Método de Projeções Ortogonais Sucessivas para Resolução de Problemas de Programação Linear

Percy Antonio Ticona Centeno

Método de Projeções Ortogonais Sucessivas para Resolução de Problemas de Programação Linear

Este exemplar corresponde a redação final da tese devidamente corrigida e defendida pelo Sr. PERCY ANTONIO TICONA CENTENO e aprovada pela Comissão Julgadora.

Campinas, 11 de Outubro de 1996.

Prof. Dr. Antonio Moretti.
Antonio C. Moretti.

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica, UNICAMP, como requisito parcial para obtenção do Título de MESTRE em MATEMÁTICA APLICADA.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Ticona Centeno, Percy Antonio

T437m Método de projeções ortogonais sucessivas para resolução de problemas de programação linear / Percy Antonio Ticona Centeno -- Campinas, [S.P. :s.n.], 1996.

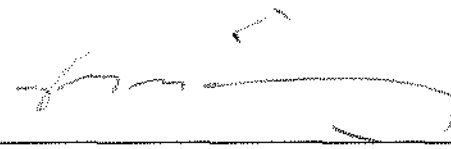
Orientador : Antonio Carlos Moretti

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

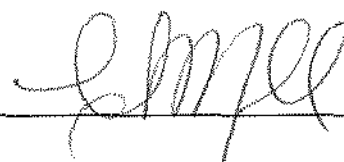
1. Programação linear. 2. Projeção ortogonal. I. Moretti, Antonio Carlos. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Tese de Mestrado defendida e aprovada em 11 de outubro de 199 6

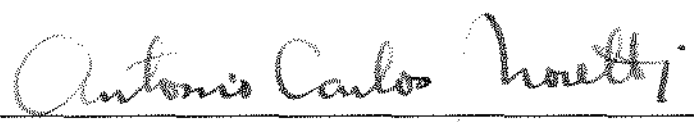
pela Banca Examinadora composta pelos Profs. Drs.



Prof (a). Dr (a). JOSÉ MÁRIO MARTÍNEZ PÉREZ



Prof (a). Dr (a). CLOVIS PERIN FILHO



Prof (a). Dr (a). ANTONIO CARLOS MORETTI

Método de Projeções Ortogonais Sucessivas para Resolução de Problemas de Programação Linear

Percy Antonio Ticona Centeno

orientado por:
Prof. Dr. Antonio C. Moretti

Matemática Aplicada
IMECC-UNICAMP
11 de Outubro, 1996.

Esta tese é dedicada à
minha família.

Agradecimentos.

- Eu quero agradecer primeiramente ao Prof. Antonio C. Moretti, pela sua orientação e confiança, que fizeram possível o desenvolvimento integral deste trabalho.

- Em segundo, ao Instituto de Matemática, Estatística e Ciência da Computação (IMECC-UNICAMP), pela oportunidade. E também a entidade CAPES, pelo apoio financiero.

- Agradeço também aos professores que integraram a comissão julgadora.

- Finalmente, um agradecimento muito especial a todas as pessoas amigas que me acompanharam nesta caminhada.

Conteúdo

Capítulo 1

Introdução.

Depois do primeiro algoritmo de tempo polinomial para se resolver o problema de programação linear, os métodos de pontos interiores tomaram grande importância, e um após outro, demostram uma melhor eficiência e desempenho. Podemos classificar estes métodos, em geral, em duas grandes categorias: o método das transformações projetivas e todas as suas variantes, os quais têm como representante o algoritmo de Karmarkar, e os métodos de trajetórias centrais, os quais são representados pelo centro analítico.

Neste trabalho, desenvolvemos um procedimento para se resolver o problema de programação linear, o qual usa a trajetória descrita pelos centros das maiores bolas inscritas em politopos. Naturalmente, calcular tais centros não é um procedimento trivial, portanto, nós procuramos uma aproximação usando uma modificação de projeções ortogonais sucessivas sobre os hiperplanos que definem o politopo de programação linear.

Nós procuramos discorrer sobre todos os temas que dizem respeito à resolução de um problema de programação linear. Para tanto, no capítulo 2, nós falamos de uma maneira concisa sobre complexidade de algoritmos; a razão disto é que, até meados de 1978, não se conhecia nenhum algoritmo polinomial para se resolver o problema de programação linear. Havia uma dúvida sobre em que classe os problemas de programação linear se encaixavam. O método simplex, apesar de ser muito eficiente na prática, tem complexidade exponencial na análise do pior dos casos. Portanto, para que nos capítulos subsequentes pudéssemos falar com mais clareza sobre complexidade dos algoritmos de pontos interiores, classe $\mathcal{NP}$-completo, etc, nós fizemos, no capítulo 2, uma breve introdução ao tema.

O capítulo 3 descreve o problema de programação linear, nele, apresentamos o método simplex e os métodos de pontos interiores. Falamos sobre os métodos dos elipsóides, embora este método seja muito ineficiente na resolução prática de problemas de programação linear, ele tem um valor histórico, pois retirou a dúvida a que nos referimos no parágrafo anterior, colocando assim, os problemas de programação linear na classe $\mathcal{P}$, isto é, na classe dos problemas para os quais existem algoritmos polinomiais para resolvê-los.

Centros de politopos é o tema do capítulo 4. Este assunto é de fundamental importância para a polinomialidade dos algoritmos de pontos interiores. Todo algoritmo de pontos interiores, que é provado ser polinomial, tem um mecanismo explícito ou implícito de centragem. Mas a noção de centro de um politopo não é trivial, pois dado um politopo geral de programação linear, existem várias definições de centros. Centro de Helly, de Fritz John, centro analítico são alguns exemplos de centros de politopos, sendo este último o mais usado para definir centros de politopos para programação linear. Ainda no capítulo 4, nós mostramos que se um desses centros nos é dado de graça, então nós temos um algoritmo polinomial para resolver problemas de programação linear. Este fato nos motivou a estudar uma maneira de se achar centros em politopos de uma forma eficiente. Portanto, no capítulo 5, apresentamos um método para resolver problemas de programação linear usando o centro da maior bola inscrita num politopo. Nós mostramos que o problema de achar a maior bola inscrita num politopo é um problema de programação linear por si só, mas nós desenvolvemos um procedimento que nada mais é do que uma modificação do método de projeções ortogonais sucessivas adaptado para achar a maior bola dentro de um politopo de programação linear. Exemplificamos este método com vários politopos no plano $\mathbb{R}^2$, com alguns politopos gerados de várias dimensões e com alguns problemas tirados do NETLIB. Nosso propósito inicial neste capítulo foi mostrar como calcular o centro da maior bola num politopo de programação linear. Depois, nós mostramos como usar este centro para resolver problemas de programação linear nos mesmos politopos anteriores.

Por fim, o capítulo 6 descreve as nossas conclusões sobre este trabalho e sugere sequências para futuros trabalhos.

Capítulo 2

Complexidade.

2.1 Complexidade de Algoritmos.

Um algoritmo é um conjunto de instruções, que executadas, num número finito de iterações, é possível obter uma resposta para um problema [*Br*]. É de nosso interesse ter à mão vários algoritmos para um determinado problema, e então, escolher o melhor, o mais eficiente para resolvê-lo. O objetivo deste capítulo é tentar classificar os algoritmos de acordo com sua eficiência, segundo algum critério estabelecido, determinando deste modo quando um algoritmo é mais eficiente que outro e o que é um bom algoritmo .

No final do capítulo, definiremos a complexidade de um problema, classificando deste modo os problemas nas classes $\mathcal{P}$ e $\mathcal{NP}$. Veremos também quando um problema é considerado $\mathcal{NP}$-completo.

Definição 2.1.1 *Um exemplar para o Problema de Programação Linear está caracterizado por um conjunto F e uma função custo c, de modo que*

$$F = \{ x \in \mathbb{R}^n \mid Ax = b,\ x \geq 0\}$$

$$c : x \longmapsto c^t x,$$

onde $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ e m, n inteiros positivos.

Intuitivamente, num exemplar, nós estamos introduzindo os dados suficientes de modo que o problema fique bem determinado, isto é, ingressar numericamente a matriz A , o vetor b e o vetor c. Assim, por exemplo, para

o problema de multiplicar dois números inteiros x e y, um exemplar constitui ingressar os dados (números) $x = 5$ e $y = 7$. Notemos que problemas interessantes incluem um número infinito de exemplares.

Naturalmente, estamos interessados em estudar problemas que sejam do tipo *resolvíveis*, isto é, existe um algoritmo para cada um deles capaz de obter uma solução, apesar de no caso geral não podermos garantir a solução ótima. Cabe notar também que podem existir problemas para os quais não é possível definir um algoritmo, como demostra Alán M. Turing (1936) no problema típico "Halting Problem".

Um algoritmo está *corretamente definido* se este resolve o problema para todos os exemplares; para demostrar que um algoritmo é *incorreto*, só temos que achar um exemplar para o qual não é possível encontrar uma resposta correta. Notamos também que é necessário definir o domínio de definição ao especificar o problema, isto é, o conjunto de exemplares a ser considerado; por outro lado, é mais fácil mostrar que um algoritmo é incorreto que correto.

Representamos a medida de um exemplar de um problema de programação linear por (m, n, L), onde L é o número de bits binários necessários para armazenar todos os dados do problema e é conhecido como comprimento do ingresso de um exemplar do problema. Dado um número α, notamos que é possível usar $1 + \log_2(1 + \alpha)$ bits para representar α no sistema binário, logo.

$$L = 1 + [1 + \log_2(1 + m)] + [1 + \log_2(1 + n)] + [m + \sum_i \log_2(1 + b_i)] +$$
$$+[n + \sum_j \log_2(1 + c_i)] + [mn + \sum_i \sum_j \log_2(1 + a_{ij})].$$

Para afirmar quando um algoritmo é mais eficiente que outro, nós vemos primeiro que não é possível impor uma unidade para expressar a eficiência teórica de um algoritmo (por exemplo, segundos) pois, não existe um computador padrão para realizar todas as medidas.

Assim, quando escolhemos um algoritmo para resolver um determinado problema, pelo acesso teórico (que é a análise que tentamos fazer), nós adotamos o *princípio de invariância*, isto é, duas implementações diferentes do mesmo algoritmo não diferem em eficiência por mais de uma constante multiplicativa, mais precisamente, se duas implementações tomam $t_1(n)$ e $t_2(n)$ segundos, respectivamente, ao resolver um exemplar de medida n, então deve sempre existir uma constante c tal que $t_1(n) \leq ct_2(n)$, onde n é suficientemente grande. Assim, uma mudança de máquina (computador) pode nos

permitir resolver o problema 10 ou 200 vezes mais rápido, mas só uma mudança de algoritmo nos dá um melhoramento quando o tamanho do exemplar aumenta. Cabe notar que existe outro tipo de acesso, o empírico, onde um algoritmo é escolhido conforme seu comportamento quando é ensaiado com vários exemplares com ajuda de um computador.

Resumindo, não usamos unidades para especificar a eficiência de um algoritmo. Falamos que um algoritmo toma um tempo na *ordem de* $t(n)$, para uma função real t dada, se existe uma constante positiva e uma implementação do algoritmo, capaz de resolver o problema num tempo limitado superiormente por $ct(n)$ segundos, onde n é o tamanho do exemplar considerado. Agora introduzimos a notação para o termo *ordem de* formalmente.

Definição 2.1.2 *Sejam $\mathbb{N}$ e $\mathbb{R}$ os conjuntos dos números naturais e reais, respectivamente, $\mathbb{R}^+$ são os reais positivos e $f : \mathbb{N} \longmapsto \mathbb{R}^+$ é uma função arbitrária; nós definimos*

$$O(f(n)) = \{t : \mathbb{N} \longmapsto \mathbb{R}^+ \,|(\exists c \in \mathbb{R}^+)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)[t(n) \leq cf(n)]\}. \tag{2.1}$$

$O(f(n))$, lemos ordem de $f(n)$, é o conjunto de todas as funções reais $t(n)$ limitadas superiormente por um múltiplo real positivo de $f(n)$, onde n é suficientemente grande ($n > n_0$).

Cabe ressaltar que existem outras notações para *ordem de*, tais como:

$$\Omega(f(n)) = \{t : \mathbb{N} \longmapsto \mathbb{R}^+ \,|(\exists c \in \mathbb{R}^+)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)[t(n) \geq cf(n)]\} \tag{2.2}$$

$\Omega(f(n))$ é o conjunto de todas as funções $t(n)$ limitadas inferiormente por um múltiplo positivo real de $f(n)$, com n grande, notamos imediatamente a seguinte propriedade.

Teorema 2.1.1 *Sejam as funções arbitrárias f e $g : \mathbb{N} \longmapsto \mathbb{R}^+$, $f(n) \in O(g(n))$ se, e somente se, $g(n) \in \Omega(f(n))$.*

Com $O(f(n))$ estimamos o limite superior do tempo que algum algoritmo toma sobre um exemplar dado. Com $\Omega(f(n))$ estimamos o limite inferior deste tempo.

Se um algoritmo toma um tempo da $O(f(n))$ no pior caso, então existe um $c \in \mathbb{R}^+$ tal que o tempo $cf(n)$ é suficiente para que o algoritmo resolva

o pior exemplar de tamanho n, para n suficientemente grande, este tempo é obviamente suficiente para resolver qualquer exemplar desse tamanho n. Se um algoritmo toma um tempo da $\Omega(f(n))$, no pior caso, deve existir $d \in I\!R^+$, tal que o algoritmo toma um tempo maior que $df(n)$ para resolver o pior exemplar de tamanho n, para n grande, podendo tomar menor tempo para resolver outros exemplares do mesmo tamanho n. Assim, deve existir uma infinidade de exemplares para as quais o algoritmo toma um tempo menor que $df(n)$.

As condições favoráveis se mostram, quando na análise do comportamento assintótico de um algoritmo, seu tempo de execução é limitado tanto inferiormente como superiormente; definimos uma outra notação:

$$\Theta(f(n)) = O(f(n)) \bigcap \Omega(f(n)), \tag{2.3}$$

chamada *a ordem exata de* $f(n)$.

Propriedades e operações sobre estas notações podem ser vistas por exemplo em $[Br]$, além de outras.

Não existe nenhuma fórmula mágica para analisar a eficiência de um algoritmo, isso depende do critério, juízo, intuição e experiência, muitas vezes um primeiro análise alcança uma complicada função, na qual se envolvem somatórias e recorrências pouco esperançosas.

Exemplo 2.1.1 *O problema da programação linear, que é o que mais nos interessa, tem como um bom algoritmo para a sua resolução o algoritmo simplex, este algoritmo é muito eficiente em grande parte dos exemplares, mas é catastrófico em uns poucos. O algoritmo simplex, no pior caso, toma um tempo exponencial para resolver o problema de programação linear.*

Exemplo 2.1.2 *Dentro dos algoritmo de tipo guloso, o algoritmo de Kruskal toma um tempo de* $O(a\log n)$ *para resolver o problema da Árvore Geradora Mínima, onde* a *é o número de arcos (arestas) e* n *o número de nós de um determinado grafo conectado. Existe outro algoritmo que resolve o mesmo problema, o algoritmo de Prim, o qual toma um tempo de* $O(n^2)$. *A diferença entre os dois é que num caso esparso, o algoritmo de kruskal é relativamente melhor, pois* $a \to n$ *e temos* $O(n\log n)$; *no entanto, que o algoritmo de Prim toma um tempo de* $O(n^2)$. *No caso denso, claramente* $a \to n(n-1)/2$ *e o algoritmo de Kruskal toma* $O(n^2\log n)$, *sendo mais vantajoso usar aqui o algoritmo de Prim.*

Nós estamos interessados em obter algoritmos cada vez melhores para determinados problemas, por exemplo, obter um algoritmo que toma um tempo de $O(p(n))$, onde p é um polinômio e n é o tamanho do exemplar do problema, quando só tínhamos um algoritmo de tempo exponencial, $O(e(n))$, onde e é uma função exponencial. Obter avanços como esse são os constantes objetivos de pesquisadores nessa área.

O algoritmo Simplex, embora tenha uma complexidade de tempo exponencial, é muito eficiente na prática, para a maioria dos exemplares do problema de Programação Linear, falamos de exemplares de tamanho considerável, mas obtém resultados ruins quando os tamanhos dos exemplares aumentam e quando estes possuem uma estructura particular. Nessas circunstâncias, um melhor comportamento apresentam os chamados Métodos de Pontos Interiores, os quais possuem, em geral (teoricamente), uma complexidade polinomial.

São aceitáveis como eficientes, algoritmos que na sua ordem apresentem funcões polinomiais p com grau não muito grande, $p(t) = t^8$ por exemplo, obviamente não podemos aceitar como eficiente um algoritmo que toma um tempo polinomial na $O(t^{800})$. Por outro lado, um tema de grande discussão é que se um algoritmo de tempo polinomial é necessariamente melhor que um algoritmo de tempo exponencial, a resposta é relativa, teoricamente, o algoritmo simplex é menos eficiente que algum algoritmo de tempo polinomial, de pontos interiores por exemplo, pois tem complexidade exponencial, mas na prática, experimentos computacionais mostram que o algoritmo simplex tem bom desempenho.

2.2 Complexidade de um Problema.

Anteriormente, nós definimos complexidade de algoritmos, agora, falaremos sobre um campo que caminha paralelamente com os algoritmos, Complexidade Computacional, considerando globalmente a classe de todos os algoritmos que são competentes ao resolver um determinado problema.

