GENERALIZAÇÕES DOS ALGORITMOS AFIM ESCALA PRIMAL E DUAL E UM ALGORITMO PROXIMAL PARA PROGRAMAÇÃO QUASE-CONVEXA

Francisco Gêvane Muniz Cunha

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Paulo Roberto Oliveira, Dr.Ing.

Prof. João Xavier da Cruz Neto, D.Sc.

Prof. Carlos Humes Júnior, Ph.D.

Prof. Manoel Bezerra Campêlo Neto, D.Sc.

Prof. Márcia Helena Costa Fampa, D.Sc.

Prof. Nair Maria Maia de Abreu, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

JANEIRO DE 2007

CUNHA, FRANCISCO GÊVANE MUNIZ

Generalizações dos Algoritmos Afim Escala Primal e Dual e um Algoritmo Proximal para Programação Quase-Convexa [Rio de Janeiro] 2007

XIII, 91 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2007)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1 - Programação Convexa

2 - Programação Quase-Convexa

3 - Métodos de Pontos Interiores

4 - Algoritmos Afim Escala

5 - Algoritmos de Ponto Proximal

I. COPPE/UFRJ II. Título (série)

"Não desampares a sabedoria, e ela te guardará;
ama-a, e ela te conservará."
(Provérbios 4:6)

- À minha esposa Aldenora e filho Gabriel.
- Aos meus pais Evilásio e Ormezinda.
- Aos meus irmãos e demais familiares.

Agradecimentos

À Deus, razão da minha vida, por sua infinita sabedoria e por me proteger e iluminar a cada dia.

À minha querida esposa Aldenora e ao meu dileto filho Gabriel, em especial, por sua compreensão e incentivo. Aos meus amados pais Evilásio e Ormezinda, irmãos e demais familiares, grandes responsáveis pelo meu empenho na realização desta conquista. O amor a todos vocês sempre me serviu de consolo e estímulo.

Aos meus ilustres orientadores Paulo Roberto Oliveira e João Xavier da Cruz Neto, pela amizade, excelente orientação e apoio durante todo o curso.

Aos membros da banca examinadora, por sua colaboração e contribuições para o aperfeiçoamento do trabalho.

Aos colegas e amigos Alexandre Salles, Erik Quiroz, Felipe Garcia, Flávia Morgana, Jurandir Lopes, Kely Diana, Luís Carlos, Nilomar Vieira, Paulo Sérgio, Roberto Cristovão, Roberto Prata, Ronaldo Gregório, Sérgio Assunção, Sissy Souza, Valtemir Martins e aos demais das bancadas do Ceará, Piauí e Amazonas com quem dividi momentos de concentração e também de descontração, pela amizade e agradável convívio.

Aos amigos Tibérius Bonates e Disney Douglas e ao SNTE/Piauí que me acolheram em algumas viagens.

Aos amigos Ivoneide Pinheiro e Luiz Barreto, pelo companheirismo e a todos os demais amigos, de longa ou de recente data, por tornarem a minha vida mais alegre.

Aos professores do PESC/UFRJ, pelos conhecimentos transmitidos.

Aos funcionários da secretaria, das bibliotecas e do suporte, pela atenção e ajuda em todas as ocasiões.

Ao CEFET-CE, por me conceder a oportunidade de realizar este curso.

Ao CNPq, pelo suporte financeiro.

A todos que de algum modo contribuíram para a realização deste trabalho. Muito obrigado.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

GENERALIZAÇÕES DOS ALGORITMOS AFIM ESCALA PRIMAL E DUAL E UM ALGORITMO PROXIMAL PARA PROGRAMAÇÃO QUASE-CONVEXA

Francisco Gêvane Muniz Cunha

Janeiro/2007

Orientadores: Paulo Roberto Oliveira
João Xavier da Cruz Neto

Programa: Engenharia de Sistemas e Computação

Em Pinto et al. [73] é proposta uma classe de algoritmos primais de pontos interiores para programação convexa com restrições lineares, de um certo tipo afim escala generalizado. Esta classe, dependente de um parâmetro r, generaliza alguns algoritmos conhecidos. De um modo similar, trabalhando no espaço dual, propomos uma classe de algoritmos afim escala duais para programação linear que generaliza o algoritmo afim escala dual de Adler et al. [1]. Esta classe é construída através da transformação dada pela potência S^{-r}, $r \geq 1$, onde S é a matriz diagonal do vetor de iteradas. Provamos a convergência global da classe afim escala dual generalizada para problemas de programação linear cujo dual seja não-degenerado. Em um outro enfoque, analisamos o método de ponto proximal com φ-divergência para programação quase-convexa sobre $\mathbb{R}^n_+$. Provamos, sem hipótese de limitação de nível para a função objetivo, que a seqüência gerada converge para um ponto estacionário. Em adição, provamos também que quando os parâmetros de regularização vão para zero, a seqüência converge para uma solução ótima. Observamos o desempenho computacional das classes de algoritmos afim escala primal e dual, realizando testes com problemas de programação linear da biblioteca NETLIB. Para a classe primal experimentamos também alguns problemas de programação quadrática descritos no repositório Maros e Mészáros. Verificamos a eficiência do algoritmo proximal em problemas quase-convexos sobre $\mathbb{R}^n_+$ por meio de experimentos realizados com problemas testes gerados aleatoriamente.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

GENERALIZATIONS OF THE PRIMAL AND DUAL AFFINE SCALING ALGORITHMS AND A PROXIMAL ALGORITHM FOR QUASICONVEX PROGRAMMING

Francisco Gêvane Muniz Cunha

January/2007

Advisors: Paulo Roberto Oliveira
João Xavier da Cruz Neto

Department: Computing and Systems Engineering

In Pinto et al. [73], it is proposed a class of primal interior point algorithms for convex programming with linear constraints, belonging to a certain generalized affine scaling type. That class, depending on a r-parameter, generalizes some known algorithms. In a similar way, working in the dual space, we propose a class of dual affine scaling algorithms to linear programming that generalize the dual affine scaling algorithm of Adler et al. [1]. This class is constructed through the transformation given by the power S^{-r}, $r \geq 1$, where S is the diagonal iterate vector matrix. We show the global convergence of the generalized dual affine scaling class for linear programs whose dual is nondegenerate. In another focus, we analyze the proximal point method with φ-divergence to quasiconvex programming on $\mathbb{R}^n_+$. We prove, without assumption of boundedness level to the objective function, that the generated sequence converges to a stationary point. In addition, we also prove that when the regularization parameters go to zero, the sequence converges to an optimal solution. We observe the computational performance of the classes of primal and dual affine scaling algorithms, accomplishing tests with linear programs from the NETLIB library. For the primal class we also experiment some quadratic programming problems described in the Maros and Mészáros repository. We verify the effectiveness of the proximal algorithm in quasiconvex problems on $\mathbb{R}^n_+$ via numerical experiments accomplished with tests problems randomly generated.

Sumário

Lista de Figuras

Lista de Tabelas

Notações

$\mathbb{R}^n$	Espaço Euclideano real $n$dimensional.
x_i	$i$ésima componente de um vetor x.
$\mathbb{R}^n_+$	Octante não-negativo de $\mathbb{R}^n$, $\mathbb{R}^n_+ \equiv \{x \in \mathbb{R}^n \mid x_i \geq 0,\ i = 1, ..., n\}$.
$\mathbb{R}^n_{++}$	Interior de $\mathbb{R}^n_+$, $\mathbb{R}^n_{++} \equiv \{x \in \mathbb{R}^n \mid x_i > 0,\ i = 1, ..., n\}$.
x_J	Subvetor cujas componentes são as de x indexadas por J.
A_J	Submatriz consistindo das colunas de A indexadas por J.
A^T	Transposta de uma matriz A.
$\langle x, y \rangle$	Produto interno Euclideano dos vetores x e y, $\langle x, y \rangle \equiv x^T y = \sum_{i=1}^n x_i y_i$.
$\|x\|$	Norma Euclideana do vetor x, $\|x\| \equiv \langle x, x \rangle^{1/2}$.
x^r	Vetor das $r$potências das componentes de x, $x^r \equiv (x_1^r, ..., x_n^r)^T$.
X^r	Matriz diagonal com diagonal dada pelas componentes de x^r, $X^r \equiv \text{diag}(x^r) = \text{diag}(x_1^r, ..., x_n^r)$.
$X \succeq 0$	X é uma matriz simétrica semidefinida positiva.
$X \succ 0$	X é uma matriz simétrica definida positiva.
$f'(x)$	Derivada de f em x, para funções $f : \mathbb{R} \to \mathbb{R}$.
$\nabla f(x)$	Gradiente de f em x.
$(\nabla f(x))_i$	$i$ésima derivada parcial de f em x.
$\nabla_x f(x, y)$	Derivada parcial de f com respeito a x.
$\nabla^2 f(x)$	Matriz Hessiana de f em x.
$\text{dom} f$	Domínio da função f.
$\text{niv}_f(\alpha)$	Conjunto de nível de f em α, $\text{niv}_f(\alpha) \equiv \{x \in C \mid f(x) \leq \alpha\}$.
epi_f	Epígrafo de f, $\text{epi}_f \equiv \{(x, \alpha) \in C \times \mathbb{R} \mid f(x) \leq \alpha\}$.
$\mathcal{S}^*(P)$	Conjunto das soluções do problema (P).

Capítulo 1

Introdução

O objeto de estudo desta tese é a *programação matemática*. Este ramo da *pesquisa operacional* tem o propósito de estudar a teoria de problemas de *otimização* em espaços de dimensão finita e os conceitos e a implementação de métodos de solução. Com um certo grau de elegância e simplicidade operacional, a otimização é um princípio subjacente à análise de muitas decisões complexas ou problemas de distribuição. Usando a metodologia da otimização, pode-se abordar um problema de decisão complexo, envolvendo a seleção de valores para um determinado número de variáveis relacionadas, focando a atenção em um único objetivo. Este objetivo, que pode ser lucro ou perda em um cenário de negócios, velocidade ou distância em um problema físico, retorno esperado em um ambiente de investimentos de risco, ou equilíbrio da previdência social no contexto de planejamento governamental é maximizado (ou minimizado, dependendo da formulação) sujeito a restrições que podem limitar a seleção dos valores das variáveis de decisão.

Em muitas situações, será impossível representar completamente todas as complexidades de interações das variáveis, restrições, e objetivos apropriados quando enfrentando um problema de decisão complexo. Assim, como toda técnica quantitativa de análise, uma particular formulação de otimização só deve ser considerada como uma aproximação. Portanto, habilidade em modelagem para capturar os elementos essenciais de um problema, e bom julgamento na interpretação de resultados são exigidos para obter conclusões significativas. Estas exigências são aprimoradas pela compreensão completa da teoria pertinente e por experiência prática concreta.

Problemas de Otimização

Um dos elementos constituintes de um problema de otimização é o conjunto de variáveis ou parâmetros independentes, e freqüentemente se incluem condições ou

restrições que definem valores aceitáveis das variáveis. Tais restrições são denominadas as *restrições do problema* e determinam sua *região viável.* O outro componente essencial de um problema de otimização é a medida de qualidade, denominada *função objetivo* que depende de algum modo das variáveis. Uma solução de um problema de otimização é um conjunto de valores permitidos das variáveis para os quais a função objetivo assume o valor *ótimo.* Em termos da matemática, otimização usualmente envolve maximização ou minimização. Podemos, por exemplo, desejar maximizar lucros ou minimizar perdas.

Para propor técnicas de solução, é útil assumir que os problemas de otimização podem ser colocados em uma forma padrão. Para evitar a necessidade de reformulação, é claramente desejável selecionar uma forma padrão que surja naturalmente da natureza da maioria dos problemas de otimização. A forma geral do problema de otimização a ser considerada pode ser expressa nos seguintes termos:

$$\begin{array}{ll} \text{minimize}_x & f(x) \\ \text{sujeito a} & c_i(x) = 0, \quad i \in I_= \equiv \{1, 2, ..., m'\} \\ & c_i(x) \leq 0, \quad i \in I_\leq \equiv \{m' + 1, ..., m\}. \end{array}$$

A função objetivo f e as funções restrições $\{c_i\}$ (que, tomadas juntas, são denominadas as *funções do problema*) são funções escalares de valor real.

Programação Matemática

Historicamente, a programação matemática marca o começo de um enorme movimento de unificação e generalização de temas atacados durante os anos 1945 a 1960, quando Dantzig [24] propôs o termo *programação linear* para o estudo de problemas teóricos e algorítmicos relacionados à otimização de funções lineares, sujeito a restrições lineares. No mesmo sentido, Kuhn e Tucker [56] propuseram o termo *programação não-linear* para o estudo de problemas de otimização não-linear com ou sem restrições. *Programação inteira* foi sugerido por Gomory [35] para problemas de otimização onde as variáveis são restritas a tomar apenas valores inteiros, enquanto o termo *programação dinâmica* foi aplicado por Bellman [12] para descrever os processos de otimizar sistemas dinâmicos (isto é, aqueles que se desenvolvem ao longo do tempo). Apesar da aparente diversidade destes temas, percebeu-se progressivamente uma profunda semelhança entre as várias classes de problemas, em matéria de estruturas como também métodos, que levou a submetê-las a uma disciplina nova e mais ampla, a *programação matemática.* Esta terminologia reflete a íntima relação entre a atividade de análise matemática de um problema e sua interpretação

econômica (a procura por programa econômico ótimo).

A programação matemática é um ramo particularmente ativo da matemática aplicada. Ela oferece um ambiente conceitual adequado para a análise e a solução de muitos problemas importantes em engenharia e em outros campos, como exemplificado abaixo:

- Em pesquisa operacional: otimização de sistema técnico-econômico (planejamento, economia), problemas de transporte, programação, controle de estoque, etc.
- Em análise numérica: aproximação, regressão, solução de sistemas lineares e não-lineares, métodos numéricos conectados com a implementação de métodos de elementos finitos, etc.
- Em automação: identificação de sistemas, controle ótimo de sistemas, filtragem, programação de loja de trabalho, controle de robôs, etc.
- Em engenharia: dimensionamento e otimização de estruturas, desenvolvimento de sistemas técnicos complexos ótimos como sistemas de informação, redes de computadores, redes de transportes e de telecomunicações, etc.
- Em economia matemática: solução de grandes modelos macroeconômicos (o modelo tipo Leontiev e suas variações), modelos microeconômicos ou modelos de empresa, teoria da decisão e teoria dos jogos.

Problemas em todas as áreas da matemática, ciências aplicadas, engenharias, economia, administração, medicina e estatística podem ser interpretados e resolvidos pela programação matemática.

Métodos de Pontos Interiores

Atualmente, os métodos clássicos para resolver problemas de otimização com restrições podem ser distinguidos a grosso modo em: métodos de *pontos interiores*, métodos de *pontos exteriores* e métodos de *restrições ativas*. Os métodos de pontos interiores, que é a classe em que estamos interessados nesta tese, operam no interior relativo da região viável do problema.

Métodos de pontos interiores para programação matemática são conhecidos há muito tempo e atualmente representam um tema central e notável da otimização com restrições. Mas não foi sempre assim. Principalmente na forma de métodos

de barreira, técnicas de ponto interior foram extensamente usadas durante os anos sessenta para resolver problemas não-lineares restritos [31, 32]. Porém, o uso deles para programação linear não era nem mesmo considerado por causa do domínio total do método simplex de G. B. Dantizg [24]. Durante os anos setenta, os métodos de barreira foram substituídos por novas alternativas aparentemente mais eficientes como métodos de lagrangeano aumentado e métodos de programação quadrática seqüencial, veja Gill et al. [34] para maiores detalhes destes métodos. Pelo início dos anos oitenta, os métodos de barreira foram considerados quase universalmente como um capítulo fechado na história da otimização.

Este quadro mudou drasticamente no meio dos anos 1980. Em 1984, Karmarkar anunciou um método interior com complexidade em tempo-polinomial para programação linear. Em 1986, Gill et al. [33] estabeleceram uma conexão formal entre o método de Karmarkar e os métodos de barreira clássicos. Desde então, a busca por novos métodos interiores avançou muito e tão rápido que a influência destes métodos transformou a teoria e a prática da otimização com restrições.

Segundo Den Hertog [25], os métodos de pontos interiores podem ser agrupados em quatro categorias (na ordem histórica):

- Métodos seguidores da trajetória.
- Métodos afim escala.
- Métodos projetivos de redução potencial.
- Métodos afim de redução potencial.

As diferenças entre estas classes são algumas vezes vagas e todos estes métodos baseiam-se em algumas noções e conceitos comuns. Pode-se também distinguir quando estes algoritmos são algoritmos apenas primais (apenas duais) ou algoritmos primais-duais.

Objetivos e Escopo da Tese

Nosso trabalho está dividido em duas partes independentes. Na primeira parte, correspondente aos Capítulos 2, 3 e 4, abordamos algoritmos do tipo *afim escala* para programação convexa com restrições lineares. Mais precisamente, apresentamos classes de algoritmos afim escala primais e duais que generalizam certos algoritmos conhecidos. Na segunda parte, correspondente aos Capítulos 5, 6 e 7, analisamos o

método de ponto proximal para a minimização de funções quase-convexas sujeitas a restrições de não-negatividade. A seguir descrevemos o que será feito nos próximos capítulos.

No Capítulo 2, como forma de fundamentação teórica, reunimos as principais definições e resultados básicos em *programação convexa* úteis para nosso trabalho.

No Capítulo 3, apresentamos as idéias básicas dos *algoritmos afim escala* e um pouco de sua história. Exibimos o *algoritmo afim escala primal generalizado* de Pinto et al. [73] para programação convexa com restrições lineares. Em adição, propomos um *algoritmo afim escala dual generalizado*. Provamos também a convergência global deste algoritmo aplicado a problemas de programação linear cujo dual seja não-degenerado. Os resultados obtidos com os algoritmos afim escala primal e dual generalizados, bem como o desempenho computacional destes algoritmos, podem ser vistos em Cunha et al. [22].

O objetivo do Capítulo 4 é observar o desempenho computacional das classes de algoritmos afim escala descritas no Capítulo 3. Para isto, implementamos as famílias de algoritmos aplicadas a alguns problemas de programação linear obtidos da biblioteca NETLIB. No caso da família primal, realizamos testes também com alguns problemas de programação quadrática descritos no repositório Maros e Mészáros. Os resultados obtidos em nossos experimentos nos permitem comparar os algoritmos afim escala generalizados com algoritmos afim escala clássicos (quando $r = 1$ ou $r = 2$).

No Capítulo 5, apresentamos definições e resultados básicos em *programação quase-convexa* que consideramos importantes em nossa tese.

O Capítulo 6 é dedicado à apresentação e análise do *algoritmo de ponto proximal com φ-divergência para programação quase-convexa*. Reservamos uma seção para apresentar a definição de φ-divergências, algumas de suas propriedades e listar alguns exemplos. Este capítulo corresponde ao trabalho desenvolvido por Cunha et al. [23].

No Capítulo 7, verificamos a eficiência do algoritmo proximal estudado por meio de diversos experimentos numéricos realizados com problemas testes gerados aleatoriamente.

Finalizamos com o Capítulo 8, onde fazemos nossas considerações finais, apresentando algumas conclusões e sugestões de possíveis trabalhos futuros de continuidade.

Capítulo 2

Conceitos e Resultados Básicos em Programação Convexa

Apresentamos, neste capítulo, conceitos e resultados básicos em programação convexa que são essenciais para o desenvolvimento de todo o trabalho. Nosso enfoque tem em vista os aspectos teóricos que efetivamente contribuem para o desenvolvimento de algoritmos práticos.

Convexidade é uma noção muito importante na teoria de otimização. Uma das propriedades mais fortes obtidas com hipóteses de convexidade (grosso modo, quando a função objetivo é convexa e a região viável do problema é um conjunto convexo) em um problema de minimização é que as condições necessárias de otimalidade passam a ser suficientes. Em outras palavras, todo ponto estacionário torna-se uma solução do problema. Em particular, qualquer minimizador local é global. A extensa teoria da análise convexa pode ser vista nos livros Hiriart-Urruty e Lemaréchal [40], Rockafellar [80] ou Stoer e Witzgall [85].

Os resultados apresentados neste capítulo são propriedades bem conhecidas em programação convexa. Suas demonstrações podem ser vistas em literaturas básicas [11, 18, 59, 80, 94].

2.1 Conjuntos Convexos

Um conjunto convexo se caracteriza por conter todos os segmentos cujos extremos são pontos do conjunto.

Definição 2.1.1. *([11], Definição 2.1.1, p.34) Um conjunto $C \subset \mathbb{R}^n$ é convexo se o segmento de reta que liga quaisquer dois pontos de C pertencer também a C. Equivalentemente, o conjunto $C \subset \mathbb{R}^n$ é convexo se, e somente se, para todo $x, y \in C$*

e $\lambda \in \mathbb{R}$ *com* $0 \leq \lambda \leq 1$ *tem-se*

$$\lambda x + (1 - \lambda)y \in C.$$

Chamamos um ponto da forma $\lambda_1 x_1 + ... + \lambda_k x_k$ onde $\lambda_1 + ... + \lambda_k = 1$ e $\lambda_i \geq 0$, $i = 1, ..., k$, uma *combinação convexa* dos pontos $x_1, ..., x_k$. A *envoltória convexa* de um conjunto C, denotado por conv$C$, é a interseção de todos os conjuntos convexos contendo C.

Teorema 2.1.1. *([80], Teorema 2.3, p.12) Para qualquer conjunto* $C \subset \mathbb{R}^n$, *conv*C *consiste de todas as combinações convexas de elementos de* C, *isto é,*

$$conv C = \{\lambda_1 x_1 + ... + \lambda_k x_k \mid x_i \in C,\ \lambda_1 + ... + \lambda_k = 1,\ \lambda_i \geq 0,\ i = 1, ..., k, \},$$

onde k é um número inteiro positivo.

Pode-se provar que conv$C$ é o menor conjunto convexo que contém C.

Convexidade é preservada por algumas operações algébricas, entre elas a interseção de conjuntos.

Exemplos importantes de conjuntos convexos são os subespaços de $\mathbb{R}^n$, os semi-espaços em $\mathbb{R}^n$, isto é, os conjuntos $\{x \in \mathbb{R}^n \mid \langle a, x \rangle \leq c\}$, onde $a \in \mathbb{R}^n$ e $c \in \mathbb{R}$ e os hiperplanos em $\mathbb{R}^n$, isto é, os conjuntos $\{x \in \mathbb{R}^n \mid \langle a, x \rangle = c\}$, onde $a \in \mathbb{R}^n$ e $c \in \mathbb{R}$. Poliedros representam um caso especial importante de conjuntos convexos.

Definição 2.1.2. *([11], Definição 2.6.1, p.54) Um conjunto* $S \subset \mathbb{R}^n$ *é chamado um poliedro se ele é a interseção de um número finito de semi-espaços fechados, isto é,*

$$S = \{x \mid a_i^T x \leq b_i,\ i = 1, ..., m\},$$

onde a_i *é um vetor não-nulo e* b_i *é um escalar para* $i = 1, ..., m$.

Desde que uma equação pode ser representada por duas inequações, um poliedro pode ser representado por um número finito de inequações e/ou equações. Sendo a interseção de convexos um convexo, um poliedro é um conjunto convexo fechado.

São exemplos de poliedros: o octante não-negativo de $\mathbb{R}^n$, isto é, $\mathbb{R}^n_+ = \{x \in \mathbb{R}^n \mid x_i \geq 0,\ i = 1, ..., n\}$ e os conjuntos $\{x \mid Ax \leq b\}$, $\{x \mid Ax = b,\ x \geq 0\}$ e $\{x \mid Ax \geq b,\ x \geq 0\}$, onde A é uma matriz $m \times n$ e b é um vetor do $\mathbb{R}^m$.

Um poliedro pode ser descrito completamente por meio de seus pontos extremos e direções extremas. Este fato é fundamental para diversos procedimentos em programação linear e não-linear.

Definição 2.1.3. *([11], Definição 2.6.2, p.55) Seja $C \subset \mathbb{R}^n$ um conjunto convexo não-vazio. Um vetor $x \in \mathbb{R}^n$ é um ponto extremo de C se $x = \lambda x_1 + (1-\lambda)x_2$ com $x_1, x_2 \in C$, e $\lambda \in (0,1)$ implica que $x = x_1 = x_2$, isto é, x não pode ser obtido como combinação linear convexa de nenhum par de pontos distintos de C.*

Definição 2.1.4. *([11], Definição 2.6.3, p.56) Seja $C \subset \mathbb{R}^n$ um conjunto convexo fechado, não-vazio. Um vetor não-nulo $d \in \mathbb{R}^n$ é uma direção, ou uma direção de ilimitação, de C se para cada $x \in C$, $x + \lambda d \in C$ para todo $\lambda \geq 0$. Duas direções d_1 e d_2 de C são ditas distintas se $d_1 \neq \alpha d_2$ para todo $\alpha > 0$. A direção d de C é dita uma direção extrema se ela não pode ser escrita como uma combinação linear positiva de duas direções distintas, isto é, se $d = \lambda_1 d_1 + \lambda_2 d_2$ para λ_1, $\lambda_2 > 0$, então $d_1 = \alpha d_2$ para algum $\alpha > 0$.*

O resultado principal pode ser estabelecido como segue. Seja o poliedro da forma $\mathcal{S} = \{x \mid Ax = b,\ x \geq 0\}$. Então, qualquer ponto em $\mathcal{S}$ pode ser representado como uma combinação convexa de seus pontos extremos mais uma combinação linear não-negativa de suas direções extremas. Se $\mathcal{S}$ é limitado, ele não contém direções, e assim qualquer ponto em $\mathcal{S}$ pode ser descrito como combinação convexa de seus pontos extremos. No teorema abaixo, está implícito que os pontos extremos e direções extremas de $\mathcal{S}$ são em número finito.

Teorema 2.1.2. *([11], Teorema 2.6.7, p.60 - Teorema de Representação) Seja $\mathcal{S}$ um poliedro não-vazio em $\mathbb{R}^n$ e sejam $x_1, ..., x_k$ os pontos extremos de $\mathcal{S}$ e $d_1, ..., d_l$ as direções extremas de $\mathcal{S}$. Então, $x \in S$ se, e somente se, x pode ser escrito como*

$$\begin{array}{rcl} x & = & \sum_{j=1}^{k} \lambda_j x_j + \sum_{j=1}^{l} \mu_j d_j \\ \sum_{j=1}^{k} \lambda_j & = & 1 \\ \lambda_j & \geq & 0, \quad j = 1, ..., k \\ \mu_j & \geq & 0, \quad j = 1, ..., l \end{array}$$

2.2 Funções Convexas

Apresentamos aqui algumas propriedades básicas de funções convexas e côncavas. Em particular, fornecemos caracterizações de funções convexas diferenciáveis.

Definição 2.2.1. *([11], Definição 3.1.1, p.79 e [40], Definição 1.1.1, p.143) Seja $C \subset \mathbb{R}^n$ um conjunto convexo não-vazio. A função $f : C \to \mathbb{R}$ é convexa em C*

quando para quaisquer $x, y \in C$ *e* $\lambda \in (0,1)$, *tem-se*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y).$$

A função f *diz-se estritamente convexa quando a desigualdade acima é verdadeira como uma desigualdade estrita para todos distintos* x *e* y *em* C *e* $\lambda \in (0,1)$. *A função* f *diz-se fortemente convexa com módulo* $\gamma > 0$, *quando para quaisquer* x *e* y *em* C *e* $\lambda \in [0,1]$, *tem-se*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) - \frac{1}{2}\gamma\lambda(1-\lambda)\|x-y\|^2.$$

A função $f : C \to \mathbb{R}$ *é chamada côncava, estritamente côncava ou fortemente côncava em* C *conforme* $-f$ *seja, respectivamente, convexa, estritamente convexa ou fortemente convexa em* C.

Uma função fortemente convexa é estritamente convexa, e uma função estritamente convexa é convexa.

A relação entre convexidade de conjuntos e de funções é estabelecida no teorema seguinte que dá uma caracterização de funções convexas.

