0(X)$ l'ensemble de toutes les fonctions convexes, propres et s.c.i sur X.

2.2.1 Fonctions DC

Définition 2.14. (*fonction DC*)
Une fonction $f : X \longrightarrow \mathbb{R} \cup \{+\infty\}$ est dite DC si elle peut être représentée sous forme de différence de deux fonctions convexes g, $h : X \longrightarrow \mathbb{R} \cup \{+\infty\}$:

$$f(x) = g(x) - h(x), \tag{2.5}$$

où g et h sont deux fonctions de $\Gamma_0(X)$. Elles sont appelées les composantes DC de f.

La représentation (2.5) est appelée *décomposition DC* de f. L'ensemble des fonctions DC sur X est noté $DC(X)$.

Proposition 2.3. *[54]*
Soient f et f_i, $i = 1, \ldots, m$, des fonctions DC. Les fonctions suivantes sont aussi DC :

- $\sum_{i=1}^{m} \lambda_i f_i(x), \quad \lambda_i \in \mathbb{R}, \quad i = 1, \ ..., \ m;$
- $\max_{i=1, \ ..., \ m} f_i(x), \min_{i=1, \ ..., \ m} f_i(x);$
- $|f(x)|, \quad f^+(x) = \max\{0, \ f(x)\}, \quad f^-(x) = \min\{0, \ f(x)\};$
- $\prod_{i=1}^{m} f_i(x).$

■

2.2.2 Programmation DC

Définition 2.15. (*programmation DC*)
Les problèmes de programmation DC sont des programmes traitant des fonctions DC. En général, un programme DC a la forme :

$$\alpha = \inf\{f(x) = g(x) - h(x) \ : \ x \in X\}, \tag{2.6}$$

où g et h sont des fonctions de $\Gamma_0(X)$.

En plus du problème (2.6), considérons les deux programmes suivants :

$$\sup\{f(x) \ : \ x \in X\}, \tag{2.7}$$

où f est une fonction convexe sur l'ensemble convexe X et

$$\inf\{g(x) - h(x) \ : \ x \in X, \ f_1(x) - f_2(x) \leq 0\}, \tag{2.8}$$

où g, h, f_1, f_2 sont des fonctions convexes sur l'ensemble convexe X. Les programmes (2.6), (2.7) et (2.8) sont équivalents. En effet, (2.7) est un cas particulier de (2.6) où $g = \chi_X$ la fonction indicatrice de X et $h = -f$. Le programme (2.7) peut être écrit d'une manière équivalente à (2.6) en ajoutant une variable scalaire et le programme (2.8) peut être transformé en (2.7) par le biais d'une pénalisation exacte de la contrainte DC : $f_1(x) - f_2(x) \leq 0$.

Propriétés de la programmation DC

- Une fonction DC possède une infinité de décompositions DC. Par exemple, si $f = g - h$, alors $f = (g + \xi) - (h + \xi)$ pour tout $\xi \in \Gamma_0(X)$ finie sur tout X.
- Tous les résultats obtenus concernant la théorie de la programmation DC (dualité, conditions d'optimalité, ...) sont basés uniquement sur les composantes DC de f ainsi que leurs conjuguées et non pas sur f elle-même.
- L'algorithme DCA qui sera vu dans la section 2.2.5 est également construit à partir des composantes DC et leurs conjuguées. De plus, il y a autant de DCA que de décompositions DC pour la fonction f. Il est donc nécessaire de trouver la bonne décomposition DC pour f puisqu'elle influence considérablement l'efficacité de DCA ainsi que sa solution.

Remarque 6. La non convexité dans (2.6) vient de la concavité de $(-h)$, à moins que h soit affine ; dans ce cas, le problème (2.6) est convexe.

Définition 2.16.

1. x^* est un minimum local de $(g-h)$ si $g(x^*)-h(x^*)$ est finie (i.e., $x^* \in domg \cap domh$) et il existe un voisinage $\mathcal{U}$ de x^* tel que

$$g(x^*)-h(x^*) \leq g(x)-h(x), \quad \forall x \in \mathcal{U}. \tag{2.9}$$

2. x^* est un point critique de $(g-h)$, si $\partial h(x^*) \cap \partial g(x^*) \neq \emptyset$.

Théorème 2.2. *[3]*
Soit $f = g-h$, où g et $h \in \Gamma_0(X)$. Alors, x^* est minimum global du problème (2.6), si et seulement si,

$$\partial_\epsilon h(x^*) \subset \partial_\epsilon g(x^*), \quad \forall \epsilon \geq 0. \tag{2.10}$$

■

2.2.3 Dualité en programmation DC

Soit $g,\ h \in \Gamma_0(X)$, donc nous avons $h^{**} = h$. En utilisant la définition de la conjuguée, nous aurons :

$$h(x) = \sup\{\langle x, y\rangle - h^*(y) : y \in Y\}. \tag{2.11}$$

En remplaçant h par son expression (2.11) dans (2.6), on obtient

$$\begin{aligned}
\inf\{g(x) - \sup\{\langle x, y\rangle - h^*(y) : y \in Y\} : x \in X\} &= \inf_{x\in X} \inf_{y\in Y}\{g(x) + \{h^*(y) - \langle x, y\rangle\} \\
&= \inf_{y\in Y} \inf_{x\in X}\{g(x) + \{h^*(y) - \langle x, y\rangle\} \\
&= \inf_{y\in Y} \inf_{x\in X}\{h^*(y) - \langle x, y\rangle + g(x)\} \\
&= \inf_{y\in Y}\{h^*(y) + \inf_{x\in X}\{-\langle x, y\rangle + g(x)\}\} \\
&= \inf_{y\in Y}\{h^*(y) - \sup_{x\in X}\{\langle x, y\rangle - g(x)\}\} \\
&= \inf_{y\in Y}\{h^*(y) - g^*(y)\}.
\end{aligned}$$

Le problème

$$\alpha = \inf\{h^*(y) - g^*(y) : y \in Y\} \tag{2.12}$$

est le problème dual du problème DC (2.6).
On remarque que le problème (2.6) et son dual (2.12) sont symétriques. Par ce fait, la résolution de l'un implique la résolution de l'autre. En pratique, ceci est très utile quand l'un des deux problèmes est plus facile à résoudre que l'autre. Nous avons les résultats suivants.

Théorème 2.3. *[54]*

1. *Si le problème (2.6) possède une solution optimale, alors nous avons*

$$\inf\{g(x) - h(x) : x \in X\} = \inf\{h^*(y) - g^*(y) : y \in Y\}. \tag{2.13}$$

2. *Le problème dual de (2.12) est le problème (2.6).*

■

Pour la preuve de ce théorème, nous avons besoin du lemme suivant.

Lemme 2.2. [54]

1. Soit $x_0 \in \mathbb{R}^n$ et $y_0 \in \partial h(x_0)$. Alors,

$$h^*(y_0) = \langle x_0, y_0\rangle - h(x_0). \tag{2.14}$$

2. Soient $g : X \longrightarrow \mathbb{R} \cup \{+\infty\}$ une fonction quelconque et $h : X \longrightarrow \mathbb{R}$ est une fonction convexe. Soit $f = g - h$. Alors, la fonction conjuguée f^* de f donnée par :

$$f^*(z) = \sup\{g^*(y + z) - h^*(y) : y \in Y\}. \tag{2.15}$$

■

Preuve du théorème 2.3 :

1. Par définition de f^*, nous avons

$$f^*(0) = \sup\{-f(x) : x \in X\}, \tag{2.16}$$

i.e.,

$$\inf\{f(x) : x \in X\} = -f^*(0). \tag{2.17}$$

De (2.15), il résulte

$$\begin{aligned} \inf\{f(x) : x \in X\} &= -f^*(0) \\ &= -\sup\{g^*(y) - h^*(y) : y \in Y\} \\ &= \inf\{h^*(y) - g^*(y) : y \in Y\}. \end{aligned}$$

2. En remplaçant g par $g^*(y) = \sup\{\langle x, y\rangle - g(x) : x \in X\}$ dans (2.12), on obtient le résultat désiré.

Puisque les fonctions h^* et g^* sont convexes, le problème (2.12) est exactement un problème de programmation DC du type (2.6). De plus, la propriété de symétrie apparaît également dans la relation entre les solutions optimales de ces problèmes. Nous avons la proposition suivante.

Proposition 2.4. *[54]*

1. *Si x^* est une solution optimale de (2.6), alors tout $y^* \in \partial h(x^*)$ est solution optimale pour le problème (2.12).*
2. *Si y^* est une solution optimale de (2.12), alors tout $x^* \in \partial g^*(y^*)$ est solution optimale pour le problème (2.6).*

■

Preuve :

1. Soit x^* une solution optimale du problème (2.6) et soit $y^* \in \partial h(x^*)$. Alors, d'après (2.10), on a $y^* \in \partial g(x^*)$. De la relation (2.14), on a

$$h^*(y^*) = \langle x^*, y^* \rangle - h(x^*), \quad g^*(y^*) = \langle x^*, y^* \rangle - g(x^*).$$

   Donc

$$h^*(y^*) - g^*(y^*) = h(x^*) - g(x^*).$$

   Ce qui implique, en utilisant (2.13), que y^* est une solution optimale du problème (2.12).
2. En prenant y^* une solution optimale du problème (2.12) et suivant le même raisonnement que dans la première partie, on obtient le résultat désiré.

2.2.4 Conditions d'optimalité en programmation DC

Soient $\mathcal{S}_P$ et $\mathcal{S}_D$ les ensembles de solutions des problèmes (2.6) et (2.12) respectivement, et posons

$$P_l = \{x^* \in X : \partial h(x^*) \subset \partial g(x^*)\} \quad \text{et} \quad D_l = \{y^* \in Y : \partial g^*(y^*) \subset \partial h^*(x^*)\}.$$

Nous avons le théorème suivant.

Théorème 2.4. *[3]*

i) Transport de minima globaux :

$$\cup\{\partial h(x) : x \in \mathcal{S}_P\} \subset \mathcal{S}_D \subset dom(h^*)$$

La première inclusion se transforme en égalité si g^* est sous-différentiable sur $\mathcal{S}_D$ (en particulier si $\mathcal{S}_D \subset ir(dom(g^*))$ ou si g^* est sous-différentiable sur $dom(h^*)$ et dans ce dernier cas $\mathcal{S}_D \subset (dom(\partial g^*) \cap dom(\partial h^*))$.

ii) Si x^* est un minimum local de $g - h$, alors $x^* \in P_l$. L'implication inverse est vraie, si h est une fonction convexe polyédrale.

iii) *Soit x^* un point critique de $g-h$ et $y^* \in \partial g(x^*) \cap \partial h(x^*)$. Soit $\mathcal{U}$ un voisinage de x^* tel que $(\mathcal{U} \cap dom(g)) \subset dom(\partial h)$. Si pour tout $x \in \mathcal{U} \cap dom(g)$ il existe un $y \in \partial h(x)$ tel que $h^*(y) - g^*(y) \geq h^*(y^*) - g^*(y^*)$, alors x^* est un minimum local de $g - h$. Plus précisément,*

$$g(x) - h(x) \geq g(x^*) - h(x^*), \quad \textit{pour tout} \quad x \in \mathcal{U} \cap dom(g).$$

iv) *Transport de minima locaux : Soit $x^* \in dom(\partial h)$ un minimum local de $g - h$ et soit $y^* \in \partial h(x^*)$. Sous l'hypothèse*

$$y^* \in int(dom(g^*)) \quad et \quad \partial g^*(y^*) \subset \mathcal{U},$$

par exemple, si g^* est différentiable en y^*, alors y^* est un minimum local de $h^* - g^*$.

■

2.2.5 Algorithme de programmation DC : DCA

Il existe deux formes de DCA : la forme complète et la forme simplifiée. En pratique, l'utilisation de la forme complète de DCA est une tâche difficile et coûteuse. Elle est donc remplacée par la forme simplifiée que nous allons décrire dans la suite.

DCA simplifié

L'algorithme DCA est basé sur l'optimalité locale et la dualité en programmation DC. Il consiste en la génération de deux suites $\{x^k\}$ et $\{y^k\}$ candidates à être solutions optimales locales des problèmes primal et dual respectivement. Ces suites sont améliorées à chaque itération de façon à vérifier les conditions suivantes :

1. Les suites $\{g(x^k)-h(x^k)\}$ et $\{h^*(y^k)-g^*(y^k)\}$ décroissent à chaque itération.
2. Si $(g-h)(x^{k+1}) = (g-h)(x^k)$(resp. $(h^*-g^*)(y^{k+1}) = (h^*-g^*)(y^k)$) l'algorithme s'arrête à l'itération $k+1$ et le point x^k (resp. y^k) est un point critique de $g-h$ (resp. h^*-g^*).
3. Sinon toute valeur d'adhérence x^* de la suite $\{x^k\}$ (resp. y^* de la suite $\{y^k\}$) est un point critique de $g-h$ (resp. h^*-g^*).

Ces suites sont générées comme suit : x^{k+1} (resp. y^k) est solution du problème convexe (P_k) (resp. (D_k)) défini par

$$(P_k) \quad \alpha_k = \inf\{g(x) - [h(x^k) + \langle x - x^k,\ y^k\rangle] \mid x \in X\}, \tag{2.18}$$

$$(D_k) \quad \inf\{h^*(y) - [g^*(y^{k-1}) + \langle x^k,\ y - y^{k-1}\rangle] \mid y \in Y\}. \tag{2.19}$$

Interprétation de DCA simplifié

DCA peut avoir l'interprétation suivante : A chaque itération, on remplace dans le programme DC primal (2.6) (resp. dual (2.12)) la fonction h (resp. g^*) par sa minorante affine définie par $h(x^k) + \langle x - x^k, y^k \rangle$ (resp. $g^*(y^k) + \langle y - y^k, x^{k+1} \rangle$), ce qui donne le problème (P_k) (resp. (D_k)).

DCA opère donc une double linéarisation en utilisant les sous-gradients de h et g^*, ce qui nous donne le schéma suivant :

$$y^k \in \partial h(x^k) \quad et \quad x^{k+1} \in \partial g^*(y^k). \tag{2.20}$$

L'algorithme qui découle de ce schéma est le suivant.

Algorithme DCA simplifié
0 : x^0 donné.
itération k
1 : Calculer $y^k \in \partial h(x^k)$.
2 : Calculer $x^{k+1} \in \partial g^*(y^k)$.
3 : Si test d'arrêt vérifié **Stop** ; sinon $k \leftarrow k+1$ et **aller** en 1.

On dira que le problème (2.6) est bien défini (i.e. α est finie et l'ensemble des solutions de (2.6) est non vide), si on peut construire les suites $\{x^k\}$ et $\{y^k\}$ à partir d'un point $x^0 \in X$ et on a le résultat suivant.

Lemme 2.3. [80]
Les suites $\{x^k\}$, $\{y^k\}$ dans l'algorithme DCA sont bien définies, si et seulement si

$$dom(\partial g) \subset dom(\partial h), \quad \text{et} \quad dom(\partial h^*) \subset dom(\partial g^*).$$

■

Dans le lemme suivant, on trouve les conditions pour que les suites $\{x^k\}$ et $\{y^k\}$ générées par DCA simplifié soient bornées.

Lemme 2.4. [3]
Si $(g - h)$ est coercive [1], alors on a :

1. la suite $\{x^k\}$ est bornée ;
2. si $\{x^k\} \subset int(dom(h))$, alors la suite $\{y^k\}$ est aussi bornée.

Par dualité, si $(h^* - g^*)$ est coercive, alors on a :

1. la suite $\{y^k\}$ est bornée ;
2. si $\{y^k\} \subset int(dom(g^*))$, alors la suite $\{x^k\}$ est aussi bornée.

