Gilberto Souto

Visualização de Poliedros em Algoritmos de Programação Linear e Inteira

Florianópolis

Maio de 2008

Gilberto Souto

Visualização de Poliedros em Algoritmos de Programação Linear e Inteira

Orientador:
Clóvis Caesar Gonzaga, Dr.

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CURSO DE PÓS-GRADUAÇÃO EM MATEMÁTICA E COMPUTAÇÃO CIENTÍFICA

Florianópolis

Maio de 2008

Visualização de Poliedros em Algoritmos de Programação Linear e Inteira

por

Gilberto Souto

Esta Dissertação foi julgada adequada para obtenção do Título de "Mestre em Matemática", Área de Concentração em Matemática Aplicada, e aprovada em sua forma final pelo Curso de Pós-Graduação em Matemática e Computação Científica.

Prof. Dr. Clóvis Caesar Gonzaga
Coordenador

Comissão Examinadora

Prof. Dr. Clóvis Caesar Gonzaga
MTM – UFSC, Orientador

Prof^{a}. Dra. Elizabeth Wenger Karas
MTM – UFPR

Prof. Dr. Danilo Royer
MTM – UFSC

Prof. Dr. Juliano de Bem Francisco
MTM – UFSC

Florianópolis

Maio de 2008

Em memória de
Carlos Alberto Françozi e Marco Aurélio Alberto,
grandes amigos e heróis eternos...

Agradecimentos

Primeiramente, gostaria de agradecer a Deus que me proporcionou este caminho para que eu possa tornar-me, além de matemático, uma pessoa melhor.

Sou muito grato aos meus pais, Osvaldo e Carmen Mary de Souza Souto, pelo sempre auxílio financeiro e pela educação que me proporcionaram.

Obrigado a todos os amigos, professores e familiares que de certa forma acrescentaram à minha vida deixando um pouco de si. Em especial aos meus colegas Rafael Casali e Danilo Royer pelo apoio e companheirismo.

Agradeço ao meu professor orientador Clóvis Caesar Gonzaga pela maneira humana e científica de ensinar e pelo modo apaixonante como encara a Matemática, a história da Matemática e em particular sua pesquisa em Otimização.

Por fim, agradeço todo suporte financeiro cedido pela CAPES durante minha formação nesse mestrado.

Resumo

Neste trabalho será apresentado o programa POLIEDRO_V1 que tem como objetivo visualizar poliedros limitados (em $\mathbb{R}^2$ e $\mathbb{R}^3$), para auxiliar na compreensão da evolução dos algoritmos Simplex e Branch-and-Bound. Primeiramente, serão estudados alguns conceitos fundamentais de otimização irrestrita e restrita, as condições de otimalidade e o problema dual em Programação Linear (PL). A segunda fase do texto concentra-se no estudo teórico de poliedros, formas de representar poliedros e caracterização de faces. Após, é apresentado o programa POLIEDRO_V1 em linguagem Matlab, que consiste em visualizar politopos, gerando faces através da seleção e ordenação de vértices. A última fase do trabalho concentra-se no estudo e desenvolvimento dos algoritmos Simplex e Branch-and-Bound. Aplica-se o programa POLIEDRO_V1 em Programação Linear visualizando a região viável como politopo e apresenta-se a evolução, passo a passo, do método Simplex. Já em Programação Inteira (PI) visualiza-se graficamente o método Branch-and-Bound, resolvendo o problema de Programação Linear Inteira (PLI) através de subproblemas gerados por planos de corte. Os subproblemas são resolvidos recursivamente pelo método Simplex.

Abstract

This work presents the program POLIEDRO_V1, whose objective is to visualize bounded polyhedra in $\mathbb{R}^2$ and $\mathbb{R}^3$, to help the understanding of the evolution of the simplex and branch and bound algorithms. Initially some fundamental concepts in unconstrained and constrained optimization are studied, including the optimality conditions and the dual problem in Linear Programming. The second part of the text is dedicated to the theoretical study of polyhedra, ways of representing polyhedra and the characterization of faces. The program POLIEDRO_V1 is then presented in the language Matlab, consisting in the visualization of polytopes by generating facets through the selection and ordering of vertices. The last phase of the work concentrates on the study and development of the Simplex and Branch-and-Bound algorithms. The program POLIEDRO_V1 is applied to visualize the feasible region of a linear programming problem, on which the step by step evolution of the simplex method is drawn. In Integer Programming, the branch-and-bound algorithm is graphically represented by solving recursively the linear programming subproblems generated by adding cutting planes. These subproblems are solved by the simplex method and visualized.

Sumário

Lista de Figuras

Lista de Algoritmos

Lista de Símbolos

f	Função;
$\mathscr{L}$	Função Lagrangeano;
$\nabla f(x)$	Gradiente de f em x;
$H(x)$	Matriz hessiana de f em x;
Ω	Região viável;
$\partial\Omega$	Fronteira de Ω;
linΩ	Subespaço gerado por Ω;
convΩ	Envoltória convexa de Ω;
cone(Ω)	Envoltória cônica de Ω;
$aff\Omega$	Envoltória afim de Ω;
$\mathscr{E}$, $\mathscr{I}$	Conjunto de índices associados as restrições de igualdades e desigualdades;
$I_0(x)$	Conjunto de índices em que $x = 0$;
$I_+(x)$	Conjunto de índices em que $x > 0$;
$I(x)$	Conjunto de índices das restrições ativas em x;
$\mathscr{D}(x)$	Cone das direções de descida de f em x;
$\mathscr{C}_\Omega(x)$	Cone de direções viáveis de Ω em x;
$\mathscr{V}_\Omega(x)$	Cone de direções viáveis linearizadas de Ω em x;
$\mathscr{T}_\Omega(x)$	Cone tangente de Ω em x;
$P(C)$	Cone polar de um cone qualquer C;
$\mathscr{N}_\Omega(x)$	Cone de direções normais de Ω em x;
$\mathscr{R}_\Omega$	Cone de recessão de Ω;
$\|K\|$	Cardinalidade do conjunto K;
A_K	Matriz $m \times \|K\|$ composta pelas colunas de A com índices em K;
A^K	Matriz $\|K\| \times n$ composta pelas linhas de A com índices em K;
$F(x)$	Face de dimensão mínima em x;
relint$F(x)$	Interior relativo da face $F(x)$;
KKT	Condições de Otimalidade de Karunsh-Kuhn-Tucker;
MFCQ	Condições de Qualificação de Mangasarian-Fromovitz;
LICQ	Condição de Qualificação de Independência Linear;

1 Introdução

1.1 Considerações Iniciais

Esta dissertação iniciou a partir do interesse despertado pela teoria de otimização durante os cursos de Programação Linear e Pesquisa Operacional ministrados pelo professor Clóvis Gonzaga na Universidade Federal de Santa Catarina durante o segundo semestre de 2005. Durante as aulas a importância da visualização geométrica para compreender a teoria e os algoritmos era constantemente ressaltada, juntamente com as dificuldades de representar graficamente os problemas.

"A cada três ou quatro parágrafos, um desenho...
...se você não souber desenhar, é por que você não entendeu o problema."

"É complicado desenhar em três dimensões!"

Clóvis Gonzaga

Essas frases motivaram-me a criar o programa "*poliedro*". Este programa constrói politopos que servirão de auxílio na compreensão geométrica de problemas de otimização que utilizam como região viável poliedros. Posteriormente, desenvolvemos os algoritmos "*poliedro-simplex*" e "*poliedro-bb*". Outra característica desta dissertação são os relatos históricos em cada capítulo, motivados pela experiência acadêmica e pelo respeito aos precursores da teoria abordada neste trabalho.

Do ponto de vista matemático, a importância de desenvolver estes programas é obter uma ferramenta didática para auxiliar na compreensão da teoria proposta neste texto, através da visualização geométrica dos métodos Simplex e Branch-and-Bound.

1.2 Objetivos

1.2.1 Objetivo Geral

Este trabalho tem como objetivo desenvolver algoritmos para visualizar geometricamente a evolução "passo a passo" dos algoritmos Simplex (Programação Linear) e Branch-and-Bound (Programação Linear Inteira).

1.2.2 Objetivos Específicos

O objetivo geral deste texto pode ser dividido nos seguintes objetivos específicos:

1. Estudar os principais conceitos de otimização irrestrita e restrita, poliedros, Programação Linear e Inteira.
2. Implementar os algoritmos Simplex e Branch-and-Bound.
3. Desenvolver e implementar os programas "*poliedro*", "*poliedro_simplex*" e "*poliedro_bb*".

1.3 Estrutura do Texto

No início de cada capítulo, exibe-se uma breve passagem histórica, apresentando os matemáticos pioneiros em cada assunto proposto. Este trabalho está dividido em cinco capítulos, sendo que os capítulos dois e três dão suporte para o estudo dos capítulos quatro e cinco. Nos dois últimos capítulos estuda-se o método Simplex (Programação Linear) e o método Branch-and-Bound (Programação Inteira), respectivamente. Os capítulos que formam este texto estão organizados da seguinte forma:

No Capítulo II faz-se uma revisão dos principais conceitos de otimização irrestrita e restrita. Apresentam-se as condições de otimalidade e as condições de qualificação. Em seguida, discutem-se as condições de otimalidade para problemas de programação linear seguidas do problema de dualidade. Neste capítulo, o objetivo não é aprofundar-se nos conceitos, mas repassar as idéias centrais da importância dos poliedros (região viável) e os resultados relacionados a eles (condições de otimalidade e qualificação).

No Capítulo III define-se o poliedro através da intersecção finita de semi-espaços fechados e mostra-se o teorema da separação com o auxilio das propriedades básicas de conjuntos convexos. Em seguida, caracterizam-se as faces do poliedro mostrando a sua relação com os

problemas de otimização. Além disso, apresentam-se as diversas maneiras de representar poliedros e os diferentes conceitos de vértice. Na última seção deste capítulo, faz-se a visualização espacial de politopos, apresenta-se o programa `POLIEDRO_V1` e exibem-se exemplos de politopos utilizando essa nova técnica.

No Capítulo IV resolve-se graficamente problemas de Programação Linear com o auxílio do programa `POLIEDRO_V1`. Na sequência, apresenta-se o método Simplex destacando a importância da solução básica viável (vértice). Da relação entre o problema de programação linear primal e o problema dual, mostra-se o algoritmo Simplex e seus detalhes. Na última seção, faz-se a visualização passo a passo do método Simplex com exemplos bi e tridimensionais. Além disso, apresenta-se o programa "*poliedro_simplex*" composto basicamente dos programas "*poliedro*" e "*Simplex*".

No Capítulo V formula-se o problema de Programação Linear Inteira e discute-se uma técnica para resolver esta classe de problemas. Mostra-se uma idéia geral do método Branch-and-Bound através da recursividade e faz-se um exemplo bidimensional. Apresenta-se o algoritmo Branch-and-Bound e a visualização passo a passo deste método em três dimensões. Finaliza-se o capítulo apresentando o programa "*poliedro_bb*". O programa Poliedro_bb é o algoritmo Branch-and-Bound com as rotinas de representação gráfica, que utiliza o algoritmo Simplex e o programa "*poliedro*" apresentados nos capítulos anteriores.

Finalmente, expomos as considerações finais e propostas para trabalhos futuros. Além disso, anexo um guia básico com os procedimentos para utilizar o programa `POLIEDRO_V1`.

2 O Problema de Otimização

> [1] *"Dido, uma fenícia, persuadiu um chefe africano a dar-lhe tanta terra quanto ela pudesse cercar com a tripa de um touro. Assim foi. Primeiro, ela cortou as tripas em centenas de tiras bem fininhas. Depois, espertamente, uniu-as para traçar um semi-círculo no chão, a beira do mar Mediterrâneo. Era a máxima área costeira que ela poderia envolver. Neste lugar ela construiu uma cidade. A famosa Cartago."*

Antes de se tornar rainha de Cartago, Dido teria resolvido o primeiro problema de otimização da história. Mesmo sendo literário, o relato demonstraria que os povos da antiguidade possuíam conhecimento a respeito de áreas e comprimentos. Sabiam que, dentre as figuras de igual perímetro, o círculo é aquela com maior área. Por outro lado, poderia tratar-se apenas de um mito, em que o fato teria mais a ver com a simbologia feminina associada à cidade de Cartago, onde a forma do terreno em semi-círculo representa um útero engravidado, simbolizando o máximo de adequação ao arquétipo da mãe-terra para gerar uma cidade. Bennaton (2001).

A história do cálculo matemático de máximos e mínimos (Otimização) tem a ver com as noções de derivada e integral. Mas não como as conhecemos hoje. No século XVII estes conceitos estavam presos ao traçado de tangentes e à avaliação de áreas. Como era de se esperar, o cálculo, criado por Leibniz e Newton, não se afastou da tradição geométrica consagrada, por exemplo, em Arquimedes. Daí a importância de Euler e Lagrange. Ambos, no século XVIII, trabalharam no sentido de "desgeometrizar" a matemática criada pelos seus antecessores. Entre inúmeros outros feitos, Euler é o responsável pelo conceito de função matemática, generalizada por Dirichlet no século XIX. Lagrange, com o seu "Mécanique Analytique", reconstruiu a mecânica newtoniana em bases analíticas, desvencilhada de figuras geométricas. Sobretudo, um e depois o outro plantaram e fortaleceram as raízes do cálculo variacional. Tanto Euler e Lagrange, quanto Cauchy merecem destaque neste capítulo.

[1] A Eneida é uma das maiores audácias literária da Antiguidade, seguindo a trilha de Ilíada e da Odisséia, a Eneida, de Virgílio, reinventa Roma, narrando fatos anteriores à fundação da cidade. A lenda de Dido é pitoresca e tem sido muito usada em livros de otimização.

Finalmente, Cauchy. Hoje em dia, estamos acostumados a definir integrais e diferenciais em termos de "limites". Mas não era bem assim antes de Cauchy. Foi ele quem livrou o cálculo dos "infinitésimos" introduzindo os conceitos de limite e de continuidade. Deu fundamentos às idéias de derivada (gradiente) e integral. Inaugurou a análise matemática. Com a forma, a autonomia e o rigor atuais.

De certa forma, tomando o sentido contrário ao de Euler, Lagrange e Cauchy, mas mantendo a estrutura algébrica por eles apresentada, esse texto resgata a visualização geométrica de Leibniz e Newton, isto é, apresenta a "geometria" dos elementos que compõem o problema de otimização na interpretação formulada por Dantzig (1991), Chvátal (1983), entre outros.

Pode-se dizer que a diferenciação se originou de problemas relativos ao traçado de tangentes a curvas e de questões objetivando a determinação de máximos e mínimos de funções. Embora essas considerações remontem aos gregos antigos, parece razoável afirmar que a primeira manifestação realmente clara de método diferencial se encontra em algumas idéias de Fermat, expostas no século XVII em 1629.

