UNIVERSIDADE FEDERAL DO PARANÁ

ADREAN DE OLIVEIRA CEBOLA

RESOLUÇÃO DE PROGRAMAS LINEARES COM MÉTODO DE PONTOS INTERIORES

CURITIBA
2017

ADREAN DE OLIVEIRA CEBOLA

RESOLUÇÃO DE PROGRAMAS LINEARES COM MÉTODO DE PONTOS INTERIORES

Monografia apresentada à disciplina de Projeto de Matemática Industrial como requisito à conclusão do curso de Bacharelado em Matemática Industrial, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Dr. Abel Soares Siqueira

CURITIBA
2017

Agradecimentos

Agradeço principalmente meus pais, Eliane e Elias, que me estiveram comigo em todos os momentos me apoiando.

Agradeço também aos meus amigos, que durante os anos de faculdade me apoiaram.

Agradeço a minha namorada, Marceli, por me ajudar a ter determinação e foco para terminar este trabalho.

Agradeço ao Prof.Dr. Abel Soares Siqueira, por ser o meu orientador neste trabalho e me dar o suporte necessário.

E agradeço a todas as pessoas que me ajudaram durante esta etapa muito importante de minha vida.

Muito obrigado a todos.

Resumo

Este trabalho será um estudo sobre métodos de Pontos Interiores para Programação Linear. Serão apresentados teoremas de convergência e um método eficiente para resolução de problemas. Comparamos o método de Pontos Interiores com o Clp em relação ao tempo de execução e número de iterações. Os problemas utilizados para os testes são encontrados no repositório Netlib.

Palavras-chaves: Programação Linear, Método de Pontos Interiores.

Abstract

This work will be a study on Interior Point methods to Linear Programming. Convergence theorems and an efficient method for solving problems will be presented. We compare the interior point method with Clp regarding the time and number of iterations. The problems used in the tests are found in the netlib repository.

Keywords: Linear Programming, Interior Point Methods

Lista de ilustrações

Lista de tabelas

Lista de Algoritmos

Sumário

1 Introdução

A Programação Linear foi desenvolvida durante a segunda guerra mundial, quando um sistema para maximizar a eficiência dos recursos era de extrema importância. A expressão “Programação” era um termo militar que se referia às atividades tais como planejar horários de forma eficiente. Dantzig [6], membro da Força Aérea dos EUA, desenvolveu o método Simplex em 1947 para providenciar um algoritmo a fim de resolver problemas de programação que tinham estruturas lineares. Desde então, especialistas de vários ramos, tais como matemáticos e economistas desenvolveram teoria e algoritmos para problemas de Programação Linear.

Por volta de 1980, anos depois de muita pesquisa na área de programação linear, foi descoberto que muitos problemas lineares podem ser resolvidos eficientemente usando formulações e algoritmos da programação não-linear e equações não-lineares. Esses métodos caminhavam até a solução gerando iterações dentro da região factível, assim, ficaram conhecidos como métodos de Pontos Interiores. Apoś os métodos de pontos interiores, uma subclasse desses métodos ficou conhecida como Métodos Primais-Duais, por serem métodos que utilizam as informações do problema Primal e do problema Dual. São métodos com abordagens práticas mais eficientes e que se provaram fortes competidores para o método Simplex em problemas grandes.

Os métodos de Pontos Interiores surgiram da necessidade de um algoritmo com propriedades teóricas melhores do que o Simplex, já que ele pode ser ineficiente em problemas patológicos. Para problemas práticos, o método Simplex é eficiente, mas a complexidade do seu pior caso motivou o desenvolvimento de um novo algoritmo com a garantia de um desempenho melhor.

Cada iteração de um método de Pontos Interiores é computacionalmente cara, mas faz um progresso significante na procura da solução, enquanto que o método Simplex geralmente exige um grande número de iterações baratas. Geometricamente, o método Simplex traça o caminho para encontrar a solução através da fronteira da região factível, testando uma sequência de vértices até encontrar a solução ótima. Já o método de Pontos Interiores aproxima-se da fronteira apenas no limite, caminhando pelo interior da região factível, mas nunca se encontra na fronteira da região factível.

2 Programação Linear

Neste capítulo vamos utilizar algumas referências nessa área, tais como Bertsimas e Tsitsiklis [8], com exemplos de programação linear e o problema dual, Wright [3], para teorias de dualidade e, também, Nocedal e Wright [10], para algumas relações de dualidade.

Programação Linear (PL) é um tipo de problema de otimização que pode ser escrito da seguinte forma geral,

$$\begin{aligned} \min \quad & c^T x \\ \text{s.a.} \quad & A\,x = b, \\ & x \geq 0, \end{aligned} \tag{P}$$

onde o objetivo pode ser tanto minimizar quanto maximizar uma função linear. O vetor $x \in \mathbb{R}^n$ é um vetor coluna com as variáveis a serem encontradas, $c \in \mathbb{R}^n$ o vetor de coeficientes do problema e $b \in \mathbb{R}^m$ o vetor com valores aos quais as restrições da matriz $A \in \mathbb{R}^{m \times n}$ devem se igualar. A função $c^T x$ é chamada de função objetivo.

Se um ponto x satisfaz as restrições $Ax = b$ e $x \geq 0$, então esse ponto é dito factível e a série de todos os pontos factíveis é dito como um conjunto factível. Dizemos que o problema (P) é infactível se o conjunto factível é vazio, e que é ilimitado se a função objetivo $c^T x$ é ilimitada inferiormente, isto é, existe uma sequência de pontos x^k tal que $c^T x^k \downarrow -\infty$.

Um exemplo de problema de programação linear. É um problema de produção onde o objetivo é decidir quais mercadorias produzir com o intuito de maximizar o lucro da empresa. Se $b_i,\ i = 1, 2, \ldots, m$ representa a quantidade disponível da i-ésima matéria-prima, a j-ésima mercadoria, $j = 1, 2, \ldots, n$, exige a_{ij} unidades da i-ésima matéria-prima e o resultado é o lucro de c_j por unidade produzida. A formulação para este problema é a seguinte [8]

$$\begin{aligned} \max \quad & c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \\ \text{s.a.} \quad & a_{i1} x_1 + a_{i2} x_2 + \cdots + a_{in} x_n \leq b_i,\ i = 1, \ldots, m, \\ & x_j \geq 0,\ j = 1, \ldots, n. \end{aligned}$$

Podemos transformar um problema com restrições de desigualdade em um problema com restrições de igualdade, apenas adicionando uma variável de folga na restrição. Utilizando a formulação do problema (P), se temos uma restrição de desigualdade $Ax \leq b$,

adicionamos uma variável $z = (z_1, z_2, \ldots, z_n)^T$, conhecida como variável de folga. Assim temos

$$\begin{aligned} \min \quad & c^T x \\ \text{s.a.} \quad & Ax + z = b, \\ & x, z \geq 0. \end{aligned}$$

Concluímos que um problema com restrições de desigualdade pode ser transformado no problema padrão. Assim é necessário somente o desenvolvimento de um algoritmo que resolva a forma padrão (P).

Associado com qualquer problema de programação linear, também chamado de Primal (P), está o problema Dual. Para deduzir o Dual, utilizamos a função Lagrangiana

$$\mathcal{L}(x, \lambda, s) = c^T x - \lambda^T (b - Ax) - s^T x, \tag{2.1}$$

e definimos a função objetiva do Dual como sendo

$$q(\lambda, s) = \inf_{x \geq 0} \mathcal{L}(x, \lambda, s).$$

Podemos notar que o cálculo do ínfimo exige que encontremos o mínimo global da função $\mathcal{L}(x, \lambda, s)$ para λ e s. E como estamos em programação linear, todos os minimizadores locais são minimizadores globais, então o problema Dual pode ser definido encontrando o mínimo da função $q(\lambda, s)$. Assim, temos que:

$$q(\lambda, s) = \min_{x \geq 0}[c^T x - \lambda^T (b - Ax) - s^T x] \leq c^T x^* - \lambda^T (b - Ax^*) - s^T x^*) = c^T x^*,$$

onde a última igualdade é devido a x^* ser a solução ótima do problema primal e satisfaz $Ax^* = b$. Assim, temos que $q(\lambda, s)$ é um limitante inferior para o valor ótimo do problema (P).

Temos também que

$$\begin{aligned} q(\lambda, s) &= \min_{x \geq 0}[c^T x - \lambda^T (b - Ax) - s^T x] \\ &= -b^T \lambda + \min_{x \geq 0} x^T (c + A^T \lambda - s). \end{aligned}$$

Podemos notar que

$$\min_{x \geq 0} x^T (c + A^T \lambda - s) = \begin{cases} 0, & \text{se } c + A^T \lambda - s = 0, \\ -\infty, & \text{caso contrário.} \end{cases}$$

Portanto, concluímos que problema dual pode ser definido da seguinte maneira

$$\begin{aligned} \max \quad & b^T \lambda \\ \text{s.a.} \quad & A^T \lambda + s = c, \\ & s \geq 0. \end{aligned} \tag{D}$$

2.1 Condições de Otimalidade e Dualidade

As soluções das equações em (P) e (D) são caracterizadas pelas Condições de Otimalidade de Primeira Ordem, conhecidas também como condições de Karush-Kuhn-Tucker (KKT), que são definidas através do seguinte teorema:

Teorema 1 (Teorema A.1 de [3])**.** *Suponha que x é uma solução de* (P)*. Então existem vetores λ e s tais que as seguintes condições são satisfeitas:*

$$A^T\lambda + s = c, \tag{2.2a}$$

$$Ax = b, \tag{2.2b}$$

$$x_i s_i = 0,\ i = 1, 2, \ldots, n, \tag{2.2c}$$

$$x, s \geq 0. \tag{2.2d}$$

As condições (2.2a) e (2.2b) são as viabilidades do problema primal (P) e dual (D), respectivamente. A condição (2.2c) é a condição de complementariedade.

Das condições descritas no Teorema 1, podemos provar alguns teoremas sobre a relação entre o problema (P) e (D). Tais teoremas que podem ser encontrados em Nocedal e Wright [10].

Teorema 2 (Teorema 12.11 de [10])**.** *[Dualidade Fraca] Seja $x \in \mathbb{R}^n$ um ponto factível para o problema primal* (P) *e $\lambda \in \mathbb{R}^m$, $s \in \mathbb{R}^n$ factíveis para o problema dual* (D)*, então*

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

Demonstração. Podemos realizar a prova em apenas uma linha, utilizando as relações já apresentadas, assim temos

$$c^T x = (A^T\lambda + s)^T x = \lambda^T(Ax) + s^T x = b^T\lambda + s^T x \geq b^T\lambda.$$

□

Podemos notar que o valor da função objetivo do problema Primal é limitado inferiormente pelo valor objetivo do problema Dual. E o valor da função objetivo do problema Dual é limitado superiormente pelo valor da função objetivo do problema Primal.

Teorema 3 (Teorema 13.1 de [10])**.** *[Dualidade Forte]*

(i) Se o problema primal (P) *ou o problema dual* (D) *tem uma solução, então ambos tem solução e seus valores ótimos são iguais.*

(ii) Se o problema primal (P) *ou o problema dual* (D) *são ilimitados, então o outro é infactível.*

Demonstração. (i) Supondo que o problema primal tenha uma solução ótima x^*, então pelas condições apresentadas em (2.2), existem vetores λ^* e s^* que são solução do problema dual. Assim, temos

$$c^T x^* = (A^T \lambda^* + s^*)^T x^* = (\lambda^*)^T (A x^*) + s^T x^* = b^T (\lambda)^* + (s^*)^T x^*,$$

como (x^*, λ^*, s^*) é solução para as equações (2.2) e temos das condições (2.2a), (2.2c) e (2.2d), que $(x^*)^T s^* = 0$, assim

$$c^T x^* = b^T \lambda^* + (s^*)^T x^* = b^T \lambda^*.$$

Para provarmos que o Dual tem solução ótima igual a solução do Primal, apenas supomos que o problema Dual tem uma solução.

(ii) Suponha que o problema primal é ilimitado, ou seja, existe uma sequência de pontos $x^k, k = 1, \ldots$ tal que

$$c^T x^k \downarrow -\infty, \quad Ax^k = b, \quad x^k \geq 0.$$

Suponha também que o problema dual é factível, ou seja, existem vetores λ e s tal que $A^T \lambda + s = c$ e $s \geq 0$. Portanto,

$$(A^T \lambda + s)^T x^k = c^T x^k \Rightarrow \lambda^T A x^k \leq c^T x^k.$$

Da equação (2.2b), temos

$$\lambda^T A x^k = \lambda^T b \leq c^T x^k \downarrow -\infty,$$

como supomos que λ é factível e b é um vetor dado, $\lambda^T b$ é um valor finito e isso nos leva a uma contradição. Então, o problema dual deve ser infactível. Para provarmos que o Primal é infactível, basta supor que o problema Dual é ilimitado.7 □

Este resultado nos diz que os problema Primal e Dual estão totalmente interligados pelos valores de suas funções objetivo.