■

[1] une fonction ψ est coercive si $\lim_{||x|| \to +\infty} \psi(x) = +\infty$.

Convergence de DCA simplifié

Soient $\rho_1,\ \rho_2$ deux nombres réels non négatifs. Posons $dx^k = x^{k+1} - x^k$ et $dy^k = y^{k+1} - y^k$. Le théorème suivant donne les conditions de convergence de l'algorithme DCA simplifié.

Théorème 2.5. *[3]*

1. *On suppose les suites $\{x^k\}$ et $\{y^k\}$ bien définies. Alors on a*

$$(g-h)(x^{k+1}) \leq (h^*-g^*)(y^k) - \delta_1 \leq (g-h)(x^k) - \delta_2, \tag{2.21}$$

avec

$$\begin{aligned}\delta_1 &= \max\{\frac{\rho_2}{2}||dx^k||^2,\ \frac{\rho_2^*}{2}||dy^k||^2\},\\ \delta_2 &= \max\{\frac{\rho_1+\rho_2}{2}||dx^k||^2,\ \frac{\rho_1^*}{2}||dy^{k-1}||^2 + \frac{\rho_2}{2}||dx^k||^2,\ \frac{\rho_1^*}{2}||dy^{k-1}||^2 + \frac{\rho_2^*}{2}||dy^k||^2\}.\end{aligned}$$

L'égalité $(g-h)(x^{k+1}) = (g-h)(x^k)$ a lieu si et seulement si $x^k \in \partial g^*(y^k),\ y^k \in \partial h(x^{k+1})$ et $(\rho_1+\rho_2)dx^k = \rho_1^* dy^{k-1} = \rho_2^* dy^k = 0$. Dans ce cas on a

• *$(g-h)(x^{k+1}) = (h^*-g^*)(y^k)$ et $x^k,\ x^{k+1}$ sont des points critiques de la fonction $(g-h)$ tels que*

$$y^k \in (\partial g(x^k) \cap \partial h(x^k)) \quad et \quad y^k \in (\partial g(x^{k+1}) \cap \partial h(x^{k+1})). \tag{2.22}$$

• *y^k est un point critique de (h^*-g^*) tel que*

$$[x^k, x^{k+1}] \subset (\partial g^*(y^k) \cap \partial h^*(y^k)).$$

• *$x^{k+1} = x^k$ si $\rho(g) + \rho(h) > 0$, $y^k = y^{k-1}$ si $\rho(g^*) > 0$ et $y^k = y^{k+1}$ si $\rho(h^*) > 0$.*

2. *Si α est fini alors les suites décroissantes $\{(g-h)(x^k)\}$ et $\{(h^*-g^*)(y^k)\}$ sont convergentes et ont la même limite $\beta \geq \alpha$.*

Si $\rho(g) + \rho(h) > 0$, alors $\lim\limits_{k\to+\infty}\{x^{k+1} - x^k\} = 0$.

Si $\rho(g^*) + \rho(h^*) > 0$, alors $\lim\limits_{k\to+\infty}\{y^{k+1} - y^k\} = 0$.

On a en plus

$$\lim_{k\to+\infty}\{g(x^k) + g^*(y^k) - \langle x^k, y^k\rangle\} = 0$$
$$\lim_{k\to+\infty}\{h(x^{k+1}) + h^*(y^k) - \langle x^{k+1}, y^k\rangle\} = 0$$

3. *Si α est fini et si les suites $\{x^k\}$ et $\{y^k\}$ sont bornées alors pour toute valeur d'adhérence x^* de $\{x^k\}$ (resp. y^* de $\{y^k\}$), il existe une valeur d'adhérence y^* de $\{y^k\}$ (resp. x^* de $\{x^k\}$) telle que*
   (a) $y^* \in (\partial g(x^*) \cap \partial h(x^*))$ *et* $g(x^*) - h(x^*) = \beta$;
   (b) $x^* \in (\partial g^*(y^*) \cap \partial h^*(y^*))$ *et* $h^*(y^*) - g^*(y^*) = \beta$.

■

2.2.6 Optimisation DC polyédrale et convergence finie de DCA

Programmation DC polyédrale

On dit qu'un programme DC est polyhédral, si une au moins des composantes DC de $f = g - h$ est une fonction convexe polyédrale (voir définition 2.2). Cette classe de programmes est fréquemment rencontrée en pratique et possède des propriétés intéressantes, notamment, la convergence finie de l'algorithme DCA [45, 83].

Dans la suite, nous supposons que la valeur optimale α du problème DC défini par (2.6) est finie, ce qui implique que $dom(g) \subset dom(h) = C$.
Supposons que h est une fonction convexe polyédrale donnée par définition 2.2. Alors (2.6) s'écrit sous la forme

$$\alpha = \inf\{f(x) = g(x) - \tilde{h}(x) : x \in X\} \quad (\tilde{P}),$$

où

$$\tilde{h}(x) = \max\left\{\langle a_i, x\rangle - b_i,\ i = \overline{1,m}\right\},\ \forall x \in X.$$

On aura donc

$$\begin{aligned} \alpha &= \inf_{x\in X}\inf_{i\in\mathcal{M}}\{f(x) = g(x) - \langle a_i, x\rangle - b_i\} \\ &= \inf_{i\in\mathcal{M}}\inf_{x\in X}\{f(x) = g(x) - \langle a_i, x\rangle - b_i\}, \end{aligned}$$

avec $\mathcal{M} = \{1, \ldots, m\}$.

Pour chaque $i \in \mathcal{M}$, nous avons

$$\alpha(i) = \inf_{x\in X}\{f(x) = g(x) - \langle a_i, x\rangle - b_i\} \quad (\tilde{P}_i).$$

On remarque que l'ensemble des solutions du problème $(\tilde{P}_i)$ est $\partial g^*(a_i)$.

Définissons les ensembles

$$\mathcal{M}(\alpha) = \{i \in \mathcal{M} : \alpha(i) = \alpha\}.$$

$$\mathcal{M}(x) = \{i \in \mathcal{M} : \langle a_i, x\rangle - b_i = \tilde{h}(x)\}.$$

Alors, l'ensemble $\tilde{\mathcal{P}}$ des solutions de $(\tilde{P})$ est donné par

$$\alpha = \min\{\alpha(i) : i \in \mathcal{M}\}.$$

$$\tilde{\mathcal{P}} = \cup\{\tilde{\mathcal{P}}_i : i \in \mathcal{M}(\alpha)\} = \cup\{\partial g^*(a_i) : i \in \mathcal{M}(\alpha)\}.$$

Ainsi la résolution du problème DC $(\tilde{P})$ revient à résoudre les problèmes convexes $(\tilde{P}_i)$ pour $i \in \mathcal{M}$.

Le problème dual $(\tilde{D})$ de $(\tilde{P})$ est donné par

$$\alpha = \inf\{\tilde{h}^*(y) - g^*(y) : y \in Conv\{a_i : i \in \mathcal{M}\}\} \quad (\tilde{D})$$

où $Conv\{a_i : i \in \mathcal{M}\} = dom(\tilde{h}^*)$ et la valeur optimale α vérifie

$$\alpha = \inf\{\tilde{h}^*(a_i) - g^*(a_i) : i \in \mathcal{M}\}.$$

Nous avons les résultats suivants.

Théorème 2.6. *[80]*

i) $x^* \in \tilde{\mathcal{P}}$ *si et seulement si* $\mathcal{M}(x^*) \subset \mathcal{M}(\alpha)$ *et* $x^* \in \cup\{\partial g^*(a_i) : i \in \mathcal{M}(x^*)\}$.

ii) $\tilde{\mathcal{P}} = \cup\{\tilde{\mathcal{P}}_i : i \in \mathcal{M}(\alpha)\}$. *Si* $\{a_i : i \in \mathcal{M}\} \subset dom(\partial g^*)$, *alors* $\tilde{\mathcal{P}} \neq \emptyset$.

∎

Lemme 2.5. [80]

i) $\tilde{h}^*(a_i) \leq b_i,\ \forall i \in \mathcal{M}$. L'égalité aura lieu si et seulement si il existe $x \in X$ tel que $i \in \mathcal{M}(x)$.

ii) $\tilde{h}(x) = \max\{\langle x, y\rangle - \tilde{h}^*(y) : y \in Conv\{a_i : i \in \mathcal{M}\}\} = \max\{\langle a_i, x\rangle - \tilde{h}^*(a_i) : i \in \mathcal{M}\}$.

∎

Convergence finie de DCA

La résolution (d'une manière globale) du problème d'optimisation DC polyédrale $(\tilde{P})$ revient à résoudre m programmes convexes $(\tilde{P}_i)$, avec $i \in \mathcal{M}$. Pour générer $\tilde{\mathcal{P}}$, on peut déterminer, en premier lieu, $\mathcal{M}(\alpha)$ et ensuite appliquer le théorème 2.6. En pratique, ceci est possible si m est relativement petit. Dans le cas où m est grand, on utilise la forme simplifiée de l'algorithme DCA pour résoudre (localement) $(\tilde{P})$.

D'après le lemme 2.3, l'algorithme DCA simplifié est bien défini si et seulement si $Conv\{a_i : i \in \mathcal{M}\} \subset \partial dom(\partial g^*)$. Puisque α est finie, nous avons $dom(g) \subset dom(h) = C$ et $Conv\{a_i : i \in \mathcal{M}\} \subset \partial dom(\partial g^*)$. Dans ce cas, DCA est décrit comme suit :
Soit x^0 choisi au départ. Posons

$$y^k \in \partial\tilde{h}(x^k) = Conv\{a_i : i \in \mathcal{M}(x^k)\}; \quad x^{k+1} \in \partial g^*(y^k).$$

En posant $y^k = a_i, \quad i \in \mathcal{M}(x^k)$, le calcul de x^{k+1} est réduit à la résolution du programme convexe

$$\min\{g(x) - \langle y^k, x\rangle : x \in X\}.$$

Si $y^k = a_i$ avec $i \in \mathcal{M}(\alpha)$, alors, en utilisant le théorème 2.6, on obtient $x^{k+1} \in \tilde{\mathcal{P}}$.

Soient $\tilde{H}$ et G^* deux applications définies dans $dom(\partial\tilde{h}) = X$ et dans $dom(\partial g^*)$ respectivement et telles que

$$\tilde{H}(x) \in \partial\tilde{h}(x), \ \forall x \in X \quad \text{et} \quad G^*(y) \in \partial g^*(y), \ \forall y \in dom(\partial g^*(y)).$$

L'algorithme DCA simplifié, avec un choix fixé des sous-gradients, est donné par [79, 80] :

$$y^k = \tilde{H}(x^k); \quad x^{k+1} = G^*(y^k).$$

Dans le cas où h et g sont polyédrales convexes, les séquences $\{x^k\}$ et $\{y^k\}$ sont discrètes (i.e., elles possèdent uniquement un nombre fini d'éléments). Nous avons le théorème de convergence de DCA dans le cas de la programmation DC polyédrale.

Théorème 2.7. *[80]*

i) Les suites discrètes $\{(g - \tilde{h})(x^k)\}$ et $\{(\tilde{h}^* - g^*)(y^k)\}$ sont décroissantes et convergentes.

ii) Les suites discrètes $\{x^k\}$ et $\{y^k\}$ sont de la même nature : soit elles sont convergentes, soit elles sont cycliques de même période p. Dans ce dernier cas, les suites $\{x^k\}$ et $\{y^k\}$ contiennent exactement p points limites qui sont tous des points critiques de $g - h$. De plus, si $\rho(g) + \rho(g^*) > 0$, alors ces suites sont convergentes.

■

3
Méthodes de pénalité

3.1 Introduction

Les méthodes de pénalité sont des procédures utilisées pour approximer des problèmes d'optimisation avec contraintes par des problèmes sans contraintes. Lorsque cette approximation est accomplie en ajoutant à la fonction objectif un terme qui pénalise toute violation des contraintes, on dit que la pénalité est *extérieure*. Dans le cas de la pénalité *intérieure* ou *barrière*, on ajoute à la fonction objectif un terme qui favorise les points intérieurs à l'ensemble réalisable sur ceux proches de la frontière ; en d'autres termes il évite de sortir de l'ensemble réalisable.

Un paramètre appelé *paramètre de pénalité* est associé à ces méthodes. Il détermine la sévérité de la pénalité ou de la barrière, et par conséquent, le degré avec lequel le problème pénalisé (sans contraintes) approxime le problème original (avec contraintes). Plus le paramètre de pénalité est grand, plus l'approximation est bonne (proche de la solution du problème original).

D'autres part, il existe quelques fonctions de pénalité particulières pour lesquelles la résolution du problème pénalisé donne exactement la solution du problème original, pour une certaine valeur finie du paramètre de pénalité. Dans ce cas, on parle de pénalité *exacte*.

Dans ce chapitre, nous allons décrire brièvement ces méthodes et les illustrer

avec quelques exemples simples. Plus de détails sur ces méthodes peuvent être trouvés dans [60, 19, 75].

Pour l'étude de ces méthodes, considérons le problème

$$\begin{cases} \min f(x) \\ x \in S \end{cases} \tag{3.1}$$

3.2 Pénalité extérieure

Supposons que dans le problème (3.1), f est continue sur $\mathbb{R}^n$ et S est un ensemble de contraintes dans $\mathbb{R}^n$. L'idée de la méthode de pénalité est de remplacer le problème (3.1) par le problème sans contraintes de la forme

$$\min \big(f(x) + cP(x)\big) \tag{3.2}$$

où c est une constante positive appelée *paramètre de pénalité* et P est une fonction qui peut être définie formellement comme suit :

Définition 3.1. Une fonction $P : \mathbb{R}^n \longrightarrow \mathbb{R}$ est appelée fonction de pénalité si elle satisfait les conditions :

i) P est continue.

ii) $P(x) \geq 0$ pour tout $x \in \mathbb{R}^n$.

iii) $P(x) = 0$ si et seulement si $x \in S$.

En supposant que $S = \{x \in \mathbb{R}^n,\ g_i(x) \leq 0,\ i = \overline{1,m}\}$, la fonction P est généralement définie par la relation

$$P(x) = \sum_{i=1}^{m} \Big[\max\{0, g_i(x)\}\Big]^{\beta} \tag{3.3}$$

où β est une constante non négative.

Exemple 3.1. Soit l'ensemble S défini comme précédemment, en prenant dans (3.3), $\beta = 2$, on a la fonction pénalité suivante.

$$P(x) = \frac{1}{2} \sum_{i=1}^{m} \Big[\max\{0, g_i(x)\}\Big]^{2} \tag{3.4}$$

Cette fonction est illustrée dans la figure 3.1 avec

$$\begin{cases} g_1(x) = x - b \\ g_2(x) = a - x \end{cases}$$

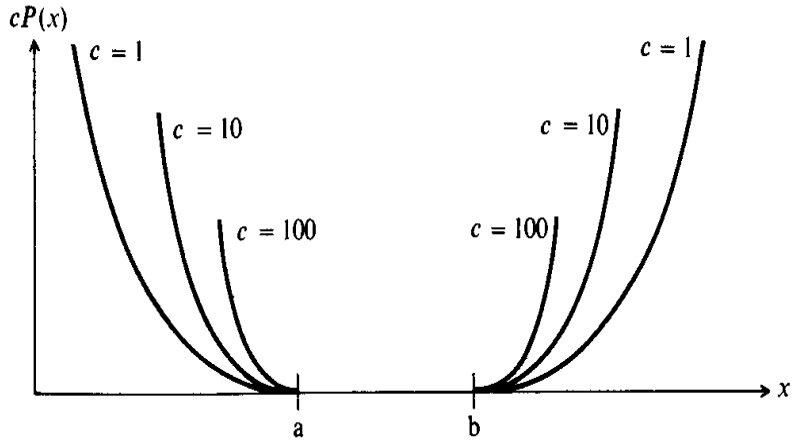

FIG. 3.1 – Représentation de $cP(x)$

3.2.1 Description de la méthode

Soit $\{c_k\}$, $k = 1, 2, \ldots$ une séquence qui tend vers l'infini telle que pour tout k on a $c_k \geq 0$, $c_{k+1} > c_k$. On définit la fonction

$$q(c_k, x) = f(x) + c_k P(x) \tag{3.5}$$

Pour tout k, on résout le problème

$$\min q(c_k, x) \tag{3.6}$$

et on obtient x_k.
On suppose que pour tout k, le problème (3.6) admet une solution.