Fermat usava um procedimento de cálculo algébrico idêntico ao que temos hoje. Mas o utilizava como uma espécie de algoritmo, sem o amparo dos conceitos de função, derivada ou diferencial. Neste sentido, foi um pioneiro. Mais de cem anos depois, Laplace reconheceu isto. Creditou a Fermat a invenção do cálculo diferencial. (EVES, 2004)

Já nos dias de hoje, em sua formulação mais geral, o problema de otimização é apresentado da seguinte forma:

Considere um conjunto Ω e uma função $f : \Omega \to \mathbb{R}$, dados. Encontrar, se existir, um ponto $\overline{x} \in \Omega$ tal que

$$(\forall x \in \Omega) \qquad f(\overline{x}) \leq f(x).$$

O problema de otimização pode ser escrito no seguinte formato

$$\underset{x \in \Omega}{\text{minimizar}} \quad f(x) \qquad \text{ou} \qquad \underset{\text{sujeito a} \quad x \in \Omega}{\text{minimizar}} \quad f(x) \tag{2.1}$$

O conjunto Ω é chamado *conjunto viável* e f é denominada a *função objetivo*.

Definição 1 *Um ponto $\overline{x} \in \Omega$ é um minimizador local de (2.1) se existe uma vizinhança V de $\overline{x}$ tal que*

$$(\forall x \in V \cap \Omega) \qquad f(\overline{x}) \leq f(x).$$

Uma solução ótima de (2.1) é chamada de minimizador global se

$$(\forall x \in \Omega) \qquad f(\overline{x}) \leq f(x).$$

Um minimizador (local ou global) é estrito se nas desigualdades acima $f(\overline{x}) < f(x)$ *para* $x \neq \overline{x}$.

O problema assim formulado é excessivamente geral. Neste texto estudaremos os casos em que:

- $\Omega \subset \mathbb{R}^n$ é convexo e fechado, em geral poliédrico; (2.2)
- $f : \mathbb{R}^n \to \mathbb{R}$ é de classe C^1 (C^2 se necessário), em geral analítica. (2.3)

Dependendo das definições da função objetivo f e do conjunto viável Ω, temos:

- **Programação Linear:** f é linear e Ω poliédrico.
- **Programação Quadrática:** f é quadrática e Ω poliédrico.
- **Programação Convexa:** Ω é convexo e f é uma função convexa.
- **Programação Inteira:** Ω é um conjunto de pontos com componentes inteiras. Em geral, f é linear e Ω é definido por um poliedro.
- **Programação Não Linear:** f é contínua (em geral C^1 ou C^2) e Ω é definido por equações e inequações.
- **Controle ótimo:** Ω é um conjunto de funções definido por equações diferenciais ordinárias e f é um funcional.
- **Otimização Combinatória:** Ω é um conjunto discreto, em geral finito (mas muito grande). A Programação Inteira é classificada como um problema de Otimização Combinatória. A Otimização Combinatória é uma área de pesquisa importantíssima, cujo estudo ocupa milhares de pesquisadores em todo o mundo.

Além da classificação acima, o problema de otimização pode ser dividido naturalmente em duas categorias: problemas com variáveis contínuas e problemas com variáveis discretas. Estes dois tipos de categorias possuem características bem diferentes, bem como os métodos para resolvê-los tornaram-se bastante divergentes. Discutiremos mais adiante a relação entre

os algoritmos de otimização contínua (Simplex) e de otimização combinatória (Branch and Bound), onde retomaremos essa questão.

Independentemente de sua classificação, a resolução de um problema complexo será feita através da resolução de uma sequência de problemas mais fáceis. Parte-se de um ponto inicial dado x^1 e gera-se uma sequência $(x^1,\ x^2,\ x^3,\dots)$, com a esperança de que esta sequência convirja para uma solução do problema original. O procedimento para isto é formalmente descrito por um *algoritmo*, que deve gerar como produto final um programa de computador. Desta forma podemos dizer que um algoritmo é uma receita. Uma seqüência de passos e decisões que devem ser tomadas para que uma determinada tarefa seja completada. Os programas de computador são compostos por um ou vários algoritmos, com diferentes graus de complexidade.

Depois desta breve passagem histórica e da idéia do problema de otimização serão apresentados conceitos básicos de otimização, baseados em Nocedal e Wright (1999), Izmailov e Solodov (2005) e Gonzaga (2005b), que servirão de apoio teórico para os capítulos seguintes. O objetivo desta parte introdutória não é aprofundar-se nos conceitos, mas repassar as idéias centrais da importância dos poliedros e os resultados relacionados a eles (condições de otimalidade). Alguns teoremas são simplesmente enunciados, pois suas demonstrações saem do enfoque proposto para este capítulo.

2.1 Otimização irrestrita

Considere o problema de otimização em que definimos o conjunto Ω por $\mathbb{R}^n$, isto é, definimos um problema de otimização irrestrita,

$$\text{(P)} \quad \underset{x \in \mathbb{R}^n}{\text{minimizar}} \quad f(x).$$

Definição 2 *Uma função $f : \mathbb{R}^n \to \mathbb{R}$ é diferenciável em $x \in \mathbb{R}^n$ se e somente se existe $\nabla f(x) \in \mathbb{R}^n$ tal que* $(\forall h \in \mathbb{R}^n)$

$$f(x+h) = f(x) + \nabla f(x)^T h + o(x,h),$$

com $\lim_{h \to 0} \dfrac{o(x,h)}{\|h\|} = 0.$

O gradiente de f em x é denotado por $\nabla f(x)$.

Definição 3 *Uma função f é duplamente diferenciável em x se e somente se* $(\forall h \in \mathbb{R}^n)$

$$f(x+h) = f(x) + \nabla f(x)^T h + \frac{1}{2} h^T H(x) h + o(x,h),$$

com $\lim_{h\to 0} \dfrac{o(x,h)}{\|h\|} = 0.$

A matriz hessiana de f *em* x *é denotada por* $H(x)$*, ou seja,* $H(x) = \nabla^2 f(x)$.

Definição 4 *Considere uma função* $f : \mathbb{R}^n \to \mathbb{R}$ *e uma direção* $d \in \mathbb{R}^n$*. A derivada direcional de* f *em* $x \in \mathbb{R}^n$ *na direção* d *é*

$$f'(x,d) = \lim_{\lambda \to 0^+} \frac{f(x+\lambda d) - f(x)}{\lambda},$$

se existir o limite.

Lema 1 *Se* f *é diferenciável em* x*, então* $f'(x,d) = \nabla f(x)^T d$.

Definição 5 *Uma direção* $d \in \mathbb{R}^n$ *é direção de descida a partir de* $x \in \mathbb{R}^n$ *se* $f'(x,d) < 0$.

Dado um ponto $\bar{x} \in \mathbb{R}^n$, não necessariamente um ponto viável ou um minimizador local do problema (P), deseja-se saber em quais direções teremos melhoras na função objetivo f.

Lema 2 *Se* d *é direção de descida a partir de* $x \in \mathbb{R}^n$*, então existe* $\bar{\lambda} > 0$ *tal que para todo* $\lambda \in (0, \bar{\lambda})$,

$$f(x+\lambda d) < f(x).$$

Tomando o conjunto de todas as direções, que satisfazem o Lema 2, a partir de $\bar{x}$, isto é, o conjunto de todas as direções de descida de f em $\bar{x}$, obtemos um *cone de direções de descida* que definimos da seguinte forma.

Definição 6 *O cone das direções de descida de* f *em* $\bar{x}$

$$\mathscr{D}(\bar{x}) = \{d \mid \nabla f(\bar{x})^T d < 0\}.$$

Os vetores $d \in \mathscr{D}(\bar{x})$*, pelo Lema 1, são direções de descida.*

Geometricamente, uma direção d será uma direção de descida se esta formar um ângulo obtuso com o gradiente de f, como mostra a Figura (2.1).

Tendo a noção de direção de descida, imagina-se como será resolvido o problema (P). Ou seja, caso se encontre uma direção de descida em um determinado ponto $x \in \mathbb{R}^n$, então pode-se andar, calcular o passo λ, nesta direção para encontrar um ponto com um valor menor de f. Uma pergunta a se fazer é: *Existe tal direção?* Para responder essa pergunta necessitamos das condições de otimalidade.

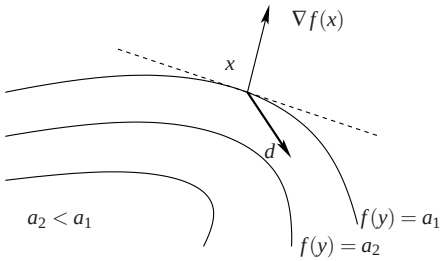


Figura 2.1: Cone das Direções de Descida em $\bar{x}$

2.1.1 Condições de Otimalidade

Os problemas irrestritos não aparecem frequentemente em aplicações práticas. Entretanto, as condições de otimalidade para este tipo de problema podem ser estendidas para problemas com restrições e algumas estratégias para encontrar um minimizador para um problema restrito se resumem à resolução de uma sequência de problemas irrestritos.

Dado um ponto $x \in \mathbb{R}^n$, deseja-se determinar, se possível, se este ponto é ou não um minimizador local do problema. Para isto, é preciso caracterizar um ponto de mínimo. Felizmente a hipótese de diferenciação de f ajuda na obtenção desta caracterização.

Teorema 1 (Teorema de Taylor) *Suponha que $f : \mathbb{R}^n \to \mathbb{R}$ é de classe C^1 e que $h \in \mathbb{R}^n$. Então temos que*

$$f(x+h) = f(x) + \nabla f(x+\lambda h)^T h, \tag{2.4}$$

para algum $\lambda \in (0,1)$. Por outro lado, se f é de classe C^2, temos que

$$\nabla f(x+h) = \nabla f(x) + \int_0^1 H(x+\lambda h)h \, d\lambda, \tag{2.5}$$

e que

$$f(x+h) = f(x) + \nabla f(x)^T h + \frac{1}{2} h^T H(x+\lambda h)h, \tag{2.6}$$

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

A demonstração do Teorema de Taylor pode ser encontrada em qualquer livro didático de análise - cálculo. O teorema de Taylor, aqui enunciado, tem como propósito dar sequência aos Teoremas 2, 3 e 4; pois suas demonstrações, encontradas em Nocedal e Wright (1999), utilizam esta fundamental ferramenta de aproximação de funções.

Teorema 2 (Condição necessária de primeira ordem) *Se $\bar{x}$ é um minimizador local do problema (P) e f é diferenciável em $\bar{x}$, então*

$$\nabla f(\bar{x}) = 0. \tag{2.7}$$

Teorema 3 (Condição necessária de segunda ordem) *Se $\bar{x}$ é um minimizador local do problema (P) e f é duplamente diferenciável em $\bar{x}$, então $H(\bar{x})$ é semi-definida positiva.*

Além de Nocedal e Wright (1999), as provas para estes teoremas podem ser obtidas em Bazaraa, Sherali e Shetty (1993), Bertsekas (1995) e Gonzaga (2006).

As condições necessárias devem ser verdadeiras para todos os minimizadores do problema. Por outro lado, um ponto que satisfaz a estas condições não é necessariamente um minimizador local, podendo ser um ponto de máximo local ou um ponto de sela (ponto onde a função tem uma mudança da sua curvatura).

Teorema 4 (Condição suficiente de segunda ordem) *Suponha que $\bar{x}$ satisfaz $\nabla f(\bar{x}) = 0$ e $H(\bar{x})$ é definida positiva. Então $\bar{x}$ é minimizador local de f. (minimizador local estrito)*

Este resultado garante, se as condições são satisfeitas em $\bar{x}$, que o *cone de direções de descida* $\mathscr{D}(\bar{x})$ é vazio, isto é, encontramos um minimizador local. Entretanto, verificar se a Hessiana da função é definida positiva é muito dispendioso.

Estas condições são para um problema irrestrito, em que o conjunto Ω é o $\mathbb{R}^n$. No entanto, deseja-se referenciar estas condições de otimalidade quando o conjunto Ω for um subconjunto de $\mathbb{R}^n$. Para isto, é necessário definir outros conceitos.

2.2 Otimização restrita

A otimização restrita trata de minimizar funções sujeitas a restrições sobre as variáveis, de fato, queremos estudar o problema de otimização restrita quando o conjunto de restrições forma um poliedro. Mas, este fato será abordado mais adiante.

Neste texto, consideramos a formulação do problema de otimização restrita dada por:

$$\begin{array}{ll} \text{minimizar} & f(x) \\ \text{sujeito a} & c_i(x) = 0,\ i \in \mathscr{E} \\ & c_i(x) \leq 0,\ i \in \mathscr{I}, \end{array} \tag{2.8}$$

em que f e as funções c_i são suaves e assumem valores reais, além disso $\mathscr{I}$ e $\mathscr{E}$ são dois conjuntos finitos de índices, $\mathscr{E} = \{1,\ldots,p\}$ e $\mathscr{I} = \{p+1,\ldots,m\}$. Como anteriormente, f é a função objetivo, enquanto c_i, $i \in \mathscr{E}$ são as restrições de igualdade e c_i, $i \in \mathscr{I}$ são as restrições de desigualdade. Definimos o conjunto viável $\Omega \subset \mathbb{R}^n$ como o conjunto dos pontos x que

satisfazem as restrições, isto é,

$$\Omega = \{x \mid c_i(x) = 0,\ i \in \mathscr{E}; \quad c_i(x) \leq 0,\ i \in \mathscr{I}\}. \tag{2.9}$$

Assim, reescrevemos o problema (2.8) na forma compacta

$$\begin{array}{l} \text{minimizar} \quad f(x). \\ x \in \Omega \end{array} \tag{2.10}$$

Definição 7 *Seja x um ponto viável. O conjunto ativo $I(x)$ é a união do conjunto $\mathscr{E}$ com os índices das restrições de desigualdade ativas; isto é,*

$$I(x) = \mathscr{E} \cup \{i \in \mathscr{I} \mid c_i(x) = 0\}. \tag{2.11}$$

Diz-se que uma restrição de desigualdade de índice i está *ativa* em x se $c_i(x) = 0$, está *inativa* (com folga) se $c_i(x) < 0$ e está violada se $c_i(x) > 0$.

Definição 8 *Um conjunto $C \subset \mathbb{R}^n$ é um* cone *se e somente se para todo $x \in C$ e para todo $\lambda > 0$,*

$$\lambda x \in C.$$

Definição 9 (Cone de direções viáveis) *Dizemos que $d \in \mathbb{R}^n$ é uma direção viável em relação ao conjunto Ω no ponto $\bar{x} \in \Omega$, quando d "penetra" no conjunto Ω, isto é, quando existe $\varepsilon > 0$ tal que*

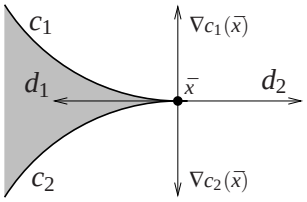
$$\bar{x} + td \in \Omega \quad \forall t \in [0, \varepsilon].$$

Denotamos por $\mathscr{C}_\Omega(\bar{x})$ o conjunto de todas as direções viáveis em relação ao conjunto Ω no ponto $\bar{x} \in \Omega$.

