Étude adaptative et comparative des principales variantes dans l'algorithme de Karmarkar

Abdelkrim Keraghel

THESE

Présentée à

L'Université Joseph Fourier - Grenoble I

pour obtenir le grade de
Docteur de l'Université Joseph Fourier
"Mathématiques Appliquées"

par

Abdelkrim KERAGHEL

ETUDE ADAPTATIVE ET COMPARATIVE DES PRINCIPALES VARIANTES DANS L'ALGORITHME DE KARMARKAR

Thèse soutenue le 4 Juillet devant la commission d'Examen

Monsieur	**P.J. LAURENT**	Président
Mesieurs	**J.P. CROUZEIX**	
	PHAM DINH TAO	
	F. ROBERT	
	P.TOLLA	Examinateurs
	H. IBISCH	
	NGUYEN VAN HIEN	Invités

UNIVERSITE Joseph FOURIER (GRENOBLE I)

Président de l'Université :
M. PAYAN Jean Jacques

Année Universitaire 1987 - 1988

MEMBRES DU CORPS ENSEIGNANT DE SCIENCES ET DE GEOGRAPHIE

PROFESSEURS DE 1ère Classe

ARNAUD Paul	Chimie Organique
ARVIEU ROBERT	Physique Nucléaire I.S.N.
AUBERT Guy	Physique C.N.R.S
AURIAULT Jean-Louis	Mécanique
AYANT Yves	Physique Approfondie
BARBIER Marie-Jeanne	Electrochimie
BARJON Robert	Physique Nucléaire ISN
BARNOUD Fernand	Biochimie Macromoléculaire Végétale
BARRA Jean-René	Statistiques-Mathématiques Appliquées
BECKER Pierre	Physique
BEGUIN Claude	Chimie Organique
BELORISKY Elie	Physique
BENZAKEN Claude	Mathématiques Pures
BERARD Pierre	Mathématiques Pures
BERNARD Alain	Mathématiques Pures
BERTRANDIAS Françoise	Mathématiques Pures
BERTRANDIAS Jean-Paul	Mathématiques Pures
BILLET Jean	Géographie
BOELHER Jean-Paul	Mécanique
BONNIER Jane Marie	Chimie Générale
BOUCHEZ Robert	Physique Nucléaire ISN
BRAVARD Yves	Géographie
CARLIER Georges	Biologie Végétale
CAUQUIS Georges	Chimie Organique
CHARDON Michel	Géographie
CHIBON Pierre	Biologie Animale
COHEN ADDAD Jean-Pierre	Physique
COLIN DE VERDIERE Yves	Mathématiques Pures
CYROT Michel	Physique du Solide
DEBELMAS Jacques	Géologie Générale
DEGRANGE Charles	Zoologie
DEMAILLY Jean-Pierre	Mathématiques Pures
DENEUVILLE Alain	Physique
DEPORTES Charles	Chimie Minérale
DOLIQUE Jean-Michel	Physique des Plasmas
DOUCE Roland	Physiologie Végétale
DUCROS Pierre	Cristallographie
FONTAINE Jean-Marc	Mathématiques Pures
GAGNAIRE Didier	Chimie Physique
GERMAIN Jean-Pierre	Mécanique,
GIRAUD Pierre	Géologie
HICTER Pierre	Chimie
IDELMAN Simon	Physiologie Animale
JANIN Bernard	Géographie
JOLY Jean-René	Mathématiques Pures
KAHANE André, détaché	Physique
KAHANE Josette	Physique
KRAKOWIAK Sacha	Mathématiques Appliquées

LAJZEROWICZ Jeanine	Physique
LAJZEROWICZ Joseph	Physique
LAURENT Pierre-Jean	Mathématiques Appliquées
LEBRETON Alain	Mathématiques Appliquées
DE LEIRIS Joël	Biologie
LHOMME Jean	Chimie
LLIBOUTRY Louis	Géophysique
LOISEAUX Jean-Marie	Sciences Nucléaires I.S.N.
LUNA Domingo	Mathématiques Pures
MACHE Régis	Physiologie Végétale
MASCLE Georges	Géologie
MAYNARD Roger	Physique du Solide
OMONT Alain	Astrophysique
OZENDA Paul	Botanique (Biologie Végétale)
PAYAN Jean-Jacques	Mathématiques Pures
PEBAY-PEYROULA Jean-Claude	Physique
PERRIER Guy	Géophysique
PIERRARD Jean-Marie	Mécanique
PIERRE Jean-Louis	Chimie Organique
RENARD Michel	Thermodynamique
RINAUDO Marguerite	Chimie CERMAV
ROSSI André	Biologie
SAXOD Raymond	Biologie Animale
SENGEL Philippe	Biologie Animale
SERGERAERT Francis	Mathématiques Pures
SOUCHIER Bernard	Biologie
SOUTIF Michel	Physique
STUTZ Pierre	Mécanique
TRILLING Laurent	Mathématiques Appliquées
VALENTIN Jacques	Physique Nucléaire I.S.N.
VAN CUTSEM Bernard	Mathématiques Appliquées
VIALON Pierre	Géologie

PROFESSEURS de 2ème Classe

ADIBA Michel	Mathématiques Pures
ANTOINE Pierre	Géologie
ARMAND Gilbert	Géographie
BARET Paul	Chimie
BLANCHI J.Pierre	STAPS
BLUM Jacques	Mathématiques Appliquées
BOITET Christian	Mathématiques Appliquées
BORNAREL Jean	Physique
BRUANDET J.François	Physique
BRUGAL Gérard	Biologie
BRUN Gilbert	Biologie
CASTAING Bernard	Physique
CERFF Rudiger	Biologie
CHIARAMELLA Yves	Mathématiques Appliquées
COURT Jean	Chimie
DUFRESNOY Alain	Mathématiques Pures
GASPARD François	Physique
GAUTRON René	Chimie
GENIES Eugène	Chimie
GIDON Maurice	Géologie
GIGNOUX Claude	Sciences Nucléaires
GILLARD Roland	Mathématiques Pures
GIORNI Alain	Sciences Nucléaires
GONZALEZ SPRINBERG Gérardo	Mathématiques Pures
GUIGO Maryse	Géographie
GUMUCHAIN Hervé	Géographie
GUITTON Jacques	Chimie

HACQUES Gérard	Mathématiques Appliquées
HERBIN Jacky	Géographie
HERAULT Jeanny	Physique
JARDON Pierre	Chimie
JOSELEAU Jean-Paul	Biochimie
KERCKHOVE Claude	Géologie
LONGEQUEUE Nicole	Sciences Nucléaires I.S.N.
LUCAS Robert	Physique
MANDARON Paul	Biologie
MARTINEZ Francis	Mathématiques Appliquées
NEMOZ Alain	Thermodymamique CNRS - CRTBT
OUDET Bruno	Mathématiques Appliquées
PECHER Arnaud	Géologie
PELMONT Jean	Biochimie
PERRIN Claude	Sciences Nucléaires I.S.N.
PFISTER Jean-Claude	Physique du Solide
PIBOULE Michel	Géologie
RAYNAUD Hervé	Mathématiques Appliquées
RICHARD Jean-Marc	Physique
RIEDTMANN Christine	Mathématiques Pures
ROBERT Gilles	Mathématiques Pures
ROBERT Jean-Bernard	Chimie Physique
SARROT-REYNAULD Jean	Géologie
SAYETAT Françoise	Physique
SERVE Denis	Chimie
STOECKEL Frédéric	Physique
SCHOLL Pierre-Claude	Mathématiques Appliquées
SUBRA Robert	Chimie
VALLADE Marcel	Physique
VIDAL Michel	Chimie Organique
VIVIAN Robert	Géographie
VOTTERO Philippe	Chimie

MEMBRES DU CORPS ENSEIGNANT DE L' IUT 1

PROFESSEURS de 1ère Classe

BUISSON Roger	Physique IUT 1
DODU Jacques	Mécanique Appliquée IUT 1
NEGRE Robert	Génie Civil IUT 1
NOUGARET Marcel	Automatique IUT 1
PERARD Jacques	EEA. IUT 1

PROFESSEURS de 2ème classe

BOUTHINON Michel	EEA. IUT 1
CHAMBON René	Génie Mécanique IUT 1
CHEHIKIAN Alain	EEA. IUT 1
CHENAVAS Jean	Physique IUT 1
CHOUTEAU Gérard	Physique IUT 1
CONTE René	Physique IUT 1
GOSSE Jean-Pierre	EEA.IUT 1
GROS Yves	Physique IUT 1
KUHN Gérard, (Détaché)	Physique IUT 1
MAZUER Jean	Physique IUT 1
MICHOULIER Jean	Physique IUT 1
MONLLOR Christian	EEA.IUT 1
PEFFEN René	Métallurgie IUT 1
PERRAUD Robert	Chimie IUT 1
PIERRE Gérard	Chimie IUT 1
TERRIEZ Jean-Michel	Génie Mécanique IUT 1
TOUZAIN Philippe	Chimie IUT 1
VINCENDON Marc	Chimie IUT 1

PROFESSEURS DE PHARMACIE

AGNIUS-DELORD Claudine	Physique	Faculté La Tronche
ALARY Josette	Chimie Analytique	Faculté La Tronche
BERIEL Hélène	Physiologie et Pharmacologie	Faculté La Tronche
CUSSAC Max	Chimie Therapeutique	Faculté La Tronche
DEMENGE Pierre	Pharmacodynamie	Faculté La Tronche
FAVIER Alain	Biochimie	C.H.R.G.
JEANNIN Charles	Pharmacie Galénique	Faculté Meylan
LATURAZE Jean	Biochimie	Faculté La Tronche
LUU DUC Cuong	Chimie Générale	Faculté La Tronche
MARIOTTE Anne-Marie	Pharmacognosie	Faculté La Tronche
MARZIN Daniel	Toxicologie	Faculté Meylan
RENAUDET Jacqueline	Bactériologie	Faculté La Tronche
ROCHAT Jacques	Hygiéne et Hydrologie	Faculté La Tronche
SEIGLE-MURANDI Françoise	Botanique et Cryptogamie	Faculté Meylan
VERAIN Alice	Pharmacie Galénique	Faculté Meylan

MEMBRES DU CORPS ENSEIGNANT DE MEDECINE

PROFESSEURS CLASSE EXEPTIONNELLE ET 1ère CLASSE

AMBLARD Pierre	Dermatologie	C.H.R.G.
AMBROISE-THOMAS Pierre	Parasitologie	C.H.R.G.
BEAUDOING André	Pédiatrie-Puericulture	C.H.R.G.
BEZEZ Henri	Orthopédie-Traumatologie	Hopital SUD
BONNET Jean-Louis	Ophtalmologie	C.H.R.G.
BOUCHET Yves	Anatomie	Faculté La Merci
	Chirurgie Générale et Digestive	C.H.R.G.
BUTEL Jean	Orthopédie-Traumatologie	C.H.R.G.
CHAMBAZ Edmond	Biochimie	C.H.R.G.
CHAMPETIER Jean	Anatomie-Topographique et Appliquée	C.H.R.G.
CHARACHON Robert	O.R.L.	C.H.R.G.
COLOMB Maurice	Immunologie	Hopital sud
COUDERC Pierre	Anatomie-Pathologique	C.H.R.G.
DELORMAS Pierre	Pneumophtisiologie	C.H.R.G.
DENIS Bernard	Cardiologie	C.H.R.G.
GAVEND Michel	Pharmacologie	Faculté La Merci
HOLLARD Daniel	Hématologie	C.H.R.G.
LATREILLE René	Chirurgie Thoracique et Cardiovasculaire	C.H.R.G.
LE NOC Pierre	Bactériologie-Virologie	C.H.R.G.
MALINAS Yves	Gynécologie et Obstétrique	C.H.R.G.
MALLION Jean-Michel	Médecine du Travail	C.H.R.G.
MICOUD Max	Clinique Médicale et Maladies Infectieuses	C.H.R.G.
MOURIQUAND Claude	Histologie	Faculté La Merci
PARAMELLE Bernard	Pneumologie	C.H.R.G.
PERRET Jean	Neurologie	C.H.R.G.
RACHAIL Michel	Hépato-Gastro-Entérologie	C.H.R.G.
DE ROUGEMONT Jacques	Neurochirurgie	C.H.R.G.
SARRAZIN Roger	Clinique Chirurgicale	C.H.R.G.
STIEGLITZ Paul	Anestésiologie	C.H.R.G.
TANCHE Maurice	Physiologie	Faculté La Merci
VIGNAIS Pierre	Biochimie	Faculté La Merci

PROFESSEURS 2ème CLASSE

BACHELOT Yvan	Endocrinologie	C.H.R.G.
BARGE Michel	Neurochirurgie	C.H.R.G.
BENABID Alim Louis	Biophysique	Faculté La Merci
BENSA Jean-Claude	Immunologie	Hopital Sud
BERNARD Pierre	Gynécologie-Obstétrique	C.H.R.G.
BESSARD Germain	Pharmacologie	ABIDJAN
BOLLA Michel	Radiothérapie	C.H.R.G.
BOST Michel	Pédiatrie	C.H.R.G.
BOUCHARLAT Jacques	Psychiatrie Adultes	Hopital Sud
BRAMBILLA Christian	Pneumologie	C.H.R.G.
CHIROSSEL Jean-Paul	Anatomie-Neurochirurgie	C.H.R.G.
COMET Michel	Biophysique	Faculté La Merci
CONTAMIN Charles	Chirurgie Thoracique et Cardiovasculaire	C.H.R.G.
CORDONNIER Daniel	Néphrologie	C.H.R.G.
COULOMB Max	Radiologie	C.H.R.G.
CROUZET Guy	Radiologie	C.H.R.G.
DEBRU Jean-Luc	Médecine Interne et Toxicologie	C.H.R.G.
DEMONGEOT Jacques	Biostatistiques et Informatique Médicale	Faculté La Merci
DUPRE Alain	Chirurgie Générale	C.H.R.G.
DYON Jean-François	Chirurgie Infantile	C.H.R.G.
ETERRADOSSI Jacqueline	Physiologie	Faculté La Merci
FAURE Claude	Anatomie et Organogénèse	C.H.R.G.
FAURE Gilbert	Urologie	C.H.R.G.
FOURNET Jacques	Hépato-Gastro-Entérologie	C.H.R.G.
FRANCO Alain	Médecine Interne	C.H.R.G.
GIRARDET Pierre	Anesthésiologie	C.H.R.G.
GUIDICELLI Henri	Chirurgie Générale et Vasculaire	C.H.R.G.
GUIGNIER Michel	Thérapeutique et Réanimation Médicale	C.H.R.G.
HADJIAN Arthur	Biochimie	Faculté La Merci
HALIMI Serge	Endocrinologie et Maladies Métaboliques	C.H.R.G.
HOSTEIN Jean	Hépato-Gastro-Entérologie	C.H.R.G.
HUGONOT Robert	Médecine Interne	C.H.R.G.
JALBERT Pierre	Histologie-Cytogénétique	C.H.R.G.
JUNIEN-LAVILLAUROY Claude	O.R.L.	C.H.R.G.
KOLODIE Lucien	Hématologie Biologique	C.H.R.G.
LETOUBLON Christian	Chirurgie Générale	C.H.R.G.
MACHECOURT Jacques	Cardiologie et Maladies Vasculaires	C.H.R.G.
MAGNIN Robert	Hygiène	C.H.R.G.
MASSOT Christian	Médecine Interne	C.H.R.G.
MOUILLON Michel	Ophtalmologie	C.H.R.G.
PELLAT Jacques	Neurologie	C.H.R.G.
PHELIP Xavier	Rhumatologie	C.H.R.G.
RACINET Claude	Gynécologie-Obstétrique	Hopital Sud
RAMBAUD Pierre	Pédiatrie	C.H.R.G.
RAPHAEL Bernard	Stomatologie	C.H.R.G.
SCHAERER René	Cancérologie	C.H.R.G.
SEIGNEURIN Jean-Marie	Bactériologie-Virologie	Faculté La Merci
SELE Bernard	Cytogénétique	Faculté La Merci
SOTTO Jean-Jacques	Hématologie	C.H.R.G.
STOEBNER Pierre	Anatomie Pathologique	C.H.R.G.
VROUSOS Constantin	Radiothérapie	C.H.R.G.

بسم الله الرحمن الرحيم

قيل للإمام الشافعي رحمه الله : كيف حصلت العلم ؟
فقال: حصلته بأربع خصال:
بكور كبكور الغراب وحرص كحرص الخنازير وتملق كتملق الهرة وصبر كصبر الحمار.

ثم قال: أما بعد، فإن العلم بطيء اللزام، بعيد المرام لا يدرك بالسهام ولا يرى في المنام ولا يورث عن الآباء والأعمام. إنما هو شجرة لا تصلح إلا بالغرس ولا تغرس إلا في النفس ولا تسقى إلا بالدرس ولا تثبت إلا بإدمان السهر وقلة النوم وصلة الليل بالليل ...

Ce travail a été réalisé au sein du Laboratoire TIM3 de l'Institut d'Informatique et de Mathématiques Appliquées de Grenoble .

Je tiens à exprimer toute ma reconnaissance à Monsieur Pham Dinh TAO, Directeur de l'Equipe Modélisation & Optimisation du Laboratoire TIM3, qui a guidé mes premiers pas dans la recherche, pour sa disponibilité, son soutien, ses conseils et sa bienveillance durant l'élaboration de cette thèse .

Mes sincères remerciements s'adressent à Monsieur Pierre-Jean Laurent, Professeur à l'Université Joseph Fourier - Grenoble1, pour tout ce qu'il a fait pour moi au niveau administratif . Je suis très sensible à l'honneur qu'il me fait en présidant ce jury . On peut dire que l'Equipe Modélisation & Optimisation est un héritage de ses contributions au développement de l'Analyse et l'Optimisation convexe au niveau nationnal et international . Nous sommes tous ses élèves . Qu'il nous permette de lui témoigner notre sincère et respectueuse reconnaissance .

Mes vifs remerciements vont également à :

Monsieur François Robert, Professeur de l'I.N.P.G, pour ses enseignements, et pour son amabilité d'avoir bien voulu siéger dans ce jury .

Messieurs Jean Pierre Crouzeix, Professeur à l'Université de Clermont Ferrand et Pierre Tolla, Professeur à l'Université de Paris-Dauphine, pour avoir accepté de juger cette thèse et d'en être les rapporteurs . Leurs suggestions et les discussions fructueuses que j'ai pu avoir avec chacun d'eux m'ont permis de mener à bien ce travail . Qu'ils veuillent bien agréer l'expression de mes sincères et respectueuses considérations .

Messieurs H. Ibisch, Professeur à l'Université de Nantes et Nguyen Van Hien, Professeur et Directeur du Département de Mathématiques à l'Université de Namur (Belgique), pour s'être intéressés à ce travail et pour avoir accepté de se déplacer pour participer au jury .

Mademoiselle C. Fraley, de l'Université de Stanford, pour les nombreuses discussions que nous avons pu avoir ensemble .

Je tiens enfin à exprimer toute ma reconnaissance à tous les membres de l'Equipe Modélisation & Optimisation avec lesquels j'ai passé des années d'amitié et d'harmonie dans le travail .

Merci également à tous mes amis et collègues qui ont rendu mon séjour en France agréable .

Je remercie toute l'équipe du service de reprographie pour l'exellente qualité de leur travail .

TABLE DES MATIERES

INTRODUCTION GENERALE

Le terme *"programmation linéaire"* a été introduit en même temps que la méthode du simplexe par G. DANTZIG au lendemain de la seconde guerre mondiale .

C'est un outil mathématique très riche qui constitue la technique la plus célèbre de la recherche opérationnelle .

Or, depuis longtemps la programmation linéaire présente la particularité d'être bien résolue par le praticien mais mal résolue en théorie

En effet , l'algorithme du simplexe , réputé très efficace dans ce domaine résoud dans la majorité des cas un programme linéaire de m contraintes et n variables en un temps polynomial (le nombre d'itérations est un polynôme en m et n) alors que sa complexité théorique est de l'ordre de 2^n itérations .

En (1979) KHACHIYAN [38] propose un algorithme polynomial pour la programmation linéaire . Ce fut un grand succès théorique, malheureusement cet algorithme est beaucoup moins efficace que l'algorithme du simplexe pour résoudre la plupart des programmes linéaires . Ce qui montre en partie que l'efficacité pratique d'un algorithme ne découle pas nécessairement de sa polynomialité théorique . De plus prouver la polynomialité d'un algorithme est une chose et retrouver son efficacité pratique en est une autre, les deux étant d'une importance indépendante .

En fait , l'inefficacité pratique de l'algorithme de KHACHIYAN s'explique par le fait que l'algorithme en question provient de la méthode des ellipsoïdes qui n'est autre qu'une méthode de sous-gradient (cas particulier de la méthode de dilatation d'espace dans la direction d'un sous-gradient) .

En (1984) KARMARKAR [35] propose un algorithme polynomial plus attractif : C'est une approche originale et pratique pour résoudre des programmes linéaires . Il est (aux dires de son auteur) 50 à 100 fois plus rapide que l'algorithme du simplexe , ce qui a provoqué quelques controverses dans la communauté de la programmation mathématique et incité plusieurs recherches dans le domaine de la programmation linéaire qui en plus de leur intérêt théorique confirment en partie les propos de KARMARKAR .

Cet algorithme minimise une fonction linéaire (qui vaut zéro à l'optimum) sur un simplexe régulier S qui est un ensemble de n-vecteurs non-négatifs (≥ 0) dont la somme des composantes est égale à une constante (par exemple 1 ou n) .

L'analyse de l'algorithme est basée essentiellement sur les propriétés géométriques de ce simplexe . En particulier la plus petite sphère de rayon R (qui le contient) et la plus grande sphère de rayon r (qu'il contient) ont le même centre et $R/r = n$.

A chaque itération, KARMARKAR utilise une transformation projective de S dans lui même qui envoie la solution réalisable courante au centre de S . L'algorithme trouve alors une direction de descente suivant laquelle on effectue un déplacement convenable (αr) à partir du centre Finalement, on revient aux variables d'origine en utilisant l'inverse de la transformation projective . On répète alors le processus jusqu'à la verification du test d'optimalité .

Pour prouver la polynomialité, KARMARKAR introduit une " fonction potentiel " possédant des propriétés importantes aussi bien sur le plan pratique que sur le plan théorique .

Les quatres chapitres de cette thèse constituent une étude allant du développement théorique à l'implémentation numérique de cette méthode .

Le chapitre 1 est consacré à la présentation générale de la méthode de KARMARKAR . Cette présentation, qui est voulue claire et précise, tient compte de nombreux travaux récents sur la méthode . Elle permet une profonde compréhension de la méthode de KARMARKAR, en facilite l'implémentation tout en dégageant les difficultés pratiques qui seront étudiées en détail dans les chapitres suivants .