3.2.2 Convergence de la méthode

Le lemme suivant nous donne quelques inégalités qui découlent de la définition du point x_k.

Lemme 3.1. [60]
Supposant que c_k est une séquence telle que $c_{k+1} > c_k$. Alors pour tout k, on a

$$q(c_k, x_k) \leq q(c_{k+1}, x_{k+1}) \tag{3.7}$$

$$P(x_k) \geq P(x_{k+1}) \tag{3.8}$$

$$f(x_k) \leq f(x_{k+1}) \tag{3.9}$$

■

Preuve Nous avons

$$\begin{array}{rcl} q(c_{k+1}, x_{k+1}) & = & f(x_{k+1}) + c_{k+1}P(x_{k+1}) \\ & \geq & f(x_{k+1}) + c_k P(x_{k+1}) \quad (\text{car} \quad c_{k+1} > c_k) \\ & \geq & f(x_k) + c_k P(x_k) \quad (\text{car} \quad x_k \quad \text{minimise} \quad q(c_k, x)) \\ & = & q(c_k, x_k), \end{array}$$

ce qui prouve (3.7).

Nous avons également

$$f(x_k) + c_k P(x_k) \leq f(x_{k+1}) + c_k P(x_{k+1}) \tag{3.10}$$

$$f(x_{k+1}) + c_{k+1}P(x_{k+1}) \leq f(x_k) + c_{k+1}P(x_k) \quad (\text{car} \quad x_{k+1} \quad \text{minimise} \quad q(c_{k+1}, x)) \tag{3.11}$$

En additionnant (3.10) et (3.11), on aura

$$(c_{k+1} - c_k)P(x_{k+1}) \leq (c_{k+1} - c_k)P(x_k)$$

Donc $P(x_{k+1}) \leq P(x_k)$, ce qui démontre (3.8).

Puisque x_k minimise $q(c_k, x)$, on obtient

$$f(x_{k+1}) + c_k P(x_{k+1}) \geq f(x_k) + c_k P(x_k)$$

Donc

$$f(x_{k+1}) \geq f(x_k) + c_k(P(x_k) - P(x_{k+1})),$$

ce qui donne $f(x_{k+1}) \geq f(x_k)$ (car $P(x_k) \geq P(x_{k+1})$ et $c_k \geq 0$).

■

Lemme 3.2. [60]
Soit x^* une solution du problème (3.1). Alors, pour chaque k, on a

$$f(x^*) \geq q(c_k, x_k) \geq f(x_k)$$

■

Preuve : Le fait que x_k minimise $q(c_k, x)$ nous donne

$$f(x^*) + c_k P(x^*) \geq f(x_k) + c_k P(x_k) = q(c_k, x_k)$$

Puisque x^* est solution de (3.1), nous avons $P(x^*) = 0$.
Par conséquent, $f(x^*) \geq f(x_k) + c_k P(x_k)$.
Puisque $P(x_k) \geq 0$ et $c_k \geq 0$, on a

$$f(x^*) \geq q(c_k, x_k) \geq f(x_k),$$

ce qui démontre le lemme.

■

En se basant sur les lemmes 3.1 et 3.2, le théorème suivant établit la convergence de la méthode.

Théorème 3.1. *[60]*
Soit $\{x_k\}$ une séquence générée par la méthode de pénalité. Alors, tout point limite de cette séquence est une solution de (3.1).

■

Preuve : Supposons qu'une sous-suite $\{x_k\}$, $k \in \mathcal{K}$ de $\{x_k\}$ est convergente. Soit $\bar{x}$ la limite de cette sous suite. Donc, par continuité de f, on a

$$\lim_{k\in\mathcal{K}} f(x_k) = f(\bar{x}). \tag{3.12}$$

Soit f^* la valeur optimale du problème (3.1). D'après les lemmes précédents, la séquence de valeurs $q(c_k, x_k)$ est non décroissante et bornée supérieurement par f^*. Donc

$$\lim_{k\in\mathcal{K}} q(c_k, x_k) = q^* \leq f^* \tag{3.13}$$

En soustrayant (3.12) de (3.13), on aura

$$\lim_{k\in\mathcal{K}} c_k P(x_k) = q^* - \bar{f} \tag{3.14}$$

Puisque, $P(x_k) \geq 0$ et $c_k \to \infty$, en utilisant (3.14), on aura

$$\lim_{k\in\mathcal{K}} P(x_k) = 0.$$

En utilisant la continuité de P, ceci implique $P(\bar{x}) = 0$, ce qui démontre que tout point limite $\bar{x}$ est réalisable pour (3.1).
D'après le lemme 3.2, $f(x_k) \leq f^*$ et donc

$$f(\bar{x}) = \lim_{k\in\mathcal{K}} f(x_k) \leq f^*,$$

ce qui preuve que $\bar{x}$ est solution de (3.1).

■

3.3 Pénalité intérieure (barrière)

Soit le problème (3.1) et supposons que S possède un intérieur non vide (i.e. $int(S) \neq \emptyset$). La méthode de fonction barrière fonctionne en établissant une barrière sur la frontière de la région réalisable qui évite la sortie de cette région. Nous pouvons définir une fonction barrière comme suit.

Définition 3.2. Une fonction barrière $B : \mathbb{R}^n \longrightarrow \mathbb{R}$ est une fonction qui satisfait :

i) B est continue.

ii) $B(x) \geq 0$.

iii) $B(x) \longrightarrow \infty$, quand x s'approche de la frontière S.

■

Supposons que l'ensemble S est défini comme dans l'exemple 3.1, les fonctions $g_i, i = \overline{1,m}$ sont continues sur $\mathbb{R}^n$ et que l'intérieur de S est l'ensemble des points x où $g_i(x) < 0, i = \overline{1,m}$, i.e. $S = \{x \in \mathbb{R}^n,\ g_i(x) < 0,\ i = \overline{1,m}\}$. Les deux fonctions barrière les plus utilisées sont :

$$B(x) = -\sum_{i=1}^{m} \frac{1}{g_i(x)}, \quad x \in int(S). \tag{3.15}$$

$$B(x) = -\sum_{i=1}^{m} \ln\{-g_i(x)\}, \quad x \in int(S). \tag{3.16}$$

Exemple 3.2. Comme exemple, nous allons illustrer la fonction (3.15) (voir figure (3.2)), avec

$$\begin{cases} g_1(x) = x - a \\ g_2(x) = x - b \end{cases}$$

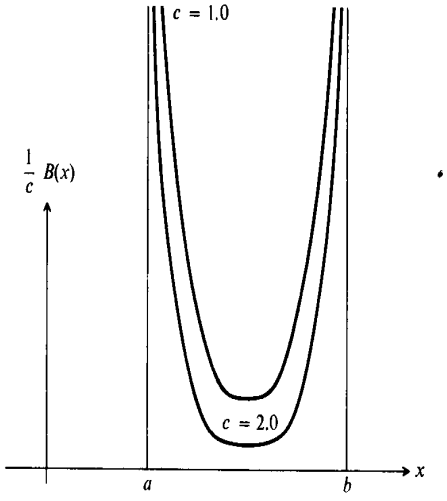

FIG. 3.2 – Représentation de $\frac{1}{c}B(x)$

3.3.1 Description de la méthode

La méthode de barrière est tout à fait analogue à la méthode de pénalité. Soit $\{c_k\}$ une séquence qui tend vers l'infini telle que pour tout k, $k = 1, 2, \ldots$ on a $c_k \geq 0$, $c_{k+1} > c_k$. On définit la fonction

$$r(c, x) = f(x) + \frac{1}{c}B(x) \tag{3.17}$$

Pour chaque k, on résout le problème

$$\begin{cases} \min r(c_k, x) \\ x \in int(S) \end{cases} \tag{3.18}$$

et on obtient x_k.

3.3.2 Convergence de la méthode

La méthode de barrière possède pratiquement les mêmes propriétés de convergence que la méthode de pénalité. Nous avons le théorème suivant.

Théorème 3.2. *[60]*
Tout point limite de la séquence $\{x_k\}$ générée par la méthode de barrière est une solution du problème (3.1).

■

3.4 Pénalité exacte

Il est possible de construire des fonctions de pénalité exacte dans le sens où, la solution du problème pénalisé est exactement la solution du problème original pour une certaine valeur finie du paramètre de pénalité. L'avantage avec cette classe de fonctions, est qu'il n'est pas nécessaire de résoudre une suite infinie de problèmes pénalisés pour obtenir la solution. Cependant, la difficulté qui apparaît est la non différentiabilité des fonctions utilisées.

Cette classe de fonctions a été introduite, pour la première fois, par Zangwill [93] où la fonction pénalité est obtenue en prenant dans (3.3), $\beta = 1$, ce qui donne

$$P(x) = \sum_{i=1}^{m} \max\{0, g_i(x)\}$$

Dans ce cas, il a été montré par le même auteur qu'il existe un c_0 suffisamment large tel que pour tout $c > c_0$, la résolution du problème (3.2) donne exactement

la solution de (3.1). Cette fonction est non différentiable. Elle est illustrée dans la figure (3.3) avec le problème

$$\begin{cases} \min f(x) = \alpha x \\ \quad g(x) = b - x \leq 0 \end{cases}$$

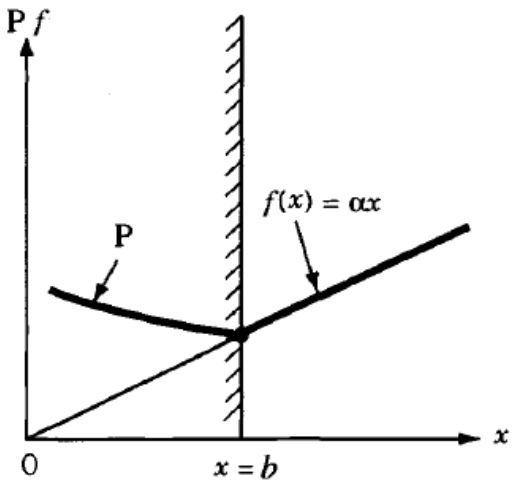

FIG. 3.3 – Représentation de la fonction $P(x)$

Considérons le problème d'optimisation avec contraintes :

$$\begin{cases} \min \ f(x) \\ \quad h_i(x) = 0 \quad i \in \{1, \ldots, s\} \\ \quad g_j(x) \leq 0 \quad j \in \{1, \ldots, m\} \end{cases} \tag{3.19}$$

La fonction de pénalité la plus populaire pour ce problème est la fonction valeur absolue (ou l_1) définie par

$$P(x) = \sum_{i=1}^{s} |h_i(x)| + \sum_{j=1}^{m} \max\{0, g_j(x)\}. \tag{3.20}$$

Le problème pénalisé est donc donné par

$$\min f(x) + cP(x) \tag{3.21}$$

pour une certaine constante positive c.

Comme nous l'avons déjà mentionné, la solution du problème (3.19) est obtenue en résolvant (3.21), pour une certaine valeur finie de c. Nous allons voir cette propriété à travers l'exemple suivant.

Exemple 3.3. Considérons le problème

$$\begin{cases} \min \ 2x^2 + 2xy + y^2 - 2y \\ \quad x = 0. \end{cases} \tag{3.22}$$

La solution de ce problème est obtenue directement en remplaçant $x = 0$ dans la fonction objectif. Ce qui nous donne $(x^*, y^*) = (0, 1)$.

Si on applique une pénalité quadratique à ce problème, on obtient le problème pénalisé

$$\min \ 2x^2 + 2xy + y^2 - 2y + \frac{1}{2}cx^2 \tag{3.23}$$

pour $c > 0$. La solution de (3.23) est $x = -2/(2 + c)$ et $y = 1 - 2/(2 + c)$. Quand $c \to \infty$, cette solution devient proche de la solution du problème initial (3.22). C'est ce qui est prévu par la théorie des méthodes de pénalité et qui n'est pas vrai pour une valeur finie de c.

Utilisons maintenant la fonction l_1 définie par (3.20). Le problème pénalisé devient

$$\min \ 2x^2 + 2xy + y^2 - 2y + c|x|. \tag{3.24}$$

Ecrivons (3.24) sous la forme

$$\begin{aligned} 2x^2 + 2xy + y^2 - 2y + c|x| &= 2x^2 + 2xy + c|x| + (y-1)^2 - 1 \\ &= 2x^2 + 2x + c|x| + (y-1)^2 + 2x(y-1) - 1 \\ &= x^2 + (2x + c|x|) + (y - 1 + x)^2 - 1 \end{aligned} \tag{3.25}$$

Si $c > 2$, tous les termes dans (3.25) sont non négatifs (sauf -1). Donc, la valeur minimale de cette expression est -1, qui est atteinte pour $x = 0, y = 1$.

Par conséquent, pour $c > 2$, la solution du problème pénalisé (3.24) est exactement la solution du problème original (3.22).

Le théorème suivant démontre que, sous certaines hypothèses de convexité, il existe une valeur finie de c pour laquelle on obtient la solution du problème (3.19) en résolvant (3.21).

Théorème 3.3. *[19]*
Considérons le problème (3.19). Soit x^* un point vérifiant le système KKT associé à (3.19) avec des multiplicateurs de Lagrange u_i, $i \in I$, et v_i, $i = 1, \ldots, m$ associés aux contraintes d'inégalité et d'égalité respectivement, et où $I = \{i \in \{1, \ldots, m\} : g_i(x^*) = 0\}$ est l'ensemble des indices des contraintes actives. De plus, supposons que f et g_i, $i \in I$ sont des fonctions convexes et que h_i, $i = 1, \ldots, s$ sont affines. Alors, pour $c \geq \max\{u_i, i \in I, \ |v_i|, i = 1, \ldots, s\}$, x^* est solution du problème pénalisé défini par (3.21).

■

Il peut être montré aussi que si x^* vérifie les conditions suffisantes de second ordre pour un minimum local du problème (3.19) (voir [19], *Théorème 4.4.2*), alors pour une valeur de c donnée par le théorème 3.3, x^* est aussi un minimum de (3.21).

3.5 Remarques sur les méthodes de pénalité

Dans les méthodes de pénalité, la solution du problème pénalisé devient proche de la solution optimale du problème original en choisissant une valeur du paramètre de pénalité c suffisamment large. Cependant, il n'est pas connu au départ quelle grandeur est suffisante. En effet, le choix d'une valeur trop grande de c peut provoquer des difficultés de calcul liées au mal conditionnement de la fonction de pénalité. Dans ce cas, la plupart des algorithmes d'optimisation sans contraintes convergent rapidement vers un point réalisable. Bien que ce point soit loin de l'optimum, l'algorithme de recherche peut s'arrêter.

En pratique, pour éviter ces difficultés numériques, les algorithmes utilisant les fonctions de pénalité commencent par un paramètre de pénalité de valeur pas trop élevée et l'augmente à chaque itération, par exemple, en le multipliant par un scalaire positif. De plus, cette augmentation ne devrait pas être trop rapide au point de provoquer le mal conditionnement de la fonction.