Definição 10 (Cone de direções viáveis linearizadas) *Considerando as restrições do problema (2.8), defini-se o cone de direções viáveis linearizadas em $\bar{x} \in \Omega$ por:*

$$\mathscr{V}_\Omega(\bar{x}) = \{d \in \mathbb{R}^n | \nabla c_i(\bar{x})^T d = 0, i \in \mathscr{E}, \nabla c_i(\bar{x})^T d \leq 0, i \in I(\bar{x}) \setminus \mathscr{E}\}.$$

Geometricamente, uma direção $d \in \mathscr{V}_\Omega(\bar{x})$ se e somente se d é tangente a todas as restrições de igualdade (forma um ângulo de 90 graus com os gradientes das restrições de igualdade) e se d faz um ângulo obtuso com os gradientes das restrições de desigualdade *ativas*. Na figura ao lado, $\mathscr{V}(\bar{x})$ é composto pelas direções d_1 e d_2, e $\mathscr{C}(\bar{x})$ é composto apenas pela direção d_1.



Para apresentarmos as condições de otimalidade para o problema restrito, isto é, quando Ω é definido através de funções, temos que definir o *cone tangente*.

Definição 11 *Dado um conjunto $\Omega \subset \mathbb{R}^n$ e um vetor $\bar{x} \in \Omega$, uma direção d é dita* tangente *a Ω em $\bar{x}$ se ou $d = 0$ ou existe uma seqüência $\{x^k\} \subset \Omega$ tal que $x^k \neq \bar{x}$ para todo k e*

$$x^k \to \bar{x}, \qquad \frac{x^k - \bar{x}}{\|x^k - \bar{x}\|} \to \frac{d}{\|d\|}.$$

O conjunto de todas direções tangentes de Ω em $\bar{x}$ é chamado de cone tangente *de Ω em $\bar{x}$, e é denotado por $\mathscr{T}_\Omega(\bar{x})$.*

Em outras palavras, se d é uma direção que "penetra" ou tangencia Ω a partir de $\bar{x}$, dizemos que d é tangente. Em razão disso,

$$\mathscr{C}_\Omega(\bar{x}) \subset \mathscr{T}_\Omega(\bar{x}) \subset \mathscr{V}_\Omega(\bar{x}).$$

2.3 Condições de otimalidade para problemas com restrições

Pode-se reescrever a condição necessária de primeira ordem para o problema (2.10) em função das direções tangentes.

Teorema 5 (Condição Necessária de Primeira Ordem – direções tangentes)[2] *Se $\bar{x} \in \Omega$ é um minimizador local do problema (2.10), então para qualquer direção d tangente a Ω em $\bar{x}$,*

$$\nabla f(\bar{x})^T d \geq 0.$$

O Teorema 5 nos diz que uma direção tangente não pode ser uma direção de descida. Ou ainda, se $\bar{x}$ é um minimizador local, então a intersecção do cone tangente em $\bar{x}$ com o cone de direções de descida em $\bar{x}$ é vazia.

Outra abordagem interessante deste resultado é feita através da relação entre os cones tangente, polar e normal. Estes, por sua vez, determinam a condição de qualificação para enunciar o teorema de KKT na forma cônica.

[2] Outra formulação das condições de otimalidade é apresentada em Izmailov e Solodov (2005), em que se define o cone de Bouligand.

2.3.1 Teorema de Karush - Kuhn - Tucker na forma cônica

Definição 12 (Convexidade)[3] *Um conjunto $\Omega \subset \mathbb{R}^n$ é chamado* conjunto convexo *se para quaisquer $x \in \Omega$, $y \in \Omega$ e $\alpha \in [0,1]$, tem-se*

$$\alpha x + (1-\alpha)y \in \Omega.$$

O ponto $\alpha x + (1-\alpha)y$, em que $\alpha \in [0,1]$, é denominado a combinação convexa *de x e y com parâmetro α.*

Definição 13 *Dados $x^1,\ x^2, \ldots,\ x^p$ pertencentes a $\mathbb{R}^n$, $x \in \mathbb{R}^n$ é uma combinação convexa de $x^1,\ x^2, \ldots,\ x^p$ se, e somente se, pode-se escrever*

$$x = \sum_{i=1}^{p} \lambda_i x^i,\ \textit{com}\ \lambda_i \geq 0,\ i = 1, \ldots, p\ \ \textit{e}\ \sum_{i=1}^{p} \lambda_i = 1.$$

Definição 14 *Envoltória convexa de um conjunto $\Omega \subset \mathbb{R}^n$ é o menor conjunto convexo que o contém, denotado por* conv Ω.

A envoltória convexa de Ω é o conjunto de todas as combinações convexas de pontos de Ω:

$$\text{conv}\,\Omega = \Big\{ \sum_{i=1}^{p} \lambda_i x^i \mid p \in \mathbb{N},\ x^i \in \Omega,\ \lambda_i \in [0,1],\ i = 1, \ldots, p,\ \sum_{i=1}^{p} \lambda_i = 1 \Big\}.$$

Teorema 6 (Teorema de Carathéodory) *Seja $x \in \mathbb{R}^n$ uma combinação convexa de pontos do conjunto $\Omega \subset \mathbb{R}^n$. Então existem $x^i \in \Omega$ e $\lambda_i \in \mathbb{R}_+$, $i = 1, \ldots, n+1$, tais que*

$$x = \sum_{i=1}^{n+1} \lambda_i x^i \quad \text{e} \quad \sum_{i=1}^{n+1} \lambda_i = 1.$$

Definição 15 *A envoltória cônica*[4] *de um conjunto $\Omega \subset \mathbb{R}^n$ é o menor cone convexo que contém Ω.*

Em outras palavras, a envoltória cônica pode ser vista como o conjunto de todas as *combinações cônicas* de pontos de Ω,

$$\text{cone}(\Omega) = \Big\{ \sum_{i=1}^{p} \lambda_i x^i \mid p \in \mathbb{N},\ x^i \in \Omega,\ \lambda_i \geq 0,\ i = 1, \ldots, p \Big\}.$$

Além disso, se Ω é compacto, a envoltória cônica é um cone fechado.

[3] Neste texto, a teoria de convexidade serve de base para um melhor entendimento das propriedades a seguir, além disso, auxilia nas demonstrações dos teoremas de separação do próximo capítulo. Esta teoria pode ser encontrada em Izmailov e Solodov (2005), entre outros.

[4] Izmailov e Solodov (2005), denomina este conjunto por *fecho cônico*.

Definição 16 *Dizemos que um cone $K \subset \mathbb{R}^n$ tem um número finito de geradores quando K é a envoltória cônica de um número finito de pontos em $\mathbb{R}^n$, ou seja, se existem $p \in \mathbb{N}$ e $v^i \in \mathbb{R}^n$, $i = 1, \ldots, p$, tais que*

$$K = \text{cone}\big(\{v^i,\ i = 1, \ldots, p\}\big).$$

Lema 3 *Qualquer cone que tem um número finito de geradores é convexo e fechado.*

Definição 17 *Dado um cone $C \in \mathbb{R}^n$, o* cone polar *de C, denotado por $P(C)$, é*

$$P(C) = \{v \in \mathbb{R}^n \mid v^T x \leq 0,\ \forall x \in C\}.$$

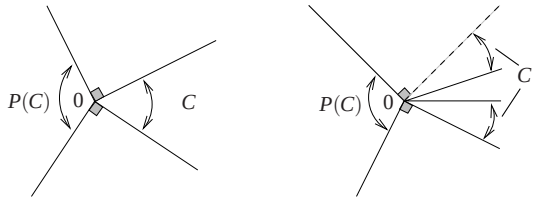


Figura 2.2: O cone polar $P(C)$ é convexo e fechado para qualquer cone C.

Lema 4 *Dado um cone $C \subset \mathbb{R}^n$, as seguintes afirmações são verdadeiras.*

1. *$P(C)$ é convexo e fechado;*
2. *$P(C) = P(\overline{C})$, em que $\overline{C}$ denota o fecho do conjunto C.*

Lema 5 (Lema de Farkas para cones) *Seja $C \subset \mathbb{R}^n$ um cone fechado e convexo. Então,*

$$P(P(C)) = C.$$

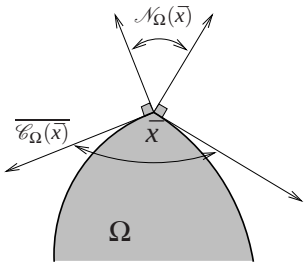
Definição 18 *Sejam $\Omega \subset \mathbb{R}^n$ um conjunto convexo e $\bar{x} \in \Omega$. O cone normal (cone de direções normais) no ponto $\bar{x}$ em relação ao conjunto Ω é dado por*

$$\mathcal{N}_\Omega(\bar{x}) = \{d \in \mathbb{R}^n \mid \langle d, x - \bar{x} \rangle \leq 0\ \forall x \in \Omega\}.$$

A convexidade imposta no conjunto Ω, na definição de *cone normal*, implica que as restrições de igualdade do conjunto Ω são lineares. Como este é o caso que nos interessa neste texto, esta exigência nos garante a condição de qualificação definida adiante.

A condição de qualificação

Lema 6 *Suponha que $\bar{x}$ é solução de (2.10). Então $\overline{\mathscr{C}_\Omega(\bar{x})} = P(\mathcal{N}_\Omega(\bar{x}))$.*



O lema (6) afirma que toda direção viável em Ω a partir de $\bar{x}$ faz ângulo obtuso com $\nabla c_i(\bar{x}),\ i \in \mathscr{I}(\bar{x})$. Mas a recíproca nem sempre é verdadeira; há exemplos de direções que não são viáveis e fazem ângulo reto com gradientes. Neste texto vamos fazer a hipótese de que isto não ocorre.

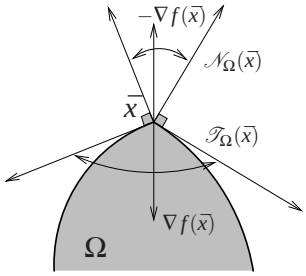
Seja $\bar{x}$ solução ótima de (2.10). Dizemos que o problema satisfaz a condição de qualificação se

$$\overline{\mathscr{C}_\Omega(\bar{x})} = P(\mathscr{N}_\Omega(\bar{x})), \tag{2.12}$$

isto é, a aderência do cone de direção viáveis coincide com o polar do cone normal.

É importante ressaltar que ao utilizar a condição de qualificação, devemos antes caracterizar o cone tangente para o problema (2.8) em um ponto $\bar{x} \in \Omega$. Vimos que o cone $\mathscr{V}_\Omega(\bar{x})$ tem uma descrição algébrica simples, enquanto o cone tangente é de caracterização difícil. Sob condições bastantes simples os dois cones coincidem, essas condições serão apresentadas na próxima secção.

Pelas definições, sabemos que $\mathscr{T}_\Omega(\bar{x}) = P(\mathscr{N}_\Omega(\bar{x}))$ e impondo a condição $\mathscr{T}_\Omega(\bar{x}) = \mathscr{V}_\Omega(\bar{x})$ na equação (2.12), temos que



$$\overline{\mathscr{C}_\Omega(\bar{x})} = \mathscr{T}_\Omega(\bar{x}) = \mathscr{V}_\Omega(\bar{x}) = P(\mathscr{N}_\Omega(\bar{x})).$$

Teorema 7 *Sejam Ω um conjunto convexo e $\bar{x} \in \Omega$. Então*

$$P(\mathscr{T}_\Omega(\bar{x})) = P(\mathscr{V}_\Omega(\bar{x})) = P(\text{cone}(\Omega - \bar{x})) = \mathscr{N}_\Omega(\bar{x}).$$

Teorema 8 *Considere o problema (2.10) e suponha que $\bar{x}$ é solução ótima satisfazendo a condição de qualificação. Então,*

$$-\nabla f(\bar{x}) \in \mathscr{N}_\Omega(\bar{x}).$$

Dem. Com as hipóteses do teorema e pelo Teorema (5), temos que $-\nabla f(\bar{x}) \in P(\mathscr{T}_\Omega(\bar{x}))$. Pela condição de qualificação, segue que,

$$-\nabla f(\bar{x}) \in P(P(\mathscr{N}_\Omega(\bar{x}))).$$

Sabemos que $\mathscr{N}_\Omega(\bar{x})$ é um cone fechado e convexo, em razão disso, segue do lema de Farkas que

$$\mathscr{N}_\Omega(\bar{x}) = P(P(\mathscr{N}_\Omega(\bar{x}))).$$

■

2.3.2 Teorema de Karush - Kuhn - Tucker na forma algébrica

Considere $\bar{x}$ um minimizador do problema (2.8), satisfazendo a condição de qualificação. Para enunciar o teorema de KKT em forma algébrica, basta tomar o cone normal da seguinte forma:

$$\mathscr{N}_\Omega(\bar{x}) = P(\mathscr{V}_\Omega(\bar{x})).$$

Assim, do Teorema (8), segue que

$$-\nabla f(\bar{x}) = \sum_{j\in\mathscr{E}} \lambda_j \nabla c_j(\bar{x}) + \sum_{i\in\mathscr{I}} \lambda_i \nabla c_i(\bar{x}), \quad \lambda_i \geq 0,$$

ou seja, $-\nabla f(\bar{x}) = \sum_{i\in I(\bar{x})} \lambda_i \nabla c_i(\bar{x})$, em que $\lambda_i \geq 0$ se $i \in \mathscr{I}$.

Note que podemos substituir o somatório sobre $I(\bar{x})$ por $\sum_{i=p+1}^{m} \lambda_i \nabla c_i(\bar{x})$, em que $\lambda_i \geq 0$. Basta fazer $\lambda_i = 0$ para $i \notin I(\bar{x})$, assim, garantimos que

$$\lambda_i c_i(\bar{x}) = 0, \quad i = 1, \ldots, m$$

e consequentemente,

$$-\nabla f(\bar{x}) = \sum_{i=1}^{m} \lambda_i \nabla c_i(\bar{x}), \quad \text{em que } \lambda_i \geq 0, \text{ se } i > p.$$

Juntando as condições de viabilidade e otimalidade do problema (2.8), chegamos ao teorema de Karush - Kuhn - Tucker.

Teorema 9 (Condições Necessárias de Otimalidade de KKT) *Suponha que $\bar{x}$ é minimizador local do problema (2.8) e que a condição de qualificação é satisfeita. Então existem escalares $\bar{\lambda}_1, \bar{\lambda}_2, \ldots, \bar{\lambda}_m$ tais que*

$$\begin{aligned}
\nabla f(\bar{x}) + \sum_{i=1}^{m} \bar{\lambda}_i \nabla c_i(\bar{x}) &= 0 && && (2.13)\\
c_{\mathscr{I}}(\bar{x}) &\leq 0 && && (2.14)\\
c_{\mathscr{E}}(\bar{x}) &= 0 && && (2.15)\\
\bar{\lambda}_i &\geq 0 && \textit{para } i \in \mathscr{I} && (2.16)\\
\bar{\lambda}_i c_i(\bar{x}) &= 0 && \textit{para } i \in \mathscr{E} \cup \mathscr{I} && (2.17)
\end{aligned}$$

Os escalares $\bar{\lambda}_1, \ldots, \bar{\lambda}_m$ são chamados de *Multiplicadores de Lagrange*. Os multiplicadores de Lagrange têm informações importantes sobre as restrições.