La difficulté principale de cette méthode est de supposer, au départ, connue, la valeur optimale . Condition très restrictive en pratique . Plusieurs techniques sont proposées dans la littérature en vue de relaxer cette hypothèse et d'étendre ainsi l'algorithme à un programme linéaire plus général . KARMARKAR lui même suggère deux méthodes polynomiales :

(a) - La méthode " primale - duale " qui consiste à combiner le problème primal avec son dual pour obtenir un problème de faisabilité qui, d'après KARMARKAR est équivalent à un programme linéaire dont la valeur optimale vaut zéro . Du point de vue pratique, cette méthode est

la plus simple à mettre en œuvre, cependant, la taille du problème augmente énormément : (m + n + 1) contraintes et 2(m + n + 1) variables sachant que le problème de départ comporte m contraintes et n variables, ce qui rend l'algorithme beaucoup plus coûteux . Une implémentation préliminaire de cette méthode est présentée par GOLDFARB et MEHROTRA [27] .

(b) - La " sliding objective function method " qui consiste à localiser la valeur optimale exacte z* du problème dans un intervalle qu'on améliore au cours des itérations par une sorte de recherche binaire, les deux extrémités de cet intervalle étant à chaque fois une borne inférieure et une borne supérieure de z* . Nous avons pu simplifier la description de cette approche mais elle reste tout de même indésirable en pratique car d'une part, c'est un algorithme primal seulement, et d'autre part, elle est compliquée par l'utilisation de deux bornes alors qu'on pourrait se contenter d'une seule . A notre connaissance, cette variante n'a jamais été implémentée .

La première modification pratique de l'algorithme de KARMARKAR qui ne nécessite pas la connaissance de z* est proposée par TODD et BURRELL [48] . Elle est basée sur l'approximation de z* par des bornes inférieures fournies par le problème dual . Dans le but de convertir un programme général en un programme dans S, ces derniers utilisent la méthode de " big M " dont on connait les inconvénients sur le plan numérique lors de son implémentation . Nous avons pensé alors à combiner cette technique avec la transformation projective de KARMARKAR . Cette démarche a été aussi celle de GAY [20] , YE et KOJIMA [58] , dont nous sommes au courant plus tard lors de la publication de leurs résultats .

YE développe de sa part un algorithme primal dans lequel la valeur z* est approximée à chaque itération par une borne supérieure qui est la valeur de l'objectif au point courant . Même si la polynomialité de son algorithme n'a pas pu être prouvée théoriquement, il semble actuellement être l'un des plus performants en pratique . Ces différentes extensions et variantes seront abordées au chapitre 2 dans lequel nous présenterons une approche unificatrice des variantes les plus significatives .

Dans ce type d'algorithmes, le coût d'une itération est dominé par le calcul de la direction de descente qui nécessite la résolution du système linéaire :

$$AD_k^2A^tu = AD_k^2c \qquad (1)$$

où A , c désignent respectivement, la matrice des contraintes et le vecteur coût du problème envisagé et D_k la matrice diagonale ayant pour éléments diagonaux les composantes de l'itéré x^k .

Le succès de l'implémentation de l'algorithme de KARMARKAR ou de l'une de ses variantes dépend d'une part de l'efficacité du procédé de résolution du système (1) qui peut réduire significativement le coût d'une itération, et d'autre part, du pas de déplacement choisi qui peut accélérer la convergence de l'algorithme et réduire par voie de conséquence la complexité totale de l'algorithme . Ces deux points seront abordés en détail au chapitre 3 .

Pour résoudre le système (1) , on peut utiliser soit une méthode directe dans laquelle on factorise la matrice du système (comme par exemple la méthode d'élimination de GAUSS) soit une méthode itérative qui produit une suite de solutions approchées du système et n'utilise en général que des multiplications du type matrice-vecteur telle que la méthode du gradient conjugué .

On ne peut pas savoir à priori quelle pourrait être la meilleure approche . En effet, le choix d'une méthode dépend de plusieurs facteurs : De la dimension du système, la structure de la matrice (creux, conditionnement, etc...), et même de l'architecture de la machine utilisée

Ceci étant, le système (1) possède deux caractéristiques importantes :

1 - Sa matrice est symétrique définie positive . La méthode fréquemment utilisée dans ce cas est la factorisation de CHOLESKY . De même, la méthode du gradient conjugué préconditionné est considérée actuellement comme l'une des méthodes les plus efficaces pour résoudre de tels systèmes .

2 - Au cours de l'exécution de l'algorithme, seuls les éléments de la matrice D_k peuvent changer d'une itération à l'autre . KARMARKAR avait exploité cet avantage pour modifier son algorithme de façon à obtenir la matrice du système (à l'itération k+1) à partir de la matrice (à l'itération k) après un nombre fini de modifications de rang 1. SHANNOS [45] avait remarqué que le nombre nécessaire de modifications est très petit, ce qui favorise une méthode de CHOLESKY extrapolée basée sur l'algorithme de FLETCHER et POWELL [18] . Ce procédé conduit à une réduction considérable de la complexité totale de l'algorithme .

GILL et AL. [23] , I. ADLER et N. KARMARKAR [2] , ont proposé indépendamment des implémentations sophistiquées qui utilisent une méthode de gradient conjugué préconditionné par un facteur triangulaire de CHOLESKY d'une matrice " creuse " approximant $AD_k^2A^t$.

Nous avons utilisé la factorisation de CHOLESKY ainsi que la méthode du gradient conjugué classique . Les résultats obtenus sont pratiquement les mêmes pour les " petits " exemples que nous avons testés .

La vitesse de convergence dépend en grande partie de la valeur choisie pour le pas de déplacement α . KARMARKAR lui attribue la valeur 1/4 . Nous avons montré au chapitre 1 que l'on peut très bien prendre $\alpha = 1/2$ tout en conservant les propriétés fondamentales de l'algorithme .

En pratique, l'algorithme est deux fois plus rapide pour $\alpha = 1/2$ que pour $\alpha = 1/4$. Ceci nous a conduit à prendre α le plus grand possible tout en restant dans l'intérieur du simplexe , nous avons alors constaté que plus " α est grand plus vite l'algorithme converge " , ce résultat est devenu actuellement classique, on peut même prendre $\alpha > 1$ en effectuant par exemple une recherche linéaire suivant la direction de descente minimisant approximativement la fonction potentiel comme le suggèrent TODD et BURRELL et beaucoup d'autres . On peut également utiliser une technique plus simple et moins laborieuse qui garantit la faisabilité et la monotonie comme nous avons fait au chapitre 3 .

Nous avons implémenté plusieurs versions de l'algorithme de KARMARKAR qui correspondent aux différentes variantes présentées ci-dessus . Ces expérimentations numériques et les commentaires sur l'algorithme sont exposés dans le dernier chapitre .

Le succès théorique de la méthode de KARMARKAR est indiscutable, car si la méthode du simplexe a motivé toute seule 40 ans de développement, il faut reconnaître aussi que celle de KARMARKAR a connu des progrès impressionnants dans un temps minime . La profondeur et l'importance de cette nouvelle méthode peuvent être mesurées à partir du nombre et de la qualité des exposés présentés dans les grandes conférences et les papiers parus dans les principales revues, et aussi à partir du nombre de chercheurs de très haut niveau engagés dans l'étude de cette méthode .

Reste à savoir s'il s'agit vraiment d'un algorithme pratique susceptible de résoudre des programmes linéaires réels ? . Pour répondre à cette question, il n'est pas nécessaire d'opposer l'algorithme de KARMARKAR à celui du simplexe, car la réponse semble maintenant claire : Nombreux chercheurs ont mis en évidence l'utilisation pratique des méthodes de point intérieur (suggérées par la méthode projective originale de KARMARKAR) même si cette évidence ne signifie pas que ces nouvelles méthodes sont uniformément 50 fois (ou plus) plus rapides comme c'est prétendu au début, et même si leur implémentation effective est plus sophistiquée et demande plus de soins .

A titre indicatif, on peut dire qu'à travers nos expérimentations numériques sur PC relatives à des programmes linéaires de faible taille, l'algorithme du simplexe est nettement plus rapide . Cependant, pour des problèmes de grande taille, des résultats très supérieurs à ceux fournis par les méthodes simpliciales classiques ont été obtenus par ADLER et AL. [1] , LISSER [39] .

Pour terminer, nous pensons que la méthode du simplexe reste la méthode de choix pour la plupart des programmes linéaires à cause de sa disponibilité et ses avantages calculatoires . Les performances de l'algorithme de KARMARKAR seront à apprécier dans les programmes linéaires à grande dimension, là où la méthode du simplexe n'est pas applicable ou s'avère non satisfaisante . C'est d'ailleurs ça le but de la méthode .

Ceci étant, avec la méthode de KARMARKAR naissent de nombreuses directions et idées de recherche particulièrement intéressantes en optimisation . Et c'est le mot de la fin .

Notations générales

Soit x et y deux vecteurs de $\mathbb{R}^n$:

la notation $<x,y> = x^t y$ désigne le produit scalaire de x par y .

e_n désignera le vecteur $(1,.....,1)^t \in \mathbb{R}^n$.

On note par $\sum_n x_i$ la somme des n composantes de x .

La norme euclidienne de x sera notée par $||x||^2 = \sum_n x_i^2$.

Une matrice A à m lignes et n colonnes sera considérée comme un élément de $\mathbb{R}^{mxn}$. On écrira $A \in \mathbb{R}^{mxn}$.

Soit $A \in \mathbb{R}^{nxn}$ et $b \in \mathbb{R}^n$, alors :

L'écriture A = diag{b} signifie que A est une matrice diagonale dont les éléments diagonaux sont les composantes du vecteur b .

A^t , b^t désignent respectivement la matrice transposée de A et le vecteur transposé de b .

L'inverse d'une matrice A sera noté par A^{-1} .

CHAPITRE 1

PRESENTATION GENERALE DE LA METHODE DE KARMARKAR

I . 0 - Introduction

Dans ce chapitre on parlera essentiellement de la méthode de KARMARKAR tout en signalant - chaque fois qu'il est nécessaire de le faire - les imprécisions théoriques et les difficultés pratiques que nous avons rencontrées au cours de l'implémentation de cet algorithme . Nous montrerons à l'occasion comment s'en sortir .

La première partie (§ I . 1) de ce chapitre constitue un bref rappel de certaines propriétés fondamentales dans la programmation linéaire .

La deuxième partie (§ I . 2) est consacrée aux grandes lignes de la méthode .

Dans la troisième et dernière partie (§ I . 3) on décrira la méthode tout en la justifiant .

I . 1 - NOTIONS FONDAMENTALES :

Dans ce court paragraphe , on présente simplement quelques notions fondamentales de la programmation linéaire qui serviront de support à l'intuition dans les développements ultérieurs .

I . 1.1- Définition d'un Programme mathématique :

Etant donné un domaine $D \subset \mathbb{R}^n$ et une fonction $f : D \mapsto \mathbb{R}^n$.
Un problème de programmation mathématique consiste à minimiser (ou maximiser) f sur D .
Autrement dit , il s'agit de trouver un point $x^* \in D$ (si un tel point existe) pour lequel :

$f(x) \geq f(x^*)$ pour tout $x \in D$ (dans le cas de minimisation)

et

$f(x) \leq f(x^*)$ pour tout $x \in D$ (dans le cas de maximisation)

Symboliquement ce problème de programmation mathématique est représenté par la notation :

$$(*) \begin{cases} \min f(x) \\ x \in D \end{cases}$$

. Un point de D est appelé *"solution réalisable"* de (*) .

. On appelle *"solution optimale"* de (*) toute solution réalisable x* réalisant le minimum de f .

. la valeur f* = f(x*) est appelée *"valeur optimale"*.

I . 1. 2 - Définition d'un Programme linéaire :

Un programme linéaire est un programme mathématique dans lequel

. Le domaine D des solutions réalisables est défini par un ensemble d'équations et/ou d'inéquations linéaires appelées *"contraintes"*.

. La fonction f , dite *"fonction objectif ou économique"* , est linéaire .

I . 1. 3 - Equivalence de deux programmes mathématiques :

Deux programmes mathématiques (p) et (p') (en particulier deux programmes linéaires) sont dits *"équivalents"* si à toute solution réalisable de l'un on sait faire correspondre une solution réalisable de l'autre de telle sorte que les fonctions objectives soient égales pour cette paire de solutions .

I . 1. 4 - Forme canonique et forme standard d'un programme linéaire :

. Un programme linéaire est écrit sous *"forme canonique"* si le domaine des solutions réalisables est défini par un système d'inéquations linéaires .

. Si par contre le domaine des solutions réalisables est défini par un système d'équations linéaires , on dit que le programme est écrit sous *"forme standard"*.

. On peut toujours mettre un programme linéaire quelconque sous forme standard en introduisant des variables supplémentaires appelées *"variables d'écart"*.

Exemple :

$$(1)\ \begin{cases} \min\ 5x_1 - 3x_2 \\ x_1 - x_2 \geq 2 \\ 2x_1 + 3x_2 \leq 4 \\ -x_1 + 6x_2 = 10 \\ x_1 \geq 0\ ,\ x_2 \geq 0 \end{cases}$$

En introduisant les variables d'écart $x_3 \geq 0$ et $x_4 \geq 0$ dans la première et la deuxième contrainte , on met (1) sous la forme équivalente :

$$(2)\ \begin{cases} \min 5x_1 - 3x_2 + 0.x_3 + 0.x_4 \\ x_1 - x_2 - x_3 = 2 \\ 2x_1 + 3x_2 + x_4 = 4 \\ -x_1 + 6x_2 = 10 \\ x_1 \geq 0\ ,\ x_2 \geq 0\ ,\ x_3 \geq 0\ ,\ x_4 \geq 0 \end{cases}$$

(2) est la forme standard de (1) .

Pour cette raison , on ne considérera dans ce qui suit que des programmes linéaires sous forme standard du type :

$$(P)\ \begin{cases} \min c^t x \\ Ax = b \\ x \geq 0 \end{cases}$$

où

n désigne le nombre de variables
m = nombre de contraintes
$A \in \mathbb{R}^{mxn}$ est la matrice des contraintes
$c \in \mathbb{R}^n$ est le vecteur coût
$b \in \mathbb{R}^m$ est le second membre .

Remarques :

1- On peut toujours supposer que l'on a rang(A) = m (c'est le cas dans la méthode de KARMARKAR) .
En effet , si rang(A) < m , une ou plusieurs lignes de A peuvent s'exprimer comme combinaisons linéaires des autres . Suivant la valeur des composantes b_i de b , les contraintes correspondantes sont soit redondantes (auquel cas elles peuvent être éliminées du problème) soit incompatibles avec les autres (auquel cas le système Ax = b n'a pas de solutions).

2 - Il n'est pas restrictif de supposer que la variable x est astreinte à être non-négative . En effet, si la variable x est non contrainte en signe , on pourra remplacer x par la différence $x^+ - x^-$ de deux variables x^+ et x^- astreintes , elles , à ne prendre que des valeurs non-négatives . Le programme qui en résulte est évidemment un programme linéaire de la forme (p) .

I . 2 - PRELIMINAIRES

I . 2 . 1 - Minimisation d'une fonction linéaire sur un ellipsoide

Considérons le problème suivant :

$$(0)\ \begin{cases} \min c^t x \\ x \in E \end{cases}$$

où $x, c \in \mathbb{R}^n$ et E un ellipsoide défini par son centre a^0 , par r et la matrice symétrique définie positive A .

Algebriquement : $E = \{ x \in \mathbb{R}^n : (x - a^0)^t A(x - a^0) \leq r^2 \}$

Lemme 1 : La solution du problème (0) est donnée par :

$$X^* = a^0 - \frac{A^{-1}c\, r}{\sqrt{< c , A^{-1}c >}}$$

Preuve :

En posant $y = x - a^0$, le problème revient à résoudre :

$$(0')\ \begin{cases} \min\ c^t y \\ y^t A y \leq r^2 \end{cases}$$

En écrivant les conditions d'optimalité en programmation convexe, y^* est solution de (0') si et seulement si , il existe un réel $\lambda^* \geq 0$ tel que

$$\begin{cases} c + \lambda^* A y^* = 0 \\ \lambda^* (y^{*t} A y^* - r^2) = 0 \end{cases}$$

On en déduit : $\lambda^* > 0$, $y^* = (-1/\lambda^*)\, A^{-1}c$, $y^{*t} A y^* = r^2$ et donc

$$\frac{1}{\lambda^{*2}} c^t A^{-1} c = r^2$$

puis $y^* = - \dfrac{A^{-1}c\, r}{\sqrt{c^t A^{-1} c}} \Rightarrow x^* = a^0 - \dfrac{A^{-1}c\, r}{\sqrt{c^t A^{-1} c}}$ c.q.f.d

Cas particulier du lemme 1 :

Si l'ellipsoïde E est réduit à une sphère (A = I) alors la solution de (0) est

$$x^* = a^0 - \frac{c}{\|c\|} r$$

Autrement dit , il suffit de se déplacer (figure 1) à partir du centre a^0 dans la direction opposée du vecteur coût (i.e. - c) d'une distance égale au rayon (r) de la sphère .

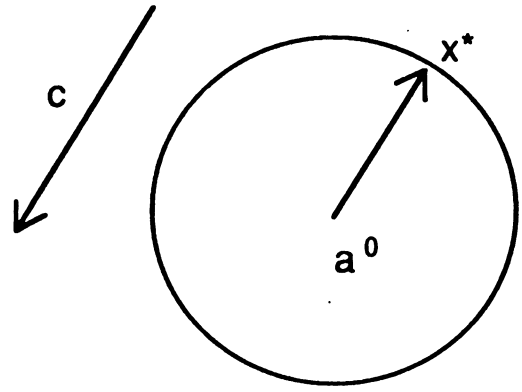


Figure I.1

I . 2 . 2 - L'idée générale de l'algorithme

Soit A un élément de $\mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ et $c \in \mathbb{R}^n$ et posons par définition

$\Omega = \{ x \in \mathbb{R}^n : Ax = b \}$

$P = \{x \in \mathbb{R}^n : Ax = b , x \geq 0\} = \Omega \cap \mathbb{R}^n_+$, où $\mathbb{R}^n_+$ désigne l'orthant positif .

Considérons alors le programme linéaire général :

$$(p.l.g) \begin{cases} \min c^t x \\ x \in P \end{cases}$$

Si l'on remplace $\mathbb{R}^n_+$ par une sphère (ou un ellipsoïde) $E_1 \subset \mathbb{R}^n_+$, on obtient le problème :

$$(1)\ \begin{cases} \min\ c^t x \\ x \in \Omega \cap E_1 \end{cases}$$

Or, l'intersection d'une sphère et d'un espace affine est une sphère de dimenson plus petite dans cet espace affine, donc $(\Omega \cap E_1 = E_2)$. Le problème devient :

$$(2)\ \begin{cases} \min\ c'^t x \\ x \in E_2 \end{cases} \quad , \quad c' \text{ désigne la projection orthogonale de } c \text{ sur } \Omega$$

D'après ce qui précède , le problème (2) est trivial : Il suffit de se déplacer à partir du centre de E_2 dans la direction (- c') d'une distance égale au rayon de E_2 .

Finalement , tout revient à remplacer P par un ellipsoïde (ou une sphère) E de centre a^0 ($a^0 \in P : a^0_i > 0$, i = 1,....., n) et minimiser $c^t x$ sur E au lieu de P. La question est donc de savoir quelle sera la qualité de la solution obtenue comparée à la solution réelle du problème .

Pour donner une réponse à la question on construit un deuxième ellipsoïde E' contenant P de même centre que E et de rayon égal à ν fois le rayon de E ($E \subset P \subset E'$) , ν étant un réel strictement plus grand que 1

Notons par $f_E = f(a^1)$, $f_P = f(x_P)$ et $f_{E'} = f(x_{E'})$ les minima de $f(x) = c^t x$ sur E , P et E' respectivement comme le montre la figure 2 .

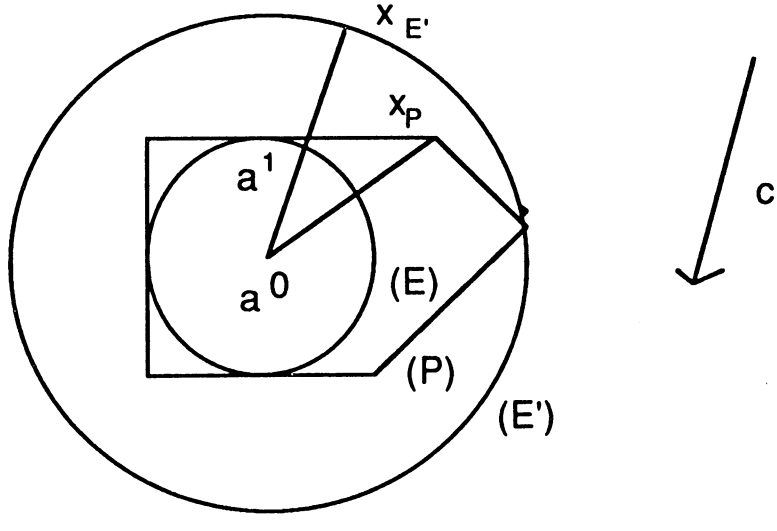


Figure I-2

On a alors le résultat suivant :

Lemme 2 : $0 \leq \dfrac{f_E - f_P}{f(a^0) - f_P} \leq 1 - \dfrac{1}{\nu}$

Preuve :

On a évidemment : $f_{E'} \leq f_P \leq f_E \Rightarrow f(a^0) - f_E \leq f(a^0) - f_P \leq f(a^0) - f_{E'}$. Comme f est linéaire f_E et $f_{E'}$ sont colinéaires et on a en particulier :

$f(a^0) - f_{E'} = \nu[f(a^0) - f_E]$ d'où

$$f(a^0) - f_P \leq \nu\,[f(a^0) - f_E] \Rightarrow \frac{f(a^0) - f_E}{f(a^0) - f_P} \geq \frac{1}{\nu} \Rightarrow 0 \leq \frac{f_E - f_P}{f(a^0) - f_P} \leq 1 - \frac{1}{\nu}$$, c.q.f.d

---

En partant donc de a^0 on arrive à un point a^1 minimisant f sur E ($f(a^1) = f_E$) et f_P est approchée par un facteur de $(1-1/\nu)$. On recommence l'opération en choisissant a^1 comme centre d'un ellipsoïde E_1 contenu dans P et d'un ellipsoïde $E'_1 = \nu_1 E_1$ contenant P . Ainsi :

$$0 \leq f_{E1} - f_P \leq (1 - 1/\nu_1)\,(1 - 1/\nu)\,[f(a^0) - f_P]$$

On peut encore réitérer avec un point a^2 centre d'un ellipsoïde E_2 et ainsi de suite ... Le but étant de borner $f_{Ei} - f_P$ par une valeur de plus en plus petite jusqu'à atteindre une précision voulue . Bien entendu la vitesse de convergence dépend de ν_i : Plus ce dernier est petit plus la convergence est rapide .

L'idée serait bonne si seulement on pouvait facilement construire les ellipsoïdes E_i et E'_i et si les quantités ν_i restent bornées . Malheureusement ce n'est pas le cas pour un polyèdre P quelconque, c'est pourquoi KARMARKAR introduit une transformation projective qui envoie P dans un n-simplexe de $\mathbb{R}^{n+1}$ de manière à obtenir les propriétés ci-dessus dans le problème transformé .