Teorema 2.2.1. *([11], Teorema 3.2.2, p.84) Seja* C *um conjunto convexo não-vazio em* $\mathbb{R}^n$. *A função* $f : C \to \mathbb{R}$ *é convexa em* C *se, e somente se, o epígrafo de* f, $\text{epi}_f = \{(x,\alpha) \in C \times \mathbb{R} \mid f(x) \leq \alpha\}$ *é convexo.*

Algumas operações algébricas como a soma de múltiplos não negativos de um número finito de funções convexas e o supremo de funções convexas preservam a convexidade. Sejam $g : \mathbb{R}^n \to \mathbb{R}$ uma função convexa e $h : \mathbb{R} \to \mathbb{R}$ uma função convexa não-decrescente então, a função composta $f : \mathbb{R}^n \to \mathbb{R}$ definida por $f(x) = h[g(x)]$ é uma função convexa. Se $g : \mathbb{R}^n \to \mathbb{R}$ é uma função côncava e $C = \{x \mid g(x) > 0\}$ então, a função $f : C \to \mathbb{R}$ definida por $f(x) = \frac{1}{g(x)}$ é convexa sobre C.

Os conjuntos de nível de uma função convexa são convexos.

Teorema 2.2.2. *([11], Lema 3.1.2, p.80) Seja* C *um conjunto convexo não-vazio em* $\mathbb{R}^n$ *e seja* $f : C \to \mathbb{R}$ *uma função convexa. Então, o conjunto de nível* $\text{niv}_f(\alpha) = \{x \in C \mid f(x) \leq \alpha\}$ *é convexo para todo* $\alpha \in \mathbb{R}$.

Algumas vezes, este conjunto é chamado *conjunto de nível inferior*, para diferenciá-lo do *conjunto de nível superior* $\text{niv}_f^{\geq}(\alpha) = \{x \in C \mid f(x) \geq \alpha\}$, $\alpha \in \mathbb{R}$, que tem propriedades similares para funções côncavas.

Quando uma função é diferenciável, a convexidade admite caracterizações úteis.

Teorema 2.2.3. *([11], Teorema 3.3.3, p.89 e Teorema 3.3.7, p.91 - Caracterizações de Funções Convexas Diferenciáveis) Sejam $C \subset \mathbb{R}^n$ um conjunto convexo e aberto e $f : C \to \mathbb{R}$ uma função diferenciável em C. Então, as propriedades seguintes são equivalentes:*

(i) *A função f é convexa em C.*

(ii) *Para todo $x \in C$ e todo $y \in C$,*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle.$$

(iii) *Para todo $x \in C$ e todo $y \in C$,*

$$\langle \nabla f(y) - \nabla f(x), y - x \rangle \geq 0.$$

Quando f é duas vezes diferenciável em C, as propriedades acima também são equivalentes a:

(iv) *A matriz Hessiana de f é semidefinida positiva em todo ponto de C, isto é,*

$$\langle \nabla^2 f(x)d, d \rangle \geq 0, \quad \forall x \in C, \quad \forall d \in \mathbb{R}^n.$$

2.3 Otimização Convexa com Restrições Lineares

Este capítulo da otimização pode ser visto, por exemplo, em Bazaraa et al. [11], Luenberger [59] e Wright [94].

Consideremos o problema de otimização com restrições lineares

$$\begin{array}{lll} \text{(PORL)} & \text{minimize}_u & f(u) \\ & \text{sujeito a} & Cu = d \\ & & Gu \geq h \\ & & u \in U, \end{array}$$

onde $f : \mathbb{R}^n \to \mathbb{R}$ é uma função diferenciável, $U \subset \mathbb{R}^n$ é um conjunto aberto, C e G são matrizes, e d e h vetores.

Definição 2.3.1. *([94], Apêndice A, p.240) Um ponto $u^* \in \mathbb{R}^n$ é uma solução local de (PORL) se:*

(i) *É viável para (PORL): $u^* \in U$, $Cu^* = d$ e $Gu^* \geq h$.*

(ii) *Existe um escalar* $\rho > 0$ *tal que* $f(u) \geq f(u^*)$ *para todo* u *viável com* $\|u-u^*\| < \rho$.

O ponto u^* *é uma solução local estrita se, em adição,* $f(u) > f(u^*)$, *para todo* $u \neq u^*$ *com* u *viável e* $\|u - u^*\| < \rho$.

Estas definições podem ser extendidas para *solução global* e *solução global estrita* tomando $\rho = \infty$ na definição local.

As condições necessárias de primeira ordem, também conhecidas como condições de Karush-Kuhn-Tucker, ou condições KKT, são definidas no seguinte teorema.

Teorema 2.3.1. *([94], Teorema A.1, p.240 - Condições de Otimalidade) Suponha que* u^* *é uma solução local de (PORL) e que* f *é diferenciável em uma vizinhança de* u^*. *Então existem vetores* y *e* z *tais que as seguintes condições acontecem:*

$$\nabla f(u^*) - C^T y - G^T z = 0 \tag{2.1}$$

$$Cu^* = d \tag{2.2}$$

$$Gu^* \geq h \tag{2.3}$$

$$z \geq 0 \tag{2.4}$$

$$z^T(Gu^* - h) = 0 \tag{2.5}$$

Os vetores y e z são os *multiplicadores de Lagrange* para as restrições $Cu = d$ e $Gu \geq h$, respectivamente. A condição (2.5) implica que para cada i, uma das componentes z_i ou $(Gu^* - h)_i$ deve ser nula. Esta é conhecida como *condição de complementaridade*, pois implica que componentes não nulas de z_i ou $(Gu^* - h)_i$ aparecem em localizações complementares.

Definição 2.3.2. *([94], Definição 1, p.241) Uma solução* u^* *do problema (PORL) e os correspondentes multiplicadores de Lagrange* (y, z) *que satisfazem (2.1)-(2.5) são estritamente complementares se* $z + (Gu^* - h) > 0$.

Esta definição, em conjunção com as condições (2.3)-(2.5), implicam que para qualquer índice i temos ou

$$z_i = 0, \quad (Gu^* - h)_i > 0,$$

ou

$$z_i > 0, \quad (Gu^* - h)_i = 0.$$

Se U é um conjunto aberto convexo ($\mathbb{R}^n$, por exemplo), então a região viável para (PORL) é um conjunto convexo (interseção de conjuntos convexos). Se a função objetivo f é também convexa no conjunto viável, então (PORL) é um tipo de problema de *programação convexa*. Mais especificamente, (PORL) é um problema de programação convexa com restrições lineares, que é uma classe em que estamos interessados. Neste caso, soluções locais e globais são relacionadas da seguinte maneira.

Teorema 2.3.2. *([94], Teorema A.2, p.240) Se (PORL) for um problema de programação convexa, então*

(a) *As condições KKT são suficientes para u^* ser uma solução global. Isto é, se existem vetores y e z tais que (2.1)-(2.5) são satisfeitas, então u^* é uma solução global de (PORL).*

(b) *Se, em adição, f é estritamente convexa em sua região viável, então qualquer solução local é a solução global definida unicamente.*

2.4 Otimização Linear

A *Programação Linear* é uma das áreas de grande sucesso da *Otimização* devido à sua grande aplicabilidade em quase todos os setores do cotidiano, sendo objeto de estudo para economistas, analistas de finanças e engenheiros. Algumas referências básicas são Bazaraa et al. [10, 11], Luenberger [59], Chvátal [18] e Wright [94].

O Problema de Programação Linear (PPL) foi formulado na década de 30 e pela metade da década de 40, Dantzig [24] apresentou o algoritmo *Simplex*, um dos mais interessantes algoritmos para resolvê-lo por ser simples e bastante eficiente na prática. Este algoritmo foi por quase quarenta anos o único método para resolver problemas lineares. Entretanto, no ano de 1972, Klee e Minty [53] apresentaram um problema teórico com n variáveis e $2n$ restrições para o qual o método resolvia em tempo exponencial ($2^n - 1$ iterações), o que motivou a pesquisa por um método que tivesse o chamado desempenho polinomial.

Este objetivo foi alcançado por Khachian [51] em 1978, através de seu *Método de Elipsóides*. Ele obteve solução do problema linear em $O(n^2L)$ iterações requerendo um total de $O(n^4L)$ operações aritméticas, onde n é o número de variáveis e L representa o tamanho da palavra necessária para representar, em código binário,

os dados do problema. Todavia descobriu-se que, apesar do algoritmo desenvolvido por Khachian possuir propriedades teóricas interessantes, era extremamente lento em problemas reais.

Desse modo, podemos sintetizar a situação àquela época. Tínhamos, para a otimização linear:

- O algoritmo simplex, de desempenho excelente em problemas do mundo real, mas de péssimo desempenho em estudo de pior caso.
- O algoritmo de elipsóides, com belas propriedades teóricas, mas ineficiente na prática.

Assim, permanecia o desafio pela existência de um método com bons resultados teóricos e práticos.

Em 1984 Karmarkar [48] publicou um algoritmo alternativo com bons resultados teóricos e práticos. Mostrou, inclusive, que seu *Método Projetivo* em certos casos era mais eficiente que o algoritmo simplex. Ele obteve $O(nL)$ iterações e $O(n^{3.5}L)$ operações aritméticas. O mérito de Karmarkar foi considerar o problema linear como um caso particular do problema de *Otimização Não Linear*, nesse tempo considerada como linha de pesquisa distinta. Depois da apresentação do método e comprovada sua eficiência prática, foi descoberta uma relação muito estreita entre o algoritmo de Karmarkar e os métodos de barreira, ver por exemplo Gill et al. [33] e Renegar [76]. Esta relação levou ao renascimento dos métodos de pontos interiores e ao surgimento de métodos mais poderosos como os métodos primais-duais.

Usaremos a terminologia PPL para Problema de Programação Linear.

Forma-Padrão de um PPL

Para resolver um PPL alguns algoritmos exigem a apresentação de seu modelo em um formato denominado *forma-padrão*. Dizemos que um PPL encontra-se na forma-padrão quando seu modelo é dado por

$$\begin{array}{ll} \text{minimize}_x & c^T x \\ \text{sujeito a} & Ax = b \\ & x \geq 0 \end{array}$$

A redução de um PPL qualquer à forma-padrão é uma tarefa simples. Devido à equivalência entre as diferentes formulações, não há perda de generalidade em considerar apenas a forma-padrão. Computacionalmente, algumas formulações podem ser mais eficientes que outras para problemas e algoritmos específicos.

Dualidade, Condições de Otimalidade e Conjuntos Soluções

Consideremos o problema de programação linear

$$\begin{array}{lll} \text{(P)} & \text{minimize}_x & c^T x \\ & \text{sujeito a} & Ax = b \\ & & x \geq 0, \end{array}$$

onde x, c são vetores do $\mathbb{R}^n$, b é vetor do $\mathbb{R}^m$, e A é uma matriz $m \times n$.

Associado a (P) está outro problema de programação linear chamado *dual*, que consiste em

$$\begin{array}{lll} \text{(D)} & \text{maximize}_{(y,s)} & b^T y \\ & \text{sujeito a} & A^T y + s = c \\ & & s \geq 0, \end{array}$$

onde y é vetor do $\mathbb{R}^m$ e s é vetor do $\mathbb{R}^n$.

As componentes de y são chamadas *variáveis duais*, enquanto s é o vetor de *folgas duais*. O problema dual pode ser estabelecido mais compactamente eliminando s de (D) e reescrevendo as restrições como $A^T y \leq c$.

O problema (P) é chamado *primal* e os dois problemas juntos são referidos como *par primal-dual*.

Denotaremos os conjuntos das soluções viáveis e das soluções estritamente viáveis do problema (P) por $\mathcal{P}^+$ e $\mathcal{P}^{++}$, respectivamente. Similarmente, $\mathcal{D}^+$ e $\mathcal{D}^{++}$ indicam, respectivamente, os conjuntos das soluções viáveis e das soluções estritamente viáveis do problema (D). Portanto, temos

$$\begin{aligned} \mathcal{P}^+ &\equiv \{x \in \mathbb{R}^n \mid Ax = b,\ x \geq 0\}; \\ \mathcal{P}^{++} &\equiv \{x \in \mathbb{R}^n \mid Ax = b,\ x > 0\}; \\ \mathcal{D}^+ &\equiv \{(y,s) \in \mathbb{R}^m \times \mathbb{R}^n \mid A^T y + s = c,\ s \geq 0\}; \\ \mathcal{D}^{++} &\equiv \{(y,s) \in \mathbb{R}^m \times \mathbb{R}^n \mid A^T y + s = c,\ s > 0\}. \end{aligned}$$

Teorema 2.4.1. *([10], Teorema 3.3, p.92 - Teorema Fundamental da Programação Linear) Suponha que o conjunto $\mathcal{P}^+$ das soluções viáveis do problema (P) é não-vazio. Então, o problema (P) ou é ilimitado inferiormente, isto é, $c^T x \to -\infty$, ou existe um vértice v tal que*

$$c^T v \leq c^T x, \forall x \in \mathcal{P}^+.$$

As condições algébricas que devem ser satisfeitas pelas soluções de um PPL são derivadas de princípios primitivos da teoria da dualidade e podem também ser estabelecidas como casos especiais das condições de otimalidade para um problema

geral de otimização com restrições. Considerando a segunda opção, as condições de otimalidade para (P) derivam de (2.1)-(2.5) e são estabelecidas como segue.

Teorema 2.4.2. *([94], Capítulo 1, p.4 - Condições de Otimalidade) O vetor $x \in \mathbb{R}^n$ é uma solução de (P) se, e somente se, existem vetores $s \in \mathbb{R}^n$ e $y \in \mathbb{R}^m$ para os quais as seguintes condições acontecem:*

$$\begin{aligned} A^T y + s &= c && (2.6)\\ Ax &= b && (2.7)\\ x_i s_i &= 0, \quad i = 1, ..., n && (2.8)\\ (x, s) &\geq 0 && (2.9) \end{aligned}$$

Demonstração. É obtida como particularização dos Teoremas 2.3.1 e 2.3.2. □

Note que na equação anterior os multiplicadores de Lagrange são denotados pelos mesmos símbolos y e s - que usamos para as variáveis do problema dual (D). Isto não é por acaso, pois se aplicarmos as condições KKT ao problema dual, encontramos as mesmas condições estabelecidas em (2.6)-(2.9). Esta observação crucial define a relação entre os problemas primal e dual. Formalmente, as condições de otimalidade para o dual são estabelecidas como segue:

Teorema 2.4.3. *([94], Capítulo 1, p.4) O vetor $(y^*, s^*) \in \mathbb{R}^m \times \mathbb{R}^n$ é uma solução de (D) se, e somente se, existe um vetor $x^* \in \mathbb{R}^n$ tal que as condições (2.6)-(2.9) acontecem para $(x, y, s) = (x^*, y^*, s^*)$.*

Examinando as condições de KKT de ambos os pontos de vista, primal e dual, concluímos que

Corolário 2.4.1. *([94], Capítulo 1, p.4) Um vetor (x^*, y^*, s^*) resolve (2.6)-(2.9) se, e somente se, x^* resolve o problema primal (P) e (y^*, s^*) resolve o problema dual (D). O vetor (x^*, y^*, s^*) é chamado solução primal-dual.*

A teoria de dualidade explica a relação entre os dois problemas (P) e (D). O conjunto viável e o conjunto solução para o primal dizem-nos muito sobre o dual, e vice-versa.

Lema 2.4.1. *([59], Seção 4.2, p.89 - Lema da Dualidade Fraca) Dados quaisquer vetores viáveis x para (P) e (y, s) para (D), temos que*

$$c^T x \geq b^T y.$$

Demonstração. Sejam x viável para (P) e (y, s) viável para (D), então

$$c^T x - b^T y = c^T x - (Ax)^T y = c^T x - x^T A^T y = x^T (c - A^T y) = x^T s \geq 0$$

Portanto, $c^T x \geq b^T y$. □

Em outras palavras, a função objetivo dual dá um limite inferior para a função objetivo primal, e a primal dá um limite superior para a dual.

A função $h : (\mathcal{P}^+, \mathcal{D}^+) \to I\!R_+$ definida por

$$h(x, y, s) = c^T x - b^T y = x^T s,$$

é chamada *gap de dualidade.*

Se temos um ponto (x^*, y^*, s^*) que satisfaz todas as quatro condições (2.6)-(2.9) segue de $(x^*)^T s^* = 0$ que $c^T x^* = b^T y^*$, assim os valores ótimos para os problemas primal e dual coincidem. Analogamente, se temos (x^*, y^*, s^*) que satisfaz as condições (2.6), (2.7) e (2.9) e a igualdade $b^T y^* = c^T x^*$ então (x^*, y^*, s^*) resolve o problema primal-dual.

Supondo que o problema primal-dual tenha solução, denotamos o valor ótimo comum por Z^*, isto é,

$$b^T y^* = Z^* = c^T x^*. \tag{2.10}$$

Desde que $c^T x \geq c^T x^*$, $\forall x$ primal viável e $b^T y \leq b^T y^*$, $\forall (y, s)$ dual viável, podemos escrever

$$(x, y, s) \in \mathcal{F} \Rightarrow b^T y \leq Z^* \leq c^T x,$$

onde $\mathcal{F} = \{(x, y, s) \mid Ax = b,\ A^T y + s = c,\ (x, s) \geq 0\}$ é o conjunto primal-dual viável.

Usamos Ω_P e Ω_D para denotar, respectivamente, os conjuntos de soluções primal e dual:

$$\Omega_P = \{x^* \mid x^* \text{ resolve (P)}\}, \quad \Omega_D = \{(y^*, s^*) \mid (y^*, s^*) \text{ resolve (D)}\}$$

Pode-se ver que o conjunto solução primal-dual Ω é exatamente o produto cartesiano

$$\Omega = \Omega_P \times \Omega_D = \{(x^*, y^*, s^*) \text{ satisfazendo } (2.6) - (2.9)\}$$

Os conjuntos soluções Ω_P, Ω_D e Ω são todos fechados. Esta afirmação é facilmente verificada no caso de Ω_P se usamos a seguinte caracterização equivalente

$$\Omega_P = \{x^* \in I\!R^n \mid Ax^* = b,\ x^* \geq 0,\ c^T x^* = Z^*\}$$

onde Z^* é o valor objetivo ótimo definido em (2.10). Esta definição mostra que Ω_P é uma interseção de subconjuntos fechados do $I\!R^n$ e, portanto, é fechado. Lógica similar é aplicada para Ω_D e Ω.

A parte (b) do teorema seguinte é conhecida como *Teorema da Dualidade* da Programação Linear.

Teorema 2.4.4. *([94], Teorema 2.1, p.25) Os seguintes enunciados são verdadeiros.*

1. *Se os problemas primal e dual são ambos viáveis ($\mathcal{F} \neq \emptyset$), o conjunto Ω das soluções primal-dual é não-vazio.*

2. *Se um dos problemas (P) ou (D) tem uma solução ótima, o outro também tem, e os valores das funções objetivo são iguais.*

Pode acontecer que o problema primal ou dual (possivelmente ambos) seja inviável. Quando exatamente um dos dois problemas é inviável, o Teorema 2.4.4 diz-nos algo sobre o outro problema, como explicamos no resultado seguinte.

Corolário 2.4.2. *([94], Corolário 2.2, p.25) Suponha que o problema primal (P) é viável. Então sua função objetivo $c^T x$ é limitada inferiormente em sua região viável se, e somente se, o problema dual (D) é viável. Similarmente, suponha que o problema dual (D) é viável. Então sua função objetivo $b^T y$ é limitada superiormente em sua região viável se, e somente se, o problema primal (P) é viável.*

O Teorema 2.4.4 e o Corolário 2.4.2 indicam que as propriedades do conjunto solução primal Ω_P são estreitamente relacionadas com a existência de pontos viáveis para o problema dual (e, similarmente, Ω_D e o conjunto primal viável).

Uma condição obviamente utilizada em todos os métodos de pontos interiores primais-duais é a viabilidade estrita das variáveis primais e duais. Pode-se provar que esta condição é necessária e suficiente para obter limitação dos conjuntos de soluções ótimas Ω_P e $\{s^* \mid (y^*, s^*) \in \Omega_D$ para algum $y^* \in I\!R^m\}$.

Teorema 2.4.5. *([94], Teorema 2.3, p.26) Suponha que os problemas primal (P) e dual (D) são viáveis, isto é, $\mathcal{F} \neq \emptyset$. Então, se o problema dual tem um ponto estritamente viável, o conjunto de soluções primais Ω_P é não-vazio e limitado. Similarmente, se o problema primal tem um ponto estritamente viável, o conjunto*

$$\{s^* \mid (y^*, s^*) \in \Omega_D \quad \textit{para algum} \quad y^* \in I\!R^m\},$$

é não-vazio e limitado.

Capítulo 3

Métodos Afim Escala Generalizados

Neste capítulo apresentamos a classe de algoritmos afim escala primais para programação convexa com restrições lineares de Pinto et al. [73] e propomos uma classe de algoritmos afim escala duais para programação linear que generaliza, por meio de um parâmetro r, o algoritmo afim escala dual de Adler et al. [1].

3.1 Introdução

O algoritmo afim escala (o método primal) surge a partir de 1985 como uma simplificação do algoritmo que Karmarkar [48] publicou em 1984 e que foi objeto de estudo por vários matemáticos. Mais tarde descobriu-se que esse algoritmo já existia desde 1967, publicado por Dikin [26] na União Soviética, mas desconhecido pela comunidade científica ocidental.

Devido a sua importância teórica e prática vários artigos estudando a convergência global e local do algoritmo afim escala e sua associação com trajetórias contínuas têm sido publicados.

Nossa motivação para o trabalho com métodos afim escala generalizados vem de:

- Pinto et al. [73] (2002): propõe uma classe de algoritmos afim escala potencial para programação convexa com restrições lineares que generaliza alguns algoritmos conhecidos.

- Monteiro et al. [69] (1993): apresenta uma prova simplificada da convergência global do algoritmo afim escala primal aplicado a problemas de programação linear para a versão de passo longo.

- Saigal [83] (1993): apresenta a prova de convergência de uma generalização do

algoritmo afim escala primal aplicado a problemas de programação linear.

- Adler et al. [1] (1989): apresenta uma implementação em série de potências da variante afim escala dual para programação linear.

Citamos ainda:

- Cunha et al. [22] (2005): apresenta generalizações dos algoritmos afim escala primal e dual e onde são apresentados resultados computacionais que permitem fazer alguma comparação dos algoritmos afim escala generalizados com certos algoritmos afim escala clássicos.

3.2 Algoritmos Afim Escala

Entre todas as variantes relatadas do algoritmo original de Karmarkar, a aproximação *afim escala* atraiu especialmente a atenção dos pesquisadores. Esta aproximação usa uma simples *transformação afim* para substituir a *transformação projetiva* original de Karmarkar e permite trabalhar com o problema de programação linear na forma-padrão. A estrutura especial simplex requerida pelo algoritmo de Karmarkar é relaxada.

O algoritmo afim escala (o método primal) foi inicialmente introduzido em 1967 pelo matemático Russo I. Dikin [26], que publicou sua prova de convergência, sob a hipótese de não-degenerescência primal, em 1974 [27]. Em 1985, o algoritmo afim escala foi redescoberto pela comunidade ocidental por Barnes [8], Karmarkar e Ramakrishnan [49] e Vanderbei et al. [93]. Seguindo o algoritmo projetivo escala introduzido por Karmarkar [48] em 1984, eles propuseram o algoritmo afim escala para resolver o problema de programação linear na forma-padrão. Ambos Vanderbei et al. e Barnes assumiram não-degenerescência primal e dual na análise de convergência global, mas em oposição à versão elipsoidal de Barnes e Dikin, Vanderbei et al. estudaram a versão de passos longos do algoritmo afim escala para qualquer $\lambda \in (0, 1)$. Uma prova de convergência global (sem hipótese de não-degenerescência) foi dada por Tsuchiya e Muramatsu [91] para versão de passos longos com $\lambda \leq 2/3$. Uma prova mais simples deste resultado foi desenvolvida por Monteiro et al. [69]. Tsuchiya e Monteiro [90] desenvolveram também versão acelerada do método, que atinge convergência superlinear.

Este algoritmo simples funciona bem na prática, e vários resultados demonstrando boa performance computacional são referenciados em [1, 2, 70, 79]. Apesar de sua grande performance na prática, alguns problemas teóricos ou relacionados às implementações devem ser mencionados.

O primeiro problema a ser discutido é a convergência global, que é uma das propriedades fundamentais a ser mostrada do ponto de vista teórico. Vários artigos têm sido publicados sobre este problema [3, 8, 27, 93] sob várias escolhas de tamanho de passo e hipóteses de não-degenerescência; veja [39] para uma visão geral. Como no algoritmo simplex, esta análise torna-se particularmente difícil quando removemos a condição de não-degenerescência. Muitas das implementações eficientes adotam o procedimento de passo-longo para a escolha do tamanho-de-passo proposta por Vanderbei et al. [93] que determina a próxima iterada no ponto obtido procedendo uma proporção (fixa) $\lambda < 1$ da distância à fronteira. Usualmente λ é tomado entre 0.9 e 0.99.

O segundo problema concerne à terminação do algoritmo e à recuperação da solução ótima dual. Em contraste aos algoritmos de pontos interiores primais-duais [54, 55, 67, 68], o algoritmo afim escala primal gera uma seqüência apenas no espaço do problema primal, e não há solução dual viável calculada durante as iterações. Esta é uma séria desvantagem do algoritmo, desde que sem soluções duais viáveis, é muito difícil saber que as iteradas aproximam-se da face ótima para parar as iterações. Para remediar este problema, calculamos uma quantidade chamada *estimativa dual* que satisfaz apenas as restrições de igualdade linear do problema dual [8, 27, 93], esperando sua convergência para uma solução ótima do problema dual. Se a convergência é confirmada teoricamente sob hipóteses realísticas do ponto de vista de implementação, obtemos um significativo (e poderoso) critério de parada calculando o gap de dualidade. Entretanto, novamente a existência de degeneração torna o problema difícil. A convergência da estimativa dual é mostrada apenas sob hipótese de não-degenerescência em [8, 27, 93] ou, se não requeremos qualquer hipótese de não-degenerescência, em [89, 92] para versões de passo-curto, em [69, 91] para versões de passo-longo ou para versões contínuas em [3].

Um algoritmo similar, o chamado algoritmo afim escala dual, foi desenvolvido e implementado por Adler et al. [1] para problemas na forma de desigualdades. Eles ainda compararam sua implementação com o código simplex MINOS 4.1 [72]. O algoritmo afim escala dual está entre os primeiros métodos de pontos interiores a

ser mostrado como uma alternativa competitiva ao método simplex.

Apesar de seu sucesso na prática, nenhuma prova de tempo polinomial foi dada para as variantes primal ou dual do algoritmo.

Finalmente, no que se refere à complexidade, o trabalho de Megido e Shub [66] indicou que a trajetória conduzindo para a solução ótima dada pelos algoritmos afim escala dependem da solução de partida. Uma solução inicial ruim, que está muito próxima de um vértice do domínio viável, pode resultar em uma longa trajetória passando por todos os vértices. Todavia, a complexidade em tempo-polinomial dos algoritmos afim escala primal e dual pode ser restabelecida incorporando uma função barreira logarítmica na fronteira do octante positivo para evitar que uma solução interior seja "presa"pelo comportamento limite.

Ao longo desta direção, uma terceira variante, o chamado algoritmo afim escala primal-dual, foi apresentado e analisado por Monteiro et al. [68] e também por Kojima et al. [55]. A complexidade em tempo-polinomial foi alcançada com sucesso mas, por usar passos muito curtos, tornava o algoritmo impraticável.

Idéias Básicas dos Algoritmos Afim Escala

Os dois princípios fundamentais que guiam os algoritmos afim escala são:

- Se a solução interior atual está próxima do centro do politopo, então faz sentido mover na direção de máximo declive da função objetivo para alcançar um valor mínimo.

- Uma transformação apropriada é aplicada ao espaço solução tal que a solução interior atual é colocada próxima do centro no espaço solução transformado.