4

La méthode DC pour la résolution du (PBL)

Introduction

Le problème de programmation bi-niveaux linéaire (PBL) est un problème d'optimisation non convexe difficile. Le but de cette partie est de présenter une approche pour la résolution du (PBL) . Cette approche est basée sur la transformation de ce dernier en un programme DC en utilisant une pénalité exacte passant par sa reformulation en un programme à un seul niveau. Les résultats obtenus dans ce chapitre seront comparés aux résultats d'un autre algorithme de résolution des problèmes de programmetion bi-niveaux linéaires.

4.1 Reformulation du (PBL)

Avant d'aborder la résolution, nous allons d'abord développer la formulation du (PBL) en un problème d'optimisation à un seul niveau.

Soit le problème de programmation bi-niveaux linéaire :

$$(PBL)\quad \begin{cases} \max\limits_{x} \; F(x,y) = c_1^t x + d_1^t y, \\ \quad s.c. \;\; A_1 x + B_1 y \leq b_1, \\ \qquad x \geq 0; \\ \qquad \max\limits_{y} \; f(x,y) = c_2^t x + d_2^t y, \\ \qquad\quad s.c. \;\; A_2 x + B_2 y \leq b_2, \\ \qquad\qquad y \geq 0, \end{cases} \tag{4.1}$$

où $F : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \longrightarrow \mathbb{R}$; $f : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \longrightarrow \mathbb{R}$ sont les fonctions objectifs du Leader et du Suiveur respectivement ; $c_1, c_2 \in \mathbb{R}^{n_1}$; $d_1, d_2 \in \mathbb{R}^{n_2}$; $b_1 \in \mathbb{R}^{m_1}$, $b_2 \in \mathbb{R}^{m_2}$; A_1-$m_1 \times n_1$-matrice, B_1-$m_1 \times n_2$-matrice, A_2-$m_2 \times n_1$-matrice et B_2-$m_2 \times n_2$-matrice.

Le problème du Suiveur, pour une décision x du Leader, est donné par

$$(P)\begin{cases} \max\limits_{y} \; f(x,y) = c_2^t x + d_2^t y \\ \quad s.c. \;\; A_2 x + B_2 y \leq b_2, \\ \qquad y \geq 0. \end{cases}$$

Considérons la fonction de Lagrange associée au problème (P) :

$$L(y,u,v) = -c_2^t x - d_2^t y + u^t(A_2 x + B_2 y - b_2) - v^t y,$$

où $u \in \mathbb{R}^{m_2}$, $v \in \mathbb{R}^{n_2}$ sont les variables duales associées aux contraintes $A_2 x + B_2 y \leq b_2$ et $y \geq 0$ respectivement. Alors une condition nécessaire et suffisante pour que (x^*, y^*) soit solution de (P) est qu'il existe des vecteurs $u^* \geq 0$ et $v^* \geq 0$ tels que (x^*, y^*, u^*, v^*) soit solution du système :

$$A_2 x + B_2 y \leq b_2, \tag{4.2}$$
$$\frac{\partial L(y,u,v)}{\partial y} = -d_2 + B_2^t u - v = 0, \tag{4.3}$$
$$u^t(A_2 x + B_2 y - b_2) = 0, \tag{4.4}$$
$$-v^t y = 0, \tag{4.5}$$
$$x \geq 0,\; y \geq 0,\; u \geq 0,\; v \geq 0. \tag{4.6}$$

Soit $w \geq 0$ une variable d'écart,

$$(4.2) \Leftrightarrow A_2 x + B_2 y + w = b_2 \tag{4.7}$$

$(4.7) \quad \Rightarrow \quad w = b_2 - A_2 x - B_2 y \quad \Rightarrow \quad w = -(A_2 x + B_2 y - b_2)$
$(4.4) \quad \Leftrightarrow \quad -u^t w = 0 \Leftrightarrow u^t w = 0$

$$(4.3) \quad \Leftrightarrow \quad B_2^t u - v = d_2$$
$$(4.5) \quad \Leftrightarrow \quad v^t y = 0$$

Donc le système (4.2)-(4.6) peut être écrit :

$$\begin{cases} A_2 x + B_2 y + w = b_2 \\ B_2^t u - v = d_2 \\ u^t w + v^t y = 0 \\ x \geq 0,\ y \geq 0,\ u \geq 0,\ v \geq 0,\ w \geq 0. \end{cases}$$

En remplaçant (P) par ce système dans le (PBL) , on obtient un programme mathématique avec une seule fonction objectif sous des contraintes non linéaires de la forme suivante :

$$(PBL)_{KKT} \begin{cases} \max\limits_{x,y} \ F(x,y) = c_1^t x + d_1^t y \\ \qquad A_1 x + B_1 y \leq b_1 \\ \qquad A_2 x + B_2 y + w = b_2 \\ \qquad B_2^t u - v = d_2 \\ \qquad v^t y + u^t w = 0 \\ \qquad x \geq 0,\ y \geq 0,\ u \geq 0,\ v \geq 0,\ w \geq 0. \end{cases} \tag{4.8}$$

En utilisant la proposition 1.1 du chapitre 1, nous aurons la relation entre ce programme et le (PBL) .

Dans la suite, nous allons écrire le $(PBL)_{KKT}$ sous une forme plus explicite.

Commençons par introduire une variable d'écart $e \geq 0$. Le $(PBL)_{KKT}$ est alors équivalent à :

$$(PBL)'_{KKT} \begin{cases} \max\limits_{x,y} \ F(x,y) = c_1^t x + d_1^t y \\ \qquad A_1 x + B_1 y + e = b_1 \\ \qquad A_2 x + B_2 y + w = b_2 \\ \qquad B_2^t u - v = d_2 \\ \qquad v^t y + u^t w = 0 \\ \qquad x \geq 0,\ y \geq 0,\ u \geq 0,\ v \geq 0,\ w \geq 0, e \geq 0. \end{cases}$$

Ensuite, nous allons adopter les notations suivantes.

$$z = \begin{pmatrix} x & y & e & w & v & u \end{pmatrix}^t \in \mathbb{R}^n, \quad c = \begin{pmatrix} -c_1 & -d_1 & 0 & 0 & 0 & 0 \end{pmatrix}^t \in \mathbb{R}^n,$$

$$A = \begin{pmatrix} A_1 & B_1 & I_{m_1} & 0 & 0 & 0 \\ A_2 & B_2 & 0 & I_{m_2} & 0 & 0 \\ 0 & 0 & 0 & 0 & -I_{n_2} & B_2^t \end{pmatrix} \in R^{m \times n}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ d_2 \end{pmatrix} \in \mathbb{R}^m,$$

$$E_u = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & I_{m_2} \end{pmatrix}, \quad E_v = \begin{pmatrix} 0 & 0 & 0 & 0 & I_{n_2} & 0 \end{pmatrix},$$

$$E_w = \begin{pmatrix} 0 & 0 & 0 & I_{m_2} & 0 & 0 \end{pmatrix}, \quad E_y = \begin{pmatrix} 0 & I_{n_2} & 0 & 0 & 0 & 0 \end{pmatrix},$$

où I_k est une $k \times k$ matrice identité ;
0 est la matrice nulle avec la dimension appropriée pour chaque cas,
avec $n = n_1 + 2n_2 + m_1 + 2m_2$; $m = m_1 + m_2 + n_2$.

En utilisant ces notations, on aura :

$$u^t w = (E_u z)^t (E_w z) = z^t (E_u^t E_w) z = z^t D^1 z \text{ avec } E_u^t E_w = D^1,$$
$$v^t y = (E_v z)^t (E_y z) = z^t (E_v^t E_y) z = z^t D^2 z \text{ avec } E_v^t E_y = D^2.$$

Donc,

$$u^t w + v^t y = z^t D^1 z + z^t D^2 z = z^t (D^1 + D^2) z = z^t D z \text{ avec } D^1 + D^2 = D.$$

Notons que les éléments $d_{ij} (i = \overline{1,n},\ j = \overline{1,n})$ de la matrice D sont tous non négatifs, *i.e*,

$$d_{ij} \geq 0 \quad \forall i, j = \overline{1,n}. \tag{4.9}$$

Le $(PBL)'_{KKT}$ peut être réécrit comme suit :

$$P(z) \begin{cases} \min F(z) = c^t z \\ Az = b, \\ z^t D z = 0, \\ z \geq 0. \end{cases}$$

En posant

$$Dz = q(z) \Longrightarrow D_i z = q_i(z) \text{ avec } D_i \text{ est la } i-ème \text{ ligne de } D,$$

on aura :

$$z^t D z = 0 \Longleftrightarrow z^t q(z) = 0.$$

Donc, le problème $P(z)$ s'écrit :

$$\begin{cases} \min F(z) = c^t z \\ Az = b, \\ z^t q(z) = 0, \\ z \geq 0. \end{cases}$$

Comme $d_{ij} \geq 0 \quad \forall i, j = \overline{1,n}$, alors $\forall z \geq 0$, on a $q(z) \geq 0$.
Les fonctions $q_i(z), i = \overline{1,n}$ sont linéaires sur $\mathbb{R}^n$.
A présent, définissons la fonction

$$\Psi(z) = \sum_{i=1}^{n} \Psi_i(z) = \sum_{i=1}^{n} \min\{q_i(z),\ z_i\}.$$

Proposition 4.1. Ψ *est concave sur* $\mathbb{R}^n$.

■

Preuve : Soient $z^1,\ \ z^2 \in \mathbb{R}^n,\ \ \lambda \in [0,\ 1]$, et $z^0 = \lambda\ z^1 + (1-\lambda)\ z^2$. On a

$$\begin{aligned}
\Psi_i(z^0) &= \min\{q_i(z^0),\ z_i^0\} \\
&= \min\{q_i(\lambda\ z^1 + (1-\lambda)\ z^2),\ \lambda\ z_i^1 + (1-\lambda)\ z_i^2\} \\
&= \min\{\lambda q_i(z^1) + (1-\lambda) q_i(z^2),\ \lambda z_i^1 + (1-\lambda) z_i^2)\} \\
&\geq \lambda \min\{q_i(z^1),\ z_i^1\} + (1-\lambda) \min\{q_i(z^2),\ z_i^2\} \\
&= \lambda \Psi_i(z^1) + (1-\lambda)\Psi_i(z^2).
\end{aligned}$$

D'où

$$\Psi(z^0) = \sum_{i=1}^{n} \Psi_i(z^0) \geq \lambda \sum_{i=1}^{n} \Psi_i(z^1) + (1-\lambda) \sum_{i=1}^{n} \Psi_i(z^2) = \lambda\Psi(z^1) + (1-\lambda)\Psi(z^2).$$

Donc la fonction Ψ est concave sur $\mathbb{R}^n$.

■

Posons

$$\mathcal{Z} = \{z \in \mathbb{R}^n, Az = b, z \geq 0\}.$$

Puisque Ψ est une fonction concave, finie et non négative sur l'ensemble $\mathcal{Z}$, nous avons :

$$\{z \in \mathcal{Z},\ z^t q(z) = 0\} = \{z \in \mathcal{Z},\ \Psi(z) \leq 0\}.$$

Notre problème est donc équivalent à :

$$\begin{cases} \min F(z) = c^t z \\ \quad Az = b, \\ \quad z \geq 0, \\ \quad \Psi(z) \leq 0, \end{cases}$$

qu'on peut écrire

$$\alpha = \min\{F(z)\ :\ z \in \mathcal{Z},\ \Psi(z) \leq 0\}, \tag{4.10}$$

où $\mathcal{Z}$ est un ensemble convexe supposé non vide. Les fonctions F et Ψ sont des fonctions concaves sur $\mathcal{Z}$.

4.2 Reformulation via une pénalité exacte

Le problème (4.10) n'est pas nécessairement un programme convexe. Pour sa résolution, on essaie d'utiliser une pénalité exacte qui nous donne le problème :

$$\alpha(k) = \min\{F(z) + k\Psi^+(z) : z \in \mathcal{Z}\}, \tag{4.11}$$

où k est un nombre positif, appelé coefficient de pénalité et $\Psi^+(z) = \max\{0,\ \Psi(z)\}$.
La pénalité exacte a lieu s'il existe un nombre positif k_0 tel que pour tout $k \geq k_0$, les ensembles de solutions de (4.10) et (4.11) sont identiques. Une telle propriété est généralement vraie en optimisation convexe.
Cependant, pour les problèmes d'optimisation non convexe, l'étude devient beaucoup plus complexe et la technique de pénalité exacte dans ce cas vise à transformer le problème (4.10) en un problème équivalent de programmation DC.

Pour cela, on commence par donner un théorème sur l'équivalence des deux problèmes (4.10) et (4.11). Ensuite nous allons prouver que le problème (4.11) est un programme DC.

Théorème 4.1. *[77] Soit $\mathcal{Z}$ un ensemble polyédral convexe, non vide et borné, F une fonction finie concave sur $\mathcal{Z}$ et Ψ une fonction finie et concave sur $\mathcal{Z}$. Supposons que le domaine de définition de Ψ est non vide et que Ψ est non négative sur $\mathcal{Z}$. Alors il existe $k_0 \geq 0$ tel que, $\forall\ k > k_0$ les deux problèmes suivants ont les mêmes ensembles de solutions :*

$$\alpha = \min\{F(z)\ :\ z \in \mathcal{Z},\ \Psi(z) \leq 0\},$$

et

$$\alpha(k) = \min\{F(z) + k\Psi(z) : z \in \mathcal{Z}\}, \tag{4.12}$$

De plus

(i) *Si $\Psi(z) \leq 0$ pour chaque z dans l'ensemble des sommets de $\mathcal{Z}$, noté $V(\mathcal{Z})$, alors $k_0 = 0$*

(ii) *Si $\Psi(z) > 0$ pour un certain $z \in V(\mathcal{Z})$, alors $k_0 \leq \frac{F(z^0)-\alpha(0)}{S}$ pour tout $z^0 \in \mathcal{Z}$ vérifiant $\Psi(z^0) \leq 0$, où $S = \min\{\Psi(z) :\ z \in V(\mathcal{Z}),\ \Psi(z) > 0\}$.*

∎

4.3 DCA pour la résolution du problème pénalisé

Le théorème 4.1 nous permet de transformer le problème (4.10) en un programme DC. En effet, si toutes les hypothèses de ce théorème sont vérifiées, alors

(4.10) et (4.12) sont équivalents et sont des programmes DC :

$$\begin{aligned}\alpha(k) &= \min\{F(z)+k\Psi(z) : z\in\mathcal{Z}\}\\ &= \min\{\chi_{\mathcal{Z}}(z)-[-F(z)-k\Psi(z)] : z\in\mathbb{R}^n\}\\ &= \min\{g(z)-h(z) : z\in\mathbb{R}^n\},\end{aligned}$$

avec

$$h(z)=-F(z)-k\Psi(z),\quad g(z)=\chi_{\mathcal{Z}}(z) \tag{4.13}$$

La fonction h est une fonction convexe car F est linéaire, Ψ est concave sur $\mathcal{Z}$ et $k\in\mathbb{R}_+$ et g est la fonction indicatrice de l'ensemble $\mathcal{Z}$ qui est convexe, car $\mathcal{Z}$ est convexe. Elle est définie par

$$\chi_{\mathcal{Z}}(z)=\begin{cases}0, & si\quad z\in\mathcal{Z},\\ +\infty, & sinon.\end{cases}$$

En appliquant le schéma DCA (2.20), nous avons :

$$t^i\in\partial h(z^i)=\partial\Big(-F(z^i)-k\Psi(z^i)\Big). \tag{4.14}$$

et

$$z^{i+1}\in\partial g^*(t^i)=\partial\big(\chi^*_{\mathcal{Z}}(t^i)\big). \tag{4.15}$$

Calcul de t^i :

En développant (4.14), on obtient

$$t^i\in\partial\Big(-c^tz^i-k\sum_{j=1}^{n}\min\{q_j(z^i),\ z^i_j\}\Big).$$

Donc

$t^i=-c+k\theta^i$, où $\theta^i\in\partial\Big(-\sum_{j=1}^{n}\min\{q_j(z^i),\ z^i_j\}\Big)$, où $q_j(z^i)=D_jz^i$,

ce qui donne

$$\theta^i\in\partial\Big(\sum_{j=1}^{n}\max\{-D_jz^i,\ -z^i_j\}\Big).$$

Calcul explicite de θ^i :

Nous donnons d'abord le résultat suivant (voir proposition 2.2 et remarque 5).