I . 2 . 3 - La transformation projective de KARMARKAR

Soit $a = (a_1 , a_2 , ... , a_n)^t$ un point strictement intérieur à P (i.e. tel que $a \in \Omega$ et $a_i > 0$) et posons $D = diag\{a\}$. Définissons alors le simplexe S_{n+1} de dimension n contenu dans $\mathbb{R}^{n+1}$:

$$S_{n+1} = \{ x \in \mathbb{R}^{n+1} , x \geq 0 \text{ et } e^t_{n+1} x = 1 \}$$

La transformation projective (notée T) est une fonction :
$\mathbb{R}^n_+$ ------> $\mathbb{R}^{n+1}_+$ définie par :

$$y = T(x) \quad \text{avec} \quad \begin{cases} y_i = \dfrac{x_i/a_i}{1 + \sum_{i=1}^{n} x_i/a_i} , & i = 1,....., n \\ \text{et} \\ y_{n+1} = 1 - \sum_{i=1}^{n} y_i \end{cases}$$

Il est facile de voir que : $y_i = \dfrac{x_i}{a_i} y_{n+1}$, $i = 1,....., n$

ou encore : $y_{[n]} = (D^{-1}x) y_{n+1}$, où $y_{[n]}$ désigne les n premières composantes de y . Ceci montre que T est bijective : En effet ,

$$T^{-1}(y) = x = \frac{D\, y_{[n]}}{y_{n+1}}$$

Evidemment : . $T(\mathbb{R}^n_+) = S_{n+1} \subset \mathbb{R}^{n+1}$.
. $T(a) = a^0 = (1/n+1)e_{n+1}$ (centre de S_{n+1})

Quelques Propriétés de T :

1- L'image d'un espace affine Ω par T est un espace affine Ω',donc puisque $a \in \Omega$, son image $a^0 \in \Omega'$.

2 - L'image d'une face ($x_i = 0$) de $\mathbb{R}^n_+$ est la face correspondante ($y_i = 0$, $i = 1,...n$) du simplexe S_{n+1} . Quant à la face $y_{n+1} = 0$ de S_{n+1} , elle est l'image des points à l'infini .

Il est facile de voir que la plus grande sphère inscrite dans S_{n+1} et la plus petite sphère contenant S_{n+1} centrées en a^0 sont respectivement :

$$B(a^0, r) = \{x \in \mathbb{R}^{n+1} : e_{n+1}{}^t x = 1, \| x - a^0 \| \leq r\}$$

$$B(a^0, R) = \{x \in \mathbb{R}^{n+1} : e_{n+1}{}^t x = 1, \| x - a^0 \| \leq R\}$$

Où $r = 1/\sqrt{n(n+1)}$ et $R = \sqrt{n}/(n+1)$ d'où $(R/r) = n$

La situation est donc la suivante : $B(a^0,r) \subset S_{n+1} \subset B(a^0,R)$, ce qui entraine $B(a_0,r) \cap \Omega' \subset S_{n+1} \cap \Omega' \subset B(a_0,R) \cap \Omega'$.

Or, $S_{n+1} \cap \Omega' = P' = T(P = \Omega \cap \mathbb{R}^n{}_+)$. De plus l'intersection d'une sphère B avec un espace affine Ω est une sphère B' de dimension plus petite et de même rayon que B si Ω contient le centre de B , c'est le cas ici puisque
$a^0 \in \Omega'$. Ainsi : $B'(a^0,r) \subset P' \subset B'(a^0,R)$, où $B'(a^0,r) = B(a^0,r) \cap \Omega'$ et $B'(a^0,R) = B(a^0,R) \cap \Omega'$. Ceci prouve que la minimisation sur la sphère inscrite dans la région admissible réduit la valeur de l'objectif d'au moins $(1 - 1/n)$.

I . 3 - DESCRIPTION DE LA METHODE

I . 3 . 1 - Le problème traité par KARMARKAR et les hypothèses de travail

On se place désormais dans $\mathbb{R}^n$: Le simplexe $S_n = \{ x \in \mathbb{R}^n : e_n^t x = 1 , x \geq 0 \} \subset \mathbb{R}^n$ est donc de dimension (n-1) .

La méthode projective de KARMARKAR résoud directement le programme linéaire réduit suivant :

$$(p.l.r) \begin{cases} \min\ c^t x \\ Ax = 0 \\ x \in S_n \end{cases}$$

en suposant que :

1- La valeur optimale est nulle , i.e : si x* est une solution optimale de (p.l.r) alors $c^t x^* = z^* = 0$.

2 - Le point $a^0 = (1/n)e_n$ (centre du simplexe S_n) est une solution réalisable de (p.l.r) .

3 - La matrice A est de plein rang : $rg(A) = m$.

Ces trois hypothèses seront appelées les conditions de KARMARKAR

On suppose également que $c^t a^0 > 0$, puisque si $c^t a^0 = 0$ on s'arrête immédiatement avec a^0 optimal . Ceci implique que $c^t x$ n'est pas constant sur la région admissible et par conséquent il est strictement positif pour tout x réalisable .

Remarques

a - Si la valeur optimale est connue à priori mais non nulle l'égalité $e_n^t x = 1$ permet de se ramener à un objectif nul . En effet , soit x une solution optimale du problème et c* la valeur optimale de l'objectif . Alors $c^t x = z^* = z^* e^t_n x \Rightarrow (c - z^* e_n)^t x = (c')^t x = 0$. On minimise alors l'objectif $(c')^t x$ au lieu de $c^t x$, où $c'_i = c_i - z^*$, $i = 1 \ldots n$.

b - Si le système de contraintes est de la forme $Ax = b$, $b \neq 0$, on se ramène facilement à un système homogène , il suffit d'écrire :

$$Ax = be_n^t x \quad \Rightarrow$$

$$\Rightarrow \quad (A - be_n^t)x = 0$$

Autrement dit on obtient un système de la forme : $A'x = 0$ où les élements de A' sont : $a'_{ij} = (a_{ij} - b_i)$, $\{ i = 1..... m , j = 1,....., n \}$.

c - Le problème (p.l.r) ainsi défini peut être vu comme un problème de faisabilité (voir G. DE GHELLINCK & J-PH. VIAL [17]) .
En effet , puisqu'on suppose la valeur optimale (z^*) nulle ou connue à priori , il s'agit alors :

de trouver $x \geq 0$ tel que : $\begin{cases} c^t x = z^* \\ Ax = 0 \\ e_n^t x = 1 \end{cases}$

I . 3 . 2 - L'Algorithme de base

Nous présentons dans ce paragraphe l'algorithme de base de KARMARKAR pour résoudre un programme linéaire du type (p.l.r) . Pour cela on se donne une précision ε (par exemple $\varepsilon = 2^{-q}$ où q est un entier , $q \geq 1$) .

Partant de la solution initiale $x^0 = a^0$, l'algorithme produit une suite de points intérieurs qui converge vers une solution optimale du problème en un temps polynomial .
Dans le but de ramener l'objectif $c^t x$ à zéro on le minimise localement sur une sphère inscrite dans la région admissible . A chaque itération (k) l'itéré ($x^k > 0$) est ramené au centre de S_n par la transformation projective T_k définie par :

$$T_k : x \in S_n \text{ ---------> } T_k(x) = y \in S_n \quad \text{avec}$$

$$T_k(x) = \frac{D_k^{-1} x}{e_n^t D_k^{-1} x} = y \quad ; \quad T_k^{-1}(y) = \frac{D_k y}{e_n^t D_k y} \quad , \quad D_k = \text{diag}\{x^k\}$$

et ainsi de suite jusqu'à ce que le test d'optimalité ($c^t x^k \leq \varepsilon$) soit vérifié .

Début algorithme

initialisation $x^0 = a^0 = (1/n)e_n \quad , \quad k = 0$

tant que $c^t x^k > \varepsilon$ **faire**

pas 0
$$\begin{cases} D_k = \text{diag}\{x^k\} \\ B_k = \begin{bmatrix} A_k \\ \cdots\cdots \\ e_n^t \end{bmatrix} \end{cases} \quad , \quad A_k = AD_k$$

pas 1 $$p_k = \{I - B_k^t(B_kB_k^t)^{-1}B_k\}D_kc = \{I - A_k^t(A_kA_k^t)^{-1}A_k - \frac{1}{n}e_ne_n^t\}D_kc$$

pas 2 $$d_k = \frac{p_k}{\| p_k \|}$$

pas 3 $$y^k = a^0 - \alpha r d_k \quad , \quad r = \frac{1}{\sqrt{n(n-1)}} \quad , \quad 0 < \alpha < 1$$

pas 4 $$x^{k+1} = \frac{D_k y^k}{e_n^t D_k y^k} = T_k^{-1}(y^k) \quad , \quad k = k + 1$$

fin tant que

fin algorithme

Dans le pas 0 , on ne fait que construire la matrice des contraintes (i.e B_k)

Le pas 1 consiste à projeter $c_k = D_kc$ sur le noyau de B_k (c'est l'opération la plus couteuse de l'algorithme) . La formule donnant p_k est obtenue par un calcul élémentaire en utilisant le fait que $A_ke_n = 0$.

Au pas 2 on calcule le vecteur normé d_k corréspondant à p_k .

Au pas 3 on choisit y^k à une distance αr de a^0 dans la direction $-d_k$ KARMARKAR choisit $\alpha = 1/4$.

Et en fin au pas 4 on effectue la transformation inverse T_k^{-1} pour calculer le nouvel itéré x^{k+1} .

I . 3 . 3 - Dérivation et analyse de l'algorithme

Le but de ce paragraphe est de montrer en quelque sorte comment est obtenu l'algorithme précédent .

On a vu que la transformation T_k applique le simplexe S_n dans lui même , en même temps l'itéré $x^k > 0$ (dont les composantes forment la matrice diagonale D_k) est envoyé au centre de S_n . Cependant le transformé du programme linéaire (p.l.r) est le programme suivant :

$$(p.n.l) \begin{cases} \min \dfrac{c^t D_k y}{e_n^t D_k y} \\ \dfrac{AD_k y}{e_n^t D_k y} = 0 \\ e_n^t y = 1 \ , \ y \geq 0 \end{cases}$$

Mais on a pour tout $y \in S_n$: $e_n^t D_k y = \sum_{i=1}^{n} x_i^k y_i \geq \min \{x_i^k : i = 1,....., n\} > 0$

Les égalités $\dfrac{c^t D_k y}{e_n^t D_k y} = 0$ et $\dfrac{AD_k y}{e_n^t D_k y} = 0$ sont donc satisfaites si et seulement si :

$c^t D_k y = 0$ et $AD_k y = 0$.

(p.n.l) est alors équivalent au programme linéaire :

$$(p.l) \begin{cases} \min c^t D_k y \\ AD_k y = 0 \\ e_n^t y = 1 \ , \ y \geq 0 \end{cases}$$

qui est de la forme (p.l.r) et vérifie les conditions de KARMARKAR .

Une remarque fondamentale :

Si on ajoute au problème (p.l) la contrainte $\{ y \in \mathbb{R}^n : \| a^0 - y \| \leq \alpha r \}$, où

$0 < \alpha < 1$ et $r = \dfrac{1}{\sqrt{n\,(n-1)}}$

i.e la sphère $B(a^0, \alpha r)$ de centre a^0 et de rayon αr, la contrainte de positivité ($y \geq 0$) devient redondante . Ce résultat est une conséquence du lemme général suivant :

Lemme 3 :

Si pour un programme linéaire donné on connait une solution réalisable y telle que ($y_i > 0$, $i = 1,\ldots..., n$) , alors l'ellipsoïde :

$$E = \{ x \in \mathbb{R}^n : \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{y_i^2} \leq \beta^2 \quad , \quad 0 < \beta < 1 \}$$

est à l'intérieur de l'orthant positif de $\mathbb{R}^n$.

Preuve :

Supposons au contraire qu'il existe $j \in \{ 1..... n \}$ tel que $x_j \leq 0$, alors :

$$\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{y_i^2} \geq \frac{(x_j - y_j)^2}{y_j^2} \geq 1 > \beta^2 \quad , \quad \text{c.q.f.d.}$$

Pour retrouver le résultat précédent il suffit de prendre $y = a^0$ et $\beta = \alpha r$.

Finalement , le problème (p.l) devient :

$$(\text{Paux}) \begin{cases} \min c^t D_k y \\ AD_k y = 0 \\ e_n^t y = 1 \\ \| y - a^0 \| \leq \alpha r \end{cases}$$

On vient de remplacer l'orthant positif par la sphère $B(a^0, \alpha r)$ qui est beaucoup plus simple à manier .

Théorème 1 : Le point y^k (du pas 3 de l'algorithme) est une solution optimale de (Paux) .

Démonstration :

Posons $x = y - a^0$, alors $B_k x = 0$, où B_k est la matrice des contraintes de (Paux) définie au pas 0 de l'algorithme . (Paux) est alors équivalent (à une translation près) au problème suivant :

$$(p^*) \begin{cases} \min c^t D_k x \\ B_k x = 0 \\ \| x \|^2 = \sum_{i=1}^{n} x_i^2 \leq (\alpha r)^2 \end{cases}$$

Or x^* est une solution optimale de (p^*) si et seulement si $\exists\, \lambda \in \mathbb{R}^{m+1}$ et un scalaire $\mu \geq 0$ tels que :

$$\mathbf{D_k c + B_k{}^t\lambda + \mu x^* = 0 \text{ -----> (i)}} \ .$$

En multipliant les deux membres de (i) par B_k on trouve :

$$\mathbf{B_k D_k c + B_k B_k{}^t\lambda + \mu B_k x^* = B_k D_k c + B_k B_k{}^t\lambda = 0} \quad \text{(puisque } B_k x = 0\text{)}$$

$$\Rightarrow \lambda = \mathbf{- (B_k B_k{}^t)^{-1}(B_k D_k c)}$$

d'où en substituant dans (i)

$$\mathbf{x^* = (-1/\mu)[I - B_k{}^t(B_k B_k{}^t)^{-1} B_k] D_k c = (-1/\mu) p_k}$$ (ce qui justifie le pas 1)

Mais x^* vérifie : $\| x^* \| = \frac{1}{\mu} \| p_k \| = \alpha r \Rightarrow -\frac{1}{\mu} = \frac{\alpha r}{\| p_k \|} \Rightarrow$

$$\Rightarrow \mathbf{x^* = - \alpha r \frac{p_k}{\| p_k \|} = - \alpha r d_k}$$

Soit finalement :

$$\mathbf{y^k = y^* = a^0 + x^* = a^0 - \alpha r d_k}$$ c.q.f.d

Propriété fondamentale de y^k :

Le point y^k est tel que :

$$\frac{c^t D_k y^k}{c^t D_k a^0} \leq 1 - \frac{\alpha}{n-1}$$

Preuve :

Cette propriété est démontrée dans [PADBERG [43] lemme 4] et même dans [KARMARKAR [37] théorème 3] . Cependant , on peut la démontrer facilement en appliquant le lemme 2 : Il suffit de considérer le problème (p.l) et prendre :

$$P = \{ y \in \mathbb{R}^n : AD_k y = 0 \ , \ e_n^t y = 1 \ , \ y \geq 0 \}$$

$$E = B(a^0 , \alpha r)$$

dans ce cas on a :

$$f_E = c^t D_k y^k \quad , \quad f_P = 0 \quad , \quad f(a^0) = c^t D_k a^0 \quad \text{et} \quad \nu = \frac{R}{r} = \frac{\sqrt{(n-1)/n}}{\frac{1}{\sqrt{n(n-1)}}} = (n-1)$$

Le résultat suit immédiatement . c.q.f.d

N.B : Cette propriété constitue pour nous un résultat très important que nous utiliserons dans le chapitre suivant pour modifier légèrement la *"sliding objecive méthod "* de KARMARKAR .

I . 3 . 4 - Fonction potentiel (étude de la convergence) :

Pour établir la convergence de l'algorithme, il faut montrer que :

$$\frac{c^t x^{k+1}}{c^t x^k} < q_0 \quad , \text{ où } \quad 0 < q_0 < 1 \quad \text{est indépendant de } k$$

Or, il est difficile de trouver directement q_0 . Pour surmonter cette difficulté, KARMARKAR associe à l'objectif linéaire $c^t x$ la fonction potentiel

$$f(x) = \sum_{i=1}^{n} \log\left(\frac{c^t x}{x_i}\right) \text{ , définie sur } \quad F = \{ x \in \mathbb{R}^n : x > 0 \ , \ Ax = 0 \ , \ e_n^t x = 1 \}$$

Le lemme suivant montre que la minimisation (réduction) de f(x) conduit droit à celle de $c^t x$.

Lemme 4 :

Soit x^k le $k^{ième}$ itéré de l'algorithme , alors :

$$\frac{c^t x^k}{c^t x^0} \leq (\exp[f(x^k)-f(x^0)])^{1/n} \quad , \quad \text{où} \quad x^0 = \frac{e_n}{n}$$

Preuve :

$$\exp [f(x^k) - f(x^0)] = \left(\frac{c^t x^k}{c^t x^0}\right)^n \prod_{i=1}^{n} \frac{x_i^0}{x_i^k} \quad \Rightarrow \frac{c^t x^k}{c^t x^0} = \left(\prod_{i=1}^{n} \frac{x_i^k}{x_i^0}\right)^{1/n} (\exp [f(x^k) - f(x^0)])^{1/n}$$

$$\text{Or , } \left(\prod_{i=1}^{n} \frac{x_i^k}{x_i^0}\right)^{1/n} = n\left(\prod_{i=1}^{n} x_i^k\right)^{1/n} \leq \frac{n \sum_{i=1}^{n} x_i^k}{n} = 1 \ , \ \text{(puisque } x^k \in S_n) \ , \text{ d'où le résultat .}$$

Conséquence immédiate :

Si la suite $f(x^k)$ tend vers $-\infty$ alors la suite $c^t x^k$ tend vers zéro , autrement dit pour diminuer $c^t x$ il suffit de diminuer suffisamment f(x) .

Propriétés de f(x) :

1- Une propriété cruciale de f(x) est qu'elle conserve toutes ses caractéristiques par T_k i.e : elle est transformée en une fonction de la même forme :

$$g(y) = \sum_{i=1}^{n} \log\left(\frac{c^t D_k y}{y_i}\right) - \sum_{i=1}^{n} \log\,(x_i^k) \quad , \quad \text{où} \quad y = T_k(x) \ .$$

g(y) est bien définie sur $F_k = \{ y \in \mathbb{R}^n : y > 0 \ , \ AD_k y = 0 \ , \ e_n{}^t y = 1\}$

2 - Soit $y = T(x)$ et $y^0 = T(x^0)$ alors on a :
$f(x) \leq f(x^0) - d \Leftrightarrow g(y) \leq g(y^0) - d$, d étant un réel positif . En d'autres termes toute réduction de f(x) entraîne la même réduction de g(y) et réciproquement .

3 - La fonction potentiel g(y) est associée à la fonction linéaire $c^t D_k y$ (objectif du problème (Paux)) .

4 - g(y) est une fonction monotone croissante de $c^t D_k y$.

Théorème de convergence de KARMARKAR

Si $0 < \alpha \leq 1/4$, alors en partant de $x^0 = e_n/n$ après **O(nq+nlogn)** itérations l'algorithme trouve un point réalisable x tel que:

(i) $c^t x = 0$

ou

(ii) $\dfrac{c^t x}{c^t x^0} \leq 2^{-q}$ où q est une précision fixée .

Un résultat plus précis est donné juste après par PADBERG [31] qui montre en utilisant la fonction potentiel :

$$h(x) = \frac{c^t x}{\left(\prod_{i=1}^{n} x_i\right)^{1/n}}$$

que la convergence est réalisée en O(nq) itérations pour : $0 < \alpha < 0.7968...$

Nous allons nous attarder quelque peu sur le choix du paramètre de convergence α dont dépend en partie la vitesse de convergence .

Choix de KARMARKAR :

KARMARKAR montre en fait que pour n grand ($n \mapsto +\infty$) la fonction potentiel f(x) diminue à chaque itération d'une quantité : $\delta(\alpha) = \alpha - \alpha^2/2 - \alpha^2/(1-\alpha)$, i.e $f(x^k) \leq f(x^{k-1}) - \delta(\alpha)$. Il suggère alors de prendre $\alpha = 1/4$, valeur qui correspond approximativement à la valeur maximale de $\delta(\alpha)$: $\max \{ \delta(\alpha) , 0 < \alpha < 1\} = 0.245122...$

Autre choix de α :

Nous allons montrer en s'inspirant de la démonstration de PADBERG que l'on peut très bien choisir $(\alpha > 1/4)$.

Considérons ainsi le théorème suivant :

Théorème 2 : Soit x^k le $k^{ième}$ itéré de l'algorithme , alors on a :

$$\frac{c^t x^k}{c^t x^0} \leq \left(\frac{\exp(-2\alpha)}{1-\alpha}\right)^{k/n}$$

Démonstration :

Montrons d'abord que $\frac{h(x^k)}{h(x^0)} \leq \left(\frac{\exp(-2\alpha)}{1-\alpha}\right)^{k/n}$ -----> (1)

Padberg montre que : $\frac{h(x^k)}{h(x^0)} \leq [g(\alpha , n)]^k$, où $g(\alpha , n)$ est définie par :

$$g(\alpha , n) = \frac{1 - \alpha/(n-1)}{1 + \alpha/(n-1)}\left[\frac{1 + \alpha/(n-1)}{1 - \alpha}\right]^{1/n}$$

Or g(α,n) peut être écrit sous la forme suivante :

$$g(\alpha , n) = \left(\frac{[1- \alpha/(n-1)]^{n-1}\,[1- \alpha/(n-1)]\,[1+ \alpha/(n-1)]}{[1+ \alpha/(n-1)]^{n-1}\,[1+ \alpha/(n-1)]\,[1- \alpha\,]}\right)^{1/n}$$

De plus $[1- \alpha/(n-1)] \leq 1$, il vient que :

$$g(\alpha , n) \leq \left(\frac{[1- \alpha/(n-1)]^{n-1}}{[1+ \alpha/(n-1)]^{n-1}[1- \alpha]}\right)^{1/n}$$

Or , pour $|x| < 1$ on a : $\log\left(\frac{1- x}{1+ x}\right) \leq -2x$, d'où pour $x = \frac{\alpha}{n-1}$

$$\log\left(\frac{1- \alpha/(n-1)}{1+ \alpha/(n-1)}\right) \leq -\frac{2\alpha}{n-1} \quad \Rightarrow \left(\frac{1- \alpha/(n-1)}{1+ \alpha/(n-1)}\right)^{n-1} \leq \exp(-2\alpha)$$

Soit finalement : $g(\alpha , n) \leq \left(\frac{\exp(-2\alpha)}{1- \alpha}\right)^{1/n}$

Cette dernière inégalité entraine (1) .