Usando algoritmos podemos provar o seguinte, que um determinado problema pode ser resolvido num tempo de $O(f(n))$ para alguma função f, a qual nós tentamos reduzir tanto como seja possível. Enquanto que, usando complexidade (para problemas), nós queremos procurar uma função $g(n)$, tão grande como seja possível, e então provar que qualquer algoritmo que é capaz

de resolver o nosso problema corretamente, sobre todas os seus exemplares, deve tomar necessariamente um tempo $\Omega(g(n))$.

A satisfação seria completa se obtivéssemos uma função f em $\Theta(g(n))$, pois então, achamos o algoritmo mais eficiente possível (exceto talvez por mudanças na constante multiplicativa); neste caso, falaremos que a complexidade do problema é conhecida exatamente, mas nem sempre acontece assim.

Anteriormente falamos sobre algoritmos eficientes, imediatamente temos em mente os de tempo polinomial, algoritmos eficientes para problemas como do Emparelhamento e da Árvore Geradora Mínima existem e são de tempo polinomial, mas na realidade, para a maioria dos problemas de otimização combinatória, não se conhecem algoritmos deste tipo. Assim, estes problemas em geral são difíceis de resolver, temos por exemplo, os problemas Programação Linear Inteira, "satisfability", Clique, Problema do Caixeiro Viajante, etc.

Agora definimos intuitivamente um problema $\mathcal{NP}$-completo, como um problema computacional que é tão difícil de resolver como qualquer problema razoável, problemas que estão dirigidos à determinação de traços e desenhos otimais de objetos tais como rotas, conjuntos de nós, partições, e listas de inteiros, todos estes conceitos sujeitos a uma formulação precisa. Mais adiante definiremos formalmente este tipo de problema, quando falemos das classes $\mathcal{P}$, $\mathcal{NP}$, reduções e transformações a tempo polinomial. Agora introduzimos duas interesantes propriedades que caracterizam um problema $\mathcal{NP}$-completo:

- Nenhum problema $\mathcal{NP}$-completo pode ser resolvido por algum algoritmo de tempo polinomial conhecido.
- Se existe um algoritmo de tempo polinomial para algum problema $\mathcal{NP}$-completo, então existem algoritmos polinomiais para todos os problemas $\mathcal{NP}$-completos.

Falando computacionalmente, os problemas $\mathcal{NP}$-completos são pouco tratáveis, se existe um algoritmo capaz de obter uma solução para um determinado exemplar deste problema, este apresentará na sua execução um tempo de ordem exponencial no pior caso, daí que são impraticáveis para todos os exemplares. Definimos agora um exemplar de um problema de otimização de forma geral.

Definição 2.2.1 *Um exemplar de um problema de otimização é um par (F,c), onde F é qualquer conjunto (ou domínio de pontos viáveis), c é a função custo*

$$c : F \longmapsto \mathbb{R};$$

deseja-se determinar $f \in F$, tal que $c(f) \leq c(y)$, $\forall y \in F$. Neste caso, f é chamada solução global ótima, ou simplemente uma solução ótima.

Assim, um problema de otimização tem definido um determinado conjunto de exemplares (F, c), onde F é o conjunto de soluções viáveis e c é a função custo. Naturalmente, em otimização combinatória, é possível enumerar todas as soluções (finitas) viáveis e seus respectivos valores de c para cada uma delas, mas isso não sempre é possível pois os problemas mais interessantes têm exemplares com um número muito grande de soluções viáveis.

Agora assumimos que F e c são dados implicitamente em termos de dois algoritmos $@_F$ e $@_c$. O algoritmo $@_F$, dado um objeto combinatorial f e um conjunto de parâmetros S, ele decidirá quando f é um elemento de F. O algoritmo $@_c$, dado uma solução viável f e um conjunto de parâmetros Q, ele retorna o valor $c(f)$. Resumindo, $@_F$ verifica se existe um f viável e $@_c$ avalia f e obtém $c(f)$. Um exemplar do problema combinatorial pode agora se definir como a representação dos parâmetros em S e Q.

Exemplo 2.2.1 *Um exemplar do Problema do Caixeiro Viajante tem como parâmetro S o inteiro n, o parâmetro Q consta das entradas da matriz das distâncias $[d_{ij}]$. O algoritmo $@_F$, dado um objeto f e o parâmetro n, verifica se f é um tour de n cidades. o algoritmo $@_c$ calcula o custo $c(f)$ somando todas as entradas d_{ij} correspondentes ao tour f.*

Exemplo 2.2.2 *Num exemplar do problema de Programação Linear Inteira, o algoritmo $@_F$ tem como parâmetro S a matriz A e o vetor b, dado qualquer vetor inteiro x, $@_F$ testa quando $Ax = b$ e $x \geq 0$, o algoritmo $@_c$ tem como parâmetro Q, o vetor custo c, e avalia $c^t x$ para cada solução viável x dada por $@_F$.*

Vemos que nos dois exemplos, $@_F$ e $@_c$ são algoritmos de tempo polinomial.

2.2.1 Versões de um Problema de Otimização Combinatorial.

São as seguintes as formas de problemas computacionais:

i. Dadas as representações dos parâmetros S e Q, e os algoritmos $@_F$ e $@_c$, o problema é achar a solução viável ótima. Esta versão é chamada "Versão Otimização" do problema.

ii. Dados S e Q, achar o custo da solução ótima, esta é a "Versão Avaliação".

iii. Dado um exemplar, consideremos sua representação, isto é, S e Q, além de um número inteiro L; existe uma solução viável $f \in F$ tal que $c(f) \leq L$? ($c(f) \geq L$ para um máximo), esta é conhecida como a "Versão Reconhecimento", a qual tem muito uso no estudo de complexidade.

Cabe notar que ii) não é mais difícil de resolver que i), além disso, a terceira versão, a qual será de muito uso no estudo da complexidade de um problema, é quase um protótipo de um problema computacional tradicionalmente estudado pela teoria da computação, esta versão é uma questão, a diferença das outras duas; além disso, iii) não é mais difícil de resolver que ii) e portanto, que i), pois é só achar uma solução viável e avaliá-la, para logo fazer uma comparação e finalmente dar uma resposta "sim" ou "não". Com determinadas hipóteses, é possível resolver a versão avaliação desde a versão reconhecimento.

Notemos também que a versão de reconhecimento não é mais difícil que a versão de otimização. Além disso, qualquer resultado negativo provado com respeito à complexidade do problema de reconhecimento pode-se aplicar à versão de otimização.

Os problemas antes referidos como Halting e satisfability são exemplos típicos de um problema na versão de reconhecimento.

2.2.2 Classe $\mathcal{P}$.

Problemas de reconhecimento desta classe podem ser resolvidos por um algoritmo de tempo polinomial (para todo exemplar do domínio). No entanto,

é possível definir esta classe $\mathcal{P}$ precisamente, em termos de qualquer formalismo matemático para algoritmos, como a Máquina de Turing [*Ba*]. Em outras palavras, $\mathcal{P}$ é a classe de problemas de reconhecimento relativamente fáceis de resolver, alguns problemas que estão na classe $\mathcal{P}$ são por exemplo o Emparelhamento Máximo, Árvore Geradora Mínima, Conectitividade em Grafos.

2.2.3 Classe $\mathcal{NP}$.

Consideremos um determinado problema de reconhecimento; para que o problema esteja na classe $\mathcal{NP}$, não precisamos que todo exemplar seja resolvido em tempo polinomial por algum algoritmo, só exigimos que se ξ é um exemplar "sim" do problema (i.e, que existe uma solução viável f em iii), então existe um algoritmo @ e um "certificado conciso" c_ξ, i.e, confirmar a existência de uma solução viável c_ξ, pudendo não ser determinada explicitamente, c_ξ deve ser limitado em medida por um polinômio avaliado na medida do exemplar ξ, é dizer, $|c_\xi| \leq p(|\xi|)$, onde $|.|$ denota a medida do exemplar ξ, tal que c_ξ deve ser verificado em tempo polinomial pelo algoritmo @.

A intenção de garantir o certificado conciso c_ξ desse modo, deve-se ao fato que não estamos interessados em resultados ótimos os quais não podem ser desenhados (escritos) em tempo razoável.

Exemplo 2.2.3 *O problema de Programação Linear Inteira está em $\mathcal{NP}$. Consideremos este problema na versão reconhecimento; é possível transformar $c^t x \leq L$ numa equação com ajuda de variáveis de folga, para logo ser adicionada no sistema $Ax = b$. Deste modo, a versão reconhecimento para este problema toma a forma seguinte:*

Dada a matriz A e o vetor b, existe um vetor inteiro x tal que $Ax = b$ e $x \geq 0$?

Para verificar que este problema pertence à classe $\mathcal{NP}$, consideremos um exemplar "sim" para o problema de PLI, agora, se este tem uma solução viável, então todos os seus componentes estão limitadas por

$$M = n(ma_1)^{2m+3}(1 + a_2),$$

onde $a_1 = max\{|a_{ij}|\}$ e $a_2 = max\{|b_i|\}$, logo podemos tomar justamente essa solução viável como o "certificado conciso" para este exemplar "sim" pois,

por exemplo, o comprimento da sua representação binária é polinomial com respeito à medida do seu ingresso.

Exemplo 2.2.4 *Consideremos o problema do Clique Máximo na versão de reconhecimento, dado um grafo $G = (V, E)$ e um inteiro k, existe um clique de cardinalidade k ?*

Uma forma óbvia de resolver o problema seria de enumerar os $\binom{|v|}{k}$ subconjuntos de v com cardinalidade k e testar quando algum deles forma um clique, o problema é que existem um número muito grande, exponencialmente longo de tais subconjuntos. Nenhum algoritmo de tempo polinomial é conhecido para este problema. Assim, o problema do clique aqui tratado não está na classe $\mathcal{P}$. Agora verificamos que este problema está em $\mathcal{NP}$. Suponhamos que temos dado um exemplar "sim" ξ, isto é, que temos um clique C de cardinalidade k do grafo G, uma lista de nós no conjunto C determinam um "certificado conciso" c_ξ do exemplar, claramente a sua medida está limitada polinomialmente pela medida do ingresso no exemplar deste problema, c_ξ pode ser verificado eficientemente por um algoritmo, pois C tem só n nós, o polinômio p pode ser escolhido como $2n^2$, só resta testar se os nós realmente estão conectados por arcos em E, isto último pode ser feito claramente em tempo polinomial.

Problemas como do Caixeiro Viajante e "Satisfability", por exemplo, pertencem também à classe $\mathcal{NP}$. É natural afirmar agora que $\mathcal{P}$ é um subcunjunto de $\mathcal{NP}$, no entanto, nenhuma prova formal é conhecida para afirmar que $\mathcal{P} = \mathcal{NP}$, isso ainda é uma questão aberta para pesquisadores na matéria.

2.2.4 Reduções Para Tempo Polinomial.

Definição 2.2.2 *Sejam P_1 e P_2 problemas de reconhecimento ("sim" ou "não"), P_1 se reduz em tempo polinomial a P_2, se, e somente se, existe um algoritmo de tempo polinomial $@_1$ para P_1, o qual usa em várias chamadas (como uma subrutina a custo unitário) um algoritmo $@_2$ para P_2. $@_1$ é denominado redução de tempo polinomial de P_1 a P_2.*

Notemos que chamamos de "custo unitário" ao fato que o algoritmo $@_2$ é considerado como uma simples instrução , tomando um tempo unitário de execução, isto é só uma hipótese teórica pois, $@_2$ pode ser pouco eficiente

e o problema se complicaria. O fato interessante acontece quando $@_2$ é de tempo polinomial.

Teorema 2.2.1 *Se P_1 se reduz polinomialmente a P_2, e existe um algoritmo de tempo polinomial para P_2, então deve existir um algoritmo polinomial para P_1.*

Nós agora definimos informalmente uma "Transformação para Tempo Polinomial." Dizemos que P_1 se transforma polinomialmente para P_2, se existe uma redução de tempo polinomial de P_1 a P_2 , com só uma chamada do algoritmo $@_2$ para P_2 exatamente no final do algoritmo $@_1$ para P_1; o resto do algoritmo $@_1$ é dedicado a construir o ingresso para $@_2$. Nós usamos esta última definição para definir a sua vez um problema $\mathcal{NP}$-completo.

2.2.5 Problema $\mathcal{NP}$-completo.

Definição 2.2.3 *Um problema de reconhecimento $R \in \mathcal{NP}$ é chamado $\mathcal{NP}$-completo, se todo outro problema em $\mathcal{NP}$ se transforma polinomialmente a R.*

Notamos imediatamente, que se um problema R é $\mathcal{NP}$-completo, e existe um algoritmo eficiente para R, então, pelo teorema anterior, devem existir algoritmos eficientes para todos os problemas em $\mathcal{NP}$. No caso de existir um algoritmo polinomial para algum problema $\mathcal{NP}$-completo, aconteceria que $\mathcal{P} = \mathcal{NP}$, lamentavelmente não se conhece da existência alguma.

Teorema 2.2.2 *(Transitividade) Sejam os problemas R_1, R_2, R_3, se o problema R_1 se transforma polinomialmente em R_2, e, R_2 se transforma polinomialmente em R_3, então R_1 se transforma polinomialmente em R_3.*

Para provar que um determinado problema X é $\mathcal{NP}$-completo, nós devemos mostrar duas coisas:

1. Que o problema está em $\mathcal{NP}$ e
2. Que todo outro problema em $\mathcal{NP}$ se transforma polinomialmente em X.

A técnica para verificar que um problema X cumpre 2), faz uso do teorema (2.2.2) para demostrar que um problema conhecido é transformável polinomialmente em X, assim, se esse problema é $\mathcal{NP}$-completo, então qualquer problema em $\mathcal{NP}$ é transformável polinomialmente nele, e portanto, em X.

Teorema 2.2.3 *(COOK) O problema "Satisfability" é $\mathcal{NP}$-completo.*

Para verificar que grande parte dos problemas são $\mathcal{NP}$-completos, é possível transformar o problema de "Satisfability" em tais, isto geralmente apresenta complicadas provas, uma ampla análise pode-se encontrar por exemplo em [*P*], nós só falamos aqui de alguns exemplos.

O problema de programação linear inteira é $\mathcal{NP}$-completo. Vimos anteriormente que este problema está na classe $\mathcal{NP}$, além disso, é possível transformar polinomialmente o problema "Satisfability" num problema de Programação Linear Inteira. Problemas como do Clique, Circuito Hamiltoniano, Caixeiro Viajante, 0-1 Knapsack (mochila), Partição, e outros são também $\mathcal{NP}$-completos.

Capítulo 3

O Problema da Programação Linear.

O ano de 1947 separa o desenvolvimento teórico da programação linear em duas fases, a primeira, caracteriza-se pelo pouco conhecimento de um problema importante como a programação linear, apesar de alguns trabalhos feitos por pesquisadores nesses anos, como Fourier (1823), Valle Poussin (1911), e alguns outros. Para ter uma visão do pouco conhecimento do problema, Motzkin, na sua tese de doutorado listou apenas 32 artigos antes de 1936, sobre sistemas de inequações lineares, segundo nos fala George B. Dantzig [*D*] no seu reportagem (1981), no entanto, uma pessoa obteve notórios avanços, Kantorovich (1939), só que pela falta de interação entre a U.R.S.S. e o Ocidente, ocasionado pela guerra fria, estes artigos não foram divulgados. A segunda, caracteriza-se pela existência de um amplo aporte de métodos, os quais, com diferentes estruturas, procuram uma melhor complexidade.

3.1 O Algoritmo Simplex.

No transcorrer da Segunda Guerra Mundial, e durante o ano 1946, George Dantzig foi consultor matemático da Tesouraria da Força Armada dos U.S. Com o objetivo de mecanizar o processo de planejamento e inspirado no trabalho de Wassily Leontief (o qual propôs em 1932, uma estrutura de matriz simples com o chamado "Modelo Input-output para a economia da interindústria americana"), é que em meados de 1947, Dantzig formula o

"Problema de planejamento" como um conjunto de axiomas, o sistema matemático resultante a ser resolvido envolvia a minimização de uma forma linear, sujeita a equações e inequações lineares. Posteriormente veio a pergunta nada trivial: pode-se resolver tal problema ? .

A resposta não demorou muito, quando no verão de 1947, tomando idéias da sua tese de pós-graduação, Dantzig desenvolve o algoritmo simplex, uma técnica muito eficiente para resolver programas lineares que, para sorte sua, segundo suas próprias palavras, deu certo !

Posteriormente, tentou conhecer quão eficiente que era o seu algoritmo, e depois de visitar em outubro 3 de 1947 a John Von Neumann (considerado por muitos nesse então, como um dos matemáticos mais importantes da época) e ao professor Tuker, em junho de 1948, ele obteve informação sobre o lema de Farkas e dualidade. Durante o período de 1947-1948, problemas como o "Processo de Planejamento" e outros foram resolvidos. A sugestão na demostração da convergência do algoritmo simplex, de assumir a não degeneração, foi feita por Koopmans, a convergência do algoritmo foi demostrado formalmente em março de 1951. No verão do mesmo ano, Philip Wolfe, estudante da universidade de Berkeley, propôs uma forma lexicográfica para evitar a ciclagem quando o problema era degenerado. Todas estas variantes conseguiram fazer do algoritmo simplex uma ferramenta prática como teórica, pois, também é usado para provar teoremas, mas é preciso que para tal fim, assumir a não degeneração. Nos anos de 1950-1960, novas áreas surgiram, tais como Programação Não Linear, Teoria de Fluxo em Redes, Métodos de Grande Escala, Programação Estocástica, Programação Inteira, e outros.