Na formulação de Karmarkar, a estrutura simplex

$$\Delta = \{x \in I\!R^n \mid x_1 + ... + x_n = 1, x_i \geq 0, i = 1, ..., n\}$$

e seu centro $e/n = (1/n, 1/n, ..., 1/n)^T$ foi introduzida propositalmente para realização das idéias acima. Quando trabalhamos diretamente com o problema na forma-padrão, a estrutura simplex não está disponível, e o domínio viável pode vir a ser um conjunto poliédrico ilimitado. Toda a estrutura restante é a interseção do espaço afim $\{x \in I\!R^n \mid Ax = b\}$ formado pelas restrições explícitas com o octante positivo $\{x \in I\!R^n \mid x \geq 0\}$ definido pelas restrições de não-negatividade. É óbvio que o octante não-negativo não tem um "centro"real. Entretanto, se nos posicionamos

no ponto $e = (1, 1, ..., 1)^T$, ao menos ainda guardamos distância igual de cada faceta ou "parede"do octante não-negativo. Quando a distância do movimento é menor que uma unidade, qualquer novo ponto que desloca de e permanece no interior do octante não-negativo. Conseqüentemente, se estamos habilitados a encontrar uma transformação apropriada que mapeie a solução interior atual para o ponto e, então, em paralelo com o algoritmo projetivo escala de Karmarkar, podemos estabelecer uma estratégia modificada como segue:

Tome uma solução interior, aplique a transformação apropriada ao espaço solução tal que mude a solução atual para e *no espaço transformado, e então caminhe na direção de máximo declive no espaço nulo transformado das restrições explícitas, mas não todo caminho para as barreiras de não-negatividade para manter a condição de uma solução interior. Então tome a transformação inversa para mapear a solução melhorada de volta para o espaço solução original como uma nova solução interior. Repita este processo até a otimalidade ou outras condições de parada serem encontradas.*

3.3 Algoritmo Afim Escala Primal Generalizado (GPAS)

Nesta seção, apresentamos a classe de algoritmos afim escala primais para programação convexa com restrições lineares de Pinto et al. [73]. Esta classe generaliza os algoritmos propostos por Gonzaga e Carlos [36] e por Eggermont [29], cuja convergência foi estudada por Iusem [41].

No caso particular de programação linear, esta classe é similar à classe afim escala potencial de Saigal [83], cujos resultados de convergência são válidos inclusive para o caso degenerado.

Limitamos-nos a apresentar o algoritmo e a verificar que ele é de pontos interiores. Os resultados de convergência obtidos para esta classe podem ser vistos em Pinto et al. [73], onde é também mostrado que a direção afim escala primal é o oposto do gradiente de f.

Consideremos o problema

$$\begin{array}{lll} \text{(PCRL)} & \text{minimize}_x & f(x) \\ & \text{sujeito a} & Ax = b \\ & & x \geq 0, \end{array}$$

onde a função objetivo $f : \mathbb{R}^n \to \mathbb{R}$ é uma função diferenciável e convexa, A é uma matriz $m \times n$ e b um vetor do $\mathbb{R}^m$.

Desde que f é convexa, x^* é uma solução para (PCRL) se, e somente se, existirem multiplicadores lagrangeanos $s^* \in \mathbb{R}^n$ e $y^* \in \mathbb{R}^m$ tais que

$$\begin{array}{rcl} A^T y^* + s^* & = & \nabla f(x^*) \\ Ax^* & = & b \\ x_i^* s_i^* & = & 0, \quad i = 1, ..., n \\ (x^*, s^*) & \geq & 0. \end{array} \tag{3.1}$$

Se $\left(AX_*^r A^T\right)^{-1}$ existe, alguns cálculos simples levam (3.1) para

$$\begin{array}{rcl} y(x^*) & = & \left(AX_*^r A^T\right)^{-1} AX_*^r \nabla f(x^*) \\ s(x^*) & = & \nabla f(x^*) - A^T y(x^*) \\ X_*^r s^* & = & 0, \end{array}$$

onde $X_* \equiv \mathrm{diag}(x^*)$ e r é um número maior ou igual a 1.

Portanto, é natural considerar as seguintes funções que são usadas na descrição e análise da classe afim escala primal. Para todo $x \in \mathbb{R}^n_{++}$, sejam

$$y(x) \equiv \left(AX^r A^T\right)^{-1} AX^r \nabla f(x) \tag{3.2}$$

$$s(x) \equiv \nabla f(x) - A^T y(x) \tag{3.3}$$

$$d(x) \equiv X^r s(x), \tag{3.4}$$

onde $X \equiv \mathrm{diag}(x)$ e $r \geq 1$.

Sob a hipótese de não-degenerescência primal, todas as funções acima podem ser estendidas para $\mathcal{P}^+ = \{x \in \mathbb{R}^n \mid Ax = b,\ x \geq 0\}$.

O pseudocódigo abaixo apresenta o algoritmo afim escala primal generalizado (GPAS) descrito em [73].

Algoritmo GPAS:

inicialização: Dados $(A, b, c, x^0 \in \mathcal{P}^{++}$, *critério de parada*, $\beta > 0$ e $\delta \in (0,1))$

$k = 0$

enquanto *critério de parada* não satisfeito

$$\begin{array}{rcl} y^k & = & \left(AX_k^r A^T\right)^{-1} AX_k^r \nabla f(x^k) \\ s^k & = & \nabla f(x^k) - A^T y^k \\ d^k & = & X_k^r s^k \\ \mu^k & = & \chi\left(X_k^{-1} d^k, 0\right) \\ \alpha^k & = & \frac{\delta}{\beta + \mu^k} \\ t^k & = & \arg\min_{t \in [0, \alpha^k]} f(x^k - t d^k) \\ x^{k+1} & = & x^k - t^k d^k \\ k & = & k + 1 \end{array}$$

fim enquanto

fim Algoritmo GPAS

onde $\chi(v)$ denota a maior componente de v.

As direções propostas por Eggermont [29] e por Gonzaga e Carlos [36] são casos particulares das direções produzidas pelo algoritmo GPAS quando $r = 1$ ou $r = 2$, respectivamente.

Uma conseqüência imediata é o fato de que o algoritmo é um método de pontos interiores.

De fato, para $x^k \in \mathcal{P}^{++}$, $k \geq 0$, ocorre

$$\left(X_k^{-1} x^{k+1}\right)_i = 1 - t^k (x_i^k)^{-1} d_i^k \geq 1, \quad \text{se} \quad d_i^k \leq 0,$$

caso contrário, temos

$$t^k (x_i^k)^{-1} d_i^k \leq \frac{\delta \mu^k}{\beta + \mu^k} < 1.$$

Portanto, $1 - t^k (x_i^k)^{-1} d_i^k > 0$. Desde que $x^0 > 0$, a afirmação segue por indução.

Para garantir um largo intervalo de busca $[0, \alpha^k]$, os *fatores de segurança* δ e β devem ser, respectivamente, próximo de 1 e próximo de 0.

A classe de algoritmos afim escala primal acima é analisada em [73] sob as seguintes hipóteses:

H1: posto$(A) = m$;

H2: f é convexa e continuamente diferenciável;

H3: O problema (PCRL) tem uma solução interior viável, isto é, $\mathcal{P}^{++} \neq \emptyset$;

H4: O conjunto de nível $niv_f(f(x^0)) \equiv \{x \in \mathcal{P}^{++} \mid f(x) \leq f(x^0)\}$ é limitado;

H5: O problema (PCRL) é não-degenerado.

Os resultados de convergência obtidos foram de convergência fraca, a saber:

1. A seqüência $\{f(x^k)\}$ é monótona decrescente e convergente.
2. $X_k s^k \to 0$.
3. A seqüência $\{x^k\}$ é limitada.
4. $(x^{k+1} - x^k) \to 0$.
5. Qualquer ponto de acumulação de $\{x^k\}$ resolve (PCRL).

3.4 Algoritmo Afim Escala Dual Generalizado (GDAS)

Nesta seção, propomos uma classe de algoritmos afim escala duais para programação linear que generaliza, por meio de um parâmetro r, o algoritmo afim escala dual de Adler et al. [1]. Esta classe é construída através da transformação dada pela potência S^{-r}, $r \geq 1$, onde S é a matriz diagonal do vetor de iteradas.

O algoritmo afim escala dual tem mostrado boa performance na prática em problema de programação linear [1, 2, 50, 70], problemas de fluxo em redes de larga-escala [79] e problemas de atribuição de larga-escala [77, 78].

Como o primal, o algoritmo afim escala dual também consiste de três partes essenciais, a saber, começando com uma solução dual viável interior, mover para uma solução interior melhor, e parar com uma solução dual ótima.

Dados c e x vetores do $\mathbb{R}^n$, b e y vetores do $\mathbb{R}^m$ e A uma matriz $m \times n$, consideraremos o problema de programação linear

$$\begin{array}{lll} \text{(PL)} & \text{minimize}_x & c^T x \\ & \text{sujeito a} & Ax = b \\ & & x \geq 0, \end{array}$$

e seu dual

$$\begin{array}{lll} \text{(DL)} & \text{maximize}_y & b^T y \\ & \text{sujeito a} & A^T y \leq c. \end{array}$$

Introduzindo variáveis de folga à formulação de (DL), temos

$$\begin{array}{lll} \text{(DL)} & \text{maximize}_{(y,s)} & b^T y \\ & \text{sujeito a} & A^T y + s = c \\ & & s \geq 0, \end{array}$$

onde s é o n-vetor de variáveis de folga.

O algoritmo afim escala dual resolve o problema de programação linear (PL), resolvendo indiretamente seu dual (DL).

O algoritmo começa com uma solução inicial $y^0 \in \mathbb{R}^m$ tal que $(y^0, s^0) \in \mathcal{D}^{++}$, onde

$$\mathcal{D}^{++} = \left\{(y, s) \in \mathbb{R}^m \times \mathbb{R}^n \mid \ A^T y + s = c, \ s > 0\right\},$$

e gera uma seqüência de pontos interiores viáveis $\{y^1, y^2, ..., y^k, ...\}$ que crescem monotonicamente o valor da função objetivo, isto é,

$$c - A^T y^k > 0$$

$$b^T y^{k+1} > b^T y^k,$$

terminando quando um *critério de parada* a ser discutido depois é satisfeito.

Como usual, nossa variante afim do algoritmo de Karmarkar consiste de uma operação de escala aplicada a (DL), seguido por um procedimento de busca que determina a próxima iterada. Em cada iteração k, com y^k e s^k como iteradas correntes, a transformação linear aplicada ao espaço solução, denominada *transformação afim escala dual*, é dada por

$$\hat{s} = S_k^{-r/2} s,$$

onde $S_k = \text{diag}(s^k)$ e r é um número maior ou igual a 1.

Em termos das variáveis de folga escaladas, o problema (DL), pode ser reescrito como

$$\begin{array}{lll} (\hat{\text{DL}}) & \text{maximize}_{(y,s)} & b^T y \\ & \text{sujeito a} & A^T y + S_k^{r/2} \hat{s} = c \\ & & \hat{s} \geq 0 \end{array} \tag{3.5}$$

Sob a hipótese de posto completo para A^T, há uma relação bijetiva entre o conjunto das soluções viáveis para (DL),

$$Y = \{y \in I\!R^m \mid A^T y \leq c\}$$

e o conjunto das folgas viáveis escaladas para ($\hat{\text{DL}}$),

$$\hat{S} = \{\hat{s} \in I\!R^n \mid \exists y \in Y, A^T y + S_k^{r/2} \hat{s} = c\},$$

dada por

$$\hat{s}(y) = S_k^{-r/2}(c - A^T y) \tag{3.6}$$

e

$$y(\hat{s}) = (A S_k^{-r} A^T)^{-1} A S_k^{-r/2}(S_k^{-r/2} c - \hat{s}). \tag{3.7}$$

Similarmente, existe também uma correspondência bijetiva relacionando as direções dy em Y e $d\hat{s}$ em $\hat{S}$, com

$$d\hat{s} = -S_k^{-r/2} A^T dy \tag{3.8}$$

e

$$dy = -(A S_k^{-r} A^T)^{-1} A S_k^{-r/2} d\hat{s}. \tag{3.9}$$

Observe de (3.8) que a direção viável em $\hat{S}$ pertence ao espaço imagem de $S_k^{-r/2} A^T$.

Como em outras apresentações de variantes afim do algoritmo de Karmarkar, o gradiente projetado da função objetivo com respeito às variáveis escaladas é a direção de busca selecionada em cada iteração.

Desde que apenas as variáveis de folga são mudadas pela transformação afim, usando (DL) e (3.7), o gradiente da função objetivo com respeito a $\hat{s}$ é,

$$\nabla_{\hat{s}} b(y(\hat{s})) = (\nabla_{\hat{s}} y(\hat{s}))^T \nabla_y b(y) = -S_k^{-r/2} A^T (A S_k^{-r} A^T)^{-1} b,$$

que pertence ao espaço imagem de $S_k^{-r/2} A^T$. Conseqüentemente, a projeção é desnecessária e a direção de busca em $\hat{S}$ é dada por

$$d\hat{s} = -S_k^{-r/2} A^T (A S_k^{-r} A^T)^{-1} b. \tag{3.10}$$

De (3.9) e (3.10), calculamos a direção viável correspondente em Y,

$$dy = (A S_k^{-r} A^T)^{-1} b. \tag{3.11}$$

A direção viável correspondente para as folgas não-escaladas é obtida aplicando a transformação afim inversa a $d\hat{s}$,

$$ds = -A^T (A S_k^{-r} A^T)^{-1} b. \tag{3.12}$$

Ilimitação é detectada se $ds \geq 0$, no caso de $b \neq 0$. De outra forma, a próxima iterada é calculada tomando o máximo passo viável na direção ds, e retraindo de volta para um ponto interior de acordo com um *fator de segurança* γ, $0 < \gamma < 1$, isto é,

$$y^{k+1} = y^k + \alpha dy, \tag{3.13}$$

onde

$$\alpha = \gamma \times \min\{-s_i^k/(ds)_i \mid (ds)_i < 0,\ i = 1, ..., n\}. \tag{3.14}$$

Em cada iteração, uma tentativa de solução primal é dada por

$$x^k = S_k^{-r} A^T (A S_k^{-r} A^T)^{-1} b. \tag{3.15}$$

É fácil checar que $Ax^k = b$, mas x^k pode não ser necessariamente positivo.

A formulação acima segue para projeções inexatas sem perda de viabilidade, ou seja, mesmo que dy não seja calculada exatamente em (3.11), ainda podemos obter um par de direções viáveis calculando

$$ds = -A^T dy. \tag{3.16}$$

O algoritmo afim escala dual generalizado (GDAS) esboçado acima pode ser descrito no seguinte pseudocódigo.

Algoritmo GDAS:

inicialização: Dados $(A,\ b,\ c,\ y^0 \mid (y^0, s^0) \in \mathcal{D}^{++},$ *critério de parada* e $\gamma \in (0,1))$

$k = 0$

enquanto *critério de parada* não satisfeito

$$\begin{aligned} s^k &= c - A^T y^k \\ dy^k &= \left(AS_k^{-r}A^T\right)^{-1} b \\ ds^k &= -A^T dy^k \\ \text{se } & ds^k \geq 0 \rightarrow \textbf{pare} \\ \alpha_k &= \gamma \times \min\left\{-s_i^k/(ds^k)_i \mid (ds^k)_i < 0,\ i = 1, ..., n\right\} \\ y^{k+1} &= y^k + \alpha_k dy^k \\ k &= k+1 \end{aligned}$$

fim enquanto

fim Algoritmo GDAS

Quando $r = 2$, o algoritmo GDAS reduz-se ao algoritmo afim escala dual de Adler et al. [1].

3.5 Convergência do Algoritmo GDAS

Provaremos agora a convergência do algoritmo afim escala dual generalizado sob as seguintes hipóteses:

H1: posto$(A) = m$;

H2: $b \neq 0$;

H3: O problema (DL) tem uma solução interior viável, isto é, $\mathcal{D}^{++} \neq \emptyset$;

H4: O problema (PL) tem uma solução viável;

H5: O problema (DL) é não-degenerado.

Introduzimos agora algumas funções que são usadas na descrição e na análise do algoritmo afim escala dual generalizado. Para todo $s \in \mathbb{R}^n_{++}$, sejam

$$x(s) \equiv S^{-r}A^T(AS^{-r}A^T)^{-1}b \tag{3.17}$$

$$dy(s) \equiv (AS^{-r}A^T)^{-1}b \tag{3.18}$$

$$ds(s) \equiv -A^T(AS^{-r}A^T)^{-1}b = -A^T dy(s) = -S^r x(s), \tag{3.19}$$

onde $S \equiv \text{diag}(s)$ e $r \geq 1$.

$x(s)$ é chamada *estimativa primal* associada com o ponto s e o par $(dy(s), ds(s))$ é referido como par de *direções afim escala dual* associado com s.

É fácil notar que a hipótese H1 implica que a inversa de $AS^{-r}A^T$ existe para todo $s > 0$.

De fato, suponha que a inversa de $AS^{-r}A^T$ não exista para algum $s > 0$, ou seja,

$$AS^{-r}A^T y = 0 \text{ para algum } y \in \mathbb{R}^m,\ y \neq 0$$

Assim, $\exists y \in \mathbb{R}^m,\ y \neq 0$ tal que

$$0 = y^T AS^{-r}A^T y = (S^{-r/2}A^T y)^T S^{-r/2}A^T y = \left\|S^{-r/2}A^T y\right\|^2.$$

Logo, devemos ter $S^{-r/2}A^T y = 0$. Desde que $S^{-r/2}$ é invertível, esta igualdade implica em $A^T y = 0$. Portanto, $\sum_{i=1}^{m} y_i \left(A^T\right)_i = 0$, com $y_i \neq 0$ para algum i, onde $\left(A^T\right)_i$ representa a $i-ésima$ coluna de A^T. Absurdo, pois as linhas de A, e portanto as colunas de A^T, são L.I., já que $posto(A) = m$. Logo $AS^{-r}A^T$ é invertível.

Note ainda que as hipóteses H1-H4 implicam que se $(y, s) \in \mathcal{D}^{++}$ então ds deve ter ao menos uma componente negativa.

De fato, supondo que $\exists (y, s) \in \mathcal{D}^{++}$ tal que $ds \geq 0$, temos:

Se $ds = 0$, temos $-A^T dy = 0$. Desde que A tem posto m, esta igualdade implica em $dy = 0$. Assim, $(AS^{-r}A^T)^{-1}b = 0$ e, então, $b = 0$. Mas, isto contradiz a hipótese H2.

Se $ds \geq 0$, com $ds \neq 0$, definindo $y(t) = y + tdy$ e $s(t) = s + tds$, temos $(y(t), s(t)) \in \mathcal{D}^{++}$, para $t \geq 0$. Quando $t \to +\infty$, temos $b^T y(t) = b^T y + tb^T dy = b^T y + tb^T (AS^{-r}A^T)^{-1}b \to +\infty$, pois $b^T (AS^{-r}A^T)^{-1}b > 0$ por hipóteses H1 e H2 e, pelo Corolário 2.4.2, isto contradiz a hipótese H4.

Assim, a expressão que determina y^{k+1} no algoritmo afim escala está bem-definida.

O resultado seguinte fornece caracterizações da direção afim escala dual $dy(s)$ e da estimativa primal $x(s)$ como soluções ótimas de certos problemas de programação quadrática.

Proposição 3.5.1. *As seguintes propriedades são verdadeiras:*

(a) *Para todo $s > 0$, $dy(s) = \sqrt{b^T (AS^{-r}A^T)^{-1}b}\, u(s)$, onde $u(s)$ é a única solução do seguinte problema de Programação Quadrática:*

$$\begin{array}{ll} maximize_u & b^T u \\ sujeito\ a & u^T AS^{-r}A^T u \leq 1. \end{array} \tag{3.20}$$

(b) *Para todo* $s > 0$, $x(s)$ *é a única solução do seguinte problema de Programação Quadrática:*

$$\begin{array}{ll} minimize_x & \frac{1}{2}\left\|S^{r/2}x\right\|^2 \\ sujeito\ a & Ax = b. \end{array} \tag{3.21}$$

onde $S \equiv diag(s)$.

Demonstração. Desde que a solução de (3.20) pertencerá a fronteira do elipsóide, usando a teoria lagrangeana, esta solução é

$$\begin{aligned} u(s) &= \tfrac{1}{\lambda(s)}(AS^{-r}A^T)^{-1}b \\ \lambda(s) &= \sqrt{b^T(AS^{-r}A^T)^{-1}b} \end{aligned}$$

e, então, para cada s,

$$dy(s) = \lambda(s)u(s)$$

Assim, segue a parte (a).

Para provar a parte (b), pelo Teorema dos Multiplicadores de Lagrange, se $x^*(s)$ é solução de (3.21), existe $\lambda^*(s) \in I\!R^m$ tal que

$$S^r x^*(s) + A^T\lambda^*(s) = 0 \tag{3.22}$$

$$Ax^*(s) = b \tag{3.23}$$

Multiplicando (3.22) por AS^{-r} e usando (3.23), obtemos

$$\lambda^*(s) = -(AS^{-r}A^T)^{-1}b$$

Agora, substituindo $\lambda^*(s)$ em (3.22) multiplicada por S^{-r}, obtemos

$$x^*(s) = S^{-r}A^T(AS^{-r}A^T)^{-1}b$$

Portanto, a parte (b) também ocorre. □

Lema 3.5.1.

$$\left\|S^{-r/2}ds(s)\right\|^2 = \left\|S^{r/2}x(s)\right\|^2 = b^T(AS^{-r}A^T)^{-1}b > 0, \quad \forall s > 0, \tag{3.24}$$

onde $S \equiv diag(s)$.

Demonstração. A primeira igualdade é uma conseqüência imediata do fato que $d(s) = -S^r x(s)$, dado em (3.19). A segunda igualdade pode ser obtida do seguinte cálculo simples:

$$\begin{aligned} \left\|S^{-r/2} ds(s)\right\|^2 &= \left\|-S^{-r/2} A^T (AS^{-r}A^T)^{-1} b\right\|^2 \\ &= b^T (AS^{-r}A^T)^{-1} AS^{-r/2} S^{-r/2} A^T (AS^{-r}A^T)^{-1} b \\ &= b^T (AS^{-r}A^T)^{-1} b \end{aligned}$$

onde a primeira equação segue de (3.19).

A desigualdade em (3.24) segue da hipótese H1, que implica que $AS^{-r}A^T$ é definida positiva, e da hipótese H2. □

Teorema 3.5.1. *Seja $\{y^k\}$ a seqüência gerada pelo algoritmo GDAS. Então,*

(a) $s^k > 0$, *para cada* $k \geq 0$.

(b) $\{b^T y^k\}$ *é uma seqüência monótona crescente e convergente.*

(c) $\sum_{k=0}^{\infty} b^T (y^{k+1} - y^k) < \infty$.

Demonstração. A parte (a) segue por indução. Desde que $s^0 > 0$, é suficiente provar que $s^k > 0$ implica que $s^{k+1} > 0$. Da descrição do algoritmo e (3.19), temos

$$s^{k+1} = c - A^T y^{k+1} = c - A^T (y^k + \alpha_k dy^k) = s^k - \alpha_k A^T dy^k = s^k + \alpha_k ds^k.$$

Assuma que $s^k > 0$. Se $ds_i^k \geq 0$, então $s_i^k + \alpha_k ds_i^k > 0$, $\forall \alpha_k \geq 0$. Caso contrário, se $ds_i^k < 0$, então $s_i^k + \alpha_k ds_i^k > 0$ se, e somente se, $\alpha_k < -\frac{s_i^k}{ds_i^k}$, que é satisfeita pela escolha de α_k.

Para obter a parte (b), usamos (3.18) e o Lema 3.5.1 para ter

$$\begin{aligned} b^T y^{k+1} &= b^T y^k + \frac{\gamma}{\chi[-S_k^{-1} ds^k]} b^T (AS_k^{-r} A^T)^{-1} b \\ &\geq b^T y^k + \gamma \frac{\|S_k^{-r/2} ds^k\|^2}{\|S_k^{-1} ds^k\|}. \end{aligned} \tag{3.25}$$

De (3.24), o termo adicional na fórmula acima é positivo. Isto implica que $b^T y^{k+1} > b^T y^k$. Portanto, $\{b^T y^k\}$ é monótona crescente. Devido à hipótese H4, esta seqüência é limitada e, portanto, converge, digamos para b^*.

Finalmente, $\sum_{k=0}^{\infty} b^T (y^{k+1} - y^k) = b^* - b^T y^0$, onde $b^T y^k \to b^*$, e isto prova a parte (c). □

Os seguintes lemas técnicos serão usados na demonstração da Proposição 3.5.2.

Lema 3.5.2. *São dados um vetor $c \in \mathbb{R}^k$ com $c \neq 0$, uma matriz $Q \in \mathbb{R}^{k\times l}$ de posto k e uma matriz diagonal $D \in \mathbb{R}^{l\times l}$ com entradas diagonais positivas. Então existe uma constante $p(Q,c) > 0$, independente de D, tal que se x^* resolve*

$$\begin{array}{ll} maximize_x & c^T x \\ sujeito\ a & x^T QDQ^T x \leq 1 \end{array} \tag{3.26}$$

então

$$|c^T x^*| \geq p(Q,c)\|x^*\|. \tag{3.27}$$

Demonstração. Veja em [83]. □

Lema 3.5.3. *(Lema de Hoffman) Seja $F \in \mathbb{R}^{p\times q}$ dado. Então existe uma constante $C = C(F)$ com a seguinte propriedade: para $f \in \mathbb{R}^p$ tal que o sistema $Fw = f$ é viável e $z \in \mathbb{R}^q$, existe uma solução $\bar{w}$ de $Fw = f$ tal que*

$$\|\bar{w} - z\| \leq C\|f - Fz\|$$

Demonstração. Veja em [57]. □

Lema 3.5.4. *Sejam $H \in \mathbb{R}^{l\times p}$ e $h \in R^l$ dados tal que $h \in Im(H)$. Então existe uma constante $M > 0$ tal que para toda matriz diagonal $D > 0$, a (única) solução ótima $\bar{w} = \bar{w}(D) \in \mathbb{R}^p$ do problema*

$$\begin{array}{ll} minimize_w & \|Dw\| \\ sujeito\ a & Hw = h \end{array} \tag{3.28}$$

satisfaz $\|\bar{w}\| \leq M$.

Demonstração. Assuma por contradição que existe uma seqüência de matrizes diagonais $D^k \succ 0$ tal que a solução $w^k = w\left(D^k\right)$ de (3.28) satisfaz

$$\lim_{k\to\infty} \left\|w^k\right\| = \infty$$

Isto implica que existe uma constante $L > 0$, um conjunto de índices $J \subseteq \{1,2,...,p\}$, $J \neq \emptyset$, e uma subseqüência $\{w^k\}_{k\in K}$ com a propriedade que

$$\left|w_j^k\right| \leq L, \quad \forall j \notin J,\ \forall k \in K; \tag{3.29}$$

$$\lim_{k\in K} \left|w_j^k\right| = \infty, \quad \forall j \in J. \tag{3.30}$$

Dado um inteiro $k \geq 0$, considere o sistema

$$Hw = h \tag{3.31}$$

$$w_j = w_j^k, \quad \forall j \notin J \tag{3.32}$$

e observe que w^k é uma solução deste sistema. Pelo Lema de Hoffman, com $z = 0$ e usando a norma da soma do lado direito, esse sistema tem uma solução $\bar{w}^k$ tal que

$$\left\|\bar{w}^k\right\| \leq L_1(\|h\|_1 + \sum_{j \notin J} \left|w_j^k\right|) \tag{3.33}$$

onde L_1 é uma constante independente de k. Segue de (3.29) e (3.33) que

$$\left\|\bar{w}^k\right\| \leq L_1(\|h\|_1 + (p - |J|)L) \equiv L_2, \quad \forall k \in K, \tag{3.34}$$

que mostra que $\bar{w}^k$ é limitada. Diante de (3.30), existe um inteiro $k_0 \geq 0$ tal que

$$\left|w_j^k\right| > L_2, \quad \forall k \geq k_0, \ k \in K, \ \forall j \in J. \tag{3.35}$$

Então, segue de (3.32), (3.34) e (3.35) que

$$\begin{aligned}
\left\|D^k \bar{w}^k\right\|^2 &= \sum_{j \in J} (d_{jj}^k \bar{w}_j^k)^2 + \sum_{j \notin J} (d_{jj}^k \bar{w}_j^k)^2 \\
&\leq \sum_{j \in J} (d_{jj}^k L_2)^2 + \sum_{j \notin J} (d_{jj}^k w_j^k)^2 \\
&< \sum_{j \in J} (d_{jj}^k w_j^k)^2 + \sum_{j \notin J} (d_{jj}^k w_j^k)^2 \\
&= \left\|D^k w^k\right\|^2, \quad \forall k \geq k_0, \ k \in K,
\end{aligned}$$

que junto com (3.31) contradiz o fato que w^k é uma solução ótima de (3.28) com $D = D^k$. □

De agora em diante, denotamos a seqüência das estimativas primais $\{x(s^k)\}$ simplesmente por $\{x^k\}$.