As condições (2.13) – (2.17) são denominadas *condições de Karush-Kuhn-Tucker*, ou *condições de KKT*. Note que a condição de complementariedade (2.17), como vimos anteriormente, implica que os multiplicadores de Lagrange correspondentes às restrições inativas sejam nulos.

Definição 19 (Condição de complementariedade) *Dada uma solução local $\bar{x}$ do problema (2.8) e um vetor $\bar{\lambda}$ satisfazendo o Teorema 9, dizemos que a condição de complementariedade (2.17) é estrita, se exatamente um dos $\bar{\lambda}_i$ e $c_i(\bar{x})$ é zero para cada índice $i \in \mathscr{I}$. Em outras palavras, temos que $\bar{\lambda}_i > 0$ para cada $i \in \mathscr{I} \cap I(\bar{x})$.*

Muitos métodos procuram pontos e multiplicadores que satisfaçam as equações de KKT (2.13)-(2.17). Entretanto, isto não garante que o ponto encontrado seja um minimizador local. Os pontos que satisfazem as equações são conhecidos como *pontos estacionários*.

Teorema 10 (Condições Suficientes de Segunda Ordem de KKT) *Seja $\bar{x} \in \Omega$ um ponto estacionário do problema (2.8) e $\bar{\lambda} \in \mathbb{R}^m$ um vetor de multiplicadores de Lagrange associados a $\bar{x}$ satisfazendo as equações de KKT (2.13)-(2.16). Se*

$$d^T \left(\nabla^2 f(\bar{x}) + \sum_{i=1}^{m} \bar{\lambda}_i \nabla^2 c_i(\bar{x}) \right) d > 0, \tag{2.18}$$

para todo $d \in \mathbb{R}^n$, não nulo, tal que

$$\nabla c_i(\bar{x})^T d = 0, \qquad \forall i \in I(\bar{x}), \text{ com } \bar{\lambda}_i > 0,$$

então $\bar{x}$ é um minimizador local estrito para o problema (2.8).

Para enunciar as condições de otimalidade de um problema de Programação Linear, na seção 2.5, é conveniente escrever as condições necessárias em termos da *função Lagrangeano*, e além disso, apresentar condições de otimalidade para problemas convexos.

Definição 20 *A função Lagrangeano associada ao problema (2.8) é dada por $\mathscr{L} : \mathbb{R}^{n+m} \to \mathbb{R}$,*

$$\mathscr{L}(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i c_i(x), \tag{2.19}$$

em que m é a cardinalidade do conjunto $\mathscr{E} \cup \mathscr{I}$. Além disso, $x \in \mathbb{R}^n$ e $\lambda \in \mathbb{R}^m$ é o vetor dos multiplicadores de Lagrange.

A primeira equação das condições de KKT (2.13) pode ser escrita como

$$\nabla_x \mathscr{L}(x, \lambda) = \nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla c_i(x) = 0.$$

A matriz Hessiana da função Lagrangeano em relação a x é dada por:

$$\nabla^2_{xx}\mathscr{L}(x,\lambda) = \nabla^2 f(x) + \sum_{i=1}^{m} \lambda_i \nabla^2 c_i(x),$$

ou seja, pode-se reescrever (2.18) na forma:

$$d^T \nabla^2_{xx}\mathscr{L}(\bar{x},\bar{\lambda})d > 0.$$

Definição 21 *Se $\Omega \subset \mathbb{R}^n$ é um conjunto convexo, diz-se que a função $f : \Omega \to \mathbb{R}$ é convexa em Ω quando para quaisquer $x \in \Omega$, $y \in \Omega$, e $\alpha \in [0,1]$, tem-se*

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

Teorema 11 (Condições necessárias e suficientes para otimização convexa) *Sejam $\Omega \subset \mathbb{R}^n$ um conjunto convexo e $f : D \to \mathbb{R}$ uma função convexa e diferenciável no conjunto aberto D que contém Ω.*

Então $\bar{x}$ é um minimizador de f em Ω se, e somente se,

$$\nabla f(\bar{x})^T (x - \bar{x}) \geq 0, \quad \forall x \in \Omega, \tag{2.20}$$

ou, equivalentemente,

$$\nabla f(x)^T (x - \bar{x}) \geq 0, \quad \forall x \in \Omega. \tag{2.21}$$

Na seção seguinte, serão mostradas as condições para que os multiplicadores de Lagrange existam, isto é, as condições de qualificação das restrições.

2.4 Condições de Qualificação

As condições de qualificação são condições "simples" que garantem que o cone de direções viáveis linearizadas $\mathscr{V}$ seja idêntico ao cone tangente $\mathscr{T}$ num mesmo ponto $\bar{x}$ viável.

Definição 22 (Condição de Qualificação de Mangasarian-Fromovitz – MFCQ) *Dado um ponto viável $\bar{x}$ e o conjunto ativo $I(\bar{x})$ definido em (2.11), dizemos que a condição de qualificação de Mangasarian-Fromovitz (MFCQ) é satisfeita se existe um vetor $d \in \mathbb{R}^n$ tal que*

$$\begin{aligned} \nabla c_i(\bar{x})^T d &< 0, \qquad \forall i \in I(\bar{x}) \cap \mathscr{I}, \\ \nabla c_i(\bar{x})^T d &= 0, \qquad \forall i \in \mathscr{E}, \end{aligned}$$

e o conjunto de gradientes das restrições de igualdade $\{\nabla c_i(\bar{x}),\ i \in \mathscr{E}\}$ *é Linearmente Independente.*

Teorema 12 *Se a condição de qualificação de Mangasarian-Fromovitz é satisfeita em* $\bar{x}$*, então* $\mathscr{V}_\Omega(\bar{x}) = \mathscr{T}_\Omega(\bar{x})$.

Segundo as condições de qualificação de Mangasarian-Fromovitz, existem multiplicadores de Lagrange para o problema (2.8) em um ponto $\bar{x}$ minimizador local se houver uma direção que "penetra estritamente" no cone da direções viáveis de primeira ordem e a matriz Jacobiana das restrições de igualdade tiver posto completo.

Proposição 1 (Condição dual de Mangasarian-Fromovitz) *Seja* $\bar{x} \in \Omega$*, em que* Ω *é dado por (2.9). O ponto* $\bar{x}$ *satisfaz a condição de regularidade de Mangasarian-Fromovitz (22) se, e somente se, o seguinte sistema na variável* $\lambda \in \mathbb{R}^{|I(\bar{x})|}$,[5]

$$0 = \sum_{i \in I(\bar{x})} \lambda_i \nabla c_i(\bar{x}), \quad \lambda_i \geq 0,\ (i \in \mathscr{I} \mid c_i(\bar{x}) = 0), \tag{2.22}$$

tem unicamente a solução nula.

Outra condição de qualificação que garante a igualdade entre os cones tangentes e o cone das direções viáveis de primeira ordem é a condição de independência linear (LICQ, do inglês *Linear Independence Contraint Qualification*).

Definição 23 (Condição de Independência linear – LICQ) *Dados* $\bar{x}$ *e o conjunto ativo* $I(\bar{x})$ *definido por (2.11), dizemos que é satisfeita a condição de qualificação de independência linear se o conjunto* $\{\nabla c_i(\bar{x}),\ i \in I(\bar{x})\}$ *é Linearmente Independente(LI).*

Teorema 13 *Se a LICQ é satisfeita em* $\bar{x} \in \Omega$*, então a condição de Mangasarian-Fromovitz também é satisfeita em* $\bar{x}$.

Dada uma solução $\bar{x}$ do problema (2.8), podem existir muitos vetores $\bar{\lambda}$ tais que as condições de KKT sejam satisfeitas. No entanto, se a condição LICQ é satisfeita o vetor dos multiplicadores de Lagrange é único.

As condições de qualificação vão garantir que existem multiplicadores de Lagrange para as condições de otimalidade de KKT. Vale novamente lembrar que as condições necessárias não dão garantia de que um ponto seja um minimizador local para o problema (2.8). Mas se o ponto é um minimizador local, este deve satisfazer as condições necessárias.

[5] $|V|$ representa a cardinalidade do conjunto V.

Definição 24 (Condição de Slater) *Considere o problema (2.8), em que as restrições de igualdade são funções afins e as restrições de desigualdade são funções convexas. Dizemos que a condição de Slater é satisfeita se existe* $\bar{x} \in \mathbb{R}^n$ *tal que*

$$c_i(\bar{x}) < 0, \qquad \forall i \in \mathscr{I},$$
$$c_i(\bar{x}) = 0, \qquad \forall i \in \mathscr{E}.$$

Lema 7 *Seja* f *convexa definida no problema (2.10), em que* Ω *satisfaz as hipóteses da condição de Slater. Se* $\bar{x}$ *é um minimizador local em* Ω*, então* $\bar{x}$ *é um minimizador global em* Ω*.*

No caso de um problema de programação convexa, como visto no Teorema 11, as condições de otimalidade de KKT tornam-se suficientes. Com efeito, dizer que $\bar{x}$ é um ponto estacionário que satisfaz a condição de Slater, implica que $\bar{x}$ é uma solução do problema convexo.

2.5 Condições de otimalidade para problemas de Programação Linear

Relembramos que chamamos de um problema de programação linear um problema de otimização, em que a função objetivo é linear e o conjunto viável é um poliedro (ou seja, conjunto de soluções de um sistema de equações e inequações lineares). Considere o problema de programação linear:

$$\begin{aligned} \text{minimizar} \quad & c^T x \\ \text{sujeito a} \quad & Ax = b, \\ & x \geq 0. \end{aligned} \tag{2.23}$$

em que $A_{m\times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, sendo $m < n$. Esse formato é bastante geral, uma vez que restrições de desigualdade podem ser reduzidas a igualdades com a introdução de variáveis de folga. Em particular tal problema é convexo. Logo, toda solução local é global. Mais ainda, todo ponto estacionário é uma solução.

As condições de otimalidade do problema (2.23) podem ser obtidas a partir da teoria vista anteriormente na seção 2.3, considerando apenas as condições de KKT de primeira ordem, pois a convexidade do problema garante que estas condições sejam suficientes para um mínimo global. Além disso, aplicando o Teorema 7, a condição de qualificação do problema de programação linear é satisfeita trivialmente.

Os multiplicadores de Lagrange para o problema (2.23) são apresentados através de dois vetores y e s, em que $y \in \mathbb{R}^m$ é o vetor dos multiplicadores para as restrições de igualdade $Ax = b$,

enquanto $s \in \mathbb{R}^n$ é o vetor dos multiplicadores das restrições $-x_i \leq 0,\ i = 1,\ldots,n.$ Usando a definição (20), podemos escrever a função lagrangeano associado ao problema (2.23) como

$$\mathscr{L}(x,y,s) = c^T x - y^T(Ax-b) - s^T x. \tag{2.24}$$

Aplicando o Teorema de KKT (9), as condições de primeira ordem para que $\bar{x}$ seja solução do problema (2.23) são de que existam os vetores y e s tais que

$$A^T y + s = c, \tag{2.25}$$
$$Ax = b, \tag{2.26}$$
$$x \geq 0, \tag{2.27}$$
$$s \geq 0, \tag{2.28}$$
$$x_i s_i = 0, \qquad i = 1,2,\ldots,n. \tag{2.29}$$

A não negatividade das variáveis x e s, condições (2.27) e (2.28), implica que a condição de complementariedade (2.29) seja escrita de forma equivalente por $x^T s = 0$.

Seja $(\bar{x},\ \bar{y},\ \bar{s})$ um vetor que satisfaça as condições de otimalidade (2.25) a (2.29). Combinando as equações (2.25), (2.26) e (2.29), temos que

$$c^T\bar{x} = (A^T\bar{y}+\bar{s})^T\bar{x} = (A\bar{x})^T\bar{y} = b^T\bar{y}. \tag{2.30}$$

As equações acima nos conduzem à teoria de dualidade, em que formulamos dois problemas de otimização intimamente ligados pelas condições de KKT, apresentado pelo Teorema 14 da próxima seção.

2.6 O problema dual

Considere os seguintes problemas de programação linear:

$$\text{(P)}\quad \begin{array}{ll} \text{minimizar} & c^T x \\ \text{sujeito a} & Ax = b, \\ & x \geq 0. \end{array} \qquad \text{e} \qquad \text{(D)}\quad \begin{array}{ll} \text{maximizar} & b^T y \\ \text{sujeito a} & A^T y + s = c, \\ & s \geq 0, \end{array}$$

em que $c \in \mathbb{R}^n,\ b \in \mathbb{R}^m,\ A \in \mathbb{R}^{m\times n}$ e $s \in \mathbb{R}^n$.

Chamamos o problema (P) de problema primal, por outro lado, denominamos o problema (D) como sendo o problema dual associado ao problema (P). Estes dois problemas estão forte-

mente relacionados, como veremos nos resultados seguintes.

Podemos reformular o problema (D) como

$$\begin{aligned} &\text{minimizar} && -b^T y, \\ &\text{sujeito a} && A^T y + s = c, \\ & && s \geq 0. \end{aligned} \tag{2.31}$$

A função lagrangeano associada ao problema (2.31) é

$$\bar{\mathscr{L}}(x,s,y) = -b^T y + x^T (A^T y + s - c).$$

Observe que as condições de KKT do problema linear (P), ou (2.23), ou (2.31) são dadas por:

$$\begin{aligned} A^T y + s &= c, \\ Ax &= b, \\ x &\geq 0, \\ s &\geq 0, \\ x_i s_i &= 0, \qquad i = 1,2,\ldots,n. \end{aligned} \tag{2.32}$$

Teorema 14 *As condições de KKT são satisfeitas para $\bar{x} \in \mathbb{R}^n$, $\bar{y} \in \mathbb{R}^m$, e $\bar{s} \in \mathbb{R}^n$ se e somente se*

1. *$\bar{x}$ é solução ótima do problema (P);*
2. *$(\bar{y}, \bar{s})$ é solução ótima do problema (D).*

Dem. Aplicando as condições de otimalidade a (P) com multiplicadores $y \in \mathbb{R}^m$ associados às restrições de igualdade e $s \in \mathbb{R}^n$ associados às restrições de desigualdade, $-x \leq 0$, obtém-se imediatamente as condições de KKT (2.32).

Aplicando as condições de otimalidade ao problema:

$$\begin{aligned} (\bar{D}) \quad &\text{minimizar} && -b^T y, \\ &\text{sujeito a} && A^T y \leq c, \end{aligned} \tag{2.33}$$

com multiplicadores $x \in \mathbb{R}^n$, obtém-se

$$\begin{aligned} -b + Ax &= 0 \\ x &\geq 0, \\ (A^T y - c)_i \, x_i &= 0, \qquad i = 1,2,\ldots,n. \end{aligned} \tag{2.34}$$

Fazendo $s = c - A^T y$, vemos que essas condições coincidem com as condições de KKT (2.32), e $(\bar{D})$ coincide com (D), completando a demonstração. ■

Retomando as equações (2.30), temos que as funções objetivo dos problemas (P) e (D), possuem o mesmo valor nas soluções que satisfazem as condições de KKT. A seguir são apresentados outros resultados importantes relacionados à dualidade.