Para cada solução (x^*, λ^*, s^*), sabemos da condição de complementariedade (2.2c) que $x_j^* = 0$ e/ou $s_j^* = 0$ para todo $j = 1, 2, \ldots, n$. Dessa maneira, vamos definir dois conjuntos

$$\mathcal{P}_{\mathcal{B}} = \{j \in 1, 2, \ldots, n \mid x_j^* \neq 0 \text{ para algum } x^* \in \Omega_P\}$$

e

$$\mathcal{D}_N = \{j \in 1, 2, \ldots, n \mid s_j^* \neq 0 \text{ para algum } (\lambda^*, s^*) \in \Omega_D\},$$

onde $\Omega_P = \{x \mid \text{é solução de (P)}\}$ e $\Omega_D = \{(\lambda, s) \mid \text{é solução de (D)}\}$ são conjuntos de soluções dos problemas primal e dual.

Note que não é possível que um índice j pertença a ambos conjuntos $\mathcal{P}_\mathcal{B}$ e $\mathcal{D}_N$, já que haveria uma solução do primal x^* e uma solução do dual (λ^*, s^*) tal que $x_j^* > 0$ e $s_j^* > 0$, porém, teríamos $x_j^* s_j^* > 0$, contradizendo a condição de complementariedade (2.2c). Já quando temos a união desses conjunto $\mathcal{P}_\mathcal{B} \cup \mathcal{D}_N = \{1, 2, \ldots, n\}$, temos as soluções dos problemas primal e dual satisfazendo as condições (2.2), em especial a condição de complementariedade estrita (2.2c), já que os conjuntos foram definidos para esse fim. A definição de união desses conjuntos é também conhecido como Teorema de Goldman-Tucker [3].

Teorema 4 (Teorema 2.4 de [3])**.** *[Teorema de Goldman-Tucker]* $\mathcal{P}_\mathcal{B} \cup \mathcal{D}_N = \{1, 2, \ldots, n\}$. *Ou seja, existe pelo menos uma solução do primal* x^* *e uma solução do dual* (λ^*, s^*) *tal que* $x^* + s^* > 0$.

Soluções dos problemas primal e dual que satisfazem esse teorema, são conhecidas como soluções estritamente complementares.

Lema 1 (Lema 2.6 de [3])**.** *[Lema de Farkas] Para cada matriz* $G \in \mathbb{R}^{p \times n}$ *e cada vetor* $g \in \mathbb{R}^n$*, ou*

(i) $Gd \geq 0, g^T d < 0$*, tem uma solução* $d \in \mathbb{R}^n$*; ou*

(ii) $G^T z = g, z \geq 0$*, tem uma solução* $z \in \mathbb{R}^p$*,*

mas nunca ambos.

Podemos reinterpretar esse teorema geometricamente, considerando uma matriz de duas linhas no $\mathbb{R}^2$

$$G = \begin{pmatrix} g_1^T \\ g_2^T \end{pmatrix},$$

sendo g_1^T e g_2^T as linhas da matriz G. Para o caso (i), existe uma solução $d \in \mathbb{R}^n$, tal que $g_1^T d \geq 0$, $g_2^T d \geq 0$ e $g^T d < 0$, ou seja, existe um plano definido pelo vetor d que separa o vetor g do espaço formado pelas linhas de G, como podemos ver na Figura 1.

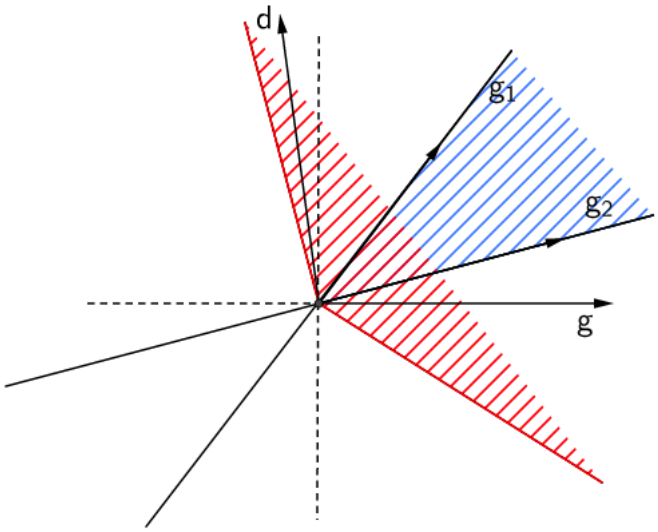

Figura 1 – Lema de Farkas, caso (i)

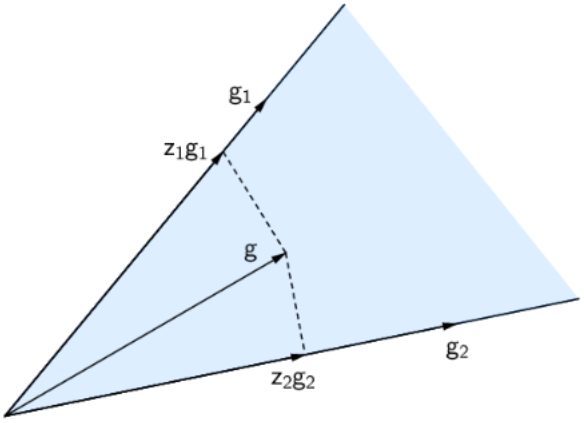

Figura 2 – Lema de Farkas, caso (ii)

E para o caso (ii), existe um vetor $z \in \mathbb{R}^p$ que é solução, onde z_1 e z_2 são as componentes do vetor z, tal que $g_1z_1 + g_2z_2 = g$ e $z \geq 0$. A Figura 2 nos mostra que qualquer vetor contido neste cone pode ser uma combinação linear das linhas de G.

Para ambos os casos, o espaço formado pelas linhas da matriz G são Cones, conforme definição:

Definição 2 (Definição 7.2 de [9])**.** Um subconjunto não vazio $C \subset \mathbb{R}^n$ é um cone quando, $\forall t \geq 0$ e $d \in C$ tem-se $td \in C$.

Nos casos (i) e (ii) o cone formado pelas linhas de G satisfaz essa definição. Além disso, o cone também é convexo, ou seja, dado vetores $x, y \in C$ e escalares $\alpha, \beta > 0$, a combinação linear $\alpha x + \beta y \in C$.

O Lema de Farkas pode ser utilizado para provar alguns resultados como a Dualidade Fraca (Teorema 2), a Dualidade Forte (Teorema 3), e as condições de KKT (Teorema 1).

3 Método Simplex

Neste capítulo vamos discutir o método Simplex utilizando algumas definições, teoremas e o algoritmo para uma iteração do método, de acordo com Nocedal e Wright [10].

Vamos nos referir a x como um ponto factível básico, se existe um subconjunto $\mathcal{B}$ do conjunto de índices $\{1, 2, \ldots, n\}$ tal que

- $\mathcal{B}$ contém exatamente m índices;
- $i \notin \mathcal{B} \Rightarrow x_i = 0$ (isto é, a restrição $x_i \geq 0$ pode ser inativa somente se $i \in \mathcal{B}$);
- A matriz $B \in \mathbb{R}^{m \times m}$ definida por $B = [A_i]_{i \in \mathcal{B}}$ é não singular, onde A_i é a i-ésima coluna de A.

Um conjunto $\mathcal{B}$ satisfazendo essas propriedades é chamado de base para o problema (P). A matriz correspondente B é chamada de matriz base.

A partir de $\mathcal{B}$ e das condições (2.2), podemos derivar valores tanto para as variáveis primais x quanto para as duais (λ, s). Definindo um conjunto de índices não básicos $\mathcal{N}$ como complemento de $\mathcal{B}$, ou seja,

$$\mathcal{N} = \{1, 2, \ldots, n\} \setminus \mathcal{B}.$$

Assim como B é a matriz básica onde as colunas são A_i para $i \in \mathcal{B}$, usaremos N como a matriz não básica definida por $N = [A_i]_{i \in \mathcal{N}}$. Assim, particionaremos os elementos de x, s e c de acordo com os conjuntos $\mathcal{B}$ e $\mathcal{N}$

$$x_B = [x_i]_{i \in \mathcal{B}}, \quad x_N = [x_i]_{i \in \mathcal{N}}, \tag{3.1}$$

$$s_B = [s_i]_{i \in \mathcal{B}}, \quad s_N = [s_i]_{i \in \mathcal{N}}, \tag{3.2}$$

$$c_B = [c_i]_{i \in \mathcal{B}}, \quad c_N = [c_i]_{i \in \mathcal{N}}. \tag{3.3}$$

Da condição (2.2b), temos que

$$Ax = Bx_B + Nx_N = b.$$

E então

$$x_B = B^{-1}b, \quad x_N = 0. \tag{3.4}$$

Uma vez que utilizamos somente pontos factíveis básicos e sabemos que B é não singular e $x_B \geq 0$, então a escolha de x satisfaz as condições (2.2b) (restrição linear para o Primal) e (2.2d).

Escolhemos s que satisfaça a condição de complementariedade (2.2c), fazendo $s_B = 0$. As componentes restantes λ e s_N podem ser encontradas usando (3.1), $c = c_B + c_N$ e $s_B = 0$ para obter

$$B^T\lambda + s_B = c_B \Rightarrow B^T\lambda = c_B, \quad N^T\lambda + s_N = c_N. \tag{3.5}$$

Como $B \in \mathbb{R}^{m\times m}$ é não singular, temos que

$$\lambda = B^{-T}c_B. \tag{3.6}$$

Assim, temos de (3.5)

$$s_N = c_N - N^T\lambda = c_N - (B^{-1}N)^T c_B. \tag{3.7}$$

A única condição que não aplicamos explicitamente é a condição de não negatividade $s \geq 0$. A componente s_B certamente satisfaz essa condição, pela escolha de $s_B = 0$. Se o vetor s_N definido por (3.7) também satisfaz $s_N \geq 0$, encontramos os valores ótimos (x, λ, s) e podemos terminar o algoritmo. Entretanto, caso não sejam ótimos, uma ou mais componentes de s_N são negativas. O novo índice para entrar na base $\mathcal{B}$ é escolhido para ser um índice $q \in \mathcal{N}$ tal que $s_q < 0$. A função objetivo $c^T x$ irá decrescer quando permitimos que x_q se torne positivo se, e somente se, $s_q < 0$ e se é possível aumentar x_q para longe de zero enquanto mantém a factibilidade de x. A maneira como é alterada a base $\mathcal{B}$ e a mudança de x e s são descritas abaixo:

- permitir x_q para aumentar de zero durante o próximo passo;
- descobrir os efeitos de aumentar x_q no vetor básico atual x_B, já que queremos manter a factibilidade com respeito à restrição $Ax = b$;
- continuar aumentando x_q até que uma das componentes de x_B (x_p, por exemplo) seja direcionada para zero ou determinar que tal componente não existe (caso ilimitado);
- remover o índice p (conhecido como índice de saída) de $\mathcal{B}$ e substituí-lo pelo índice q de entrada.

Uma vez que a nova iteração x^+ e a iteração atual x satisfaçam $Ax = b$ e que $x_N = 0$ e $x_i^+ = 0$ para $i \in \mathcal{N} \setminus q$, temos

$$Ax^+ = Bx_B^+ + A_q x_q^+ = Bx_B = Ax.$$

Multiplicando essa equação por B^{-1}, obtemos

$$x_B^+ = x_B - B^{-1}A_q x_q^+. \tag{3.8}$$

Geometricamente falando, (3.8) é geralmente um movimento ao longo de uma aresta da região factível que diminui $c^T x$ até encontrar um vértice. Neste vértice, uma nova restrição $x_p \geq 0$ deve se tornar ativa, ou seja, uma das componentes de x_p, $p \in \mathcal{B}$, diminuiu para zero. Então, removemos o índice p da base $\mathcal{B}$ e substituímos por q.

Mostraremos como o passo definido por (3.8) afeta a função objetivo $c^T x$:

$$c^T x^+ = c_B^T x_B^+ + c_q x_q^+ = c_B^T x_B - c_B^T B^{-1} A_q x_q^+ + c_q x_q^+. \tag{3.9}$$

De (3.6) nós temos $c_B^T B^{-1} = \lambda^T$, enquanto que de (3.5), desde que $q \in \mathcal{N}$, temos $A_q^T \lambda = c_q - s_q$. Portanto,

$$c^T x^+ = c_B^T x_B - (c_q - s_q)x_q^+ + c_q x_q^+ = c^T x + s_q x_q^+.$$

Uma vez que q foi escolhido para obter $s_q < 0$, segue que o passo em (3.8) produz uma diminuição da função objetivo $c^T x$ para qualquer $x_q^+ > 0$. É possível que possamos aumentar o valor de x_q^+ infinitamente e nunca encontrar um novo vértice, ou seja, a restrição $x_B^+ = x_B - B^{-1}A_q x_q^+$ é válida para todo valor positivo de x_q^+ e, quando isso acontece, o problema linear é ilimitado. Neste caso o método Simplex identificou uma aresta que se encontra inteiramente dentro da região factível ao longo da qual a função objetivo $c^T x$ diminui para $-\infty$.

Definição 3. Uma base $\mathcal{B}$ é dita degenerada se $x_i = 0$ para algum $i \in \mathcal{B}$, onde x é solução factível básica correspondente a $\mathcal{B}$. O problema (P) é dito degenerado se existe ao menos uma base degenerada.

Se a base $\mathcal{B}$ é não degenerada, então garantimos que $x_q^+ > 0$ e podemos garantir uma diminuição estrita em $c^T x$ para esse passo. Se o problema (P) é não degenerado,

podemos ter certeza da diminuição de $c^T x$ em cada passo. Porém, se temos um problema degenerado, temos um $x_i = 0, i \in \mathcal{B}$, mesmo x sendo um ponto factível básico.