Em 1978, L.G. Khachian demostrou que um algoritmo de tipo elipsoidal resolve o problema de programação Linear (PL) em tempo polinomial, este algoritmo teve só um valor teórico mas não prático.

Está demostrado que o problema de Programação Linear não pode ser resolvido sem o uso de um algoritmo, estes problemas devem ser resolvidos eficientemente usando um número de iterações que possam garantir a obtenção de uma solução ótima, o algoritmo simplex ou uma de suas variantes fazem esse trabalho.

Consideraremos a seguir, o problema de programação linear na sua forma padrão

$$\begin{aligned} &\text{Minimizar} \quad c^t x \\ &\text{sujeito a:} \\ &\qquad Ax \;\; = b \\ &\qquad x \;\; \geq 0, \end{aligned} \tag{3.1}$$

onde $x, c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, e $A \in \mathbb{R}^{m \times n}$. Notemos que todo problema de programação linear pode ser expressado na forma padrão, nós usaremos frequentemente essa notação.

Agora, sem perda de generalidade, no problema (3.1), suponhamos que $m < n$ e o posto de A seja completo (ie. posto$(A) = m$). Se o problema tem uma solução viável ótima, então ele tem uma solução básica viável ótima. Sabemos que o algoritmo simplex gera soluções básicas, então ele é capaz de resolver o problema num número finito de passos; caso contrário, reportará uma solução ilimitada ou a infactibilidade do problema.

Como falamos no capítulo anterior, na prática, o algoritmo simplex é bom na maioria dos exemplares, mas é ruim em poucos. Isto deve-se ao seguinte fato.

O Método Simplex não é um Algoritmo de Tempo Polinomial. Este resultado deve-se a Klee e Minty [KM] (1972), que com um argumento simples, estabeleceram a existência de uma família de exemplares do problema de programação linear, nos quais o algoritmo simplex toma um tempo exponencial. Eles constroem uns exemplares nos quais o algoritmo simplex leva seu pior caso. Analisar o pior caso de um algoritmo significa:

(a) Achar o exemplar mais desfavorável.

(b) obter a sequência mais desfavorável de eleições dos passos especificados incompletamente.

Para mostrar que o algoritmo simplex é não polinomial, é suficiente exibir uma família de exemplares para os quais este deve usar um número exponencial de pivoteamentos, ou seja, exibir uma sequência exponencial (com respeito à medida do exemplar do problema PL) de SBV's $x_1, x_2, ..., x_k$, tais que x_i e x_{i+1} são adjacentes e satisfazem

$$c^t x_{i+1} \leq c^t x_i \quad \text{i=1,2,...,k},$$

o teorema seguinte formaliza a afirmação.

Teorema 3.1.1 *(Klee e Minty [KM]) Para todo $d > 1$, existe um problema de programação linear com $2d$ equações, $3d$ variáveis, e coeficientes inteiros com valor absoluto limitado por 4, tal que o algoritmo simplex deve tomar $2^d - 1$ iterações para achar a solução ótima.*

Conclusão, o algoritmo simplex tem complexidade exponencial no pior caso. O primeiro algoritmo (procedimento) de pontos interiores de tempo polinomial, que resolve o problema de programação linear, foi mostrado por L.G. Khachian (1978). Isso, no estudo de complexidade, permitiu classificar este problema do seguinte modo.

O Problema de Programação Linear está na classe $\mathcal{P}$.

Surgiram muitos outros métodos de pontos interiores para o problema de programação linear, depois do método do elipsóide, alguns conseguiram melhorar a complexidade do algoritmo inicial, mas geralmente, por sua natureza intrínseca, os algoritmos polinomiais (de pontos interiores) estão relacionados infelizmente com o pior caso.

3.2 Métodos de Pontos Interiores.

A existência de um algoritmo de tempo polinomial para um problema PL, foi mostrada em 1978, o qual coloca este problema na classe $\mathcal{P}$, se deve a L.G. Khachian. Ele propôs um algoritmo de tempo polinomial para resolver o problema de factibilidade linear.

3.2.1 O Método dos Elipsóides.

Este método foi desenvolvido originalmente para programação não linear por Shor [*Sh*] (1970-1977), Yudin e Nemirovskii [*YN*] (1976). E foi Khachian (1978) quem demostrou que uma extensão resolve o problema de PL em tempo polinomial. Um outro procedimento é feito para um problema de PL sobre poliedros implicitamente dados, o qual gera uma solução em tempo polinomial para vários problemas em otimização combinatória, aqui nos restringimos a falar sobre o trabalho de Khachian.

Alguns autores comparam o método do elipsóide com o método de relaxação, o qual é mais geral pelo seguinte argumento.

- O método de relaxação acha uma série de pontos $z_0, z_1, z_2, \ldots$ onde cada z_{i+1} vem dado desde a projeção de z_i sobre uma inequação violada do sistema $Ax \leq b$, o objetivo é encontrar uma solução viável para tal sistema, é possível também adaptar este método para variáveis não negativas e problemas PL.

Por outro lado, Levin ([LE] 1965) e Newmann ([NE] 1965) apresentam um método de seções centrais o qual aproxima-se ao mínimo de uma função convexa f sobre um politopo P como segue: seja $P_0 = P$, se P_i é determinado, seja z_i o centro de gravidade de P_i, α o gradiente de f em z_i e

$$P_{i+1} = \{x \in P_i \mid \alpha^t x \leq \alpha^t z_i\},$$

como f é convexa, seu mínimo será alcançado em P_{i+1}, pois

$$\text{vol}(P_{i+1}) \leq (1 - e^{-1})\text{vol}(P_i),$$

logo z_i converge a um ponto mínimo.

Calcular esses centros de gravidade pode ser difícil na prática, Yudin e Nemirovskii (1976), Shor (1977), verificaram que, se substituindo os politopos por elipsóides (E_0 é um elipsóide contendo P e E_{i+1} é o menor elipsóide contendo $\{x \in E_i \mid \alpha^t x \leq \alpha^t z_i\}$), este é mais vantajoso e é aplicável para qualquer problema de programação convexa. O método do elipsóide é baseado justamente nestes argumentos. Além disso, eles observaram que este método pode ser visto como um caso especial do método de Shor, o qual usa transformações projetivas do espaço, neste caso, a matriz definida positiva que define o elipsóide representa a transformação.

Khachian demostrou que para problemas de programação linear, o método do elipsóide pode ser modificado, dando assim soluções em tempo polinomial.

Os métodos anteriormente falados, de Levin e Newmann, e o método do elipsóide, chamados também "Métodos de Centragem" ou de "Seções Centrais", podem ser incluídos numa classe $\mathcal{K}$ de corpos geométricos (conjuntos com interior não vazio) onde: dado um z_i num conjunto $C_i \in \mathcal{K}$, achar o espaço médio afim H, com z_i sobre sua fronteira, tal que os pontos procurados estão em H, $C_{i+1} \in \mathcal{K}$ tem menor volume, $\{C_i \bigcap H\} \subset C_{i+1}$ e z_{i+1} é o centro de gravidade de C_{i+1}.

Nessa forma, o método de relaxação pode ser considerados como um "ball method" onde $\mathcal{K}$ é uma classe de bolas. Yamnistsky e Levin demostraram

que se tomamos por $\mathcal{K}$ as classes de todos os "simplices," obtemos um método similar ao algoritmo simplex o qual é de tempo polinomial para problemas PL.

3.2.2 Resumo do Algoritmo de Khachian.

Este método é inicialmente aplicado para resolver o problema de inequações lineares com $n \geq 2$, isto é, dada uma matriz $A_{m\times n}$ e um vetor b m-dimensional, existe um vetor x n-dimensional tal que $Ax \leq b$? Vamos assumir, inicialmente, que

1. O poliedro $P = \{x \mid Ax \leq b\}$ é limitado e de dimensão completa.
2. Podemos fazer os nossos cálculos com precisão infinita.

Seja $v = 4n^2\xi$, onde ξ é o máximo da medida do ingresso nas linhas de $[A\ b]$, logo, cada vértice de P tem medida limitada por v (cada solução extrema de P), se tomamos $R = 2^v$, então $P \subset \{x \mid \ ||x|| \leq R\}$.

Determinamos recursivamente uma sequência de elipsóides $E_0, E_1, E_2, ...$ de volume decrescentes, tal que $P \subset E_i\ \forall i$, também uma sequência de centros $z_0, z_1, z_2, ...$ e uma sequência de matrizes definidas positivas $D_0, D_1, D_2, ...$ onde

$$E_i = \text{elip}(z_i, D_i),$$

o primeiro elipsóide é a bola $\{x \mid \ ||x|| \leq R\}$, com $z_0 = 0$ e $D_0 = R^2 I$.

Se z_i satisfaz $Ax \leq b$ paramos, pois já temos uma solução, caso contrário, seja $a_k x \leq b_k$ a inequação de $Ax \leq b$ a qual foi violada por z_i e seja

$$E_{i+1} = \text{elip}(z_{i+1}, D_{i+1}),$$

o elipsóide de menor volume contendo

$$E_i \bigcap \{x \mid a_k x \leq a_k z_i\},$$

os valores de z_{i+1} e D_{i+1} são dados pelo teorema (3.2.1), com $z' = z_{i+1}$, $D' = D_{i+1}$ e $\alpha = a_k$. Notemos agora que o elipsóide E_{i+1} contém por sua vez P, como $P \subset E_i$, temos

$$P \subset \{x \mid \ a_k x \leq b_k\} \subset \{x \mid \ a_k x \leq a_k z_i\},$$

isto é repetido iterativamente até obter uma solução para o problema principal (até alcançar uma solução factível).

Notamos também que $\frac{\text{vol}(E_{i+1})}{\text{vol}(E_i)} \leq e^{-\frac{1}{2n+2}}$, onde $\text{vol}(E_0) \leq (2R)^n$, logo, $\text{vol}(E_i) \leq e^{-\frac{i}{2n+2}}(2R)^n$.

O volume de P é limitado inferiormente, pois P tem dimensão completa, isto pode-se verificar se consideramos $x_0, x_1, x_2, ..., x_n$, vértices afins independentemente de P, então

$$\begin{aligned} \text{vol}(P) &\geq \text{vol}(\text{sup.conv}\{x_0, x_1, x_2, ..., x_n\}) \\ &= \frac{1}{n!}|det[1 \ \ (x_0, x_1, ..., x_n)]^t| \\ &\geq n^{-n}2^{-nv} \\ &\geq 2^{-2nv}, \end{aligned} \tag{3.2}$$

se alcançarmos o elipsóide E_N, na iteração N, com $N = 16n^2v$, devemos obter a seguinte contradição

$$2^{-2nv} \leq \text{vol}(P) \leq \text{vol}(E_N) < e^{-\frac{N}{2n+2}}(2R)^n \leq 2^{-2nv},$$

pois $P \subset E_N$ e $R = 2^v$, logo, se $\text{int}(P) \neq \emptyset$, temos que antes de alcançar o elipsóide E_N, acharemos uma solução z_i para $Ax \leq b$ (com $i < N$), como N é polinomialmente limitado pelo tamanho do exemplar do problema, encontramos uma solução para o problema em tempo polinomial. Observe que se após N iterações, a convergência não for alcançada, então podemos concluir que não existe solução factível para o problema.

É importante lembrar, que o algoritmo tem esse comportamento, frente as hipóteses assumidas em 1) e 2). Na realidade não acontece assim, pois, existem cálculos complexos, como por exemplo, avaliar as raízes quadradas para obter z_i e D_i.

Os valores dos parâmetros z_i e D_i, em cada iteração, podem ser obtidos como resultado do seguinte teorema.

Teorema 3.2.1 *Seja o elipsóide $E = elip(z, D)$ em $\mathbb{R}^n$, α um vetor linha, $E' = elip(z', D')$ é o único elipsóide de menor volume contendo $E \bigcap \{x \mid \alpha x \leq \alpha z\}$, onde*

$$z' = z - \frac{1}{n+1}\frac{D\alpha^t}{\alpha D\alpha^{t\,1/2}}$$

$$D' = \frac{n^2}{n^2-1}(D - \frac{2}{n+1})\frac{D\alpha^t\alpha D}{\alpha D\alpha^t},$$

$$além \quad \frac{vol(E')}{vol(E)} < e^{\frac{-1}{2n+2}},$$

a prova pode-se ver por exemplo em $[Sch]$.

3.2.3 Complexidade do Algoritmo de Khachian.

O algoritmo é teoricamente polinomial, mas experimentos computacionais mostram que o método não é muito eficiente, este método não é um competidor para o ineficiente, do ponto de vista teórico, algoritmo simplex.

Alguns melhoramentos foram propostos tais como: tomando cortes profundos $[Sh]$ (1979), ou representando as matrizes definidas positivas pela sua descomposição de Cholesky, na procura de reduzir a instabilidade numérica $[GT]$ (1982). Uma das razões para o mau comportamento do algoritmo de Khachian é que o número de iterações depende de uma função dos dados, se T é o máximo valor absoluto dos dados em A e b, então o método resolve o problema de inequações com $O(n^5 \log T)$ operações aritméticas sobre números com $O(n^3 \log T)$ dígitos, totalizando assim $O(n^8 \log T)$ bit-operações.

3.2.4 O Algoritmo de Transformações Projetivas de Karmarkar.

O método Simplex não teve competidor até o ano de 1984, quando N. Karmarkar $[K]$ propôs um novo algoritmo de tempo polinomial para problemas de programação linear, este algoritmo é aplicado para resolver problemas na forma

$$\begin{aligned} &\text{Minimizar} && c^t x \\ &\text{sujeito a:} && \\ & && Ax = 0 \\ & && x \geq 0 \\ & && \mathbb{1}x = 1, \end{aligned} \tag{3.3}$$

onde a matriz $A_{m\times n}$ tem posto m, c e x são vetores n-dimensionais, con $n \geq 2$, e $\mathbb{1} = (1, 1, 1, ..., 1)$ é um vetor linha n-dimensional.

Nós assumimos o seguinte:

1. O ponto $x^0 = (\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n})^t$ é viável para o problema (3.3).
2. O valor ótimo da função objetivo no problema (3.3) é zero.

Notemos que qualquer problema de programação linear pode ser colocado nessa forma [*Ba*].

Frente as hipótese 1) e 2) o problema é viável e limitado, nessas circunstâncias existe uma solução ótima para o problema (3.3).

Seja $x^k = (x_1^k, ..., x_n^k)^t > 0$ um ponto interior, com $k = 0, 1, 2, 3, ...$ e $D_k = \text{diag}(x_1^k, ..., x_n^k)$, consideremos a transformação definida por

$$T : x \longmapsto y$$

$$y = Tx = \frac{D_k^{-1}x}{\mathbb{1}D_k^{-1}x} \text{ ou}$$

$$y_i = \frac{\frac{x_i}{x_i^k}}{\sum_j \frac{x_j}{x_j^k}} \text{ para i=1,2,...n,} \tag{3.4}$$

claramente D_k é uma matriz inversível.

A região viável no problema (3.3) é descrita pela intersecção dos subespaços de dimensão $n - m$, definidos por $Ax = 0$ e o simplex $S_x = \{x \mid \mathbb{1}x = 1,\ x \geq 0\}$ de dimensão $n - 1$, a interseção resultante é alguma região de dimensão $n - m - 1$.

Frente à transformação T em (3.4), qualquer ponto em S_x é transformado num ponto no simplex *(n-1)*-dimensional $S_y = \{y \mid \mathbb{1}y = 1,\ y \geq 0\}$, no qual será mais vantajoso trabalhar. Notamos particularmente que o ponto x^k é transformado no ponto $y^0 = (\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n})^t$, o qual é o centro do simplex S_y.

Para considerar a transformação projetiva inversa, expressamos $x \in S_x$, segundo (3.4), na forma

$$x = D_k y \mathbb{1} D_k^{-1}x, \tag{3.5}$$

como $\mathbb{1}x = 1$, temos que

$$\mathbb{1}D_k y \mathbb{1} D_k^{-1}x = 1,$$

portanto $\mathbb{1}D_k^{-1}x = \frac{1}{\mathbb{1}D_k y}$, substituindo em (3.5) obtemos

$$x = \frac{D_k y}{\mathbb{1}D_k y}. \tag{3.6}$$

Logo, a aplicação (3.6) aplica pontos desde o simplex S_x sobre o simplex S_y e vice-versa. Com a transformação (3.4), o problema (3.3) fica transformado num problema no y-espaço sobre um subconjunto do simplex S_y, isto é

$$\begin{aligned} \text{Minimizar} \quad & \frac{c^t D_k y}{\mathbb{1} D_k y} \\ \text{sujeito a:} \quad & \\ & AD_k y = 0 \\ & \mathbb{1} y = 1 \\ & y \geq 0, \end{aligned} \tag{3.7}$$

este é conhecido como "Problema de Programação Linear Fracionária" (PLF), pela condição 2) e usando (3.6), o valor da função objetivo ótima também é zero, no entanto existe a fração no problema. Nós podemos minimizar equivalentemente (3.7) sem considerar o denominador da fração, pois é positivo e limitado $\forall y \in S_y$, o seguinte problema é equivalente a (3.7).

$$\begin{aligned} \text{Minimizar} \quad & c^t_{novo} y \\ \text{sujeito a:} \quad & \\ & Py = P_0 \\ & y \geq 0, \end{aligned} \tag{3.8}$$

onde $c_{novo} = c^t D_k$, $P = \begin{bmatrix} AD_k \\ \mathbb{1} \end{bmatrix}$ e $P_0 = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}$.