Lema 8 *Se x é um ponto viável para o problema primal e (y,s) é viável para o problema dual, então*

$$\Delta = c^T x - b^T y = x^T s \geq 0. \tag{2.35}$$

Dem. Suponha que as condições de viabilidade são satisfeitas por $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$ e $s \in \mathbb{R}^n$. Então,

$$\begin{aligned} \Delta = c^T x - b^T y &= c^T x - (Ax)^T y \\ &= c^T x - x^T A^T y \\ &= x^T (c - A^T y) \\ &= x^T s \geq 0, \quad (\text{pois } x \geq 0,\ s \geq 0). \end{aligned}$$

■

O valor $\Delta = c^T x - b^T y$ é chamado de "salto de dualidade" ou "gap de dualidade".

O Lema 8 é conhecido como "dualidade fraca": o salto é sempre não negativo para quaisquer x, y, s viáveis.

Teorema 15 ("Dualidade forte") *Considere o problema primal (P) e o problema dual (D). Uma das seguintes asserções é verdadeira:*

- *(P) e (D) são inviáveis*[6]*;*
- *(P) é ilimitado e (D) é inviável;*
- *(P) é inviável e (D) é ilimitado;*
- *(P) e (D) são viáveis, suas soluções ótimas tem o mesmo valor ($c^T \bar{x} = b^T \bar{y}$). Quaisquer soluções ótimas $\bar{x}$, $\bar{y}$, $\bar{s}$ satisfazem (KKT).*

Há uma grande classe de algoritmos, os métodos de trajetória central, que trabalham simultaneamente com variáveis primais duais. Nesses algoritmos cada iteração encontra x e s tais

[6] Dizemos que um problema é inviável quando seu conjunto viável é vazio.

que $x, s > 0,\ A^T y + s = c,\ Ax = b$, mas não se satisfaz $x^T s = 0$ (pois então teríamos soluções ótimas). Faz-se o seguinte: para um número $\mu > 0$, procura-se obter $x_i s_i \cong \mu,\ i = 1, \ldots, n$. É fácil ver que então

$$\Delta = x^T s \cong n\mu,$$

fazendo $\mu \to 0$, obtém-se $x^T s \to 0$, o salto de dualidade tende a zero.

Por outro enfoque o algoritmo Simplex é um algoritmo primal, porque trabalha com as variáveis $x \in \mathbb{R}^n$, mas cada iteração opera construindo uma solução dual. Veremos que a solução dual s construída a cada iteração não é viável (a não ser quando termina a busca – critério de parada), pois $s_i < 0$ para algum $i \in \{1, \ldots, n\}$.

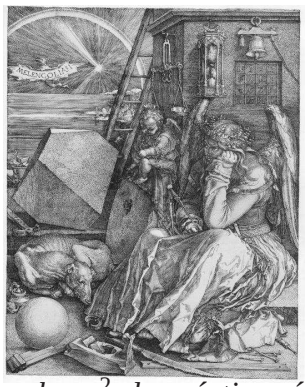
Por fim, analisando as condições de KKT (2.32) concluímos que:

- o algoritmo de trajetória central opera com KKT sem a condição $x^T s = 0$;
- o algoritmo Simplex opera com KKT sem a condição $s \geq 0$.

Por tais razões, o problema dual possui fundamental importância neste texto, de fato retomamos a relação primal-dual no estudo do algoritmo Simplex no capítulo 4.

3 Poliedro

O fascínio de matemáticos e filósofos pelos poliedros vem de longa data, destacando-se significativamente o trabalho de Euclides, na Grécia Antiga. Para alguns, sua obra "Os Elementos", em 13 volumes, teria o estudo dos poliedros como motivação principal.



O prazer estético que os poliedros proporcionam vem não somente de sua visualização mas também de simples e surpreendentes teoremas descobertos após o Renascimento[1]*. Destaca-se o Teorema de Euler, afirmando que a soma do número de vértices com o número de faces excede em duas unidades o número de arestas, bastando a hipótese de que o poliedro seja equivalente, no sentido topológico, a uma esfera (intuitivamente, significa que podemos "inflar"o poliedro até que ele se torne uma esfera). Para poliedros convexos vale a Fórmula de Descartes, garantindo que a soma das deficiências angulares*[2] *dos vértices é sempre igual a* 4π (COLLI, 2003).

"Porque Poliedro?" Poderíamos responder esta pergunta simplesmente relembrando a formulação (2.2) do problema de otimização, que nos apresenta o conjunto viável Ω como sendo convexo e fechado, em geral poliédrico. Mesmo com este fato, não nos é claro o porquê desta estrutura. Desse modo, este capítulo tem como objetivo responder tal questão, conceituando, ou melhor, estruturando o problema de otimização a ser abordado nos próximos capítulos.

3.1 Definição de poliedro

Para estudarmos os poliedros deveríamos antes *definir* o conceito de poliedro. Surpreendentemente, esta não é uma questão tão simples quanto parece. Muitas definições podem ser adotadas, cada uma englobando uma certa classe de objetos (algumas delas são simplesmente

[1] Embora não pareça, até o presente há pesquisa sendo feita sobre poliedros e politopos, que são sua generalização para dimensões mais altas e para geometrias não euclidianas, como hiperbólicas e a esférica.

[2] A deficiência angular de um vértice é o quanto falta para a soma dos ângulos de faces incidentes naquele vértice atingir 2π.

mais restritivas do que outras). Deve pesar na escolha da definição o que se pretende fazer com ela, ou que objeto se deseja incluir ou excluir da definição. Por estas razões, inicialmente apresentaremos conceitos que nos ajudarão a definir e construir poliedros e além disso, sustentar resultados importantes como por exemplo o teorema da separação e suas consequências.

Definição 25 (Subspaço gerado por um conjunto) *Dado um conjunto $\Omega \subseteq \mathbb{R}^n$, o subespaço gerado por Ω é o menor subespaço que contém Ω.*

Denotamos tal conjunto por $\text{lin}\Omega$. Sabemos que $\text{lin}\Omega$ é o conjunto de todos os vetores que podem ser obtidos como combinações lineares de pontos de Ω,

$$\text{lin}\Omega = \Big\{ \sum_{i=1}^{p} \lambda_i x^i \mid p \in \mathbb{N},\ x^i \in \Omega,\ \lambda_i \in \mathbb{R},\ i = 1, \ldots, p \Big\}.$$

Estamos tomando todas as combinações de elementos de Ω p a p, com $p = 1, 2, \ldots$. Mas, da álgebra linear, sabemos que é suficiente escolher vetores linearmente independentes, portanto basta tomar $p = 1, \ldots, n$, ou equivalentemente, $p = n$.

$$\text{lin}\Omega = \Big\{ \sum_{i=1}^{n} \lambda_i x^i \mid x^i \in \Omega,\ \lambda_i \in \mathbb{R},\ i = 1, \ldots, n \Big\}$$

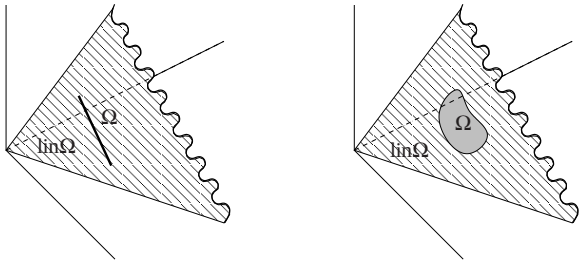


Figura 3.1: $\text{lin}\Omega$ de dimensão 2 gerado pelos conjuntos Ω com dimensões 1 e 2.

Definição 26 (Subespaço afim) *Um subespaço afim em $\mathbb{R}^n$ é um conjunto definido por um sistema de equações $Ax = b$, em que $A \in \mathbb{R}^{m \times n}$ e $b \in \mathbb{R}^m$.*

Definição 27 (Envoltória Afim) *A envoltória afim de um conjunto $\Omega \in \mathbb{R}^n$ é o menor subespaço afim que contém Ω. Denotamos por $\text{aff}\,\Omega$.*

Lema 9 *Dado $\Omega \subseteq \mathbb{R}^n$, $\text{aff}\,\Omega = \Big\{ x = \sum_{i=1}^{p} \lambda_i x^i \mid x^i \in \Omega,\ \lambda_i \in \mathbb{R},\ \sum_{i=1}^{p} \lambda_i = 1 \Big\}$. Basta tomar combinações com $p \leq n+1$ (ou equivalentemente, $p = n+1$).*

Dem. Considere o conjunto $\Omega \subset \mathbb{R}^n$ e seja $x^0 \in \Omega$ um ponto arbitrário. Suponha que a envoltória afim de Ω é dada por $Ax = b$. Note que $Ax^0 = b$, em razão disso, $A(x - x^0) = 0$ se e somente se $Ax = b$. O menor subespaço tal que, para todo $x \in \Omega$ e $x - x^0$ pertencente ao núcleo de A é

$$\text{lin}(\Omega - \{x^0\}) = \Big\{ \sum_{i=1}^{n} \lambda_i (x^i - x^0) \mid x^i \in \Omega,\ \lambda_i \in \mathbb{R},\ i = 1, \ldots, n \Big\}.$$

Portanto, $x \in aff\,\Omega$ se e somente se $x - x^0 = \sum_{i=1}^{n} \lambda_i (x^i - x^0)$, com $x^i \in \Omega,\ \lambda_i \in \mathbb{R},\ i = 1, \ldots, n$. Equivalentemente,

$$x = x^0 - \Big(\sum_{i=1}^{n} \lambda_i \Big) x^0 + \sum_{i=1}^{n} \lambda_i x^i, \qquad \text{ou} \qquad x = \sum_{i=0}^{n} \lambda_i x^i, \text{ com } \lambda_0 = 1 - \sum_{i=1}^{n} \lambda_i,$$

ou finalmente,

$$x = \sum_{i=0}^{n} \lambda_i x^i, \text{ com } \sum_{i=0}^{n} \lambda_i = 1,$$

completando a demonstração. ■

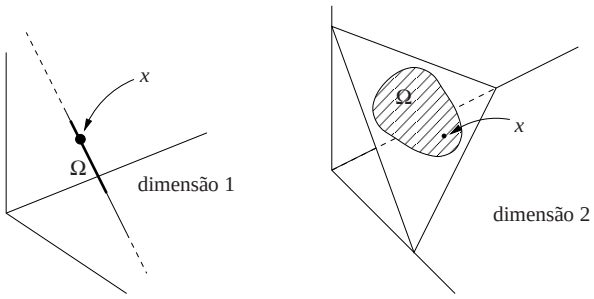


Figura 3.2: $aff\,\Omega$ de dimensões 1 e 2.

Definição 28 *A dimensão de um conjunto $\Omega \subseteq \mathbb{R}^n$ é a dimensão de sua envoltória afim.*

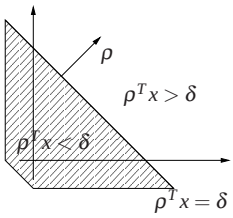
Se $aff\,\Omega$ é definido por $Ax = b$, a dimensão de Ω é a dimensão do núcleo de A, igual a $n - \text{posto}(A)$.

Definição 29 (Interior Relativo) *Dado um conjunto $\Omega \subseteq \mathbb{R}^n$, um ponto $x \in \mathbb{R}^n$ pertence ao interior relativo de Ω se e somente se existe uma vizinhança V de x tal que $V \cap aff\,\Omega \subseteq \Omega$.*

O interior relativo, denotado por $\text{rel int}\Omega$, é o que intuitivamente se considera como o interior de um conjunto de dimensão menor imerso em $\mathbb{R}^n$. A importância da definição (29) vem do fato que uma grande classe de algoritmos encontra a cada iteração um ponto viável, percorrendo uma trajetória no interior relativo do conjunto viável Ω.

Definição 30 (Hiperplano) *Um hiperplano em $\mathbb{R}^n$ é uma variedade linear de dimensão $n-1$. É caracterizado pela equação $\rho^T x = \delta$, em que $\rho \in \mathbb{R}^n$ é a <u>normal</u> do hiperplano e $\delta \in \mathbb{R}$.*

Nesse texto, por abuso de linguagem, o hiperplano é denotado por $\rho^T x = \delta$.



Definição 31 (Semi-espaços) *Um hiperplano $\rho^T x = \delta$ divide o $\mathbb{R}^n$ em um semi-espaço fechado $\rho^T x \leq \delta$ e um semi-espaço aberto $\rho^T x > \delta$.*

Definição 32 (Poliedro) *Um conjunto $P \subset \mathbb{R}^n$ é um poliedro se e somente se pode ser representado como uma intersecção finita de semi-espaços fechados.*

Definição 33 (Politopo) *Um politopo é a envoltória convexa de um conjunto finito de pontos em $\mathbb{R}^n$.*

Teorema 16 *Um conjunto $P \subset \mathbb{R}^n$ é um politopo se e somente se é um poliedro limitado.*

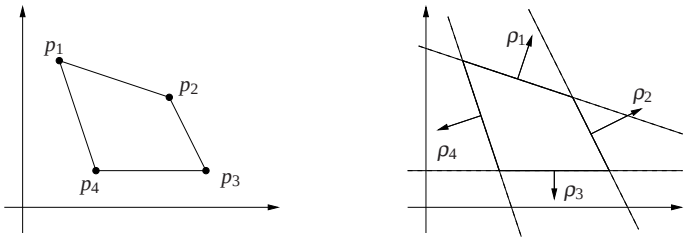


Figura 3.3: Representação de um politopo e um poliedro limitado.

Ao escolhermos as definições de poliedro (32) e politopo (33), a mudança de representação de uma definição para a outra é difícil e não será estudada, por essa razão, não apresentaremos a demonstração do Teorema 16. Na Figura (3.3) é apresentado um polígono definido de duas formas diferentes, uma através de envoltória convexa e outra através de semi-espaços.

Um poliedro P é representado por um sistema de desigualdades definidas por semi-espaços $\rho_i^T x \leq \delta_i,\ i = 1, \ldots, m$, ou seja,

$$P = \{x \in \mathbb{R}^n \mid \rho_i^T x \leq \delta_i,\ i = 1, \ldots, m\}.$$

Construindo a matriz $A^T = [\rho_1\ \rho_2\ \ldots\ \rho_m]_{n \times m}$, o poliedro é descrito por

$$Ax \leq b, \text{ em que } b = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_m \end{bmatrix}.$$

3.2 Teorema da Separação

Primeiramente, apresentaremos uma propriedade básica de conjuntos convexos que será necessária na demonstração do teorema da separação. Os resultados de convexidade e os lemas preliminares ao teorema da separação, aqui apresentados, não serão demonstrados, mas suas demonstrações são encontradas em Izmailov e Solodov (2005).

Lema 10 *Sejam Ω_1, Ω_2 conjuntos convexos do $\mathbb{R}^n$ e a soma de conjuntos definida por:*

$$\Omega_1 + \Omega_2 := \{x \in \mathbb{R}^n \mid x = \omega^1 + \omega^2, \ \ \omega^1 \in \Omega_1 \ e \ \omega^2 \in \Omega_2\}.$$

Então $\Omega_1 + \Omega_2$ é convexo.