Il reste à montrer que : $\frac{c^t x^k}{c^t x^0} \leq \frac{h(x^k)}{h(x^0)}$ -----> (2)

En effet , $\frac{h(x^k)}{h(x^0)} = \frac{c^t x^k}{c^t x^0}\left(\prod_{i=1}^{n} \frac{x_i^0}{x_i^k}\right)^{1/n} = \frac{c^t x^k}{c^t x^0}\left(\prod_{i=1}^{n} \frac{1}{n x_i^k}\right)^{1/n} \geq \frac{c^t x^k}{c^t x^0}$

Puisque $\left(\prod_{i=1}^{n} x_i^k\right)^{1/n} \leq \frac{1}{n}\sum_{i=1}^{n} x_i^k = \frac{1}{n} \Rightarrow \left(\prod_{i=1}^{n} \frac{1}{n x_i^k}\right)^{1/n} \geq 1$, c.q.f.d

considérons alors la fonction : $t(\alpha) = \dfrac{\exp(-2\alpha)}{1-\alpha}$, où $\alpha \in (0\ ,\ 1)$

On peut montrer que : $\min \{ t(\alpha) : \alpha \in (0\ ,\ 1) \} = t(1/2) = 0.735 < 1$.
i.e : $(\alpha = 1/2)$ est un choix théoriquement plus convenable que celui de KARMARKAR .

Remarques :

1 - Du point de vue pratique pour $(\alpha = 1/2)$ l'algorithme est deux fois plus rapide que pour $(\alpha = 1/4)$.

2 - Le fait que $t(1/2)$ soit la plus petite valeur de $t(\alpha)$ pour $\alpha \in (0\ ,\ 1)$ ne signifie pas que c'est pour cette valeur que l'algorithme est plus rapide , d'ailleurs il converge plus vite pour $\alpha > 1/2$ (voir exemple I.1ci - dessous) .

Exemple I .1:

Considérons le programme linéaire suivant :

$$(I.1)\begin{cases} \min x_1 + x_2 \\ x_1 - x_2 = 0 \\ x_1 + x_2 + x_3 = 1 \\ x_i \geq 0\ ,\ i = 1,\ldots..,\ 3 \end{cases}$$

Ce programme satisfait toutes les conditions de KARMARKAR, en particulier sa valeur optimale est nulle .

Sa solution optimale exacte est : $\begin{cases} x_1^* = x_2^* = 0 \\ x_3^* = 1 \end{cases}$

Le tableau ci-dessous présente les résultats obtenus en appliquant l'algorithme de KARMARKAR au problème (I .1) pour différentes valeurs de α avec une précision de l'odre de 10^{-4}.

α	itérations	Temps (secondes)	valeur de l'objectif
0.25	23	0.6600	0.0005
0.50	12	0.3899	0.0004
0.70	9	0.2700	0.0003
0.90	7	0.2200	0.0003
0.99	6	0.1599	0.0005

Tableau I.1
Algorithme de KARMARKAR pour différentes valeurs de α

La solution trouvée est pratiquement la même pour toutes les valeurs de α

$x_1^* = x_2^* = 0.0002...$
$x_3^* = 0.999...$

I . 3 . 5 - Interprétation géométrique de la méthode

Contrairement à la méthode du simplexe qui -géométriquement- s'interprète comme un cheminement de sommet en sommet adjacent le long de la frontière de l'ensemble des solutions réalisables, la méthode de KARMARKAR est "*une méthode de point intérieur* "qui produit une suite de solutions réalisables (dans l'intérieur relatif de la région admissible) convergeant vers une solution optimale du problème traité .

Exemple I . 2

Prenons à titre illustratif l'exemple suivant :

$$\begin{cases} \min\ x + y \quad , \quad (x\,;y)^t \in \mathbb{R}^2 \\ 2x + y \leq 2 \\ -2x + 4y \leq 3 \\ x \geq 0\ ,\ y \geq 0 \end{cases}$$

La solution optimale étant le point $(0\,;0)^t$.

Le tableau I.2 présente les résultats obtenus par l'algorithme de KARMARKAR avec un pas constant ($\alpha = 0.99$) et une précision de l'ordre de 10^{-3} et la figure I.3 représente les 3 premières itérations .

itération	0	1	2	3	4	5	6	7
x	0.3454	0.2296	0.1503	0.0407	0.0137	0.0029	0.0011	0.0001
y	0.7224	0.4478	0.1313	0.0463	0.0112	0.0040	0.0007	0.0003
x + y	1.0679	0.6775	0.2817	0.0870	0.0249	0.0069	0.0019	0.0005

Tableau I . 2
Méthode de KARMARKAR à pas constant $\alpha = 0.99$

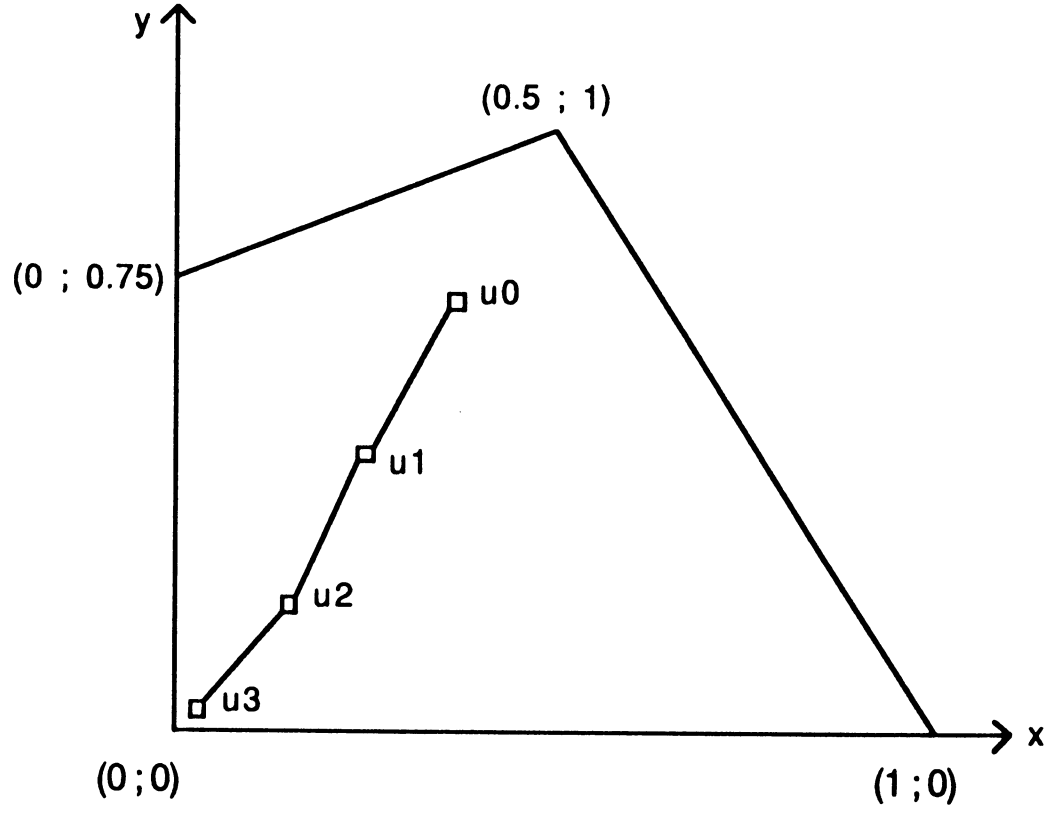


Figure I . 3
Représentation géométrique de la méthode de KARMARKAR

uk représente le point $(x ; y)^t$, pour $k = 0 , 1, 2 , 3$.

Conclusion :

La démonstration de PADBERG offre plus de liberté dans le choix de α, cependant sa fonction potentiel est - bien que très liée à celle de KARMARKAR $f(x) = n\log[h(x)]$ - inutilisable en pratique puisqu'il est très probable que son dénominateur tende très vite vers 0 .

Le résultat du théorème 2 explique en partie ce qui se passe actuellement en pratique : Des choix de $\alpha > 1/4$ et même $\alpha > 1$ ont permis l'accélération de la convergence de la méthode .

Il reste bien évidemment l'étude théorique du choix optimal de α . Il est peut être possible de prouver la convergence pour des valeurs de α plus cohérentes aux valeurs pratiques . Nous reviendrons sur ce point au chapitre III .

Une difficulté principale de l'algorithme de KARMARKAR est qu'on suppose la valeur optimale connue au départ . Condition très restrictive en pratique . Plusieurs procédés d'approximation de cette valeur seront présentés dans le chapitre suivant .

CHAPITRE 2

EXTENSION ET VARIANTES DE L'ALGORITHME DE KARMARKAR

II . 0 - INTRODUCTION :

La mise en œuvre de l'algorithme de KARMARKAR, son implémentation effective nécessite l'étude de ces deux problèmes :

(1) - La transformation du problème (p.l.g) (ci-dessous) à la forme réduite

$$(p.l.g)\begin{cases} \min\ c^t x = z^* \\ Ax = b \\ x \geq 0 \end{cases}$$

(2) - La modification de l'algorithme décrit dans le chapitre 1 en vue de son extension au cas où la valeur optimale z^* n'est pas connue .

II . 1 - RESOLUTION DE (p.l.g)

Pour résoudre (p.l.g) KARMARKAR suggère indirectement une méthode à deux phases qui consiste à ramener (p.l.g) à la forme (p.l.r) en appliquant la transformation projective T ce qui nécessite la connaissance d'une solution réalisable ou la résolution du problème de faisabilité (phase1) et passer ensuite à la minimisation (phase 2) .

II . 1 . 1 - Résolution du problème de faisabilité (Phase 1)

Un point réalisable de (p.l.g) est une solution du problème :

$$(P1)\begin{cases} Ax = b \\ x \geq 0 \end{cases}$$

Or, d'après KARMARKAR (P1) est équivalent au problème de minimisation :

$$(P2)\begin{cases} \min \lambda \\ Ax + \lambda(b - Ax^0) = b \\ x \geq 0\ ,\ \lambda \geq 0 \end{cases}$$

où $x^0 > 0$ est choisi arbitrairement dans l'orthant positif et λ une variable artificielle .

Notons que (P2) est toujours réalisable , il suffit de prendre : $x = x^0$ et $\lambda = 1$.

Soit λ_m la valeur minimale de l'objectif . Alors , $\lambda_m = 0$ corréspond à une solution réalisable de (P1) , cependant on peut se contenter d'une solution presque nulle . Plus précisément KARMARKAR démontre le théorème suivant :

Théorème : $\exists\ \varepsilon_0 > 0$ tel que : les deux propositions suivantes sont équivalentes :

1- (P1) est réalisable .

2 - (P2) admet une solution (x,λ) telle que $\lambda \leq \varepsilon_0$.

La résolution de (P1) se ramène donc à celle de (P2) , problème auquel on appliquera l'algorithme de base , puisque connaissant la valeur optimale et une solution réalisable de ce dernier il est facile de le mettre sous la forme (p.l.r) .

Remarque :

Le problème de faisabilité (P1) peut être résolu par une quelconque méthode à condition que la solution trouvée soit dans l'intérieur relatif du domaine de faisabilité $\{ x \geq 0 , Ax = b \}$. Par exemple la méthode du simplexe ne convient pas puisqu'elle donne une solution sur la frontière .

II . 1 . 2 - Problème de minimisation (Phase 2)

Soit a $(a_i > 0 \ , \ i = 1, ..., n)$ une solution réalisable de (p.l.g) obtenue (par exemple par la méthode précédente). En appliquant la transformation T le système **Ax = b** devient **A'y = 0** où A' est une mx(n+1) matrice définie par :

$$A' = [AD_a \quad -b] \quad , \quad D_a = \text{diag}\{a\} .$$

Quant à l'objectif il est transformé en une fonction non linéaire :

$$\frac{c^t D_a y[n]}{y_{n+1}}$$

où $y[n]$ est un vecteur de $\mathbb{R}^n$ formé des n premières composantes de $y = T(x) \in \mathbb{R}^{n+1}$.

On se retrouve alors avec le problème :

$$(p.l.f)\begin{cases} \min \dfrac{c^t D_a y[n]}{y_{n+1}} \\ A'y = 0 \\ e^t_{n+1} y = 1 \\ y \geq 0 \end{cases}$$

Or, si la valeur optimale z^* est connue, on peut écrire

$$\frac{c^t D_a y[n]}{y_{n+1}} = z^* \Rightarrow c^t D_a y[n] - z^* y_{n+1} = c'^t y = 0 \quad , \text{ où } \begin{cases} c'_i = c_i a_i \ , \ i = 1, \ldots, n \\ c'_{n+1} = -z^* \end{cases}$$

on obtient le programme linéaire réduit :

$$(p.l.r)\begin{cases} \min c'^t y \\ A'y = 0 \\ e^t_{n+1} y = 1 \\ y \geq 0 \end{cases}$$

Il est clair que toute solution réalisable de (p.l.g) est transformée par T en une solution réalisable de (p.l.r) et réciproquement, toute solution réalisable y de (p.l.r) avec $(y_{n+1} > 0)$ est transformée par T^{-1} en une solution réalisable x de (p.l.g). De plus, on peut montrer facilement que l'image par T d'une solution optimale finie $(x^* < +\infty)$ de (p.l.g) est une solution optimale y^* de (p.l.r) (voir LUSTIG [40]).

Notons en fin que si la valeur optimale est inconnue, le problème (p.l.f) ne peut pas être mis sous la forme (p.l.r). Il faudra avoir recours à d'autres techniques. ANSTREICHER [3] propose d'appliquer l'algorithme directement à la forme fractionnelle (p.l.f). Toutefois, il existe une technique (fréquemment rencontrée dans la littérature) qui permet d'écrire (p.l.f) sous la forme convenable sans connaître à priori sa valeur optimale.

II . 1. 3 - Transformation de (p.l.g) à la forme (p.l.r)

Nous présentons ici deux formulations diffèrentes tout en signalant qu'il en existent d'autres (voir par exemple : J. PH. VIAL [53]) .

Supposons alors qu'il existe un nombre réel $\sigma > 0$ assez grand pour que l'on ait $\sum_n x_i \leq \sigma$, $\forall\, x \in P = \{ x \geq 0 ,\ Ax = b \}$ ce qui revient à supposer P borné . Ajoutons une variable d'écart $(x_{n+1} \geq 0)$ telle que $\sum_{n+1} x_i = \sigma$ ou encore $\sum_{n+1}(x_i/\sigma) = 1$ et posons $y_i = x_i/\sigma$ d'où $x_i = \sigma y_i$, $i = 1,.....,n+1$.

L'objectif s'écrit : $c^t x = (\sigma c^t , 0)\, y$

Le système $Ax = b$ s'écrit :

$$[A \quad 0]\begin{bmatrix} x \\ x_{n+1} \end{bmatrix} = \sigma\, [A \quad 0]\, y = b$$

$$\Rightarrow \quad [A \quad 0]\, y = b/\sigma$$

Première formulation

Dans cette formulation (due essentiellement à KARMARKAR), l'équation $e_{n+1}{}^t y = 1$, permet d'écrire $[A \quad 0]\, y = (b/\sigma)\, e_{n+1}{}^t y$, ce qui donne $[A - (b/\sigma) e_n{}^t \quad - b/\sigma]\, y = 0$, d'où le programme linéaire :

$$(1) \begin{cases} \min\ (\sigma c^t , 0)\, y \\ B\, y = [A - (1/\sigma)\, b e_n^t \quad - b/\sigma]\, y = 0 \\ e_{n+1}^t y = 1 \ , \ y \geq 0 \end{cases}$$

Pour obtenir une solution réalisable de (1) on peut soit résoudre le problème de faisabilité $(By = 0 , y \geq 0)$, soit utiliser la technique suivante connue sous le nom de

Méthode de big M : On ajoute une colonne d à la matrice B et donc une variable $(y_{n+2} \geq 0)$ au vecteur y telles que le point $e_{n+2}/(n+2)$ soit solution du système $B\, y + d\, y_{n+2} = 0$ et $e_{n+1}{}^t y + y_{n+2} = 1$.

Il est facile de voir que $d = (n+1)b/\sigma - A e_n$. Le problème (1) s'écrit finalement :

$$(1.1)\quad \begin{cases} \min\ (\sigma c^t\ ,\ 0\ ,\ c_{n+2}) \begin{bmatrix} y \\ y_{n+2} \end{bmatrix} \\ [B \qquad (n+1)\ b/\sigma - Ae_n] \begin{bmatrix} y \\ y_{n+2} \end{bmatrix} = 0 \\ e_{n+1}^t y + y_{n+2} = 1\ , \quad y \geq 0\ ,\ y_{n+2} \geq 0 \end{cases}$$

c_{n+2} étant le coût associé à y_{n+2} . Si $(y\ ,\ y_{n+2})^t$ est une solution optimale de (1.1) avec $y_{n+2} = 0$ alors y est une solution optimale de (1) . Cette possibilité peut être réalisée en imposant à la composante c_{n+2} d'être assez grande . Dans le cas où y_{n+2} reste loin de zéro, le problème (1) est non réalisable .

Notons que la matrice de contraintes du programme (1.1) est généralement pleinne . Nous proposons ci-dessous une formulation similaire à celle de TOMLIN [52] et qui préserve le creux .

Deuxième formulation

Pour rendre le système $[A \quad 0]\ y = b/\sigma$ homogène, on introduit une variable artificielle (notée y_{n+2}) en lui attribuant la valeur 1. Les contraintes s'écrivent :

$$\begin{aligned} [A \quad 0]\ y - y_{n+2}(b/\sigma) &= 0 \\ y_{n+2} &= 1 \\ e_{n+1}^t y &= 1 \end{aligned}$$

En remplaçant les deux dernières contraintes par

$$\begin{aligned} e_{n+1}^t y - y_{n+2} &= 0 \\ e_{n+1}^t y + y_{n+2} &= 2 \end{aligned}$$

et en multipliant toutes les variables par (1/2) on obtient :

$$A'y' = \begin{bmatrix} A & 0 & -b/\sigma \\ e_n^t & 1 & -1 \end{bmatrix} y' = \begin{bmatrix} 0 \\ 0 \end{bmatrix}\ ,\quad y' \geq 0$$

$$e_{n+2}^t y' = 1$$

avec $y_i = 2y'_i\ ,\ i = 1,\ldots..,\ n+2 \Rightarrow\ x_i = 2\sigma y'_i\ ,\ i = 1,\ldots..,\ n$

Comme précédemment, on ajoute une colonne d' à A' et une variable y'_{n+3} à y' telles que :

$e_{n+3}/(n+3)$ soit solution du système $A'y' + d'y_{n+3} = 0$
et
$e_{n+2}{}^t y' + y'_{n+3} = 1$

on obtient le programme linéaire suivant :

$$(1.2)\quad \begin{cases} \min\ (2\sigma c^t\ ,\ 0\ ,\ 0\ ,\ c_{n+3}) \begin{bmatrix} y' \\ y'_{n+3} \end{bmatrix} \\ \begin{bmatrix} A & 0 & -b/\sigma & b/\sigma - Ae_n \\ e_n^t & 1 & -1 & -n \end{bmatrix} \begin{bmatrix} y' \\ y'_{n+3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ e_{n+2}^t y' + y'_{n+3} = 1 \quad , \quad y' \geq 0 \ , \ y'_{n+3} \geq 0 \end{cases}$$

Le coût c_{n+3} associé à y'_{n+3} est assez grand pour que y'_{n+3} soit proche de zéro .

Remarquons que le programme (1.2) a une contrainte et une variable de plus que (1.1) , cependant sa matrice de contraintes est moins pleinne . La deuxième formulation est donc plus acceptable . Pour résoudre l'un ou l'autre de ces deux programmes par la méthode de KARMARKAR, la connaissance de la valeur optimale est indisponsable . On verra par la suite comment contourner cette difficulté .

Conclusion :

Nous concluons ce paragraphe en signalant que sur le plan numérique la détermination d'une borne σ convenable n'est pas facile . De plus, d'une part on ajoute inutilement des lignes et des colonnes au programme linéaire et d'autre part, on risque d'avoir une instabilité numérique au cours de la résolution surtout dans le cas de gros exemples obligeant la borne σ à être très grande . On appliquera plutôt la transformation projective T .

II . 2 - MODIFICATIONS DE L'ALGORITHME DE BASE :

On s'intéresse toujours à la résolution du problème linéaire général

$$(p.l.g)\quad \begin{cases} \min c^t x = z^* \\ Ax = b \\ x \geq 0 \end{cases}$$

en supposant cette fois la valeur optimale z^* inconnue . KARMARKAR propose deux variantes :

a - La méthode " **primale-duale** " qui n'est autre que l'application de la phase2 de l'algorithme de base à un problème de faisabilité obtenu par la combinaison de (p.l.g) avec son dual .

b - La " **sliding objective function method** " qui consiste à localiser la valeur optimale dans un intervalle suffisamment petit .

Bien que ces deux approches préservent la polynomialité elles semblent inefficaces en pratique .

La première modification pratique de l'algorithme qui élimine l'hypothèse de connaître au départ la valeur optimale (en considérant des bornes inférieures construites à partir du problème dual) et fournit en même temps les solutions duales est proposée par TODD et BURRELL [48] . ANSTREICHER présente dans [3] une méthode similaire traitant un programme linéaire fractionnel vérifiant les conditions de KARMARKAR . En combinant ces deux approches , GOLDFARB et MEHROTRA développent une variante de l'algorithme de KARMARKAR pour résoudre un programme linéaire réduit à valeur optimale inconnue . Dans le même ordre d'idée GAY [20] , YE , M. KOJIMA [58] proposent indépendemment des variantes pour résoudre directement le problème linéaire général écrit sous sa forme standard . Nous montrerons à la fin de ce chapitre que la plupart de ces variantes peuvent être unifiées .

II . 2 . 1 - Méthode primale - duale

Considérons le dual de (p.l.g) :

$$(D)\ \begin{cases} \max b^t y \\ A^t y \le c \\ y \text{ quelconque dans } \mathbb{R}^m . \end{cases}$$

En combinant les deux problèmes et en utilisant le théorème de dualité de la programmation linéaire , le problème revient tout simplement à chercher une solution réalisable du problème "primal-dual" suivant :

$$\begin{cases} Ax = b \ , \ x \ge 0 \\ A^t y \le c \ , \ y \text{ quelconque} \\ c^t x - b^t y = 0 \end{cases}$$

En écrivant y sous forme d'une différence de deux vecteurs non négatifs et en ajoutant quelques variables d'écart , le système précédent prend la forme :

(1) $SX = b_s$, $X \ge 0$ avec

$$S = \begin{bmatrix} A & 0 & 0 & 0 \\ 0 & A^t & -A^t & I \\ c^t & -b^t & b^t & 0 \end{bmatrix} \quad , \quad b_s = \begin{bmatrix} b \\ c \\ 0 \end{bmatrix}$$

$S \in \mathbb{R}^{(m+n+1)x(2m+2n+1)}$, $b_s \in \mathbb{R}^{(m+n+1)}$, $X \in \mathbb{R}^{(2m+2n+1)}$.

Evidemment (1) est réalisable " si et seulement si " (p.l.g) admet une solution optimale finie . De plus, d'après les résultats du (§ II . 1.1) le problème (1) est équivalent au problème d'optimisation :

$$(2)\ \begin{cases} \min \lambda \\ SX + (b_s - SX_0)\lambda = b_s \\ \begin{pmatrix} X \\ \lambda \end{pmatrix} \ge 0 \end{cases}$$

$X_0 > 0$ est choisi arbitrairement, par exemple $X_0 = (1,1.....,1)^t \in \mathbb{R}^{(2m+2n+1)}$

Le point $a = (X_0, 1)^t$ est une solution réalisable évidente de (2). La transformation projective T_a nous permet de ramener (2) à la forme convenable (p.l.r) et appliquer alors l'algorithme de KARMARKAR pour trouver une solution (X , λ) telle que $\lambda \leq \varepsilon$, ε étant une précision donnée . X sera alors la solution du problème (1) qui comprend les solutions primales et duales du problème de départ .