Consideremos agora a bola n-dimensional $B(y^0, r)$, com centro $y^0 = (\frac{1}{n}, ..., \frac{1}{n})^t$ e com raio r, de modo que $\{y \mid \mathbb{1} y = 1\} \bigcap B(y_0, r)$ seja a maior bola *(n-1)*-dimensional com o mesmo raio r e com centro incluido em S_y, então r é o comprimento desde o centro y_0 do simplex para alguma das suas faces, usando este fato, obtemos facilmente que $r = 1/(\sqrt{n(n-1)}$, com isto, nós formulamos o seguinte problema.

$$\begin{aligned} \text{Minimizar} \quad & c^t_{novo} y \\ \text{sujeito a:} \quad & \\ & Py = P_0 \\ & y \geq 0 \\ & y \in B(y_0, \alpha r), \end{aligned}$$

onde $0 < \alpha < 1$, notamos que as variáveis y são estritamente positivas ($y > 0$). Nós podemos otimizar uma restrição do anterior problema resolvendo:

$$\begin{aligned} &\text{Minimizar} && c^t_{novo} y \\ &\text{sujeito a:} && \\ & && Py = P_0 \\ & && (y - y^0)^t (y - y^0) \leq \alpha^2 r^2, \end{aligned} \tag{3.9}$$

uma solução para o problema (3.9) é produzida apenas projetando o vetor c_{novo}, centrado em y^0, sobre a região viável (uma bola), representemos esta projeção por c_{proj}, ela é definida por

$$y^* = y^0 - \alpha r \frac{c_{proj}}{||c_{proj}||}, \tag{3.10}$$

por razões de convergência, a escolha de α é feita como $\alpha = \frac{n-1}{3n}$.

Fazendo algumas manipulações algébricas obtemos

$$c_{proj} = [I - P^t (P P^t)^{-1} P] c^t_{novo},$$

notamos que não é esta a melhor forma de calcular c_{proj}, podemos optar por outros métodos computacionais.

Usando (3.10) podemos calcular x^{k+1} da seguinte forma

$$x^{k+1} = \frac{D_k y^*}{\mathbb{1} D_k y^*},$$

onde $x^{k+1} > 0$ pois $y^* > 0$, tudo isto constitui um passo iterativo.

É possível mostrar que $c_{novo} y^* \leq c_{novo} y^0$, além disso, $cx^k \to 0$ quando $k \to \infty$.

O Algoritmo.

- **passo 0:** k=0, $r = \frac{1}{\sqrt{n(n-1)}}$, $\alpha = \frac{n-1}{3n}$, $x^0 = (\frac{1}{n}, ..., \frac{1}{n})^t$
  e L um limite inferior para o tamanho do exemplar do problema.
- **passo 1:** se $c^t x^k < 2^{-L}$, **parar.**
- **passo 2:** definimos $D_k = \text{diag}(x^k_1, ... x^k_n)$, $y^0 = (\frac{1}{n}, ..., \frac{1}{n})^t$, $P = \begin{bmatrix} AD_k \\ \mathbb{1} \end{bmatrix}$
  e $c_{novo} = c^t D_k$.

- **passo 3:** calcular $y_{novo} = y^0 - \alpha r \frac{c_{proj}}{\|c_{proj}\|}$, onde

$$c_{proj} = [I - P^t(PP^t)^{-1}P]c_{novo}.$$

- **passo 4:** obtemos $x^{k+1} = \frac{D_k y_{novo}}{\mathbb{1} D_k y_{novo}}$.

- **passo 5:** k=k+1.

- **passo 6: ir ao passo 1.**

Seja L o tamanho de um exemplar do problema, frente a essas circunstâncias, quando fazemos $10nL$ iterações, o algoritmo produz uma solução factível, na qual, o valor objetivo é menor que 2^{-L}, com um esforço total de complexidade polinomial da $O(n^{3.5}L)$. Acontecido isso, um ponto extremo ótimo no problema principal (3.3) pode ser conseguido em base a um procedimento de refinamento, também de tempo polinomial, $O(m^2n)$, denominado "Esquema de Purificação."

Cabe ressaltar que os métodos de pontos interiores, como os que estamos tratando, convergem geralmente a uma face do politopo. O seguinte algoritmo, que complementa ao de Karmarkar, corrige o resultado, daí o seu nome.

Algoritmo de Purificação.

- Uma vez obtido $cx^k < 2^{-L}$:

- Se n restrições linearmente independentes são ativas em x^k, então já temos uma SBV ótima.

- Caso contrário, existe uma direção $d \neq 0$ no espaço nulo das restrições ativas tal que:

  se $c^t d < 0$, nos movemos na direção d.

  se $c^t d \geq 0$, nos movemos na direção $-d$. até que alguma restrição seja violada, esse procedimento é repetido até obter uma SBV x^*, isto acontece quando a região viável é limitada.

Claramente $c^t x^* \leq c^t x^k < 2^{-L}$.

3.2.5 O Algoritmo Afim Escala de Barnes.

Seus origens deve-se a Dikin (1967), mas os trabalhos mais recentes foram propostos independentemente por Barnes [*B*] (1986) e Vanderbei [*V*] (1986).

O algoritmo é uma extensão e tem algumas vantagens sobre o algoritmo original de Karmarkar (1984), tais como, o problema não precisa estar na forma (3.3), e pode-se aplicar diretamente a um problema de programação linear na forma padrão (3.1). O algoritmo produz uma sequência monótona decrescente de valores da função objetivo, o valor ótimo deste não precisa ser conhecido previamente, como no algoritmo de Karmarkar. Por outro lado, na ausência de degeneração, o algoritmo converge a uma solução básica viável ótima (um ponto extremo) com as variáveis não básicas convergendo monotonicamente para zero, o qual faz possível conhecer uma base ótima com antecedência.

Consideremos o problema de programação linear na forma padrão (3.1), onde posto$(A) = m$ (posto completo), assumamos também que o nosso problema não tem SBV's degeneradas. Denotemos o dual do problema (3.1) por

$$\begin{aligned} &\text{Maximizar} \quad b^t\lambda \\ &\text{sujeito a:} \\ &\qquad A^t\lambda \leq c, \quad \lambda \text{ livre.} \end{aligned} \tag{3.11}$$

Suponhamos que (3.11) não tem soluções viáveis degeneradas, quanto estas hipótese, vemos que é possível restringir as restrições das variáveis $x \geq 0$ como segue:

Dado um n-vetor $y > 0$, solução viável de (3.1), definimos

$$D = \text{diag}(y_1, y_2, ..., y_n)$$

e $0 < R < 1$, de acordo com isto, formamos o seguinte elipsóide centrado em y

$$E = \{x \mid (x-y)^t Q^{-1}(x-y) \leq R^2\}, \tag{3.12}$$

onde $Q = D^t D$. Claramente D é não singular, portanto Q, além disso Q é definida positiva. note-se que se $x \in E$, então $x > 0$, caso contrário existiria um $x_j \leq 0$ (para algum j), onde x_j é uma componente de x, tal que

$$\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{y_i^2} \geq \frac{(x_j - y_j)^2}{y_j^2} \geq 1 > R^2$$

o qual representa uma contradição.

Consideremos agora o novo problema

$$\begin{gathered} \text{Minimizar } c^t x \\ \text{sujeito a: } Ax = b \\ x \in E, \end{gathered} \tag{3.13}$$

notemos que a restrição de não negatividade ($x \geq 0$) podemos substituí-la por $x \in E$, as quais são mais vantajosas.

Usando o vetor m-dimensional dos multiplicadores Lagrangianos, i.e, o vetor das variáveis duais do problema (3.1), para x satisfazendo (3.13) temos

$$\begin{aligned} c^t y - c^t x &= [c - A^t\lambda]^t(y - x) = [D(c - A^t\lambda)]^t D^{-1}(x - y) \\ &\leq ||D(c - A^t\lambda)||\ ||D^{-1}(x - y)|| \\ &\leq ||D(c - A^t\lambda)||R. \end{aligned} \tag{3.14}$$

Com a finalidade de obter uma estimativa do comprimento do incremento entre $c^t x$ e $c^t y$, vemos que a igualdade em (3.14) tem validade se

$$D(c - A^t\lambda) = \gamma D^{-1}(x - y) \tag{3.15}$$

o qual é a condição de paralelismo, onde γ é alguma constante, tomando $||D^{-1}(x - y)|| = R$ e desde (3.12) temos que $\gamma = \frac{||D(c - A^t\lambda)||}{R}$, substituindo em (3.15) temos

$$x = y - \frac{RD^2(c - A^t\lambda)}{||D(c - A^t\lambda)||}, \tag{3.16}$$

se multiplicamos por A em ambos os lados da equação (3.16), lembrando que $Ax = Ay = b$, obtemos

$$AD^2(c - A^t\lambda) = 0$$

de onde

$$\lambda = (AD^2A^t)^{-1}AD^2c, \tag{3.17}$$

portanto

$$c^t x \geq c^t y - R||D(c - A^t\lambda)||. \tag{3.18}$$

Vemos que para x verificando (3.13), o valor mínimo para a função objetivo é limitada inferiormente pela direita da inequação (3.18), a igualdade se cumpre quando x e λ são dados por (3.16) e (3.17), respectivamente.

Cabe ressaltar que não é preciso resolver o problema (3.13) explicitamente (por algum algoritmo conhecido), contrariamente pode ser resolvido simplesmente avaliando (3.16) de acordo com (3.17) diretamente. Esse procedimento constitui um passo do seguinte algoritmo.

O Algoritmo.

Seja $x^0 > 0$ um ponto interior inicial, com $Ax^0 = b$, se x^k é conhecido:

- definimos $D^k = diag(x_1^k, x_2^k, ..., x_n^k)$
- definimos x^{k+1} como

$$x^{k+1} = x^k - \frac{R(D^k)^2(c - A^t\lambda^k)}{||D^k(c - A^t\lambda^k)||}$$

$$\lambda^k = (A(D^k)^2A^t)^{-1}A(D^k)^2c.$$

A convergência e propriedades são demostradas por Barnes [B], no mesmo artigo. Nós só enunciaremos alguns teoremas que garantem a convergência.

Teorema 3.2.2 *Se (3.1) tem uma solução limitada, a sequência $\{x^k\}$ obtida pelo algoritmo converge à solução de (3.1) e é um ponto extremo (vértice).*

Teorema 3.2.3 *Seja x^* a solução de (3.1), a sequência $\{x^k\}$ gerada pelo algoritmo satisfaz:*

$$c^tx^{k+1} - c^tx^* \leq [1 - \frac{R}{\sqrt{n - m + \xi_k}}](c^tx^k - c^tx^*)$$

onde $\{\xi_k\}$ é uma sequência de números positivos que converge para zero quando $k \to \infty$.

Um inconveniente na maioria dos métodos de pontos interiores é de obter um ponto inicial factível, nós consideraremos algumas sugestões importantes para inicializar o algoritmo.

1. Para obter x^0, adicionamos uma nova coluna a_{n+1} à matriz A definida por

$$a_{n+1} = b - \sum_{i=1}^{n} a_i,$$

se definimos um vetor $(n+1)$-dimensional por $\overline{x} = (1, 1, 1, ..., 1)^t$, então ele verifica

$$\overline{A}\overline{x} = \sum_{i=1}^{n+1} a_i x_i = b \text{ e } \quad x \geq 0.$$

o qual é válido para inicializar uma extensão do problema principal, adicionamos também o correspondente coeficiente $c_{n+1} > 0$ grande, no vetor custo c. Notamos que isto é feito similarmente ao caso quando trabalhamos com o método "M-grande" no algoritmo simplex. Assim, uma solução para o problema (3.13) terá $x_{n+1} = 0$, onde o vetor $(x_1, x_2, x_3, ..., x_n)^t$ representaria a verdadeira solução.

2. É possível variar o valor de R para acelerar a convergência, pois, $c^t x$ decresce quando R se incrementa.

3. É possível obter um critério de parada para o algoritmo, isto pode ser feito do seguinte modo: podemos iterar o algoritmo até que m dos custos reduzidos em valor absoluto $|c_i - a_i^t \lambda_k|$ sejam pequenos, e, os $n - m$ restantes sejam positivos. Logo, uma solução básica pode ser conhecida com antecedência.

Capítulo 4

Centros de Politopos.

Os métodos para achar centros são aplicados, num caso geral, sobre conjuntos convexos limitados num espaço n-dimensional, mas o caso de nosso maior interesse está relacionado com politopos. Existem vários critérios definindo centros em politopos e conjuntos convexos, nós fazemos aqui um resumo de alguns deles, os quais consideramos relacionados, de certa forma, com o trabalho desenvolvido no capítulo seguinte. O motivo principal para estudar centros se fundamenta no seguinte: um problema de programação linear se reduz polinomialmente a um problema de achar um centro. Além disso, todo algoritmo de pontos interiores, que é provado ser polinomial, tem um mecanismo explícito ou implícito de centragem. Mas a noção de centro de um politopo não é trivial, pois dado um politopo geral de programação linear, existem várias definições de centros. Centro de Helly, de Fritz John, centro analítico são alguns exemplos de centros de politopos, sendo este último o mais usado para definir centros de politopos para programação linear. Neste capítulo, nós mostramos que se um desses centros nos é dado de graça, então temos um algoritmo de tempo polinomial para resolver problemas de programação linear.

Um politopo é um poliedro limitado, o nosso interesse são politopos definidos pelo conjunto de soluções viáveis de um problema de programação linear, onde as inequações lineares são da forma: $\sum_{j=1}^{n} a_{ij}x_j \leq b_i$, com $i = 1, 2, 3, ..., m$.

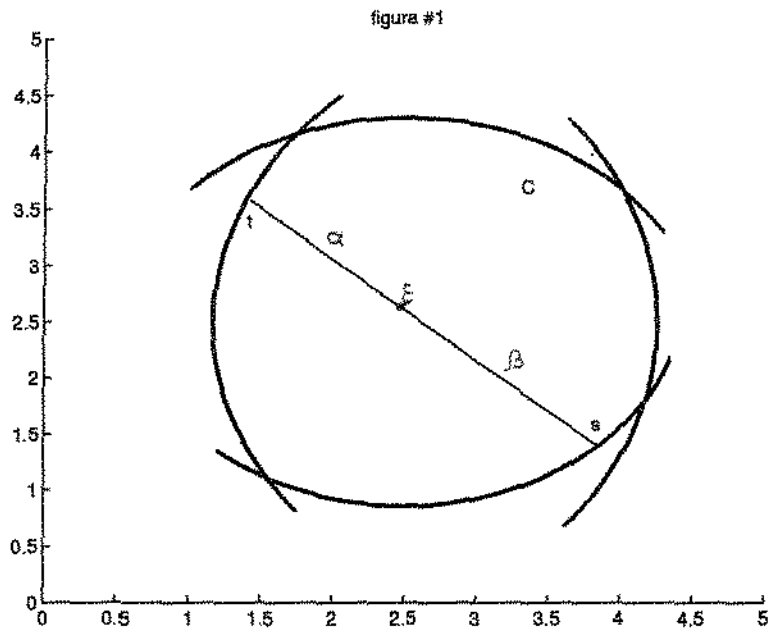
Generalizando as idéias de centros de gravidade, ou um centróide num triângulo em $\mathbb{R}^2$, podemos estender esse conceito para corpos convexos de maior dimensão.

Definição 4.0.1 *Um corpo convexo é um conjunto convexo S com $int(S) \neq \emptyset$.*

4.1 Centro de Helly.

Definição 4.1.1 *Seja C um corpo convexo em $\mathbb{R}^n$, $\xi \in C$ é denominado um centro de Helly (H-centro), se ξ divide qualquer corda $[s,t] \subset C$, passando por ξ, em duas partes de comprimentos α e β, tais que*

$$\frac{1}{n+1} \leq \frac{\alpha}{\alpha+\beta} \leq \frac{n}{n+1}. \tag{4.1}$$



O nome tem relação com Eduard Helly (1913) que propôs o teorema que faz possível estabelecer a existência de tal centro para corpos convexos em $\mathbb{R}^n$: seja K uma família de pelo menos $n+1$ conjuntos convexos em $\mathbb{R}^n$, onde K é finito ou compacto, se cada $n+1$ membros de K tem um ponto em comum, então existe um ponto em comum para todos os membros de K.

Teorema 4.1.1 *Se C é um corpo convexo em $\mathbb{R}^n$, então existe um ponto $z \in C$, tal que, para cada corda $[s,t]$ de C, passando por z, se cumpre*

$$\frac{1}{n+1} \leq \frac{||z-s||}{||s-t||} \leq \frac{n}{n+1}. \tag{4.2}$$

Notemos que z verifica a inequação (4.1) se tomamos $||z-s||$ e $||s-t||$ como os comprimentos induzidos pela norma euclidiana $||\ ||_2$, os quais correspondem aos valores α e β, respectivamente.