Proposição 3.5.2. *As seguintes propriedades valem para o algoritmo afim escala dual generalizado:*

(a) *A seqüência $\{(y^k, s^k)\}$ converge.*

(b) *Existe* $\rho > 0$ *tal que para todo* $k = 1, 2, ...$

$$\frac{b^* - b^T y^k}{\|y^* - y^k\|} \geq \rho,$$

onde $b^* = \lim_{k \to +\infty} \{b^T y^k\}$.

(c) *A seqüência* $\{x^k\}$ *é limitada.*

(d) *Para cada* $r \geq 1$, $X_k s^k \to 0$.

Demonstração. Da parte (a) da Proposição 3.5.1, $dy^k = \lambda_k u^k$, onde u^k resolve

$$\begin{array}{ll} \text{maximize}_u & b^T u \\ \text{sujeito a} & u^T A S_k^{-r} A^T u \leq 1 \end{array}$$

Do Lema 3.5.2, $\exists q(A,b) > 0$ tal que $|b^T u^k| \geq q(A,b)\|u^k\|$.

Então,

$$|b^T dy^k| = \lambda_k |b^T u^k| \geq \lambda_k q(A,b)\|u^k\| = q(A,b)\|dy^k\|.$$

Da parte (c) do Teorema 3.5.1, segue, observando que $b^T dy^k > 0$,

$$\begin{aligned} \infty &> \sum_{k=0}^{\infty} b^T (y^{k+1} - y^k) = \sum_{k=0}^{\infty} \alpha_k b^T dy^k \\ &\geq \sum_{k=0}^{\infty} \alpha_k q(A,b)\|dy^k\| = q(A,b) \sum_{k=0}^{\infty} \|y^{k+1} - y^k\| \end{aligned}$$

Assim, $\{y^k\}$ é de Cauchy e, portanto, converge. Desde que para cada k, $s^k = c - A^T y^k$, a seqüência $\{s^k\}$ também converge. Isto prova a parte (a).

Para provar a parte (b), usando a desigualdade triangular, obtemos

$$\begin{aligned} b^* - b^T y^k &= \sum_{j=0}^{\infty} b^T (y^{k+1+j} - y^{k+j}) \geq q(A,b) \sum_{j=0}^{\infty} \|y^{k+1+j} - y^{k+j}\| \\ &\geq q(A,b)\|y^* - y^k\|. \end{aligned}$$

A parte (c) segue da parte (b) da Proposição 3.5.1 e Lema 3.5.4.

Finalmente, para provar a parte (d), desde que $\{b^T y^k\}$ converge e usando (3.25), temos

$$\gamma \frac{\|S_k^{-r/2} ds^k\|^2}{\|S_k^{-1} ds^k\|} = \gamma \frac{\|S_k^{r/2} x^k\|^2}{\|S_k^{r-1} x^k\|} \to 0, \text{ com } 0 < \gamma < 1.$$

Desde que $\{s^k\}$ converge e $\{x^k\}$ é limitada, quando $r \geq 1$, o denominador da expressão acima é limitado. Então $S_k^{r/2} x^k \to 0$. Mas isto só é possível se $S_k x^k \to 0$.

□

Mostraremos, agora, que com a hipótese de não-degenerescência dual, a seqüência $\{x^k\}$ também converge, e que os pontos limites das seqüências primais e duais são ótimos para seus respectivos problemas.

Teorema 3.5.2. *Assuma que as hipóteses H1 – H5 ocorrem. Então, existem vetores x^*, y^* e s^* tais que*

(a) $x^k \to x^*$.

(b) $y^k \to y^*$.

(c) $s^k \to s^*$.

onde x^* é uma solução ótima do problema primal e (y^*, s^*) é uma solução ótima do problema dual.

Demonstração. Usando a parte (a) da Proposição 3.5.2, seja (y^*, s^*) um ponto de acumulação da seqüência (y^k, s^k). É claro que este ponto limite pertence à fronteira do poliedro do problema de programação linear. Defina os conjuntos:

$$\begin{aligned} B &= \{j \mid s_j^* = 0\} \\ N &= \{j \mid s_j^* > 0\}. \end{aligned}$$

Então, $A_B^T y^* = c_B$. Como todas as soluções duais são não-degeneradas, A_B tem posto coluna completo m.
Da parte (c) da Proposição 3.5.2, seja x^* um ponto de acumulação qualquer da seqüência $\{x^k\}$. Desde que $x_j^k s_j^k \to 0$, pela parte (d) da Proposição 3.5.2, então $x_j^* = 0$ para cada $j \notin B$. Assim,

$$b = Ax^* = A_B x_B^*.$$

Desde que A_B tem posto coluna completo m, este sistema tem uma única solução. Mas cada ponto de acumulação x^* de $\{x^k\}$ resolve este sistema, então a seqüência tem um único ponto de acumulação x^*, e assim

$$x^k \to x^*.$$

Agora, assuma que $j \in B$. Se $x_j^* < 0$, então existe um inteiro $L \geq 1$ tal que $x_j^k < 0$, $\forall k \geq L$. Mas, $0 > x_j^k = -(s_j^k)^{-r} ds_j^k$, que implica $ds_j^k > 0$. Então,

$$s_j^{k+1} = s_j^k + \alpha_k ds_j^k > s_j^k.$$

Assim $s_j^k \nrightarrow 0$, que contradiz $j \in B$. Então, $x^* \geq 0$. Desde que, y^* e s^* são viáveis para o dual, e x^* é viável para o primal e eles satisfazem as condições de folgas complementares, a prova está completa. □

Capítulo 4

Experimentos Numéricos com os Algoritmos GPAS e GDAS

Neste capítulo, verificamos a eficiência das classes de algoritmos afim escala, apresentadas no Capítulo 3, através de vários experimentos numéricos. Todos os experimentos foram realizados em um PC *Pentium* III com CPU 866MHz e 256MB de memória e implementados no *MATLAB* versão 6.0 no ambiente *Windows XP*.

Com o objetivo de observar o comportamento com relação ao parâmetro r e comparar com os algoritmos clássicos (quando $r = 1$ ou $r = 2$), implementamos as famílias de algoritmos GPAS e GDAS propostas aplicadas a alguns problemas testes.

4.1 Problemas Testados

Foram realizados testes com problemas de programação linear de pequeno porte da biblioteca NETLIB. No caso do GPAS, testamos também alguns problemas de programação quadrática do repositório Maros e Mészáros [60]. Neste caso, a função objetivo é dada por

$$f(x) = \frac{1}{2}x^T Qx + c^T x + c_0,$$

onde Q é uma matriz $n \times n$ simétrica semidefinida positiva.

A Tabela 4.1 fornece os dados dos problemas lineares e a Tabela 4.2 fornece os dados dos problemas quadráticos. As dimensões dos problemas são dadas nas colunas 2 e 3, e incluem as variáveis de folgas. O número de linhas não inclui a função objetivo. Na coluna 4, é mostrado o número de elementos não-nulos de A. As colunas 5 e 6 da tabela 4.2 fornecem, respectivamente, o número de variáveis quadráticas e o número de entradas fora da diagonal na parte triangular inferior

da matriz Q. A última coluna de cada tabela dá o valor ótimo fornecido pelas bibliotecas.

Tabela 4.1: Dados dos problemas lineares.

Problema	Lin	Col	NZ	Valor Ótimo
adlittle	56	138	424	2.2549496e+05
afiro	27	51	102	-4.6475314e+02
bandm	305	472	2494	-1.5862802e+02
blend	74	114	522	-3.0812150e+01
israel	174	316	2443	-8.9664482e+05
kb2	52	77	331	-1.7499001e+03
sc105	105	163	340	-5.2202061e+01
sc205	205	317	665	-5.2202061e+01
sc50a	50	78	160	-6.4575077e+01
sc50b	50	78	148	-7.0000000e+01
scagr7	129	185	465	-2.3313893e+06
share1b	117	253	1179	-7.6589319e+04
share2b	96	162	777	-4.1573224e+02
stocfor1	117	165	501	-4.1131976e+04

Tabela 4.2: Dados dos problemas quadráticos.

Problema	Lin	Col	NZ	QN	QNZ	Valor Ótimo
genhs28	8	20	48	20	46	9.2717369e-01
hs118	47	62	128	15	0	6.6482045e+02
hs21	5	7	11	2	0	-9.9960000e+01
hs35	1	4	4	3	2	1.1111111e-01
hs76	3	7	13	4	2	-4.6818182e+00
lotschd	7	12	54	6	0	2.3984159e+03
qpcblend	74	114	522	83	0	-7.8425409e-03
qptest	3	5	8	2	1	4.3718750e+00
zecevic2	4	6	10	1	0	-4.1250000e+00

4.2 Solução Inicial e Critério de Parada

Os algoritmos propostos requerem que uma solução viável inicial seja fornecida. Para obter uma tal solução, usamos o esquema Big-M para o GPAS e o esquema Fase I/Fase II proposto em [1] para o GDAS. Nestes esquemas, o algoritmo é inicialmente aplicado a um problema artificial com um critério de parada modificado. Neste estágio, se não existe solução interior viável, o algoritmo encontra uma solução que satisfaz o critério de parada para (PCRL).

Em nossos experimentos computacionais, os algoritmos GPAS e GDAS terminam quando temos um incremento relativo da função objetivo pequeno, isto é,

$$|f(x^k) - f(x^{k-1})|/\max\{1, |f(x^{k-1})|\} < \epsilon \text{ no caso do GPAS,}$$

e,

$$|b^T y^k - b^T y^{k-1}|/\max\{1, |b^T y^{k-1}|\} < \epsilon \text{ no caso do GDAS,}$$

onde ϵ é uma pequena tolerância positiva dada. Usamos $\epsilon = 10^{-8}$ para os algoritmos GPAS e GDAS.

4.3 Resultados para GPAS

Em todos os testes a variação do parâmetro r foi de $r = 1$ até $r = 2$ com um incremento de 0.05. Para os parâmetros *fator de segurança* foram atribuídos os valores $\delta = 0.99$ e $\beta = 0.001$.

Nas Tabelas 4.3 e 4.5 apresentamos o comportamento da família GPAS segundo o parâmetro r. Na coluna N1 fornecemos o número de iterações para obter uma solução viável inicial x^0, onde aplicamos o algoritmo GPAS com $r = 1.5$ ao problema artificial. As demais colunas apresentam o número de iterações requerido pelo

Tabela 4.3: Número de iterações do algoritmo GPAS para os problemas lineares.

		r								
Problema	N1	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9
adlittle	7	62	42	35	30	29	31	521	151	890
afiro	9	21	14	13	14	15	19	26	38	58
blend	10	60	42	35	51	38	113	92	133	95
kb2	15	138	110	81	80	83	100	122	77	88
sc105	8	79	65	102	129	157	155	192	233	271
sc50a	8	41	37	39	44	56	73	97	134	180
sc50b	7	34	33	36	41	50	65	87	122	169
share2b	15	71	64	81	86	125	58	151	113	164
stocfor1	14	44	41	44	60	63	77	66	78	89
total		550	448	466	535	616	691	1354	1079	2004

algoritmo GPAS para alguns valores de r para resolver o problema (PCRL) a partir de x^0. Fornecemos também, a soma dos números de iterações para cada valor de r apresentado.

Tabela 4.4: Soma dos tempos de CPU do algoritmo GPAS para os problemas lineares.

r	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9
$tcpu(s)$	67.9	56.9	68.5	81.3	100.7	95.8	190.6	147.6	280.9

Nas Tabelas 4.4 e 4.6 fornecemos as somas dos tempos de CPU, em segundos, para resolver os problemas apresentados nas Tabelas 4.3 e 4.5, respectivamente.

Tabela 4.5: Número de iterações do algoritmo GPAS para os problemas quadráticos.

		r								
Problema	N1	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9
genhs28	6	15	15	15	15	15	15	15	15	15
hs118	7	34	27	24	23	23	27	31	35	40
hs21	7	17	17	20	26	39	60	91	131	175
hs35	6	25	59	112	39	81	671	881	1171	2472
hs76	6	14	24	48	68	169	204	494	901	1484
lotschd	6	19	14	13	12	11	12	13	14	18
qpcblend	10	137	68	55	76	108	188	307	596	1121
qptest	5	8	9	15	10	16	22	38	70	128
zecevic2	6	6	5	5	17	38	77	100	74	200
total		275	238	307	286	500	1276	1970	3007	5653

Tabela 4.6: Soma dos tempos de CPU do algoritmo GPAS para os problemas quadráticos.

r	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9
$tcpu(s)$	12.6	6.2	5.0	7.1	9.6	17.2	27.8	53.4	94.9

As Tabelas 4.7 e 4.8 fornecem uma análise mais sucinta dos resultados obtidos. Nelas fornecemos o número de iterações dos métodos clássicos ($r = 1$ ou $r = 2$), o número de iterações mínima (n_{min}) e os valores de r onde se obtém o menor número de iterações (r_{min}). O valor $\overline{r_{min}}$, dado na coluna r_{min}, é a média das médias dos valores r_{min} para cada problema. A coluna 3 dá o número de iterações para o valor r mais próximo de $\overline{r_{min}}$. As colunas 7 e 8 apresentam resultados detalhados do algoritmo GPAS encontrados na iterada t para $r = 1.5$, onde t é o número de iterações geradas pelo algoritmo. O valor da função objetivo primal encontrado é dado na coluna 7, e a coluna 8 apresenta o gap de dualidade normalizado (G.D.N.),

$$|(x^t)^T Q x^t + c^T x^t - b^T y^t| / \max(1, |\frac{1}{2}(x^t)^T Q x^t + c^T x^t + c_0|).$$

Tabela 4.7: Resultados numéricos com o algoritmo GPAS para os problemas lineares.

	r						
Problema	1.00	1.35	2.00	n_{min}	r_{min}	$f(x^t)$	G.D.N.
adlittle	2221	31	97	29	1.50	2.2549493e+05	1.61e-07
afiro	217	13	89	13	1.25 - 1.35	-4.6475315e+02	5.82e-09
blend	905	34	164	34	1.35	-3.0812126e+01	7.51e-07
kb2	1463	96	109	72	1.95	-1.7238748e+03	8.30e-07
sc105	619	86	303	65	1.20	-5.2202053e+01	1.61e-07
sc50a	369	41	240	37	1.20 - 1.25	-6.4575069e+01	1.29e-07
sc50b	203	38	225	33	1.15 - 1.20	-6.9999993e+01	9.86e-08
share2b	2132	59	210	54	1.45	-4.1573190e+02	8.24e-07
stocfor1	1098	61	71	35	1.15	-4.0975037e+04	3.30e-03
total	9227	459	1508	372	$\overline{r_{min}}$=1.37		
tcpu	379.0	63.6	78.4	49.6			

Tabela 4.8: Resultados numéricos com o algoritmo GPAS para os problemas quadráticos.

	r						
Problema	1.00	1.25	2.00	n_{min}	r_{min}	$f(x^t)$	G.D.N.
genhs28	15	15	16	15	1.00 - 1.95	9.2717370e-01	3.57e-05
hs118	613	26	39	22	1.45	6.6482049e+02	6.48e-08
hs21	43	18	215	17	1.10 - 1.20	-9.9960000e+01	4.28e-09
hs35	23	121	1696	23	1.00	1.1111883e-01	3.16e-04
hs76	62	38	1960	14	1.10	-4.6818159e+00	2.05e-05
lotschd	384	13	23	11	1.50	2.3984159e+03	3.42e-09
qpcblen	1117	62	2038	55	1.30	-7.8420206e-03	2.34e-06
qptest	9	9	222	6	1.05	4.3718751e+00	4.44e-05
zecevic2	46	6	237	5	1.15 - 1.20, 1.30	-4.1249998e+00	4.68e-05
total	2312	308	6446	168	$\overline{r_{min}}$=1.25		
tcpu	34.7	5.7	78.2	4.9			

Fornecemos também, a soma dos tempos de CPU, em segundos, para os valores de r apresentados. Na coluna n_{min}, essa soma é dos tempos de CPU médios dos valores de r iguais a r_{min}.

Dos resultados com a classe GPAS apresentados, observamos que:

- Esta classe apresenta, para determinados valores de r entre 1 e 2, um desempenho muito superior aos algoritmos clássicos quando $r = 1$ ou $r = 2$ em número de iterações e um ganho considerável em tempos de CPU.
- O valor $r = 1.35$ para os problemas lineares testados e $r = 1.25$ para os problemas quadráticos, são os valores de r, entre os testados, para os quais a classe GPAS obteve o melhor desempenho.
- Com a tolerância adotada, as soluções obtidas são idênticas às soluções fornecidas pela biblioteca NETLIB para os problemas lineares testados e pelo repositório Maros e Mészáros para os problemas quadráticos.
- Parece existir uma relação convexa entre o número de iterações e o parâmetro r.

4.4 Resultados para GDAS

Em todos os testes a variação do parâmetro r foi de $r = 1.3$ até $r = 3.3$ com um incremento de 0.1. Para o parâmetro *fator de segurança* foi atribuído o valor $\gamma = 0.99$.

Na Tabela 4.9 apresentamos o comportamento da família GDAS segundo o parâmetro r. Na coluna N1 fornecemos o número de iterações para obter uma solução viável inicial y^0, onde aplicamos o algoritmo GDAS com $r = 2.0$ ao problema artificial. As demais colunas apresentam o número de iterações requerido pelo algoritmo GDAS para alguns valores de r para resolver o problema (PL) a partir de y^0. Fornecemos também, a soma dos números de iterações para cada valor de r apresentado.

Na Tabela 4.10 fornecemos as somas dos tempos de CPU, em segundos, para resolver os problemas apresentados na Tabela 4.9.

Tabela 4.9: Número de iterações do algoritmo GDAS para os problemas lineares.

		r										
Problema	N1	1.3	1.5	1.7	1.9	2.1	2.3	2.5	2.7	2.9	3.1	3.3
adlittle	1	37	27	23	24	24	24	29	33	31	33	35
afiro	1	19	18	18	19	20	22	23	24	25	27	28
bandm	8	70	34	27	26	26	28	30	32	31	33	34
blend	5	27	22	21	21	22	23	24	24	25	25	28
israel	9	164	75	44	31	65	42	44	71	65	75	82
kb2	4	35	24	21	22	22	24	25	27	31	31	32
sc105	3	28	22	21	22	23	25	26	27	28	30	31
sc205	5	29	21	21	23	25	27	30	32	34	36	37
sc50a	1	21	19	19	20	21	23	25	25	26	28	29
sc50b	2	19	18	18	20	20	22	23	24	26	28	30
scagr7	4	52	31	24	36	24	27	25	27	29	30	31
share1b	7	60	69	46	48	37	30	36	36	37	43	66
share2b	5	27	22	21	21	21	23	24	26	27	28	28
stocfor1	4	33	24	22	21	22	23	22	25	25	25	30
total		621	426	346	354	372	363	386	433	440	472	521

Tabela 4.10: Soma dos tempos de CPU do algoritmo GDAS para os problemas lineares.

r	1.3	1.5	1.7	1.9	2.1	2.3	2.5	2.7	2.9	3.1	3.3
$tcpu(s)$	814.5	362.1	268.6	253.0	294.0	271.4	291.8	344.2	335.1	371.9	403.4

A Tabela 4.11 fornece uma análise mais sucinta dos resultados obtidos. Nela fornecemos o número de iterações do método clássico ($r = 2$), o número de iterações mínima (n_{min}) e os valores de r onde se obtém o menor número de iterações (r_{min}). O valor $\overline{r_{min}}$, dado na coluna r_{min}, é a média das médias dos valores r_{min} para cada problema. A coluna 2 dá o número de iterações para o valor r mais próximo de $\overline{r_{min}}$. As colunas 6 e 7 apresentam resultados detalhados do algoritmo GDAS encontrados na iterada t para $r = 2.0$, onde t é o número de iterações geradas pelo algoritmo. O valor da função objetivo primal encontrado é dado na coluna 6, e a coluna 7 apresenta o gap de dualidade normalizado (G.D.N.),

$$|b^T y^t - c^T x^t| / \max(1, |c^T x^t|).$$

Fornecemos também, a soma dos tempos de CPU, em segundos, para os valores de r apresentados. Na coluna n_{min}, essa soma é dos tempos de CPU médios dos valores de r iguais a r_{min}.

Tabela 4.11: Resultados numéricos com o algoritmo GDAS para os problemas lineares.

	r					
Problema	1.80	2.00	n_{min}	r_{min}	$c^T x^t$	G.D.N.
adlittle	23	23	23	1.70 - 1.80, 2.00	2.2549496e+05	8.64e-10
afiro	19	20	18	1.40 - 1.70	-4.6475314e+02	1.74e-09
bandm	25	26	25	1.80	-1.5862802e+02	3.98e-09
blend	21	22	21	1.60 - 1.90	-3.0812150e+01	1.07e-09
israel	38	51	31	1.90	-8.9664482e+05	8.31e-10
kb2	22	21	21	1.70, 2.00	-1.7499001e+03	4.69e-10
sc105	22	22	21	1.60 - 1.70	-5.2202061e+01	3.34e-09
sc205	22	24	21	1.50 - 1.70	-5.2202061e+01	1.73e-09
sc50a	20	20	18	1.60	-6.4575077e+01	2.23e-09
sc50b	19	20	18	1.40 - 1.70	-7.0000000e+01	1.67e-09
scagr7	23	34	23	1.80	-2.3313898e+06	6.98e-10
share1b	35	32	30	2.30	-7.6589312e+04	8.11e-08
share2b	21	21	21	1.60 - 2.10	-4.1573224e+02	2.37e-09
stocfor1	21	21	21	1.60, 1.80 - 2.00	-4.1131976e+04	6.26e-09
total	331	357	287	$\overline{r_{min}}$=1.77		
tcpu	245.2	99.6	229.1			

Dos resultados com a classe GDAS apresentados, observamos que:

- Esta classe apresenta, para determinados valores de r diferentes de 2, um desempenho similar ao algoritmo clássico quando $r = 2$, com um ganho sensível em número de iterações, mas um desempenho inferior em tempos de CPU.
- O valor $r = 1.80$ é o valor de r, entre os testados, para os quais a classe GDAS obteve o melhor desempenho para os problemas lineares testados.
- Com a tolerância adotada, as soluções obtidas são idênticas às soluções fornecidas pela biblioteca NETLIB para os problemas lineares testados.
- Parece existir uma relação convexa entre o número de iterações e o parâmetro r.

Capítulo 5

Conceitos e Resultados Básicos em Programação Quase-Convexa

Nesta capítulo listamos conceitos e resultados básicos em programação quase-convexa importantes para nosso trabalho. Apresentamos a definição e caracterizações de funções quase-convexas e de suas subclasses, bem como os princípais resultados em otimização quase-convexa.

Convexidade desempenha um papel muito importante em teoria de otimização. Nos anos recentes houve um interesse sempre crescente nas generalizações de conceitos básicos de convexidade com a esperança de estender propriedades, caracterizações e aplicações.

A noção de quase-convexidade é um dos mais velhos e clássicos conceitos em convexidade generalizada. Funções quase-convexas podem ser definidas em meras condições geométricas postulando a convexidade de seus conjuntos de nível. Devido a sua definição simples esta classe freqüentemente é ponto de partida das pesquisas em convexidade generalizada (veja por exemplo [19, 65]).

Estas funções têm um largo domínio de aplicações em vários campos das ciências e engenharias como:

- Teoria econômica [4, 5, 64].
- Teoria de localização [37].
- Teoria do controle [9].
- Teoria de aproximação [7, 13].

5.1 Funções Quase-Convexas

Quase-convexidade é uma extensão significativa de convexidade e tem sido objeto de intensas pesquisas atualmente devido suas amplas aplicações.

A interessante classe de funções quase-convexas (ou eventualmente suas subclasses como as classes de funções estritamente ou fortemente quase-convexas) generaliza as funções convexas retendo algumas de suas propriedades importantes.

Definição 5.1.1. *([11], Definição 3.5.1, p.108) Seja $f : C \to \mathbb{R}$, onde C é um conjunto convexo não-vazio em $\mathbb{R}^n$. A função f é quase-convexa se, para cada $x, y \in C$, a seguinte inequação é verdadeira:*

$$f(\lambda x + (1-\lambda)y) \leq \max\{f(x), f(y)\}, \quad \forall \lambda \in (0,1).$$

A função f é quase-côncava se $-f$ é quase-convexa. Uma função que é ambos quase-convexa e quase-côncava é chamada quase-linear.

Da definição acima, uma função f é quase-convexa se, quando $f(y) \geq f(x)$, $f(y)$ é maior ou igual a f em todas as combinações convexas de x e y. Então, se f cresce de seu valor em um ponto ao longo de qualquer direção, ela deve permanecer não-decrescente naquela direção.

Exemplo 5.1.1. *São quase-convexas as seguintes funções crescentes:*

1. $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^3$.
2. $f : \mathbb{R} \to \mathbb{R}$, $f(x) = \arctan x$.
3. $f : \mathbb{R}_+ \to \mathbb{R}$, $f(x) = \sqrt{x}$.
4. $f : \mathbb{R}_{++} \to \mathbb{R}$, $f(x) = \log x$.

Quase-convexidade tem uma interpretação geométrica. Enquanto uma função convexa pode ser caracterizada pela convexidade de seu epígrafo, uma função quase-convexa pode ser caracterizada pela convexidade de seus conjuntos de nível.

Teorema 5.1.1. *([11], Teorema 3.5.2, p.108) Seja $f : C \to \mathbb{R}$, onde C é um conjunto convexo não-vazio em $\mathbb{R}^n$. A função f é quase-convexa se, e somente se, $\mathrm{niv}_f(\alpha) = \{x \in C \mid f(x) \leq \alpha\}$ é convexo para cada número real α.*

Demonstração. Suponha que f é quase-convexa, e sejam x, $y \in \text{niv}_f(\alpha)$. Portanto, x, $y \in C$ e $\max\{f(x), f(y)\} \leq \alpha$. Seja $\lambda \in (0,1)$, e seja $z = \lambda x + (1-\lambda)y$. Pela convexidade de C, $z \in C$. Além disso, pela quase-convexidade de f, $f(z) \leq \max\{f(x), f(y)\} \leq \alpha$. Portanto, $z \in \text{niv}_f(\alpha)$, e então $\text{niv}_f(\alpha)$ é convexo. Reciprocamente, suponha que $\text{niv}_f(\alpha)$ é convexo para cada número real α. Sejam x, $y \in C$. Além disso, seja $\lambda \in (0,1)$ e $z = \lambda x + (1-\lambda)y$. Note que x, $y \in \text{niv}_f(\alpha)$ para $\alpha = \max\{f(x), f(y)\}$. Por hipótese, $\text{niv}_f(\alpha)$ é convexo, de modo que $z \in \text{niv}_f(\alpha)$. Portanto, $f(z) \leq \alpha = \max\{f(x), f(y)\}$. Então, f é quase-convexa, e a prova está completa. □

Para uma função em $\mathbb{R}$, quase-convexidade requer que cada conjunto de nível seja um intervalo (incluindo, possivelmente, um intervalo infinito)

Funções convexas têm conjuntos de nível convexos e, então, são quase-convexas, mas o inverso não é verdade (por exemplo, a função $\log x$ em $(0, +\infty)$).