Remarques :

Cette méthode est la plus simple à mettre en œuvre, on n'a besoin ni de résoudre le problème de faisabilité ni de connaître à priori la valeur optimale exacte , cependant on optimise un problème de **2(m+n+1)** variables et **(m+n+1)** contraintes , où **m** et **n** sont respectivement le nombre de contraintes et le nombre de variables du problème de départ

Exemple II . 2 . 1

$$(p.l.g) \begin{cases} \min c^t x \\ Ax = b \\ x \geq 0 \end{cases}$$

où

$$A = \begin{bmatrix} 2 & 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \quad , \quad b = \begin{bmatrix} 8 \\ 7 \\ 3 \end{bmatrix} \quad \text{et} \quad c = (-4 , -5 , 0 , 0 , 0)^t$$

En considérant le test d'arrêt ($c^t x^k \leq 10^{-5}$) , nous avons obtenu au bout de (k = 22) itérations les solutions suivantes :

Solution primale : $(2.999981; 1.999995 ; 0.000012 ; 0.000006 ; 0.999997)^t$

Solution duale : $(-0.999993 ; -1.999998 ; -0.000005)^t$

les solutions exactes étant : $x^* = (3 , 2 , 0 , 0 , 1)^t$, $y^* = (-1 , -2 , 0)^t$

II . 2 . 2 - Sliding objective function method

Cette méthode est une variante de l'algorithme de base, elle s'applique dans le cas où la valeur optimale de l'objectif n'est pas connue d'avance et consiste à localiser cette valeur dans un intervalle qu'on améliore au cours des itérations .

On s'arrêtera une fois l'intervalle obtenu est aussi petit que l'on veut : La borne supérieure du dernier intervalle obtenu sera *la valeur optimale approchée* et le point réalisable correspondant à cette borne sera la *solution optimale approchée* .

Nous allons décrire cette méthode d'une manière quelque peu différente de celle de KARMARKAR . Considérons pour cela le problème

$$(p.l.r)\begin{cases} \min c^t x \\ Ax = 0 \\ e_n^t x = 1 \ , \ x \geq 0 \end{cases}$$

Lemme II.2.1 Soit z^* la valeur optimale de (p.l.r) , alors

$$l_0 = \min \{c_i \ , \ i = 1, \ldots\ldots, n\} \leq z^*$$

Preuve

On a pour tout $x \in S_n = \{ x \geq 0 \ , e_n^t x = 1 \}$, $c^t x = \sum_n c_i x_i \geq l_0 \sum_n x_i = l_0$ d'autre part on a

$$\min \{c_i \ , \ i = 1, \ldots\ldots, n\} \leq \begin{cases} \min c^t x \\ e_n^t x = 1 \ , \ x \geq 0 \end{cases} \leq \begin{cases} \min c^t x \\ Ax = 0 \\ e_n^t x = 1 \ , \ x \geq 0 \end{cases}$$

d'où le résultat . c.q.f.d.

Description de la méthode :

Soit $[l \ , u]$ un intervalle contenant la valeur optimale z^* (supposée inconnue), l'algorithme n'étant applicable qu'à la connaissance de la valeur optimale de l'objectif l'idée mise en œuvre ici est de supposer cette valeur .

Au départ on prend $\mathbf{u = c^t a}$, où a est la solution réalisable initiale de (p.l.r) et $\mathbf{l = l_0}$.

Si $(u - l) \leq \varepsilon$ terminer le point a est une solution optimale et u une valeur optimale approchée .

Sinon $(u - l > \varepsilon)$ on teste la validité des bornes :

$$l' = l + (u - l)/3$$
$$u' = l + 2(u - l)/3 = u - (u - l)3 \quad .$$

En prétendant qu'à l'optimum l'objectif vaut l' et en appliquant une itération de l'algorithme de KARMARKAR avec l'objectif $c'^t x = c^t x - l'$ où $c' = c - l' e_n$, on obtient le point :

$$x = T_a^{-1}(y) = \frac{D_a y}{e_n^t D_a y} \quad , \quad \text{où} \quad D_a = \text{diag}\{a\} \quad \text{et} \quad y = a^0 - \alpha r d_a$$

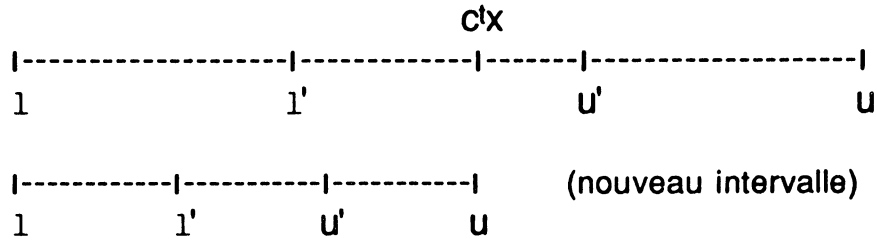
$a^0 = (e_n/n)$ étant le centre du simplexe S_n et d_a la projection de $c'_a = D_a c'$ sur le noyau de la matrice B_a où

$$B_a = \begin{bmatrix} AD_a \\ e_n^t \end{bmatrix}$$

Notons que $c'^t a = c^t a - l' = u - l' = 2(u - l)/3 > 0$.

Amélioration des bornes

Cas 1 si le point x obtenu précédemment est tel que **$c^t x < u'$**
alors u' est améliorable



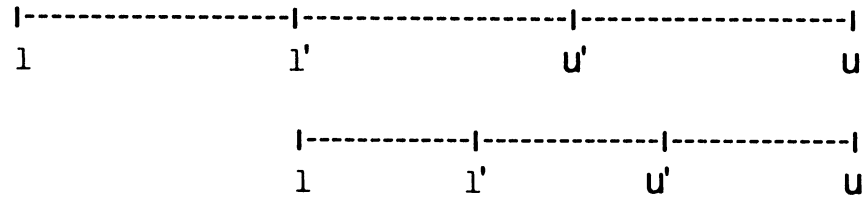
On pose $u = c^t x$ et on réitère avec $a = x$. L'intervalle se réduit ainsi d'au moins 1/3 :

En effet, $c^t x - l < u' - l = 2(u - l)/3 \Rightarrow u - c^t x > 1/3$.

Cas 2 $c^t x \geq u'$

Si $\mathbf{l' < z^*}$, l' est améliorable et on réitère avec

$$a = \begin{cases} x & \text{si } c^t x < c^t a \\ a & \text{sinon} \end{cases}$$



Là encore l'intervalle se réduit d'au moins 1/3 puisque $c^t a - l \geq u - l' = 2(u - l)/3$

Si $\mathbf{l' \geq z^*}$ alors $c^t x \leq c^t a$.

Dans ce cas on réitère avec $a = x$ et $u = c^t x$.

Reste à savoir sous quelle condition le cas 2 à t-il lieu . Le lemme suivant répond à la question :

Lemme II.2.2

$$\frac{c'^t_a y}{c'^t_a a^0} > 1 - \frac{\alpha}{n - 1} \qquad \Rightarrow l' < z^*$$

Preuve Supposons $l' \geq z^*$.

Alors $z' = \{ \min c'^t x : Ax = 0 \text{ et } x \in S_n \} = z^* - l' \leq 0$, et soit x' le point qui minimise $c'^t_a x$ sur l'ensemble $\{x : Ax = 0\} \cap S_n$:
i.e : $c'^t_a x' = \{ \min c'^t_a x : Ax = 0 \text{ et } x \in S_n \}$. Alors $z' \leq 0 \Rightarrow c'^t_a x' \leq 0$ et $c'^t a > 0 \Rightarrow c'^t_a a^0 > 0$ d'où $x' \neq a^0$.

Soit y' le point de rencontre du segment $a^0 x'$ avec la frontière de la sphère $B(a^0 , \alpha r)$. Il est facile de voir que :

$$y' = (1 - \lambda) a^0 + \lambda x' \quad \text{avec} \quad \lambda = \frac{\alpha r}{\| x' - a^0 \|} \geq \frac{\alpha r}{R} = \frac{\alpha}{n - 1}$$

$$\Rightarrow \frac{c'^{t}_{a}\, y'}{c'^{t}_{a}\, a^{0}} = (1-\lambda) + \lambda \frac{c'^{t}_{a}\, x'}{c'^{t}_{a}\, a^{0}} \leq 1 - \frac{\alpha}{n-1}$$

De plus $c'^{t}_{a} y' > 0$ puisque d'une part $y' \in B(a_0, \alpha r) \cap \{x : AD_a x = 0\} \cap S_n \Rightarrow c'^{t}_{a} y' \geq c'^{t}_{a} y$ et d'autre part $c'^{t} x \geq u' - l' > 0 \Rightarrow c'^{t}_{a} y > 0$. Ce qui implique :

$$\frac{c'^{t}_{a}\, y}{c'^{t}_{a}\, a^{0}} = \frac{c'^{t}_{a}\, y}{c'^{t}_{a}\, y'} \frac{c'^{t}_{a}\, y'}{c'^{t}_{a}\, a^{0}} \leq 1 - \frac{\alpha}{n-1}$$

Nous venons de prouver que $z^* \leq l' \Rightarrow \frac{c'^{t}_{a}\, y}{c'^{t}_{a}\, a^{0}} \leq 1 - \frac{\alpha}{n-1}$

Le lemme est alors démontré .

Exemple numérique :

Reprenons l'exemple II . 2 .1 avec le teste d'arrêt $(u - l \leq 10^{-3})$. Les résultats sont donnés dans le tableau (II.2.1) .

itération	borne inférieure	borne supérieure
0	- 35.0000	- 16.7125
5	- 23.1732	- 21.8396
10	- 22.1031	- 21.9933
15	- 22.0077	- 21.9933
20	- 22.0008	- 21.9999

Tableau II.2.1
Sliding objective function method avec $\alpha = 0.99$

Solution trouvée (au bout de 20 itérations)

x^{20} = (2.999992 ; 1.999994 ; 0.000018 ; 0.000017 ; 0.000005)

II . 2 . 3 - L'extension de TODD et BURRELL

TODD et BURRELL proposent dans [48] un algorithme " primal-dual " pour résoudre le programme linéaire réduit :

$$(p.l.r) \begin{cases} \min c^t x \\ Ax = 0 \\ x \in S_n \end{cases}$$

Cet algorithme produit en particulier deux suites de solutions réalisables (primales et duales) telles que les valeurs objectives correspondantes convergent vers la valeur optimale z^* (supposée inconnue) de (p.l.r) .

La difficulté de cette approche est que pour traiter un programme linéaire général, on doit d'abord l'écrire sous la forme (p.l.r). La valeur optimale étant inconnue, TODD et BURRELL suggèrent un procédé du même type que la technique décrite au paragraphe (II.1.3) dont on connait les inconvénients .

Inspiré des traveaux de TODD et BURRELL , ANSTREICHER [3] et YE et KOJIMA [58], nous présentons dans ce paragraphe un algorithme qui élimine cette difficulté en combinant cette approche avec la transformation projective de KARMARKAR permettant ainsi de résoudre directement un programme linéaire général .

Un algorithme pour la forme standard

Considérons le programme linéaire général :

$$(p.l.g) \begin{cases} \min c^t x \\ Ax = b \ , \ x \geq 0 \end{cases}$$

pour lequel on suppose la région admissible $\{ x \geq 0 : Ax = b \}$ bornée et contenant au moins un point $x^0 > 0$ et soit

$$(D) \begin{cases} \max b^t y \\ A^t y \leq c \ , \ y \in \mathbb{R}^m \end{cases}$$

le dual de (p.l.g) au sens de la programmation linéaire .

La transformation projective n'étant applicable qu'à la connaissance de la valeur optimale, l'idée mise en œuvre ici est de remplacer (à chaque itération) cette valeur par une borne inférieure . Or d'après le théorème de dualité on a :

$b^t y \leq z^* \leq c^t x \quad \forall$ x réalisable de (p.l.g) et $\forall$ y réalisable de (D) . Autrement dit, le programme dual (D) constitue une source de bornes inférieures de la valeur optimale .

Description de l'algorithme :

Au début de chaque itération ($k \geq 0$) on connait une solution réalisable primale ($x^k > 0$) , une solution réalisable de (D) ($y^k \in \mathbb{R}^m$) et une borne inférieure $z^k \leq z^*$. Nous allons montrer comment on détermine z^{k+1} , y^{k+1} et x^{k+1}. Considérons pour cela une borne inférieure quelconque z de z^* et appliquons la transformation projective

$$T_k(x \in \mathbb{R}^n_+) = x' \in S_{n+1}$$

en remplaçant z^* par z . Associons alors à (p.l.g) le programme linéaire réduit obtenu :

$$p(z) \begin{cases} \min \{c^t D_k x'[n] - z x'_{n+1}\} \\ [AD_k \quad - b]\, x' = 0 \\ x' \in S_{n+1} \end{cases}$$

Rappelons que si x est un point réalisable de (p.l.g) alors $T_k(x) = x'$ est un point réalisable de p(z) , et réciproquement, si x' est un point réalisable de p(z) avec ($x'_{n+1} > 0$) alors $x = T_k^{-1}(x') = (D_k x'[n]/x'_{n+1})$ est un point réalisable de (p.l.g) .

Ecrivons l'objectif de p(z) sous la forme :

$$c^t D_k x'[n] - z x'_{n+1} = [(c^t D_k , 0) - z(0 , 1)]x'$$

et posons

$$u = P_{A_k}\begin{pmatrix} D_k c \\ 0 \end{pmatrix} \quad , \quad v = P_{A_k}\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Où $A_k = [AD_k \quad - b]$ et P_{A_k} représente la projection sur le noyau de A_k

$A_k x' = 0 \Rightarrow [(c^t D_k , 0) - z(0 , 1)]x' = (u - zv)^t x'$

On peut donc écrire

$$p(z) \begin{cases} \min (u - zv)^t x' \\ A_k x' = 0 \\ x' \in S_{n+1} \end{cases}$$

Definissons $\theta(z) = \min \{u_i - zv_i : 1 \le i \le n + 1\}$. Comme le signale ANSTREICHER dans [3] $\theta(z) = \min \{(u - zv)^t x' / x' \in S_{n+1} \}$, donc en particulier, le problème définissant $\theta(z^*)$ est une relaxation de $p(z^*)$ d'où $\theta(z^*) \le 0$.

Il est donc évident que si $\theta(z) > 0$ on peut trouver un réel z' tel que $z < z' \le z^*$ et $\theta(z') = 0$.

Le lemme suivant montre que la condition $\theta(z) > 0$ joue un rôle essentiel dans cet algorithme : Elle permet d'ajuster la borne inférieure et la variable duale :

Lemme 2.3.1

Posons $c(z) = \begin{pmatrix} D_k c \\ - z \end{pmatrix}$ et $y(z) = (A_k A_k^t)^{-1} A_k c(z)$

alors

$$\theta(z) > 0 \Leftrightarrow \begin{cases} A^t y(z) < c \\ \text{et} \\ z < b^t y(z) \end{cases}$$

Preuve

Il suffit de remarquer que d'une part

$$u - zv = c(z) - A_k^t y(z) = \begin{pmatrix} D_k(c - A^t y(z)) \\ b^t y(z) - z \end{pmatrix}$$

et d'autre part $\theta(z) > 0 \Leftrightarrow u - zv > 0$ c.q.f.d

Ajustement de z^k et y^k

On procède de la manière suivante :

Si $\theta(z) \leq 0$ on prend $z^{k+1} = z^k$ et $y^{k+1} = y^k$

Si $\theta(z) > 0$ on détermine $z^{k+1} > z^k$ tel que $\theta(z^{k+1}) = 0$ et on prend

$y^{k+1} = y(z^{k+1})$. En pratique on peut prendre (voir GONZAGA [31])

$$z^{k+1} = \min \{ \frac{u_i}{v_i} \ , \ v_i > 0 \ , \ i = 1,....., n+1 \}$$

Pour calculer l'itéré suivant (x^{k+1}) on détermine d'abord la direction de déplacement :

$$d_k = P_{B_k} \begin{pmatrix} D_k c \\ - z^{k+1} \end{pmatrix} \quad , \quad B_k = \begin{bmatrix} A_k \\ e^t_{n+1} \end{bmatrix}$$

puis le point

$$x'^{(k+1)} = \frac{e_{n+1}}{n+1} - \alpha \frac{d_k}{\| d_k \|}$$

et enfin $x^{k+1} = T_k^{-1}(x'^{(k+1)})$

Lemme 2.3.2

$$d_k = u - z^{k+1} v - \frac{c^t x^k - z^{k+1}}{n+1} e_{n+1}$$

Preuve

(A_k est de plein rang et $A_k e_{n+1} = 0$) $\Rightarrow$ B_k est de plein rang et

$$P_{B_k} = P_{A_k} P_{e^t_{n+1}} = P_{e^t_{n+1}} P_{A_k}$$

où $P_{e^t_{n+1}} = \{ I - \frac{e_{n+1} e^t_{n+1}}{n+1} \}$ désigne la projection sur $\{ x : e^t_{n+1} x = 0 \}$

$$\Rightarrow P_{B_k}\begin{pmatrix} D_k c \\ -z^{k+1} \end{pmatrix} = P_{A_k}\begin{pmatrix} D_k c \\ -z^{k+1} \end{pmatrix} - \frac{e_{n+1} e_{n+1}^t}{n+1}\begin{pmatrix} D_k c \\ -z^{k+1} \end{pmatrix}$$

Or, d'après ce qui précède on a :

$$P_{A_k}\begin{pmatrix} D_k \\ -z^{k+1} \end{pmatrix} = u - z^{k+1} v$$

d'autre part, $e_{n+1}^t \begin{pmatrix} D_k c \\ -z^{k+1} \end{pmatrix} = e_n^t D_k c - z^{k+1} = c^t x^k - z^{k+1}$

d'où finalement :

$$d_k = u - z^{k+1} v - \frac{c^t x^k - z^{k+1}}{n+1} e_{n+1} \quad , \quad \text{c.q.f.d}$$

Etant donné une précision $(\varepsilon > 0)$, un point réalisable $x^0 > 0$ et une borne inférieure $z^0 \leq z^*$, l'algorithme peut être décrit de la manière suivante :

Algorithme primal - dual pour la forme standard

Début

Initialisation $k = 0$, $z^0 \leq z^*$, $x^0 > 0$, y^0 non spécifié .

Tant que $c^t x^k - z^k > \varepsilon$ **faire**

1 Ajustement de z^k et y^k

$D_k = \text{diag}\{x^k\}$, $A_k = [AD_k \quad -b]$

$$u = P_{A_k}\begin{pmatrix} D_k c \\ 0 \end{pmatrix} \quad , \quad v = P_{A_k}\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$\theta(z^k) = \{\min (u_i - z^k v_i) \quad , \quad 1 \leq i \leq n+1\}$

Si $\theta(z^k) \leq 0$ alors $z^{k+1} = z^k$ et $y^{k+1} = y^k$ et aller en 2

Sinon déterminer $z^{k+1} > z^k$ tel que $\theta(z^{k+1}) = 0$

et prendre

$$y^{k+1} = (A_k A_k^t)^{-1} A_k \begin{pmatrix} D_k c \\ - z^{k+1} \end{pmatrix}$$

2 Calcul de la direction de déplacement

$$d_k = u - z^{k+1} v - \frac{c^t x^k - z^{k+1}}{n+1} e_{n+1}$$

3 Calcul de l'itéré suivant

$$x'^{(k+1)} = \frac{e_{n+1}}{n+1} - \alpha \frac{d_k}{\| d_k \|}$$

$$x^{k+1} = T_k^{-1}(x'^{(k+1)}) = \frac{D_k x'^{(k+1)}[n]}{x'^{(k+1)}_{n+1}} \quad , \quad k = k + 1$$

Fin tant que

Fin

Remarques

1 - L'algorithme précédent génère trois suites réelles : $\{z^k\}$, $\{b^t y^k\}$ et $\{c^t x^k\}$. Les deux premières sont monotones par construction mais elles ne changent pas à chaque itération . La dernière par contre change à chaque itération mais non nécessairement d'une façon monotone .

2 - Si $z^0 = z^*$ alors $z^k = z^*$ pour tout $(k \geq 0)$ et la suite des solutions duales est vide . Ceci correspond en fait à l'algorithme originel de KARMARKAR .

Appliquons maintenant l'algorithme sur l'exemple de KLEE-MINTY

$$\begin{cases} \max \sum_{p=1}^{n} \varepsilon^{n-p} x_p \quad , \qquad 0 < \varepsilon < \dfrac{1}{2} \\ 2 \sum_{p=1}^{q-1} \varepsilon^{q-p} x_p + x_q \leq 1 \quad (q = 1, \ldots\ldots, n) \\ x_p \geq 0 \quad (p = 1, \ldots\ldots, n) \end{cases}$$

La solution optimale de ce problème étant $x_p = 0$ $(p = 1, \ldots\ldots, n - 1)$ et $x_n = 1$. La valeur optimale pour notre cas est égale à - 1 (on minimise) . Le critère d'arrêt est le suivant :

$$\frac{| c^t x^k - z^k |}{1 + | c^t x^k |} \leq 10^{-6}$$

itération	borne inférieure	objectif primal	objectif dual
1	- 1.973266782	- 0.855532358	- 1.795775028
2	- 1.155423948	- 0.956646511	- 1.121107746
3	- 1.020456258	- 0.986305393	- 1.013975508
4	- 1.008638808	- 0.996168361	- 1.006382771
5	- 1.000025742	- 0.999265899	- 1.000017477
6	- 1.000001940	- 0.999904500	- 1.000001133
7	- 1.000000436	- 0.999974085	- 1.000000312
8	- 1.000000002	- 0.999995642	- 1.000000001
9	- 1.000000000	- 0.999999730	- 1.000000000

Tableau 1 **Exemple KLEE-MINTY ($n = 4$, $\varepsilon = 0.4$ et $z^0 = -10^{-30}$) valeurs des objectifs primal et dual**

itération	solution primale	solution duale
1	0.1358673716 0.2074058433 0.2423774365 0.7145673784	-0.2391637469 -0.2500637754 -0.3065227659 -1.0000000000
2	0.1314835025 0.1803431067 0.0024595456 0.9183928520	-0.0554012398 -0.0445928223 -0.0000000000 -1.0211136842
3	0.1062873991 0.0017562726 0.0030380387 0.9780067811	-0.0058997156 -0.0000000000 -0.0026658427 -1.0054099502
4	0.0010136248 0.0024663577 0.0036041210 0.9942367071	-0.0000000000 -0.0019832133 -0.0021753475 -1.0022242100
5	0.0009835188 0.0019666288 0.0008550069 0.9985462904	-0.0000081175 -0.0000072873 -0.0000020723 -1.0000000000
6	0.0008057658 0.0001714267 0.0000065364 0.9998228882	-0.0000006252 -0.0000001397 -0.0000000000 -1.0000001176
7	0.0000064207 0.0000811686 0.0000062225 0.9999581989	-0.0000000000 -0.0000000868 -0.0000001076 -1.0000001176
8	0.0000062196 0.0000006331 0.0000050396 0.9999931274	-0.0000000010 -0.0000000000 -0.0000000001 -1.0000000005
9	0.0000050853 0.0000006037 0.0000003850 0.9999991546	-0.0000000000 -0.0000000000 -0.0000000000 -1.0000000000

Tableau 2 **Exemple de KLEE-MINTY solutions primales et duales**

Comme le montrent les résultats ci-dessus, les solutions réalisables duales convergent généralement plus vite et d'une manière plus stable que les solutions réalisables primales . De plus le fait de choisir la borne initiale très loin de la valeur optimale exacte n'influe pas sur la vitesse de convergence .