Em geral, um corpo convexo pode ter um conjunto infinito de H-centros, outro resultado estabelece que um Simplex n-dimensional tem um único H-centro. O teorema seguinte garante a infinidade de H-centros em corpos convexos de $\mathbb{R}^n$, o mesmo, usa um elipsóide para obter tal resultado.

Teorema 4.1.2 *Seja E um elipsóide n-dimensional, definido por*

$$E = \{x \mid (x-\xi)^t Q(x-\xi) = 1\},$$

seja $\overline{E}$ outro elipsóide, centrado também em ξ e definido por

$$\overline{E} = \{x \mid (x-\xi)^t Q(x-\xi) = \frac{(n-1)^2}{(n+1)^2}\},$$

então qualquer $x \in \overline{E}$ é um H-centro.

Na subseção (4.2.1), falaremos de como pode-se usar H-centros na resolução de problemas de programação linear, este último fato está relacionado diretamente com o seguinte teorema.

Teorema 4.1.3 *Seja C um corpo convexo em $\mathbb{R}^n$, ξ um H-centro de C, seja o elipsóide de menor volume contendo C, centrado em ξ e definido por $(x-\xi)^t R(x-\xi) = 1$, então o elipsóide definido por $(x-\xi)^t R(x-\xi) = \frac{1}{n^3}$ está contido completamente em C.*

Uma outra noção de centro, centro de John, o qual considera elipsóides incluindo um politopo é definida a partir disso.

4.2 Centro de John.

Definição 4.2.1 *Dado um politopo P, seja E o elipsóide de menor volume contendo P, o centro de E é chamado um centro de Fritz John (J-centro).*

É possível usar alguns algoritmos para calcular J-centros de um politopo, por exemplo, o algoritmo de Khachian. Mas, como vimos no capítulo anterior, este método não é eficiente, pois ainda sendo de tempo polinomial, na prática sua performance não é boa. O nome de centro John tem relação com o seguinte teorema, proposto por Fritz John, o qual garante a unicidade de E.

Teorema 4.2.1 *(Fritz John-1947). Seja P um politopo em $\mathbb{R}^n$, e E o elipsóide de menor volume contendo P, então E é único, além disso, se reduzirmos E por um fator n, i.e, $\overline{E} = \frac{1}{n}E$, obtemos um elipsóide contido em P.*

Claramente, a diferença entre um H-centro e um J-centro é a unicidade, aparentemente, calcular um H-centro pode ser mais fácil que calcular um J-centro. Um outro resultado mostra que um J-centro é sempre um H-centro, pois, se considerarmos uma $corda[s,t]$ passando por ξ, com pontos finais s e t na fronteira de C (um corpo convexo), obtemos

$$\frac{1}{n+1} \leq \frac{||s-\xi||}{||s-t||} \leq \frac{n}{n+1}, \tag{4.3}$$

se estendemos a $corda[s,t]$ a uma $corda[s',t']$, com pontos finais na fronteira de E (o elipsóide exterior), e reduzimos a $corda[s,t]$ a uma $corda[s'',t'']$, com pontos finais na fronteira de $\overline{E}$ (o elipsóide interior), nós temos

$$\begin{aligned} \frac{||s-\xi||}{||s-t||} &= \frac{||s-\xi||}{||s-\xi|| + ||\xi - t||} \\ &\geq \frac{||s''-\xi||}{||s''-\xi|| + ||\xi - t||} \\ &\geq \frac{||s''-\xi||}{||s''-\xi|| + ||\xi - t'||}, \end{aligned} \tag{4.4}$$

como asumimos que $\overline{E} = \frac{1}{n}E$, temos

$$||\xi - t'|| = ||\xi - s'|| = n||s''-\xi||,$$

logo

$$\frac{||s-\xi||}{||s-t||} \geq \frac{||s''-\xi||}{||s''-\xi||+n||s''-\xi||} = \frac{1}{n},$$

agora, como

$$1 = \frac{||t-s||}{||t-s||} = \frac{||t-\xi||+||s-\xi||}{||t-s||},$$

temos

$$\begin{aligned} \frac{||t-\xi||}{||t-s||} &= 1 - \frac{||s-\xi||}{||s-t||} \\ &\leq 1 - \frac{1}{n+1} \\ &= \frac{n}{n+1}. \end{aligned} \tag{4.5}$$

Mudando s por t é possível obter a inequação (4.1), com $||t-\xi|| = \alpha$, $||\xi - s|| = \beta$ e $||s-t|| = \alpha + \beta$ o que prova o afirmado.

Notamos também que um Simplex n-dimensional, i.e, o suporte convexo de $n+1$ pontos $\{x_1, x_2, ..., x_{n+1}\}$, onde $\{x_2 - x_1, ..., x_{n+1} - x_1\}$ é linearmente independente, tem só um H-centro, o qual também é um J-centro.

Consideremos o problema de programação linear na forma canônica

$$\begin{aligned} \text{Minimizar} \quad & c^t x \\ \text{sujeito a:} \quad & \\ & Ax \leq b \\ & x \geq 0, \end{aligned} \tag{4.6}$$

onde c, x são vetores n-dimensionais, b um vetor m-dimensional e a matriz $A_{m\times n}$.

Seja o politopo definido pelas restrições $Ax \leq b$ e $x \geq 0$ em (4.6), vemos que é possível usar, por exemplo, o algoritmo de escalamento afim para tentar resolver o problema (4.6). Esse algoritmo usa um mecanismo de centragem implicitamente dado, minimizando a função objetivo a cada passo, sobre uma região determinada pela intersecção de um elipsóide definido por

$$\{x \mid (x-y)^t Q(x-y) \leq R^2, \quad 0 < R < 1\}$$

e o conjunto definido por $\{x \mid Ax \leq b\}$, onde $y > 0$ é um ponto interior em (4.6). Nós tentamos aclarar isto fazendo um resumo de um método teórico de tempo polinomial, o qual usa elipsóides para construir uma sequência $\{x_k\}$ convergindo para x^*, a solução ótima para (4.6).

4.2.1 Procedimento Para Resolver o Problema de Programação Linear Usando J-centros.

Consideremos o problema de programação linear na forma (4.6).

- Seja E^0, o elipsóide de menor volume contendo o politopo definido por

$$S^0 = S = \{x \mid Ax \leq b, \ x \geq 0\}$$

e x^0 o seu centro (J-centro).

- Seja x^1 uma solução do problema

$$\begin{aligned} &\text{Minimizar} \quad c^t x \\ &\text{sujeito a:} \\ &\qquad x \quad \in E_r^0, \\ &\text{onde} \\ &\qquad E_r^0 \ = \frac{1}{n} E^0. \end{aligned}$$

- Seja E^1 o elipsóide de mínimo volume contendo o politopo

$$S^1 = S^0 \bigcap \{x \mid c^t x \leq c^t x^1\} \text{ e } x^2 \text{ seu centro.}$$

- Seja x^3 a solução do problema

$$\begin{aligned} &\text{minimizar } c^t x \\ &\text{sujeito a: } \ c \in E_r^1, \\ &\text{onde } E_r^1 = \frac{1}{n} E^1. \end{aligned}$$

- Desse modo continuamos... definindo o politopo

$$S^2 = S^1 \bigcap \{x \mid c^t x \leq c^t x^3\}.$$

A análise da convergência deste procedimento está feita em $[Mo]$, onde se mostra, tendo algumas considerações, que o procedimento toma um tempo polinomial para calcular uma aproximação à solução.

4.3 Centros Analíticos.

Calcular um H-centros ou J-centro tem um custo, no entanto podemos estender este conceito para definir novos centros, existem outras noções de centros, por exemplo: centro paramétrico [$RMF2$], W-centro [$RMF1$], além do centro analítico [So] que introduziremos agora, eles são definidos geralmente sobre politopos e são aplicados frequentemente nos métodos de pontos interiores para resolver o problema de programação linear.

O problema do centro analítico foi introduzido inicialmente por G. Sonnevend [So].

Definição 4.3.1 *Dado um sistema de m inequações lineares em $\mathbb{R}^n$ da forma $Ax \leq b$, e equações da forma $Mx = g$, isto é denominado o sistema (A,b,M,g), o centro analítico para este sistema é a solução ótima x^* do problema*

$$\begin{array}{lll} \textit{Maximize} & \sum_{i=1}^{m} log(b_i - A_i x) & \\ \textit{sujeito a :} & & \\ & Ax & \leq b \\ & Mx & = g. \end{array} \tag{4.7}$$

O fato de x ser uma variável livre (o problema não inclui $x \geq 0$) nos permite eliminar variáveis através do sistema de equações $Mx = g$. Para ver isso assumimos, sem perda de generalidade, que M é uma matriz $k \times n$ e com posto$(M) = k$, particionando: $M = [B \;\; N]$, onde B é uma matriz $k \times k$ não singular, $A = [C \;\; D]$ e $x = (y \;\; z)$, nós eliminamos as variáveis y_i de y e obtemos o problema equivalente

$$\text{Maximizar} \sum_{i=1}^{m} \log(v_i - B_i z) \tag{4.8}$$

$$\text{sujeito a: } Bz \leq v,$$

onde $B = D - CB^{-1}N$ e $v = b - CB^{-1}g$, logo, z^* resolve o problema (4.8), se e somente se, $x^* = (y^* \;\; z^*)$ resolve (4.7).

Por comodidade, podemos assumir que a equação $Mx = g$ em (4.7) não está presente, logo, a seguinte versão é equivalente ao problema (4.7)

$$\text{Maximizar} \sum_{i=1}^{m} \log(b_i - A_i x) \tag{4.9}$$
$$\text{sujeito a: } Ax \leq b$$

onde b é um vetor m-dimensional, x um vetor n-dimensional e $A_{m\times n}$ é uma matriz.

Quando o poliedro S, determinado pelos hiperplanos definindo o sistema em (4.9), é limitado (politopo) e com $\text{int}(S) \neq \emptyset$, então pode-se verificar a existência de uma única solução, deste modo, o centro analítico está bem definido, uma propriedade que caracteriza a um centro analítico é a seguinte.

Teorema 4.3.1 *O centro analítico ξ miniminiza a soma ponderada dos quadrados das distâncias desde um ponto em S aos hiperplanos que definem S, onde os pesos são dados por*

$$w_i = \frac{1}{d_i^2(\xi)} \quad i=1,2,3,...,m,$$

com $d_i(x) = \frac{b_i - \langle a_i, x\rangle}{||a_i||}$.

Analogamente como usamos H-centros e J-centros para resolver um problema de programação linear, podemos usar os centros analíticos para este fim.

Seja ξ um centro analítico, consideremos o seguinte elipsóide

$$(x-\xi)^t Q(x-\xi) \leq 1, \tag{4.10}$$

onde $Q = \sum_{i=1}^{n} \frac{a_i a_i^t}{(b_i - a_i^t \xi)^2}$ é uma matriz simétrica definida positiva.

Para x satisfazendo (4.10) temos que

$$d_i(x) = \frac{b_i - \langle a_i, x\rangle}{||a_i||} \geq 0, \quad \text{com} \quad i = 1, 2, ..., m,$$

este último fato nos permite assegurar que o elipsóide está contido em S. Agora, podemos expandir o elipsóide em (4.10), com um fator m, para obter o seguinte resultado.

Teorema 4.3.2 *Se ξ é o centro analítico de S, então, o elipsóide definido por $(x-\xi)^t Q(x-\xi) \leq m^2$ contém S.*

Agora, como na subseção (4.2.1), podemos construir um procedimento para resolver o problema de programação linear usando-se centros analíticos.

No entanto, os J-centros e H-centros são difíceis de calcular, a razão pela qual são poucos usados, os centros analíticos e suas variações (extensões) são de mais interesse e são usados com mais frequência.

4.4 O W-Centro de um Sistema Poliedral.

Segundo vimos no problema do centro analítico, tal centro maximizava a função barreira logaritmo conformada pelas inequações que definem o poliedro, um outro problema se define quando pesos w_i são atribuídos a cada uma destas inequações.

O W-centro tem propriedades semelhantes aos outros centros pois, ele constitui uma generalização de um centro analítico.

Definição 4.4.1 *Um W-centro de um sistema poliedral*

$$P = \{x \in \mathbb{R}^n \mid Ax \leq b,\ Mx = g\},$$

é definido como a solução do seguinte problema:

$$\begin{array}{ll} \textit{Maximizar} & \sum_{i=1}^{m} w_i \log s_i \\ \textit{sujeito a :} & \\ & \begin{array}{rl} Ax + s & = b \\ Mx & = g \\ s & \geq 0, \end{array} \end{array} \tag{4.11}$$

onde $s_i = b_i - A_i x$.

O problema (4.11), como vemos, é uma generalização para (4.7) e pode ser usado em algoritmos de pontos interiores para resolver o problema de programação linear.

Assumindo que P (def. 3.4.1) é limitado e $int(P) \neq \emptyset$, o problema (4.11) tem uma solução única. Se assumimos que o vetor $w = (w_1, w_2, ...w_m)$ é normalizado, isto é $w_1 + w_2 + ... + w_m = 1$, e $w_o = \min\{w_i \mid i = 1, ..., m\}$, claramente temos que $w_o \leq \frac{1}{m}$. Agora, é possível usar um W-centro para resolver um problema de programação linear, isto deve-se à existência de elipsóides inscritos e circunscritos com respeito ao poliedro P e centrados no W-centro, como afirma o seguinte teorema.

Teorema 4.4.1 *Sejam* $P = \{e \in I\!R^n \mid Ax \leq b,\ Mx = g\}$, x^o *o W-centro de* P, *e os elipsóides*

$$E_{int} = \{x \in I\!R^n \mid (x - x^o)^t A^t S_o^{-1} W S_o^{-1} A(x - x^o) \leq \frac{w_o}{1 - w_o}\}$$

e

$$E_{ext} = \{x \in I\!R^n \mid Mx = g,\ (x - x^o)^t A^t S_o^{-1} W S_o^{-1} A(x - x_o) \leq \frac{1 - w_o}{w_o}\}$$

então $E_{int} \subset P \subset E_{ext}$. *Onde* W *é a matriz diagonal contendo os pesos* w_i, S_o^{-1} *é a matriz diagonal de inversas das variáveis* s_i.

A seguinte relação também se verifica: $E_{int} = \frac{w_o}{1-w_o} E_{ext}$, isto é, o elipsóide E_{int} é uma cópia escalada do elipsóide E_{ext} com um fator de $\frac{w_o}{1-w_o}$. Se $w = \frac{1}{m}$ então $w_o = \frac{1}{m}$ e o fator de escalamento seria $\frac{1}{m-1}$.

Capítulo 5

Um Método de Pontos Interiores Usando Bolas.

Neste capítulo, nós desenvolvemos um procedimento que usa a trajetória descrita pelos centros das maiores bolas inscritas em politopos. Iterativamente, usando-se uma modificação de projeções ortogonais sucessivas, nós calculamos o centro da maior bola inscrita no politopo, após minimizarmos parcialmente a função objetivo nesta bola, nós inserimos o hiperplano da função objetivo reduzindo assim o politopo. O procedimento é repetido até que o mínimo do problema original seja alcançado.

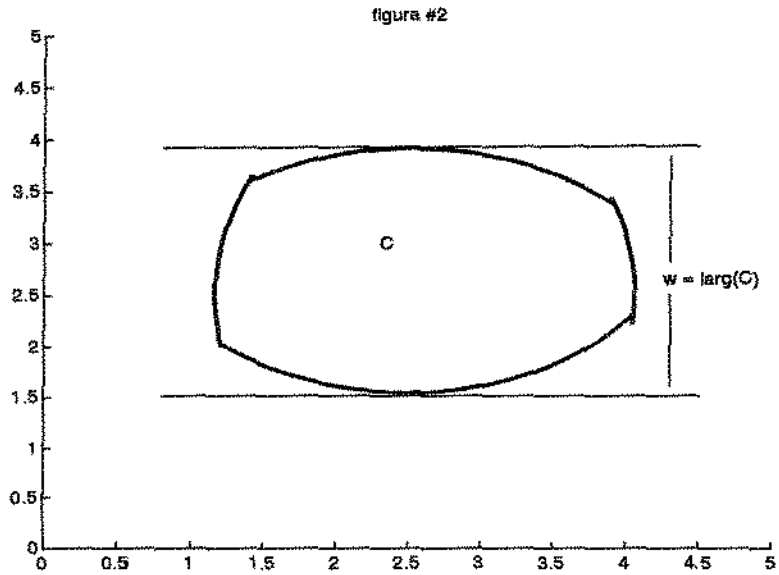
Consideremos o problema de programação linear, no qual, as restrições de não negatividade ($x \geq 0$) estão incluidas no sistema de inequações do problema

$$\begin{aligned} &\text{Minimizar} \quad c^t x \\ &\text{sujeito a:} \\ &\qquad Ax \leq b, \end{aligned} \tag{5.1}$$

onde $m > n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ e $c, x \in \mathbb{R}^n$. Suponhamos que o conjunto de soluções factíveis em (5.1) seja limitado e com interior não vazio, denotemos tal conjunto pelo politopo $\mathbb{P} = \{x \mid Ax \leq b\}$ e o seu interior por $\text{int}(\mathbb{P}) = \{x \mid Ax < b\}$. Neste capítulo, nós assumimos também que $||.|| = ||.||_2$ e lembramos que $\langle\ ,\ \rangle$ denota o produto interno de dois vetores.