Soit f et g deux fonctions convexes et continues, alors

$$\partial(f+g)(x)=\partial f(x)+\partial g(x).$$

La fonction $\max\{-D_j z^i,\ -z_j^i\}$ étant convexe et continue, alors, en utilisant le résultat précédent, nous obtenons :

$$\partial\Big(\sum_{j=1}^{n}\max\{-D_j z^i,\ -z_j^i\}\Big) = \sum_{j=1}^{n}\partial\Big(\max\{-D_j z^i,\ -z_j^i\}\Big)$$

D'où

$$\theta^i \in \sum_{j=1}^{n}\partial\Big(\max\{-D_j z^i,\ -z_j^i\}\Big).$$

Soit

$$\theta^i = -\sum_{j=1}^{n}\begin{cases} D_j^t, & si\ \ z_j^i > D_j z^i, \\ e_j, & si\ \ z_j^i < D_j z^i, \\ \gamma e_j + (1-\gamma)D_j^t, & si\ \ z_j^i = D_j z^i, \end{cases} \tag{4.16}$$

où D_j est la j-$ème$ ligne de D, e_j est le j-$ème$ vecteur unitaire de $\mathbb{R}^n$ et $\gamma \in [0,1]$. Donc θ^i, calculé avec (4.16), est un élément de $\sum_{j=1}^{n}\partial\big(\max\{-D_j z^i,\ -z_j^i\}\big)$.

Calcul de z^{i+1} :
Soit $z = (x, y, e, w, v, u)^t$. D'après (4.15), $z^{i+1} = (x^{i+1}, y^{i+1}, e^{i+1}, w^{i+1}, v^{i+1}, u^{i+1})$ est solution du problème linéaire

$$\min\{-\langle z,\ t^i\rangle : z \in \mathcal{Z}\}. \tag{4.17}$$

4.3.1 Algorithme DCA

Soit $z = (x, y, e, w, v, u)^t$ et $t = (t_x, t_y, t_e, t_w, t_v, t_u)^t$ des vecteurs de $\mathbb{R}^n$. En se basant sur ce qui précède, nous proposons l'algorithme DCA suivant :

Algorithme DCA

1 : Soit z^0 point initial, poser $i = 0$, $\epsilon > 0$, $k \in \mathbb{R}_+$, $\gamma \in [0,1]$ et $\lambda > 0$.
2 : Calculer $t^i = (t^i_x, t^i_y, t^i_e, t^i_w, t^i_v, t^i_u) \in \partial h(z^i)$ i.e.
$t^i = -c + k\theta^i$, où θ^i est calculée en utilisant (4.16)
3 : Calculer $z^{i+1} = (x^{i+1}, y^{i+1}, e^{i+1}, w^{i+1}, v^{i+1}, u^{i+1})$ en utilisant (4.17).
4 : Si $y^{i+1} \in \arg\max\{f(x^{i+1}, y) : B_2 y \leq b_2 - A_2 x^{i+1},\ y \geq 0\}$, **alors** aller à **5**, **sinon** aller à **7**
5 : Calculer (v^*, u^*) solution du problème
$\min\{u^t(b_2 - A_2 x^{i+1}) : B_2^t u - v = d_2,\ u \geq 0,\ v \geq 0\}$, donc
$z^{i+1} = (x^{i+1}, y^{i+1}, e^{i+1}, w^{i+1}, v^*, u^*)$.
6 : Si $\|z^{i+1} - z^i\|/(\|z^i\| + 1) \leq \epsilon$ **alors**
arrêter, z^{i+1} est solution optimale de (4.1), **sinon** aller à **7**
7 : Poser $z^i = z^{i+1}$, $i = i + 1$, $k = k + \lambda$ et aller à **2**.

Commentaires sur DCA

i) Le problème (4.12), avec la décomposition (4.13), est un programme DC polyédral, puisque $g(z) = \chi_{\mathcal{Z}}(z)$ est polyédrale [68]. Donc DCA appliqué à (4.12) possède une convergence finie [80, 83].

ii) Le test de l'étape 4 a été rajouté pour tester la faisabilité d'une solution (x, y) pour le (PBL) . La vérification de ce test implique que $y \in R(x)$ (voir définition 1.1). Et puisque $(x, y) \in S$, on aura alors $(x, y) \in RI$ ce qui implique que (x, y) est une solution réalisable pour (PBL) .

iii) L'algorithme DCA calcule la solution optimale $z^* = (x^*, y^*, e^*, w^*, v^*, u^*)$ du problème (4.12). Si on définit l'ensemble Z_r comme

$$Z_r = \{z = (x, y, w, e, v, u) \in \mathbb{R}^n : z \in \mathcal{Z}, (x, y) \in RI\},$$

alors, d'après *ii*), $z^* \in Z_r$. Une telle solution n'est pas forcément une solution du problème (4.1) à cause de la présence des composantes duales (v^*, u^*). Le résultat suivant est inspiré de [44].

Théorème 4.2. *Soit $k \in \mathbb{R}_+$ et soit l'ensemble $\bar{Z} \subset \mathcal{Z}$ défini par*

$$\bar{Z} = \{z \in \mathbb{R}^n : z \in Z_r,\ \exists I \subset \{1, \ldots, n\}/z_i = 0, \forall i \in I; q_i(z) = 0, \forall i \notin I\},$$

Une solution optimale z^* du problème (4.12) est solution de (4.1), si $z^* \in \bar{Z}$.

■

Preuve Soit $k \in \mathbb{R}_+$ et z^* solution optimale de (4.12). Alors, on a

$$g(z^*) - h(z^*) \leq g(z) - h(z),\ \forall z \in \mathcal{Z}.$$

Puisque $g = \chi_{\mathcal{Z}}$ et $z, z^* \in \mathcal{Z}$, nous avons

$$0 - h(z^*) \leq 0 - h(z),\ \forall z \in \mathcal{Z},$$

ce qui donne

$$F(z^*) + k\Psi(z^*) \leq F(z) + k\Psi(z),\ \forall z \in \mathcal{Z}. \tag{4.18}$$

Puisque $\bar{Z} \subset \mathcal{Z}$, nous avons donc

$$c^t z^* + k\Psi(z^*) \leq c^t z + k\Psi(z),\ \forall z \in \bar{Z}. \tag{4.19}$$

Si $z^* \in \bar{Z}$, alors on aura

$$F(z^*) \leq F(z),\ \forall z \in \bar{Z},$$

Donc

$$-F(z^*) \geq -F(z),\ \forall z \in \bar{Z},$$

ce qui donne

$$c_1^t x^* + d_1^t y^* \geq c_1^t x + d_1^t y,\ \forall (x, y) \in RI.$$

De la définition 1.2, (x^*, y^*) est solution optimale du (PBL).

■

L'étape 5 de DCA nous permet de calculer un point $z \in \bar{Z}$. En effet, la fonction de pénalité $\Psi(z)$ étant nulle en ce point (car il est optimal pour le (PBL)), ceci implique que la contrainte de complémentarité du système KKT associé au problème (P) du Suiveur est vérifiée et les variables duales (v, u) sont donc solutions du problème dual de (P).

4.3.2 Calcul du point initial pour DCA

Le choix d'un bon point de départ pour DCA est une tâche très importante. Pour cela, on utilise DCA pour la résolution du problème de programmation concave suivant [84] :

$$P_\Psi \begin{cases} \min \Psi(z) = 0 \\ \bar{A}z = \bar{b}, \\ z \geq 0. \end{cases} \tag{4.20}$$

avec

$$\bar{A} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ A_2 & B_2 & 0 & I_{m_2} & 0 & 0 \\ 0 & 0 & 0 & 0 & -I_{n_2} & B_2^t \end{pmatrix} \quad \text{et} \quad \bar{b} = \begin{pmatrix} 0 \\ b_2 \\ d_2 \end{pmatrix}$$

Le problème P_Ψ est un programme DC polyédral dont la valeur optimale est connue. Il peut être résolu comme un problème de complémentarité linéaire [3]. DCA, avec un point de départ $\bar{z}$ tel que $\Psi(\bar{z}) = 0$, converge, dans la plupart des cas, vers la solution globale du problème P_Ψ.

4.3.3 Résultats numériques

L'algorithme DCA présenté dans cette section est codé en MATLAB. Nous l'avons testé sur un ensemble de problèmes (voir ci-dessous) dont la solution est connue au préalable. Le point initial z^0 est calculé en résolvant le problème (4.20). Nous avons remarqué que la valeur du paramètre α n'a pas d'influence sur les résultats de l'algorithme donc nous avons pris comme valeur $\gamma = 0.5$. Le test d'arrêt de l'algorithme sera vérifié s'il est inférieur ou égal à $\epsilon = 10^{-3}$. Le temps est reporté en secondes.

Problèmes test

P1 [37] :
$$\begin{cases} \max F(x,y) = 8x_1 + 4x_2 - 4y_1 + 40y_2 + 4y_3 \\ x \geq 0 \\ \max f(x,y) = -x_1 - 2x_2 - y_1 - y_2 - 2y_3 \\ 0x_1 + 0x_2 - y_1 + y_2 + y_3 \leq 1 \\ 2x_1 + 0x_2 - y_1 + 2y_2 - 0.5y_3 \leq 1 \\ 0x_1 + 2x_2 + 2y_1 - y_2 - 0.5y_3 \leq 1 \\ y \geq 0 \end{cases}$$

P2 [23] :
$$\begin{cases} \max F(x,y) = x + 3y \\ x \geq 0 \\ \max f(x,y) = x - 3y \\ -x - 2y \leq -10 \\ x - 2y \leq 6 \\ 2x - y \leq 21 \\ x + 2y \leq 38 \\ -x + 2y \leq 18 \\ y \geq 0 \end{cases}$$

P3 [53] :
$$\begin{cases} \max F(x,y) = 8x_1 + 4x_2 - 4y_1 + 40y_2 + 4y_3 \\ x_1 + 2x_2 - y_3 \leq 1.3 \\ x \geq 0 \\ \max f(x,y) = -2y_1 - y_2 - 2y_3 \\ 0x_1 + 0x_2 - y_1 + y_2 + y_3 \leq 1 \\ 4x_1 + 0x_2 - 2y_1 + 4y_2 - y_3 \leq 2 \\ 0x_1 + 4x_2 + 4y_1 - 2y_2 - y_3 \leq 2 \\ y \geq 0 \end{cases}$$

P4 [90] :
$$\begin{cases} \max F(x,y) = x + y \\ x \geq 0 \\ \max f(x,y) = 0x - y \\ -4x - 3y \leq -19 \\ x + 2y \leq 11 \\ 3x + y \leq 13 \\ y \geq 0 \end{cases}$$

P5 [64] :
$$\begin{cases} \max F(x,y) = -x - 5y \\ x \geq 0 \\ \max f(x,y) = 0x + y \\ -x - y \leq -8 \\ -3x + 2y \leq 6 \\ x + 4y \leq 48 \\ x - 5y \leq 9 \\ y \geq 0 \end{cases}$$

P6 [63] :
$$\begin{cases} \max F(x,y) = x_1 + y_1 - 4y_2 \\ x \geq 0 \\ \max f(x,y) = 0x_1 + 0y_1 + y_2 \\ x + y_1 + y_2 \leq 3 \\ -x - y_1 + y_2 \leq -1 \\ -x + y_1 + y_2 \leq 1 \\ x - y_1 + y_2 \leq 1 \\ y \geq 0 \end{cases}$$

P7 [63] :
$$\begin{cases} \max F(x,y) = -0.4x_1 - y_1 - 5y_2 + 0y_3 + 0y_4 \\ x \geq 0 \\ \max f(x,y) = 0x_1 + 0y_1 + 0.5y_2 - y_3 - 2y_4 \\ -0.1x_1 - y_1 - y_2 + 0y_3 + 0y_4 \leq -1 \\ 0.2x_1 + 0y_1 + 1.25y_2 + 0y_3 - y_4 \leq -1 \\ -x + 6y_1 + y_2 - 2y_3 + 0y_4 \leq 1 \\ y \geq 0 \end{cases}$$

P8 [24] :
$$\begin{cases} \max F(x,y) = -x + 4y \\ x \geq 0 \\ \max f(x,y) = 0x - y \\ -x - y \leq -3 \\ -2x + y \leq 0 \\ 2x + y \leq 12 \\ -3x + 2y \geq -4 \\ y \geq 0 \end{cases}$$

P9 [37] :
$$\begin{cases} \max F(x,y) = 2x_1 - x_2 - 0.5y_1 \\ x_1 + x_2 \leq 2 \\ x \geq 0 \\ \max f(x,y) = 0x_1 + 0x_2 + 4y_1 - y_2 \\ -2x_1 + y_1 - y_2 \leq -2.5 \\ x_1 - 3x_2 + y_2 \leq 2 \\ y \geq 0 \end{cases}$$

P10 [37] :
$$\begin{cases} \max F(x,y) = -x + 4y \\ x \geq 0 \\ \max f(x,y) = 0x - y \\ -2x + y \leq 0 \\ 2x + 5y \leq 108 \\ 2x - 3y \leq -4 \\ y \geq 0 \end{cases}$$

P11 [28] :
$$\begin{cases} \max F(x,y) = x + 2y \\ x \geq 0 \\ \max f(x,y) = 0x - y \\ x - y \leq 0 \\ -x - y \leq 2 \\ y \leq 4 \\ y \geq 0 \end{cases}$$

P12 [52] :
$$\begin{cases} \max F(x,y) = -x - 5y \\ x \geq 0 \\ \max f(x,y) = 0x + y \\ -x - y \leq -8 \\ -3x + 2y \leq 6 \\ 3x + 4y \leq 48 \\ 2x - 5y \leq 9 \\ y \geq 0 \end{cases}$$

P13 [85] :
$$\begin{cases} \max F(x,y) = -x_1 - 2x_2 - 2y_1 + y_2 \\ x_1 + x_2 + 0.5y_1 + y_2 \leq 6 \\ x \geq 0 \\ \max f(x,y) = -y_1 + 2y_2 \\ -x_1 + 2x_2 + 0y_1 + y_2 \leq 4 \\ -x_1 - x_2 + y_1 + y_2 \leq 5 \\ y \geq 0 \end{cases}$$

$$\text{P14 [71]} : \begin{cases} \max F(x,y) = 2x_1 - x_2 - x_3 + 2x_4 + x_5 - 3.5x_6 - y_1 - 1.5y_2 + 3y_3 \\ x \geq 0 \\ \max f(x,y) = 0x_1 + 2x_2 + 0x_3 + 0x_4 - x_5 + 0x_6 + 3y_1 - y_2 - 4y_3 \\ -x_1 + 0.2x_2 + 0x_3 + 0x_4 + x_5 + 2x_6 - 4y_1 + 2y_2 + y_3 \leq 12 \\ x_1 + 0x_2 + x_3 - 2x_4 + 0x_5 + 0x_6 + 0y_1 - 4y_2 + y_3 \leq 10 \\ 5x_1 + 0x_2 + 0x_3 + x_4 + 0x_5 + 3.2x_6 + 2y_1 + 2y_2 + 0y_3 \leq 15 \\ 0x_1 - 3x_2 + 0x_3 - x_4 + x_5 + 0x_6 - 2y_1 + 0y_2 + 0y_3 \leq 12 \\ -2x_1 - x_2 + 0x_3 + 0x_4 + 0x_5 + 0x_6 + 0y_1 - y_2 + y_3 \leq -2 \\ 0x_1 + 0x_2 + 0x_3 + 0x_4 + 0x_5 + 0x_6 - y_1 - 2y_2 - y_3 \leq -2 \\ 0x_1 - 2x_2 - 3x_3 + 0x_4 - x_5 + 0x_6 + 0y_1 + 0y_2 + 0y_3 \leq -3 \\ y \geq 0 \end{cases}$$

Les résultats numériques sont donnés dans le tableau 1.1. Nous avons les notations suivantes :

k : le paramètre de pénalité.
λ : un pas pour augmenter le paramètre de pénalité.
temps : est le temps d'exécution de l'algorithme.
Iter : le nombre d'itérations de l'algorithme.
$(x^*; y^*)$: représentent la solution optimale du problème.
$(F^*; f^*)$: représentent les valeurs optimales du Leader et Suiveur respectivement.
"−" signifie que l'algorithme n'a pas pu résoudre le problème.