Muitas propriedades de funções convexas não valem para funções quase-convexas. Por exemplo, funções quase-convexas não precisam ser contínuas no interior de seus domínios.

Para uma função contínua em $\mathbb{R}$ temos uma caracterização simples. Uma função contínua $f : \mathbb{R} \to \mathbb{R}$ é quase-convexa se, e somente se, pelo menos uma das seguintes condições acontece:

- f é não-decrescente.
- f é não-crescente.
- Existe um ponto $c \in \text{dom} f$ tal que para $t \leq c$ (e $t \in \text{dom} f$), f é não-crescente, e para $t \geq c$ (e $t \in \text{dom} f$), f é não-decrescente.

Funções Quase-Convexas Diferenciáveis admitem caracterizações em termos de seu gradiente ou de sua matriz Hessiana.

Teorema 5.1.2. *([11], Teorema 3.5.4, p.109 - Caracterização de Primeira Ordem de Funções Quase-Convexas) Seja C um conjunto convexo aberto e não-vazio em $\mathbb{R}^n$ e seja $f : C \to \mathbb{R}$ diferenciável em C. Então, f é quase-convexa se, e somente se, para todo $x, y \in C$*

$$f(x) \leq f(y) \Longrightarrow \langle \nabla f(y), y - x \rangle \geq 0. \tag{5.1}$$

ou, equivalentemente,

$$\langle \nabla f(y), y - x \rangle < 0 \Longrightarrow f(x) > f(y). \tag{5.2}$$

A condição (5.1) tem uma interpretação geométrica simples quando $\nabla f(y) \neq 0$. Ela diz que $\nabla f(y)$ define um hiperplano suporte para o conjunto de nível $\{x \mid f(x) \leq f(y)\}$, no ponto y.

Apesar da semelhança entre a condição de primeira ordem para convexidade e a condição de primeira ordem para quase-convexidade, há algumas diferenças importantes. Por exemplo, se f é convexa e $\nabla f(x) = 0$, então x é um minimizador global de f. Mas esta afirmação é falsa para função quase-convexa: é possível que $\nabla f(x) = 0$, mas x não seja um minimizador global de f. Ilustramos este fato no exemplo seguinte.

Exemplo 5.1.2. *$f : \mathbb{R} \to \mathbb{R}$, dada por $f(x) = x^3$. Para checar a quase-convexidade de f, suponha que $f(x) \leq f(y)$, i.e., $x^3 \leq y^3$. Isto é verdade apenas se $x \leq y$. Por outro lado, $\langle f'(y), y - x \rangle = 3y^2(y - x)$. Desde que $x \leq y$, $3y^2(y - x) \geq 0$. Portanto, $f(x) \leq f(y)$ implica que $\langle f'(y), y - x \rangle \geq 0$ e, pelo Teorema 5.1.2, que f é quase-convexa. Observe ainda que $f'(0) = 0$. Entretanto, 0 não é um minimizador de f.*

Teorema 5.1.3. *([14], Seção 3.4, p.101) - Caracterização de Segunda Ordem de Funções Quase-Convexas) Seja C um conjunto convexo aberto e não-vazio em $\mathbb{R}^n$ e seja $f : C \to \mathbb{R}$ duas vezes diferenciável em C. Se f é quase-convexa, então para todo $x \in C$ e para todo $y \in \mathbb{R}^n$, temos*

$$\langle y, \nabla f(x) \rangle = 0 \Longrightarrow y^T \nabla^2 f(x) y \geq 0. \tag{5.3}$$

Reciprocamente, se f satisfaz

$$\langle y, \nabla f(x) \rangle = 0 \Longrightarrow y^T \nabla^2 f(x) y > 0. \tag{5.4}$$

para todo $x \in C$ e para todo $y \in \mathbb{R}^n$, $y \neq 0$, então f é quase-convexa.

Para uma função quase-convexa em $\mathbb{R}$, a condição (5.3) se reduz à condição simples $f'(x) = 0 \Longrightarrow f''(x) \geq 0$, isto é, em qualquer ponto com derivada nula, a segunda derivada é não-negativa. Para uma função quase-convexa em $\mathbb{R}^n$, temos a seguinte interpretação da condição (5.3). Como no caso $n = 1$, sempre que $\nabla f(x) =$

0, devemos ter $\nabla^2 f(x) \succeq 0$. Quando $\nabla f(x) \neq 0$, a condição (5.3) significa que $\nabla^2 f(x)$ é semidefinida positiva no subespaço $\nabla f(x)^{\perp}$.

A condição (5.4) é o mesmo que exigir que $\nabla^2 f(x)$ seja definida positiva para qualquer ponto com $\nabla f(x) = 0$, e para todos os outros pontos, exigir que $\nabla^2 f(x)$ seja definida positiva no subespaço $\nabla f(x)^{\perp}$.

Funções Estritamente Quase-convexas

Funções estritamente quase-convexas e estritamente quase-côncavas são especialmente importantes em programação não-linear porque elas asseguram que um mínimo local e um máximo local sobre um conjunto convexo são, respectivamente, um mínimo global e um máximo global.

Definição 5.1.2. *([11], Definição 3.5.5, p.111) Seja $f : C \to \mathbb{R}$, onde C é um conjunto convexo não-vazio em $\mathbb{R}^n$. A função f é estritamente quase-convexa se, para cada $x, y \in C$, com $f(x) \neq f(y)$, temos*

$$f(\lambda x + (1-\lambda)y) < \max\{f(x), f(y)\}, \quad \forall \lambda \in (0,1).$$

A função f é chamada estritamente quase-côncava se $-f$ é estritamente quase-convexa.

Da Definição 2.2.1, toda função estritamente convexa é de fato uma função convexa. Mas, nem toda função estritamente quase-convexa é quase-convexa, como ilustra a função dada por Karamardian [47] em 1967.

$$f(x) = \begin{cases} 1 & \text{se } x = 0 \\ 0 & \text{se } x \neq 0 \end{cases}$$

Entretanto, se f é semi-contínua inferior, quase-convexidade estrita implica em quase-convexidade.

Lema 5.1.1. *([11], Lema 3.5.7, p.112) Seja C um conjunto convexo não-vazio em $\mathbb{R}^n$ e seja $f : C \to \mathbb{R}$ estritamente quase-convexa e semi-contínua inferior. Então, f é quase-convexa.*

Funções Fortemente Quase-convexas

Funções estritamente quase-convexas asseguram que um mínimo local sobre um conjunto convexo é também um mínimo global. Entretanto, a quase-convexidade estrita não assegura a unicidade de mínimos globais. A quase-convexidade forte é outra versão de quase-convexidade que assegura a unicidade de mínimos globais.

Definição 5.1.3. *([11], Definição 3.5.8, p.112) Seja $f : C \to \mathbb{R}$, onde C é um conjunto convexo não-vazio em $\mathbb{R}^n$. A função f é fortemente quase-convexa se, para cada $x, y \in C$, com $x \neq y$, temos*

$$f(\lambda x + (1-\lambda)y) < \max\{f(x), f(y)\}, \quad \forall \lambda \in (0,1).$$

A função f é chamada fortemente quase-côncava se $-f$ é fortemente quase-convexa.

Das definições acima, temos:

- Toda função estritamente convexa é fortemente quase-convexa.
- Toda função fortemente quase-convexa é estritamente quase-convexa.
- Toda função fortemente quase-convexa é quase-convexa mesmo na ausência de qualquer hipótese de semi-continuidade.

5.2 Otimização Quase-Convexa

Apresentamos aqui os principais resultados em otimização quase-convexa. Note que um mínimo local de uma função quase-convexa sobre um conjunto convexo não é necessariamente um mínimo global. Entretanto, este resultado ocorre para uma função estritamente quase-convexa. A quase-convexidade forte é outra versão de quase-convexidade que assegura a unicidade de mínimo global, uma propriedade não desfrutada pelas função estritamente quase-convexas.

O teorema seguinte mostra que funções quase-convexas alcançam um máximo sobre um poliedro compacto em um ponto extremo.

Teorema 5.2.1. *([11], Teorema 3.5.3, p.109) Seja $\mathcal{S}$ um poliedro compacto não-vazio em $\mathbb{R}^n$ e seja $f : \mathbb{R}^n \to \mathbb{R}$ quase-convexa e contínua em $\mathcal{S}$. Considere o problema maximize $f(x)$ sujeito a $x \in \mathcal{S}$. Então, existe uma solução ótima $\overline{x}$ para o problema, onde $\overline{x}$ é um ponto extremo de $\mathcal{S}$.*

Demonstração. f é contínua no compacto $\mathcal{S}$ e, então, atinge um máximo, digamos em $x^* \in \mathcal{S}$. Se existe um ponto extremo de $\mathcal{S}$ cujo o valor objetivo é igual a $f(x^*)$, não há o que fazer. Do contrário, sejam $x_1, ..., x_k$ os pontos extremos de $\mathcal{S}$, e assuma

que $f(x^*) > f(x_j)$ para $j = 1, ..., k$. Desde que $\mathcal{S}$ é limitado, o Teorema 2.1.2 estabelece que x^* pode ser representado como

$$x^* = \sum_{j=1}^{k} \lambda_j x_j, \quad \sum_{j=1}^{k} \lambda_j = 1, \quad \lambda_j \geq 0, \ j = 1, ..., k$$

Desde que $f(x^*) > f(x_j)$ para cada j, então

$$f(x^*) > \max_{1 \leq j \leq k} f(x_j) = \alpha. \tag{5.5}$$

Consideremos o conjunto $\text{niv}_f(\alpha) = \{x \mid f(x) \leq \alpha\}$. Note que $x_j \in \text{niv}_f(\alpha)$ para $j = 1, ..., k$ e pela quase-convexidade de f, $\text{niv}_f(\alpha)$ é convexo. Então, $x^* = \sum_{j=1}^{k} \lambda_j x_j \in \text{niv}_f(\alpha)$. Isto implica que $f(x^*) \leq \alpha$, que contradiz (5.5). Esta contradição mostra que $f(x^*) = f(x_j)$ para algum ponto extremo x_j. □

Teorema 5.2.2. *([11], Teorema 3.5.6, p.111) Seja $f : \mathbb{R}^n \to \mathbb{R}$ estritamente quase-convexa. Considere o problema minimize $f(x)$ sujeito a $x \in C$, onde C é um conjunto convexo não-vazio em $\mathbb{R}^n$. Se $\overline{x}$ é uma solução ótima local, então $\overline{x}$ é também uma solução ótima global.*

Demonstração. Assuma, em contradição, que existe um $\hat{x} \in C$ com $f(\hat{x}) < f(\overline{x})$. Pela convexidade de C, $\lambda\hat{x} + (1 - \lambda)\overline{x} \in C$ para cada $\lambda \in (0, 1)$. Desde que $\overline{x}$ é um mínimo local por hipótese, então $f(\overline{x}) \leq f[\lambda\hat{x} + (1 - \lambda)\overline{x}]$, para todo $\lambda \in (0, \delta)$ e para algum $\delta \in (0, 1)$. Mas por f ser estritamente quase-convexa e $f(\hat{x}) < f(\overline{x})$, $f[\lambda\hat{x} + (1 - \lambda)\overline{x}] < f(\overline{x})$, para todo $\lambda \in (0, 1)$. Isto contradiz a otimalidade local de $\overline{x}$, e a prova está completa. □

Teorema 5.2.3. *([11], Teorema 3.5.9, p.113) Seja $f : \mathbb{R}^n \to \mathbb{R}$ fortemente quase-convexa. Considere o problema minimize $f(x)$ sujeito a $x \in C$, onde C é um conjunto convexo não-vazio em $\mathbb{R}^n$. Se $\overline{x}$ é uma solução ótima local, então $\overline{x}$ é a única solução ótima global.*

Demonstração. Desde que $\overline{x}$ é uma solução ótima local, existe uma ε-vizinhança $N_\varepsilon(\overline{x})$ em torno de $\overline{x}$ tal que $f(\overline{x}) \leq f(x)$ para todo $x \in C \cap N_\varepsilon(\overline{x})$. Suponha, por contradição, que existe um ponto $\hat{x} \in C$ tal que $\hat{x} \neq \overline{x}$ e $f(\hat{x}) \leq f(\overline{x})$. Pela quase-convexidade forte de f, segue que

$$f[\lambda\hat{x} + (1 - \lambda)\overline{x}] < \max\{f(\hat{x}), f(\overline{x})\} = f(\overline{x}),$$

para todo $\lambda \in (0,1)$. Mas para λ suficientemente pequeno, $\lambda\hat{x}+(1-\lambda)\overline{x} \in C \cap N_\varepsilon(\overline{x})$, de modo que a desigualdade acima viola a otimalidade local de $\overline{x}$. Isto completa a prova. □

Capítulo 6

Método de Ponto Proximal para Programação Quase-Convexa

Este capítulo é dedicado à apresentação e análise de um algoritmo de ponto proximal com φ-divergência para programação quase-convexa.

A motivação para esta parte de nossa tese vem do amplo campo de aplicações das funções quase-convexas em diversas áreas das ciências e engenharias e do crescente interesse de alguns grupos de pesquisa em estender para programação quase-convexa resultados obtidos em programação convexa. Entre outros, destacamos alguns trabalhos que foram importantes em nosso estudo:

- Teboulle [86] (1992): introduz o algoritmo proximal com φ-divergências para programação convexa.
- Teboulle [87] (1997): apresenta resultados de convergência do algoritmo proximal com φ-divergências para programação convexa.
- Iusem e Teboulle [46] (1995): estuda taxas de convergência do método proximal com uma classe de φ-divergências para programação convexa e linear e faz associações com métodos de lagrangeano aumentado.

Citamos também:

- Cunha et al. [23] (2006): apresenta resultados de convergência com o algoritmo proximal para programação quase-convexa em $I\!R^n_{++}$.
- Quiroz e Oliveira [75] (2006) e Souza et al. [84] (2006): estudam algoritmos proximais com distâncias de Bregman para programação quase-convexa.

6.1 Introdução

O método de ponto proximal foi introduzido originalmente por Martinet [61, 62, 63] e é baseado na aproximação de f definida por

$$f_\lambda(x) \equiv \inf_u \{f(u) + (2\lambda)^{-1}\|x-u\|^2\}, \quad \lambda > 0,$$

proposta por Moreau [71] em 1965. O algoritmo proximal foi mais adiante desenvolvido e estudado por Rockafellar [81, 82].

Métodos de ponto proximal são utilizados na solução e análise de diversos problemas de programação não-linear, mas originalmente foram elaborados no contexto do problema de encontrar raízes de operadores monótonos. Quando são aplicados ao problema dual do problema de programação convexa, produzem a ampla classe de métodos de multiplicadores, com subseqüente convergência de ambas iteradas primais e duais, estabelecendo uma forte relação com os métodos de lagrangeano aumentado [81].

O método de ponto proximal clássico para resolver o problema de programação não-linear

$$\begin{array}{lll} \text{(PNL)} & \text{minimize} & f(x) \\ & \text{sujeito a} & x \in C, \end{array}$$

onde C é um conjunto convexo fechado em $\mathbb{R}^n$ com interior C^0 não-vazio e $f : \mathbb{R}^n \to \mathbb{R}$ é uma função continuamente diferenciável, gera uma seqüência de iteradas de acordo com a fórmula

$$x^{k+1} = \arg\min_{x \in C} \{f(x) + \lambda_k \|x - x^k\|^2\}, \tag{6.1}$$

onde $\lambda_k > 0$ é um parâmetro de regularização.

Foi mostrado inicialmente em [82] que no caso em que (PNL) tem solução, f é convexa e $C = \mathbb{R}^n$, a seqüência x^k converge para uma solução do problema. No caso geral, quando C não é o espaço inteiro e f é não-convexa, o algoritmo proximal pode ser não-prático porque os sub-problemas (6.1) são tão difíceis de resolver quanto o próprio problema original.

A convergência da seqüência $\{x^k\}$ gerada tem atraído a atenção de muitos autores para o caso convexo [20, 30, 38, 43, 58, 63, 81, 82]. Porém, não há muitos trabalhos para o caso onde f não é convexa, citamos [6, 23, 74, 75, 84].

Algoritmos de ponto proximal têm sido estudados muito extensivamente. Recentemente, em diversas pesquisas, foram propostos métodos proximais generalizados

onde o usual termo quadrático em (6.1) é substituído por outros tipos de medidas que também têm um efeito de penalização, como distâncias de Bregman ou D-funções [15, 16, 28, 29, 42, 52, 75, 84, 86, 88] ou por φ-divergências [23, 44, 46, 74, 86, 87].

O método generalizado tem a forma

$$x^{k+1} = \arg\min_{x \in \mathbb{R}^n} \{f(x) + \lambda_k D_g(x, x^k)\}, \quad \lambda_k > 0, \tag{6.2}$$

onde D_g é uma função tipo-distância induzida por um função estritamente convexa g definida em C. O algoritmo (6.2) é mais prático que o (6.1) no sentido que a seqüência de sub-problemas restritos é substituída por uma seqüência de problemas irrestritos (estes últimos, presumivelmente, mais fáceis de resolver). Deve ser notado que quando f é não-convexa, (6.2) pode ainda ser difícil de resolver.

O algoritmo proximal com φ-divergências foi introduzido em 1992 por Teboulle [86]. Nele, o termo quadrático em (6.1) é substituído pela distância tipo entropia

$$d_\varphi(x, y) = \sum_{i=1}^{n} y_i \varphi\left(\frac{x_i}{y_i}\right),$$

com φ satisfazendo algumas condições que apresentaremos na Seção 2. Assim, o algoritmo proximal com φ-divergências para minimizar uma função convexa f sobre o octante não-negativo de $\mathbb{R}^n$ é dado pelo esquema

$$\begin{aligned} x^0 &> 0 \\ x^{k+1} &= \arg\min_{x \geq 0}\{f(x) + \lambda_k d_\varphi(x, x^k)\}, \quad \lambda_k > 0. \end{aligned} \tag{6.3}$$

Este algoritmo com $\varphi(t) = t \log t - t + 1$ foi proposto inicialmente por Eggermont [29]. Esta importante escolha de φ corresponde a d_φ conhecida função entropia de Kullback-Leibler [28, 29, 86] da estatística. Isto justifica a terminologia *Algoritmo Proximal Entrópico* usada para algoritmos proximais com φ-divergências.

O algoritmo (6.3) tem sido estudado extensivamente para programação convexa [44, 46, 86, 87]. Porém, existem poucos trabalhos para o caso não-convexo, destacamos o recente estudo em [23, 74]. Vale ressaltar que a principal vantagem do algoritmo (6.3) em relação ao algoritmo (6.1) é que o termo d_φ é usado para forçar as iteradas x^k permanecerem no interior do octante não-negativo de $\mathbb{R}^n$, isto é, o algoritmo (6.3) gera automaticamente uma seqüência positiva $\{x^k\}$.

6.2 φ-divergências

Nesta seção apresentamos a definição de φ-divergências e algumas de suas propriedades básicas usadas no contexto de otimização e listamos alguns exemplos. Esta classe de medidas foi introduzida por Csiszár [21] em 1967 como uma medida generalizada de informação. No contexto de métodos proximais, φ-divergências foram estudadas inicialmente por Teboulle [86] em 1992, onde várias de suas propriedades são apresentadas. Para maiores detalhes das φ-divergências, suas propriedades e relevância para problemas de otimização, veja [44, 45, 46, 86, 87].

Seja $\varphi : \mathbb{R}_{++} \to \mathbb{R}$ uma função convexa fechada que satisfaz as seguintes propriedades:

(i) φ é duas vezes continuamente diferenciável;

(ii) φ é estritamente convexa;

(iii) $\varphi(1) = \varphi'(1) = 0$ e $\varphi''(1) > 0$;

(iv) $\lim_{t \to 0^+} \varphi'(t) = -\infty$.

Denotaremos por Φ a classe das funções satisfazendo (i)-(iv).

Definição 6.2.1. *Se $\varphi \in \Phi$, então $d_\varphi : \mathbb{R}^n_{++} \times \mathbb{R}^n_{++} \to \mathbb{R}$ definida por*

$$d_\varphi(x, y) = \sum_{i=1}^{n} y_i \varphi\left(\frac{x_i}{y_i}\right), \tag{6.4}$$

é dita ser uma φ-divergência.

Usando a mesma notação de [44, 46, 87], apresentaremos alguns exemplos de funções da classe Φ:

$$\begin{aligned}
\varphi_1(t) &= t \log t - t + 1 \\
\varphi_2(t) &= t - \log t - 1 \\
\varphi_3(t) &= at - t^a + (1 - a), \quad (0 < a < 1) \\
\varphi_4(t) &= at + t^{-a} - (a + 1), \quad (a > 0)
\end{aligned}$$

Em geral, a análise de convergência de algoritmos proximais com φ-divergências requer algumas condições de regularidade adicionais sobre φ. Deste modo, é usual

considerar duas subclasses de Φ, a saber

$$\begin{aligned}\Phi_1 &\equiv \{\varphi \in \Phi \mid \varphi'(t) \leq \varphi''(1)\log t, \quad \forall t > 0\}\\ \Phi_2 &\equiv \{\varphi \in \Phi \mid \varphi''(1)\left(1-\frac{1}{t}\right) \leq \varphi'(t) \leq \varphi''(1)\log t, \quad \forall t > 0\}\end{aligned}$$

Assim, temos $\Phi_2 \subset \Phi_1 \subset \Phi$. As funções φ_1, φ_2 e φ_3 estão em Φ_2, enquanto φ_4 está em $\Phi_1 - \Phi_2$. Exemplos de funções em $\Phi - \Phi_1$ são:

$$\begin{aligned}\varphi_5(t) &= bt^a + at^{-b} - (a+b), \quad (0 < a < 1, \ b < -1)\\ \varphi_6(t) &= t^{1-a} + at\log t - t, \qquad (a > 1)\end{aligned}$$

Nossa atenção será para

$$\varphi(t) \equiv \varphi_2(t) = t - \log t - 1. \tag{6.5}$$

Neste caso,

$$d(x,y) \equiv d_{\varphi_2}(x,y) = \sum_{i=1}^{n}\left[y_i \log\frac{y_i}{x_i} + x_i - y_i\right], \tag{6.6}$$

que pode ser estendida continuamente para $\mathbb{R}^n_{++} \times \mathbb{R}^n_{+}$ se adotarmos a convenção $0\log 0 = 0$. Em outras palavras, d admite pontos com componentes nulas em seu segundo argumento.

Proposição 6.2.1. *Se $\varphi \in \Phi$, então*

(a) $\varphi(t) \geq 0$ *e* $\varphi(t) = 0$ *se, e somente se,* $t = 1$.

(b) $\varphi(t)$ *é decrescente em* $(0,1)$ *e crescente em* $(1,+\infty)$.

(c) $\lim_{t\to+\infty} \varphi(t) = +\infty$.

(d) *Se* $\varphi(t) = \varphi_2(t) = t - \log t - 1$, *então* $\lim_{t\to 0^+} \varphi(t) = +\infty$.

Demonstração. As partes (a), (b) e (c) seguem como conseqüência imediata das propriedades de φ. A parte (d) é trivial para a particular $\varphi(t) = t - \log t - 1$. □

Usando a parte (a) da Proposição 6.2.1, podemos verificar imediatamente que d_φ satisfaz

$$d_\varphi(x,y) \geq 0 \text{ e } d_\varphi(x,y) = 0 \text{ se, e somente se, } x = y, \ \forall (x,y) \in I\!R^n_{++} \times I\!R^n_{++}.$$

Assim, d_φ pode ser vista como uma função tipo distância. As propriedades de simetria e desigualdade triangular não são verificadas em geral.

Os seguintes lemas técnicos dão propriedades usuais de d que serão úteis para a análise do algoritmo que exporemos na Seção 3. O primeiro é semelhante àquele apresentando em [46, 87] para a φ-divergência dada por

$$\varphi(t) \equiv \varphi_1(t) = t \log t - t + 1,$$

a saber

$$H(x,y) \equiv d_{\varphi_1}(x,y) = \sum_{i=1}^{n} \left[x_i \log \frac{x_i}{y_i} + y_i - x_i \right]. \tag{6.7}$$

Lema 6.2.1. *Seja d dada por (6.6), então*

(a) *Os conjuntos de nível de $d(.,y)$ são limitados para cada $y \in I\!R^n_+$.*

(b) *Se $\{y^k\} \subset I\!R^n_{++}$ converge para $y \in I\!R^n_+$, então $\lim_{k \to +\infty} d(y^k, y) = 0$.*

(c) *Se $\{y^k\} \subset I\!R^n_{++}$ satisfaz $\lim_{k \to +\infty} y^k = y \in I\!R^n_+$, $\{z^k\} \subset I\!R^n_+$ é limitada e $\lim_{k \to +\infty} d(y^k, z^k) = 0$, então $\lim_{k \to +\infty} z^k = y$.*

Demonstração. Elementar a partir de (6.6). □

Lema 6.2.2. *Para cada x, $y \in I\!R^n_{++}$ e $z \in I\!R^n_+$, temos*

(a) *Se $\varphi \in \Phi_2$, então*

$$H(z,x) - H(z,y) \geq (\varphi''(1))^{-1} \sum_{i=1}^{n} (z_i - y_i) \varphi' \left(\frac{y_i}{x_i} \right), \tag{6.8}$$

onde H é dada por (6.7).

(b) *Se $\varphi(t) = t - \log t - 1$, então*

$$d(x,z) - d(y,z) \geq \sum_{i=1}^{n} (z_i - y_i) \varphi' \left(\frac{y_i}{x_i} \right), \tag{6.9}$$

onde d é dada por (6.6).

Demonstração. A parte (a) corresponde ao Lema 4.1 (ii) de [87]. A parte (b) é um caso particular da parte (a) para $\varphi(t) = t - \log t - 1$. Neste caso $\varphi''(1) = 1$ e usando (6.6) e (6.7), temos

$$H(u,v) = d_{\varphi_1}(u,v) = \sum_{i=1}^{n} \left[u_i \log \frac{u_i}{v_i} + v_i - u_i \right] = d_{\varphi}(v,u) = d(v,u),$$

para qualquer $u \in \mathbb{R}^n_+$ e $v \in \mathbb{R}^n_{++}$. Então segue a parte (b). □

6.3 Algoritmo de Ponto Proximal com φ-divergência para Programação Quase-Convexa (DPP)

Nesta seção apresentamos o algoritmo de ponto proximal com φ-divergência, introduzimos as hipóteses básicas e estabelecemos que o algoritmo está bem-definido.

Consideremos o problema de otimização

$$\begin{array}{ll} \text{(PQC)} & \text{minimize}_x \quad f(x) \\ & \text{sujeito a} \quad x \geq 0, \end{array}$$

onde $f : \mathbb{R}^n \to \mathbb{R}$ é uma função quase-convexa própria (isto é, $\text{dom} f \equiv \{x \in \mathbb{R}^n \mid f(x) < +\infty\} \neq \emptyset$).

Aplicamos o algoritmo proximal (6.3) com a φ-divergência dada por $\varphi(t) = t - \log t - 1$ para resolver o problema (PQC). Esta particular escolha de φ foi discutida em [46, 87] para minimização convexa.

Nosso principal objetivo é estabelecer que a seqüência $\{x^k\}$ gerada por nosso algoritmo está bem-definida e converge para um ponto estacionário quando o parâmetro λ_k satisfaz

$$0 < \lambda_k \leq \tilde{\lambda},$$

para algum $\tilde{\lambda} > 0$ (que inclui o caso de λ_k constante). Se, em adição, o parâmetro λ_k satisfaz a condição

$$\lim_{k \to +\infty} \lambda_k = 0,$$

obtemos então a convergência para uma solução do problema (PQC).