Convergence de l'algorithme (Y.YE et M.KOJIMA)

Pour prouver la polynomialité de l'algorithme, on associe au problème (p.l.g) la fonction potentiel

$$f(x , z) = (n+1)\log(c^t x - z) - \sum_{i=1}^{n} \log(x_i)$$

définie pour $(x > 0)$ réalisable et $z \leq z^* < c^t x$.

Y.YE et M.KOJIMA montrent que l'on a :

$$f(x^{k+1} , z^{k+1}) \leq f(x^k , z^k) - \delta \quad , \quad k = 1 , 2 , \ldots \quad \text{et} \quad \delta = \alpha - \frac{\alpha^2}{2 (1 - \alpha)^2}$$

On peut vérifier que $0.27 \leq \alpha \leq 0.36 \Rightarrow \delta \geq 0.2$.

Le théorème suivant est une conséquence du résultat précédent :

Théorème

$$\frac{c^t x^k - z^k}{c^t x^0 - z^0} \leq M \exp\left(\frac{-k\delta}{n+1}\right) \quad , \quad k = 1 , 2 , \ldots$$

où M désigne un certain nombre positif fini .

Si de plus $z^0 < z^* - \varepsilon$, alors il existe un entier positif K tel que y^k est réalisable pour le dual (D) et

$$\frac{c^t x^k - b^t y^k}{c^t x^0 - z^0} \leq M \exp\left(\frac{-k\delta}{n+1}\right) \quad , \quad \forall k \geq K$$

II . 2 . 4 - L'algorithme de YE - LUSTIG

Nous terminons ce chapitre par un algorithme présenté par LUSTIG dans [39] et dû essentiellement à YE .

Rappelons pour cela que si z est une approximation quelconque de z* alors, d'après les résultats du lemme 2.3.2 on peut écrire :

$$d_k(z) = P_{A_k}\begin{pmatrix} D_k c \\ -z \end{pmatrix} - \frac{c^t x^k - z}{n+1} e_{n+1} = \begin{pmatrix} D_k(c - A^t \pi^k) \\ b^t \pi^k - z \end{pmatrix} - \frac{c^t x^k - z}{n+1} e_{n+1}$$

où $\quad \pi^k = (A_k A_k^t)^{-1} A_k \begin{pmatrix} D_k c \\ -z \end{pmatrix}$

L'expression de $d_k(z)$ est d'une grande importance, elle permet d'unifier plusieurs stratégies dans les différentes modifications de l'algorithme de KARMARKAR . En effet, dans l'algorirhme précédent, on avait à chaque itération $z = z^k \leq z^*$, i.e une borne inférieure de la valeur optimale . Si $z = z^*$, on retrouve l'algorithme de base de KARMARKAR . Dans cet algorithme on prend à chaque itération $z = c^t x^k$ i.e une borne supérieure de z^* , d'où

$$d_k(z) = d_k(c^t x^k) = d_k = \begin{pmatrix} D_k(c - A^t \pi^k) \\ b^t \pi^k - c^t x^k \end{pmatrix}$$

En utilisant le fait qu'à l'optimum $\|d_k\|$ converge vers zéro LUSTIG a proposé le critère de convergence raisonnable suivant :

$$\frac{\| d_k \|}{| c^t x^0 |} < \varepsilon_0 \quad , \quad \varepsilon_0 \text{ est une précision donnée}$$

Notons que π^k n'est pas une solution réalisable du problème dual puisque $(c - A^t \pi^k)$ n'est pas toujours positif . On peut montrer cependant que si le problème primal (p.l.g) admet une solution optimale non dégénérée alors, π^k converge vers la solution optimale duale de (D) . Ceci est également valable pour l'algorithme de KARMARKAR (cas où $z^k = z^*$ pour tout $k \geq 0$) .

L'algorithme peut s'écrire sous la forme suivante :

Début

Initialisation $k = 0$, $x^0 > 0$.

Tant que $\dfrac{\| d_k \|}{| c^t x^0 |} > \varepsilon_0$ **faire**

$$D_k = \text{diag}\{x^k\} \quad , \quad A_k = [AD_k \quad -b]$$

$$d_k = P_{A_k}\begin{pmatrix} D_k c \\ -c^t x^k \end{pmatrix}$$

$$x'^{(k+1)} = \frac{e_{n+1}}{n+1} - \alpha r \frac{d_k}{\| d_k \|}$$

$$x^{k+1} = T_k^{-1}(x'^{(k+1)}) = \frac{D_k x'^{(k+1)}[n]}{x'^{(k+1)}_{n+1}} \quad , \quad k = k + 1$$

Fin tant que

Fin

En pratique, le meilleur des algorithmes que nous avons étudiés dans ce chapitre est celui de YE - LUSTIG . En théorie, cet algorithme converge (D'après YE), mais il n'est polynomial que si la solution réalisable initiale est raisonnablement proche de la solution optimale . Il serait peut être intéressant en pratique de combiner cet algorithme avec le précédent : On demarre avec des bornes inférieures et on continue avec des bornes supérieures dès que l'on s'approche suffisamment de l'optimum (pour l'exemple de KLEE-MINTY l'algorithme combiné donne des résultats un peu plus précis) . Les solutions primales trouvées au bout de 9 itérations avec $\varepsilon_0 = 10^{-6}$, sont :

algorithme ci-dessus $x^9 = (0.00000001 , 0.00000005 , 0.00000001 , 0.99999997)^t$
$c^t x^9 = -0.99999993$

algorithme combiné $x^9 = (0.00000001 , 0.00000001 , 0.00000001 , 1.00000000)^t$
$c^t x^9 = -1.00000003$

CHAPITRE 3

AMELIORATION DES PERFORMANCES DE L'ALGORITHME DE KARMARKAR

III . 0 . Introduction :

Dans ce chapitre, on s'intéresse aux problèmes d'ordre numérique rencontrés dans l'algorithme de KARMARKAR et ses variantes : En particulier le meilleur choix possible du pas de déplacement et le calcul économique de la direction de déplacement correspondante .

III . 1 . Détermination d'un pas de déplacement efficace

Dans le chapitre 1, nous avons constaté que le pas de déplacement α et le nombre d'itérations sont inversement proportionnels : *" plus α est grand plus le nombre d'itérations est petit "* . C'est pourquoi il est recommandé de choisir les pas les plus grands dans l'intérieur du simplexe . De plus , il est important de signaler qu'à l'exception de préserver la polynomialité , il n'y a pas une bonne raison pour se limiter à des valeurs de α plus petites que 1. En effet, des pas de déplacement plus grands peuvent diminuer sensiblement le nombre d'itérations et améliorer l'efficacité pratique de l'algorithme . Nous présentons ici deux méthodes différentes pour calculer de tels pas :

III . 1 . 1 . Méthode 1

C'est une méthode simple qui consiste à se déplacer le plus loin possible à partir du centre du simplexe ($a^0 = e_n$) dans la direction $-d_k$ tout en restant dans l'intérieur de ce simplexe . Autrement dit, en considérant le point : $y(\alpha) = y^{k+1} = (e_n/n) - \alpha r d_k$ du pas 3 de l'algorithme de base, on doit prendre le plus grand α possible tel que $y(\alpha) > 0$. En particulier, $0 \leq \alpha < 1$ vérifie cette condition (puisque $|d_k^i| \leq 1$, $i = 1,....., n$) .

Posons $I = \{1,....., n\}$ et $I^+ = \{i \in I : d_k^i > 0\}$, et supposons $d_k \neq 0$ (puisque si d_k est nul, le point x^k est optimal) . Alors l'ensemble I^+ est non vide : Il suffit de remarquer que $e_n^t d_k = 0$.

On a $y(\alpha) > 0 \Leftrightarrow (1/n) - \alpha r d_k^i > 0$, $\forall i \in I$, d'où
$\alpha < \min \{ 1/n r d_k^i , i \in I^+ \} = \alpha_{max}$.

Evidemment , tout $\alpha < \alpha_{max}$, garantira la faisabilité mais pas forcément la réduction de l'objectif d'une itération à l'autre . C'est pourquoi nous avons imposé une deuxième condition sur α donnant des résultats meilleurs en pratique :

Il s'agit de trouver une condition sur α garantissant la monotonie ($c^t x^{k+1} < c^t x^k$) .

En prenant toujours $D_k = \text{diag}\{x^k\}$ et en tenant compte de l'équation du simplexe ($e_n^t x^k = 1$) on peut facilement montrer que l'on a :

$$x^{k+1} = \frac{D_k y(\alpha)}{e_n^t D_k y(\alpha)} = \frac{x^k - \alpha n r D_k d_k}{1 - \alpha n r e_n^t D_k d_k} = x^k + \alpha\gamma\,(\nu\, x^k - n r D_k d_k)$$

où $\nu = n r e_n^t D_k d_k$ et $\gamma = 1/(1 - \alpha\nu)$

$$\Rightarrow \quad c^t x^{k+1} = c^t x^k + \alpha\gamma(\nu c^t x^k - \mu) \quad , \quad \mu = n r c^t x^k D_k d_k$$

Lemme 3.1 $c^t x^{k+1} < c^t x^k \Leftrightarrow \alpha(\mu - \nu c^t x^k) > 0$

Preuve

Il suffit de remarquer que $y(\alpha) > 0 \Rightarrow (1-\alpha\nu) > 0$, le résultat suit immédiatement c.q.f.d

Ainsi en tenant compte des deux conditions (faisabilité et monotonie), le pas de déplacement est déterminé à chaque itération par :

$$\begin{cases} 0 < \beta < 1 \\ \alpha_k = \beta\alpha_{max} \qquad \text{si} \quad (\mu - \nu c^t x^k) > 0 \\ \alpha_k = -\beta\alpha_{max} \quad (\text{ou } -\beta) \qquad \text{si} \quad (\mu - \nu c^t x^k) \le 0 \end{cases}$$

En fait α_k négatif correspond au cas où $-d_k$ n'est pas une direction de descente, on se déplace alors suivant la direction opposée ($+d_k$) .

Dans les tableaux ci-dessous on compare les résultats obtenus en appliquant l'algorithme de KARMARKAR avec un pas constant (tableau III-1) à ceux obtenus par le même algorithme (tableau III-2) en prenant le pas (variable) défini par la procédure précédente avec $\beta = 0.99$. La valeur optimale est connue au départ, les exemples sont testés sur (Olivetti : pc M 380) avec le test d'optimalité ($c^t x^k / c^t x^0 \le 10^{-6}$) .

Exemple III-1

$$\min\ -18x_1 + 7x_2 - 12x_3 - 5x_4 \qquad - 8x_6 \qquad = -8$$

$$\begin{aligned}
2x_1 - 6x_2 + 2x_3 + 7x_4 + 3x_5 + 8x_6 + x_7 &= 1 \\
-3x_1 - x_2 + 4x_3 - 3x_4 + x_5 + 2x_6 + x_8 &= -2 \\
8x_1 - 3x_2 + 5x_3 - 2x_4 + 2x_6 + x_9 &= 4 \\
4x_1 + 8x_3 + 7x_4 - x_5 + 3x_6 + x_{10} &= 1 \\
5x_1 + 2x_2 - 3x_3 + 6x_4 - 2x_5 - x_6 + x_{11} &= 5
\end{aligned}$$

$$x_i \geq 0\ ,\ i = 1,\ldots..,\ 11$$

Exemple III-2

$$\min\ 2x_1 - x_2 - 3x_3 + 5x_4 - 2x_5 + 4x_7 + x_8 + 2x_9 - x_{10} + x_{11} - x_{12} + 2x_{14} = -13.25$$

$$\begin{aligned}
x_1 + x_2 + x_{15} &= 8 \\
2x_2 + x_3 + x_{16} &= 4 \\
3x_3 - x_4 + x_{17} &= 6 \\
2x_4 + 4x_5 + x_{18} &= 2 \\
6x_5 + 2x_6 + x_{19} &= 5 \\
-x_6 + 2x_7 + x_{20} &= 1 \\
4x_7 - x_8 + x_{21} &= 2 \\
3x_9 + x_{10} + x_{23} &= 6 \\
3x_9 - x_{10} + x_{11} + x_{23} &= 3 \\
x_{10} + x_{11} + 2x_{12} + x_{24} &= 9 \\
x_{12} + 2x_{13} + x_{14} + x_{25} &= 4
\end{aligned}$$

$$x_i \geq 0\ ,\ i = 1,\ldots..,\ 25$$

	Objectif	Itération	Temps (s)
Exemple 1	-7.999997	23	3.629999
Exemple 2	-13.249964	33	28.059999

Tableau III-1

Méthode de KARMARKAR à pas constant : $\alpha = 0.99$

	Objectif	Itération	Temps (s)	Pas de déplacement
Exemple 1	-7.999991	9	1.43	1.4087 ≤ pas ≤ 1.9542
Exemple 2	-13.249994	12	10.33	1.6415 ≤ pas ≤ 2.6620

Tableau III.2
Méthode de KARMARKAR à pas variable

En comparant les deux tableaux on voit que l'algorithme à pas variable (qui est significativement superieur à 1) est nettement plus efficace : Il est pratiquement 3 fois plus rapide que celui à pas constant .

III . 1 . 2 - Méthode 2 (recherche linéaire)

TODD et BURRELL et beaucoup d'autres suggèrent d'effectuer à chaque itération (k) une recherche linéaire suivant la direction de descente $-d_k$ minimisant approximativement la fonction potentiel .

Considérons alors la fonction $\phi(\alpha) = g(y(\alpha))$, où $g(.)$ désigne la fonction potentiel de KARMARKAR dans l'éspace transformé .

$$\phi(\alpha) = n\log (c^t x^k - \alpha n r c^t D_k d_k) - \sum_{i=1}^{n} \log (1 - \alpha n r d_k^i)$$

définie pour $\alpha < \alpha_{max}$

$$\phi(0) = n\log (c^t x^k) = g(e_n)$$

$$\phi'(\alpha) = - \frac{n^2 r c^t D_k d_k}{c^t x^k - \alpha n r c^t D_k d_k} + \sum_{i=1}^{n} \frac{n r d^i}{1 - \alpha n r d^i}$$

En tenant compte du fait que $e_n^t d_k^i = 0$ on a :

$$\phi'(0) = - \frac{n^2 r c^t D_k d_k}{c^t x^k}$$

Il s'agit donc de trouver un réel α^* solution du problème :

$$\min \{ \phi(\alpha) : 0 < \alpha < \alpha_{max} \} .$$

Nous avons choisi pour cela la règle de GOLDSTEIN dont la procédure de base est la suivante :

Recherche linéaire économique (M.MINOUX [41])

(0) - Choisir deux coefficients réels $0 < m_1 < m_2 < 1$

(1) - Definir $\alpha_{min} = 0$ et α_{max} comme précédemment et prendre $\alpha = \alpha_0$
$\alpha_{min} \leq \alpha_0 < \alpha_{max}$

(2) - Si $\phi(\alpha) \leq \phi(0) + m_1\phi'(0)$ aller en (3)
Sinon poser $\alpha_{max} = \alpha$ et aller en (4)

(3) - Si $\phi(\alpha) \geq \phi(0) + m_2\phi'(0)$ terminer
Sinon poser $\alpha_{min} = \alpha$

(4) - Rechercher une nouvelle valeur de α dans l'intervalle $]\alpha_{min} \quad \alpha_{max}[$
(par exemple : $\alpha = (\alpha_{min} + \alpha_{max})/2$) et retourner en (2) .

Remarquons que la condition (2) est un critère de décroissance pour la fonction potentiel . Dans le contexte usuel de la programmation non linéaire ce critère empêche le pas α d'être excessivement grand . La condition (3) évite en revanche de choisir α trop petit . En ce qui nous concerne, le pas trouvé par cette procédure est souvent proche de α_{max} .

A priori, on ne sait pas à quel point cette recherche linéaire est bénifique vis à vis de la méthode 1 qui semble bien marcher en pratique et moins laborieuse .

	Itérations	Objectif	Temps (s)	Pas de déplacement
Méthode 1	12	- 13.24999	10.33000	1.6415 ≤ pas ≤ 2.6620
Méthode 2	11	- 13.24998	16.39999	1.5994 ≤ pas ≤ 2.7320

Tableau III-3
Comparaison entre la méthode 1 et la méthode 2 (exemple III-2)

III . 2 . Calcul de la projection

Dans les algorithmes de point intérieur du type KARMARKAR (proposés récemment), le coût d'une itération est dominé par le calcul de la direction de déplacement $-d_k$ où

$$d_k = P_k D_k c - c^t x^k / n \quad , \quad \text{avec} \quad A_k = AD_k \quad , \quad D_k = \text{diag}\{x^k\} \quad , \quad A \in \mathbb{R}^{m \times n} \text{ et}$$

$$P_k = \{ I - A_k^t (A_k A_k^t)^{-1} A_k \} D_k c$$

On peut calculer P_k de plusieur façons, en particulier en résolvant le problème de moindre carré suivant :

$$(M.C) \qquad \text{Min} \{ \| D_k c - A_k^t u \| \ , \ u \in \mathbb{R}^m \}$$

dont la solution satisfait les équations normales :

$$(1) \quad A_k A_k^t \, u = AD_k^2 A^t u = AD_k^2 c$$

P_k est alors le résidu de (M.C) : $P_k = D_k c - A_k^t u$

Le succès de l'implémentation de l'algorithme de KARMARKAR dépend donc de l'efficacité du procédé de résolution du système linéaire (1) .

Les méthodes proposées jusqu'à présent sont de deux types :

(1) - Les méthodes directes dans lesquelles on utilise la factorisation de la matrice du système (par exemple la méthode d'élimination de GAUSS) .

(2) - Les méthodes itératives qui, produisent une suite de solutions approchées du système et n'utilisent en général que des multiplications du type matrice-vecteur telles que la méthode de JACOBY, GAUSS-SEIDEL et la méthode du gradient conjugué . Ces méthodes sont attractives en vertu du peu d'espace mémoire qu'elles demandent . Leur convergence est par contre lente à moins qu'une technique de préconditionnement est appliquée .

A vrai dire, ces deux approches ne sont pas nécessairement différentes, il existe des implémentations efficaces de méthodes itératives basées typiquement sur l'algorithme du gradient conjugué, avec une technique de préconditionnement similaire à l'élimination de GAUSS .

Dans la plupart des implémentations de cette procédure, la méthode du gradient conjugué est utilisée pour résoudre le système linéaire résultant qui est généralement d'une structure favorable .

On ne peut pas savoir a priori quelle est la meilleure approche . En effet, le choix d'une méthode dépend de plusieurs facteurs :
De la dimension du système, la structure de la matrice (creux, conditionnement, etc...), et même de l'architecture de la machine utilisée .

Ceci étant, le système (1) possède deux caractéristiques importantes :

(1) - $A_k A_k^t$ est symétrique définie positive : Il suffit de voir que pour $x \neq 0$ $< A_k A_k^t x , x> = <A_k^t x , A_k^t x> = ||A_k^t x||^2$.

La méthode fréquemment utilisée dans ce cas est la factorisation de CHOLESKY qui est un cas particulier de la méthode d'élimination de GAUSS . De même, la méthode du gradient conjugué préconditionné est aujourd'hui l'une des meilleures méthodes pour résoudre ce type de systèmes .

(2) - Au cours de l'exécution de l'algorithme seuls les éléments de la matrice D_k peuvent changer d'une itération à l'autre . KARMARKAR a exploité cet avantage afin d'améliorer la compléxité totale de son algorithme . Nous parlerons ultérieurement de cette modification .

II . 2 . 1 - Factorisation de CHOLESKY

On appelle factorisation (ou décomposition) de CHOLESKY, la représentation d'une matrice symétrique définie positive $M \in \mathbb{R}^{m \times m}$ sous la forme : $M = LL^t$ où L est une matrice triangulaire inférieure de $\mathbb{R}^{m \times m}$

Exemple :

$$\text{Pour} \quad M = \begin{bmatrix} 2 & -2 \\ -2 & 5 \end{bmatrix} \quad \text{on a} \quad L = \begin{bmatrix} \sqrt{2} & 0 \\ -\sqrt{2} & \sqrt{3} \end{bmatrix}$$

Le calcul de la matrice L peut se faire par identification des coefficients dans l'équation :

$$M = LL^t$$

i.e par les formules suivantes :

$$M_{i,j} = \sum_{k=1}^{j} L_{i,k} \cdot L_{j,k} \qquad \text{pour} \quad i \geq j$$

compte tenu de la forme triangulaire de L qui entraîne $L_{j,k} = 0$ pour $k > j$.

Il existe plusieurs façons de programmer cette factorisation qui ne diffèrent que par l'ordre dans lequel on effectue les opérations, sachant que dans tous les cas ce sont les mêmes opérations qui sont efféctuées. On peut par exemple choisir l'ordre ligne par ligne, ou colonne par colonne. L'efficacité des différents programmes dépend du mode de rangement des coefficients de la matrice dans la mémoire de l'ordinateur .

L'algorithme (version colonne)

Ayant calculé les (j - 1) premières colonnes, on calcule la $j^{ème}$ en écrivant :

$$L_{j,j} = \sqrt{M_{j,j} - \sum_{k=1}^{j-1} L_{j,k}^2}$$

$$L_{i,j} = (M_{i,j} - \sum_{k=1}^{j-1} L_{i,k} \cdot L_{j,k})/L_{j,j} \qquad \text{pour} \quad i \geq j+1$$

Cet algorithme nécessite $0(n^3/6)$ opérations élémentaires (aditions, multiplications, etc ...)

Ecrivons le programme traduisant les formules ci-dessus en rangeant $L_{i,j}$ à la place de $M_{i,j}$.

PROGRAMME CHOLESKY

```
POUR    j = 1 à m FAIRE
        POUR    i = j à m       FAIRE
            POUR    k = 1 à (j - 1)         FAIRE
                    M[i , j] = M[i , j] - M[i , k] * M[j , k]
            FIN de boucle k
            SI  (i = j) ALORS  M[i , i] = SQRT(M[i , i])
                       SINON   M[i , j] = M[i , j]/M[j , j]
            FIN SI
        FIN de boucle i
FIN de boucle j
```

Résolution de systèmes linéaires triangulaires

1 - Système triangulaire inférieur :

Considérons le système linéaire $Ly = b$, où L est une matrice triangulaire inférieure de $\mathbb{R}^{mxm}$ et b un élément de $\mathbb{R}^m$.