Problème	*(k ; λ)*	$(x^*; y^*)$	$(F^*; f^*)$	*temps*	*Iter*
P1	(1 ;0.1)	-	-	-	-
	(1 ;0.5)	(0 0.9 0 0.6 0.4)	(29.2 ; -3.2)	4.06	19
	(1 ;1)	(0 0.9 0 0.6 0.4)	(29.2 ; -3.2)	2.48	11
	(5 ;1)	(0 0.9 0 0.6 0.4)	(29.2 ; -3.2)	2.07	7
	(5 ;5)	(0 0.9 0 0.6 0.4)	(29.2 ; -3.2)	1.26	3
	(10 ;5)	(0 0.75 0 0.5 0)	(23 ; -2)	0.7	2
P2	(0.01 ;0.01)	-	-	-	-
	(0.1 ;0.01)	(16 11)	(49 ; -17)	3.14	3
	(0.1 ;0.1)	(16 11)	(49 ; -17)	6.92	6
	(1 ;1)	(16 11)	(49 ; -17)	0.65	2
	(5 ;1)	(12 3)	(21 ; 3)	0.17	2
	(5 ;5)	(12 3)	(21 ; 3)	1.23	2
	(10 ;5)	(12 3)	(21 ; 3)	0.18	2
P3	(5 ;5)	-	-	-	-
	(10 ;5)	(0 0.78 0 0.43 0.26)	(21.36 ; -0.95)	1.18	2
	(15 ;5)	(0 0.78 0 0.43 0.26)	(21.36 ; -0.95)	1.44	2
	(20 ;5)	(0 0.78 0 0.43 0.26)	(21.36 ; -0.95)	1.20	2
P4	(0.1 ;0.1)	-	-	-	-
	(1 ;0.1)	(4 1)	(5 ; -1)	0.09	2
	(1 ;1)	(4 1)	(5 ; -1)	1.09	4
	(5 ;1)	(1 5)	(6 ; -5)	0.7	2
	(5 ;5)	(1 5)	(6 ; -5)	0.6	2
	(10 ;5)	(1 5)	(6 ; -5)	0.6	2
P5	(0.1 ;0.1)	-	-	-	-
	(1 ;0.1)	(2 6)	(-32 ; 6)	1.5	3
	(1 ;1)	(2 6)	(-32 ; 6)	1.7	3
	(5 ;1)	(2 6)	(-32 ; 6)	0.5	2
	(5 ;5)	(2 6)	(-32 ; 6)	0.6	2
	(10 ;5)	(2 6)	(-32 ; 6)	0.56	2
P6	(0.01 ;0.01)	(2 1 0)	(3 ; 0)	0.82	3
	(0.1 ;0.1)	(2 1 0)	(3 ; 0)	0.62	3
	(1 ;0.1)	(1 1 1)	(-2 ; 1)	0.56	1
	(1 ;1)	(1 1 1)	(-2 ; 1)	0.6	1
	(5 ;1)	(1 1 1)	(-2 ; 1)	0.58	1
	(10 ;5)	(1 1 1)	(-2 ; 1)	0.75	1
P7	(0.1 ;0.01)	-	-	-	-
	(0.1 ;0.1)	(0 0 1 0 2.25)	(-5 ; -4)	0.68	2
	(1 ;1)	(0 0 1 0 2.25)	(-5 ; -4)	0.6	2
	(5 ;1)	(0 0 1 0 2.25)	(-5 ; -4)	0.56	2
	(5 ;5)	(0 0 1 0 2.25)	(-5 ; -4)	1.06	2
	(10 ;5)	(0 0 1 0 2.25)	(-5 ; -4)	0.96	2
P8	(0.1 ;0.1)	-	-	-	-
	(1 ;0.1)	(4 4)	(12 ; -4)	3.96	8
	(1 ;1)	(4 4)	(12 ; -4)	1.51	3
	(5 ;1)	(2 1)	(2 ; -1)	0.56	2
	(5 ;5)	(2 1)	(2 ; -1)	0.21	2
	(10 ;5)	(2 1)	(2 ; -1)	0.65	2

Problème	*(k ; λ)*	*(x*; y*)*	*(F*; f*)*	*temps*	*Iter*
P9	(0.1 ;0.1)	-	-	-	-
	(1 ;0.1)	(2 0 1.5 0)	(3.25 ; 6)	2.12	2
	(1 ;1)	(2 0 1.5 0)	(3.25 ; 6)	1.40	4
	(5 ;1)	(2 0 1.5 0)	(3.25 ; 6)	1.42	2
	(5 ;5)	(2 0 1.5 0)	(3.25 ; 6)	1.39	2
	(10 ;5)	(2 0 1.5 0)	(3.25 ; 6)	1.43	2
P10	(0.1 ;0.1)	-	-	-	
	(1 ;0.1)	(19 14)	(37 ; -14)	0.64	2
	(1 ;1)	(19 14)	(37 ; -14)	0.17	2
	(5 ;1)	(19 14)	(37 ; -14)	1.28	6
	(5 ;5)	(19 14)	(37 ; -14)	0.5	2
P11	(0.01 ;0.01)	(4 4)	(12 ; -4)	3.78	10
	(0.1 ;0.01)	(4 4)	(12 ; -4)	3.29	3
	(0.1 ;0.1)	(4 4)	(12 ; -4)	5.84	3
	(1 ;1)	(4 4)	(12 ; -4)	0.82	2
	(5 ;1)	(0 0)	(0 0)	0.75	2
	(10 ;5)	(0 0)	(0 0)	0.68	2
P12	(0.1 ;0.1)	-	-	-	-
	(0.5 ;0.1)	(12 3)	(-27 ; 3)	6.31	4
	(1 ;0.1)	(12 3)	(-27 ; 3)	1.46	2
	(1 ;1)	(12 3)	(-27 ; 3)	1.14	2
	(5 ;1)	(12 3)	(-27 ; 3)	1.01	2
	(10 ;5)	(12 3)	(-27 ; 3)	0.96	2
P13	(0.01 ;0.01)	(4,10)	(4 ; 10)	1.21	2
	(0.1 ;0.01)	(4,10)	(4 ; 10)	1.15	2
	(1 ;0.1)	(4,10)	(4 ; 10)	1.01	2
	(5 ;1)	(4,10)	(4 ; 10)	1.3	2
	(5 ;5)	(4,10)	(4 ; 10)	0.96	2
	(10 ;5)	(4,10)	(4 ; 10)	0.5	2
P14	(0.1 ;0.1)	-	-	-	-
	(0.5 ;0.1)	(0 4 0 15 9.2 0 0 0 2)	(41.2 ; -9.2)	1.17	2
	(0.5 ;0.5)	(0 4 0 15 9.2 0 0 0 2)	(41.2 ; -9.2)	2.5	7
	(1 ;0.5)	(0 2 0 11 19.6 0 2 0 0)	(37.6 ; -13.6)	0.7	2
	(1 ;1)	(0 2 0 11 19.6 0 2 0 0)	(37.6 ; -13.6)	0.7	2
	(5 ;1)	(1 0 0 6 21 0 2 0 0)	(33 ; -19)	2.64	6
	(5 ;5)	(1 0 0 6 21 0 2 0 0)	(33 ; -19)	1.2	2
	(10 ;5)	(0.5 0 0 0 3.93 3.92 0 1 0)	(-8.04 ; -4.93)	0.6	2

TAB. 1.1 : Résultats numériques pour l'algorithme DCA

Commentaires sur les résultats de DCA

D'après les résultats numériques du tableau 1.1, on peut voir que :

– L'algorithme DCA possède une convergence finie et rapide ; le nombre moyen d'itérations est de 2.

– Le temps d'exécution de l'algorithme est petit. Ce résultat est normal car seuls des programmes linéaires sont résolus à chaque itération de l'algorithme.

– La procédure de calcul du point initial pour DCA était efficace : dans la plupart des problèmes testés, l'algorithme calcule la solution globale. Cependant, comme il peut être remarqué cette convergence vers la solution globale est également influencée par le choix du paramètre de pénalité k ainsi que le pas d'augmentation de ce paramètre, λ. Pour le même problème, l'algorithme peut converger vers une solution globale pour certaines valeurs de k et λ et converger vers la solution locale pour d'autres valeurs. Par exemple, pour le problème P1, l'algorithme a donné la solution $F^* = 29.2$ pour $(k, \lambda) = (5; 1)$ et $F^* = 23$ pour $(k, \lambda) = (10; 5)$.

– Le choix du paramètre de pénalité k dépend du problème testé.

La section qui suit sera consacrée à la présentation d'un autre algorithme (PBLP) de résolution des problèmes de programmation bi-niveaux linéaires développé dans [63] et conçu particulièrement pour les (PBL) sans contraintes du Leader. Dans la dernière partie de ce chapitre, une comparaison sera présentée.

4.4 Résolution du (PBL) par les fonctions pénalité

Soit le problème de programmation bi-niveaux linéaire de la forme

$$\begin{cases} \max\limits_{x\geq 0} \ c_1^t x + d_1^t y, \\ \qquad \max\limits_{y\geq 0} \ c_2^t x + d_2^t y, \\ \qquad\qquad s.c. \ \ A_2 x + B_2 y \leq b_2, \end{cases} \tag{4.21}$$

où c_1, c_2, d_1, d_2, b_2, A_2etB_2 sont définis comme pour le problème (4.1).

Le principe de l'algorithme présenté dans [63] est de transformer le problème (4.21) en un problème avec une seule fonction objectif en utilisant les conditions d'optimalité de KKT. On obtient le problème

$$\begin{cases} \max\limits_{x\geq 0} \ c_1^t x + d_1^t y, \\ \qquad A_2 x + B_2 y + w = b_2, \\ \qquad B_2^t u - v = d_2, \\ \qquad u^t w + v^t y = 0, \\ \qquad x, \ y, \ u, \ v, \ w \geq 0, \end{cases} \tag{4.22}$$

où $w \in \mathbb{R}^{m_2}$ est une variable d'écart et $u \in \mathbb{R}^{m_2}$, $v \in \mathbb{R}^{n_2}$ sont des variables duales.

Les contraintes de complémentarité seront ensuite ajoutées à la fonction objectif du Leader avec une pénalité. Finalement, le problème sera décomposé en une série de problèmes de programmation linéaire qui seront résolus par la méthode de programmation linéaire pour obtenir la solution du programme bi-niveaux linéaire.

4.4.1 Résultats théoriques

Pour la construction de l'algorithme l'hypothèse suivante est requise

H 3. L'ensemble $S^{'} = \{(x, y) : A_2x + B_2y \leq b_2, x, y \geq 0\}$.

En ajoutant les contraintes de complémentarité au niveau supérieur, on obtient le problème pénalisé

$$P(k) \quad \begin{cases} \max\limits_{x \geq 0} \; c_1^t x + d_1^t y - k(u^t w + v^t y), \\ \qquad A_2x + B_2y + w = b_2, \\ \qquad B_2^t u - v = d_2, \\ \qquad x,\ y,\ u,\ v,\ w \geq 0, \end{cases} \tag{4.23}$$

où $k \in \mathbb{R}_+$ est une constante positive.

Les notations suivantes ont été introduites.

$$A = \begin{pmatrix} A_2 & B_2 & 0 & I_{m_2} \end{pmatrix} \in \mathbb{R}^{m_2 \times n}, \quad B = \begin{pmatrix} 0 & -I_{n2} & B_2^t \end{pmatrix} \in \mathbb{R}^{n_2 \times n},$$
$$c = \begin{pmatrix} c_1 & d_1 & 0 \end{pmatrix}^t \in \mathbb{R}^n, \quad z = \begin{pmatrix} x & y & w \end{pmatrix}^t \in \mathbb{R}^n, \quad s = \begin{pmatrix} 0 & v & u \end{pmatrix}^t \in \mathbb{R}^n,$$

où $n = n_1 + n_2 + m_2$, I_k est la matrice identité d'ordre k et 0 est la matrice nulle avec la dimension appropriée pour chaque cas.

On note aussi les polyèdres $Z = \{z \in \mathbb{R}^n_+ : Az = b_2\}$, $S = \{s \in \mathbb{R}^n_+ : Bs = d_2\}$ et $X(V)$ l'ensemble des sommets du polyèdre X.
Le problème $P(k)$ peut être écrit sous la forme

$$P(k) \quad \begin{cases} \max F_k(z, s) = c^t z - k s^t z \\ \quad z \in Z, \quad s \in S, \\ \quad s^t z = 0. \end{cases} \tag{4.24}$$

Dans la suite, nous reprenons les principaux résultats concernant l'algorithme PBLP. Les démonstrations des théorèmes peuvent être trouvées dans [63].

Pour $s \in S$ et $k \in \mathbb{R}_+$, définissons la fonction $Q_k(s)$ par

$$Q_k(s) = \max_{z \in Z} F_k(z, s) = c^t z - k s^t z.$$

Nous avons donc les résultats suivants

Théorème 4.3. *[63]*
Soit $k \in \mathbb{R}_+$ et supposons que l'hypothèse H3 est vérifiée. Alors une solution du problème

$$\max_{s \in S} Q_k(s)$$

sera atteinte en un point $s^* \in S(V)$.

■

Pour la solution du problème $P(k)$, nous avons le théorème suivant

Théorème 4.4. *[63]*
Soit $k \in \mathbb{R}_+$ et supposons que H3 est vérifiée. Alors le problème $P(k)$ admet une solution optimale appartenant à $Z(V) \times S(V)$.

■

Théorème 4.5. *[63]*
Supposons H 3 est vérifiée. Soit $\{(z_k, s_k)\}$ une suite de solutions des problèmes $P(k)$, alors il existe $k_1 \in \mathbb{R}_+$ tel que

$$\forall k \geq k_1, \quad s_k^t z_k = 0.$$

■

Le théorème suivant montre que la pénalité est exacte.

Théorème 4.6. *[63]*
Supposons que H 3 est vérifiée. Soit $\{(z_k, s_k)\}$ une séquence de solutions du problème $P(k)$, $k \in \mathbb{R}_+$. Alors il existe $k^* \in \mathbb{R}_+$, tel que pour tout $k \geq k^*$, z_k résout le problème (4.21).