Definição 34 *Considere um conjunto $\Omega \subset \mathbb{R}^n$ e um ponto $\bar{x} \in \mathbb{R}^n$. O hiperplano $\rho^T x = \delta$ separa $\bar{x}$ de Ω se e somente se $\rho^T \bar{x} \geq \delta$ e Ω está contido no semi-espaço $\rho^T x \leq \delta$.*

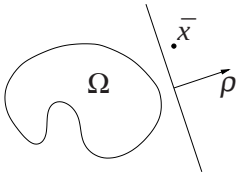


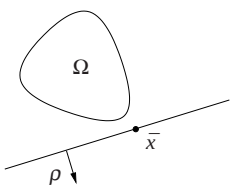
Figura 3.4: Hiperplano separador $\rho^T x = \delta$.

Definição 35 (Hiperplano Suporte) *Se o hiperplano separador "toca" em Ω, isto é, se $\rho^T x = \delta$ para algum $x \in \partial\Omega$, então $\rho^T x = \delta$ é um suporte de Ω.*

Lema 11 (Lema de Minkowski) *Seja $\Omega \subset \mathbb{R}^n$ um conjunto convexo não vazio. Se $\bar{x}$ não pertence ao fecho de Ω, então existem $\rho \in \mathbb{R}^n \setminus \{0\}$ e $\delta \in \mathbb{R}$ tais que*

$$\rho^T \bar{x} = \delta, \quad \rho^T y < \delta \quad \forall y \in \Omega,$$

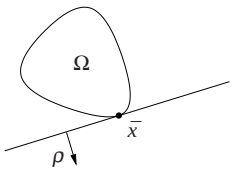
isto é, existe um hiperplano $\rho^T x = \delta$ que separa $\bar{x}$ de Ω.



Note que o hiperplano, $\rho^T x = \delta$, não pode ser um suporte de Ω, pois o hiperplano separador está contido no interior do conjunto $\mathbb{R}^n \setminus \Omega$, isto é, $\rho^T x = \delta$ não "toca" em Ω.

Lema 12 *Seja $\Omega \subset \mathbb{R}^n$ um conjunto convexo e não vazio. Se $\bar{x} \in \partial\Omega$, então existem $\rho \in \mathbb{R}^n \setminus \{0\}$ e $\delta \in \mathbb{R}$ tais que*

$$\rho^T \bar{x} = \delta, \quad \rho^T y \leq \delta \quad \forall y \in \Omega.$$



Por qualquer $\bar{x} \in \partial\Omega$, Ω convexo, passa pelo menos um hiperplano suporte. O conjunto convexo Ω é a intersecção de todos os semi-espaços definidos por esses suportes, se Ω é fechado.

Teorema 17 (Teorema da Separação) *Sejam $\Omega_1 \subset \mathbb{R}^n$ e $\Omega_2 \subset \mathbb{R}^n$ conjuntos convexos não vazios tais que $\Omega_1 \cap \Omega_2 = \emptyset$. Então existem $\rho \in \mathbb{R}^n \setminus \{0\}$ e $\delta \in \mathbb{R}$ tais que*

$$\rho^T x^1 \leq \delta \leq \rho^T x^2 \quad \forall x^1 \in \Omega_1,\ \forall x^2 \in \Omega_2,$$

ou, equivalentemente, existe um hiperplano $\rho^T x = \delta$ que separa Ω_1 de Ω_2.

Dem. Defina $\Omega := \Omega_1 - \Omega_2$. Pelo Lema 10, Ω é convexo, além disso, como $\Omega_1 \cap \Omega_2 = \emptyset$, segue que $0 \notin \Omega$. Assim $0 \in int(\mathbb{R}^n \setminus \Omega)$ ou $0 \in \partial\Omega$.

Para separar 0 do conjunto Ω, no primeiro caso, em que 0 não pertence ao fecho de Ω, usamos o Lema 11, em que $\bar{x} = 0$ e portanto $\delta = 0$. Já no segundo caso utilizamos o Lema 12.

Dos lemas, concluímos que existe $\rho \in \mathbb{R}^n \setminus \{0\}$ tal que $\rho^T x \leq \delta \quad \forall x \in \Omega$. Seja $\bar{x} = 0$, logo $0 = \rho^T \bar{x} = \delta$ e portanto,

$$\rho^T x \leq 0 \quad \forall x \in \Omega,$$

i.e., $\rho^T (x^1 - x^2) \leq 0$ para todos $x^1 \in \Omega_1$, $x^2 \in \Omega_2$. Ou ainda,

$$\rho^T x^1 \leq \rho^T x^2 \quad \forall x^1 \in \Omega_1,\ \forall x^2 \in \Omega_2. \tag{3.1}$$

Em particular, a função $\psi : x \in \mathbb{R}^n \to \rho^T x \in \mathbb{R}$, é limitada inferiormente em Ω_2 e limitada superiormente em Ω. Ainda, da equação (3.1) obtemos que

$$\beta_2 = \inf_{x^2 \in \Omega_2} \rho^T x^2 \geq \sup_{x^1 \in \Omega_1} \rho^T x^1 = \beta_1.$$

Definindo $\delta = \frac{\beta_1 + \beta_2}{2}$, temos que para todos $x^1 \in \Omega_1,\ x^2 \in \Omega_2$

$$\rho^T x^1 \leq \sup_{x^1 \in \Omega_1} \rho^T x^1 = \beta_1 \leq \delta \leq \beta_2 = \inf_{x^2 \in \Omega_2} \rho^T x^2 \leq \rho^T x^2.$$

■

Para garantir a separação estrita, são necessárias hipóteses adicionais. Além disso, a demonstração citada em Izmailov e Solodov (2005) utiliza resultados da projeção de um ponto em um conjunto, os quais não são mencionados neste texto.

Teorema 18 (Teorema da Separação Estrita) *Sejam $\Omega_1 \subset \mathbb{R}^n$ e $\Omega_2 \subset \mathbb{R}^n$ conjuntos convexos, fechados e não vazios. Suponha que um deles também seja limitado. Então $\Omega_1 \cap \Omega_2 = \emptyset$ se, e somente se, existem $\rho \in \mathbb{R}^n \setminus \{0\}$ e $\delta \in \mathbb{R}$ tais que*

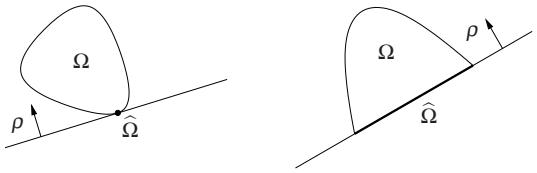
$$\rho^T x^1 < \delta < \rho^T x^2 \quad \forall x^1 \in \Omega_1,\ \forall x^2 \in \Omega_2.$$

A importância dos suportes em otimização fica clara no seguinte teorema. Considere o problema:

$$\text{(P)} \qquad \underset{x \in \Omega}{\text{minimizar}} \ \ \rho^T x$$

em que $\Omega \subset \mathbb{R}^n$ é convexo e fechado.

Teorema 19 *Seja $\widehat{\Omega}$ o conjunto de soluções ótimas de (P). O hiperplano $-\rho^T x = \delta$ é um suporte de Ω se e somente se $\widehat{\Omega}$ é intersecção de Ω com o hiperplano.*

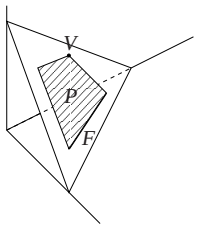


Dem. A demonstração é consequência direta das definições. Como $-\rho^T x = \delta$ é suporte de Ω se e somente se existe $\hat{x} \in \Omega$, tal que $-\rho^T \hat{x} = \delta$ e para todo $x \in \Omega$ temos que $-\rho^T x \leq \delta$. Concluímos que, $(\forall x \in \Omega) \quad \rho^T x \geq \rho^T \hat{x}$. ■

3.3 Faces de um poliedro

Definição 36 *Considere o poliedro $P \subset \mathbb{R}^n$. Um conjunto $F \subset P$ é uma face de P se, e somente se $F = \emptyset$ ou $F = P$ ou $F = P \cap S$, em que S é um suporte de P.*

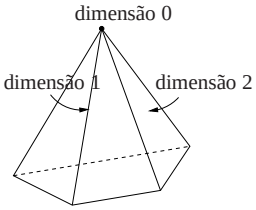
A figura ao lado exibe uma face de dimensão dois, que representa o próprio poliedro *P*, além de quatro faces F de dimensão 1 e quatro faces V de dimensão zero.

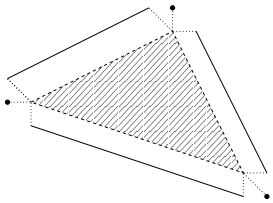


3.3.1 Nomenclatura

Se um poliedro tem dimensão q, através da dimensão, podemos nomear suas faces da seguinte forma:

1) Faces impróprias : todo o poliedro e o conjunto vazio;
2) Faces próprias : faces com dimensão $0, 1, \ldots, q-1$;
3) Faceta : face de dimensão $q-1$;
4) Vértice : face de dimensão zero;
5) Aresta : face de dimensão 1.





Desta forma, o poliedro pode ser decomposto em um conjunto de faces abertas de dimensão $m, m-1, m-2, \ldots, 1, 0$.

Definição 37 *Seja F uma face. Definimos a face aberta de F, denotada por $\overset{\circ}{F}$, como sendo o interior relativo de F.*

Na figura acima, apresentamos um polígono decomposto em suas faces, e cada face com seu interior relativo.

É importante ressaltar algumas propriedades dos poliedros. A primeira delas, é que a intersecção de duas faces de dimensão p é uma face de dimensão $q < p$, além desse fato, a *fronteira relativa* de uma face é composta de faces de dimensão menor.

Os conceitos, até aqui formulados, mostram sua importância ao relacionarmos os poliedros ao problema de otimização.

Considere o problema de programação linear,

$$\underset{x \in P}{\text{minimizar}} \quad c^T x, \tag{3.2}$$

em que, P é um poliedro.

O conjunto solução do problema (3.2) é uma face do poliedro P, denominada a *face ótima*. Portanto, do Teorema 19, concluímos que a face ótima está contida em um hiperplano suporte do poliedro P.

Neste momento, já sabemos caracterizar o conjunto ótimo do problema de programação linear como sendo uma face do poliedro. Em razão disso, se soubéssemos todas as faces, através de tentativas poderíamos solucionar o problema, mas isso é muito dispendioso. Assim

precisamos representar os poliedros de forma coerente para podermos encontrar, ou melhor, solucionar o problema de programação linear com técnicas mais eficientes.

3.4 Representações de poliedros

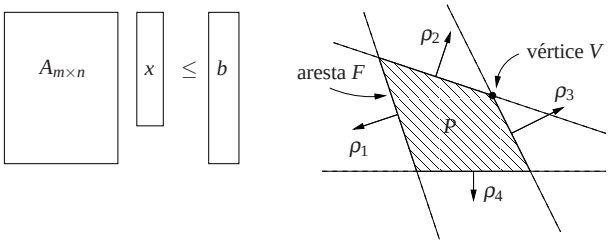
Existem três maneiras usuais para representar poliedros, e pode-se sempre passar de uma representação a outra. Elas são:

1. Desigualdades – Formato Dual;
2. Desigualdades com variáveis não negativas – Formato Canônico;
3. Formato Primal – Formato Padrão.

3.4.1 Desigualdades – Formato Dual

$$\boxed{A_{m\times n}\, x \leq b}$$

O formato dual é constituído de um sistema com m desigualdades, em que cada desigualdade representa um semi-espaço fechado. Assim, por definição, a intersecção desses semi-espaços delimita o poliedro. Para obter um poliedro limitado, isto é, um politopo, deve-se ter k linhas Linearmente Independentes tal que $m \geq k > n$.



Uma *face*, no formato dual, é caracterizada por um conjunto de restrições ativas (conjunto ativo). Na figura acima, o poliedro P é representado por $Ax \leq b$, em que

$$A = \begin{bmatrix} \rho_1^T \\ \vdots \\ \rho_4^T \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_4 \end{bmatrix}.$$

A aresta F é dada por $Ax \leq b$ e $\rho_1^T x = \delta_1$, com o conjunto ativo igual a $\{1\}$, isto é, todo x pertencente à faceta F satisfaz essas condições. Da mesma forma o vértice V é dado por

$$Ax \leq b,\ \rho_2^T x = \delta_2 \text{ e } \rho_3^T x = \delta_3, \text{ com conjunto ativo } \{2,3\}.$$

Notação: Dado um conjunto $K \subset \{1,2,\ldots,n\}$, denotamos x_K como sendo vetor de $\mathbb{R}^{|K|}$ composto pelas componentes com índices em K e A_K pela matriz $m \times |K|$ composta pelas colunas com índices em K. Dado $J \subset \{1,2,\ldots,m\}$, A^J é a matriz $|J| \times n$ composta pelas linhas com índices em J.

Lema 13 *Seja P um poliedro, e considere $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$, em que A é $(m \times n)$ e $b \in \mathbb{R}^m$. Então o conjunto $F \subset P$, não vazio, é uma face se, e somente se, existe $J \subset \{1,\ldots,m\}$ tal que*

$$F = P \cap \{x \in \mathbb{R}^n \mid A^J x = b_J\}. \tag{3.3}$$

Definição 38 *Dado um ponto $\bar{x} \in P$, em que $P = \{x \in \mathbb{R}^n \mid Ax \leq b,\ A \in \mathbb{R}^{m \times n}\}$. O conjunto ativo em $\bar{x}$ é $J(\bar{x}) = \{j \mid j \in \{1,\ldots,m\},\ a_j^T \bar{x} = b_j\}$, em que a_j^T é a j-ésima linha de A, além disso, a face de dimensão mínima em $\bar{x}$ é definida por*

$$F(\bar{x}) = \{x \in P \mid A^{J(\bar{x})} x = b_{J(\bar{x})}\}.$$

Definição 39 *Considere $F(\bar{x})$ definida por (38). O interior relativo de $F(\bar{x})$ é definido pelo conjunto*

$$\text{rel int} F(\bar{x}) = \{x \in F(\bar{x}) \mid a_i^T x < b_i,\ i \notin J(\bar{x}),\ i \in \{1,\ldots,m\}\}.$$

Lema 14 *Seja P um poliedro representado por $Ax \leq b$,*

1. *Para qualquer $x \in P$, a face F da equação (3.3) é uma face de dimensão mínima que contém x, e $x \in \text{rel int} F(x)$.*
2. *Dada qualquer face F e qualquer $x \in \text{rel int} F$, $F = F(x)$.*

Definição 40 *Para cada face $F \subset P$, definimos um conjunto ativo $J(F)$ igual a $J(x)$ para qualquer $x \in \text{rel int} F$.*

Definição 41 *Sejam $P = \{x \in \mathbb{R}^n \mid Ax \leq b,\ A \in \mathbb{R}^{m \times n}\}$ e ρ_j^T a j-ésima linha de A. Uma face $F \subset P$ é não degenerada se, e somente se, o conjunto $\{\rho_j \mid j \in J(F)\}$ é linearmente independente, ou equivalentemente, se a matriz $A^{J(F)}$ possui posto $|J(F)|$.*

Definição 42 *Um vértice $\bar{x}$ de um poliedro descrito por $Ax \leq b$ é não degenerado se, e somente se, as normais das inequações ativas em $\bar{x}$ são linearmente independentes.*

Teorema 20 (Vértice no formato dual)[3] *Seja P um poliedro representado por $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$, em que $A \in \mathbb{R}^{m \times n}$ e $b \in \mathbb{R}^m$. Um ponto $\bar{x}$ é vértice do poliedro P se, e somente se, satisfaz $Ax \leq b$ e é a única solução do sistema $A^{J(\bar{x})}x = b_{J(\bar{x})}$, em que $J(\bar{x})$ é o conjunto ativo em $\bar{x}$, com $|J(\bar{x})| \geq n$.*

Note que se $\bar{x}$ é um vértice não degenerado temos que $|J(\bar{x})| = n$, por outro lado, se $\bar{x}$ é degenerado, podemos escolher n linhas linearmente independentes. Utilizaremos esta caracterização de vértice, na próxima seção, para definir todos os vértices de um politopo.