5.1 Resolvendo o Problema de Programação Linear.

Definição 5.1.1 *Seja C um corpo convexo em $\mathbb{R}^n$, chamamos a largura de C, denotada por* $\text{larg}(C)$, *à menor distância entre dois planos paralelos distintos suportando C.*



Teorema 5.1.1 *(Bernard Bertrand) Seja C um conjunto convexo em $\mathbb{R}^n$ tal que* $\text{larg}(C) = w$, *então C contém uma bola de raio* $\frac{w}{n+1}$.

Em (5.1), se $x \in \text{int}(I\!P)$, então uma bola $B(x, r)$, com centro x e raio r, estará incluida em $I\!P$ se

$$r \leq \frac{b_i - \sum_{j=1}^{n} a_{ij}x_j}{\sqrt{\sum_{j=1}^{n} a_{ij}^2}} = \frac{b_i - \langle a_i, x\rangle}{||a_i||} \quad i = 1, 2, ..., m, \tag{5.2}$$

onde a_i é a i-ésima linha da matriz A, e $d_i(x) = \frac{b_i - \langle a_i, x\rangle}{||a_i||}$ é a distância de x ao i-hiperplano que define o politopo $I\!P$.

A bola $B(x,r) \subset I\!P$ será a maior possível, se (x,r) é uma solução do problema de programação linear

$$\begin{aligned} &\text{Maximizar} && r \\ &\text{sujeito a:} \\ &&\langle a_i, x\rangle + ||a_i||r &\leq b_i \quad i = 1,2,...,m. \end{aligned} \tag{5.3}$$

Consideremos um ponto inicial $x^0 \in \text{int}(I\!P)$, uma estimativa da solução ótima x^* para o problema (5.1), seja $L^0 > c^t x^0$ um limite superior para a função objetivo em x^0. Construimos um novo politopo $I\!P^0$ definido por

$$\begin{aligned} Ax &\leq b \\ c^t x &\leq L^0, \end{aligned} \tag{5.4}$$

onde claramente $x^0 \in I\!P^0$.

Agora, nós podemos achar a maior bola $B(x,r) \subset I\!P^0$, resolvendo o problema de programação linear seguinte

$$\begin{aligned} &\text{Maximizar} && r \\ &\text{sujeito a:} \\ &&\langle a_i, x\rangle + ||a_i||r &\leq b_i \quad i = 1,2,...,m \\ &&c^t x + ||c||r &\leq L^0 \end{aligned} \tag{5.5}$$

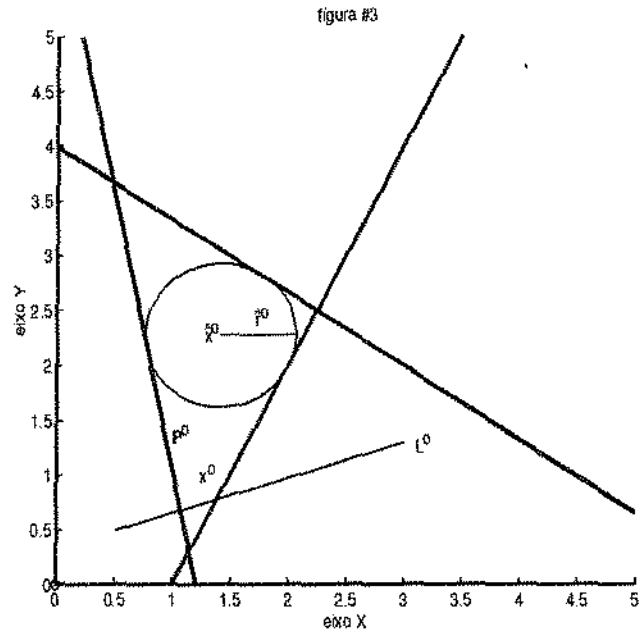
e obtemos uma solução ótima $(\hat{x}^0, \hat{r}^0)$ (ver *figura* 3). Logo definimos para $0 < \lambda \leq 1$, fixo,

$$x^1 = \hat{x}^0 - \lambda \hat{r}^0 \frac{c}{||c||}$$

e

$$L^1 = c^t \hat{x}^0. \tag{5.6}$$

Isto representa uma iteração do nosso procedimento.



Para a seguinte iteração, definimos o politopo reduzido $I\!P^1$, similar a (5.4), por

$$Ax \leq b$$
$$c^t x \leq L^1,$$

notemos que x^1 constitui agora o novo ponto inicial e L^1 o novo limite superior. Posteriormente, achamos a maior bola inscrita em $I\!P^1$ resolvendo o problema (5.5) substituindo x^0 por x^1, e L^0 por L^1.

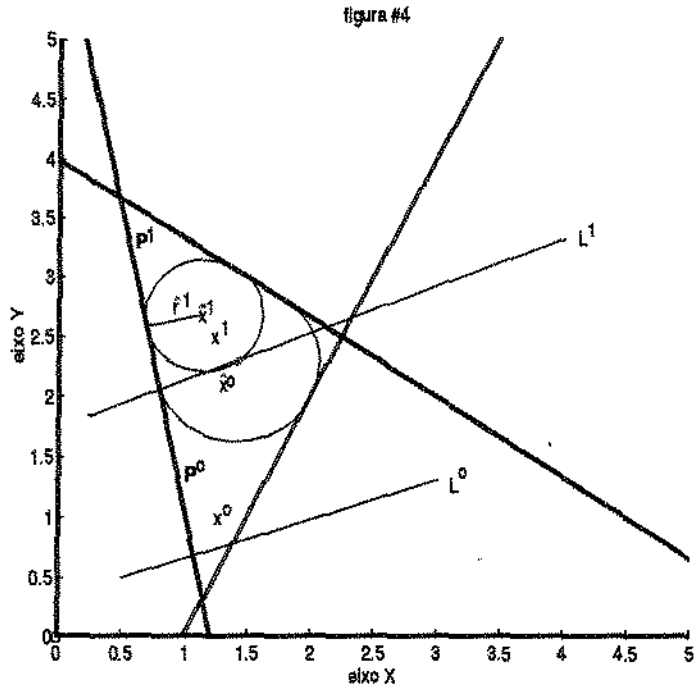
Seja $(\hat{x}^1, \hat{r}^1)$ uma solução ótima. Analogamente como em (5.6) obtemos $x^2 = \hat{x}^1 - \lambda \hat{r}^1 \frac{c}{||c||}$, e $L^2 = c^t \hat{x}^1$. Deste modo, geramos uma sequência $\{x^k\}$, e o procedimento se repete até que o mínimo do problema original (5.1) seja alcançado, ver *figura* 4, para o caso quando $\lambda = \frac{1}{2}$.

Seja $w^0 = \text{larg}(I\!P^0)$, claramente $w^0 \geq 2\hat{r}^0$ pois $B(\hat{x}^0, \hat{r}^0) \subset I\!P^0$, notamos também que

$$\hat{r}^1 \leq \frac{c^t \hat{x}^0 - c^t \hat{x}^1}{||c||}$$

de onde

$$c^t \hat{x}^1 \leq c^t \hat{x}^0 - \hat{r}^1 ||c||,$$



pelo teorema (5.1.1), $\frac{w^1}{n+1} \leq \hat{r}^1$, logo

$$c^t\hat{x}^1 \leq c^t\hat{x}^0 - \frac{w^1}{n+1}||c||,$$

e a mesma idéia se aplica para a iteração 2

$$c^t\hat{x}^2 \leq c^t\hat{x}^1 - \frac{w^2}{n+1}||c||,$$

$$c^t\hat{x}^2 \leq c^t\hat{x}^0 - \frac{w^1}{n+1}||c|| - \frac{w^2}{n+1}||c||,$$

logo, para $k \geq 1$ temos

$$c^t\hat{x}^k \leq c^t\hat{x}^{k-1} - \frac{w^k||c||}{n+1},$$

onde $w^k = \text{larg}(I\!P^k)$, em resumo

$$c^t\hat{x}^k \leq c^t\hat{x}^{k-2} - \frac{w^{k-1}||c||}{n+1} - \frac{w^k||c||}{n+1}$$

$$\leq ... \leq c^t\hat{x}^0 - \frac{||c||}{n+1}\sum_{j=1}^{k} w^j, \tag{5.7}$$

desde que $c^t x^*$ é finito, nós temos

$$0 \leq \frac{||c||}{n+1}\sum_{j=1}^{k} w^j \leq c^t\hat{x}^0 - c^t\hat{x}^k \leq c^t\hat{x}^0 - c^t x^*, \tag{5.8}$$

de onde a série $\sum_{j=1}^{\infty} w^j$ é limitada, e portanto convergente, consequentemente

$$\lim_{j\to\infty} w^j = 0,$$

o qual significa que $\text{vol}(I\!P^k) \to 0$.

A seguinte relação é consequência direta do teorema (5.1.1), $w^k \to 0$ se e somente se $\hat{r}^k \to 0$.

Se s restrições ativas definem o ponto $x = \lim_{k\to\infty} x^k$ $(s = n)$, então x é uma solução extrema ótima (degenerada se $s > n$). Caso contrário, existem soluções alternativas onde, uma ou mais restrições lineares no problema original são ativas.

5.1.1 O Algoritmo.

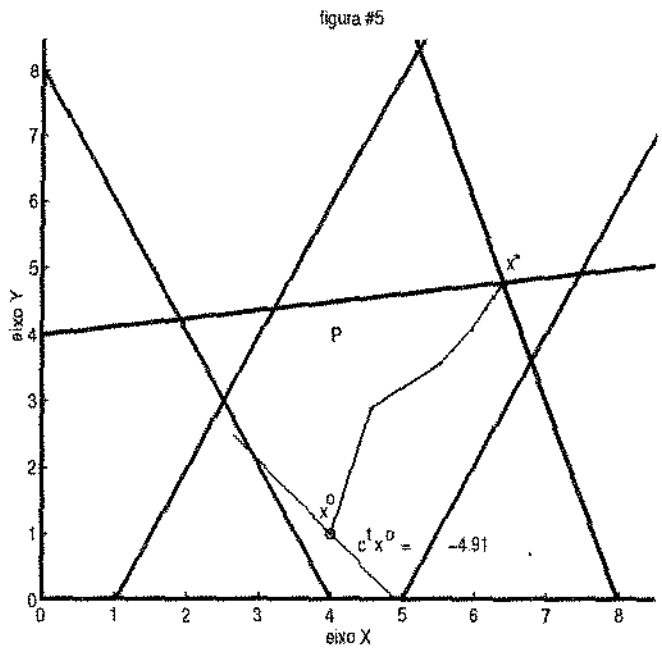
O algoritmo para resolver o problema de programação linear que descrevemos aqui pressupõe que os centros das maiores bolas são conhecidos; calcular tais centros não é trivial. No entanto, na próxima seção, nós desenvolveremos um procedimento no qual, realizando projeções ortogonais e reduzindo sucessivamente o politopo achamos uma aproximação a este centro.

A escolha dos parâmetros λ, ϵ, assim como o critério da escolha do ponto no qual o hiperplano definido pela função custo é inserido, deve-se inicialmente aos resultados obtidos fazendo experiências computacionais com diversos problemas gerados e alguns problemas do NETLIB. O algoritmo para, quando $\hat{r}^k < \epsilon$, com $\epsilon > 0$ suficientemente pequeno, o qual significa que w^k, a largura de $I\!P^k$, converge também para zero. A *figura* 5 mostra a trajetória dos centros num politopo em $I\!R^2$.

Consideremos o problema original (5.1). Seja $x^0 \in \text{int}(I\!P)$, o ponto inicial, e $L^0 > c^t x^0$, um limite superior para o valor ótimo.

Algoritmo: *Resolvendo P.L. usando-se bolas inscritas no politopo.*

- Dados x^0 e L^0.
- **Passo 0:** $k = 0, \quad \lambda = \frac{1}{2}$.
- **Passo 1:** Construir o politopo $I\!P^k = \{x \mid Ax \leq b \ e \ c^t x \leq L^k\}$.
- **Passo 2:** Calcular $(\hat{x}^k, \hat{r}^k)$, o centro e o raio da maior bola em $I\!P^k$.
- **Passo 3:** $L^{k+1} = c^t \hat{x}^k$.
- **Passo 4:** $x^{k+1} = \hat{x}^k - \lambda \hat{r}^k \frac{c}{\|c\|}$.
- **Passo 5:** Se $\hat{r}^k < \epsilon$ então **parar.**
- **Passo 6:** k=k+1.
- **Passo 7:** Voltar para o passo **1.**



5.2 Achando a Maior Bola Inscrita num Politopo.

Nesta seção desenvolvemos um procedimento para calcular o centro da maior bola inscrita num politopo. O mecanismo para resolver o problema de programação linear, como falamos na seção anterior, é baseado principalmente no fato de como achamos tal centro.

Consideremos o politopo $I\!P$ e o seu interior, definidos como na seção anterior, assumimos também que nenhuma linha da matriz A em (5.1) tem norma zero. O procedimento consiste essencialmente em projetar sucessivamente um ponto interior sobre os hiperplanos que definem $I\!P$, para logo reduzir o volume de $I\!P$ quando o ponto atual está o mais próximo possível a um centro da maior bola.

Definição 5.2.1 *Seja $y \in \text{int}(I\!P)$, denotemos de $\text{proj}_i(y)$ e $\text{comp}_i(y)$, a projeção e componente de y sobre o i-hiperplano $\langle a_i, x\rangle = b_i$, respectivamente. Eles são dados por*

$$\text{proj}_i(y) = y + \frac{(b_i - \langle a_i, y\rangle)}{||a_i||}\frac{a_i}{||a_i||}$$

e

$$\text{comp}_i(y) = ||y - \text{proj}_i(y)|| = \frac{b_i - \langle a_i, y\rangle}{||a_i||},$$

onde $a_i \in I\!R^n$ é um vetor coluna representando a i-linha de A, e b_i representa a i-coordenada de b.

Definição 5.2.2 *Dado $x \in \text{int}(I\!P)$, $\xi(x) = \min\{\text{comp}_i(x) \mid 1 \leq i \leq m\}$ representa o raio da maior bola com centro x e incluída em $I\!P$.*

Se conhecemos a priori r^*, o raio da maior bola, pela definição anterior, $\xi(x)$ nos proporciona informação de quão perto está x de ser um centro ótimo. Pode acontecer que uma solução ótima x^* não seja única, assim, se $x \in \text{int}(I\!P)$ e

$$|\xi(x) - r^*| < \epsilon,$$

então x é uma aproximação a um centro ótimo, e no entanto pode estar distante de x^*. Deste modo, um ponto interior estará mais próximo a um centro

da maior bola inscrita em $I\!P$, quanto maior seja o valor de ξ, claramente r^* verifica

$$r^* = \mathrm{Sup}\{\xi(x) \mid x \in I\!P\}.$$

Este último fato justamente, constitui uma característica do procedimento que tentamos desenvolver, no qual, o raio de uma bola em $I\!P$ é incrementado a cada passo.

Cabe ressaltar que, embora as restrições redundantes não influenciem na forma de $I\!P$, a sua presença complica a obtenção de uma solução. Nós não tentamos achar e excluir tais restrições pois, isso poderia ser tão complicado quanto resolver o problema original.

Definição 5.2.3 *Dado* $x \in \mathrm{int}(I\!P)$,

$$\mathrm{proj}_{\alpha_i}(x) = x + \alpha_i \frac{a_i}{||a_i||}$$

e

$$\mathrm{proj}_{\beta_i}(x) = x - \beta_i \frac{a_i}{||a_i||},$$

onde

$$\alpha_i = \max\{\alpha \in I\!R \mid x + \alpha \frac{a_i}{||a_i||} \in I\!P\}$$

e

$$\beta_i = \max\{\beta \in I\!R \mid x - \beta \frac{a_i}{||a_i||} \in I\!P\}.$$

A *figura* 6 exemplifica o significado geométrico destas projeções.