Algoritmo 1: Uma iteração do Método Simplex

Entrada: Dado $\mathcal{B}$, $\mathcal{N}$, $x_B = B^{-1}b \geq 0$, $x_N = 0$;
Resolva $B^T\lambda = c_B$ para λ;
Calcule $s_N = c_N - N^T\lambda$;
se $s_N \geq 0$ **então**
    Pare; (ponto ótimo encontrado);
fim
Escolha $q \in \mathcal{N}$ com $s_q < 0$ como índice de entrada;
Resolva $Bd = A_q$ para d;
se $d \leq 0$ **então**
    Pare; (problema é ilimitado);
fim
Calcule $x_q^+ = \min_{i|d_i>0}(x_B)_i/d_i$ e use p para denotar o minimizador i;
Atualize $x_B^+ = x_B - dx_q^+$, $x_N^+ = (0, \ldots, 0, x_q^+, 0, \ldots, 0)^T$;
Mude $\mathcal{B}$ adicionando q e removendo a variável básica correspondente a p de B;

Precisamos realçar 3 aspectos importantes na implementação:

(i) Decomposição LU de B para obter d e λ, já que não queremos usar B^{-1};

(ii) Seleção do índice de entrada q entre as componentes negativas de s_N. (Em geral, existem muitas componentes negativas);

(iii) Manipulação das bases degeneradas e passos degenerados, nos quais não é possível escolher um valor positivo de x_q^+ sem violar a factibilidade.

A manipulação apropriada desses itens é essencial para a eficiência da implementação do Simplex.

3.1 Método de Restrições Ativas

Assim como em todos os problemas de otimização em que restrições de desigualdade estão presentes, o objetivo principal do algoritmo é determinar quais das restrições são ativas e quais são inativas. O método Simplex pertence a uma classe de algoritmos de

otimização restrita conhecida como Método de Restrições Ativas, que mantêm explicitamente estimativas dos conjuntos de índices ativos e inativos que são atualizados em cada etapa do algoritmo.

Algoritmos de Programação Quadrática, Otimização Restrita e Programação Não Linear usam a mesma estratégia básica, como o Simplex, de fazer uma estimativa explícita do conjunto ativo e levando o passo na direção de um problema reduzido ao qual as restrições dessa estimativa são satisfeitas como igualdades. Quando a não linearidade entra no problema, muitas das ferramentas que fazem o método Simplex ser tão eficiente não são mais aplicáveis. No entanto, o método Simplex é visto como o antecedente dos métodos de conjunto ativo para otimização restrita.

Uma característica indesejável é que, embora eficiente em quase todos os problemas práticos, o método possui complexidade exponencial, isto é, o tempo de execução pode ser uma função exponencial do tamanho do problema, ou seja, se o tamanho do problema é muito grande, o tempo para encontrar uma solução ótima é exponencialmente maior. Assim, teóricos procuram por um algoritmo que tenha complexidade polinomial, ou seja, um algoritmo no qual o tempo de execução é limitado por uma função polinomial da quantidade de armazenamento necessária para definir o problema. Então, um método foi desenvolvido por Karmarkar [5], que descreve um algoritmo polinomial que se aproxima da solução pelo interior da região factível ao invés de caminhar pela fronteira como o método Simplex faz. O método de Karmarkar marcou o início de intensas pesquisas no campo de Pontos Interiores.

4 Pontos Interiores

As definições, teoremas e algoritmos foram desenvolvidos neste capítulo com referência em Wright [3], que trata somente de métodos de pontos interiores com ênfase em métodos primais-duais, e Nocedal e Wright [10], que abrange a maioria dos assuntos de otimização restrita e irrestrita.

O método de Pontos Interiores foi desenvolvido através da procura de algoritmos com propriedades teóricas melhores do que o método Simplex, apesar do Simplex ser eficiente em problemas práticos, ele falha em problemas patológicos e também possui um tempo de execução que é exponencial em relação ao tamanho do problema. Isto motivou Khachiyan [4] a desenvolver o primeiro método de Pontos Interiores, mas na prática esse método era pior que o Simplex. A partir do método Elipsóide, Karmarkar [5] desenvolveu outro método de Pontos Interiores e que ficou famoso por ser muito competitivo com o método Simplex. Seu método é baseado nos problemas Primal e Dual.

4.1 Métodos Primais

Um importante algoritmo primal é a função de aproximação logarítmica para o problema (P), definida como

$$\begin{aligned} \min \quad & c^T x - \tau \sum_{i=1}^{n} \log x_i \\ \text{s.a.} \quad & A\,x = b, \\ & x > 0. \end{aligned} \tag{4.1}$$

onde $\tau > 0$. Utilizando as condições em (2.2), a solução x_τ de (4.1) para cada valor τ satisfaz as condições

$$\tau X^{-1} e + A^T \lambda = c, \tag{4.2a}$$

$$Ax = b, \tag{4.2b}$$

$$x > 0, \tag{4.2c}$$

para algum $\lambda \in \mathbb{R}^m$ e $X = \text{diag}(x_1, x_2, \ldots, x_n)$, onde $e = (1, 1, \ldots, 1)^T$. Se definimos o vetor s como

$$s_i = \frac{\tau}{x_i} \qquad i = 1, 2, \ldots, n$$

as condições em (4.2) podem ser reescritas como

$$\begin{aligned} A^T\lambda + s &= c, && \text{(4.3a)} \\ Ax &= b, && \text{(4.3b)} \\ x_i s_i &= \tau, && \text{(4.3c)} \\ x, s &\geq 0. && \text{(4.3d)} \end{aligned}$$

Quando $\tau \downarrow 0$, x_τ se aproxima de alguma solução x^* do problema linear (P). Algoritmos baseados no subproblema de barreira (4.1) exploram esta propriedade encontrando aproximações para x_τ para valores de τ cada vez menores.

Algoritmos desenvolvidos com base na função logarítmica de barreira não se tornaram muito populares tanto em programação linear quanto não linear, principalmente porque a solução do problema de barreira fica mais difícil de encontrar quando o parâmetro de barreira τ tende para zero.

Para verificar como a função de barreira de comporta, podemos ver as curvas de nível para o seguinte exemplo

$$\begin{aligned} \min \quad & -x_1 - x_2 \\ \text{s.a.} \quad & 2x_1 + x_2 \leq 2, \\ & 2x_1 + 3x_2 \leq 4, \\ & x_1, x_2 \geq 0. \end{aligned} \tag{4.4}$$

Adicionando o termo de barreira e alterando o exemplo para a forma padrão, temos

$$\begin{aligned} \min \quad & -x_1 - x_2 - \tau\Big(\sum_{i=1}^{2} \log x_i + \sum_{i=1}^{2} \log s_i\Big) \\ \text{s.a.} \quad & 2x_1 + x_2 + s_1 = 2, \\ & 2x_1 + 3x_2 + s_2 = 4, \\ & x_1, x_2, s_1, s_2 > 0. \end{aligned}$$

Podemos ver na Figura 3, com parâmetro $\tau = 1$, como as curvas de nível do problema de barreira se comportam para o exemplo.

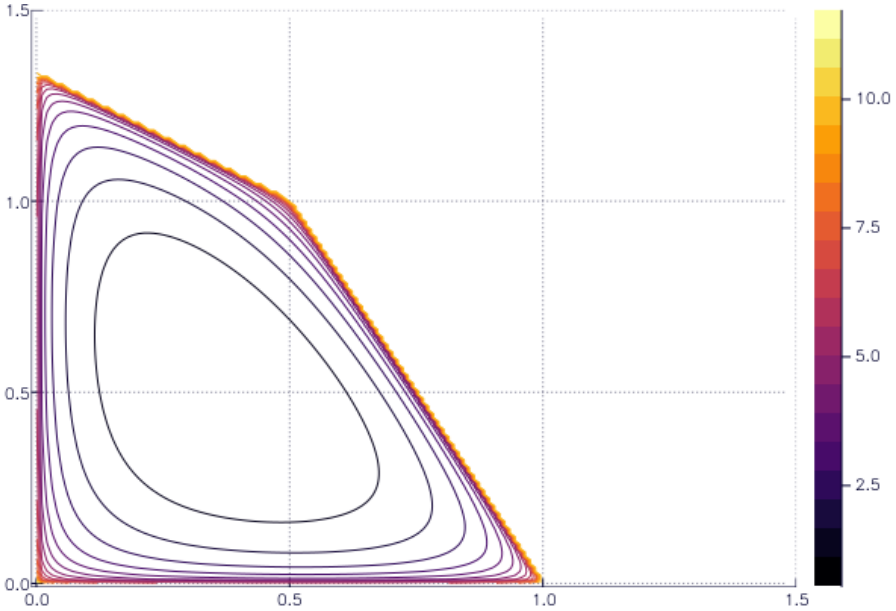

Figura 3 – Curvas de nível da função objetivo do problema de barreira com $\tau = 1$

Quando diminuímos o parâmetro τ, podemos ver que as curvas de nível ficam cada vez mais parecidas com a função objetivo do problema original, como podemos ver na Figura 4, com $\tau = 0.1$, e Figura 5, com $\tau = 0.001$.

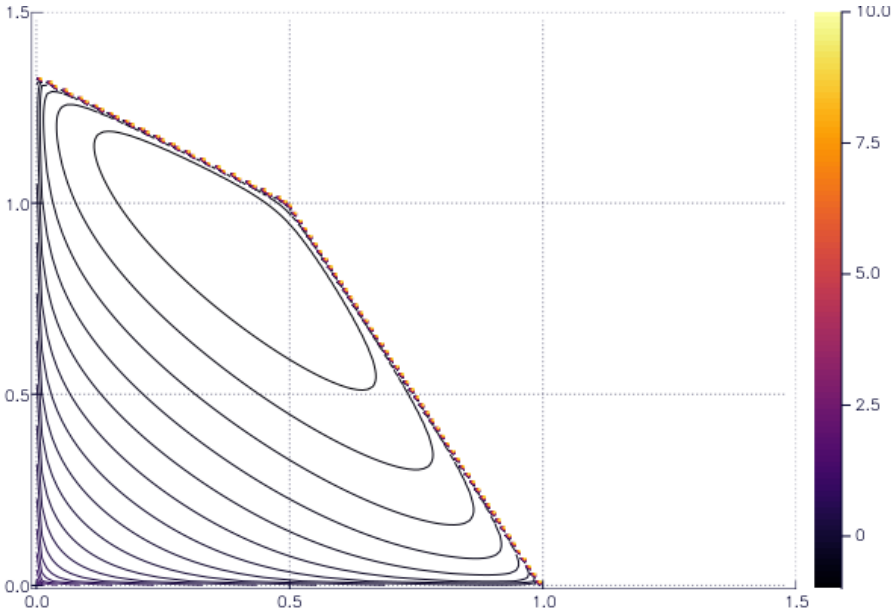

Figura 4 – Curvas de nível da função objetivo do problema de barreira com $\tau = 0.1$

Na Figura 6, vemos as curvas de nível da função objetivo original.

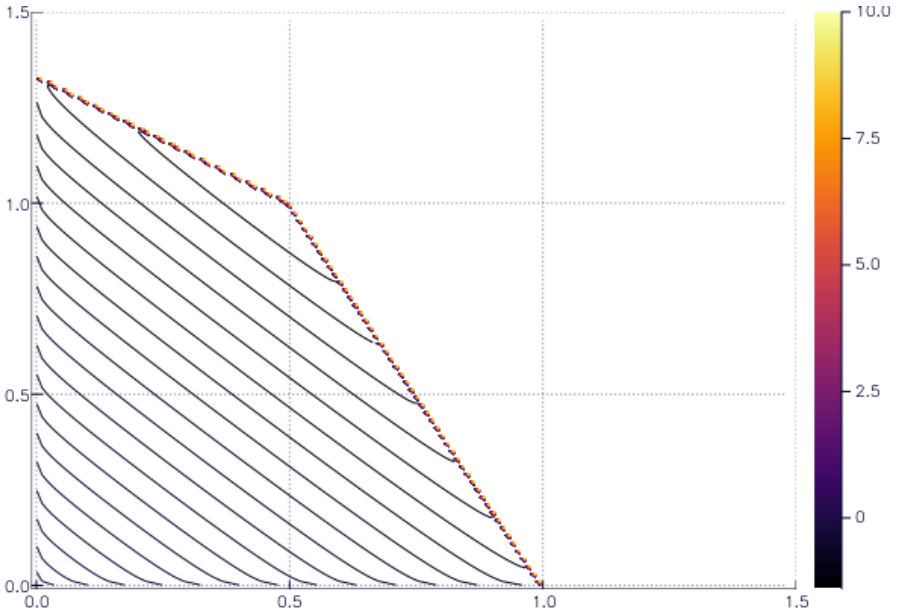

Figura 5 – Curvas de nível da função objetivo do problema de barreira com $\tau = 0.01$

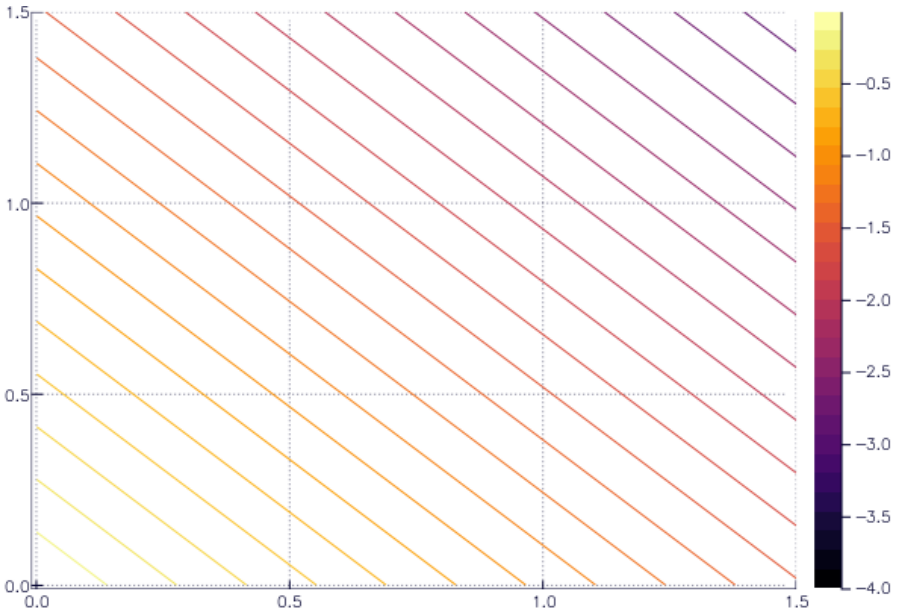

Figura 6 – Curvas de nível da função do problema original

Podemos ver, como discutido anteriormente, que quando τ está bem próximo de zero a função objetivo do problema de barreira (4.1) e a função objetivo do problema original (P) são muito parecidas, isso ocorre porque o termo de barreira se anula, pois τ é aproximadamente zero. Dessa maneira, as soluções para ambos os problemas são as mesmas.