■

Théorème 4.7. *[63]*
Supposons que H 3 est vérifiée. Soient $k \in \mathbb{R}_+$, s, $s' \in S$ et $z_k(s')$ une solution du problème

$$\max_{z \in Z} F_k(z, s') = c^t z - k(s')^t z.$$

Alors

$$Q_k(s) \geq Q_k(s') - k(s - s')^t z_k(s'). \tag{4.25}$$

■

Remarque 7. Posons

$$\alpha_k(s') = \min_{s\in S}(s - s')^t z_k(s'),$$

où $z_k(s')$ est solution de $Q_k(s')$.
D'après le théorème 4.7, si $\alpha_k(s') < 0$, alors

$$s' \notin \arg\max\{Q_k(s) : s \in S\}.$$

4.4.2 Algorithme PBLP

0 : Choisir $k > 0$ (k grand), $s_k^0 \in S$ et $\lambda > 0$, $i = 0$
1 : Résoudre $\max_{z\in Z}\{c^t z - k(s_k^i)^t z\}$, obtenir une solution $z(s_k^i)$
2 : Résoudre $\min_{s\in S}(s - s_k^i)^t z_k(s_k^i)$, obtenir une solution $s_{k,i}^*$
et une valeur optimale $\alpha_k(s_k^i) = \min_{s\in S}(s - s_k^i)^t z_k(s_k^i)$.
3 :
(1) Si $\alpha_k(s_k^i) < 0$, poser $s_k^{i+1} = s_{k,i}^*$, $i = i + 1$, aller à l'étape 1.
(2) Si $\alpha_k(s_k^i) \geq 0$ et $(s_k^i)^t z_k(s_k^i) > 0$, alors $k = k + \lambda$, $i = i + 1$, aller à l'étape 1.
(3) Si $\alpha_k(s_k^i) \geq 0$ et $(s_k^i)^t z_k(s_k^i) = 0$ alors la solution optimale du problème (4.21) est $z_k(s_k^i)$.

4.4.3 Résultats numériques

Nous avons codé l'algorithme PBLP en MATLAB. Pour le tester, nous avons pris dans l'ensemble des problèmes test donné plus haut, les problèmes sans contraintes du Leader. Le point initial s^0 est pris dans l'ensemble S défini dans la section 4.4.1. Le test d'arrêt de l'algorithme sera vérifié s'il est inférieur ou égal à $\epsilon = 10^{-3}$. Le temps est reporté en secondes. Les résultats numériques sont donnés dans le tableau 1.2 où on trouve les notations suivantes :

k : le paramètre de pénalité.
λ : un pas pour augmenter le paramètre de pénalité.
temps : est le temps d'exécution de l'algorithme.
Iter : le nombre d'itérations de l'algorithme.
(x^*, y^*) : représentent la solution optimale du problème.
$(F^*; f^*)$: représentent les valeurs optimales du Leader et Suiveur respectivement.
"−" signifie que l'algorithme n'a pas pu résoudre le problème.

Problème	s^0	*(k ; λ)*	*temps*	*Iter*	(x^*, y^*)	$(F^*; f^*)$
P1	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 2 \\ 3 \\ 1 \\ 0 \\ 0 \end{pmatrix}$	-	-	-	-	-
	$\begin{pmatrix} 0 \\ 0 \\ 6 \\ 0 \\ 0 \\ 0 \\ 1 \\ 3 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	12.5 1.46 1.07 0.58 0.03	24 4 1 1 1	$\begin{pmatrix} 0 \\ 0.9 \\ 0 \\ 0.6 \\ 0.4 \end{pmatrix}$	(29 ; -3.2)
	$\begin{pmatrix} 0 \\ 0 \\ 3 \\ 0 \\ 1.5 \\ 0 \\ 0 \\ 1 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	12.5 1.46 1.07 0.58 0.03	95 11 7 2 1	$\begin{pmatrix} 0 \\ 0.75 \\ 0 \\ 0.5 \\ 0 \end{pmatrix}$	(23 ; -2)
P2	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 3 \\ 0 \\ 0 \end{pmatrix}$	-	-	-	-	-
	$\begin{pmatrix} 0 \\ 0 \\ 1.5 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	0.6 0.57 0.82 0.56 0.03	2 1 1 1 1	$\begin{pmatrix} 0 \\ 5 \end{pmatrix}$	(15 ; -15)

Problème	s^0	*(k ; λ)*	*temps*	*Iter*	(x^*, y^*)	$(F^*; f^*)$
P2	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 3 \\ 0 \\ 0 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	0.54 0.12 0.03 0.65 0.26	1 1 1 1 1	$\begin{pmatrix} 16 \\ 11 \end{pmatrix}$	(49 ; -17)
P4	$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	-	-	-	-	-
	$\begin{pmatrix} 0 \\ 0 \\ 0.33 \\ 0 \\ 0 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	0.51 0.65 0.09 0.64 0.76	1 1 1 1 1	$\begin{pmatrix} 1 \\ 5 \end{pmatrix}$	(6 ; -5)
P5	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.25 \\ 0 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	0.6 0.57 0.82 0.56 0.03	36 5 1 1 1	$\begin{pmatrix} 30.66 \\ 4.33 \end{pmatrix}$	(-52.66 ; 4.33)
	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0.5 \\ 0 \\ 0 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	1.54 0.59 0.8 0.6 0.75	8 2 1 1 1	$\begin{pmatrix} 2 \\ 6 \end{pmatrix}$	(-32 ; 6)

Problème	s^0	*(k ; λ)*	*temps*	*Iter*	(x^*, y^*)	$(F^*; f^*)$
P6	$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$	-	-	-	-	-
	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.5 \\ 0.5 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	1.3 0.17 0.15 0.13 0.06	42 6 2 1 1	$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$	(-2 ; 1)
	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0.5 \\ 0 \\ 0 \\ 0.5 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	0.06 0.08 0.17 0.2 0.2	1 1 1 1 1	$\begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}$	(3 ; 0)
P7	$\begin{pmatrix} 0 \\ 3 \\ 0 \\ 0 \\ 2 \\ 0 \\ 0 \\ 0.5 \end{pmatrix}$	-	-	-	-	-
	$\begin{pmatrix} 0 \\ 0 \\ 2 \\ 1 \\ 0 \\ 0 \\ 2 \\ 0 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	0.07 0.05 0.06 0.07 0.09	1 1 1 1 1	$\begin{pmatrix} 10 \\ 0 \\ 0 \\ 0 \\ 3 \end{pmatrix}$	(-4 ; -6)

Problème	s^0	*(k ; λ)*	*temps*	*Iter*	(x^*, y^*)	$(F^*; f^*)$
P7	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0.2 \\ 0 \\ 2.4 \\ 2 \\ 0.4 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	0.67 0.05 0.12 0.07 0.06	1 1 1 1 1	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 2.25 \end{pmatrix}$	(-5 ; -4)
P8	$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	-	-	-	-	-
	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	0.67 0.33 0.17 0.08 0.09	15 3 1 1 1	$\begin{pmatrix} 1 \\ 2 \end{pmatrix}$	(7 ; -2)
	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.5 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	0.75 0.16 0.09 0.15 0.17	17 3 1 1 1	$\begin{pmatrix} 4 \\ 4 \end{pmatrix}$	(12 ; -4)
P10	$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	-	-	-	-	-
	$\begin{pmatrix} 0 \\ 0 \\ 0.33 \\ 0 \\ 0 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	0.49 0.11 0.08 0.12 0.14	16 3 1 1 1	$\begin{pmatrix} 19 \\ 14 \end{pmatrix}$	(37 ; -14)

Problème	s^0	$(k\,;\,\lambda)$	*temps*	*Iter*	(x^*, y^*)	$(F^*; f^*)$
P11	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$	-	-	-	-	-
	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	0.21 0.18 0.04 0.03 0.05	1 1 1 1 1	$\begin{pmatrix} 4 \\ 4 \end{pmatrix}$	(12 ; -4)
	$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	1.66 0.32 0.04 0.07 0.06	22 4 1 1 1	$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	(0 ; 0)
P12	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.25 \\ 0 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	0.56 0.22 0.16 0.24 0.16	18 3 1 1 1	$\begin{pmatrix} 12 \\ 3 \end{pmatrix}$	(-27 ; 3)
	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0.5 \\ 0 \\ 0 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	0.31 0.25 0.06 0.05 0.05	8 2 1 1 1	$\begin{pmatrix} 2 \\ 6 \end{pmatrix}$	(-32 ; 6)

Problème	s^0	$(k\,;\,\lambda)$	*temps*	*Iter*	(x^*, y^*)	$(F^*; f^*)$
P14	$\begin{pmatrix} 0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\3.5\\1\\0\\4\\0 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	0.24 0.07 0.05 0.06 0.13	3 2 1 1 1	$\begin{pmatrix} 1.16\\0\\0\\5.16\\21.6\\0\\2\\0\\0 \end{pmatrix}$	(31.83 ; -19.16)
	$\begin{pmatrix} 0\\0\\0\\0\\0\\0\\0\\0\\0\\0.4\\0\\3.5\\0\\0\\4.4\\0 \end{pmatrix}$	$(1;0.1)$ $(1;1)$ $(5;1)$ $(10;5)$ $(20;5)$	0.08 0.03 0.19 0.06 0.04	1 1 1 1 1	$\begin{pmatrix} 0\\4\\0\\15\\9.2\\0\\0\\0\\2 \end{pmatrix}$	(41.2 ; -9.2)

Problème	s^0	*(k ; λ)*	*temps*	*Iter*	(x^*, y^*)	$(F^*; f^*)$
P14	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 0 \\ 0 \\ 1.5 \\ 0 \\ 0 \\ 2 \\ 0 \end{pmatrix}$	(1; 0.1) (1; 1) (5; 1) (10; 5) (20; 5)	0.04 0.04 0.14 0.04 0.11	1 1 1 1 1	$\begin{pmatrix} 0 \\ 2 \\ 0 \\ 11 \\ 19.6 \\ 0 \\ 2 \\ 0 \\ 0 \end{pmatrix}$	(37.6 ; -13.6)

Tab. 1.2 : Résultats numériques pour l'algorithme PBLP

Commentaires et comparaison

– Les résultats numériques du tableau 1.2 montrent que l'algorithme PBLP converge plus rapidement pour les grandes valeurs du paramètre de pénalité k et le pas d'augmentation λ comme cela était le cas pour DCA, mais sa convergence est plus lente pour les petites valeurs de k et λ.

– Comme on pouvait s'y attendre, le temps d'exécution de PBLP est petit, car, comme pour DCA, on n'a que des programmes linéaires à résoudre à chaque itération.

– L'algorithme PBLP est très sensible au choix de la solution initiale s^0. A chaque fois qu'on prend une solution de départ, l'algorithme converge vers une solution différente comme le montrent les résultats du tableau 1.2. On observe également que l'algorithme n'a pas réussi à résoudre le problème pour certaines valeurs de s^0.

– Un autre point que l'on observe sur le tableau 1.2 est la constance de la solution calculée par PBLP en fonction du paramètre de pénalité k et le pas d'augmentation λ (la solution de départ s^0 étant fixée). Contrairement à DCA pour lequel la solution calculée est beaucoup influencée par les valeurs de k et λ.

4.5 Conclusion

Nous avons présenté un algorithme d'optimisation DC pour la résolution du problème de programmation bi-niveaux linéaire, où le problème du Suiveur est remplacé par les conditions d'optimalité KKT associées. Le problème est ensuite reformulé comme un programme DC à l'aide d'une pénalité exacte et une décomposition DC adéquate. L'algorithme proposé est simple et rapide, puisque seuls des programmes linéaires sont résolus à chaque itération. Les résultats numériques montrent, qu'avec un bon choix du paramètre de pénalité, qui dépend fortement du problème testé, l'algorithme converge souvent vers la solution globale. Afin de confirmer et comparer nos résultats, nous avons implémenté un autre algorithme basé sur la reformulation KKT du (PBL) et les fonctions de pénalité. Les résultats obtenus par les deux algorithmes sont comparables. Cependant, on remarque quelques points de différence concernant essentiellement la sensibilité des deux algorithmes aux paramètres d'entrée.

Conclusion générale

Le problème de programmation bi-niveaux linéaire (PBL) est un problème non convexe, compliqué et difficile à résoudre. Dans ce mémoire, nous avons traité ce problème par une technique d'optimisation non convexe et non différentiable basé sur la programmation DC et l'algorithme DCA.

Dans un premier temps, nous avons reformulé le problème en un programme mathématique en exploitant les conditions d'optimalité de Karush-Kuhn-Tucker associées au problème du niveau inférieur. Le programme obtenu est linéaire avec une contrainte de complémentarité non linéaire. C'est cette contrainte qui donne au problème sa nature non convexe et rend, par conséquent, l'application des outils de programmation convexe impossible.

Ensuite, en se basant sur les travaux de T. Pham Dinh et H.A Le Thi, nous avons utilisé les techniques de pénalité exacte en optimisation DC pour reformulé le programme résultant comme un programme de minimisation concave sous contraintes linéaires. A l'aide d'une décomposition DC adéquate, nous avons écrit ce dernier programme sous forme d'un programme DC pour lequel nous avons développé un algorithme DCA.

L'algorithme proposé est simple car, uniquement des sous programmes linéaires sont résolus à chaque étape. Les résultats numériques montrent que l'algorithme possède une convergence finie et rapide. La procédure de choix du point de départ semble efficace, mais la recherche de solution globale pour le problème par cet algorithme reste sensible au choix du paramètre de pénalité qui varie d'un problème à l'autre.

Un autre algorithme basé sur les fonctions de pénalité et conçu particulièrement pour les problèmes de programmation bi-niveaux linéaires sans contraintes du Leader a été implémenté. Des tests comparatifs sur des problèmes sans contraintes du Leader ont été effectués. Les résultats obtenus ont mis en évidence les points forts et les faiblesses des deux algorithmes.

Le travail réalisé dans ce mémoire reste dans le cadre de la formulation

optimiste du (PBL) où, le Leader peut supposer la coopération du Suiveur, dans le sens où ce dernier choisira à chaque fois la solution qui est la meilleure du point de vue du Leader. La formulation pessimiste du (PBL) est la situation où ce genre de coopération ne figure pas.

Comme perspectives de recherche, nous pouvons citer :

- L'extension de cette étude au cas pessimiste.
- L'étude des problèmes de programmation bi-niveaux non linéaires.
- L'étude des problèmes de programmation bi-niveaux multi-objectifs (linéaires et non linéaires).

Bibliographie

[1] E. Aiyoshi and K. Shimizu. A new computational method for Stackelberg and min-max problems by use of a penalty method. *IEEE Transactions on Automatic Control*, 26 :460–466, 1981.

[2] E. Aiyoshi and K. Shimizu. A solution method for the static constrained Stackelberg problem via penalty method. *IEEE Transactions on Automatic Control*, 29 :1111–1114, 1984.

[3] F.B. Akoa. *Approches de points intérieurs et de la programmation DC en optimisation non convexe. Codes et simulations numériques industrielles.* PhD thesis, Institut National Des Sciences Appliquées De Rouen, Janvier 2005.

[4] F.A. Al-Khayyal, R. Horst, and P.M. Pardalos. Global optimization of concave functions subject to quadratic constraints : An application in nonlinear bilevel programming. *Annals of Operations Research*, 34 :125–147, 1992.

[5] M.A. Amouzegar and K. Moshirvaziri. Determining optimal pollution control policies : An application of bilevel programming. *European Journal of Operational Research*, 119 :100–120, 1999.

[6] G. Anandalingam and T.L. Friesz. Hierarchical optimization : An introduction. *Annals of Operations Research*, (34) :1–11, 1992.

[7] G. Anandalingam and D.J. White. A solution for the linear static stackelberg problem using penalty functions. *IEEE Transactions on Automatic Control*, (10) :1170–1173, 1990.