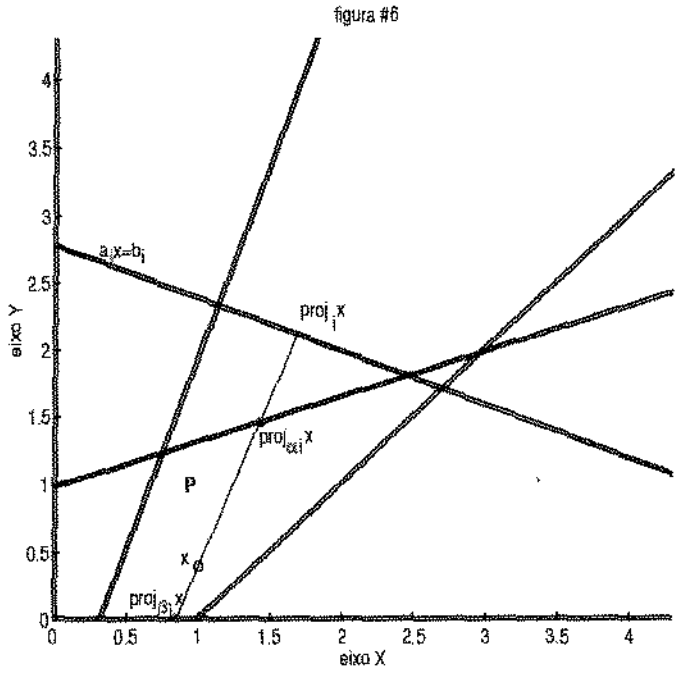
Considerando as propriedades geométricas e algébricas das projeções ortogonais, nós podemos determinar α_i e β_i de um modo eficiente:

$$\alpha_i = \min\{\frac{(b_k - a_k^t x)}{a_i^t a_k}||a_i||, \text{ se: } a_i^t a_k > 0, \ 1 \leq k \leq m\}$$

e

$$\beta_i = \min\{-\frac{(b_k - a_k^t x)}{a_i^t a_k}||a_i||, \text{ se: } a_i^t a_k < 0, \quad 1 \leq k \leq m\}.$$

Agora, dado x^k um ponto inicial interior em $I\!P$, tomemos o ponto médio do segmento $[\mathrm{proj}_{\alpha_1}(x^k), \mathrm{proj}_{\beta_1}(x^k)]$. Na *figura* 7, nós mostramos o caso quando $k = 0$, ressaltamos que as linhas em negrito são as que definem o politopo original.



$$x_1^k = \frac{\text{proj}_{\alpha_1}(x^k) + \text{proj}_{\beta_1}(x^k)}{2} = x^k + \frac{(\alpha_1 - \beta_1)}{2}\frac{a_1}{||a_1||},$$

além disso

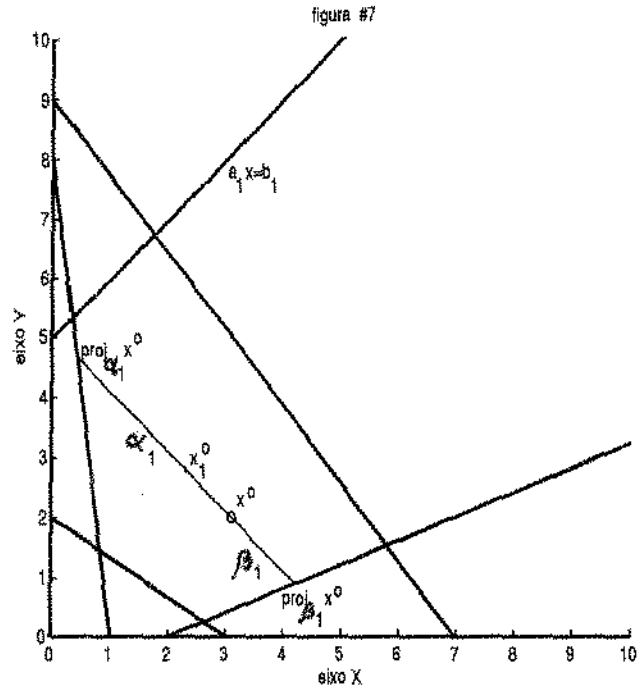
$$x_2^k = \frac{\text{proj}_{\alpha_2}(x_1^k) + \text{proj}_{\beta_2}(x_1^k)}{2} = x_1^k + \frac{(\alpha_2 - \beta_2)}{2}\frac{a_2}{||a_2||},$$

$$,\ldots,$$

$$x_m^k = \frac{\text{proj}_{\alpha_m}(x_{m-1}^k) + \text{proj}_{\beta_m}(x_{m-1}^k)}{2} = x_{m-1}^k + \frac{(\alpha_m - \beta_m)}{2}\frac{a_m}{||a_m||}, \qquad (5.9)$$

onde m é o número de restrições, logo tomamos $x^{k+1} = x_m^k$. Com isto, a obtenção de x^{k+1} a partir de x^k constitui uma iteração, a expectativa é de obter uma melhor solução que a anterior.

Podemos mostrar agora que se $x^k \in \text{int}(I\!P)$, então também x_1^k, para isto, claramente pela definição (5.2.3), temos que $\alpha_1, \beta_1 > 0$, logo, seja a_j uma linha de A, então.



- Se $\langle a_j, a_1 \rangle = 0$, então por (5.9)

$$\langle a_j, x_1^k \rangle = \langle a_j, x^k \rangle + \frac{(\alpha_1 - \beta_1)}{2} \frac{\langle a_j, a_1 \rangle}{||a_1||},$$

de onde

$$\langle a_j, x_1^k \rangle = \langle a_j, x^k \rangle < b_j.$$

- Se $\langle a_j, a_1 \rangle > 0$, temos que

$$\begin{aligned} \langle a_j, x_1^k \rangle &= \langle a_j, x^k \rangle + \frac{(\alpha_1 - \beta_1)}{2} \frac{\langle a_j, a_1 \rangle}{||a_1||} \\ &\leq \langle a_j, x^k \rangle + \{ \frac{(b_j - \langle a_j, x^k \rangle)}{\langle a_j, a_1 \rangle} ||a_1|| - \beta_1 \} \frac{\langle a_j, a_1 \rangle}{||a_1||} \\ &< \langle a_j, x^k \rangle + \frac{(b_j - \langle a_j, x^k \rangle)}{\langle a_j, a_1 \rangle} ||a_1|| \frac{\langle a_j, a_1 \rangle}{||a_1||} \\ &= \langle a_j, x^k \rangle + b_j - \langle a_j, x^k \rangle = b_j. \end{aligned} \tag{5.10}$$

O mesmo raciocínio se aplica para o caso $\langle a_j, a_1 \rangle < 0$. Este fato nós permite afirmar que x_2^k é também um ponto interior, e portanto $x^{k+1} \in \text{int}(I\!P)$.□

• Se $\xi(x^{k+1}) \leq \xi(x^k)$, isto é, x^{k+1} não é uma melhor solução que x^k, então nós mantemos x^k como um novo ponto inicial, e realizamos uma redução do politopo $I\!P$ com respeito a x^k. Isto pode ser feito reduzindo as $\text{comp}_i(x^k)$ em $\frac{\xi(x^k)}{2}$, para $i = 1, 2, ..., m$. Desde que a equação de um plano pode ser escrita por $\langle a, X - X_0 \rangle = 0$, onde X é um vetor variável, X_0 é um ponto de passo e a é o vetor normal associado ao plano; nós podemos calcular os novos b_i' por:

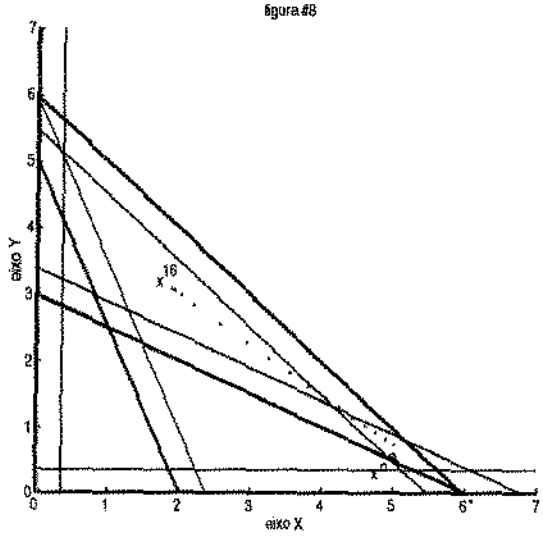
$$\begin{aligned} b_i' &= \langle a_i, x^k + (\text{comp}_i(x^k) - \frac{\xi(x^k)}{2}) \frac{a_i}{||a_i||} \rangle \\ &= \langle a_i, x^k \rangle + (\text{comp}_i(x^k) - \frac{\xi(x^k)}{2}) \frac{\langle a_i, a_i \rangle}{||a_i||} \\ &= \langle a_i, x^k \rangle + (\frac{(b_i - \langle a_i, x^k \rangle)}{||a_i||} - \frac{\xi(x^k)}{2}) ||a_i|| \\ &= \langle a_i, x^k \rangle + b_i - \langle a_i, x^k \rangle - \frac{\xi(x^k)}{2} ||a_i|| \\ &= b_i - \frac{\xi(x^k)}{2} ||a_i||. \end{aligned} \tag{5.11}$$

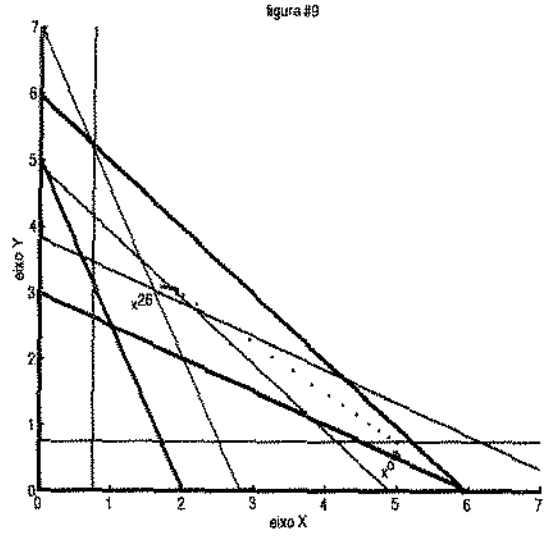
Observemos que, após a redução do politopo, x^k é um ponto inicial interior, com respeito ao novo politopo, isto pode ser verificado imediatamente desde o fato que $\xi(x^k) > 0$, pois x^k é um ponto interior, e

$$\begin{aligned} \frac{b_i' - \langle a_i, x^k \rangle}{||a_i||} &= \frac{b_i - \frac{\xi(x^k)}{2} ||a_i|| - \langle a_i, x^k \rangle}{||a_i||} \quad i = 1, 2, ..., m \\ &= \frac{b_i - \langle a_i, x^k \rangle}{||a_i||} - \frac{\xi(x^k)}{2} \\ &\geq \xi(x^k) - \frac{\xi(x^k)}{2} \\ &= \frac{\xi(x^k)}{2} > 0. \end{aligned}$$

• Se $\xi(x^{k+1}) > \xi(x^k)$, então pela definição (5.2.2), x^{k+1} é uma melhor solução que x^k. Neste caso, não reduzimos o politopo e x^{k+1} constituirá

o novo ponto inicial, e como mostramos em (5.10), $x^{k+1} \in \text{int}(I\!P)$. Logo, aplicamos uma outra iteração descrita por (5.9). A *figura* 8 e a *figura* 9 ilustram parte deste procedimento. Lembramos que as linhas em negrito são as que definem o politopo original.□





5.2.1 O Algoritmo de Projeções Ortogonais.

Pode acontecer que o incremento de ξ, controlado pela inequação $\xi(x^{k+1}) > \xi(x^k)$, tenha um caráter acumulativo, isto é,

$$\lim_{k\to\infty}[\xi(x^{k+1}) - \xi(x^k)] = 0,$$

o qual não permite reduzir o politopo, nós tentamos evitar isto substituindo a inequação $\xi(x^{k+1}) > \xi(x^k)$ por uma do tipo $\xi(x^{k+1}) > (1+\lambda)\xi(x^k)$, para $0 < \lambda < \delta$, e δ suficientemente pequeno, isto significa que o incremento do valor de ξ é maior que uma proporção de $\xi(x^k)$, i.e, $\xi(x^{k+1}) - \xi(x^k) > \lambda\xi(x^k)$.

Devido a que nós somente modificamos o vetor b, quando reduzimos o politopo, nós adotamos agora, por razões de comodidade, a seguinte notação $\xi_k(x^k)$. Isto representa a $\xi(x^k)$ (definição 5.2.2) com respeito ao politopo $I\!P^k$. Neste caso, $I\!P^k$ fica determinado diretamente pelas modificações que se faça no vetor b. O algoritmo para, quando $w^k = \text{larg}(I\!P^k) < \epsilon$, com ϵ suficientemente pequeno.

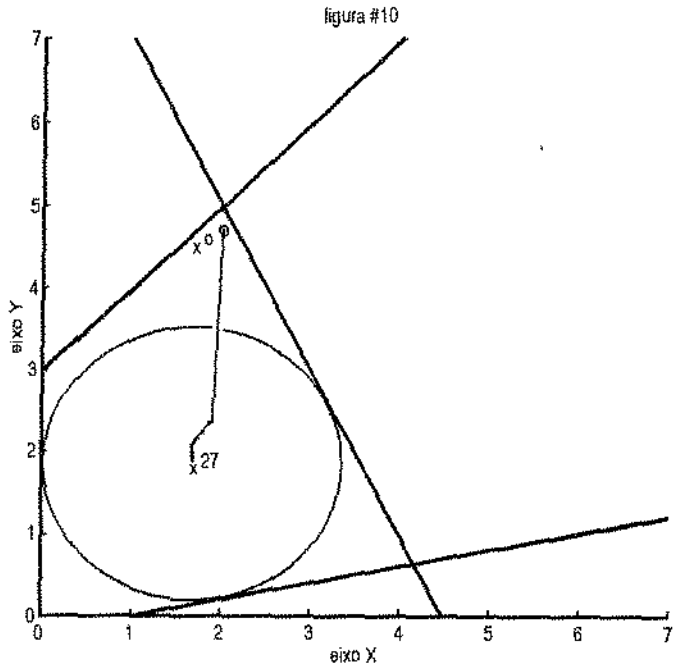
Algoritmo: *Para se calcular a maior bola inscrita num politopo* $I\!P$.

- Seja $x^0 \in \text{int}(I\!P)$, e $0 < \epsilon < \xi(x^0)$.
- **Passo 0:** Faça $k = 0$, $b^0 = b$.
- **Passo 1:** Se $\xi_k(x^k) < \epsilon$, **parar**.
- **Passo 2:** Calcular $x_1^k, x_2^k, ..., x_m^k$.
- **Passo 3:** Faça $x^{k+1} = x_m^k$.
- **Passo 4:** Se $\xi_k(x^{k+1}) > (1+\lambda)\xi_k(x^k)$, então faça $b^{k+1} = b^k$, logo, ir ao passo **7**.
- **Passo 5:** Faça $b_i^{k+1} = b_i^k - \frac{\xi_k(x^k)}{2}||a_i||$, $\quad i = 1, 2, 3, ..., m$.
- **Passo 6:** Faça $x^{k+1} = x^k$ e $\xi_{k+1}(x^{k+1}) = \frac{\xi_k(x^k)}{2}$.
- **Passo 7:** k=k+1.
- **Passo 8:** voltar para o passo **1**.

Na sequência $\{x^k\}$ obtida pelo algoritmo, $\{\xi_0(x^k)\}$ forma uma sequência não decrescente, onde x^{k+1} não é pior solução que x^k. No entanto, um ponto x^k será uma aproximação a um ponto ótimo se

$$\xi_0(x^k) \to r^* = \text{Sup}\{\xi(x) \mid x \in I\!P\}.$$

Na *figura* 10 ilustramos a trajetória completa da sequência dos pontos gerados pelo algoritmo num politopo em $I\!R^2$. Neste caso, ele para quando $\epsilon < 0.01$, isto significa $w^k = \text{larg}(I\!P^k) \leq 0.02$, aproximadamente.



5.3 Resultados Computacionais.

O nosso objetivo agora é observar primeiro, o comportamento do algoritmo de projeções ortogonais para achar o centro da maior bola inscrita num politopo. Para isso, realizamos alguns testes com problemas gerados e outros do NETLIB. No entanto, apesar de não termos provado que o algoritmo de projeções ortogonais converge ao centro da maior bola, o modo como foi construído nos permite comparar seus resultados com as soluções obtidas resolvendo diretamente o problema de programação linear

$$\begin{aligned} &\text{Maximizar} && r \\ &\text{sujeito a:} && \\ & && \langle a_i, x\rangle + ||a_i||r \leq b_i \quad i = 1, 2, ..., m. \end{aligned} \tag{5.12}$$

Para esse fim, escolhemos problemas onde o conjunto de soluções factíveis definido por (5.1) é limitado e com interior não vazio; as restrições do tipo $Ax \geq b$, $Ax = b$ e de não negatividade $x \geq 0$ foram transformadas na forma $Ax \leq b$.

Pelas vantagens, tanto no aspecto da estrutura matricial esparsa como sua utilidade prática na programação, ambos algoritmos foram implementados em MATLAB. Nós inicializamos o algoritmo de projeções ortogonais em pontos interiores x^0 diferentes, os quais foram obtidos movimentando em δ, o centro ótimo Ω, na direção e_i, onde $0 \leq \delta < \rho$, e_i é o i-ésimo vetor canônico e ρ é o raio ótimo. Cabe notar, que a única referência para determinar quando um ponto atual x^k está ou não próximo a um ponto ótimo, é a sua componente mínima atual com respeito ao politopo original, isto é, $\xi(x^k)$.

Na última subseção, nós otimizamos os problemas descritos com a presença de um vetor custo c. Notemos que, de certa forma, com a finalidade de acelerar o algoritmo para otimizar o problema de programação linear (subseção 5.1.1), nós podemos optar por não calcular exatamente um centro, isto é, reduzir o politopo até que a sua largura w seja suficientemente pequena em proporção ao politopo anterior, $w^k = 0.15w^{k-1}$. Este fato nos permite também obter maior aproximação na procura da solução ótima x^* pois, o parâmetro ϵ, o qual determina a exatidão de um centro (algoritmo, subseção 5.2.1), varia conforme ao decrescimento do politopo atual.

Devido ao fato de que podem existir soluções ótimas alternativas, nós comparamos só o valor $c^t x^k$ com o valor ótimo $c^t x^*$.

Lembramos que $w^k = \text{larg}(I\!P^k)$ denota a largura do k-ésimo politopo atual, assim como $\xi(x^k) = \min\{||x^k - \text{proj}_i x^k|| \ , \ 1 \leq i \leq m\}$ é a componente mínima, com respeito ao politopo original $I\!P$, como na definição (5.2.1). O algoritmo para, em ambos casos, quando o valor de w^k é suficientemente pequeno.