4.2 Métodos Primais-Duais

De acordo com Wright [3], os métodos Primais-Duais encontram soluções (x^*, λ^*, s^*) do sistema linear (P) aplicando variações do Método de Newton para as condições de igualdade em (2.2) e modificando as direções e tamanho do passo para que as desigualdades $(x, s) \geq 0$ sejam satisfeitas estritamente em cada iteração. As equações (2.2a) e (2.2b) são lineares, as quais podem ser resolvidas mais facilmente. Entretanto, o problema se torna muito mais difícil quando adicionamos as desigualdades de não negatividade para x e s (2.2d).

Para obter métodos de pontos interiores para sistemas Primais-Duais apresentaremos as condições (2.2) de outra maneira, sendo $F : \mathbb{R}^{2n+m} \to \mathbb{R}^{2n+m}$:

$$F(x, \lambda, s) = \begin{pmatrix} A^T\lambda + s - c \\ Ax - b \\ XSe \end{pmatrix} = 0, \quad (x, s) \geq 0 \tag{4.5}$$

onde $X = \text{diag}(x_1, x_2, \ldots, x_n), \quad S = \text{diag}(s_1, s_2, \ldots, s_n), \quad e = (1, 1, \ldots, 1)^T$.

A maioria dos métodos de pontos interiores exige que as iterações sejam estritamente factíveis, ou seja, cada (x^k, λ^k, s^k) deve satisfazer as restrições de igualdade e de não negatividade do Primal e do Dual. Se definirmos um conjunto factível para o sistema Primal-Dual e outro estritamente factível como sendo:

$$\mathcal{F} = \{(x, \lambda, s) \mid Ax = b, A^T\lambda + s = c, (x, s) \geq 0\},$$

$$\mathcal{F}_0 = \{(x, \lambda, s) \mid Ax = b, A^T\lambda + s = c, (x, s) > 0\},$$

a condição de complementariedade estrita pode ser escrita como

$$(x^k, \lambda^k, s^k) \in \mathcal{F}_0,$$

onde (x^k, λ^k, s^k), são os pontos que satisfazem as condições de igualdade (2.2a) e (2.2b) e a condição (2.2d) estritamente, ou seja, $x^k, s^k > 0$.

Como a maioria dos algoritmos de otimização, métodos primais-duais têm dois passos básicos que são: determinar o passo e a medida de dualidade de cada ponto no espaço. Para determinar o passo é utilizado o método de Newton, assim como na maioria dos algoritmos de otimização sobre a função F, temos o seguinte:

$$J(x, \lambda, s) \begin{pmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{pmatrix} = -F(x, \lambda, s),$$

onde J é a Jacobiana de F.

Se o ponto atual é estritamente factível, ou seja,$(x, \lambda, s) \in \mathcal{F}_0$, então temos que o sistema pode ser descrito da seguinte forma:

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{pmatrix} = \begin{pmatrix} -r_c \\ -r_b \\ -XS\,e \end{pmatrix}, \tag{4.6}$$

onde $r_c = A^T\lambda + s - c$ e $r_b = Ax - b$. A direção obtida desse sistema é chamada de afim-escala. Porém, o passo completo nessa direção geralmente não é possível, pois pode violar os limites $(x, s) \geq 0$. Logo, para evitar isso, é feita busca linear sobre a direção de Newton, então a nova iteração é

$$(x, \lambda, s) + \alpha(\Delta x, \Delta\lambda, \Delta s)$$

para algum parâmetro $\alpha \in (0, 1]$. Infelizmente, muitas vezes podemos tomar apenas um pequeno passo na direção ($\alpha \ll 1$) antes de violar a condição $(x, s) > 0$. Portanto, a direção pura de Newton (4.6) muitas vezes não nos permite fazer muito progresso em direção a uma solução.

Muitos dos métodos primais-duais usam uma direção de Newton menos agressiva, uma direção que não vá diretamente para a solução das condições (2.2a), (2.2b) e (2.2c), mas sim para um ponto cujo produto $x_i s_i$ é reduzido para um valor menor a cada iteração. Mais precisamente, tomamos o passo de Newton em direção a um ponto que satisfaz $x_i s_i = \sigma\mu$, onde μ é a medida de dualidade e $\sigma \in [0, 1]$ o parâmetro centralizador. Assim, a direção de Newton modificada é a seguinte:

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{pmatrix} = \begin{pmatrix} -r_c \\ -r_b \\ -XS\,e + \sigma\mu e \end{pmatrix}. \tag{4.7}$$

Com isso, podemos definir uma estrutura simples que métodos Primais-Duais basicamente segue. Discutiremos este método na Seção 4.4.

Algoritmo 2: Estrutura simples de um método Seguidor de Caminho

Entrada: (x^0, λ^0, s^0) com $(x^0, s^0) > 0$

para $k = 1, 2, \ldots$ **faça**

Escolha $\sigma_k \in [0, 1]$ e resolva;

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{pmatrix} \begin{pmatrix} \Delta x^k \\ \Delta \lambda^k \\ \Delta s^k \end{pmatrix} = \begin{pmatrix} -r_c^k \\ -r_b^k \\ -X^k S^k e + \sigma^k \mu^k e \end{pmatrix}; \tag{4.8}$$

Calcule $\mu_k = (x^k)^T s^k / n$;

Calcule

$$(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + \alpha_k (\Delta x^k, \Delta \lambda^k, \Delta s^k); \tag{4.9}$$

Escolhendo α_k tal que $(x^{k+1}, s^{k+1}) > 0$;

fim

4.3 Trajetória Central

A trajetória central C é um arco de pontos estritamente factíveis parametrizada por um escalar $\tau > 0$, e cada ponto $(x_\tau, \lambda_\tau, s_\tau) \in C$ resolve as seguintes equações:

$$A^T \lambda + s = c, \tag{4.10a}$$

$$Ax = b, \tag{4.10b}$$

$$x_i s_i = \tau, \quad i = 1, 2, \ldots, n, \tag{4.10c}$$

$$(x, s) > 0. \tag{4.10d}$$

que são as mesmas condições (4.3) do problema de barreira visto na Seção 4.1.

Podemos perceber que essas condições são parecidas com as condições (2.2), se não fosse o escalar τ na condição (4.10c). A condição que tínhamos em (2.2c), foi substituída para que o produto $x_i s_i$ tenha o mesmo valor para qualquer i.

Podemos definir nosso arco C como

$$C = \{(x_\tau, \lambda_\tau, s_\tau) \,|\, \tau > 0\},$$

e as condições (4.10) acima da seguinte maneira:

$$F_\tau(x, \lambda, s) = \begin{pmatrix} A^T \lambda + s - c \\ Ax - b \\ XSe - \tau e \end{pmatrix}, \qquad (x_\tau, s_\tau) > 0. \tag{4.11}$$

O ponto $(x_\tau, \lambda_\tau, s_\tau)$ é definido unicamente para cada $\tau > 0$ se, e somente se, $\mathcal{F}_0$ é não vazio. Dessa maneira, a trajetória C é bem definida. A condição (4.10c) faz com que mantenha o produto $x_i s_i$ sempre estritamente positivo, ao mesmo tempo em que $\tau \downarrow 0$, as equações (4.10) e (2.2) ficam mais próximas.

A maioria dos algoritmos primais-duais tomam passos de Newton em direção aos pontos em C para os quais $\tau > 0$, em vez de passos puros do método de Newton em F_τ. Para descrever a direção de busca tendenciosa, vamos introduzir uma medida de dualidade μ e um parâmetro de centralização $\sigma \in [0, 1]$, sendo $\mu = x^T s/n$ a medida para o valor médio dos produtos $x_i s_i$. No caso factível, $r_c = r_b = 0$, assim, podemos definir as equações com essas medidas da seguinte maneira:

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -XSe + \sigma \mu e \end{pmatrix}. \tag{4.12}$$

O passo $(\Delta x, \Delta \lambda, \Delta s)$ é um passo de Newton em direção ao ponto $(x_{\sigma\mu}, \lambda_{\sigma\mu}, s_{\sigma\mu}) \in C$, ao qual $x_i s_i$ é sempre igual a $\sigma\mu$.

Se $\sigma = 1$, então (4.12) define uma direção centralizadora, onde $x_i s_i$ é sempre igual a μ. As direções de centralização são geralmente inclinadas fortemente em direção ao interior do octante não negativo e fazem pouco, se algum, progresso na redução para μ. Quando $\sigma = 0$, voltamos a equação (4.6) onde o resultado obtido é a direção de afim escala. Muitos algoritmos usam valores intermediários de σ no intervalo aberto $(0, 1)$ para a troca entre o objetivo de reduzir μ e estabelecendo centralidade para o problema.

Podemos ver na Figura 7 para o exemplo (4.4), como é a trajetória central até a solução do problema.

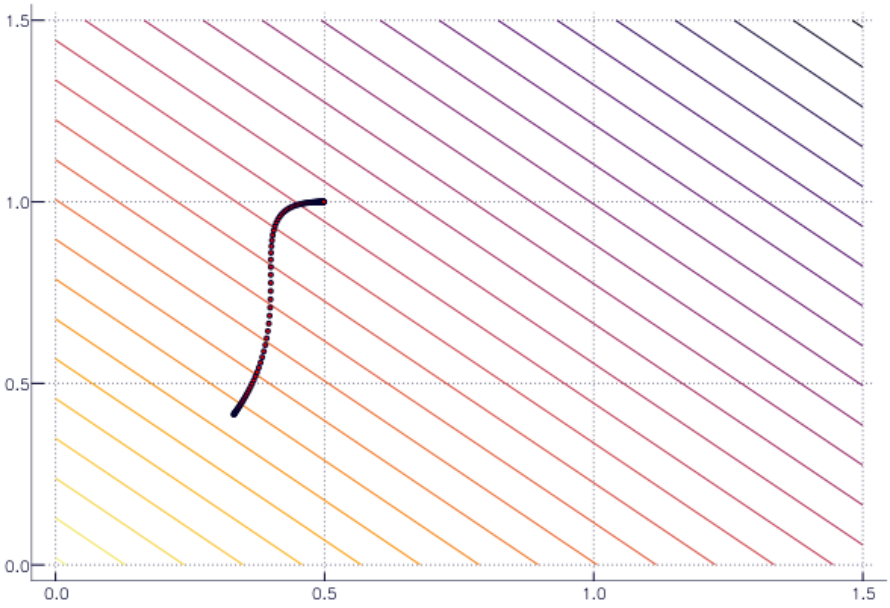

Figura 7 – Trajetória Central

4.4 Métodos Seguidores de Caminho

Um algoritmo de um método Seguidor de Caminho explicitamente restringe as iterações a uma vizinhança da trajetória central C e o segue até a solução do problema (P). A vizinhança exclui pontos (x, s) que se aproximam da região negativa. Portanto, direções de busca calculadas de cada ponto na vizinhança fazem pelo menos um progresso pequeno em direção ao conjunto de soluções.

As duas vizinhanças mais interessantes de C, são a vizinhança de norma 2, definida por

$$\mathcal{N}_2(\theta) = \{(x, \lambda, s) \in \mathcal{F}_0 \mid \|XSe - \mu e\|_2 \leq \theta\mu\}$$

para algum $\theta \in (0, 1)$, e a vizinhança unilateral de norma $-\infty$, definida por

$$\mathcal{N}_{-\infty}(\gamma) = \{(x, \lambda, s) \in \mathcal{F}_0 \mid x_i s_i \geq \gamma\mu \text{ para todo } i = 1, 2, \ldots, n\}$$

para algum $\gamma \in (0, 1)$.