O formato dual é a melhor forma de representar e definir poliedros, mas não será utilizado nos algoritmos deste texto. Entretanto, esta abordagem é adequada para a construção de poliedros, em que as restrições de positividade não são consideradas. Este estudo serve como motivação para trabalhos futuros.

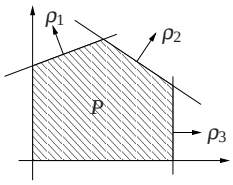
3.4.2 Desigualdades com variáveis não negativas – Formato Canônico

É um caso particular do anterior, em que as restrições $x_i \geq 0$ são separadas das demais, obtendo-se

$$\boxed{\begin{aligned} Ax &\leq b \\ x &\geq 0. \end{aligned}}$$

Este é o formato mais comum em problemas práticos, em que $Ax \leq b$ representa a limitação de recursos em algum processo produtivo e $x \geq 0$ indica que a produção não pode ser negativa.

Na seção 3.5 apresentaremos um programa para a construção de poliedros que utiliza este formato como "*Default*", pois o formato canônico facilita a implementação e a representação geométrica dos poliedros. A figura ao lado, mostra a representação geométrica de um politopo P delimitado por hiperplanos $\rho_i^T x = \delta_i$, com $i = 1, \ldots, 3$. Assim, podemos representar algebricamente o poliedro por



$$\begin{aligned} Ax &\leq b \\ x &\geq 0 \end{aligned}$$

[3] A demonstração do teorema (20) é encontrada em Chvátal (1983). Já os demais resultados desta seção, são apresentados em notas não publicadas do professor Clóvis C. Gonzaga. As demonstrações não serão feitas, visto que resultados similares são apresentados nas próximas seções.

em que a matriz A e o vetor b são dados por

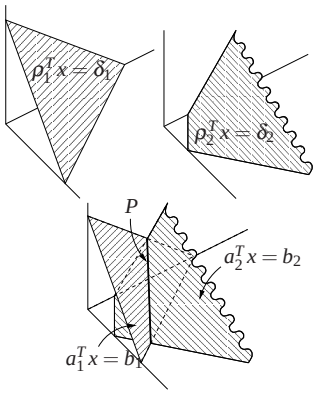
$$\begin{bmatrix} \rho_1^T \\ \rho_2^T \\ \rho_3^T \end{bmatrix} x \leq \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} \quad \text{e} \quad x \geq 0.$$

Um vértice no formato canônico é caracterizado de forma análoga ao formato dual. Isto é, um ponto $\bar{x}$ é vértice do poliedro $P = \{x \in \mathbb{R}^n \mid Ax \leq b,\ x \geq 0\}$ se, e somente se, $\bar{x}$ é um ponto viável, $|J(\bar{x})| = n$ e $\bar{x}$ é a única solução do sistema $A_{J(\bar{x})}x = b_{J(\bar{x})}$, com o acréscimo ao sistema das restrições $x_i = 0$, caso sejam necessárias.

Deste formato passa-se facilmente ao próximo formato, que será utilizado em algoritmos de otimização.

3.4.3 Formato Primal – Formato Padrão

O formato primal é muito útil na implementação de algoritmos em otimização, porém, só permite a visualização geométrica em casos muito simples. A figura, ao lado, define um poliedro P através dos hiperplanos $\rho_1^T x = \delta_1$ e $\rho_2^T x = \delta_2$, em que se define o sistema de equações pela matriz $A = \{a_i^T\} = \{\rho_i^T\}$ e o vetor independente $b = \{b_i\} = \{\delta_i\}$ para $i = 1, 2$.



De forma geral o poliedro P no formato padrão é representado pela matriz $A_{m\times n}$ com $m < n$, pelo vetor independente $b_{m\times 1}$ e pelas restrições de não negatividade.

$$\boxed{\begin{aligned} Ax &= b \\ x &\geq 0. \end{aligned}}$$

Para passar do formato canônico ao primal basta adicionar variáveis de folga x_{n+1}, x_{n+2}, $\ldots$, x_{n+m}. A restrição $a_i x \leq b_i$ torna-se equivalente a $a_i x + x_{n+i} = b_i$ para todo $i = 1, \ldots, m$. Assim, pode-se representar o sistema de desigualdades pelas equações abaixo.

$$\begin{aligned} Ax + z &= b \\ x, z &\geq 0 \\ x \in \mathbb{R}^n,\ z &\in \mathbb{R}^m. \end{aligned} \qquad \text{ou} \qquad \begin{aligned} \begin{bmatrix} A & I \end{bmatrix} x &= b \\ x &\geq 0 \\ x &\in \mathbb{R}^{n+m}. \end{aligned}$$

Exemplo 1: Considere um poliedro no $\mathbb{R}^4$, caracterizado pelas restrições

$$\rho_1^T x \geq 1 \tag{3.4}$$
$$\rho_2^T x \leq 2 \tag{3.5}$$
$$\rho_3^T x = 1 \tag{3.6}$$
$$x_1, x_3 \geq 0 \tag{3.7}$$
$$x_2 \leq 0 \tag{3.8}$$
$$x_4, \text{ irrestrita} \tag{3.9}$$

Note que este poliedro apresenta restrições de desigualdade, igualdades e uma variável irrestrita. Para representar o poliedro no formato primal basta acrescentar variáveis de folga, aumentando o número de variáveis. Adicionando as variáveis de folga nas restriçoes (3.4) e (3.5) podemos expressá-las por

$$-\rho_1^T x + z_6 = -1 \qquad \text{e} \qquad \rho_2^T x + z_7 = 2. \tag{3.10}$$

No caso da existência de variável irrestrita x_i, devemos substituí-la por $x_i = x_i^+ - x_i^-$, em que x_i^+ e x_i^- são variáveis não negativas. Considerando a restrição (3.9) do poliedro, temos que

$$x_4 = x_4^+ - x_4^-, \tag{3.11}$$

em que x_4^+, $x_4^- \geq 0$.

Já para manter a não negatividade das variáveis z_i, as restrições (3.7), (3.8) e (3.11), são descritas por

$$z_1 = x_1, \quad z_2 = -x_2, \quad z_3 = x_3, \quad z_4 = x_4^+, \quad \text{e} \quad z_5 = x_4^-. \tag{3.12}$$

Considerando $\rho_i = \begin{bmatrix} \rho_{i1} \\ \rho_{i2} \\ \rho_{i3} \\ \rho_{i4} \end{bmatrix}$, para $i = 1, 2, 3$. Podemos escrever o polinômio em sua forma matricial,

$$\begin{bmatrix} -\rho_{11} & \rho_{12} & -\rho_{13} & -\rho_{14} & \rho_{14} & 1 & 0 \\ \rho_{21} & -\rho_{22} & \rho_{23} & \rho_{24} & -\rho_{24} & 0 & 1 \\ \rho_{31} & -\rho_{32} & \rho_{33} & \rho_{34} & -\rho_{34} & 0 & 0 \end{bmatrix} z = \begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix}$$
$$z \geq 0,$$

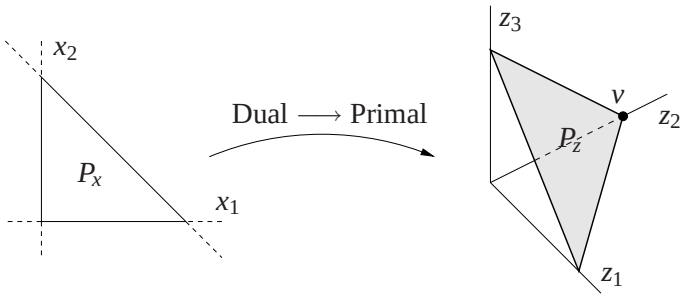
em que, $z \in \mathbb{R}^7$.

Após acrescentar as variáveis de folga, o poliedro é representado pelo sistema de equações, cuja dimensão aumenta do $\mathbb{R}^4$ para o $\mathbb{R}^7$, dificultando assim a visualização geométrica. Por esta razão, o formato canônico é mais prático geometricamente. Já o primal possui suas vantagens na elaboração de algoritmos de otimização, os quais serão abordados nos próximos capítulos.

Face no Formato Primal

No formato dual vimos que uma face é caracterizada pela intersecção do poliedro com um conjunto de hiperplanos ativos. Na transformação do formato dual em primal, o acréscimo das variáveis de folga faz com que a fronteira relativa ao poliedro esteja contida nos planos coordenados. A ilustração deste fato é vista no seguinte exemplo. Considere o poliedro

$$\begin{array}{r} x_1 + x_2 \leq 1 \\ x \geq 0 \end{array} \quad \overset{\text{variável de folga}}{\longrightarrow} \quad \begin{array}{r} z_1 + z_2 + z_3 = 1 \\ z \geq 0 \end{array} \tag{3.13}$$



O poliedro P_x, no formato dual, está contido no $\mathbb{R}^2$, enquanto as faces próprias do poliedro $P_z \subset \mathbb{R}^3$ estão contidas nos planos coordenados. Repassada a idéia do poliedro em formato primal, podemos caracterizar uma face por um *conjunto de variáveis nulas*, as quais correspondem ao conjunto ativo.

Caracterização de um vértice

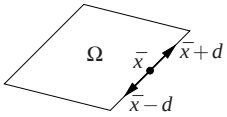
Nosso objetivo é caracterizar um vértice de um poliedro em termos de seus índices ativos. No formato primal, visto anteriormente, podemos relacionar os índices ativos com o conjunto de variáveis nulas que será de fundamental importância nas demonstrações dos teoremas que se seguem, além disso, relacionamos vértices com pontos extremos, que definiremos a seguir. Mas inicialmente, relembramos que um vértice do poliedro é definido por uma face de dimensão zero.

Definição 43 *O ponto $\bar{x}$ é um ponto extremo de um conjunto $\Omega \subset \mathbb{R}^n$ se, e somente se, $\bar{x} \in \Omega$ e*

é impossível encontrar $d \in \mathbb{R}^n, d \neq 0$ *tal que*

$$\bar{x} + d \in \Omega \qquad e \qquad \bar{x} - d \in \Omega.$$

Na figura ao lado, $\bar{x}$ não é ponto extremo do poliedro Ω. Vamos mostrar que vértices e pontos extremos são conceitos equivalentes em poliedros. O lema a seguir mostra um pouco mais do que isso.



Considere $x \in \mathbb{R}^n$. Definimos o conjunto de índices das componentes nulas de x por I_0, isto é,

$$I_0(x) = \{i = 1, \ldots, n \mid x_i = 0\}.$$

Por outro lado, o conjunto de índices das componentes positivas de x é definida por

$$I_+(x) = \{i = 1, \ldots, n \mid x_i > 0\}. \tag{3.14}$$

Considerando o poliedro P_z, definido em (3.13), o vértice

$$v = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

possui componentes nulas em z_1 e z_3. Assim, podemos representá-los por $I_0(v) = \{1, 3\}$, que define:

$$v_{I_0} = \begin{bmatrix} z_1 \\ z_3 \end{bmatrix} \quad \text{e} \quad A_{I_0} = \begin{bmatrix} A_{(:,1)} & A_{(:,3)} \end{bmatrix} = \begin{bmatrix} 1 & 1 \end{bmatrix}.$$

Já a componente não nula de v, define o conjunto $I_+(v) = \{2\}$ e analogamente,

$$v_{I_+} = [z_2] \quad \text{e} \quad A_{I_+} = [A_{(:,2)}] = [1].$$

Lema 15 *Seja* $\Omega = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$ *um poliedro em* $\mathbb{R}^n$ *e* $\bar{x} \in \Omega$ *um ponto dado. As seguintes asserções são equivalentes:*

(i) $\bar{x}$ *é ponto extremo de* Ω*;*

(ii) $\forall x \in \Omega,\ x \neq \bar{x},\ I_0(x) \neq I_0(\bar{x})$*;*

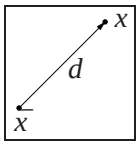
(iii) $\{\bar{x}\}$ *é uma face de* Ω*.*

Dem. Considere $\bar{x} \in \Omega$. Sabemos que $I_0(\bar{x}) = I(\bar{x})$, e considere $I_+(\bar{x})$ o conjunto de índices definido por (3.14). Sem perda de generalidade, suponha que

$$\bar{x} = \begin{bmatrix} \bar{x}_{I_0} \\ \bar{x}_{I_+} \end{bmatrix},$$

em que $\bar{x}_{I_0} = 0$ e $\bar{x}_{I_+} > 0$.

(i) $\Rightarrow$ **(ii)** Suponha $\bar{x}$ ponto extremo de Ω e considere $x \in \Omega$, tal que $x \neq \bar{x}$. Por absurdo, suponha $I_0(x) = I_0(\bar{x})$. Definindo $d = x - \bar{x}$, temos que $d_{I_0} = 0$, pois $x_{I_0} = 0$ e $\bar{x}_{I_0} = 0$.



Da convexidade de Ω, sabemos que para qualquer $\lambda \in (0,1)$, $\bar{x} + \lambda d \in \Omega$. Como $\bar{x}_{I_+} > 0$, para λ suficientemente pequeno, segue que $\bar{x}_{I_+} - \lambda d_{I_+} > 0$ e portanto $\bar{x} - \lambda d \in \Omega$. Assim, $\bar{x} - \lambda d \in \Omega$ e $\bar{x} + \lambda d \in \Omega$, o que contradiz o fato de $\bar{x}$ ser ponto extremo.

(ii) $\Rightarrow$ **(iii)** Dado $\bar{x}$, determinamos $I_0(\bar{x})$, $I_+(\bar{x})$ e definimos o vetor ρ, em que $\rho_i = 1$, se $i \in I_0(\bar{x})$ e $\rho_i = 0$, se $i \in I_+(\bar{x})$. Isto é,

$$\rho = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} I_0(\bar{x}) \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} I_+(\bar{x}) \end{matrix}$$

Sabemos que $\rho^T x \geq 0$, $\forall x \in \Omega$ e que $\rho^T \bar{x} = 0$. Por tais razões, $\bar{x}$ é minimizador de $\rho^T x$ em Ω.