5.3.1 Achando o Centro e o Raio da Maior Bola num Politopo.

Nós usamos para os seguintes exemplos, como um critério para que o algoritmo pare, $\xi_k(x^k) < \epsilon$, o qual significa que a largura de $I\!P^k$ verifica $w^k = 2\epsilon$ aproximadamente. Os valores de ϵ são supostamente pequenos e diferentes, notando deste modo, que uma maior aproximação é conseguida quando o valor do mesmo é reduzido. Lembramos também que o raio ρ, e o centro ótimo Ω, para cada politopo, foram calculados resolvendo o nosso problema por programação linear.

Exemplo 5.3.1 DAT44. *Este politopo $I\!P$ foi gerado e consiste de 9 restrições e 4 variáveis. O raio ótimo $\rho = 0.9030$ e o centro ótimo Ω, foram obtidos pela resolução do problema de P.L.*

$\mathbf{x^0}$	$\xi(\mathbf{x^0})$	$\mathbf{k}$	$\mathbf{w^k}$	$\xi(\mathbf{x^k})$	$\frac{\rho - \xi(\mathbf{x^k})}{\xi(\mathbf{x^k})}$
$\Omega + \rho e_1$	0.8568	15	0.02000	0.8903	0.014300
$\Omega + \rho e_2$	0.5176	26	0.02000	0.8935	0.010632
$\Omega + \rho e_3$	0.5440	25	0.02000	0.8942	0.009841
$\Omega + \rho e_4$	0.4203	39	0.00200	0.8972	0.006464
$\Omega - \rho e_1$	0.5885	42	0.00200	0.9003	0.002999
$\Omega - \rho e_4$	0.2306	215	0.00001	0.9030	0.000012
$\Omega - \rho e_3$	0.6826	42	0.00200	0.8993	0.004114

Exemplo 5.3.2 DAT29. *Este politopo foi gerado e consiste de* 11 *restrições e* 6 *variáveis. O raio* $\rho = 3.5300$ *e o centro ótimo* Ω*, foram obtidos por resolução do problema de P.L.*

$\mathbf{x^0}$	$\xi(\mathbf{x^0})$	k	$\mathbf{w^k}$	$\xi(\mathbf{x^k})$	$\frac{\rho-\xi(\mathbf{x^k})}{\xi(\mathbf{x^k})}$
$\Omega + \frac{9}{10}\rho e_1$	2.5055	28	0.0200	3.3976	0.038969
$\Omega + \frac{9}{10}\rho e_2$	2.3348	28	0.0200	3.3348	0.058534
$\Omega + \frac{9}{10}\rho e_3$	2.0474	46	0.0010	3.3574	0.051409
$\Omega + \frac{9}{10}\rho e_6$	2.5311	44	0.0010	3.4205	0.032013
$\Omega - \frac{9}{10}\rho e_2$	2.1426	32	0.0020	3.3630	0.049658
$\Omega - \frac{9}{10}\rho e_4$	2.1956	32	0.0020	3.5207	0.002642
$\Omega - \frac{9}{10}\rho e_5$	2.3348	32	0.0020	3.4859	0.012651
$\Omega + \frac{9}{10}\rho e_1$	2.1470	32	0.0020	3.3901	0.041267

Exemplo 5.3.3 DAT28. *Este politopo é gerado, tem* 15 *restrições e* 9 *variáveis. O raio* $\rho = 5.8680$ *e o centro ótimo* Ω*, foram obtidos pela resolução do problema de P.L.*

$\mathbf{x^0}$	$\xi(\mathbf{x^0})$	k	$\mathbf{w^k}$	$\xi(\mathbf{x^k})$	$\frac{\rho-\xi(\mathbf{x^k})}{\xi(\mathbf{x^k})}$
$\Omega + \frac{19}{20}\rho e_1$	3.4091	28	0.1000	5.6968	0.030052
$\Omega + \frac{19}{20}\rho e_3$	3.6046	28	0.1000	5.8417	0.004502
$\Omega + \frac{19}{20}\rho e_7$	4.5639	25	0.1000	5.7263	0.024745
$\Omega + \frac{19}{20}\rho e_7$	4.5639	89	0.0001	5.8628	0.000887
$\Omega + \frac{19}{20}\rho e_9$	4.5639	45	0.0100	5.8442	0.004072
$\Omega - \frac{19}{20}\rho e_4$	4.4779	25	0.1000	5.7158	0.026628
$\Omega - \frac{19}{20}\rho e_5$	1.9572	32	0.1000	5.7295	0.024173
$\Omega - \frac{19}{20}\rho e_6$	0.6521	54	0.0100	5.8570	0.001878
$\Omega - \frac{19}{20}\rho e_8$	0.2935	24	0.2000	5.5352	0.060124

Exemplo 5.3.4 AFIRO. *Este politopo é um dos problemas do NETLIB, tem 59 restrições e 32 variáveis. O raio $\rho = 5.9780$ e o centro ótimo Ω, foram obtidos pela resolução do problema de P.L.*

$\mathbf{x^0}$	$\xi(\mathbf{x^0})$	k	$\mathbf{w^k}$	$\xi(\mathbf{x^k})$	$\frac{\rho-\xi(\mathbf{x^k})}{\xi(\mathbf{x^k})}$
$\Omega + \frac{2}{3}\rho e_7$	1.9927	72	0.0600	5.6782	0.052798
$\Omega + \frac{2}{3}\rho e_2$	3.6829	101	0.0300	5.8485	0.022142
$\Omega + \frac{9}{10}\rho e_{31}$	1.8000	45	0.1000	5.3625	0.114779
$\Omega + \frac{7}{10}\rho e_{15}$	1.7934	45	0.1000	5.3443	0.118575
$\Omega - \frac{3}{5}\rho e_{13}$	2.3912	80	0.0332	5.7806	0.034149
$\Omega - \frac{4}{5}\rho e_{10}$	2.5948	123	0.0096	5.9134	0.010924
$\Omega^1 - \frac{4}{5}\rho e_{23}$	1.1921	48	0.0948	5.3415	0.119161
$\Omega^1 - \frac{2}{3}\rho e_{28}$	2.7974	179	0.0048	5.9653	0.002129

Exemplo 5.3.5 SC50A. *Este politopo é um dos problemas do NETLIB, tem 97 restrições e 48 variáveis. O raio $\rho = 5.3801$ e o centro ótimo Ω, foram obtidos pela resolução do problema de P.L.*

$\mathbf{x^0}$	$\xi(\mathbf{x^0})$	k	$\mathbf{w^k}$	$\xi(\mathbf{x^k})$	$\frac{\rho-\xi(\mathbf{x^k})}{\xi(\mathbf{x^k})}$
$\Omega + \frac{2}{3}\rho e_{37}$	2.7254	91	0.02000	5.3698	0.001918
$\Omega + \frac{2}{3}\rho e_5$	2.8527	71	0.02000	5.3625	0.003282
$\Omega + \frac{9}{10}\rho e_{48}$	1.9572	56	0.08000	5.2586	0.023105
$\Omega + \frac{3}{4}\rho e_{15}$	2.3950	59	0.08140	5.2475	0.058999
$\Omega + \frac{19}{20}\rho e_{26}$	1.5995	74	0.03000	5.3627	0.003245
$\Omega - \frac{9}{10}\rho e_4$	0.5379	22	0.37000	4.7380	0.135521
$\Omega - \frac{9}{10}\rho e_{42}$	0.5379	158	0.00012	5.3800	0.000019
$\Omega - \frac{4}{5}\rho e_{45}$	1.1819	67	0.07200	5.3279	0.009797
$\Omega - \frac{9}{10}\rho e_4$	0.5379	190	0.00002	5.3801	0.000000

Exemplo 5.3.6 ADLITTLE. *Este politopo é um dos problemas do NETLIB, tem 153 restrições e 97 variáveis. O raio $\rho = 1.2573$ e o centro ótimo Ω, foram obtidos pela resolução do problema de P.L.*

x^0	$\xi(x^0)$	k	w^k	$\xi(x^k)$	$\frac{\rho-\xi(x^k)}{\xi(x^k)}$
$\Omega + \frac{2}{3}\rho e_1$	1.0000	58	0.0010	1.2570	0.000239
$\Omega + \frac{2}{3}\rho e_4$	0.882510	81	0.0020	1.2557	0.001274
$\Omega + \frac{4}{5}\rho \frac{a_{81}}{\|\|a_{81}\|\|}$	0.253740	53	0.0020	1.2556	0.001354
$\Omega + \frac{3}{4}\rho \frac{a_{59}}{\|\|a_{59}\|\|}$	0.692025	51	0.0020	1.2560	0.001035
$\Omega + \frac{3}{4}\rho \frac{a_{27}}{\|\|a_{27}\|\|}$	0.313956	29	0.0692	1.2215	0.029308
$\Omega + \frac{3}{4}\rho \frac{a_{27}}{\|\|a_{27}\|\|}$	0.313956	52	0.0020	1.2559	0.001115
$\Omega - \frac{2}{3}\rho e_4$	0.441366	26	0.0557	1.2167	0.033369

5.3.2 Minimizando os Problemas Anteriores.

Considerando o vetor custo c, nós aqui otimizamos os problemas relacionados aos exemplos anteriores. Um centro Ω, o centro da maior bola inscrita no politopo original, é considerado como ponto inicial. O algoritmo para, quando a largura do politopo atual é pequena. Ressaltamos que o ϵ, para este algoritmo de otimização, é assumido suficientemente menor que no algoritmo que calcula a maior bola; além disso, $c^t x^k$ denota o valor obtido pelo nosso algoritmo e $c^t x^*$ é o valor ótimo obtido resolvendo programação linear.

Problema	**m × n**	**k**	$\mathbf{w^k}$	$\mathbf{c^t x^k}$	$\mathbf{c^t x^*}$	$\frac{c^t x^* - c^t x^k}{c^t x^k}$
DAT44	9 × 4	9	0.01500	-2.1738	-2.1800	0.002852
DAT44	9 × 4	13	0.00160	-2.1795	-2.1800	0.000229
DAT29	11 × 6	42	0.00190	-88.2759	-88.7948	0.005878
DAT29	11 × 6	27	0.01700	-87.2961	-88.7948	0.017168
DAT29	11 × 6	51	0.00015	-88.7922	-88.7948	0.000029
DAT28	14 × 9	33	0.00016	-213.8026	-218.5300	0.022111
AFIRO	59 × 32	152	0.00500	-464.3407	-464.7531	0.000888
SC50A	97 × 48	61	0.04700	-55.1900	-64.5750	0.170048
SC50A	97 × 48	85	0.00010	-63.9500	-64.5750	0.009773
ADLITTLE	153 × 97	163	0.00010	-1418402.6	-1854267.0	0.307292

Capítulo 6

Conclusões e Trabalhos Futuros.

No segundo capítulo deste trabalho, nós descrevemos alguns conceitos de complexidade para algoritmos, os problemas de otimização e sua classificação. No terceiro capítulo foi exposto intuitivamente o problema de programação linear e o algoritmo simplex, também foram expostos com mais detalhes três algoritmos de pontos interiores, eles constituem alguns claros exemplos devido ao seu caráter ilustrativo e por sua simplicidade na implementação.

No quarto capítulo introduzimos alguns conceitos e propriedades de centros para politopos. Consideramos isto importante, porque eles estão relacionados diretamente com nosso trabalho aqui desenvolvido. Além disso, todo algoritmo polinomial de pontos interiores usado para resolver problemas de programação linear, tem um mecanismo, implícito ou explícito, de centragem envolvido. Este último fato nos permite acreditar que novos conceitos e propriedades de centros para poliedros podem gerar interessantes novas formas de abordar o problema de programação linear.

No último capítulo, nós desenvolvemos um procedimento para se calcular o centro da maior bola inscrita num politopo. Posteriormente, o algoritmo é adaptado procurando se resolver o problema de programação linear. Notamos que a região factível pode ter uma forma qualquer; isto é, não precisa de estrutura especial, portanto, ambos algoritmos podem ser aplicados em problemas onde a matriz A pode ser densa ou esparsa. Além do mais, o erro de arredondamento não se torna acumulativo pois, estamos sempre trabalhando os elementos originais do problema (com exceção do vetor b). Outra

vantagem inerente a este processo é que o algoritmo é muito fácil de se implementar, não necessitando de um programa estruturalmente complexo, e ele é facilmente paralelizável. Finalmente, tomando alguns problemas do NETLIB, e gerando outros de dimensão menor, nós testamos ambos procedimentos, onde os resultados computacionais foram incluidos neste trabalho.

Num futuro, devido a que o algoritmo de otimização, na prática, pode convergir a uma face do politopo, nós esperamos estabelecer um esquema de purificação, similar ao feito para o algoritmo de Karmarkar, permitindo deste modo obter um ponto extremo ótimo. Esperamos também, implementar ambos algoritmos em outros linguagens como FORTRAN ou PASCAL, deste modo, podemos obter uma melhor performance no tempo de execução dos mesmos. Com o mesmo objetivo, esperamos talvez, que métodos computacionais como processamento paralelo e erro de arredondamento pequeno nos permitam obter soluções mais precisas e num tempo mais eficiente.

No entanto, as idéias dos procedimentos foram baseadas em argumentos claros, neste trabalho não incluimos a prova de convergência do algoritmo de projeções ortogonais para se calcular a maior bola inscrita num politopo, assim como o tipo de complexidade dos mesmos. No entanto, estes tópicos serão matéria para uma futura pesquisa.

Bibliografia

[Bar] Barnes, E. R. (1986), A Variation on Karmarkar's Algorithm for Solving Linear Programming Problems, Mathematical Programming 36, pages 174-182.

[Baz] Bazaraa, M., Jarvis, J. and Sherali, H. (1990), Linear Programming and Network Flows, second edition, John Wiley.

[Br] Brassard, G. and Bratley, P. (1988), Algorithm Theory and Practice, Prentice-Hall.

[D] Dantzig, G. B. (1981), Reminiscences About the Origins of Linear Programming, technical report sol 81-5.

[GMS] Gilbert, J., Moler, C. and Schreiber, R. (1992) Sparse Matrices in MATLAB: Design ans Implementation, Journal Sian Matrix Anal. Appl., Vol. 13, No. 1, pp. 333-356.

[GT] Goldfard, D. and Todd, M.J. (1982), Modifications and Implementation of the Ellipsoid Algorithm for Linear Programming, Mathematical Programming 23 (1982), 1-19.

[JRTV] Jansen, B., Roos, C., Terlaky, T. and Vial, J.P (1994) Primal-Dual Algorithms for Linear Programming Based on the Logarithmic Barrier Method, Journal of Optimization Theory and Applications, Vol. 83, No. 1, pp. 1-26.

[JSS] Jarre, F., Sonnevend, G. and Stoer, J. (1988) An Implementation of the Mhetod of Analytic Center, Lecture Notes in Control and inform. Sci., Springer-Verlag 111, 297-307.

[KGM] Murty, K. G. (1976), Linear and Combinatorial Programming, Wiley, New York.

[Mo] Moretti, A. C. (1992), A New Technique for Finding the Analytic Center of a Polytope, Ph. D Thesis, Georgia.

[MSS] Marsten, R., Subramanian, R. and Saltzman, M. (1990) Interior Point Methods for Linear Programming.

[P] Papadimitriou, C. H. and Steiglitz, K. (1982), Combinatorial Otimization, Algorithms and Complexity, Prentice-Hall, Englewood Cliffs, New Jersey.

[Re] Renegar, J. (1988) A Polynomial-Time Algorithm, Based on Newton's Method, for Linear Programming, Mathematical Programming 40, 59-93.

[RMF1] Freund, R. (1988), Projetive Transformation for Interior Point Methods, Part I: Basic Theory and Linear Programming, Paper OR 179-88.

[RMF2] Freund, R. (1989), A Method for the Parametric Center Problem, With a Strictly Monotone Polinomial-Time Algorithm for Linear Programming, Working Paper OR 192-89.

[Ro] Roos, C. and Vial, J.P. (1988) Analytic Centers in Linear Programming, Reports of the Faculty of Technical Mathematics and Informatics no. 88-74.

[Ro] Roos, C. and Vial, J.P. (1988) A Polynomial Method of Approximate Centers for Linear Programming, Technical Report of the Faculty of Technical Mathematics and Informatics no. 88-68.

[Sch] Schrijver, A. (1986), Theory of Linear and Integer Programation, Wiley-Interscience, Series in Discrete Mathematics and Optimization.

[Sh] Shor, N.Z. (1977), Cut-off Method with Space Extension in Convex Programming Problems (in russian), Kibenetika (kiev) 1977 (1)(1977) 94-95[English translation: Cybernetics 13(1977)94-96].

[So] Sonnevend, Gy. An "Analytical Centre" for Polyhedrons and New Classes of Global Algorithms for Linear (Smooth, Convex) Programming, Dept. of Numerical Analysis, Inst. of Mathematics Eotvos University, 1088. Budapest, Muzeum Krt. 6-8.

[V] Vanderbei, R.J., Meketon, M. S. and Freedman, B.A., A Modification of Karmarkar's Linear Programming Algorithm. Algorithmica, bf 1 395-407, 1986.

[YN] Yudin, D.B. and Nemirovskii, A.S. (1976), Informational Complexity and Efficient Methods for the Solution of Convex Extremal Problems (in Russian), Ekonomika i Matematicheskie Metody 12(1976)357-369[Englihs translation: Makekon 13(3)(1977) 25-47].