Se um ponto está em $\mathcal{N}_{-\infty}(\gamma)$, cada produto $x_i s_i, i = 1, 2, \ldots, n$, deve ser pelo menos um pequeno múltiplo de γ de seu valor médio μ. Esta condição é bem modesta, pois podemos fazer $\mathcal{N}_{-\infty}(\gamma)$ englobar a maior parte da região $\mathcal{F}_0$ escolhendo γ perto de zero. Já a vizinhança $\mathcal{N}_2(\theta)$ é mais restritiva, visto que alguns pontos em $\mathcal{F}_0$ não pertencem a $\mathcal{N}_2(\theta)$ não importa quão perto θ é escolhido para seu limite superior que é 1.

Alguns métodos seguidores de caminho escolhem valores para o parâmetro de centralização σ (σ bem próximo de 1) de modo que passos unitários podem ser tomados ao longo da direção resultante de (4.12) sem deixar a vizinhança escolhida. Esses métodos, conhecidos como métodos seguidores de caminho de passo curto, fazem apenas um progresso pequeno em direção a solução porque é necessária uma vizinhança restritiva $\mathcal{N}_2(\theta)$ pra fazê-los funcionar.

Métodos seguidores de caminho de passo longo fazem escolhas menos conservadoras da medida de centralização σ. Como consequência, é preciso que seja feita busca linear ao longo da direção $(\Delta x, \Delta\lambda, \Delta s)$ para evitar sair da vizinhança escolhida. Ao fazer certas escolhas para σ, métodos de passo longo podem fazer progressos muito mais rápidos do que métodos de passo curto, particularmente, quando a vizinhança $\mathcal{N}_{-\infty}(\gamma)$ é usada.

O Algoritmo 3, conhecido como algoritmo seguidor de caminho de passo longo, que mostraremos a seguir é um caso especial do Algoritmo 2. Este algoritmo pode fazer um progresso rápido, pois é usada uma vizinhança ampla $\mathcal{N}_{-\infty}(\gamma)$, para valores de γ perto de zero. Ele depende de dois parâmetros, σ_{min} e σ_{max}, os quais são os limites inferior e superior do parâmetro de centralização σ_k. A direção de busca é obtida resolvendo a equação (4.8) escolhendo α_k o maior possível, sendo que estamos dentro da vizinhança $\mathcal{N}_{-\infty}(\gamma)$.

Algoritmo 3: Algoritmo Seguidor de Caminho de Passo Longo

Entrada: Dado $\gamma, \sigma_{min}, \sigma_{max}$ com $\gamma \in (0,1), 0 < \sigma_{min} \leq \sigma_{max} < 1$ e $(x^0, \lambda^0, s^0) \in \mathcal{N}_{-\infty}(\gamma)$;

para $k = 1, 2, \ldots$ **faça**

Escolha $\sigma_k \in [\sigma_{min}, \sigma_{max}]$;

Resolva

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{pmatrix} \begin{pmatrix} \Delta x^k \\ \Delta\lambda^k \\ \Delta s^k \end{pmatrix} = \begin{pmatrix} -r_c^k \\ -r_b^k \\ -X^k S^k e + \sigma^k \mu^k e \end{pmatrix}; \tag{4.13}$$

Calcule $\mu_k = (x^k)^T s^k / n$;

Escolha α_k como o maior valor possível em $[0, 1]$ tal que

$$(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + \alpha(\Delta x^k, \Delta\lambda^k, \Delta s^k) \in \mathcal{N}_{-\infty}(\gamma);$$

fim

4.5 Métodos de Redução Potencial

Os métodos de redução potencial tomam passos da mesma forma que os métodos seguidores de caminho, mas eles não seguem explicitamente C e podem ser motivados independentemente dele. Estes métodos usam uma função de potencial logarítmica para medir o valor de cada ponto em $\mathcal{F}_0$ e visam obter uma determinada redução fixa nesta função em cada iteração. A função potencial primal-dual, que denotamos genericamente por Φ, geralmente tem duas propriedades importantes:

$$\Phi \rightarrow \infty, \text{ se } x_i s_i \rightarrow 0 \text{ para algum } i, \text{ mas } \mu = x^T s/n \not\rightarrow 0, \tag{4.14a}$$

$$\Phi \rightarrow -\infty, \text{ se e somente se } (x, \lambda, s) \text{ satisfaz (P) e (D)} \tag{4.14b}$$

A primeira propriedade (4.14a) impede que o produto $x_i s_i$,para algum i, se aproxime de zero independentemente dos outros, portanto, satisfaz a condição (2.2d). A segunda propriedade (4.14b) relaciona Φ com a solução do problema Primal e o problema Dual. Se o nosso algoritmo força Φ para $-\infty$, então (4.14b) garante que a sequência se aproxima das soluções do Primal e do Dual. A função potencial primal-dual mais interessante é a função Tanabe-Todd-Ye, definida por

$$\Phi_\rho(x, s) = \rho \log x^T s - \sum_{i=1}^{n} \log x_i s_i \tag{4.15}$$

para algum parâmetro $\rho > n$. Os algoritmos de redução de potencial obtêm suas direções de busca resolvendo (4.12) para algum $\sigma \in (0, 1)$. O tamanho do passo α_k é escolhido para minimizar aproximadamente Φ_ρ ao longo da direção calculada.

Alguns algoritmos baseados em Φ_ρ tratam as variáveis primais e duais de diferentes maneiras, usando a função potencial somente para monitorar o progresso e provar a convergência.

Podemos definir um algoritmo simples, que toma passos primais-duais da forma (4.12) e usa Φ_ρ para a escolha do tamanho do passo. Este algoritmo tem a menor complexidade entre todos os algoritmos de pontos interiores como provado por Wright [3], exigindo uma ordem de $O(\sqrt{n}\left|\log \epsilon\right|)$ iterações para reduzir $x^T s$ abaixo de um determinado $\epsilon > 0$.

Reescrevendo (4.15) como

$$\Phi_\rho(x, s) = (\rho - n) \log x^T s + \Phi_n(x, s), \tag{4.16}$$

onde a função $\Phi_n(x, s) \leq n \log n$, conforme Lema 4.2 de [3], assim

$$\Phi_\rho(x, s) = (\rho - n) \log x^T s - \sum_{i=1}^{n} \log \frac{x_i s_i}{x^T s/n} + n \log n, \tag{4.17}$$

podemos ver que a função Φ_ρ toma duas considerações: dualidade e centralidade. A função $\Phi_\rho(x, s)$ age como uma função de barreira quando $(x, \lambda, s) \in \mathcal{F}_0$ se desloca em direção a qualquer ponto tal que $x_i s_i = 0$, mas $x^T s > 0$. Nesse caso, o termo $(\rho - n) \log x^T s$ em (4.16) permanece limitado, enquanto que o termo $\Phi_n(x, s)$ explode fazendo com que $\Phi_\rho(x, s)$ se aproxime de $+\infty$. Por outro lado, o termo $\Phi_n(x, s)$ em (4.16) tem um limite inferior, portanto, $\Phi_\rho(x, s)$ se aproxima de $-\infty$ somente se o termo $(\rho - n) \log x^T s$ de (4.16) se aproxima de $-\infty$, ou seja, $\mu \downarrow 0$. E assim, motivando o desenvolvimento do algoritmo de redução potencial, que gera uma sequência de pontos $(x^k, \lambda^k, s^k) \in \mathcal{F}_0$ tal que $\Phi_\rho(x^k, s^k) \downarrow -\infty$, conduzindo a medida de dualidade μ_k para zero e forçando a sequência para otimalidade.

Algoritmo 4: Algoritmo de Redução Potencial

Entrada: Dado $\rho > n$ e $(x^0, \lambda^0, s^0) \in \mathcal{F}_0$;

1 **para** $k = 1, 2, \ldots$ **faça**

2 Seja $\sigma_k = n/\rho$ e resolva (4.13);

3 Calcule

$$\alpha_{max} = \sup\{\alpha \in [0, 1] \mid (x, s) + \alpha(\Delta x, \Delta s) \geq 0\}$$

$$\alpha_k = \arg \min_{\alpha \in (0, \alpha_{\max})} \Phi_\rho(x^k + \alpha \Delta x^k, s^k + \alpha \Delta s^k)$$

4 Seja $(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + \alpha_k(\Delta x^k, \Delta \lambda^{k+1}, \Delta s^{k+1})$;

5 **fim**

4.6 Pontos Iniciais Infactíveis

Em todos os métodos até agora, assumimos que os pontos iniciais são estritamente factíveis, satisfazem as equações lineares definidas pelas condições (2.2a) e (2.2b), e todas as iterações a partir desse ponto inicial satisfazem essa mesma condição. Porém, para a maioria dos problemas, um ponto inicial factível é difícil de encontrar. Isso pode ser trivial se reformulamos o problema, mas essas reformulações algumas vezes introduzem distorções que podem deixar o problema mais difícil de resolver. Os métodos de pontos interiores infactíveis propõem que os pontos iniciais satisfaçam apenas a condição de positividade $(x, s) > 0$. A direção de busca precisa ser modificada para que se aproxime tanto da factibilidade quanto da centralidade, mas apenas uma pequena mudança é necessária

para a equação do passo definida por (4.12). Então

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{pmatrix} = \begin{pmatrix} -r_c \\ -r_b \\ -XSe + \sigma \mu e \end{pmatrix}, \tag{4.18}$$

onde $r_c = A^T\lambda + s - c$ e $r_b = Ax - b$. A direção de busca ainda é um passo de Newton em direção ao ponto $(x_{\sigma\mu}, \lambda_{\sigma\mu}, s_{\sigma\mu}) \in C$ e que tenta em um único passo acabar com o problema de infactibilidade nas restrições lineares.

Se um passo completo é tomado, r_c e r_b se tornam zero, pois já na primeira iteração as restrições lineares são satisfeitas, assim as próximas iterações até a solução se mantém estritamente factíveis, já que o produto $x_i s_i$ é restrito ao produto $\sigma\mu$, e assim $x, s > 0$.

4.7 Método Preditor-Corretor

O método Preditor-Corretor foi originalmente desenvolvido por Mehrotra [7], utilizando os métodos Seguidores de Caminho e Redução Potencial. A contribuição de Mehrotra foi combinar as ideias desses métodos já existentes e adicionar heurísticas para a escolha do parâmetro de centralidade σ, dos tamanhos dos passos e do ponto inicial. O resultado foi um algoritmo altamente eficiente, que ainda é base para a maioria dos algoritmos de Pontos Interiores.

O algoritmo de Mehrotra gera iterações infactíveis (x_k, λ_k, s_k) que satisfazem $(x_k, s_k) > 0$. O cálculo da direção de busca para cada iteração consiste em três passos:

- Uma direção preditora de afim escala - a direção pura de Newton para a função $F(x, \lambda, s)$ definida por (4.11),
- Um termo de centralidade para o qual o tamanho é decidido pela adaptabilidade do parâmetro de centralidade σ,
- Uma direção corretora que tenta compensar algumas das não linearidades da direção de afim escala.

As duas primeiras componentes, o passo de afim escala e o termo de centralidade,

combinam-se para formar o passo de ponto interior infactível a partir das equações

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{pmatrix} = \begin{pmatrix} -r_c \\ -r_b \\ -XSe + \sigma \mu e \end{pmatrix}. \tag{4.19}$$

No algoritmo de Mehrotra, no entanto, o componente de afim escala é calculado separadamente, e antes do componente de centralidade. Fazendo os cálculos desta forma, obtemos a vantagem de escolher o parâmetro de centralidade de forma adaptativa. Se a direção de afim escala fizer um bom progresso na redução da medida de dualidade μ enquanto permanece dentro da região positiva definida por $(x, s) > 0$, concluímos que pouca centralidade é necessária nesta iteração, por isso atribuímos um pequeno valor a σ. Se, por outro lado, pudermos mover apenas uma curta distância ao longo da direção de afim escala antes de violar as restrições $(x, s) > 0$, concluímos que é necessária uma quantidade significativa de centralidade, então escolhemos σ mais perto de 1.

A direção de afim escala é obtida da aproximação linear para as condições em (2.2). Uma vez que essa direção é calculada separadamente, ela pode ser usada para avaliar o erro da aproximação linear. O conhecimento deste erro nos permite calcular a componente de correção, a terceira componente da direção de busca de Mehrotra. Uma vez que os componentes de centralidade e de correção são obtidos resolvendo sistemas lineares, com a mesma matriz de coeficientes que a direção de afim escala e, uma vez que são independentes uns dos outros, não há necessidade de os calcular separadamente. Podemos simplesmente combiná-los em uma única direção, adicionando suas componentes do lado direito e calcular a direção combinada.

O algoritmo de Mehrotra oferece uma escolha adaptativa de σ e um aprimoramento de ordem superior do passo puro de Newton. Em geral, o custo extra por iteração, é facilmente justificado por uma redução significativa no número de iterações. Ou seja, mesmo resolvendo dois sistemas lineares, o método é mais rápido na resolução do problema (P), pois precisa de menos iterações para chegar a solução.