O algoritmo de ponto proximal com φ-divergência para resolver o problema (PQC), o qual abreviamos por algoritmo DPP, é descrito no pseudocódigo abaixo.

Algoritmo DPP:

inicialização: Dados $\mathbb{R}^n \ni x^0 > 0$ e *critério de parada*

$k = 0$

enquanto *critério de parada* não satisfeito

se $\nabla f(x^k) = 0 \rightarrow$ **pare**

$$T_{k+1} = \left\{ z \mid z = \arg\min_{x \geq 0} \{ f(x) + \lambda_k d(x, x^k) \}, \ \ \lambda_k > 0 \right\}$$

tome $x^{k+1} \in T_{k+1}$

$k = k + 1$

fim enquanto

fim Algoritmo DPP

onde $d(x, y)$ é dada por (6.6).

Relembre que

$$d(x, y) = \sum_{i=1}^{n} y_i \varphi\left(\frac{x_i}{y_i}\right) = \sum_{i=1}^{n} \left[y_i \log \frac{y_i}{x_i} + x_i - y_i \right] \tag{6.10}$$

e, então

$$(\nabla_x d(x, y))_i = \varphi'\left(\frac{x_i}{y_i}\right) = -\frac{y_i}{x_i} + 1. \tag{6.11}$$

Consideraremos duas hipóteses básicas necessárias na análise do algoritmo DPP:

H1: f é quase-convexa e continuamente diferenciável;

H2: O conjunto solução $\mathcal{S}^*(PQC)$ é não-vazio.

Conforme indicado no algoritmo, denotaremos

$$T_{k+1} = \left\{ z \mid z = \arg\min_{x \geq 0} f_k(x) \right\}, \ \text{para cada} \ \ k \geq 0, \tag{6.12}$$

onde $f_k(x) = f(x) + \lambda_k d(x, x^k)$ com $\lambda_k > 0$. Faremos ainda $T_0 = \{x^0\}$.

Estabelecemos a boa-definição do algoritmo DPP através da garantia da existência de cada iterado x^k e de sua positividade, $x^k > 0$, para cada $k \in \mathbb{N}$, isto é, o algoritmo é de pontos interiores.

Teorema 6.3.1. *Para cada* $k \geq -1$, $T_{k+1} \cap \mathbb{R}^n_{++} \neq \emptyset$. *Equivalentemente, existe* $x^{k+1} \in T_{k+1}$ *com* $x^{k+1} > 0$, *para cada* $k \geq -1$.

Demonstração. Por indução. Sabemos que $T_0 \ni x^0 > 0$. Mostraremos que $x^k > 0$ implica que existe $x^{k+1} \in T_{k+1}$ com $x^{k+1} > 0$. Seja $x^* \in \mathcal{S}^*(PQC) \neq \emptyset$, então

$$f_k(x) = f(x) + \lambda_k d(x, x^k) \geq f(x^*) + \lambda_k d(x, x^k) \geq f(x^*), \ \forall x \in \mathbb{R}^n_+,$$

pois $d(x, x^k) \geq 0$ e $\lambda_k > 0$. Assim, f_k é limitada inferiormente por $f(x^*)$. Desde que $x_i^k > 0$ e usando partes (c) e (d) da Proposição 6.2.1, temos

$$x_i^k \varphi(\frac{x_i}{x_i^k}) \to +\infty \quad \text{quando} \quad x_i \to 0^+ \quad \text{ou} \quad x_i \to +\infty$$

Então,

$$\lambda_k d(x, x^k) = \lambda_k \sum_{i=1}^{n} x_i^k \varphi(\frac{x_i}{x_i^k}) \to +\infty \quad \text{quando} \quad x \to \partial \mathbb{R}^n_{++},$$

desde que $\lambda_k > 0$ e cada parcela do somatório é não-negativa devido à positividade de x_i^k e φ. Agora, desde que $\mathcal{S}^*(PQC) \neq \emptyset$, f é limitada inferiormente e então,

$$f_k(x) = f(x) + \lambda_k d(x, x^k) \to +\infty \quad \text{quando} \quad x \to \partial \mathbb{R}^n_{++}.$$

Desde que f_k é contínua, limitada inferiormente e satisfaz $f_k(x) \to +\infty$ quando $x \to \partial \mathbb{R}^n_{++}$, segue que f_k atinge seu mínimo em um ponto $\hat{x} > 0$. Portanto, existe $x^{k+1} \in T_{k+1}$ com $x^{k+1} > 0$, para cada $k \geq -1$. □

Observação 6.3.1. *$T_{k+1} \cap \partial \mathbb{R}^n_{++} = \emptyset$, para cada $k \geq -1$.*

Uma propriedade interessante é a não ciclagem do algoritmo DPP. Esta propriedade é apresentada no Teorema 6.3.2 a seguir e estabelece que dois iterados quaisquer são distintos.

Proposição 6.3.1. *Se $z \in T_{k+1}$, então $\nabla f(z) = -\lambda_k \nabla_x d(z, x^k)$, para cada $k \geq 0$. Em particular, a seqüência $\{x^k\}$ gerada pelo algoritmo DPP satisfaz $\nabla f(x^{k+1}) = -\lambda_k \nabla_x d(x^{k+1}, x^k)$, para cada $k \geq 0$.*

Demonstração. Seja $k \geq 0$. As condições de otimalidade necessárias para minimizar $f_k(x) = f(x) + \lambda_k d(x, x^k)$ sob $x \geq 0$ são expressas como

$$x \geq 0, \quad \nabla f_k(x) \geq 0 \quad \text{e} \quad x_i(\nabla f_k(x))_i = 0, \quad i = 1, ..., n.$$

Se $z \in T_{k+1}$ então, pelo Teorema 6.3.1, z é solução ótima de f_k e $z > 0$. Logo, devemos ter $\nabla f_k(z) = 0$. Ou seja, $\nabla f(z) + \lambda_k \nabla_x d(z, x^k) = 0$. Portanto, $\nabla f(z) = -\lambda_k \nabla_x d(z, x^k)$. Em particular, como no algoritmo tomamos $x^{k+1} \in T_{k+1}$, devemos ter $\nabla f(x^{k+1}) = -\lambda_k \nabla_x d(x^{k+1}, x^k)$. □

Lema 6.3.1. *$T_{k+1} \cap T_k = \emptyset$, para cada $k \geq 0$. Equivalentemente, a seqüência $\{x^k\}$ gerada pelo algoritmo DPP satisfaz $x^{k+1} \neq x^k$, para cada $k \geq 0$, isto é, dois iterados consecutivos são distintos.*

Demonstração. Seja $k \geq 0$. Suponha que $\nabla f(x^k) \neq 0$, isto é, o algoritmo continua. Tome $x^{k+1} \in T_{k+1}$ e suponha que $x^{k+1} = x^k$. Então, pela Proposição 6.3.1, temos

$$\nabla f(x^{k+1}) = -\lambda_k \nabla_x d(x^{k+1}, x^k) = -\lambda_k \nabla_x d(x^k, x^k) = 0$$

Então, $\nabla f(x^k) = \nabla f(x^{k+1}) = 0$. Mas, isto contradiz a hipótese que $\nabla f(x^k) \neq 0$. Portanto, $x^{k+1} \neq x^k$, para cada $k \geq 0$. □

Teorema 6.3.2. *$T_{k+1} \cap T_l = \emptyset$, para cada $k \geq 0$ e para cada l, $0 \leq l \leq k$, isto é, $x^{k+1} \neq x^l$, para cada $k \geq 0$ e para cada l, $0 \leq l \leq k$. Equivalentemente, $x^{k_2} \neq x^{k_1}$ sempre que $k_2 \neq k_1$, isto é, a seqüência $\{x^k\}$ gerada pelo algoritmo DPP não cicla.*

Demonstração. O caso $l = k \geq 0$ corresponde ao Lema 6.3.1. Faremos, então, $k \geq 1$ e $l < k$. Pela construção de x^{k+1}, dada pelo algoritmo DPP, $x^{k+1} \in T_{k+1}$. Assim, usando (6.12), devemos ter

$$f(x^{k+1}) + \lambda_k d(x^{k+1}, x^k) \leq f(x^k) + \lambda_k d(x^k, x^k) = f(x^k). \tag{6.13}$$

Por outro lado, usando o Lema 6.3.1, $d(x^{k+1}, x^k) > 0$, para cada $k \geq 0$. Assim, (6.13) nos fornece

$$f(x^{k+1}) < f(x^k).$$

Portanto, a seqüência $\{f(x^k)\}$ formada é decrescente, isto é,

$$f(x^0) > f(x^1) > ... > f(x^k) > f(x^{k+1}) > ...$$

Suponha $x^{k+1} = x^l$ para algum $k \geq 1$ e $l < k$. Isto implica que $f(x^l) + \lambda_k d(x^l, x^k) \leq f(x^k)$. Desde que $x^l = x^{k+1} \neq x^k$ e $\lambda_k > 0$, temos $\lambda_k d(x^l, x^k) > 0$. Assim, $f(x^l) < f(x^k)$ e isto contradiz o fato que $\{f(x^k)\}$ é decrescente. Portanto, $x^{k+1} \neq x^l$, para cada $k \geq 0$ e para cada l, $0 \leq l \leq k$. □

6.4 Convergência do Algoritmo DPP

Nesta seção, analisamos o algoritmo DPP apresentado na seção anterior. Provamos que a seqüência $\{x^k\}$ gerada pelo algoritmo é limitada.

Em adição, sob a hipótese:

H3: O parâmetro λ_k satisfaz $0 < \lambda_k \leq \tilde{\lambda}$, para algum $\tilde{\lambda} > 0$,

mostramos que a seqüência $\{x^k\}$ converge para um ponto estacionário do problema (PQC).

Finalmente, sob a hipótese adicional:

H4: $\lim_{k \to +\infty} \lambda_k = 0$,

provamos que a convergência é para uma solução do problema (PQC).

Lema 6.4.1. *Seja $\{x^k\}$ a seqüência gerada pelo algoritmo DPP. Então,*

(a) $0 < \lambda_k d(x^{k+1}, x^k) \leq f(x^k) - f(x^{k+1})$, *para cada* $k \geq 0$.

(b) $\{f(x^k)\}$ *é uma seqüência monótona decrescente e convergente.*

(c) $\sum_{k=0}^{\infty} \lambda_k d(x^{k+1}, x^k) < \infty$ *e assim,* $\lim_{k \to +\infty} \lambda_k d(x^{k+1}, x^k) = 0$.

Demonstração. Desde que $x^{k+1} \neq x^k$ e $\lambda_k > 0$ para cada $k \geq 0$, segue que $\lambda_k d(x^{k+1}, x^k) > 0$ para cada $k \geq 0$, que é a primeira desigualdade na parte (a). Para ver a segunda desigualdade, note que $x^{k+1} \in T_{k+1}$. Então, de (6.12) temos

$$f(x^{k+1}) + \lambda_k d(x^{k+1}, x^k) \leq f(x^k) + \lambda_k d(x^k, x^k) = f(x^k).$$

Isto conclui a parte (a).

A parte (b) é uma conseqüência direta da parte (a) e do fato que $\mathcal{S}^*(PQC) \neq \emptyset$.

Provaremos agora a parte (c). Da parte (a), temos

$$\sum_{k=0}^{K} \lambda_k d(x^{k+1}, x^k) \leq \sum_{k=0}^{K} [f(x^k) - f(x^{k+1})] = f(x^0) - f(x^{K+1}) \leq f(x^0) - f(x^*) < +\infty,$$

onde $x^* \in \mathcal{S}^*(PQC)$.

Então,

$$\sum_{k=0}^{\infty} \lambda_k d(x^{k+1}, x^k) = \lim_{K \to +\infty} \sum_{k=0}^{K} \lambda_k d(x^{k+1}, x^k) < +\infty.$$

□

Teorema 6.4.1. *Se f é quase-convexa e λ_k satisfaz a hipótese H3, a seqüência $\{x^k\}$ gerada pelo algoritmo DPP é limitada.*

Demonstração. Seja $x^* \in \mathcal{S}^*(PQC) \neq \emptyset$. Então, $f(x^*) \leq f(x^{k+1})$. Tomando $x = x^k$, $y = x^{k+1}$ e $z = x^*$ na parte (b) do Lema 6.2.2, obtemos

$$\begin{aligned} d(x^k, x^*) - d(x^{k+1}, x^*) &\geq \sum_{i=1}^{n} (x_i^* - x_i^{k+1}) \varphi'\left(\frac{x_i^{k+1}}{x_i^k}\right) \\ &= \langle \nabla_x d(x^{k+1}, x^k), x^* - x^{k+1} \rangle \\ &= \langle -\lambda_k^{-1} \nabla f(x^{k+1}), x^* - x^{k+1} \rangle \\ &= \lambda_k^{-1} \langle \nabla f(x^{k+1}), x^{k+1} - x^* \rangle. \end{aligned} \tag{6.14}$$

A primeira igualdade acima vem da equação (6.11) e a segunda igualdade é a Proposição 6.3.1. Desde que $\lambda_k > 0$, f é quase-convexa e $f(x^*) \leq f(x^{k+1})$, usando o Teorema 5.1.2, temos

$$\lambda_k^{-1} \langle \nabla f(x^{k+1}), x^{k+1} - x^* \rangle \geq 0. \tag{6.15}$$

De (6.14) e (6.15), $d(x^{k+1}, x^*) \leq d(x^k, x^*)$. Assim, a seqüência $\{d(x^k, x^*)\}$ é monótona não-crescente, isto é,

$$d(x^0, x^*) \geq d(x^1, x^*) \geq ... \geq d(x^k, x^*) \geq d(x^{k+1}, x^*) \geq ...$$

Portanto,

$$\{x^k\} \subset \{x \in \mathbb{R}^n_{++} \mid d(x, x^*) \leq d(x^0, x^*)\}$$

Agora, usando a parte (a) do Lema 6.2.1, os conjuntos de nível de $d(., x^*)$ são limitados e então, $\{x^k\}$ é limitada. □

Teorema 6.4.2. *Se f é quase-convexa e λ_k satisfaz a hipótese H3, a seqüência $\{x^k\}$ gerada pelo algoritmo DPP converge para um ponto estacionário de (PQC).*

Demonstração. Seja $\bar{x} \in \mathbb{R}^n_+$ um ponto de acumulação de $\{x^k\}$, que existe pelo Teorema 6.4.1. Pela parte (b) do Lema 6.4.1, $f(x^{k+1}) < f(x^k)$ para cada $k \in \mathbb{N}$. Pela continuidade de f, segue que $f(\bar{x}) \leq f(x^k)$ para cada $k \in \mathbb{N}$. Agora, usando o mesmo argumento utilizado na prova anterior, temos que $\{d(x^k, \bar{x})\}$ é monótona não-crescente. Por outro lado, desde que $\bar{x}$ é um ponto de acumulação de $\{x^k\}$, existe uma subseqüência $\{x^{p_k}\}$ de $\{x^k\}$ tal que $\lim_{k \to +\infty} x^{p_k} = \bar{x}$. Pela parte (b) do Lema

6.2.1, temos $\lim_{k\to+\infty} d(x^{p_k}, \bar{x}) = 0$. Então, $\{d(x^k, \bar{x})\}$ é uma seqüência não-negativa, monótona não-crescente e possui uma subseqüência convergindo para 0. Portanto, a seqüência toda $\{d(x^k, \bar{x}\}$ deve convergir para 0, isto é,

$$\lim_{k\to+\infty} d(x^k, \bar{x}) = 0. \tag{6.16}$$

Agora, seja $\tilde{x}$ um outro ponto de acumulação de $\{x^k\}$. Então, existe uma subseqüência $\{x^{q_k}\}$ de $\{x^k\}$ tal que $\lim_{k\to+\infty} x^{q_k} = \tilde{x}$. De (6.16), temos também $\lim_{k\to+\infty} d(x^{q_k}, \bar{x}) = 0$. Fazendo $y^k = x^{q_k}$, $y = \tilde{x}$ e $z^k = \bar{x}$ na parte (c) do Lema 6.2.1, temos então $\bar{x} = \tilde{x}$. Assim, a seqüência $\{x^k\}$ tem um único ponto de acumulação, isto é, a seqüência $\{x^k\}$ converge.

Mostraremos, agora, que $\bar{x}$ é um ponto estacionário do problema (PQC), ou seja

$$\bar{x} \geq 0, \quad \nabla f(\bar{x}) \geq 0 \quad \text{e} \quad \bar{x}_i(\nabla f(\bar{x}))_i = 0, \quad i = 1, ..., n. \tag{6.17}$$

Sejam

$$I(\bar{x}) = \{i \in \{1, ..., n\} \mid \bar{x}_i = 0\} \quad \text{e} \quad J(\bar{x}) = \{i \in \{1, ..., n\} \mid \bar{x}_i > 0\}.$$

A primeira condição em (6.17) é óbvia, desde que $\lim_{k\to+\infty} x^k = \bar{x}$ e $x^k > 0$. Para provar as outras condições em (6.17), consideraremos dois casos.

Caso (i): Se $i \in I(\bar{x})$, provaremos que $(\nabla f(\bar{x}))_i \geq 0$.

Seja $i \in I(\bar{x})$. Suponha que $(\nabla f(\bar{x}))_i < 0$. Pela contínua diferenciabilidade de f, temos $(\nabla f(x^{k+1}))_i \to (\nabla f(\bar{x}))_i < 0$. Assim, $(\nabla f(x^{k+1}))_i < 0$ para k suficientemente grande. Pela Proposição 6.3.1 e equação (6.11), temos

$$(\nabla f(x^{k+1}))_i = -\lambda_k(\nabla_x d(x^{k+1}, x^k))_i = -\lambda_k \varphi'(\frac{x_i^{k+1}}{x_i^k}).$$

Desde que $\lambda_k > 0$, $\varphi'(\frac{x_i^{k+1}}{x_i^k}) > 0$ para k suficientemente grande. Agora, pelas propriedades de $\varphi(t)$ na Seção 2, temos $x_i^{k+1} > x_i^k$ para k suficientemente grande. Por outro lado, a seqüência toda $\{x_i^k\}$ converge para $\bar{x}_i$ e $\bar{x}_i = 0$. Isto contradiz o fato que $x_i^{k+1} > x_i^k > 0$ para k suficientemente grande. Portanto, $(\nabla f(\bar{x}))_i \geq 0$, para cada $i \in I(\bar{x})$.

Caso (ii): Se $i \in J(\bar{x})$, provaremos que $(\nabla f(\bar{x}))_i = 0$.

Seja $i \in J(\bar{x})$. Desde que $\lim_{k\to+\infty} x_i^k = \bar{x}_i > 0$, temos $\lim_{k\to+\infty} \frac{x_i^{k+1}}{x_i^k} = 1$. Novamente,

pelas propriedades de $\varphi(t)$ na Seção 2, temos então, $\varphi'(\frac{x_i^{k+1}}{x_i^k}) \to \varphi'(1) = 0$. Em adição, da Proposição 6.3.1 e equação (6.11), temos

$$(\nabla f(x^{k+1}))_i = -\lambda_k \varphi'(\frac{x_i^{k+1}}{x_i^k}).$$

Portanto, $(\nabla f(x^k))_i \to (\nabla f(\bar{x}))_i = 0$, para cada $i \in J(\bar{x})$.
Dos casos (i) e (ii), concluímos que

$$\nabla f(\bar{x}) \geq 0 \quad e \quad \bar{x}_i(\nabla f(\bar{x}))_i = 0, \quad i = 1, ..., n.$$

Portanto, $\bar{x}$ é um ponto estacionário do problema (PQC). ■

Teorema 6.4.3. *Se f é quase-convexa e λ_k satisfaz a hipótese H4, a seqüência $\{x^k\}$ gerada pelo algoritmo DPP converge para uma solução de (PQC).*

Demonstração. Da construção de x^{k+1}, dada pelo algoritmo DPP, $x^{k+1} \in T_{k+1}$. Então, de (6.12), temos

$$f(x^{k+1}) + \lambda_k d(x^{k+1}, x^k) \leq f(x) + \lambda_k d(x, x^k), \ \forall x > 0.$$

Usando que $\lim_{k \to +\infty} \lambda_k d(x^{k+1}, x^k) = 0$ pela parte (c) do Lema 6.4.1, que $d(x, x^k)$ é limitada, $\forall x > 0$, e que $\lim_{k \to +\infty} \lambda_k = 0$ pela hipótese H4 temos, tomando $k \to +\infty$,

$$f(\bar{x}) \leq f(x), \ \forall x > 0, \tag{6.18}$$

onde $\bar{x}$ é tal que $\lim_{k \to +\infty} x^k = \bar{x}$. Seja $x^* \in \mathcal{S}^*(PQC) \neq \emptyset$ e tome uma seqüência y^k em $\mathbb{R}^n_{++}$ tal que $\lim_{k \to +\infty} y^k = x^*$. De (6.18), temos

$$f(\bar{x}) \leq f(y^k), \ \forall k > 0.$$

Tomando $k \to +\infty$ novamente, temos

$$f(\bar{x}) \leq f(x^*).$$

Mas, isto é possível apenas se $\bar{x} \in \mathcal{S}^*(PQC)$. □

Capítulo 7

Experimentos Numéricos com o Algoritmo DPP

Neste capítulo, verificamos a eficiência do algoritmo de ponto proximal com φ-divergência para programação quase-convexa e $\mathbb{R}^n_{++}$ apresentado no Capítulo 4, através de experimentos numéricos realizados com problemas testes gerados aleatoriamente. Como nos experimentos realizados no Capítulo 4, todos os testes foram realizados em um PC *Pentium* III com CPU 866MHz e 256MB de memória e implementados no *MATLAB* versão 6.0 no ambiente *Windows XP.*

Nos experimentos realizados com o algoritmo DPP, utilizamos a seguinte versão aproximada do algoritmo.

Algoritmo DPP:
inicialização: Dados $\mathbb{R}^n \ni x^0 > 0$ e $\tau > 0$
$k = 0$
enquanto $|\nabla f(x^k)^T x^k| < \tau$ **faça**

1. Use um método de minimização para resolver aproximadamente o problema

$$\min_{x \in \mathbb{R}^n_+} \{f(x) + \lambda_k d(x, x^k)\},$$

   e obter um x^{k+1}.
2. Tome $k = k + 1$.

fim enquanto
fim Algoritmo DPP

Em nossos experimentos, cada sub-problema do passo 1 é resolvido utilizando uma rotina de otimização do próprio *MATLAB.*

7.1 Problemas Testados

Desde que uma função f é quase-convexa se, e somente se, todos os conjuntos de nível de f são convexos, geramos a função quase-convexa f compondo a função convexa quadrática $g : \mathbb{R}^n \to \mathbb{R}$ dada por $g(x) = \frac{1}{2}x^T Mx$ com uma função crescente $h : \mathbb{R} \to \mathbb{R}$, ou seja, $f(x) = h(g(x))$, onde $M \in \mathbb{R}^{n\times n}$ é uma matriz simétrica definida positiva dada. Este é o mesmo procedimento adotado por Chen e Pan [17].

Em nossos experimentos, a matriz M foi obtida fazendo $M = NN^T + D$, onde N é uma matriz quadrada cujos elementos não-nulos são escolhidos aleatoriamente do intervalo $[-1, 1]$ e D é uma matriz diagonal cujos elementos diagonais são escolhidos aleatoriamente de $[0.01, 1]$. Neste procedimento, o número de elementos não-nulos de N é determinado de forma que a densidade de elementos não-nulos de M possa ser estimada aproximadamente. A função h é descrita em cada experimento.

Para cada um dos experimentos realizados, geramos 10 problemas teste com n fixado em 50 e com densidade de elementos não-nulos de M aproximadamente 2%, 5%, 10%, 25% e 50%. Foram gerados 2 problemas para cada uma das densidades.

A seguir, descrevemos cada um dos experimentos realizados.

Experimento 1. Seja $h(t) = -\frac{1}{1+t}$. Obviamente, h é crescente mas não convexa em $\mathbb{R}$. Conseqüentemente, a função $f(x) = h(g(x))$ é quase-convexa em $\mathbb{R}^n$. Além disso, podemos ver que para qualquer matriz definida positiva M, o problema de minimizar $f(x)$ sujeito a $x \geq 0$ tem uma solução ótima $z^* = 0_n$ com valor ótimo $f^* = -1$.

Experimento 2. Seja $h(t) = \ln(1 + t)$. É fácil verificar que h é crescente mas não convexa para $t \geq 0$. Assim, a função $f(x) = h(g(x))$ é quase-convexa em $\mathbb{R}^n$. Além disso, o problema de minimizar $f(x)$ sujeito a $x \geq 0$ tem uma solução ótima $z^* = 0_n$ e um valor ótimo $f^* = 0$ para toda matriz definida positiva M.

Experimento 3. Seja $h(t) = \arctan(t)$. A função h é crescente mas não convexa em $\mathbb{R}$. Portanto, a função $f(x) = h(g(x))$ é quase-convexa em $\mathbb{R}^n$. Podemos verificar que para toda matriz definida positiva M o problema de minimizar $f(x)$ sujeito a $x \geq 0$ tem uma solução ótima $z^* = 0_n$ com valor ótimo $f^* = 0$.

7.2 Solução Inicial e Critério de Parada

O algoritmo DPP proposto requer que uma solução viável inicial $x^0 \in \mathbb{R}^n_{++}$ seja fornecida. Cada problema teste gerado em nossos experimentos foi resolvido começando em pontos $x^0 = 0.01w$ e $x^0 = 0.1w$, onde w é escolhido aleatoriamente do intervalo $[1, 2]$.

Em nossos experimentos computacionais, o algoritmo DPP pára na iteração x^k quando

$$|\nabla f(x^k)^T x^k| < \tau$$

onde τ é uma pequena tolerância positiva dada. Usamos $\tau = 10^{-4}$.

7.3 Resultados para DPP

Os resultados numéricos obtidos com o algoritmo DPP em nossos experimentos estão resumidos nas Tabelas 7.1 - 7.9. Nelas fornecemos a densidade aproximada de elementos não-nulos de M, o número de iterações geradas pelo algoritmo, o valor da função objetivo $f(x) = h(\frac{1}{2}x^T Mx)$ encontrado na última iteração e o tempo de CPU, em segundos, para resolver cada problema. Nas Tabelas 7.1, 7.4 e 7.7 fixamos $\lambda_k = 1$. Nas Tabelas 7.2, 7.5 e 7.8 adotamos $\lambda_k = 1/k$. Nas Tabelas 7.3, 7.6 e 7.9 consideramos $\lambda_k = 1/2^k$.

Dos resultados com o algoritmo DPP apresentados, observamos que:

- O algoritmo pode encontrar soluções ótimas para a classe de problemas de minimizar um função quase-convexa sujeita a restrições de não-negatividade, a partir de pontos iniciais dados.
- O esforço computacional a partir do ponto $x^0 = 0.1w$ foi maior que a partir do ponto $x^0 = 0.01w$.
- O esforço computacional para $\lambda_k = 1$ foi maior que para $\lambda_k = 1/k$ que por sua vez foi maior que para $\lambda_k = 1/2^k$.
- Aparentemente, problemas com densidade de elementos não-nulos mais baixa requerem um maior número de iterações.
- O algoritmo DPP apresentou resultados satisfatórios, gerando soluções ótimas globais, não apenas pontos estacionários, em um tempo razoável.

Tabela 7.1: Resultados numéricos com o algoritmo DPP para o experimento 1 com $\lambda_k = 1$.