Ce système s'écrit encore (compte tenu de $L_{i,j} = 0$ pour $j > i$)

$$\sum_{j=1}^{i-1} L_{i,j} y_j + y_i L_{i,i} = b_i \quad , \quad i = 1 \text{ à } m \quad \Rightarrow$$

$$y_i = (b_i - \sum_{j=1}^{i-1} L_{i,j} y_j)/L_{i,i} \quad , \quad i = 1 \text{ à } m$$

Puisque b_i ne sert plus après le calcul de y_i , on peut ranger y_i à la place de b_i . On obtient le programme suivant :

PROGRAMME 1

```
POUR  i = 1  à  m  FAIRE
    POUR  j = 1  à  (i - 1)      FAIRE
            b[ i ] = b[ i ] - L[i , j] * b[ j ]
    FIN de boucle j
    b[ i ] = b[ i ]/L[i , i]
FIN de boucle i
```

2 - Système linéaire supérieur

On peut donner un algorithme analogue pour un système linéaire triangulaire supérieur :

$Ux = y$

PROGRAMME 2

```
POUR  i = m  à  1  FAIRE
    POUR  j = (i + 1)  à  m     FAIRE
            x[ i ] = x[ i ] - U[i , j] * x[ j ]
    FIN de boucle j
    x[ i ] = x[ i ]/U[i , i]
FIN de boucle i
```

Application au système (1)

On factorise la matrice du système (programme CHOLESKY) :

$A_k A_k^t = R_k R_k^t$ (2)

où R_k est une matrice triangulaire inférieure de $\mathbb{R}^{mxm}$.

En utilisant (2) on a : $(R_k R_k^t)u = AD_k^2 c$ (3)

En posant $y = R_k^t u$, u peut être déterminé en résolvant d'abord le système linéaire triangulaire inférieur (algorithme 1) :

$R_k y = AD_k^2 c$ (4)

puis le système linéaire triangulaire supérieur (algorithme 2) :

$R_k^t u = y$ (5)

III . 2 . 2 - L'algorithme modifié de KARMARKAR

Dans le but de réduire le nombre d'opérations par itération, KARMARKAR propose de modifier le système (1) en approximant la matrice D_k par une matrice diagonale D'_k vérifiant la relation :

$$\frac{1}{\rho} \leq \left(\frac{D_k'^{(ii)}}{D_k^{ii}} \right)^2 \leq \rho \quad , \quad i = 1, 2, \ldots, n \quad \text{et} \quad \rho > 1 \qquad (2.2)$$

$\rho = 1$ corréspond à l'algorithme originel . KARMARKAR prend $\rho = 2$, SHANNOS par contre prend $\rho > 2$.

Soit E_k la matrice diagonale vérifiant $D'_k = D_k E_k$ avec évidemment $(E^k_{ii})^2 \in [1/2 \quad 2]$ et posons $Q_k = E_k^{-2}$.

On peut voir (§ I.3.3 : démonstration du théorème 1) que dans l'algorithme de base, la direction d_k est solution du problème

$$(P_k) \begin{cases} \min \ c^t D_k y \\ AD_k y = 0 \\ e_n^t y = 0 \\ \| y \| \leq \alpha r \end{cases}$$

Dans l'algorithme modifié, on remplace dans (P_k) la sphère $\{y \in \mathbb{R}^m : \| y \| \leq \alpha r\}$ par l'ellipsoïde $\{y : y^t Q_k^{-1} y \leq (\alpha r/2)^2$. La solution notée d'_k est alors à une constante près :

$$d'_k = [\, I - Q_k^{-1} B_k^t (B_k Q_k^{-1} B_k^t)^{-1} B_k \,] Q_k^{-1} D_k c$$

où $B_k^t = [D_k A^t \quad e_n]$

$$B_k Q_k^{-1} B_k = \begin{bmatrix} A D_k Q_k^{-1} D_k A^t & A D_k Q_k^{-1} e_n \\ (A D_k Q_k^{-1} e_n)^t & e_n^t Q_k^{-1} e_n \end{bmatrix}$$

où $A D_k Q_k^{-1} D_k A^t = A D_k'^2 A^t$.

KARMARKAR souligne que si $(A D_k'^2 A^t)^{-1}$ est connue, $(B_k Q_k^{-1} B_k^t)^{-1}$ se calcule en $O(m^2)$ opérations via la formule :

$$\begin{bmatrix} M & a \\ a^t & \lambda \end{bmatrix}^{-1} = \frac{1}{\lambda - a^t M^{-1} a} \begin{bmatrix} (\lambda - a^t M^{-1} a) M^{-1} + (M^{-1} a)(M^{-1} a)^t & -M^{-1} a \\ -(M^{-1} a)^t & 1 \end{bmatrix}$$

où $M \in \mathbb{R}^{m \times m}$, $a \in \mathbb{R}^m$ et $\lambda \in \mathbb{R}$.

Ajustement de D'_k :

Après avoir calculé x^{k+1} on définit :

$$\sigma^k = \frac{1}{n} \sum_{i=1}^{n} \frac{x_i^{k+1}}{x_i^k}$$

Multiplions alors D'_k par σ^k (D'_k <----- $\sigma^k D'_k$). Si un élément $(\sigma^k D'_k)_{ii}$ de $\sigma^k D'_k$ ne vérifie pas la relation (2 . 2), on le remplace par $x^{k+1}{}_i$.

En résumé, la matrice D'_{k+1} est définie par le procédé suivant :

0 - $D'_0 = D_0 = (1/n)I$

$$\textbf{1 -} \quad D'^{(ii)}_{k+1} = \begin{cases} \sigma^k D'^{(ii)}_k & \text{si} \quad \frac{1}{2} \leq \left(\frac{\sigma^k D'^{(ii)}_k}{x_i^{k+1}} \right)^2 \leq 2 \\ x_i^{k+1} & \text{sinon} \end{cases}$$

On peut donc écrire :

$$AD'^2_{k+1}A^t = AD'^2_kA^t + \sum_{j=1}^{p} [\, (D'^{(ii)}_{k+1})^2 - (D'^{(ii)}_k)^2 \,]\, A_jA_j^t \,]$$

où A_j désigne la $j^{ème}$ colonne de A et $p \leq m$.

Autrement dit, la matrice $AD_{k+1}'^2A^t$ est une série de modifications de rang 1 de $AD_k'^2A^t$. SHANNOS avait remarqué que le nombre nécessaire de modifications est très petit, ce qui rend cette approche plus favorable et conduit à une réduction significative de la complexité de l'algorithme .

Soit M une matrice symétrique définie positive (de dimension m dont on connait la décomposition de CHOLESKY $M = LL^t$), $u \in \mathbb{R}^m$ et μ un réel négatif .

Posons $M' = M + \mu uu^t$ et cherchons la décomposition de CHOLESKY de $M' = L'L'^t$.

Les algorithmes qui donnent L' en modifiant L sont très connus, SHANNOS utilisait une simplification de l'algorithme de FLETCHER et POWELL [18] que nous reprenons ici :

Les éléments de L' seront stockés dans L

PROGRAMME 3 (s , d , t , b : réels)

POUR $j = 1$ à m FAIRE

$s = u_j / L_{jj}$

$d = 1 + \mu s^2$

$t = \sqrt{d}$

Si $j = m$ STOP

$b = \mu s / d$

POUR $i = j + 1$ à m FAIRE

$u_i = u_i - s L_{ij}$

$L_{ij} = t(L_{ij} + b u_i)$

FIN de boucle i

FIN de boucle j

Un algorithme plus compliqué peut être utilisé dans le cas où μ est positif , voir [18] .

L'inconvénient de la modification de KARMARKAR est qu'elle oblige à prendre des pas petits ($\alpha < 1$) à chaque itération, or nous avons vu que l'algorithme originel est considérablement accéléré en prenant des pas plus grands que 1 . ANSTREICHER a montré dans [6] que sur le plan théorique, on peut résoudre ce problème en procédant avec une recherche linéaire basée sur la règle de GOLDSTEIN - ARMIJO . Sur le plan numérique, des résultats préliminaires sont donnés par CHANNOS pour ($\alpha > 1$) .

III . 2 . 3 - Méthode du gradient conjugué préconditionné

1- Rappels

Définition1 Soit $\| \cdot \|$ une norme matricielle, le conditionnement d'une matrice régulière M , associé à cette norme, est le nombre

$cond(M) = \|M\| \|M^{-1}\|$, parfois noté $k(M)$.

Définition2 On dira qu'une matrice est "*bien conditionnée* " si son conditionnement n'est pas beaucoup plus grand que 1 .

2 - Principe de la méthode

La rapidité de la méthode du gradient conjugué (pour résoudre un système linéaire symétrique défini positif : $Mx = b$) dépend du conditionnement de M : Plus $k(M)$ est proche de 1, plus vite convergera l'algorithme .

Le principe de préconditionnement d'une matrice M consiste à trouver une matrice non-singulière symétrique C telle que $k(C^{-1}M)$ soit beaucoup plus petit que $k(M)$.

Théoriquement, le meilleur choix est donc $C^{-1} = M^{-1}$ car alors $k(C^{-1}M) = 1$.

En pratique, il faudra trouver C^{-1} le plus proche possible de M-1, sans que les calculs pour C^{-1} soient trop coûteux .

3 - L'algorithme

On ne peut appliquer l'algorithme du gradient conjugué à $C^{-1}M$, car même si C^{-1} est symétrique, il n'en est pas de même pour $C^{-1}M$.

Donc au lieu de considérer le système $C^{-1}Mx = C^{-1}b$
on considère le système :

$$C^{-1}MC^{-1}Cx = C^{-1}b$$

Posons $y = Cx$. Le problème est alors trouver y tel que :

$(C^{-1}MC^{-1})\, y = C^{-1}b$

On peut alors appliquer la méthode du gradient conjugué à la nouvelle matrice $C^{-1}MC^{-1}$ qui est symétrique définie positive .

Soit $\varepsilon > 0$ une précision donnée . L'algorithme du gradient conjugué préconditionné : Apliqué à M avec un préconditionnement C^{-1} s'écrit :

$x^0 = 0$

$r^0 = b$

$Cp^0 = r^0$

$z^0 = p^0$

POUR $k = 0,1,...$ FAIRE

Si $||r^k||_2 > \varepsilon$ FAIRE

$\alpha_k = <r^k, z^k>/<Mp^k, p^k>$

$x^{k+1} = x^k + \alpha_k p^k$

$r^{k+1} = r^k - \alpha_k Mp^k$

$Cz^{k+1} = r^{k+1}$

$\beta_{k+1} = <r^{k+1}, z^{k+1}>/<r^k, z^k>$

$p^{k+1} = r^{k+1} + \beta_{k+1} p^k$

Notons que si $C = I$ on retrouve l'algorithme du gradient conjugué classique .

A chaque itération, on remarque qu'il est nécessaire, en plus des opérations habituelles, de résoudre un système $Cz = r$. Il est donc indispensable que cette résolution soit facile . En effet, une bonne matrice de préconditionnement peut entraîner une convergence très rapide souvent après $O(\sqrt{n})$ itérations . La question est donc comment choisir C ?

Il existe plusieurs méthodes pour choisir C . Nous considérons ici la factorisation incomplète de CHOLESKY .

Soit alors $M = LL^t$ la factorisation exacte de CHOLESKY de la matrice M . Cette procédure consiste à calculer une matrice triangulaire inférieure " creuse " H , telle que $C = HH^t$ soit voisine de M .

H sera choisie trés voisine de L . On prendra tout simplement $L_{i,j} = 0$ dans la factorisation exacte chaque fois que l'élément correspondant $M_{i,j}$ est nul . i.e :

$$H_{j,j} = \sqrt{M_{j,j} - \sum_{i=1}^{j-1} H_{i,j}^2} \quad , \quad j = 1 \text{ à } m$$

Pour $i = j+1$ à m

Si $M_{i,j} = 0$ alors $H_{i,j} = 0$

Sinon $H_{i,j} = (M_{i,j} - \sum_{k=1}^{j-1} H_{i,k} H_{j,k}) / H_{j,j}$

Ainsi, la méthode du gradient conjugué préconditionné par la factorisation incomplète de CHOLESKY , utilise le couplage d' une méthode itérative et d' une méthode directe .

Pour la méthode de KARMARKAR , le problème de conditionnement se pose dans le cas où le programme linéaire à résoudre est dégénéré, car plus on s'approche de l'optimum, plus la matrice $AD_k^2A^t$ est male conditionnée. ce qui conduit à des erreurs numériques fatales . GILL et al. [23] , ILAN ADLER et N. KARMARKAR [2], proposent indépendemment des implémentations sophistiquées qui utilisent une méthode de gradient conjugué préconditionné par une matrice triangulaire inférieure R qui est le facteur de CHOLESKY d'une matrice " creuse " (approximation de $AD_k^2A^t$) dans [23] , ou celui de $A_mD_k^2A_m^t$ où A_m est une sous matrice de A comme dans [2] .

Nous avons implémenté la factorisation de CHOLESKY ainsi que la méthode du gradient conjugué classique . Les résultats obtenus sont les mêmes pour les " petits " exemples que nous avons testés .

CHAPITRE 4

EXPERIMENTATIONS NUMERIQUES ET COMMENTAIRES

IV . 0 - Introduction

Nous avons implémenté plusieurs versions de l'algorithme de KARMARKAR qui correspondent aux différentes variantes étudiées au chapitre 2 .

L'intérêt des résultats numériques " comparatifs " présentés dans ce chapitre est double, ils permettent :

1 - D'avoir une idée claire sur le comportement de l'algorithme en question .

2 - D'avoir des indications concernant la performance relative de ces variantes .

Notre étude comparative est organisée de la manière suivante : On vérifie au départ que la méthode à deux phases est meilleure que la méthode primale-duale . Ensuite, on passe à la comparaison des trois algorithmes suivants :

1 - L'algorithme originel de KARMARKAR (valeur optimale connue au départ) .

2 - L'algorithme primal-dual (1) basé sur l'approche de TODD et BURRELL (la valeur optimale est approximée par des bornes inférieures)

3 - L'algorithme primal de YE-LUSTIG (la valeur optimale est approximée par des bornes supérieures) .

Pour terminer, l'algorithme qui s'avère le plus performant parmi ceux cités ci-dessus sera comparé à la méthode du simplexe ainsi qu'à l'algorithme S.G.G.P (simplexe généralisé-gradient projeté) introduit récemment par PHAM DINH TAO [60, 61].

Ces expérimentations numériques sont réalisées sur un micro-ordinateur (Olivetti : PC - M380) en turbo-pascal .

---

(1) Contrairement à la méthode primale-duale, cet algorithme utilise implicitement le problème dual , la taille du problème reste alors inchangée .

IV . 1 - Méthode à deux phases et méthode primale-duale :

La supériorité de la méthode à deux phases est évidente, il suffit de la voir à travers des exemples simples :

Exemple 1 (P. TOLLA)

$$
\begin{aligned}
\text{Min } c^t x = {} & 3x_1 - x_2 + x_3 && = -0.5 \\
& 2x_1 + x_2 - x_4 && = 0 \\
& + x_3 + x_5 - x_6 && = 0 \\
& x_1 + x_2 + x_3 + x_4 + x_5 + x_6 && = 1 \\
& x_i \geq 0, \quad i = 1, 2, \ldots, 6
\end{aligned}
$$

Exemple 2

$$
\begin{aligned}
\text{Min } c^t x = {} & -4x_1 - 5x_2 - x_3 - 3x_4 + 5x_5 - 8x_6 && = -17 \\
& x_1 - 4x_3 + 3x_4 + x_5 + x_6 && \leq 1 \\
& 5x_1 + 3x_2 + x_3 - x_5 + 3x_6 && \leq 4 \\
& 4x_1 + 5x_2 - 3x_3 + 3x_4 - 4x_5 + x_6 && \leq 4 \\
& - x_2 + 2x_4 + x_5 - 5x_6 && \leq 5 \\
& -2x_1 + x_2 + x_3 + x_4 + 2x_5 + 2x_6 && \leq 7 \\
& 2x_1 - 3x_2 + 2x_3 - x_4 + 4x_5 + 5x_6 && \leq 5 \\
& x_i \geq 0, \quad i = 1, 2, \ldots, 6
\end{aligned}
$$

Les tableux ci dessous présentent les résultats obtenus par les deux méthodes . Les critères d'optimalité étant respectivement :

$c^t x^k \leq 10^{-6}$ pour la méthode primale-duale et

$|c^t x^k - c^t x^{k-1}| \leq 10^{-6} (1 + |c^t x^k|)$ pour la méthode à deux phases .

La taille effective de chaque problème sera la taille de sa forme standard correspondante .

Méthode	Taille	Itérations	Objectif	Temps enregistré
Deux phases	(3 , 6)	13	-0.499999	0.83
Primale-duale	(10 , 20)	10	-0.499999	5.00

Exemple 1

Les solutions optimales fournies par la méthode primale-duale sont

Solution primale : $x^* = (0,\ 0.499999,\ 0,\ 0.499999,\ 0,\ 0)^t$.

Solution duale : $y^* = (-0.5,\ -0.20585,\ -0.5)^t$.

La méthode à deux phases donne les résultats suivants :

Phase 1 : 4 itérations l'objectif vaut : $c^t x^0 = 0.203165$

$x^0 = (0.06757,\ 0.13258,\ 0.13302,\ 0.26774,\ 0.13302,\ 0.26604)$

Phase 2 $x^* = (0,\ 0.499999,\ 0,\ 0.499999,\ 0,\ 0)^t$.

Méthode	Taille	Itérations	Objectif	Temps enregistré
Deux phases	(6 , 12)	18	-16.999998	3.35
Primale-duale	(19 , 38)	11	-16.999996	30.38

Exemple 2

Les solutions obtenues pour cet exemple sont :

Méthode primale-duale :

Solution primale : $x^* = (0,\ 0,\ 2.49999,\ 3.49999,\ 0,\ 0.49999)^t$.

Solution duale : $y^* = (-0.5,\ -1.5,\ 0,\ 0,\ -1.5,\ 0)^t$.

Méthode à deux phases :

Phase 1 : 5 itérations $c^t x^0 = -9.097888$.

$x^0 = (0.52868,\ 1.05148,\ 1.76490,\ 1.68091,\ 1.03375,\ 0.01084)^t$

Phase 2 : $x^* = (0,\ 0,\ 2.49999,\ 3,49999,\ 0,\ 0.5)^t$.

Conclusion :

Si le nombre d'itérations nécessaires pour la méthode primale-duale est significativement meilleur, le temps d'exécution est en revanche nettement supérieur à celui de la méthode à deux phases : (6 fois plus pour le premier exemple et 10 fois plus pour le second) . Ce phénomène s'amplifie lorsque la dimension du problème augmente .

IV . 2 - Comparaison des algorithmes de point intérieur

Signalons au début de ce paragraphe que :

- Tout problème de maximisation sera transformé en un problème de minimisation via la formule classique :

$$\text{Max } c^t x = -\text{Min } \{-c^t x\}$$

- Tout problème écrit sous forme canonique sera converti à la forme standard en ajoutant des variables d'écart .

- Le pas de déplacement est calculé par la méthode 1 décrite au chapitre 3 .

- Aucune technique de préconditionnement ou d'exploitation du creux de la matrice des contraintes n'est envisagée dans le calcul de la projection .

Considérons alors les programmes linéaires suivants :

Problème 1 (problème au hasard)

$$\text{Max } e_n^t x \quad |----------> \quad \text{Min } - e_n^t x$$
$$Ax \leq 10^4$$
$$x \geq 0$$

où $A \in \mathbb{R}^{MxN}$ avec $M \leq N$ composée de nombres entiers pris au hasard entre 1 et 1000 .

Prenons le cas $M = 5$ et $N = 10$ où A est donnée par :

1	2	3	4	5	5	4	3	2	1
6	7	8	9	10	5	2	8	3	1
11	12	13	14	15	6	7	80	90	10
1	10	20	30	40	50	60	80	90	10
3	9	27	60	45	60	75	8	9	46

La valeur optimale de ce problème est : -965.732. Sa taille sous forme standard est (m = 5 , n = 15) .

Les critères d'optimalité pour chaque méthode sont respectivement :

$|c^t x^k - c^t x^{k-1}| \leq 10^{-6} (1 + |c^t x^k|)$ pour la méthode de KARMARKAR .

$|c^t x^k - b^t y^k| \leq 10^{-6}$ pour l'algorithme primal-dual, où y^k désigne la solution réalisable du problème dual à l'itération k .

$||d_k||/|c^t x^0| \leq 10^{-6}$ pour l'algorithme de YE-LUSTIG, où $-d_k$ désigne la direction de déplacement à l'itération k .

Résultats obtenus :

Problème 1	Objectif	itérations	temps (en secondes)
Karmarkar	-965.7318	20	2.74
Ye-Lustig	-965.7319	17	2.90
Todd-Burrell	-965.7316	20	4.33

Phase 1: 8 itérations $c^t x^0 = -568.46166$.

Temps enregistré = 1.04 secondes .

$x^0 = (265.3323, 143.1992, 59.5332, 14.4774, 1.7921, 22.0378, 15.7902, 18.1029, 5.29, 22.9058)^t$.

Phase 2 : $x^* = (841.1212, 0, 0, 0, 0, 124.6104, 0, 0, 0, 0)^t$.

Problème 2

Min $c^t x$
$Ax \leq b$
$x \geq 0$

$c = (-139, -88, -133, -137, -165, 0, 0, 0, 0, 0, 0)^t$.

$b = (420, 415, 355, 345, 160, 95, 380, 395, 270, 230, 310, 420, 5200, 5200, 3600, 0)^t$.

1.4	1.8	1.5	1.4	1.4	0	0	0	0	0	0
9.8	2.4	1.4	1.4	1.4	0	0	0	0	0	0
4.0	0.4	1.4	1.4	1.4	0	0	0	0	0	0
2.8	0.6	1.3	1.4	1.5	5.5	0	0	0	0	0
2.2	0.4	1.3	1.5	1.2	0	5.5	0	0	0	0
2.2	0.4	1.3	1.3	1.2	0	0	5.5	0	0	0
2.2	0.6	1.3	1.3	1.2	0	0	0	5.5	0	0
2.6	5.8	1.5	1.5	1.2	0	0	0	0	5.5	0
0.6	4.0	1.3	0	0	0	0	0	0	0	5.5
0.6	1.2	1.3	1.3	2.6	0	0	0	0	0	0
0.6	1.8	1.2	1.2	1.2	0	0	0	0	0	0
0.6	1.8	1.5	1.4	1.4	0	0	0	0	0	0
16	0	0	12	35	50	50	0	0	0	0
20	0	0	36	50	0	0	50	50	0	0
16	0	0	12	35	0	0	0	0	50	50
0	0.1	0.9	0.8	2.3	-1	-1	-1	-1	-1	-1

Matrice A du problème 2 de dimension (16, 11)

La dernière contrainte est une égalité, la dimension du problème sous forme standard est alors : (16, 26) .

Résultats obtenus :

Problème 2	Objectif	itérations	temps (en secondes)
Karmarkar	-14021.0332	18	19.67
Ye-Lustig	-14021.0338	21	22.73
Todd-Burrell	-14021.0203	24	34.77

Phase 1 : 7 itérations $c^t x^0 = -10667.1428$

Temps enregistré : 7.8 secondes

$x^0 = (1.5239,\ 10.0359,\ 0.9758,\ 3.1899,\ 54.5777,\ 43.3124,\ 14.6113,\ 1.4846,\ 41.2836,\ 16.3986,\ 12.8719)^t$.