4.7.1 Algoritmo Preditor-Corretor

Dado um ponto (x, λ, s) com $(x, s) > 0$, a direção de afim escala é encontrada resolvendo o sistema

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x^{af} \\ \Delta \lambda^{af} \\ \Delta s^{af} \end{pmatrix} = \begin{pmatrix} -r_c \\ -r_b \\ -XSe \end{pmatrix}. \tag{4.20}$$

Este sistema é obtido colocando $\sigma = 0$ na equação (4.19). Agora, encontramos os tamanhos dos passos até a fronteira ao longo dessa direção, realizando cálculos separados para as componentes primais e duais como

$$\alpha_{af}^{P} = \arg\max\{\alpha \in [0,1] \,|\, x + \alpha\Delta x^{af} \geq 0\},$$

$$\alpha_{af}^{D} = \arg\max\{\alpha \in [0,1] \,|\, s + \alpha\Delta s^{af} \geq 0\}.$$

Para medir a eficácia da direção de afim escala, definimos μ_{af} como um valor hipotético de μ resultante do passo completo até a fronteira, ou seja,

$$\mu_{af} = \left((x + \alpha_{af}^{P}\Delta x^{af})^{T}(s + \alpha_{af}^{D}\Delta s^{af})\right)/n.$$

Se $\mu_{af} \ll \mu$, a direção de afim escala é uma boa direção e que permite progresso significante na redução de μ, então escolhemos o parâmetro de centralidade σ perto de zero. Se μ_{af} é apenas um pouco menor que μ, escolhemos σ perto de 1. Esta escolha tem o efeito de nos mover mais perto da trajetória central C, assim, o algoritmo fica numa posição melhor para conseguir uma diminuição em μ na próxima iteração. Uma heurística proposta por Mehrotra [7] no parâmetro de centralidade σ, após testes computacionais, é usar

$$\sigma = \left(\frac{\mu_{af}}{\mu}\right)^{3}. \tag{4.21}$$

Como pode-se notar, quando $\mu_{af} \ll \mu$, σ fica mais perto de 0, e quando μ_{af} é um pouco menor que μ, então σ fica próximo de 1.

De (4.20), podemos ver como o produto emparelhado $x_i s_i$ é afetado pelo passo completo na direção de afim escala,

$$(x_i + \Delta x_i^{af})(s_i + \Delta s_i^{af}) = x_i s_i + x_i\Delta s_i^{af} + s_i\Delta x_i^{af} + \Delta x_i^{af}\Delta s_i^{af},$$

de (4.20), temos

$$\sum_{i=1}^{n}(x_i s_i + x_i\Delta s_i^{af} + s_i\Delta x_i^{af}) = XSe + X\Delta S^{af} + S\Delta X^{af} = 0,$$

então

$$(x_i + \Delta x_i^{af})(s_i + \Delta s_i^{af}) = \Delta x_i^{af}\Delta s_i^{af}. \tag{4.22}$$

Quando um passo completo é tomado, o produto $x_i s_i$ transforma-se em $\Delta x_i^{af}\Delta s_i^{af}$, ao invés do valor zero predito por (4.20). O componente corretor $(\Delta x^{cor}, \Delta\lambda^{cor}, \Delta s^{cor})$ tenta

compensar o desvio da linearidade, modificando a direção de busca para que produto $x_i s_i$ se aproxime de zero. Este passo satisfaz o sistema linear

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x^{cor} \\ \Delta \lambda^{cor} \\ \Delta s^{cor} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -\Delta X^{af} \Delta S^{af} e \end{pmatrix}, \tag{4.23}$$

onde $\Delta X^{af} = \text{diag}(\Delta x_1^{af}, \Delta x_2^{af}, \ldots, \Delta x_n^{af})$ e $\Delta S^{af} = \text{diag}(\Delta s_1^{af}, \Delta s_2^{af}, \ldots, \Delta s_n^{af})$.

De (4.22) e (4.23) obtemos

$$\begin{aligned}(x_i + \Delta x_i^{af} + \Delta x_i^{cor})(s_i + \Delta s_i^{af} + \Delta s_i^{cor}) &= x_i s_i + x_i \Delta s_i^{af} + x_i \Delta s_i^{cor} + s_i \Delta x_i^{af} + \Delta x_i^{af} \Delta s_i^{af} \\ &+ \Delta x_i^{af} \Delta x_i^{cor} + s_i \Delta x_i^{cor} + \Delta x_i^{cor} \Delta s_i^{af} + \Delta x_i^{cor} \Delta s_i^{cor}.\end{aligned}$$

E da última linha em (4.20) e (4.23), temos

$$S \Delta x^{af} + X \Delta s^{af} + XS e = 0,$$

$$S \Delta x^{cor} + X \Delta s^{cor} + \Delta X^{af} \Delta S^{af} e = 0.$$

Assim, temos

$$(x_i + \Delta x_i^{af} + \Delta x_i^{cor})(s_i + \Delta s_i^{af} + \Delta s_i^{cor}) = \Delta x_i^{af} \Delta x_i^{cor} + \Delta x_i^{cor} \Delta s_i^{af} + \Delta x_i^{cor} \Delta s_i^{cor}. \tag{4.24}$$

Quando a matriz em (4.20) e (4.23) se aproximam de um limite não singular, o produto $x_i s_i$ de (4.24) se aproxima mais de zero do que o produto obtido por (4.22). Obtemos uma redução maior com a escolha em (4.24). Porém, isso quando a matriz de coeficientes é não singular. Se a matriz é singular, a escolha do passo pelo sistema (4.23) pode não ser tão bom quanto a escolha do passo em (4.20), já que precisamos resolver dois sistemas lineares para obter o passo em (4.23), e como a matriz é singular esse passo pode ser até pior. Nesse caso, o uso do componente corretor melhora a eficiência do método.

O passo combinado da direção de centralidade e de correção $(\Delta x^{cc}, \Delta \lambda^{cc}, \Delta s^{cc})$ é obtido resolvendo o seguinte sistema linear:

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x^{cc} \\ \Delta \lambda^{cc} \\ \Delta s^{cc} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \sigma \mu e - \Delta X^{af} \Delta S^{af} e \end{pmatrix}, \tag{4.25}$$

os fatores da matriz de coeficientes já estão disponíveis na solução de (4.20), então (4.25) pode ser resolvida ao custo de realizar uma única substituição.

Com os elementos principais do algoritmo de Mehrotra descritos acima, podemos definir o algoritmo para resolver o problema (P) [3]:

Algoritmo 5: Algoritmo de Mehrotra Preditor-Corretor

Entrada: (x^0, λ^0, s^0) com $(x^0, s^0) > 0, \gamma \in [0.9, 1)$

para $k = 1, 2, \ldots$ **faça**

Defina $(x, \lambda, s) = (x^k, \lambda^k, s^k)$ e resolva (4.20) para $(\Delta x^{af}, \Delta\lambda^{af}, \Delta s^{af})$;

Calcule

$$\alpha_{af}^P = \arg\max\{\alpha \in [0,1] \,|\, x + \alpha\Delta x^{af} \geq 0\}$$

$$\alpha_{af}^D = \arg\max\{\alpha \in [0,1] \,|\, s + \alpha\Delta s^{af} \geq 0\}$$

$$\mu_{af} = (x + \alpha_{af}^P \Delta x^{af})^T (s + \alpha_{af}^D \Delta s^{af})/n$$

Defina o parâmetro de centralidade $\sigma = \left(\frac{\mu_{af}}{\mu}\right)^3$;

Resolva (4.25) para $(\Delta x^{cc}, \Delta\lambda^{cc}, \Delta s^{cc})$;

Calcule a direção de busca e o passo de

$$(\Delta x^k, \Delta\lambda^k, \Delta s^k) = (\Delta x^{af}, \Delta\lambda^{af}, \Delta s^{af}) + (\Delta x^{cc}, \Delta\lambda^{cc}, \Delta s^{cc})$$

$$\alpha_{max}^P = \arg\max\{\alpha \geq 0 \,|\, x^k + \alpha\Delta x^k \geq 0\}$$

$$\alpha_{max}^D = \arg\max\{\alpha \geq 0 \,|\, s^k + \alpha\Delta s^k \geq 0\}$$

Defina $\alpha_k^P = \min(\gamma\alpha_{max}^P, 1)$ e $\alpha_k^D = \min(\gamma\alpha_{max}^D, 1)$;

Defina

$$x^{k+1} = x^k + \alpha_k^P \Delta x^k$$

$$(\lambda^{k+1}, s^{k+1}) = (\lambda^k, s^k) + \alpha_k^D (\Delta\lambda^k, \Delta s^k)$$

fim

O sistema (4.20) é grande e esparso e precisa ser resolvido com mais cuidado. Uma maneira é reduzi-lo para o chamado Sistema Aumentado (4.26), é menos eficiente, mas é mais estável, pois o tamanho da matriz diminui. Como podemos ver, antes tínhamos o sistema

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{pmatrix} = \begin{pmatrix} -r_c \\ -r_b \\ -r_{xs} \end{pmatrix},$$

onde $r_c = A^T\lambda + s - c$, $r_b = Ax - b$ e $r_{xs} = XSe - \sigma\mu e$. Eliminado Δs e usando $D = S^{-1/2}X^{1/2}$, obtemos o seguinte sistema reduzido

$$\begin{pmatrix} 0 & A \\ A^T & -D^{-2} \end{pmatrix} \begin{pmatrix} \Delta\lambda \\ \Delta x \end{pmatrix} = \begin{pmatrix} -r_b \\ -r_c + X^{-1} r_{xs} \end{pmatrix}, \tag{4.26a}$$

$$\Delta s = -X^{-1}(r_{xs} + S\,\Delta X). \tag{4.26b}$$

Apesar da matriz de coeficientes em (4.26a) ser indefinida, ela é simétrica e possui vantagens, como estabilidade e flexibilidade para encontrar a direção de afim escala e também para encontrar a direção de correção e centralização, já que apenas fazemos substituições do lado direito do sistema.

Outra maneira é utilizar o Sistema Normal, definida da seguinte maneira

$$AD^2A^T = -r_b + A(-S^{-1}Xr_c + S^{-1}r_{xs}), \tag{4.27}$$

$$\Delta s = -r_c - A^T\Delta\lambda, \tag{4.28}$$

$$\Delta x = -S^{-1}(r_{xs}X\Delta s). \tag{4.29}$$

Usado na maioria dos códigos primais-duais, pois a matriz AD^2A^T em (4.27) é simétrica e semi-definida positiva.

4.7.2 Teoria de Convergência

Para a teoria de convergência, utilizaremos como base o algoritmo Seguidor de Caminho de Passo Longo, pois o algoritmo de Mehrotra Preditor-Corretor é baseado no método Seguidor de Caminho. Algumas diferenças entre eles são: para o algoritmo de Mehrotra é calculada a direção de afim escala e depois a direção de correção, ou seja, são resolvidos dois sistemas lineares, e no algoritmo Seguidor de Caminho é realizada somente o cálculo de um sistema linear. O parâmetro de centralidade de Mehrotra é adaptativa, dependendo de como o algoritmo se comporta para o problema e no Seguidor de Caminho, esse parâmetro é escolhido dentro de um intervalo fixo.

Nosso objetivo na análise é que dada alguma tolerância $\epsilon > 0$, o algoritmo exige uma ordem de $O(n\left|\log\epsilon\right|)$ iterações para reduzir a medida de dualidade μ por um fator ϵ, para encontrar um ponto (x^k, λ^k, s^k) onde $\mu_k \leq \epsilon\mu_0$. A estimativa $O(n\left|\log\epsilon\right|)$ é quantidade de iterações para o pior caso, assim podemos dizer em relação a quantidade de iterações que o algoritmo possui uma ordem de $O(n\left|\log\epsilon\right|)$.

Começaremos provando alguns resultados [10], que nos ajudarão a entender as análises de convergência.

Lema 4. *Seja u e v quaisquer vetores em* $\mathbb{R}^n$ *com* $u^Tv \geq 0$. *Então*

$$\|UVe\| \leq 2^{-3/2}\|u+v\|_2^2$$

onde

$$U = diag(u_1, u_2, \ldots, u_n), \quad V = diag(v_1, v_2, \ldots, v_n).$$

Demonstração. Notemos que para quaisquer dois escalares α e β com $\alpha\beta \geq 0$, temos que

$$\sqrt{|\alpha\beta|} \leq \frac{1}{2}|\alpha + \beta|. \tag{4.30}$$

Uma vez que $u^T v \geq 0$, temos

$$0 \leq u^T v = \sum_{u_i v_i \geq 0} u_i v_i + \sum_{u_i v_i < 0} u_i v_i = \sum_{i \in \mathcal{P}} |u_i v_i| - \sum_{i \in \mathcal{M}} |u_i v_i| \tag{4.31}$$

onde dividimos o conjunto de índices $\{1, 2, \ldots, n\}$ como

$$\mathcal{P} = \{i \,|\, u_i v_i \geq 0\}, \quad \mathcal{M} = \{i \,|\, u_i v_i < 0\},$$

ou seja, $\mathcal{P}$ é o conjunto de índices onde o produto $u_i v_i$ é não negativo e $\mathcal{M}$ é o conjunto onde $u_i v_i$ é negativo. Sendo assim, temos

$$\begin{aligned}
\|UVe\| &= (\|[u_i v_i]_{i\in\mathcal{P}}\|^2 + \|[u_i v_i]_{i\in\mathcal{M}}\|^2) \\
&\leq (\|[u_i v_i]_{i\in\mathcal{P}}\|_1^2 + \|[u_i v_i]_{i\in\mathcal{M}}\|_1^2) && \text{sendo } \|.\|_2 \leq \|.\|_1 \\
&\leq (2\,\|[u_i v_i]_{i\in\mathcal{P}}\|_1^2)^{1/2} && \text{de (4.31)} \\
&\leq \sqrt{2}\left\|[\frac{1}{4}(u_i + v_i)^2]_{i\in\mathcal{P}}\right\|_1 && \text{de (4.30)} \\
&= 2^{-3/2}\sum_{i\in\mathcal{P}}(u_i + v_i)^2 \\
&\leq 2^{-3/2}\sum_{i=1}^{n}(u_i + v_i)^2 \\
&\leq 2^{-3/2}\,\|u_i + v_i\|^2 .
\end{aligned}$$

□

Lema 5. *Se* $(x, \lambda, s) \in \mathcal{N}_{-\infty}(\gamma)$*, então*

$$\|\Delta X \Delta S\, e\| \leq 2^{-3/2}\left(1 + \frac{1}{\gamma}\right) n\mu$$

onde $\Delta X = diag(\delta x_1, \delta x_2, \ldots, \delta x_n), \Delta S = diag(\delta s_1, \delta s_2, \ldots, \delta s_n)$.