	$x^0 = 0.01w$				$x^0 = 0.1w$			
N0.	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)
1	2%	1006	-9.999500e-01	118.52	2%	1277	-9.999501e-01	134.64
2	2%	848	-9.999501e-01	95.50	2%	1739	-9.999500e-01	187.10
3	5%	921	-9.999500e-01	107.40	5%	1323	-9.999501e-01	146.27
4	5%	767	-9.999500e-01	99.26	5%	1070	-9.999500e-01	124.28
5	10%	641	-9.999501e-01	76.76	10%	691	-9.999502e-01	79.61
6	10%	718	-9.999501e-01	81.28	10%	762	-9.999501e-01	82.27
7	25%	106	-9.999501e-01	14.11	25%	593	-9.999502e-01	69.41
8	25%	233	-9.999501e-01	28.12	25%	544	-9.999500e-01	63.35
9	50%	10	-9.999749e-01	1.24	50%	380	-9.999500e-01	45.59
10	50%	14	-9.999711e-01	1.64	50%	477	-9.999501e-01	56.32

Tabela 7.2: Resultados numéricos com o algoritmo DPP para o experimento 1 com $\lambda_k = 1/k$.

	$x^0 = 0.01w$				$x^0 = 0.1w$			
N0.	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)
1	2%	44	-9.999506e-01	4.83	2%	63	-9.999518e-01	6.60
2	2%	41	-9.999523e-01	4.51	2%	73	-9.999522e-01	7.49
3	5%	40	-9.999510e-01	4.28	5%	63	-9.999504e-01	6.56
4	5%	37	-9.999535e-01	4.51	5%	49	-9.999516e-01	5.69
5	10%	35	-9.999510e-01	3.98	10%	39	-9.999556e-01	4.32
6	10%	37	-9.999524e-01	4.01	10%	41	-9.999509e-01	4.48
7	25%	14	-9.999521e-01	1.70	25%	45	-9.999550e-01	4.21
8	25%	24	-9.999535e-01	2.94	25%	38	-9.999555e-01	4.31
9	50%	8	-9.999530e-01	1.00	50%	34	-9.999556e-01	4.12
10	50%	13	-9.999695e-01	1.58	50%	35	-9.999643e-01	4.01

Tabela 7.3: Resultados numéricos com o algoritmo DPP para o experimento 1 com $\lambda_k = 1/2^k$.

	$x^0 = 0.01w$				$x^0 = 0.1w$			
N0.	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)
1	2%	11	-9.999572e-01	1.91	2%	45	-9.999520e-01	5.16
2	2%	10	-9.999527e-01	1.89	2%	52	-9.999517e-01	5.68
3	5%	10	-9.999560e-01	1.10	5%	73	-9.999503e-01	8.22
4	5%	10	-9.999572e-01	1.22	5%	30	-9.999509e-01	3.47
5	10%	10	-9.999707e-01	1.17	10%	26	-9.999652e-01	3.03
6	10%	10	-9.999675e-01	1.20	10%	25	-9.999636e-01	2.75
7	25%	8	-9.999607e-01	1.65	25%	46	-9.999515e-01	5.07
8	25%	8	-9.999578e-01	0.95	25%	34	-9.999579e-01	4.14
9	50%	9	-9.999767e-01	1.15	50%	27	-9.999535e-01	3.20
10	50%	9	-9.999502e-01	1.66	50%	27	-9.999540e-01	3.06

Tabela 7.4: Resultados numéricos com o algoritmo DPP para o experimento 2 com $\lambda_k = 1$.

	$x^0 = 0.01w$				$x^0 = 0.1w$			
N0.	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)
1	2%	848	4.994608e-05	92.13	2%	1735	4.999399e-05	181.38
2	2%	918	5.000016e-05	109.08	2%	1275	4.997148e-05	134.68
3	5%	877	4.997207e-05	100.59	5%	1029	4.992880e-05	115.48
4	5%	998	4.999755e-05	121.22	5%	1384	4.998609e-05	151.16
5	10%	788	4.996122e-05	88.03	10%	1390	4.991965e-05	151.41
6	10%	727	4.991726e-05	86.96	10%	738	4.998312e-05	80.20
7	25%	311	4.999009e-05	34.85	25%	583	4.979878e-05	66.77
8	25%	86	4.991916e-05	10.53	25%	487	4.990536e-05	54.62
9	50%	45	4.975446e-05	5.59	50%	400	4.987983e-05	50.23
10	50%	80	4.994250e-05	9.33	50%	443	4.988838e-05	50.57

Tabela 7.5: Resultados numéricos com o algoritmo DPP para o experimento 2 com $\lambda_k = 1/k$.

	$x^0 = 0.01w$				$x^0 = 0.1w$			
N0.	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)
1	2%	41	4.762381e-05	4.40	2%	74	4.906438e-05	7.88
2	2%	41	4.725505e-05	4.46	2%	67	4.932220e-05	7.15
3	5%	41	4.929967e-05	4.49	5%	84	4.981146e-05	9.17
4	5%	46	4.691596e-05	5.66	5%	63	4.854940e-05	7.79
5	10%	39	4.895844e-05	3.77	10%	61	4.939189e-05	5.62
6	10%	38	4.866497e-05	4.27	10%	42	4.838547e-05	4.67
7	25%	25	4.865151e-05	3.14	25%	43	4.328005e-05	5.17
8	25%	16	4.979759e-05	1.91	25%	62	4.596491e-05	7.27
9	50%	17	4.404248e-05	2.05	50%	37	4.579726e-05	4.02
10	50%	15	2.092527e-05	1.76	50%	32	3.362431e-05	3.46

Tabela 7.6: Resultados numéricos com o algoritmo DPP para o experimento 2 com $\lambda_k = 1/2^k$.

	$x^0 = 0.01w$				$x^0 = 0.1w$			
N0.	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)
1	2%	10	4.719220e-05	1.65	2%	51	4.978202e-05	5.65
2	2%	11	3.539594e-05	1.05	2%	44	4.911553e-05	5.22
3	5%	10	4.692220e-05	1.06	5%	46	4.831001e-05	5.19
4	5%	12	3.776050e-05	1.46	5%	51	4.958030e-05	6.03
5	10%	12	4.181708e-05	1.07	10%	54	4.930648e-05	5.44
6	10%	10	4.655893e-05	1.08	10%	39	4.786550e-05	4.26
7	25%	11	4.428856e-05	1.16	25%	48	4.539252e-05	5.56
8	25%	10	3.460521e-05	1.17	25%	46	4.672187e-05	5.02
9	50%	11	2.296646e-05	1.24	50%	27	3.382064e-05	3.24
10	50%	10	3.250271e-05	1.19	50%	27	4.750718e-05	2.94

Tabela 7.7: Resultados numéricos com o algoritmo DPP para o experimento 3 com $\lambda_k = 1$.

	$x^0 = 0.01w$				$x^0 = 0.1w$			
N0.	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)
1	2%	962	4.997779e-05	107.84	2%	1060	4.996537e-05	113.78
2	2%	809	4.994880e-05	103.26	2%	1511	4.998563e-05	165.02
3	5%	676	4.998906e-05	76.03	5%	1023	4.996251e-05	109.33
4	5%	732	4.999674e-05	77.37	5%	1148	4.990883e-05	124.22
5	10%	438	4.996348e-05	51.40	10%	606	4.994976e-05	67.18
6	10%	701	4.993867e-05	79.49	10%	966	4.994579e-05	104.32
7	25%	216	4.982114e-05	27.64	25%	588	4.990719e-05	66.80
8	25%	361	4.998463e-05	39.54	25%	551	4.992753e-05	61.64
9	50%	97	4.996408e-05	12.44	50%	406	4.997253e-05	46.99
10	50%	108	4.282839e-05	15.32	50%	450	4.975316e-05	55.27

Tabela 7.8: Resultados numéricos com o algoritmo DPP para o experimento 3 com $\lambda_k = 1/k$.

	$x^0 = 0.01w$				$x^0 = 0.1w$			
N0.	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)
1	2%	43	4.791093e-05	5.30	2%	55	4.944467e-05	5.65
2	2%	38	4.851443e-05	4.52	2%	72	4.943373e-05	8.06
3	5%	41	4.921878e-05	4.28	5%	51	4.880250e-05	5.24
4	5%	39	4.979324e-05	4.14	5%	64	4.725765e-05	6.86
5	10%	24	4.962289e-05	2.69	10%	43	4.556804e-05	4.62
6	10%	37	4.897285e-05	4.00	10%	48	4.787502e-05	6.27
7	25%	23	4.940783e-05	2.90	25%	38	3.876477e-05	4.40
8	25%	10	4.382170e-05	0.98	25%	35	4.915849e-05	3.22
9	50%	13	4.506083e-05	1.55	50%	37	4.826173e-05	4.53
10	50%	11	4.112590e-05	1.37	50%	36	3.612649e-05	4.24

Tabela 7.9: Resultados numéricos com o algoritmo DPP para o experimento 3 com $\lambda_k = 1/2^k$.

	$x^0 = 0.01w$				$x^0 = 0.1w$			
N0.	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)	*den*	n_{iter}	$f(x^t)$	*tcpu*(*s*)
1	2%	11	3.919595e-05	1.10	2%	41	4.205108e-05	4.34
2	2%	10	3.850797e-05	1.13	2%	46	4.855191e-05	5.82
3	5%	10	4.672469e-05	1.01	5%	46	4.942892e-05	4.91
4	5%	10	3.555637e-05	1.03	5%	48	4.904297e-05	5.42
5	10%	9	3.241375e-05	0.88	10%	39	4.564869e-05	3.82
6	10%	10	4.723042e-05	0.94	10%	36	4.809467e-05	3.34
7	25%	12	3.873060e-05	1.57	25%	38	4.386017e-05	4.34
8	25%	9	4.101212e-05	0.99	25%	27	3.772819e-05	2.92
9	50%	10	4.073453e-05	1.14	50%	31	4.560775e-05	3.55
10	50%	10	4.204327e-05	1.93	50%	26	4.444768e-05	2.99

Capítulo 8

Considerações Finais

Finalizamos nosso trabalho fazendo algumas conclusões e apresentando sugestões de possíveis trabalhos futuros de continuidade.

Conclusões

- Apresentamos a classe de algoritmos afim escala primal de Pinto et al. [73] para programação convexa com restrições lineares
- Propomos uma classe de algoritmos afim escala dual para programação linear que generaliza o algoritmo afim escala dual de Adler et al. [1].
- Provamos a convergência global da classe afim escala dual para problemas de programação linear cujo dual seja não-degenerado.

Em um outro enfoque:

- Analisamos o método de ponto proximal com a φ-divergência dada por $\varphi(t) = t - \log t - 1$ para minimizar funções quase-convexas sujeitas a restrições de não-negatividade.
- Obtivemos que a seqüência $\{x^k\}$ gerada pelo algoritmo proximal converge para um ponto estacionário quando o parâmetro λ_k satisfaz $0 < \lambda_k \leq \tilde{\lambda}$, para algum $\tilde{\lambda} > 0$.
- Provamos que se o parâmetro λ_k satisfaz a condição de regularidade $\lim_{k\to+\infty} \lambda_k = 0$, a convergência é para uma solução do problema.

Observamos a performance dos algoritmos propostos por meio de diversos experimentos numéricos realizados com problemas testes da biblioteca NETLIB e do

repositório Maros e Mészáros no caso dos algoritmos afim escala e gerados aleatoriamente no caso do algoritmo proximal.

Dos experimentos numéricos com os algoritmos afim escala, verificamos que:

- A família de algoritmos afim escala primal proposta por Pinto et al. [73] apresenta, para valores intermediários de r entre 1 e 2, um desempenho superior aos algoritmos clássicos quando $r = 1$ ou $r = 2$.
- A família de algoritmos afim escala dual proposta apresenta, para determinados valores de r diferentes de 2, um desempenho similar ao algoritmo clássico quando $r = 2$.
- Com a tolerância adotada, as soluções obtidas nos experimentos computacionais com as classes de algoritmos afim escala são idênticas às soluções fornecidas pelas bibliotecas.
- Parece existir uma relação convexa entre o número de iterações e o parâmetro r para ambas classes afim escala.

Dos experimentos numéricos com o algoritmo proximal, observamos que:

- O esforço computacional a partir do ponto $x^0 = 0.1w$ foi maior que a partir do ponto $x^0 = 0.01w$.
- O esforço computacional para $\lambda_k = 1$ foi maior que para $\lambda_k = 1/k$ que por sua vez foi maior que para $\lambda_k = 1/2^k$.
- Aparentemente, problemas com densidade de elementos não-nulos mais baixa requerem um maior número de iterações.
- O algoritmo DPP apresentou resultados satisfatórios, gerando soluções ótimas globais, não apenas pontos estacionários, em um tempo razoável.

Sugestões para Trabalhos Futuros

Com relação aos algoritmos afim escala, os fatos observados nos motivam a realizar pesquisas na tentativa de:

- Estudar a dependência das famílias de algoritmos GPAS e GDAS em relação ao parâmetro r.

- Estender os resultados para o caso geral de problemas sem a hipótese de não-degenerescência.
- Analisar a família de algoritmos GDAS para o caso em que a função objetivo é convexa não-linear, como o caso de programação quadrática.
- Experimentar uma metodologia afim escala primal-dual generalizada.
- Realizar testes com outros problemas das bibliotecas mencionadas ou de outras bibliotecas de referência.

Somos motivados também a realizar pesquisas com algoritmos de ponto proximal para programação quase-convexa com o intuito de:

- Estender os resultados para uma classe de φ-divergências, da qual a φ-divergência dada por $\varphi(t) = t - \log t - 1$ seja um representante.
- Obter resultados equivalentes com distâncias de Bregman ou outros tipos de distâncias.
- Estabelecer algoritmos para problemas com restrições lineares.
- Realizar mais experimentos numéricos com problemas gerados aleatoriamente ou fornecidos por alguma biblioteca de referência.

Apêndice A

Gráficos

Os gráficos seguintes ilustram o comportamento do número de iterações em relação ao parâmetro r para os experimentos realizados com os algoritmos GPAS e GDAS.

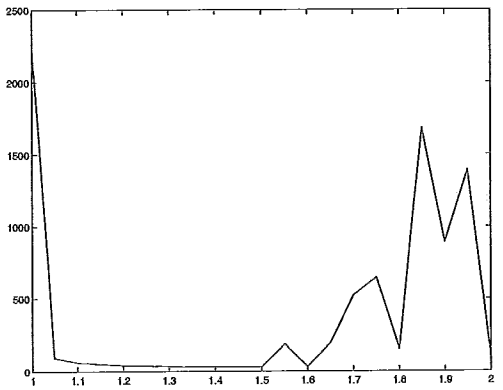

Figura A.1: $n_{iter} \times r$ do algoritmo GPAS para *adlittle*.

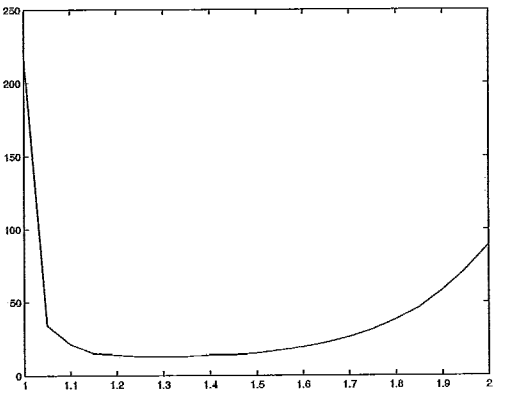

Figura A.2: $n_{iter} \times r$ do algoritmo GPAS para *afiro*.

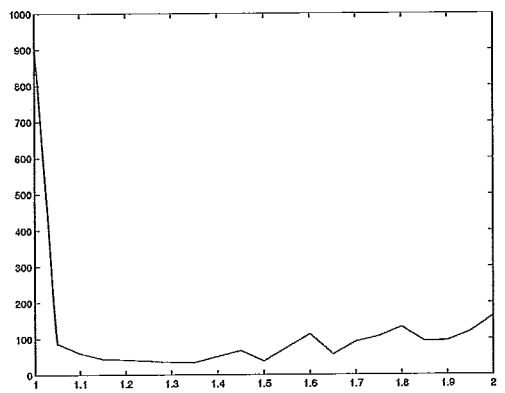

Figura A.3: $n_{iter} \times r$ do algoritmo GPAS para *blend*.

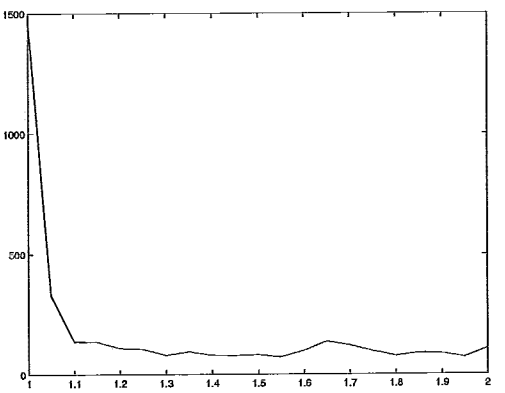

Figura A.4: $n_{iter} \times r$ do algoritmo GPAS para *kb2*.

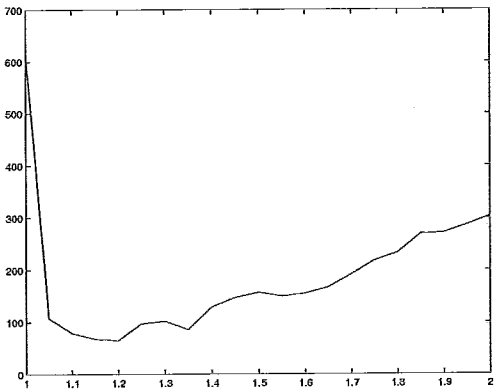

Figura A.5: $n_{iter} \times r$ do algoritmo GPAS para *sc105*.

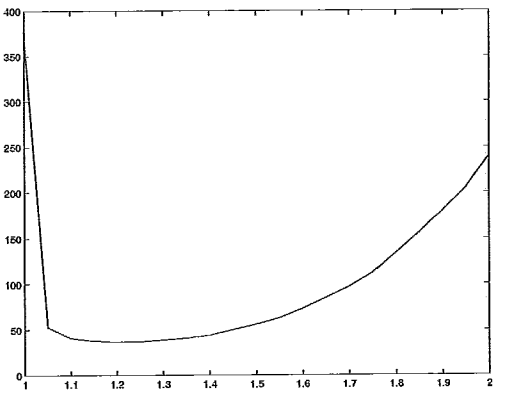

Figura A.6: $n_{iter} \times r$ do algoritmo GPAS para *sc50a*.

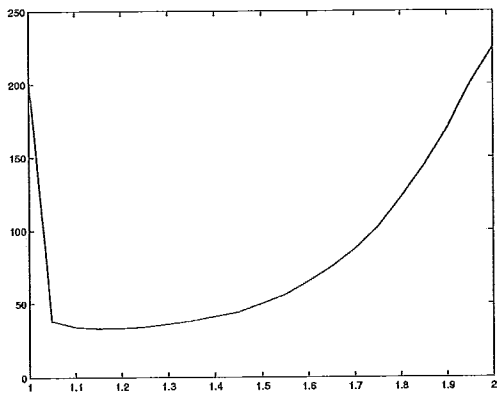

Figura A.7: $n_{iter} \times r$ do algoritmo GPAS para *sc50b*.

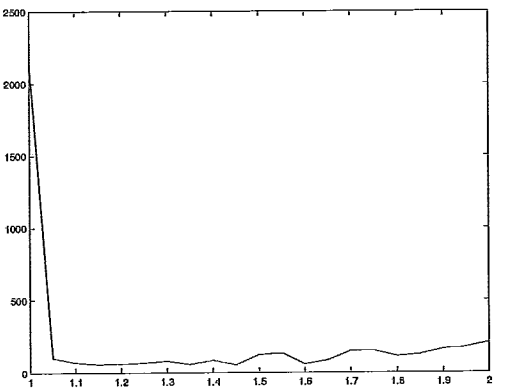

Figura A.8: $n_{iter} \times r$ do algoritmo GPAS para *share2b*.

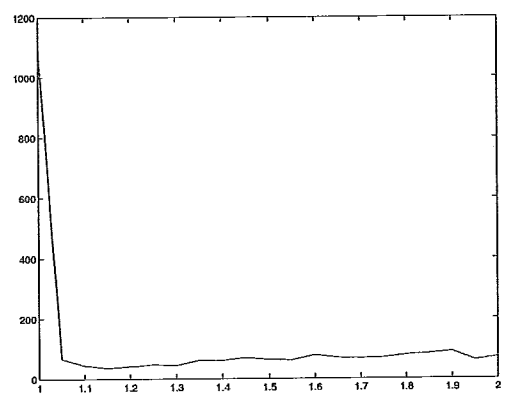

Figura A.9: $n_{iter} \times r$ do algoritmo GPAS para *stocfor1*.

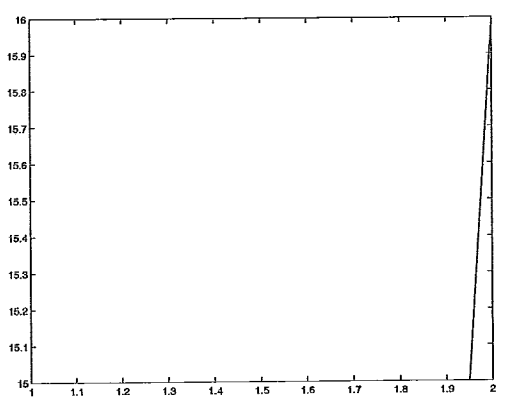

Figura A.10: $n_{iter} \times r$ do algoritmo GPAS para *genhs28*.

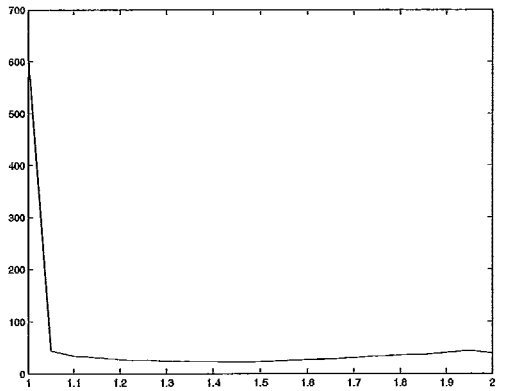

Figura A.11: $n_{iter} \times r$ do algoritmo GPAS para *hs118*.

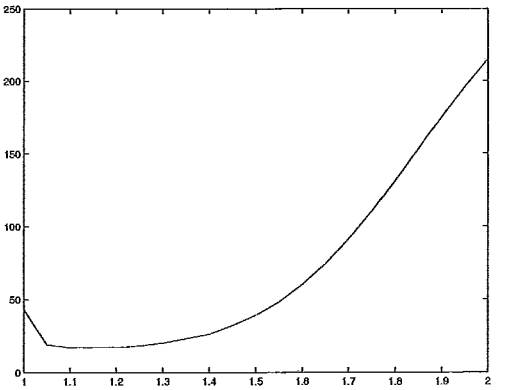

Figura A.12: $n_{iter} \times r$ do algoritmo GPAS para *hs21*.

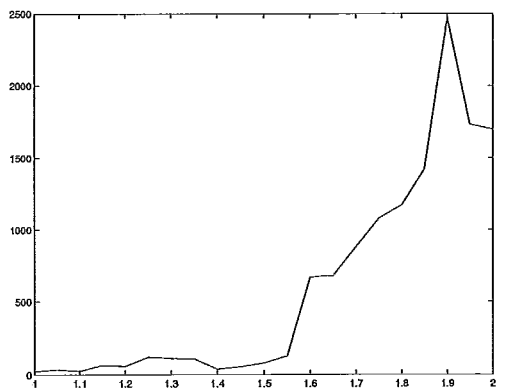

Figura A.13: $n_{iter} \times r$ do algoritmo GPAS para *hs35*.

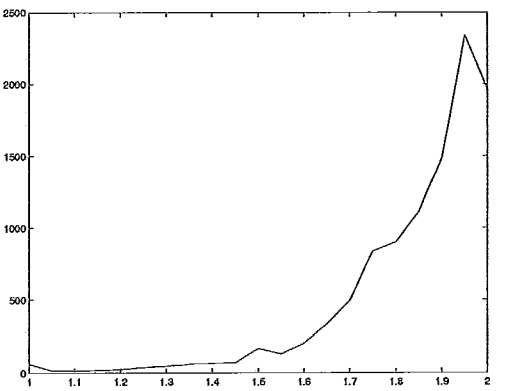

Figura A.14: $n_{iter} \times r$ do algoritmo GPAS para *hs76*.

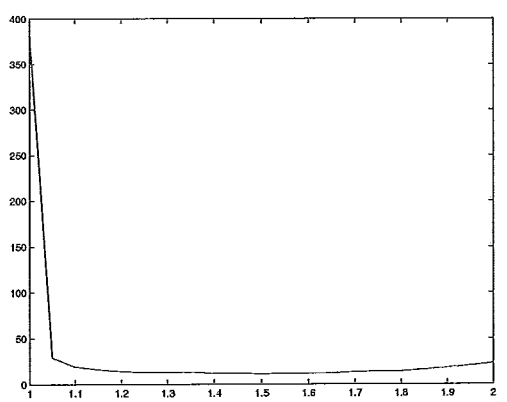

Figura A.15: $n_{iter} \times r$ do algoritmo GPAS para *lotschd*.

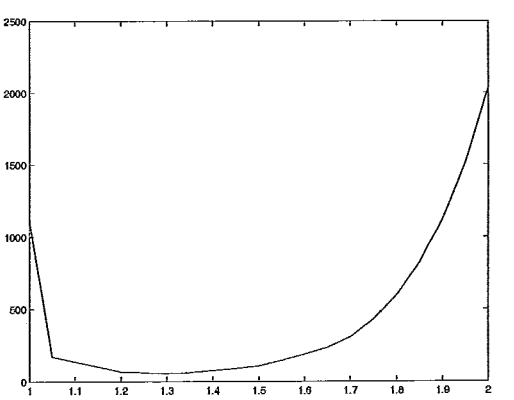

Figura A.16: $n_{iter} \times r$ do algoritmo GPAS para *qpcblend*.

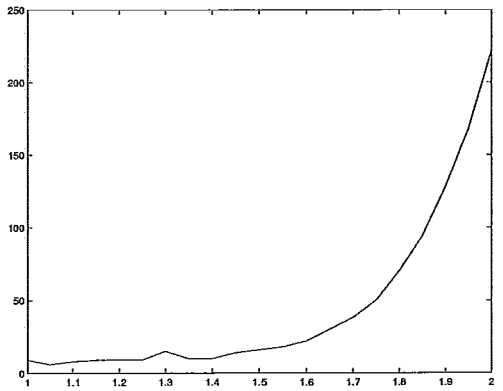

Figura A.17: $n_{iter} \times r$ do algoritmo GPAS para *qptest*.

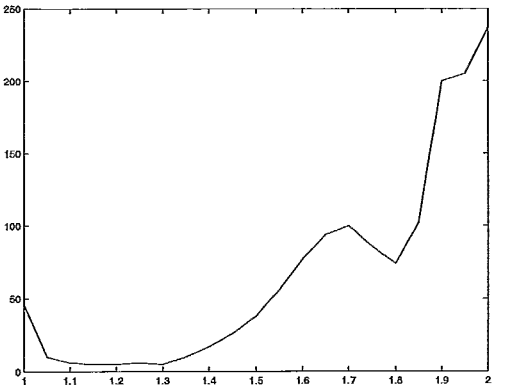

Figura A.18: $n_{iter} \times r$ do algoritmo GPAS para *zecevic2*.

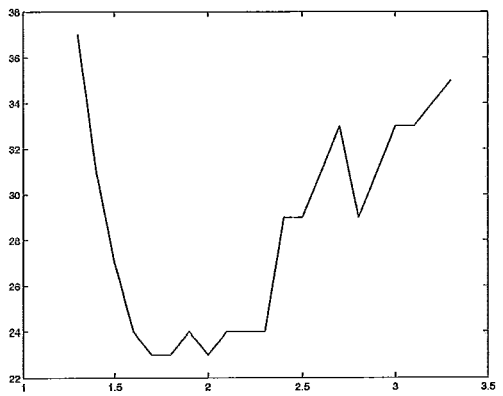

Figura A.19: $n_{iter} \times r$ do algoritmo GDAS para *adlittle*.