[8] G. Anandalingam and D.J. White. A penalty function approach for solving bilevel linear programms. *Journal of Global Optimization*, (3) :397–419, 1993.

[9] T. Antczak. Exact penalty functions method for mathematical programming problems involving invex functions. *European Journal of Operational Research*, 2008.

[10] C. Audet. *Optimisation globale structurée : propriétés, équivalences et résolution.* PhD thesis, Département de mathématiques et de génie industriel. Université de Montréal, 1997.

[11] C. Audet, P. Hansen, B. Jaumard, and G. Savard. Links between linear bilevel and mixed 0-1 programming problems. *Les cahiers du GERAD, G-95-20, Montréal*, 1995.

[12] C. Audet, G. Savard, and W. Zghal. New branch and cut algorithm for bilevel linear programming. Technical report, Département de mathématique et de génie industriel. Ecole polytechnique de Montréal, Février 2004.

[13] J.F. Bard. An efficient point algorithm for a linear two stage optimization problem. *Operations Research*, 31(4) :670–684, 1983.

[14] J.F. Bard. Convex two-level optimization. *Mathematical Programming*, 40 :15–27, 1988.

[15] J.F. Bard. *Practical bilevel optimization : algorithms and applications*. Kluwer academic publishers, Dordrecht, 1998.

[16] J.F. Bard and T.A. Edmunds. Algorithms for nonlinear bilevel mathematical programs. *IEEE Transactions on systems, man, and cybernetics*, 21(1) :83–89, 1991.

[17] J.F. Bard and J.E. Falk. An explicit solution to the multilevel programming problem. *Computers and Operations Research*, 9(1) :77–100, 1982.

[18] J.F. Bard and J.T. Moore. A branch and bound algorithm for the bilevel programming problem. *SIAM Journey on Scientific and Statistical Comput.*, (11) :281–292, 1990.

[19] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty. *Nonlinear programming : Theory and algorithms*. John Wiley and sons, USA, 2006.

[20] O. Ben-Ayed and C.E. Blair. Computational difficulties of bilevel linear programming. *Operations Research*, (38) :556–560, 1990.

[21] D.P. Bertsekas. *Nonlinear programming*. Athena scientific , Belmont, 1999.

[22] W. Bialas, M. Karwan, and J. Shaw. A parametric complementarity pivot approach for two-level linear programming. Technical report, State University of New York at Buffalo, Operations Research Program, 1980.

[23] W.F. Bialas. Multilevel mathematical programming : An introduction. Technical report, Department of Industrial Engineering. University at Buffalo, 2002.

[24] W.F. Bialas and M.H. Karwan. Multilevel linear programming. Technical Report 78-1, Operations Research Program, Dept. of Industrial Eng., State University of New York at Buffalo, 1978.

[25] W.F. Bialas and M.H. Karwan. On two level optimization. *IEEE Transactions on Automatic Control*, 25(1) :211–215, 1982.

[26] W.F. Bialas and M.H. Karwan. Two-level linear programming. *Management Science*, 30(8) :1004–1020, 1984.

[27] J. Bracken and J.T. McGill. Mathematical programs with optimization problems in the constraints. *Mathematical Programming and Its Applications*, 21(1) :37–44, 1973.

[28] M. Campêlo and S. Scheimberg. A note on a modified simplex approach for solving bilevel linear programming problems. *European Journal of Operational Research*, 126 :454–458, 2000.

[29] M. Campêlo and S. Scheimberg. A study of local solutions in linear bilevel programming. *Journal of Optimization Theory and Applications*, 125(1) :63–84, 2005.

[30] W. Candler and R. Townsley. A linear two level programming problem. *Computers and Operations Research*, 9(1) :59–76, 1982.

[31] E.K.P. Chong and S. H. Zak. *An introduction to optimization.* John Wiley and sons, USA, 2001.

[32] F.H. Clarke, Y.S. Ledyaev, R.J. Stern, and P.R. Wolenski. *Nonsmooth analysis and control theory.* Springer-Verlag, New York, 1998.

[33] P.A. Clarke and A.W. Westerberg. A note on the optimality conditions for the bilevel programming problem. *Naval Research. Logistics*, 35 :413–418, 1988.

[34] J. Clegg, M. Smith, Y. Xiang, and R. Yarrow. Bilevel programming applied to optimising urban transportation. *Transportation Research Part B*, 35 :41–70, 2001.

[35] B. Colson, P. Marcotte, and G. Savard. A trust-region method for nonlinear bilevel programming : algorithm and computational experience. *Computational Optimization and Applications*, 30 :211–227, 2005.

[36] B. Colson, P. Marcotte, and G. Savard. Bilevel programming : A survey. *4OR A Quarterly Journal of Operations Research*, 2007.

[37] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of Operations Research*, (153) :235–256, 2007.

[38] M. Campêlo, S. Dantas and S. Scheimberg. A note on a penalty function approach for solving bilevel linear programs. *Journal of Global Optimization*, 16 :245–255, 2000.

[39] S. Delabriere and Y. Raynaud. *Optimisation convexe.* Université Pierre et Marie Curie - Paris 6, 2000-2001.

[40] S. Dempe. *Bilevel programming : A survey*, chapter 1.

[41] S. Dempe. First order optimality conditions for general bilevel programming. *Journal of Optimization : Theory and Applications*, 95(3) :735–739, 1997.

[42] S. Dempe. Annotated bibliography on bilevel programmaing and mathematical programs with equilibrium constraints. 2000.

[43] S. Dempe. *Foundations of Bilevel Programming*. Kluwer Academic Publishers, Dordrecht, 2002.

[44] S. Dempe and A.G. Mersha. Linear bilevel programming with upper level constraints depending on the lower level solution. *Applied Mathematics and Computation*, (180) :247–254, 2006.

[45] M. Dür. *Duality in global optimization : optimality conditions and algorithmical aspects*. Shaker Verlag, Aachen, Allemagne, 1999.

[46] L. Ekland and R. Temam. *Analyse convexe et problèmes variationnels*. Dunod, Paris, 1974.

[47] J.B.E. Etoa. *Optimisation hiérarchique : théorie, algorithmes et applications*. Publibook, France, 2008.

[48] J.E. Falk and J. Liu. On bilevel programming, part i : nonlinear cases. *Mathematical Programming*, (70) :47–72, 1995.

[49] J. Fliege and L.N. Vicente. A multicriteria approach to bilevel optimization. Technical report, Département de Mathématique, Université de Coimbre, 2004.

[50] J. Fortuny-Amat and B. McCarl. A representation and economic interpretation of a two level programming problem. *Operations Research*, 321(9) :783–792, 1981.

[51] J. Haddad, G. Savard, and C. Audet. A note on the definition of a linear bilevel programming solution. *Applied Mathematics and Computation*, (181) :351–355, 2006.

[52] A. Haurie, G. Savard, and D.J. White. A note on : An efficient point algorithm for a linear two stage optimization problem. *Operations Research*, 38(3) :553–555, 1990.

[53] S.R. Hejazi, A. Memariania, G. Jahanshahloob, and M.M. Sepehria. Solving method for a class of bilevel linear programming based on genetic algorithms. *Computers and Operations Research*, (29) :1913–1925, 2002.

[54] R. Horst and N.V. Thaoi. Dc programming : overview. *Journal of Optimization Theory and Applications*, 103(1) :1–43, 1999.

[55] Y. Ishizuka and E. Aiyoshi. Double penalty method for bilevel optimization problems. *Annals of Operations Research*, 34 :73–88, 1992.

[56] J.J. Júdice and A. Faustino. A sequential lcp method for bilevel linear programming. *Annals of Oprations Research*, 34 :89–106, 1992.

[57] J.J. Júdice and A. Faustino. The linear-quadratic bilevel programming problem. *Information Systems and Operational*, 32 :87–98, 1994.

[58] L. Kuen-Ming, W. Ue-Pyng, S. Hsu-Shih, and E. Stanley Leec. A hybrid neural network approach to bilevel programming problems. *Applied Mathematics Letters*, 2006.

[59] G. Liu, J. Han, and S. Wang. A trust region algorithm for bilevel programming problems. *Chinese Science Bulletin*, 43 :820–824, 1998.

[60] D.G. Luenberger. *Linear and nonlinear programming*. Addison Wesley publishing company, 1984.

[61] Z. Lukac, K. Sorie, and V.V. Rosenzweig. Production planning problem with sequence dependent setups as a bilevel programming problem. *European Journal of Operational Research*, 187 :1504–1512, 2008.

[62] Yibing Lv, Tiesong Hu, and Zhongping Wan. A penalty function method for solving weak price control problem. *Applied Mathematics and Computation*, (186) :1520–1525, 2007.

[63] Yibing Lv, Tiesong Hu, Guangmin Wang, and Zhongping Wan. A penalty function method based on Kuhn–Tucker condition for solving linear bilevel programming. *Applied Mathematics and Computation*, 2006.

[64] P. Marcotte and G. Savard. *Bilevel programming : A combinatorial perspective*, chapter 1.

[65] B. Martos. *Nonlinear programming : theory and methods*. American elsivier publisher, New York, 1975.

[66] Z. Dao Li, X. Qing and L. Zhenghua. A homotopy method for solving bilevel programming problem. *Nonlinear Analysis*, 57 :917 – 928, 2004.

[67] L. Zhi Quan, P. Jong Shi, R. Daniel, and W. Shi Quan. Exact penalization and stationarity conditions of mathematical programs with equilibrium constraints. *Mathematical Programming*, 75 :19–76, 1996.

[68] R.T. Rockafellar. *Convex analysis*. Princeton, USA, 1970.

[69] W. Schirotzek. *Nonsmooth Analysis*. Springer-Verlag, Berlin, 2007.

[70] T. Schüle, C. Schnörr, S. Weber, and J. Horneggerb. Discrete tomography by convex–concave regularization and d.c. programming. *Discrete Applied Mathematics*, 151 :229–243, 2005.

[71] Chenggen Shi, Guangquan Zhang, and Jie Lu. An extended kth-best approach for linear bilevel programming. *Applied Mathematics and Computation*, (164) :843–855, 2005.

[72] Chenggen Shi, Guangquan Zhang, and Jie Lu. An extended Kuhn–Tucker approach for linear bilevel programming. *Applied Mathematics and Computation*, (162) :51–63, 2005.

[73] Chenggen Shi, Guangquan Zhang, and Jie Lu. On the definition of linear bilevel programming solution. *Applied Mathematics and Computation*, (160) :169–176, 2005.

[74] Chenggen Shi, Guangquan Zhang, Jie Lu, and Hong Zhou. An extended branch and bound algorithm for linear bilevel programming. *Applied Mathematics and Computation*, (180) :529–537, 2006.

[75] S.R. Singiresu. *Engineering optimization : Theory and methods*. John Wiley and sons , USA, 1996.

[76] M. Soismaa. A note on efficient solutions for the linear bilevel programming problem. *European Journal of Operational Research*, 1999(112) :427–431.

[77] H.A. Le Thi. D.c. programming for solving a class of global optimization problems via reformulation by exact penalty. Technical report, Laboratory of modelling, optimization and operations research. National institute for applied sciences Rouen, 2003.

[78] H.A. Le Thi and T. Pham Dinh. A continuous approach for the concave cost supply problem via dc programming and dca. *Discrete Applied Mathematics*.

[79] H.A. Le Thi and T. Pham Dinh. Convex analysis approach to dc programming : theory and applications. *Acta Mathematica Vietnamica*, 22 :289–355, 1997.

[80] H.A. Le Thi and T. Pham Dinh. Solving a class of linearly constrained indefinite quadratic problems by dc algorithms. *Journal of Global Optimization*, 11 :253–285, 1997.

[81] H.A. Le Thi and T. Pham Dinh. A continuous approch for globally solving linearly constrained quadratic zero-one programming problems. *Optimization*, 50 :93–120, 2001.

[82] H.A. Le Thi and T. Pham Dinh. Dc programming : theory, algorithms and applications. the state of the art. Proceedings of The First International-Workshop on Global Constrained Optimization and Constraint Satisfaction (Cocos' 02), Valbonne-Sophia Antipolis, France, October 2002.

[83] H.A. Le Thi and T. Pham Dinh. The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems. *Annals of Operations Research*, 133 :23–46, 2005.

[84] H.A. Le Thi, P. Nguyen Trong, and T. Pham Dinh. A continuous dc programming approach to the strategic supply chain design problem from qualified partner set. *European Journal of Operational Research*, 183 :1001–1012, 2007.

[85] N.V. Thoai, Y. Yamamoto, and A. Yoshise. Global optimization method for solving mathematical programs with linear complementarity constraints. *Journal of Optimization Theory and Applications*, 124(2) :467–490, 2005.

[86] H. Tuy, A. Migdalas, and N.T. Hoai-Phuong. A novel approach to bilevel nonlinear programming. *Journal of Global Optimization*, 38 :527–554, 2007.

[87] L. Vicente, G. Savard, and J. Jùdice. Descent approaches for quadratic bilevel programming. *Journal of Optimization Theory and Applications*, 81(2) :379–399, 1994.

[88] L.N. Vicente. Bilevel programming : Introduction-history. 1997.

[89] L.N. Vicente and P.H. Calamai. Bilevel and multilevel programming : A bibliography review.

[90] G. Wang, Z. Wan, and X. Wang. Linear bilevel programming solution by genetic algorithm.

[91] S. Wenyu and Y. Ya xiang. *Optimization theory and methods. Nonlinear Programming*. Springer, Berlin, 2006.

[92] M. Willem. *Analyse convexe et optimisation*. Ciaco, 1987.

[93] W.I. Zangwill. Nonlinear programming via penalty functions. *Management Science*, 13 :344–358, 1967.

Résumé :

La résolution des problèmes d'optimisation multi-niveaux est devenue un sujet d'actualité sur le plan théorique et application.

Etant donnée la difficulté de résolution numérique de cette classe de problèmes, même pour le cas des programmes bi-niveaux linéaires, on rencontre différentes approches dans la littérature.

Dans le cadre de cette thèse, l'intérêt est porté à la résolution numérique d'un programme bi-niveaux linéaire avec des contraintes du Leader. L'approche utilisée consiste à remplacer le problème du Suiveur par ses conditions d'optimalité de Karush-Kuhn-Tucker. Le problème obtenu est résolu par une combinaison de la méthode de pénalité exacte, la méthode DC et l'algorithme DCA. Une étude comparative avec d'autres méthodes de résolution est donnée.

Mots clés : programmation bi-niveaux linéaire, programmation DC, conditions d'optimalité KKT, algorithme DCA, pénalité exacte.

ملخص :

هذه المذكرة مخصصة للحل الرقمي لنموذج رياضي خطي ذي مستويين. المنهج المتبع هو أن تحل محل المستوى الثاني شروط الأمثلية المعروفة بشروط KKT . النموذج المحصل عليه يتم حله بطريقه DC و النظام الخوارزمي DCA . في ختام هذه المذكرة تتم مقارنة مع طرق حل أخرى.

مفاتيح: نموذج رياضي خطي ذو مستويين، شروط أمثلية KKT ، طريقة DC ، نظام خوارزمي DCA .

Abstract :

The resolution of multilevel optimization problems became a topical subject on both theoretical and application framework.

Being given the numerical difficulty to solve this class of problems, even for the case of linear bilevel programs, one meets various approaches in the literature.

This thesis is devoted to the numerical resolution of a linear bilevel program with upper level constraints. The approach used consists in replacing the Follower's problem by its Karush-Kuhn-Tucker optimality conditions. The obtained problem is solved by a combination of exact penalty method, DC method and the DCA algorithm. A comparative study with other methods of resolution is given.

Key words : bilevel linear programming, DC programming, KKT optimality conditions, DCA algorithm, exact penalty.