Demonstração. Do sistema linear

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix}\begin{pmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -XSe + \sigma\mu e \end{pmatrix}, \tag{4.32}$$

temos que $r_b = r_c = 0$, nesse caso,

$$A\Delta x = 0, \quad \text{e} \quad A^T\Delta\lambda + \Delta s = 0.$$

Então, temos que

$$\Delta x^T \Delta s = -\Delta x^T A^T \Delta\lambda = -(A\Delta x)^T \Delta\lambda = 0.$$

Multiplicando a última linha em (4.32) por $(XS)^{-1/2}$ e usando $D = X^{1/2}S^{-1/2}$, obtemos o seguinte

$$\begin{aligned} S\Delta x + X\Delta s &= -XSe + \sigma\mu e \\ (XS)^{-1/2}(S\Delta x + X\Delta s) &= (XS)^{-1/2}(-XSe + \sigma\mu e) \\ D^{-1}\Delta x + D\Delta s &= (XS)^{-1/2}(-XSe + \sigma\mu e). \end{aligned} \tag{4.33}$$

E além disso, temos o seguinte

$$(D^{-1}\Delta x)^T (D\Delta s) = \Delta x^T \Delta s = 0. \tag{4.34}$$

Aplicando o Lema (4), com $u = D^{-1}\Delta x$ e $v = D\Delta s$, temos

$$\begin{aligned} \|\Delta X\Delta S e\| &= (\left\|(D^{-1}\Delta x)^T D\Delta s e\right\|) \\ &\leq 2^{-3/2}\left\|D^{-1}\Delta x + D\Delta s\right\|^2 && \text{Lema 4} \\ &= 2^{-3/2}\left\|(XS)^{-1/2}(-XSe + \sigma\mu e)\right\|^2 && \text{de (4.33).} \end{aligned}$$

Expandindo a norma Euclidiana e usando algumas relações, tais como $x^T s = n\mu$ e $e^T e = n$, e utilizando a definição de vizinhança $\mathcal{N}_{-\infty}$ onde $x_i s_i \geq \gamma\mu$ obtemos

$$\begin{aligned} \|\Delta X\Delta S e\| &= 2^{-3/2}\left[x^T s - 2\sigma\mu e^T e + \sigma^2\mu^2 \sum_{i=1}^{n} \frac{1}{x_i s_i}\right] \\ &\leq 2^{-3/2}\left[x^T s - 2\sigma\mu e^T e + \sigma^2\mu^2 \frac{n}{\gamma\mu}\right] \\ &\leq 2^{-3/2}\left[1 - 2\sigma + \frac{\sigma^2}{\gamma}\right] n\mu \\ &\leq 2^{-3/2}(1 + \frac{1}{\gamma})n\mu, \end{aligned}$$

como queríamos provar. □

O resultado acima nos ajudará a mostrar o que queremos, que o algoritmo Seguidor de Caminho de Passo Longo, possui a ordem de $O(n\left|\log\epsilon\right|)$ iterações para o seu pior caso, que exige mais iterações, então o algoritmo vai fazer no máximo $n\left|\log\epsilon\right|$ iterações para qualquer problema.

Teorema 5. *Dados os parâmetros $\gamma, \sigma_{min}, \sigma_{max}$ no Algoritmo 3, existe uma constante δ independente de n tal que*

$$\mu_{k+1} \leq \left(1 - \frac{\delta}{n}\right)\mu_k, \tag{4.35}$$

para todo $k \geq 0$.

Demonstração. Para qualquer $i = 1, 2, \ldots, n$, temos do Lema 5 que

$$\left|\Delta x_i^k \Delta s_i^k\right| \leq \left\|\Delta X^k \Delta S^k e\right\| \leq 2^{-3/2}\left(1 + \frac{1}{\gamma}\right)n\mu. \tag{4.36}$$

De (4.19), temos pela linha 3 que

$$S\Delta x + X\Delta s = -XSe + \sigma\mu e. \tag{4.37}$$

Pela definição da vizinhança $\mathcal{N}_{-\infty}(\gamma)$, temos que $x_i s_i \geq \gamma\mu$. Assim,

$$\begin{aligned} x_i^k(\alpha)s_i^k(\alpha) &= (x_i^k + \alpha\Delta x_i^k)(s_i^k + \alpha\Delta s_i^k) \\ &= x_i^k s_i^k + \alpha(x_i^k\Delta s_i^k + s_i^k\Delta x_i^k) + \alpha^2\Delta x_i^k\Delta x_i^k \\ &\geq x_i^k s_i^k(1-\alpha) + \alpha\sigma_k\mu_k - \alpha^2\left|\Delta x_i^k\Delta s_i^k\right| \qquad \text{de (4.37)} \\ &\geq \gamma(1-\alpha)\mu_k + \alpha\sigma_k\mu_k - \alpha^2 2^{-3/2}\left(1 + \frac{1}{\gamma}\right)n\mu. \end{aligned} \tag{4.38}$$

Somando as n componentes de (4.37) e usando que $\mu = x^T s/n$, temos

$$s^T\Delta x + x^T\Delta s = -x^T s + \sigma\mu n = -x^T s + \sigma x^T s = -(1-\sigma)x^T s.$$

Da fórmula anterior e usando que $\Delta x^T\Delta s = 0$ de (4.32), temos

$$x(\alpha)^T s(\alpha) = x^T s + \alpha(s^T\Delta x + x^T\Delta s) + \alpha^2\Delta x^T\Delta s = x^T s(1 - \alpha(1-\sigma))$$

onde $(x^k(\alpha), \lambda^k(\alpha), s^k(\alpha)) = (x^k, \lambda^k, s^k) + \alpha(\Delta x^k, \Delta\lambda^k, \Delta s^k)$. Assim, dividindo a equação acima por n, temos

$$\mu_k(\alpha) = (1 - \alpha(1-\sigma_k))\mu_k. \tag{4.39}$$

De (4.38) e (4.39), podemos ver que a condição de proximidade

$$x_i^k(\alpha)s_i^k(\alpha) \geq \gamma\mu_k(\alpha)$$

é satisfeita, desde que

$$\gamma(1-\alpha)\mu_k + \alpha\sigma_k\mu_k - \alpha^2 2^{-3/2}\left(1 + \frac{1}{\gamma}\right)n\mu \geq (1 - \alpha(1-\sigma_k))\mu_k.$$

Rearranjando, temos que

$$\alpha\sigma_k\mu_k(1-\gamma) \geq \alpha^2 2^{-3/2} n\mu_k(1+1/\gamma),$$

somente se

$$\alpha \leq 2^{3/2}\frac{\sigma_k}{n}\gamma\frac{1-\gamma}{1+\gamma}.$$

Provamos que $(x^k(\alpha), \lambda^k(\alpha), s^k(\alpha))$ satisfaz a condição de proximidade para a vizinhança $\mathcal{N}_{-\infty}(\gamma)$ quando α está no intervalo dado por

$$(x^k(\alpha), \lambda^k(\alpha), s^k(\alpha)) \in \mathcal{N}_{-\infty}(\gamma) \text{ para todo } \alpha \in \left[0, 2^{3/2}\gamma\frac{1-\gamma}{1+\gamma}\frac{\sigma_k}{n}\right]. \tag{4.40}$$

Então, o tamanho do passo α é pelo menos tão grande quanto o limite desse intervalo, ou seja,

$$\alpha_k \geq 2^{3/2}\frac{\sigma_k}{n}\gamma\frac{1-\gamma}{1+\gamma}. \tag{4.41}$$

Assim, provamos (4.40) e (4.41).

Estimando a redução em μ no k-ésimo passo. De (4.40), (4.41) e (4.37), temos

$$\begin{aligned}
\mu_{k+1} &= x^k(\alpha_k)^T s^k(\alpha)/n \\
&= \left[(x^k)^T s^k + \alpha_k((x^k)^T\Delta s^k + (s^k)^T\Delta x^k) + \alpha_k^2(\Delta x^k)^T\Delta s^k\right]/n \\
&= \mu_k + \alpha(-(x^k)^T s^k/n + \sigma_k\mu_k) \\
&= (1-\alpha_k(1-\sigma_k))\mu_k \\
&\leq \left(1 - 2^{3/2}\frac{\gamma}{n}\frac{1-\gamma}{1+\gamma}\sigma_k(1-\sigma_k)\right)\mu_k.
\end{aligned} \tag{4.42}$$

Agora, temos $\sigma(1-\sigma)$ como uma função côncava quadrática. Assim, em qualquer intervalo dado ela atinge seu valor mínimo em um dos pontos de extremidade. Desta forma, temos

$$\sigma_k(1-\sigma_k) \geq \min\{\sigma_{\min}(1-\sigma_{\min}), \sigma_{\max}(1-\sigma_{\max})\}, \text{ para todo } \sigma_k \in [\sigma_{\min}, \sigma_{\max}].$$

Para completarmos a prova, apenas substituímos essa estimativa dentro de (4.42), assim

$$\delta = 2^{3/2}\gamma\frac{1-\gamma}{1+\gamma}\min\{\sigma_{\min}(1-\sigma_{\min}), \sigma_{\max}(1-\sigma_{\max})\}.$$

E temos

$$\mu_{k+1} \leq \left(1 - \frac{\delta}{n}\right)\mu_k.$$

□

Para completar a nossa análise da convergência do Algoritmo 3, concluímos que a redução do fator ϵ na medida de dualidade μ pode ser obtida em $O(n \log 1/\epsilon)$ iterações.

Teorema 6. *Dado $\epsilon \in (0,1)$ e $\gamma \in (0,1)$, suponha que o ponto inicial no Algoritmo 3 satisfaz $(x^0, \lambda^0, s^0) \in \mathcal{N}_{-\infty}(\gamma)$. Então existe um índice k com $K = O(n \log 1/\epsilon)$ tal que*

$$\mu_k \leq \epsilon \mu_0, \qquad \text{para todo } k \geq K.$$

Demonstração. Tomando logaritmo de ambos os lados em (4.35), temos

$$\log \mu_{k+1} \leq \log\left(\left(1 - \frac{\delta}{n}\right)\mu_k\right) = \log\left(1 - \frac{\delta}{n}\right) + \log \mu_k.$$

Por recursividade, temos

$$\begin{aligned} \log \mu_k &\leq \log\left(1 - \frac{\delta}{n}\right) + \log \mu_{k-1} \\ &\leq \log\left(1 - \frac{\delta}{n}\right) + \left(\log\left(1 - \frac{\delta}{n}\right) + \log \mu_{k-2}\right) \\ &= 2\log\left(1 - \frac{\delta}{n}\right) + \log \mu_{k-2}. \end{aligned}$$

Assim, aplicando isso k vezes, temos

$$\log \mu_k \leq k \log\left(1 - \frac{\delta}{n}\right) + \log \mu_0.$$

Subtraindo $\log \mu_0$ em ambos os lados, temos

$$\log \mu_k - \log \mu_0 \leq k \log\left(1 - \frac{\delta}{n}\right).$$

E então

$$\log\left(\frac{\mu_k}{\mu_0}\right) \leq k \log\left(1 - \frac{\delta}{n}\right).$$

A regra

$$\log(1 - \beta) \leq \beta, \qquad \text{para todo } \beta > -1,$$

onde a igualdade só acontece quando $\beta = 0$, implica que

$$\log\left(\frac{\mu_k}{\mu_0}\right) \leq k\left(\frac{-\delta}{n}\right).$$

Portanto, a condição $\mu_k/\mu_0 \leq \epsilon$ é satisfeita se

$$k\left(-\frac{\delta}{n}\right) \leq \log \epsilon.$$

E então

$$k\left(\frac{\delta}{n}\right) \geq \log\left(\frac{1}{\epsilon}\right).$$

Esta desigualdade é garantida para todo k que satisfaz

$$k \geq K = \frac{n}{\delta} \log \frac{1}{\epsilon} = \frac{n}{\delta} \left| \log \epsilon \right|$$

Assim, a prova está completa. □

5 Testes Computacionais

O algoritmo Mehrotra Preditor-Corretor foi implementado na linguagem Julia e o algoritmo Clp [2] em C, que usa o método Simplex, um método mais estabelecido e robusto e que não esperamos vencê-lo, devido a maneira simples que o algoritmo de Mehrotra foi implementado. Queremos apenas comparar os valores da função objetivo obtidos na tentativa de resolver os problemas do netlib [1], problemas de minimização, que serão utilizados para a realização dos testes computacionais.