Phase 2 : $x^* = (0,\ 54.8015,\ 0,\ 20.3572,\ 38.8457,\ 40.9727,\ 11.0778,\ 0,\ 49.8253,\ 0,\ 9.2352)^t$.

Problème 3 (Hitac)

$$\text{Min } c^t x$$
$$Ax \geq b$$
$$x \geq 0$$

$c = (25, 30, 350, 150, 40, 20, 100, 40, 60, 100, 17, 20, 20, 12, 75, 900, 20)^t$.
$b = (2300, 75, 38, 660, 1300, 10, 1900, 1.2, 1.2, 63, 400)^t$.

A est de dimension (11, 17) . La forme standard correspondante est donc de taille (11 , 28) .

Matrice A : problème Hitac

351	270	260	451	156	59	721	58	130	118	77	51	40	24	28	311	40
5.2	8	17.5	12	12.7	2.9	0.6	6	17.5	20	1.9	1.3	1.2	1.6	3	34.2	0.8
0.8	1.5	20.5	44.3	11.2	3.3	81.6	3.5	6	3.5	0.1	0.2	0.2	0.2	0.4	0.7	0.3
6	11	6	9	65	100	10	120	80	12	5	35	40	45	98	470	14
2	480	90	90	90	36	780	5	100	90	12	57	10	15	25	600	4
0.4	1	1.3	1.2	2.6	0.1	0.1	1.4	3	0.7	0.5	0.5	0.5	0.4	3.3	23	0.2
0	0	25	0	800	120	2400	0	60	40	0	1300	6	33	2600	10000	40
0.09	0.1	0.04	0.4	0.1	0.04	0.01	0.02	0.02	0.15	0.1	0.06	0.03	0.08	0.12	0.21	0.09
0.03	0.03	0.11	0.1	0.4	0.15	0.03	0.02	0.15	0.2	0.03	0.04	0.02	0.05	0.3	1	0.02
0	0	0	0	0	2	0	0	1	1	15	7	10	50	100	20	50
0	0	0	0	10	0	0	0	530	0	0	0	0	0	0	0	0

La valeur optimale de ce problème est : 354.03042.

Résultats obtenus :

Problème 3	Objectif	itérations	temps (en secondes)
Karmarkar	354.03045	23	35.26
Ye-Lustig	354.03046	22	34.77
Todd-Burrell	354.06328	28	104.25

Phase 1 : 8 itérations $c^t x^0 = 505.6001$

Temps enregistré : 12.52 secondes.

$x^0 = (2.4698,\ 1.8322,\ 0.0574,\ 0.3454,\ 0.3544,\ 4.0358,\ 0.0021,\ 0.6060,\ 0.7541,\ 0.3065,\ 3.4814,\ 0.3356,\ 0.0131,\ 0.0453,\ 0.0153,\ 0.0592,\ 0.0268)^t$

Phase 2 : $x^* = (3.4509,\ 1.7709,\ 0,\ 0,\ 0.7557,\ 2.0717,\ 0.1406,\ 0,\ 0.7404,\ 0,\ 0,\ 0.3473,\ 0,\ 6.4601,\ 0,\ 0,\ 0)^t$.

Remarquons que la phase 2 de l'approche de TODD et BURRELL n'est pas tout à fait convergente pour ce problème .

Il est clair que si l'on se passe des solutions duales, l'algorithme à retenir (dans le cas où la valeur optimale du problème envisagé est inconnue) est celui de YE-LUSTIG .

IV . 3 - Comparaison avec la méthode du simplexe et l'algorithme S.G.G.P

Appliquons maintenant la méthode du simplexe et l'algorithme S.G.G.P aux trois problèmes précédents : Les résultats sont présentés dans le tableau ci-dessous .

SIMPLEXE	S.G.G.P	Valeur de l'objectif		Temps de calcul	
Problème 1		**-965.732**	-965.732	**0.29**	0.22
Problème 2		**-14021.030**	-14021.030	**6.53**	6.36
Problème 3		**354.030**	354.030	**6.38**	5.87

Ces deux algorithmes sont implémentés d'une façon optimale qui réduit significativement leurs complexités . En particulier, une technique exploitant le creux des matrices est mise en œuvre .

L'algorithme S.G.G.P (le plus performant ici) est de 3 à 12 fois plus rapide que la méthode de KARMARKAR . Ces rapports sont répartis de la manière suivante :

12 fois pour le problème 1
5 fois pour le problème 3
3 fois pour le problème 2

Ceci montre que plus la dimension du problème est grande plus l'algorithme de KARMARKAR atteint ses performances .

En effet, reprenons par exemple le problème 1 avec M = N = 10 , ce qui conduit à une forme standard de taille (10, 20) . La matrice correspondante est donnée par :

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 5 & 4 & 3 & 2 & 1 \\ 6 & 7 & 8 & 9 & 10 & 5 & 2 & 8 & 3 & 1 \\ 11 & 12 & 13 & 14 & 15 & 6 & 7 & 80 & 90 & 10 \\ 1 & 10 & 20 & 30 & 40 & 50 & 60 & 80 & 90 & 10 \\ 3 & 9 & 27 & 60 & 45 & 60 & 75 & 8 & 9 & 46 \\ 90 & 100 & 100 & 20 & 30 & 1000 & 900 & 25 & 1 & 1 \\ 3 & 30 & 300 & 2 & 20 & 200 & 1 & 10 & 100 & 150 \\ 5 & 10 & 15 & 20 & 25 & 30 & 35 & 40 & 45 & 50 \\ 1 & 11 & 111 & 2 & 22 & 222 & 3 & 33 & 333 & 4 \\ 7 & 70 & 8 & 80 & 9 & 90 & 10 & 100 & 15 & 155 \end{pmatrix}$$

La valeur optimale de ce problème est égale à : -286.4139

Les temps de calcul enregistrés sont respectivement :

7.8 secondes pour l'algorithme de YE-LUSTIG

1.15 secondes pour S.G.G.P . Le rapport de vitesse de convergence est alors égal à 6 en faveur de ce dernier .

IV . 4 - Commentaires

1- En ce qui concerne la comparaison entre l'algorithme de KARMARKAR et celui du simplexe, nous pouvons dire qu'à travers nos expérimentations numériques relatives à des programmes linéaires de " faible taille " , l'algorithme du simplexe est plus rapide . Les performances de l'algorithme de KARMARKAR seront à apprécier dans les programmes à grande dimension : Des résultats très supérieurs à ceux fournis par les méthodes simpliciales classiques ont été obtenus par ADLER et AL. [1] , LISSER [39] .

2 - Les méthodes de point intérieur pour la programmation linéaire sont conçues principalement pour des programmes linéaires mis sous forme standard . Cependant, on peut voir (en étudiant n'importe quel algorithme primal-dual) qu'en général, les solutions réalisables duales convergent vers l'optimum plus vite et d'une manière plus stable que les solutions réalisables primales . Ceci incite à adapter ce type d'algorithmes à des programmes linéaires avec des contraintes d'inégalité et des variables non astreintes . Cette formulation considérée pour la première fois par IRI et IMAI [33] et reprise récemment par GONZAGA [30] est reconnue actuellement très convenable en pratique .

3 - D'après KARMARKAR, pour $0 < \alpha < 1$, l'algorithme converge après $O(nq + n\log n)$ itérations où q est un entier indiquant la précision voulue pour la valeur de l'objectif . PADBERG [43] réduit ce nombre à $3nq = O(nq)$. Une question (ouverte) importante qu'on peut soulever ici est de savoir à quel point peut-on améliorer cette complexité ? . Autrement dit , il s'agit d'en trouver l'estimation optimale, car il est bien difficile de se prononcer sur la qualité des bornes précédentes . Sur un exemple de taille (11 , 25), pour atteindre une précision de 10^{-6} (i.e $q \simeq 18$), avec un pas $\alpha = 0.99$ seulement 33 itérations suffisent alors que $3nq = 1350$. De plus, on sait que sur le plan numérique, le nombre d'itérations grandit bien moins vite que la taille du problème . Peut être que la méthode de KARMARKAR admet elle aussi comme la méthode du simplexe un comportement, concernant le nombre d'itérations, qui est bien meilleur en pratique qu'en théorie ? .

CONCLUSION FINALE

La méthode de KARMARKAR a enrichi le domaine de la programmation linéaire aussi bien sur le plan théorique que sur le plan pratique.

Cette méthode, comprend trois grandes idées innovatrices :

1 - Plonger le problème initial dans l'espace projectif (contenant le simplexe unitaire) .

2 - Utiliser une projection pour trouver la direction de déplacement .

3 - Utiliser une fonction potentiel contenant simultanément des informations de l'objectif et des contraintes de positivité en vue de contrôler l'évolution de l'algorithme .

Notre travail apporte une synthèse des principales approches actuelles concernant la méthode de KARMARKAR .

Les expérimentations numériques comparatives réalisées sur (P.C) ont pu donner certaines indications relatives au comportement de cette méthode .

D'une façon générale, c'est la méthode S.G.G.P (introduite par PHAM DINH TAO) qui est la plus performante (surtout au niveau de la phase 1) suivie de la méthode du simplexe et puis la méthode de KARMARKAR .

Il est intéressant de noter que le rapport de gain diminue lorsque la dimension augmente . Cette constatation corrobore la théorie dans la mesure où la méthode de KARMARKAR par sa convergence polynomiale assure une stabilité numérique .

Les travaux visant à améliorer les performances de la méthode passent, à notre sens, par l'étude des techniques appropriées dans le calcul de la projection (plus particulièrement dans le cas des structures creuses des problèmes concrets de grandes tailles) .

REFERENCES

[1] **ILAN ADLER , MAURICIO G.C. RESENDE , GERALDO VEIGA**
An implementation of Karmarkar's Algorithm for Linear Programming :
ORC 86 - 8 May 1986
Industrial Engineering & Operations research , University of Berkley CA 94 720 .

[2] **ILAN ADLER , N. KARMARKAR , M. G.C.RESENDE ,G. VEIGA**
Data Structures and Programming Techniques for the implementation of Karmarkar's Algorithm
AT&T Bell laboratories, Murray Hill, NJ 07974
December 1987

[3] **KURT M. ANSTREICHER**
Analysis of a modified Karmarkar algorithm for linear programming :
Working paper Series B #84 . Operation Research :
August 1 , 1985 .

[4] **KURT M. ANSTREICHER**
A Strengthened Acceptance Criteterion for Approximate Projections in Karmarhkar's Algorithm
Operations Research Letters , Volume 5 , Number 4
October 1986

[5] **KURT M. ANSTREICHER**
A monotonic projective algorithme for fractional linear programming :
Algorithmica (1986) 1 : 483 - 498 .

[6] **KURT M. ANSTREICHER**
A Standard form variant, and Safeguarded Linesearch, for the modified Karmarkar algorithm
Yale School of Organization and Management
Box 1A New Haven, CT 06520
May 15, 1988

[7] **KURT M. ANSTREICHER , ROBERT A. BOSCH**
Long Steps in a $O(n^3L)$ Algorithm for Linear Programming
Yale School of Organization and Management
Box 1A New Haven, CT 06520
October 15, 1988

[8] **KURT M. ANSTREICHER**
A Combined Phase I-Phase II Projective Algprithm for Linear Programming
Mathematical Programming 43(1989) 209 - 223

[9] **EARL R. BARNES**
A variation on Karmarkar's Algorithm for Linear Programming Problems
Mathematical Programming 36(1986) 174 - 182 .

[10] **MOKHTAR S. BAZARAA , C.M. SHETTY**
Nonlinear Programming : Theory and Algorithms
John Wiley & Sons , Inc . 1979 .

[11] **M. BENICHOU , J.M. GAUTHIER , G. HENTGES , G. RIBIERGE**
The efficient Solution of large-scale Linear Programming Problems
Some Algorithmic Techniques and Computationnal Results .
IBM France , Paris , France . April 1977 .

[12] **XAVIER BENVENISTE**
L'algorithme de Karmarkar .
Centre de Mathématiques : Ecole Polytechnique
L.A. 169 C.N.R.S.
91 128 Palaiseau France June 24 , 1987 .

[13] **J.R. BIRGE**
A Dantzig-Wolfe Decomposition Variant Equivalent to Basis Factorization
Mathematical Programming Study 24(1985) 43 - 64 .

[14] **J.R. BIRGE and LIQUN QI**
Computing block-angular Karmarkar Projections with Application to Stochastic Programming .
Departement of industrial and operation engineering :
University of Michigan (Ann Abor , MI 48 109) .

[15] **N. CAMERON**
Introduction to Linear and Convex Programming
(Australian Mathematical Society Lecture Series ; 1)

[16] **J.E. DENNIS , Jr. , A.M. MORSHEDI and K. TURNER**
A Variable-Metric Variant of the Karmarkar Algorithm for Linear Programming .
Mathematical Programming 39(1987) 1- 20

[17] **I.S. DUFF , J. NOCEDAL and J.K. REID**
The use of Linear Programming for the Solution of Sparse Sets of Nonlinear Equations .
Siam J. Sc. Stat. Comput. Vol. 8 , N°2 , March 1987

[18] **R. FLETCHER and M.J.D. POWELL**
On the Modification of LDL^t Factorizations
Math. of Comput. , Vol. 28 , N°128 , october 1974 , p . 1067 - 1087

[19] **M. FORTIN & R. GLOWINSKI**
Méthodes de Lagrangien Augmenté : Application à la résolution numérique de problèmes aux limites .
Dunod , 1982

[20] **D.M. GAY**
A Variant of Karmarkar's Linear Programming Algorihm for Problems in Standard Form .
Mathematical Programming 37(1987) 81- 90

[21] **A. GEORGE and M.T. HEATH**
Solution of Sparse Linear Least Squares Problems Using Given's Rotations

[22] **G. DE GHELLINCK & J-PH. VIAL**
An Extension of Karmarkar's Algorithm for Solving a System of Linear Homogeneous Equations on the Simplexe .
Mathematical Programming 39(1987) 79 - 92

[23] **PH.E. GIII, W. MURRAY, M.A. SAUNDERS , J.A. TOMLIN and M.H.WRIGHT**
On projected Newton Barrier Method for Linear Programming and an Equivalence to Karmarkar's Projective Method .
Mathematical Programming 36 (1986) 183 - 209

[24] **PH. E. GILL , W. MURRAY and M.H. WRIGHT**
Practical optimization .
Academic Press INC (London) LTD . 1981

[25] **J.L. GOFFIN**
Affine and Projective Transformations in Nondifferentiable Optimization
International Series of Numerical Mathematics , Vol. 84
Trends in Mathematical Optimization (c) 1988

[26] **D. GOLDFARB & S. MEHROTRA**
Relaxed Variant of Karmarkar's Algorithm for Linear Programs Whith Unknown Optimal Objective Value .
Mathematical Programming 40(1988) 183 - 195

[27] **D. GOLDFARB & S. MEHROTRA**
A Relaxed Version of Karmarkar's Method .
Mathematical Programming 40(1988) 289 - 315

[28] **G.H. GOLUB & C.F. VAN LOAN**
Matrix Computations .
Johns Hopkins University press . 1983

[29] **C.C. GONZAGA**
Search Directions for Interior Linear Programming Methods .
University of California , Berkley , CA 94720
Memorandum N° UCB/ERL M 87/44 , June 1987

[30] **C.C. GONZAGA**
Interior Point Algorithms for Linear Programming Problems whith Inequality Contraints
COPPE-Federal University of Rio de Janeiro Cx. Postal 68511
21941 Rio de Janeiro, RJ, Brasil
December, 1987

[31] **C.C. GONZAGA**
Conical Projection Algorithms for Linear Programming
Mathematical Programming 43(1989) 151-173

[32] **H. IMAI**
On the Convexity of the Multiplicative Version of Karmarkar's Potential Function .
Mathematical Programming 40(1988) 29 - 32

[33] **M. IRI and H. IMAI**
A Multiplicative Penalty Function Method for Linear Programming , another new and fast Algorithm .
Department of Mathematical Engineering and Instrumentation Physics .
July 1985

[34] **M. KALLIO E. PORTEUS**
A Class of Methods for Linear Programming
Mathematical Programming 14(1978) 161-169

[35] **N. KARMARKAR**
A new Polynomial-time Algorithm for Linear Programming .
Combinatorica 4(4) , November 1984

[36] **N. KARMARKAR**
An Interior-point Approach to NP-complete Problems
AT&T Bell Laboratories
Murray Hill, New Jersey 07974

[37] **N. KARMARKAR , M. G.C. RESENDE , K.G. RAMAKRISHNAN**
An Interior Point Algorithm for Zero-One Integer Programming
August 1988

[38] **L.G. KHACHIYAN**
A polynomial algorithm in linear programming .
Doklady Akademiia , Nauk SSSR 224 : S (1979) , p. 1093 - 1096

[39] **ABDEL - ILAH LISSER**
Un logiciel Dérivé de L'algorithme de Karmarkar pour la Résolution de Programmes Linéaires de Grande Taille
Thèse de Doctorat de L'université (1987)
Université Paris-Dauphine (U.E.R. Sciences des Organisations)

[40] **IRVING J. LUSTIG**
A Practical Approach to KARMARKAR'S Algorithm
Systems Optimization Laboratory : Department of Operations Research
Stanford University , California 94305

[41] **M. MINOUX**
Programmation Mathématique : Théorie et Algorithmes
Tome 1 Dunod , Paris , 1983

[42] **M. MINOUX**
New Suggested Implementation of Karmarkar's Algorithm .
Lamsade , Cahier 71 , May 1986
Université de Paris-Dauphine

[43] **M. PADBEGR**
A Different Convergence Proof of the Projective Method for Linear Programming .
New York University , February 1985

[44] **J. RENEGAR**
A Polynomial-Time Algorithm , Based on Newton's Method , for Linear Programming .
Mathematical Programming 40(1988) 59 - 93

[45] **D.F. SHANNO**
Computing Karmarkar Projection Quickly .
Mathematical Programming 41(1988) 61- 71

[46] **M. SAKROVITCH**
Programmation Linéaire .
E.N.S.I.M.A.G. - Grenoble , Novembre 1983

[47] **M. SAKAROVITCH**
Optimisation Combinatoire .
E.N.S.I.M.A.G. - Grenoble , Mai 1983

[48] **M.J. TODD and B.P. BURRELL**
An Extension of Karmarkar's Algorithm for Linear Programming using Dual Variables .
Report 648 , January 1985
Coruell University , Ithaca , New York

[49] **M.J. TODD**
Exploiting Special Structure in Karmarkar's Linear Programming Algorithm
Mathematical Programming 41(1988) 97 - 113

[50] **M.J. TODD**
Recent Developments and new Directions in Linear Programming
Report 827 , November 1988
School of Operations Research and Industrial Engineering
College of Engineering, Cornell University , Ithaca, New York 14853

[51] **P. TOLLA**
Elaboration de Logiciels Efficaces Utilisant L'algorithme de Karmarkar .
Lamsade , Fevrier 1988 , Université de Paris-Dauphine

[52] **J.A. TOMLIN**
An Experimental Approach to Karmarkar's Projective Method for Linear Programming .
Math. Prog. Study 31(1987) 175 - 191

[53] **J. Ph. Vial**
A Unified Approach to Projective Algorithms for Linear Programming .
September , 1987 . Université de Genève

[54] **J. VIGNES**
Implémentation des Méthodes D'optimisation :
Test D'arrêt Optimal , Contrôle et Précision de la Solution .
R.A.I.R.O. Recherche Opérationnelle/Operations Research .
(Vol. 18 , N°1 , Fevrier 1984 , P. 1- 18)

[55] **YINYU YE , SAMUEL S. CHIU**
Recovering the Shadow Price in Projection Methods of Linear Programming
Engineering-Economic Systems Department , Stanford University
Stanford, California 94305
February 1985

[56] **YINYU YE , SAMUEL S. CHIU**
Simlex Method and Karmarkar's Algorithm : A Unifying Structure
Engineering-Economic Systems Department , Stanford University
Stanford, California 94305

[57] **YINYU YE**
Karmarkar's Algorithm and the Ellipsoid Method .
Operations Research Letters Vol. 6 N°4 , September 1987

[58] **YINYU YE and M. KOJIMA**
Recovring Optimal Dual Solution in Karmarkar's Polynomial Algorithm for Linear Programming .
Mathematical Programming 39(1987) 305 - 317

[59] **U. ZIMMERMANN**
On the Recent Developments in Linear Programming .
International Series of Numerical Mathematics , Vol. 84
Trends in Mathematical Optimization (c) 1988

[60] **PHAM DINH TAO**
Un Algorithme pour la Résolution du Programme Linéaire Général
Rapport technique, Université de Grenoble, Soumis à Publication .

[61] **A. YASSINE**
Etudes Adaptatives et Comparatives de Certains Algorithmes en Optimisation .
Thèse de Doctorat de Mathématiques Appliquées de l'Université JOSEPH FOURIER de Grenoble .

AUTORISATION DE SOUTENANCE

DOCTORAT 3ème CYCLE, DOCTORAT INGENIEUR, DOCTORAT DE L'UNIVERSITE JOSEPH FOURIER - GRENOBLE 1

Vu les dispositions de l'Arrêté du **16 avril 1974**,

Vu les dispositions de l'Arrêté du **5 juillet 1984**,

Vu les rapports de M^rs J.-P. CROUZEIX

M^r P. TOLLA

M^r K.ERAGHEL ABDELKRIM est autorisé(e) à présenter une thèse en vue de l'obtention du Doctorat de l'Université Joseph Fourier

Grenoble, le 22 JUIN 1989

Le Président de l'Université
Joseph Fourier - Grenoble 1

UNIVERSITE JOSEPH FOURIER GRENOBLE

A. NEMOZ

Résumé

Dans cette thèse, nous nous intéressons à l'étude adaptative et comparative des principales variantes dans l'algorithme de KARMARKAR . Au chapitre 1, après avoir décrit la méthode de KARMARKAR, nous montrons que la valeur du pas de déplacement proposée par ce dernier peut être largement améliorée . Nous dégageons ensuite les principales difficultés pratiques de cette méthode . Plus particulièrement, l'hypothèse de connaitre, au départ, la valeur optimale de l'objectif . Au chapitre 2, nous étudions diverses extensions et variantes de cette méthode dans le but de relaxer l'hypothèse ci-dessus . Nous présentons une approche unificatrice des variantes les plus significatives . Au chapitre 3, nous étudions les performances de cette méthode à l'aide des techniques appropriées dans la détermination du pas de déplacement ainsi que le calcul de la projection . Les expérimentations numériques obtenues par les différentes variantes étudiées au chapitre 2 et leur comparaison aux méthodes simpliciales (simplexe - S.G.G.P) sont présentées au chapitre 4 . Notre travail apporte une synthèse des principales approches actuelles concernant la méthode de KARMARKAR .

Mots clés

Programmation linéaire, méthode de KARMARKAR, transformation projective, fonction potentiel, méthode de point intérieur, algorithme primal-dual, méthode du simplexe, algorithme S.G.G.P .