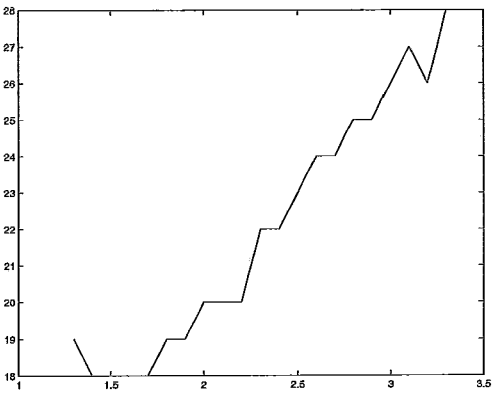

Figura A.20: $n_{iter} \times r$ do algoritmo GDAS para *afiro*.

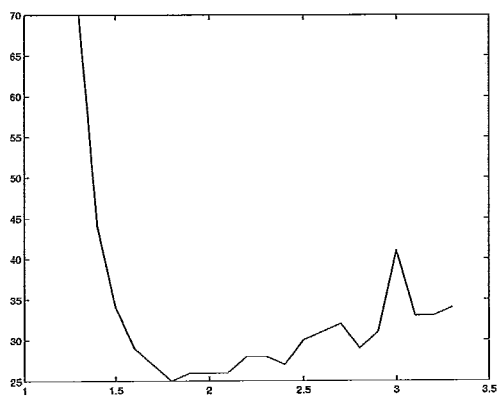

Figura A.21: $n_{iter} \times r$ do algoritmo GDAS para *bandm*.

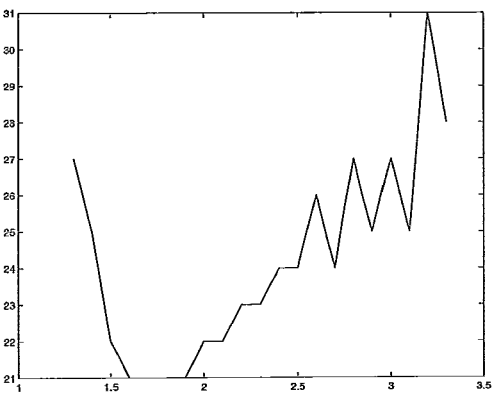

Figura A.22: $n_{iter} \times r$ do algoritmo GDAS para *blend*.

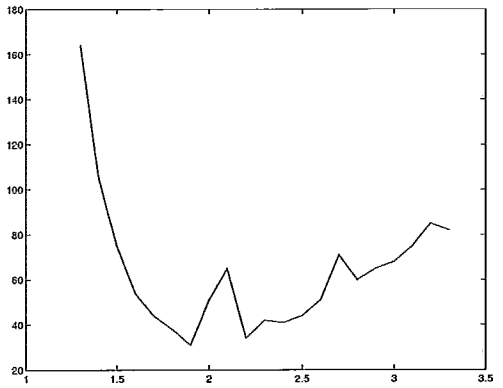

Figura A.23: $n_{iter} \times r$ do algoritmo GDAS para *israel*.

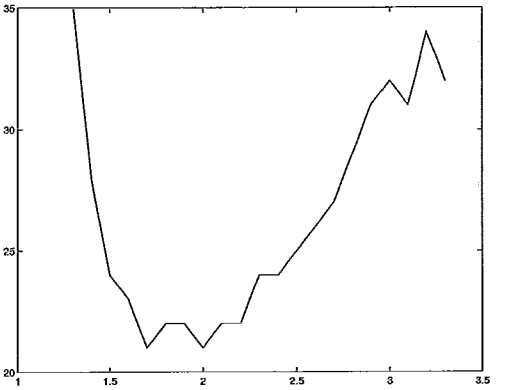

Figura A.24: $n_{iter} \times r$ do algoritmo GDAS para *kb2*.

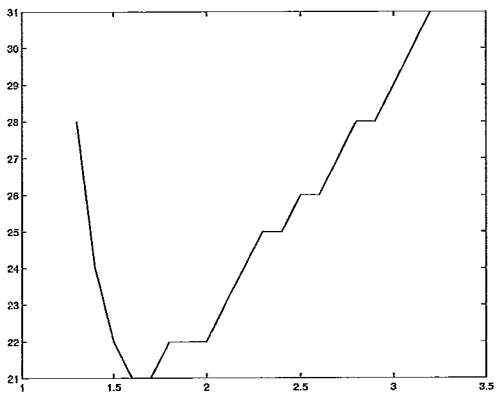

Figura A.25: $n_{iter} \times r$ do algoritmo GDAS para *sc105*.

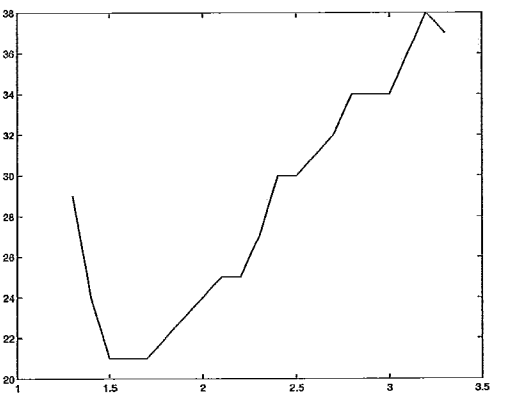

Figura A.26: $n_{iter} \times r$ do algoritmo GDAS para *sc205*.

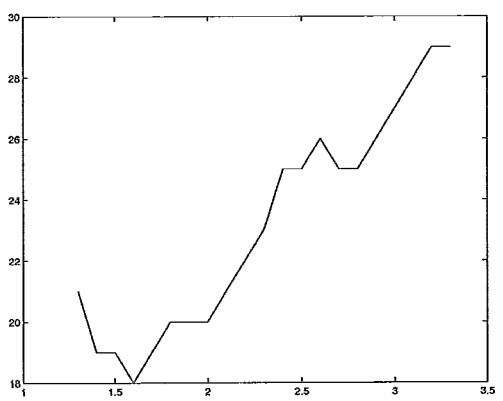

Figura A.27: $n_{iter} \times r$ do algoritmo GDAS para *sc50a*.

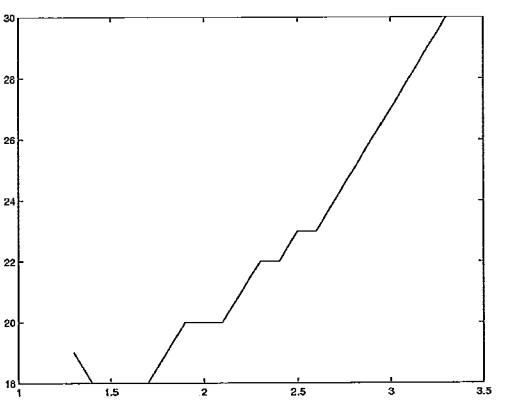

Figura A.28: $n_{iter} \times r$ do algoritmo GDAS para *sc50b*.

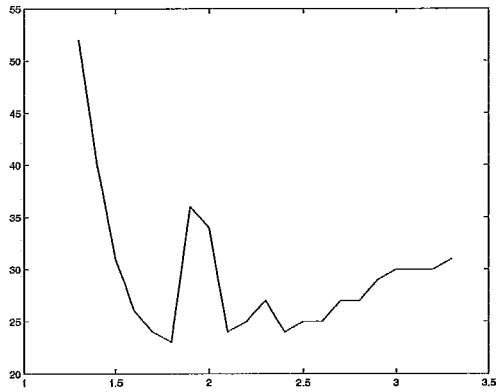

Figura A.29: $n_{iter} \times r$ do algoritmo GDAS para *scagr7*.

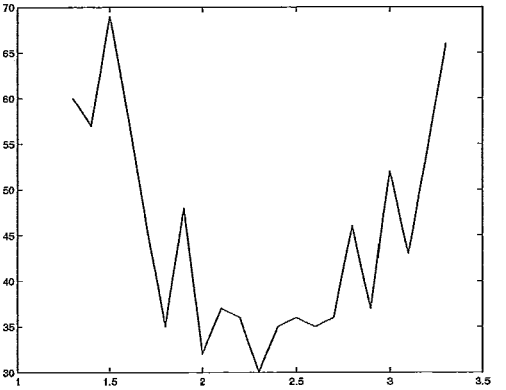

Figura A.30: $n_{iter} \times r$ do algoritmo GDAS para *share1b*.

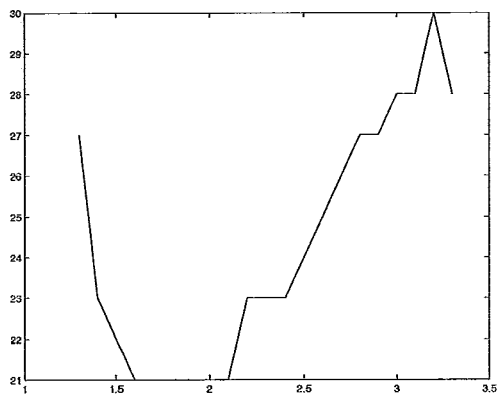

Figura A.31: $n_{iter} \times r$ do algoritmo GDAS para *share2b*.

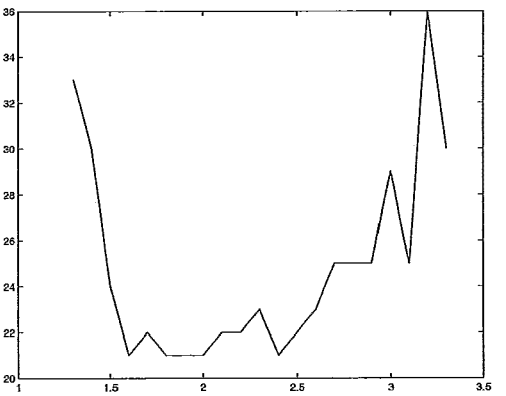

Figura A.32: $n_{iter} \times r$ do algoritmo GDAS para *stocfor1*.

Referências Bibliográficas

[1] Adler, I., Resende, M. G. C., Veiga, G., et al., "An Implementation of Karmarkar's Algorithm for Linear Programming", *Mathematical Programming*, v. 44, pp. 297-335, 1989. Errata in *Mathematical Programming*, v. 50, pp. 415, 1991.

[2] Adler, I., Karmarkar, N. K., Resende, M. G. C., et al., "Data Structures and Programming Techniques for the Implementation of Karmarkar's Algorithm", *ORSA Journal on Computing*, v. 1, n. 2, pp. 84-106, 1989.

[3] Adler, I. and Monteiro, R. D. C., "Limiting Behavior of the Affine Scaling Continuous Trajectories for Linear Programming Problems", *Mathematical Programming*, v. 50, pp. 29-51, 1991.

[4] Arrow, K. J. and Enthoven, A. C., "Quasi-Concave Programming", *Econometrica*, v. 29, n. 4, pp. 779-800, 1961.

[5] Ashton, D. J. and Atkins, D. R., "Multicriteria Programming for Financial Planning", *Journal of Operational Research Society*, v. 30, n. 3, pp. 259-270, 1979.

[6] Attouch, H. and Teboulle, M., "Regularized Lotka-Volterra Dynamical System as Continuous Proximal-like Method in Optimization", *Journal Optimization Theory and Applications*, v. 121, n. 3, pp. 541-580, 2004.

[7] Bajona-Xandri, C. and Martinez-Legaz, J. E., "Lower Subdifferentiability in Minimax Fractional Programming", *Optimization*, v. 45, pp. 1-12, 1999.

[8] Barnes, E. R., "A Variation on Karmarkar's Algorithm for Solving Linear Programming Problems", *Mathematical Programming*, v. 36, n. 2, pp. 174-182, 1986.

[9] Barron, E. N. and Liu, W., "Calculus of Variation in L^{∞}", *Applied Mathematics and Optimization*, v. 35, pp. 237-263, 1997.

[10] Bazaraa, M. S., Jarvis, J. J. and Sherali, H. D., *Linear Programming and Network Flows*. 2 ed., New York, Jonh Wiley & Sons, 1990.

[11] Bazaraa, M. S., Sherali, H. D. and Shetty, C. M., *Nonlinear Programming: Theory and Algorithms*. 2 ed., New York, John Wiley & Sons, 1993.

[12] Bellman, R. E., *Dynamic Programming*. New Jersey, Princeton University Press, 1957.

[13] Boncompte, M. and Martinez-Legaz, J. E., "Fractional Programming by Lower Subdifferentiability Techniques", *Journal of Optimization Theory and Applications*, v. 68, n. 1, pp. 95-116, 1991.

[14] Boyd, S. P. and Vandenberghe, L., *Convex Optimization*. Cambridge, Cambridge University Press, 2004.

[15] Censor, Y. and Zenios, A., "The Proximal Minimization Algorithms with D-functions", *Journal of Optimization Theory and Applications*, v. 73, n. 3, pp. 451-464, 1992.

[16] Chen, G. and Teboulle, M., "Convergence Analysis of a Proximal-like Minimization Algorithm Using Bregman Functions", *SIAM Journal on Optimization*, v. 3, n. 3, pp. 538-543, 1993.

[17] Chen, J.-S. and Pan, S., "A Proximal-like Algorithm for Quasiconvex Programming", Preprint, 2006.

[18] Chvátal, V., *Linear Programming*. New York, W. H. Freeman and Company, 1983.

[19] Crouzeix, J.-P. and Ferland, J. A., "Criteria for Quasi-Convexity and Pseudo-Convexity: Relationships and Comparisons", *Mathematical Programming*, v. 23, n. 1, pp. 193-205, 1982.

[20] Cruz Neto, J. X., Ferreira, O. P., Iusem, A. N., et al., "Dual Convergence of the Proximal Point Method with Bregman Distances for Linear Programming", *Optimization Methods and Software*, v. Online, pp. 1-23, 2006.

[21] Csiszár, I., "Information-type Measures of Difference of Probability Distributions and Indirect Observations", *Studia Scientiarum Mathematicarum Hungarica*, v. 2, pp. 299-318, 1967.

[22] Cunha, F. G. M., Pinto, A. M., Oliveira, P. R., et al., "Generalization of the Primal and Dual Affine Scaling Algorithms", Submmited, 2005.

[23] Cunha, F. G. M., Cruz Neto, J. X. and Oliveira, P. R., "A Proximal Point Algorithm with φ-divergence to Quasiconvex Programmimg", Submmited, 2006.

[24] Dantzig, G. B., "Programming in a Linear Structure", *Econometrica*, v. 17, n. 1, pp. 73-74, 1949.

[25] Den Hertog, D., *Interior Point Approach to Linear, Quadratic and Convex Programming - Algorithms and Complexity.* Ph.D. thesis, Faculty of Mathematics and Informatics, Delft, The Netherlands, 1992.

[26] Dikin, I. I., "Iterative Solution of Problems of Linear and Quadratic Programming", *Soviet Mathematics Doklady*, v. 8, pp. 674-675, 1967.

[27] Dikin, I. I., "On the Convergence of an Iterative Process", *Upravlyaemye Sistemi*, v. 12, pp. 54-60, 1974. (In Russian).

[28] Eckstein, J., "Nonlinear Proximal Point Algorithms Using Bregman Functions, with Applications to Convex Programming", *Mathematics of Operations Research*, v. 18, n. 1, pp. 202-226, 1993.

[29] Eggermont, P. P. B., "Multiplicative Iterative Algorithms for Convex Programming", *Linear Algebra and its Applications*, v. 130, pp. 25-42, 1990.

[30] Ferreira, O. P. and Oliveira, P. R., "Proximal Point Algorithm on Riemannian Manifolds", *Optimization*, v. 51, n. 2, pp. 257-270, 2002.

[31] Fiacco, A. V. and McCormick, G. P., *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* New York, John Wiley & Sons, 1968.

[32] Frisch, K. R., *The Logarithmic Potential Method of Convex Programming.* Technical Report, University Institute of Economics, Oslo, Norway, 1955.

[33] Gill, P. E., Murray, W., Saunders, M. A., et al., "On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method", *Mathematical Programming*, v. 36, n.2, pp. 183-209, 1986.

[34] Gill, P. E., Murray, W. and Wright, M. H., *Practical Optimization.* New York, Academic Press, 1981.

[35] Gomory, R. E., *An Algorithm for Integer Solutions to Linear Programs.* Technical Report No. 1, IBM Mathematics Research Project, Princeton, NJ, USA, 1958.

[36] Gonzaga, C. C. and Carlos, L. A., *A Primal Affine-Scaling Algorithms for Linearly Constrained Convex Programs.* Technical Report ES - 238/90, Department of Systems Engineering and Computer Science, COPPE, Federal University of Rio de Janeiro, Rio de Janeiro, RJ, Brasil, 1990.

[37] Gromicho, J. A. S., *Quasiconvex Optimization and Location Theory.* Dordrecht, The Netherlands, Kluwer Academic Publishers, 1998.

[38] Güler, O., "On the Convergence of the Proximal Point Algorithm for Convex Minimization", *SIAM Journal on Control and Optimization*, v. 29, n. 2, pp. 403-419, 1991.

[39] Güller, O., Den Hertog, D., Roos, C., et al., *Degeneracy in Interior Point Methods for Linear Programming.* Technical Report, Faculty of Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands, 1991.

[40] Hiriart-Urruty, J.-B. and Lemaréchal, C., *Convex Analysis and Minimization Algorithms.* Berlin, Springer-Verlag, 1993.

[41] Iusem, A. N., "An Interior Point Multiplicative Methods for Optimization Under Positivity Constraints", *Acta Mathematicae*, v. 38, pp. 163-184, 1995.

[42] Iusem, A. N., "On Some Properties of Generalized Proximal Point Methods for Quadratic and Linear Programming", *Journal of Optimization Theory and Applications*, v. 85, pp. 593-612, 1995.

[43] Iusem, A. N., Svaiter, B. F. and Cruz Neto, J. X., "Central Paths, Generalized Proximal Point Methods and Cauchy Trajectories in Riemannian Manifolds", *SIAM Journal on Control and Optimization*, v. 37, n. 2, pp. 566-588, 1999.

[44] Iusem, A. N., Svaiter, B. F. and Teboulle, M., "Entropy Like Proximal Methods in Convex Programming", *Mathematics of Operations Research*, v. 19, pp. 790-814, 1994.

[45] Iusem, A. N. and Teboulle, M., "On the Convergence Rate of Entropic Proximal Optimization Algorithms", *Computational and Applied Mathematics*, v. 12, pp. 153-168, 1993.

[46] Iusem, A. N. and Teboulle, M., "Convergence Rate Analysis of Nonquadratic Proximal Methods for Convex and Linear Programming", *Mathematics of Operations Research*, v. 20, pp. 657-677, 1995.

[47] Karamardian, S., "Strictly Quasi-Convex (Concave) Functions and Duality in Mathematical Programming", *J. Mathematical Analysis and Applications*, v. 20, pp. 344-358, 1967.

[48] Karmarkar, N. K., "New Polynomial-Time Algorithm for Linear Programming", *Combinatorica*, v. 4, pp. 373-395, 1984.

[49] Karmarkar, N. K. and Ramakrishnan, K. G., *Futher Developments in the New Polynomial-Time Algorithm for Linear Programming.* Talk given at ORSA/TIMS National Meeting, Boston, April 1985.

[50] Karmarkar, N. K. and Ramakrishnan, K. G., "Computational Results of an Interior Point Alghorithm for Large Scale Linear Programming", *Mathematical Programming*, v. 52, pp. 555-586, 1991.

[51] Khachian, L. G., "A Polinomial Algorithm in Linear Programming", *Soviet Mathematics Doklady*, v. 20, pp. 191-194, 1979.

[52] Kiwiel, K. C., "Proximal Minimization Methods with Generalized Bregman Functions", *SIAM Journal on Control and Optimization*, v. 35, pp. 1142-1168, 1997.

[53] Klee, V. and Minty G. J., "How good is the simplex algorithm?". In: O. Shisha (ed.), *Inequalities-III*, New York, Academic Press, pp. 159-175, 1972.

[54] Kojima, M., Mizuno, S. and Yoshise, A., "A Polynomial-Time Algorithm for a Class of Linear Complementarity Problems", *Mathematical Programming*, v. 44, pp. 1-26, 1989.

[55] Kojima, M., Mizuno, S. and Yoshise, A., "A Primal-Dual Interior Point Method for Linear Programming", In: N. Megiddo (ed.), *Progress in Mathematical Programming: Interior-Point and Related Methods*, Springer-Verlag, New York, pp. 29-47, 1989.

[56] Kuhn, H. W. and Tucker, A. W., "Nonlinear Programming", *Econometrica*, v. 19, pp. 50-51, 1951.

[57] Hoffman, A. J., "On Aproximate Solucions of Systems of Linear Inequalities", *Journal of Research of the National Bureau of Standards*, v. 49, pp. 263-265, 1952.

[58] Lemaire, B. "About the Convergence of the Proximal Method", In: Advances in Optimization, *Lecture Notes in Economics and Mathematical Systems*, v. 382, Springer-Verlag, Berlin, 39-51, 1992.

[59] Luenberger, D. G., *Linear and Nonlinear Programming*. 2 ed., Massachusetts, Addison-Wesley Publishing Company, 1984.

[60] Maros, I. and Mészáros, C., "A Repository of Convex Quadratic Programming Problems", *Optimization Methods and Software*, v. 11-12, pp. 671-681, 1999.

[61] Martinet, B., "Détermmination Approchée d'un Point Fixe d'une Application Pseudo-Contractante", *Comptes Rendus Hebdomadaires des Séances de L'cademie des Sciences*, v. 274, pp. A163-A165, 1970.

[62] Martinet, B., "Regularisation d'Inéquations Variationelles par Approximations Successives", *Revue Française d'Informatique et de Recherche Opérationnelle*, v. 4, pp. 154-159, 1970.

[63] Martinet, B., "Perturbation des Méthodes d'Optimisation: Applications", *R.A.I.R.O, Analyse Numérique*, v. 12, pp. 153-171, 1978.

[64] Mas-Colell, A., Whinston, M. D. and Green, J. R., *Microeconomic Theory*. New York, Oxford University Press, 1995.

[65] Martinez-Legaz, J. E., "Quasiconvex Duality Theory by Generalized Conjugation Methods", *Optimization*, v. 19, pp. 603-652, 1988.

[66] Megiddo, N. and Shub, M. "Boundary Behavior of Interior Point Algorithms in Linear Programming", *Mathematics of Operations Research*, v. 14, pp. 97-146, 1989.

[67] Monteiro, R. D. C. and Adler, I. "Interior Path Following Primal-Dual Algorithms. Part I: Linear Programming", *Mathematical Programming*, v. 44, pp. 27-42, 1989.

[68] Monteiro, R. D. C., Adler, I. and Resende, M. G. C., "A Polynomial-Time Primal-Dual Affine Scaling Algorithm for Linear and Convex Quadratic Programming and Its Power Series Extensions", *Mathematics of Operations Research*, v. 15, pp. 191-214, 1990.

[69] Monteiro, R. D. C., Tsuchiya, T. and Wang, Y., "A Simplified Global Convergence Proof of the Affine Scaling Algorithm", *Annals of Operations Research*, v. 47, pp. 443-482, 1993.

[70] Monma, C. and Morton, A., "Computational Experiments with a Dual Affine Variant of Karmarkar's Method for Linear Programming", *Operations Research Letters*, v. 6, pp. 261-267, 1987.

[71] Moreau, J. J., "Proximité et Dualité dans un Espace Hilbertien", *Bulletin de la Societé Mathématique de France*, v. 93, pp. 273-299, 1965.

[72] Murtagh, B. and Saunders, M., *MINOS 5.0 User's Guide*. Technical Report SOL 77-9, Systems Optimization Laboratory, Department of Operations Research Letters, Stanford, CA, USA, 1977.

[73] Pinto, A. W. M., Oliveira, P. R. and Cruz Neto, J. X., *A New Class of Potential Affine Algorithms for Linear Convex Programming*. Technical Report ES - 576/02, Department of Systems Engineering and Computer Science, COPPE, Federal University of Rio de Janeiro, Rio de Janeiro, RJ, Brasil, 2002.

[74] Quiroz, E. A. P. and Oliveira, P. R., "Proximal Methods for Quasiconvex Minimization on Euclidean Spaces, Nonnegative Orthants and Hadamard Manifolds", Preprint, 2006.

[75] Quiroz, E. A. P. and Oliveira, P. R., "Proximal Point Methods for Quasiconvex and Convex Functions with Bregman Distances on Hadamard Manifolds". To appear in *Journal of Convex Analysis.*

[76] Renegar, J., "A Polynomial-Time Algorithm, Based on Newton's Method, for Linear Programming", *Mathematical Programming,* v. 40, pp. 59-93, 1988.

[77] Resende, M. G. C. and Veiga, G., *Computational Study of Two Implementations of the Dual Affine Scaling Algorithm.* Technical Report, AT&T Bell Laboratories, Murray Hill, NJ, USA, 1990.

[78] Resende, M. G. C. and Veiga, G., "An Implementation of the Dual Affine Scaling Algorithm for Minimun Cost flow on Bipartite Uncapaciated Networks", *SIAM Journal on Optimization,* v. 3, pp. 516-537, 1993.

[79] Resende, M. G. C. and Veiga, G., "An Efficient Implementation of Network Interior Point Method". In: *DIMACS Series in Discrete Mathematics and Teoretical Computer Science,* v. 12, *Network Flows and Matching: First DIMACS Implementation Challenge,* D. S. Johnson and C. C. McGeoch (eds.), pp. 299-348, 1993

[80] Rockafellar, R. T., *Convex Analysis.* New Jersey, Princeton University Press, 1970.

[81] Rockafellar, R. T., "Augmented Lagrangians and Applications of Proximal Point Algorithm in Convex Programming", *Mathematics of Operation Research,* v. 1, pp. 97-116, 1976.

[82] Rockafellar, R. T., "Monotone Operators and the Proximal Point Algorithm", *SIAM Journal on Control and Optimization,* v. 14, n. 5, pp. 877-898, 1976.

[83] Saigal, R., *Efficient Variants of the Affine Scaling Method.* Technical Report 93-21, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, USA, 1993.

[84] Souza, S. S., Oliveira, P. R., Cruz Neto, J. X., et al., "A Proximal Point Algorithm with Bregman Distances for Quasiconvex Optimization over the Positive Orthant", Preprint, 2006.

[85] Stoer, J., and Witzgall, C., *Convexity and Optimization in Finite Dimensions I.* Berlin, Springer-Verlag, 1970.

[86] Teboulle, M., "Entropic Proximal Mappings with Applications to Nonlinear Programming", *Mathematics of Operation Research*, v. 17, pp. 670-690, 1992.

[87] Teboulle, M., "Convergence of Proximal-like Algorithms", *SIAM Journal on Optimization*, v. 7, pp. 1069-1083, 1997.

[88] Tseng, P. and Bertsekas, D. P., "On the Convergence of the Exponential Multiplier Method for Convex Programming", *Mathematical Programming*, v. 60, pp. 1-19, 1993.

[89] Tsuchiya, T., *A Study on the Global and Local Convergence Properties of the Interior Point Algorithms for Linear Programming.* Ph.D. thesis, Faculty of Engineering, University of Tokyo, Tokyo, 1991. (In Japanese).

[90] Tsuchiya, T. and Monteiro, R. D. C., "Superlinear Convergence of the Affine Scaling Algorithm", *Mathematical Programming*, v. 75, n. 1, pp. 77-110, 1996.

[91] Tsuchiya, T. and Muramatsu, M., "Global Convergence of a Long-Step Affine Scaling Algorithm for Degenerate Linear Programming Programs", *SIAM Journal on Optimization*, v. 5, n. 3, pp. 525-551, 1995.

[92] Tsuchiya, T. and Tanabe, K., "Local Convergence Properties of New Methods in Linear Programming", *The Journal of the Operations Research Society of Japan*, v. 33, n. 1, pp. 22-45, 1990.

[93] Vanderbei, R. J., Meketon M. J. and Freedman B. A., "A Modification of Karmarkar's Linear Programming Algorithm", *Algorithmica*, v. 1, pp. 395-407, 1986.

[94] Wright, S. J., *Primal-Dual Inteior Point Methods.* Philadelphia, Society for Industrial and Applied Mathematics (SIAM), 1997.