O parâmetro de parada para a aproximação da solução de ambos os algoritmos foi definido como sendo $\epsilon = 10^{-7}$. Para o algoritmo Mehrotra Preditor-Corretor foram definidos o máximo de 1000 iterações, e um tempo máximo de 30 segundos, assim como para o algoritmo Clp.

Para o algoritmo de Mehrotra foram definidos os seguintes critérios de parada:

Saída	Significado
0	Aproximação da solução encontrada
1	Atingiu máximo de iterações
2	Atingiu o tempo máximo
3	O algoritmo falhou

Tabela 1 – Critérios de parada Mehrotra

Para o algoritmo Clp:

Saída	Significado
0	Aproximação da solução encontrada
1	Atingiu máximo de iterações

Tabela 2 – Critérios de parada Clp

Na Tabela 3 podemos ver os testes realizados para 37 problemas do repositório do netlib [1] os quais estão no padrão do algoritmo de Mehrotra.

Na Tabela 4, podemos ver os resultados para os mesmos problemas acima, porém, para o algoritmo Clp.

Na Tabela 5, podemos ver os resultados para o algoritmo Clp sem o presolve. Dentro do presolve estão alguns métodos de remover singularidades, por exemplo, linhas da

matriz de restrições do problema original que são combinações lineares de outras linhas, transformando-o em um problema mais fácil de resolver do que o problema original, e como veremos faz diferença na resolução dos problemas. Nos casos em que o algoritmo Clp convergiu para a solução em 0 iterações, aconteceu pois durante o presolve, o algoritmo já encontrou a solução. Note que há alguns problemas em que o algoritmo Clp sem o presolve não convergiu e com presolve convergiu.

Comparando os resultados dos algoritmos, podemos ver que o algoritmo de Mehrotra Preditor-Corretor para os problemas em que convergiu chega na solução, em alguns problemas, em menos iterações do que o algoritmo Clp, como podemos ver para os problemas STOCFOR1 e STOCFOR2, por exemplo. Isso acontece, pois o algoritmo de Mehrotra encontra a solução por dentro da região factível, enquanto que o Clp caminha por vários vértices antes de encontrar a solução.

Em relação ao tempo, o algoritmo Mehrotra Preditor-Corretor sai perdendo. Pois como vimos anteriormente, o algoritmo resolve dois sistemas lineares para cada iteração, e dependendo do tamanho do problema, esses sistemas podem ser muito grandes, fazendo com que o algoritmo demore para chegar a solução do problema, mesmo que em poucas iterações. Há também o fato da implementação ter sido simples, sem robustez em relação ao cálculo desses sistemas lineares. Uma maneira de evitar que isso aconteça é utilizar somente as componentes não zero da matriz de coeficientes em (4.26).

Problema	f(x)	Restrições	Variáveis	Saída	Iterações	Tempo (seg)
AFIRO	-4.64753e+02	27	32	0	79	0.0394
AGG2	NaN	516	302	3	49	1.3714
AGG3	NaN	516	302	3	47	1.3546
AGG	NaN	488	163	3	62	1.1336
BANDM	4.09883e+02	305	472	1	1000	15.1754
BLEND	-3.08121e+01	74	83	0	19	0.0541
DEGEN2	-1.43518e+03	444	534	0	13	0.5143
DEGEN3	-9.87294e+02	1503	1818	2	81	30.2870
E226	-1.87519e+01	223	282	0	28	0.4421
FARM	1.75000e+04	7	12	0	24	0.0067
ISRAEL	-8.96645e+05	174	142	0	795	9.4469
KLEEMIN3	-1.00000e+04	3	3	0	19	0.0033
KLEEMIN4	-1.00000e+06	4	4	0	30	0.0066
KLEEMIN5	-1.00000e+08	5	5	0	59	0.0123
KLEEMIN6	-1.00000e+10	6	6	0	82	0.0234
KLEEMIN7	-1.00000e+12	7	7	0	117	0.0297
KLEEMIN8	NaN	8	8	3	43	0.0153
LOTFI	NaN	153	308	3	535	4.6556
SC105	-5.22021e+01	105	103	0	58	0.1218
SC205	-5.22021e+01	205	203	0	55	0.1865
SC50A	-6.45751e+01	50	48	0	26	0.0174
SC50B	-7.00000e+01	50	48	0	22	0.0162
SCAGR25	NaN	471	500	3	758	8.4081
SCAGR7	1.09842e+63	129	140	1	1000	1.9228
SCFXM1	3.55989e+32	330	457	1	1000	18.0226
SCFXM2	NaN	660	914	3	300	16.2714
SCFXM3	8.45375e+28	990	1371	2	288	30.0473
SCSD1	8.66667e+00	77	760	0	11	0.1548
SCSD6	5.05000e+01	147	1350	0	13	0.5962
SCSD8	9.05000e+02	397	2750	0	12	1.8279
SCTAP1	1.41225e+03	300	480	0	19	0.4224
SCTAP2	1.72481e+03	1090	1880	0	15	2.3029
SCTAP3	1.42400e+03	1480	2480	0	17	4.4216
SHARE1B	NaN	117	225	3	199	1.1921
STOCFOR1	-4.11320e+04	117	111	0	47	0.1233
STOCFOR2	-3.90244e+04	2157	2031	0	61	16.5382
WOOD1P	1.44290e+00	244	2594	0	18	4.4073

Tabela 3 – Mehrotra Preditor-Corretor

Problema	f(x)	Restrições	Variáveis	Saída	Iterações	Tempo (seg)
AFIRO	-4.64753e+02	27	32	0	5	0.0492
AGG2	-2.02393e+07	516	302	0	165	0.0063
AGG3	1.03121e+07	516	302	0	188	0.0070
AGG	-3.59918e+07	488	163	0	90	0.0038
BANDM	-1.58628e+02	305	472	0	264	0.0117
BLEND	-3.08121e+01	74	83	0	83	0.0037
DEGEN2	-1.43518e+03	444	534	0	447	0.0266
DEGEN3	-1.09233e+03	1503	1818	1	1000	0.0863
E226	-1.16389e+01	223	282	0	300	0.0149
FARM	1.75000e+04	7	12	0	0	0.0029
ISRAEL	-8.96645e+05	174	142	0	109	0.0069
KLEEMIN3	-1.00000e+04	3	3	0	0	0.0007
KLEEMIN4	-1.00000e+06	4	4	0	0	0.0006
KLEEMIN5	-1.00000e+08	5	5	0	0	0.0005
KLEEMIN6	-1.00000e+10	6	6	0	0	0.0006
KLEEMIN7	-1.00000e+12	7	7	0	0	0.0006
KLEEMIN8	-1.00000e+14	8	8	0	0	0.0006
LOTFI	-2.52647e+01	153	308	0	135	0.0031
SC105	-5.22021e+01	105	103	0	39	0.0013
SC205	-5.22021e+01	205	203	0	88	0.0022
SC50A	-6.45751e+01	50	48	0	15	0.0009
SC50B	-7.00000e+01	50	48	0	14	0.0008
SCAGR25	-1.47534e+07	471	500	0	260	0.0063
SCAGR7	-2.33139e+06	129	140	0	76	0.0022
SCFXM1	1.84168e+04	330	457	0	330	0.0099
SCFXM2	3.66603e+04	660	914	0	733	0.0237
SCFXM3	5.45859e+04	990	1371	1	1000	0.0380
SCSD1	8.66667e+00	77	760	0	102	0.0053
SCSD6	5.05000e+01	147	1350	0	247	0.0093
SCSD8	9.05000e+02	397	2750	0	848	0.0516
SCTAP1	1.41225e+03	300	480	0	191	0.0062
SCTAP2	1.72481e+03	1090	1880	0	312	0.0142
SCTAP3	1.42400e+03	1480	2480	0	436	0.0176
SHARE1B	-7.65893e+04	117	225	0	184	0.0039
STOCFOR1	-4.11320e+04	117	111	0	65	0.0020
STOCFOR2	-3.90244e+04	2157	2031	0	934	0.0384
WOOD1P	1.44290e+00	244	2594	0	143	0.0605

Tabela 4 – Algoritmo Clp com presolve

Problema	f(x)	Restrições	Variáveis	Saída	Iterações	Tempo (seg)
AFIRO	-4.64753e+02	27	32	0	22	0.0008
AGG2	-2.02393e+07	516	302	0	245	0.0076
AGG3	1.03121e+07	516	302	0	303	0.0094
AGG	-3.59918e+07	488	163	0	189	0.0054
BANDM	-1.58628e+02	305	472	0	560	0.0219
BLEND	-3.08121e+01	74	83	0	86	0.0018
DEGEN2	-1.43518e+03	444	534	0	591	0.0261
DEGEN3	-1.25201e+03	1503	1818	1	1000	0.0938
E226	-1.16389e+01	223	282	0	319	0.0091
FARM	1.75000e+04	7	12	0	1	0.0007
ISRAEL	-8.96645e+05	174	142	0	150	0.0041
KLEEMIN3	-1.00000e+04	3	3	0	1	0.0005
KLEEMIN4	-1.00000e+06	4	4	0	1	0.0004
KLEEMIN5	-1.00000e+08	5	5	0	1	0.0004
KLEEMIN6	-1.00000e+10	6	6	0	1	0.0005
KLEEMIN7	-1.00000e+12	7	7	0	1	0.0005
KLEEMIN8	-1.00000e+14	8	8	0	1	0.0005
LOTFI	-2.52647e+01	153	308	0	284	0.0061
SC105	-5.22021e+01	105	103	0	103	0.0018
SC205	-5.22021e+01	205	203	0	213	0.0049
SC50A	-6.45751e+01	50	48	0	48	0.0011
SC50B	-7.00000e+01	50	48	0	48	0.0009
SCAGR25	-1.47534e+07	471	500	0	560	0.0163
SCAGR7	-2.33139e+06	129	140	0	171	0.0038
SCFXM1	1.84168e+04	330	457	0	405	0.0159
SCFXM2	3.66603e+04	660	914	0	821	0.0276
SCFXM3	4.72663e+04	990	1371	1	1000	0.0300
SCSD1	8.66667e+00	77	760	0	102	0.0042
SCSD6	5.05000e+01	147	1350	0	247	0.0082
SCSD8	9.05000e+02	397	2750	0	848	0.0420
SCTAP1	1.41225e+03	300	480	0	283	0.0060
SCTAP2	1.72481e+03	1090	1880	0	781	0.0137
SCTAP3	1.39528e+03	1480	2480	1	1000	0.0160
SHARE1B	-7.65893e+04	117	225	0	179	0.0038
STOCFOR1	-4.11320e+04	117	111	0	79	0.0017
STOCFOR2	-8.14136e+12	2157	2031	1	1000	0.0288
WOOD1P	1.44290e+00	244	2594	0	305	0.0659

Tabela 5 – Algoritmo Clp sem presolve

Conclusão

Podemos concluir que os métodos de Pontos Interiores resolvem os problemas em menos iterações na maioria dos casos, pois as iterações ocorrem dentro da região factível do problema e utilizam trajetórias para auxiliar em cada iteração até encontrar a solução do problema. Ao contrário do método Simplex, por exemplo, que encontra a solução visitando os vértices da região factível até encontrar a solução, o que pode demorar para acontecer, já que existem problemas em que caminhar por todos os vértices pode ser muito demorado, pois a complexidade do problema é muito grande.

Com base no que foi apresentado em teorias e testes realizados com os problemas do Netlib [1], podemos ter certeza que quando os métodos de Pontos Interiores são bem implementados, eles são muito eficientes. A prova disso foi o algoritmo de Mehrotra implementado para a realização dos testes, que mesmo ingênuo, ganhou em alguns problemas no número de iterações do algoritmo Clp, que já é um algoritmo melhor implementado.

Referências

[1] BROWNE, S., DONGARRA, J.,GREEN, S., GROSSE,E., KEITH, M., ROWAN, T., WADE, R. *Netlib Services and Resources (Revised)*. University of Tennessee Knoxville, 1982.

[2] FORREST, J., NUEZ, L.D., LOUGEE-HEIMER, R. *CLP User Guide*. IBM Research, 1990, .

[3] WRIGHT, S.J. *Primal-Dual Interior-Point Methods*.SIAM, 1997.

[4] KHACHIYAN, L.G. *Polynomial Algorithms in Linear Programming*.New York, NY, USA,ACM, ,P.51–68, 1984.

[5] KARMAKAR, N. *A New Polynomial-time Algorithm for Linear Programming*.New York, NY, USA, ACM, p.302–311, 1984.

[6] DANTZIG, G.B. *Origins of the Simplex Method*New York, NY, USA, ACM, p.141–151, 1990.

[7] MEHROTRA, S.*On The Implementation of a Primal-Dual Interior Point Method*. SIAM Journal on Optimization, vol.2, 1992.

[8] BERTSIMAS, D., TSITSIKLIS, J. *Introduction to Linear Optimization*. Athena Scientific, ed. 1, 1997.

[9] RIBEIRO, A.A., KARAS, E.W. *Otimização Contínua: Aspectos teóricos e Computacionais*. Cengage Learning, 2014.

[10] NOCEDAL, J., WRIGHT, S.J. *Numerical Optimization*. New York, Springer, ed. 2, 2006.