Seja $x \in \Omega$, tal que $x \neq \bar{x}$. Por hipótese, temos que $I_0(x) \neq I_0(\bar{x})$ e assim, para algum $j \in I_0(\bar{x})$, $x_j > 0$. Daí, segue que $\rho^T x > 0$, isto é, $\bar{x}$ é o único minimizador. Portanto $\{\bar{x}\}$ é uma face de Ω definida por ρ.

(iii) $\Rightarrow$ **(i)** Suponhamos que $\bar{x}$ é solução única de

$$\underset{x \in \Omega}{\text{minimizar}} \quad \rho^T x$$

Seja $d \in \mathbb{R}^n$ uma direção arbitrária. Se $\bar{x} + d \in \Omega$, temos que $\rho^T d > 0$, pois $\bar{x}$ é solução única. Logo, $\rho^T(-d) < 0$.

Supondo por absurdo que $\bar{x} - d \in \Omega$. Teremos que

$$\rho^T(\bar{x} - d) = \rho^T \bar{x} - \rho^T d < \rho^T \bar{x},$$

contrariando a otimalidade de $\bar{x}$. Deste modo, concluímos que é impossível encontrar d tal que $\bar{x}+d \in \Omega$ e $\bar{x}-d \in \Omega$. ■

Teorema 21 (Vértice no formato primal) *Seja $\Omega \subset \mathbb{R}^n$ um poliedro representado por $\Omega = \{x \in \mathbb{R}^n \mid Ax = b \text{ e } x \geq 0\}$. O ponto v é um vértice de Ω se, e somente se, as colunas de A com índices em $I_+(v)$ são linearmente independentes.*

Dem. Dado um ponto $v \in \Omega$, defina:

$$A_{I_0} = \text{submatriz com colunas } i \in I_0(v)$$

$$A_{I_+} = \text{submatriz com colunas } i \in I_+(v)$$

Sem perda de generalidade, suponha que

$$A = \begin{bmatrix} A_{I_0} & A_{I_+} \end{bmatrix} \quad \text{e} \quad v = \begin{bmatrix} v_{I_0} \\ v_{I_+} \end{bmatrix} = \begin{bmatrix} 0 \\ v_{I_+} \end{bmatrix}.$$

As colunas de A_{I_+} são linearmente *dependentes* se e somente se existe

$$d = \begin{bmatrix} 0 \\ d_{I_+} \end{bmatrix},$$

em que $d \neq 0$ e, tal que $Ad = 0$, ou equivalentemente, $A(\lambda d) = 0, \ \forall \lambda \in (0,1)$.

Para qualquer escolha de d com essas propriedades, tomando λ suficientemente pequeno, podemos garantir que $v \pm \lambda d \geq 0$, pois $v_{I_+} > 0$ e $d_{I_0} = 0$. Por tais razões, as colunas de A_{I_+} são linearmente dependentes se, e somente se, existe uma direção $\tilde{d}$ tal que

$$\bar{x}+\tilde{d} \in \Omega \quad \text{e} \quad \bar{x}-\tilde{d} \in \Omega,$$

isto é, se e somente se, $\bar{x}$ não é um vértice. Isto completa a demonstração. ■

Teorema 22 *Considere um poliedro em $\mathbb{R}^n$definido por*

$$Ax = b$$
$$x \geq 0.$$

Se existir um ponto viável, então existe um vértice.

Dem. Se existir ponto viável, então podemos escolher um ponto viável $\bar{x}$ com o menor número possível de componentes não nulos. Vamos mostrar que $\bar{x}$ é um vértice.

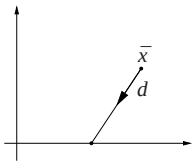
Por absurdo, suponha que $\bar{x}$ não é vértice. Do Teorema 21, as colunas de índices

$$I_+(\bar{x}) = \{i \mid i \in \{1,\ldots,n\},\ \bar{x_i} > 0,\ \bar{x} \in \mathbb{R}^n\}$$

são linearmente dependentes. Além disso, da demonstração do teorema (21), encontramos uma direção $d \neq 0$, tal que $d_{I_0} = 0$ e $Ad = 0$.

Se $d_{I_+(\bar{x})} \geq 0$, tome $d = -d$. Assim d tem alguma componente negativa.

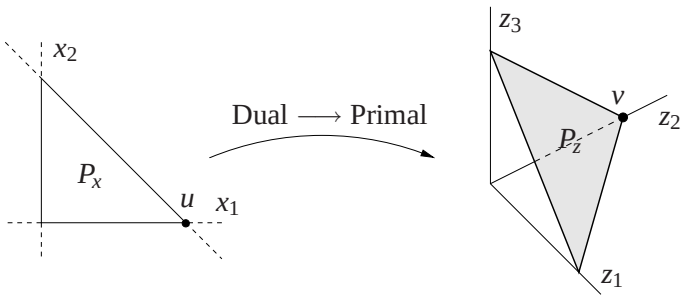
Se caminharmos ao longo de $\bar{x} + \lambda d$, encontraremos um ponto $\hat{x}$ tal que $\hat{x}$ é viável e $|I_+(\hat{x})| < |I_+(\bar{x})|$, contradizendo o fato de $\bar{x}$ ter o menor número de variáveis nulas.



■

Considere o poliedro P, representado nos formatos canônico P_x e primal P_z, da seguinte maneira:

$$\begin{array}{rcl} x_1 + x_2 \leq 1 & \xrightarrow{\text{variável de folga}} & z_1 + z_2 + z_3 = 1 \\ x \geq 0 & & z \geq 0 \end{array}$$



A representação matricial de P_x no formato dual, é dado por $Rx \leq d$, isto é,

$$\begin{bmatrix} 1 & 1 \\ 0 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \tag{3.15}$$

Podemos caracterizar o vetor $u = [1 \quad 0]^T$, no formato dual, pelo conjunto de restrições ativas,

$$J(u) = \{j = 1,2,3 \mid r_j^T u = d_j\} = \{1,2\},$$

em que r_j^T é a j-ésima linha da matriz R. Daí definimos

$$R^{J(u)}x = d_{J(u)} \qquad \Leftrightarrow \qquad \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix} x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{3.16}$$

Note que u é a única solução do sistema (3.16), além disso, $Ru \leq d$ e $|J(u)| = 2$. Portanto, do Teorema 20, u é um vértice de P_x.

No formato primal, representamos o poliedro P_z por $Az \leq b$ e $z \geq 0$. Caracterizamos o vetor $v = [0 \ 1 \ 0]^T$ pelas variáveis nulas, em que definimos $I_0(v) = \{1,3\}$ e $I_+(v) = \{2\}$. Daí,

$$A_{I_0} = [1 \ 1] \quad \text{e} \quad A_{I_+} = [1].$$

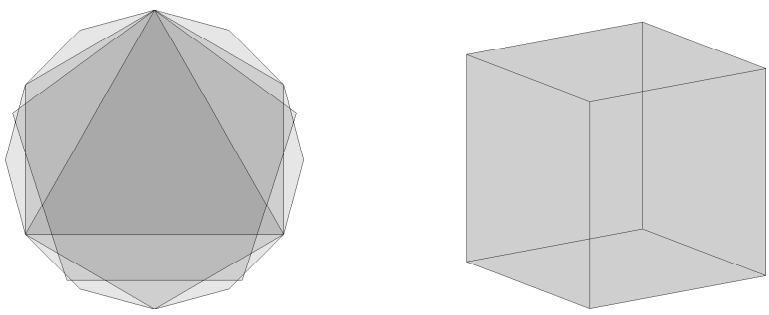
Note que as colunas de A com índices em I_+ são linearmente independentes. Portanto, do teorema (21), v é um vértice de P_z.

É válido ressaltar que, no formato dual, a matriz R^J é uma submatriz composta pela linhas da R. Por outro lado, no formato primal, as matrizes A_{I_0} e A_{I_+} são submatrizes compostas por colunas de A.

Os vértices de um poliedro, caracterizados no formato primal, têm importância fundamental na resolução de problemas de programação linear. De nossa análise anterior, vimos que uma face pode ser decomposta em faces de dimensões inferiores, até chegar-se aos vértices. No estudo do algoritmo Simplex, concluiremos que basta examinar os vértices do poliedro para obter um ponto da face ótima de um problema. Para apresentarmos o desenvolvimento destes algoritmos, a próxima seção retoma o formato canônico e dual na elaboração de um programa que constrói poliedros no $\mathbb{R}^2$ e $\mathbb{R}^3$.

3.5 Visualização espacial de Politopos

Nesta seção, vamos nos deter na construção de politopos em $\mathbb{R}^3$, pois os politopos em $\mathbb{R}^2$ são comparados a facetas dos poliedros em três dimensões. Apresentaremos um programa denominado `POLIEDRO_V1`, que tem o objetivo de construir os politopos que servirão de auxílio na compreensão geométrica dos algoritmos que utilizam como conjunto viável poliedros. O programa constrói somente poliedros limitados, ou seja, politopos.



Vimos anteriormente que os poliedros no formato dual são representados por restrições lineares $Ax \leq b$, em que $A \in \mathbb{R}^{m\times 3}$, $x \in \mathbb{R}^3$ e $b \in \mathbb{R}^m$. A intersecção dos semi-espaços definidos por estas restrições, definem um poliedro convexo. A construção deste poliedro consiste em 4 passos fundamentais:

1. Primeiramente, são determinados todos os vértices do poliedro. Isto é possível, pois estamos trabalhando em dimensão menor ou igual a 3 e o número de restrições é finito. Os vértices são organizados através de uma matriz V, em que cada linha de V representa um vértice. Este processo é baseado no Teorema 20 e se divide em três etapas:

   - Na primeira etapa fazemos a intersecção de três planos (restrições), que define um ponto $p \in \mathbb{R}^3$, caso essas restrições sejam linearmente independentes.
   - Na segunda etapa verificamos se o ponto p é viável. Para isso, basta satisfazer as restrições
   $$Ap \leq b.$$
   Caso contrário o ponto é descartado.
   - A terceira etapa, consiste em adicionar o vértice como uma linha da matriz V, caso esse vértice não tenha sido selecionado anteriormente.

   A partir deste primeiro passo, podemos restringir o Teorema 20 para caracterizar um vértice em $\mathbb{R}^3$, apresentado no seguinte resultado.

   Lema 16 *Dado um poliedro P definido por $Ax \leq b$, em que A é $m \times 3$. Um ponto $\bar{x}$ é um vértice de P se as seguintes asserções são verdadeiras:*

   (a) $A\bar{x} \leq b$;

   (b) $\bar{x} \geq 0$;

   (c) Existe pelo menos um conjunto de índices J, tal que $\bar{x}$ é a única solução do sistema
   $$A^J x = b_J,$$
   em que $|J| = 3$. Isto é, A^J é não singular.

   Os itens (a) e (b) caracterizam um ponto viável.

   Um dado interessante, é com relação ao número de combinações que devemos fazer para encontrar todos os vértices. Seja $\bar{x}$ um ponto viável. O Lema 16 determinar que $\bar{x}$ é um vértice do poliedro se $\bar{x}$ pertencer a intersecção de três planos Linearmente Independentes definidos pelas retrições. Como o número de restrições é m, devemos repetir este processo

$\binom{m}{3}$ vezes, para garantir que todas as restrições sejam analisadas. Este procedimento é dispendioso, mas como estamos trabalhando com dimensão menor ou igual a 3, torna-se desprezível seu custo computacional.

Outra questão para determinarmos os vértices do poliedro P, é quanto ao seu formato:

(a) dual: Os dados a serem considerados são apenas a matriz A e o vetor b;

(b) canônico: Mudamos o formato do poliedro P, adicionando a restrição de positividade $x \geq 0$, e definindo um novo sistema $Dx \leq d$, isto é,

$$\begin{bmatrix} A \\ -I \end{bmatrix} x \leq \begin{bmatrix} b \\ 0 \end{bmatrix}.$$

O algoritmo a seguir determina os vértices de um poliedro $P = \{x \in \mathbb{R}^3 \mid Ax \leq b\}$, em que $A \in \mathbb{R}^{m\times 3}$ e $b \in \mathbb{R}^m$.

Algoritmo 3.1: Vértices

Dados: $A \in \mathbb{R}^{m\times 3}$ e $b \in \mathbb{R}^m$;

Faça $V = [\,]$; Matriz dos vértices.

Para *cada conjunto de índices* $J \subset \{1,\ldots,m\}$ *com elementos distintos, tal que* $|J| = 3$.
Defina o sistema

$$A^J x = b_J.$$

Se A^J *é não singular* **então**
$x = (A^J)^{-1} b_J$.
Se $Ax \leq b$ *e* $x^T \notin V$ **então**
adicione x^T como uma linha de V
Senão
x é descartado.
Fim-se
Fim-se

Fim

Utilizando a teoria de grafos, podemos determinar todos os vértices do poliedro $P \subset \mathbb{R}^n$, por um processo que reduz significativamente o número de combinações a serem consideradas. Este estudo é apresentado em Chvátal (1983).

2. Após obter todos os q vértices do poliedro, representados pelas linhas da matriz $V_{q\times 3}$, o segundo passo é fixar um plano e verificar se o mesmo contém uma faceta do poliedro. Este passo é obtido através da multiplicação de matrizes

$$VA(i,:) = b(i)$$

em que $A(i,:)$ é a i-ésima linha da matriz das restrições (o vetor normal ao plano) e $b(i)$ o termo independente relacionado ao plano i fixo.

Se existir três ou mais vértices então o plano contém uma faceta do poliedro, caso contrário dizemos que a restrição é redundante (Em $\mathbb{R}^2$, se não existirem dois vértices, a restrição é considerada redundante).

É importante ressaltar que os passos 2, 3 e 4 são efetuados em uma única face, sendo assim necessário, após o quarto passo, retornar a este passo para a verificação dos planos restantes.

3. O terceiro passo é ordenar os vértices da face selecionados no sentido horário ou anti-horário. Este passo é muito importante, pois o programa é fundamentado no comando "*patch*" do Matlab, no qual consiste em construir a envoltória convexa dos pontos no plano. A ordenação dos vértices é essencial e será estudada mais adiante.

4. O quarto e último passo é utilizar o comando "*patch*" do Matlab. Esse comando tem como entrada de dados a matriz dos vértices V e um vetor s que indica a ordem em que esses vértices devem ser apresentados, para a construção da envoltória convexa (faceta).

$$V = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} \qquad s = \begin{bmatrix} 1 \\ 5 \\ 3 \\ 2 \\ 6 \\ 4 \end{bmatrix}$$

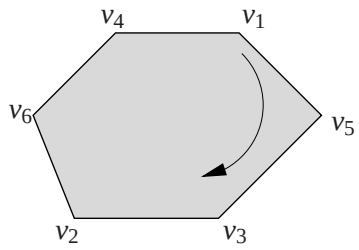


Figura 3.5: O comando "*patch*" constrói a combinação convexa dos vértices em V, na sequência do vetor s.

A linha de comando do Matlab para a construção da envoltória convexa é definida por:

```
patch('vertices',V,'faces',s,'FaceColor',cor);
```

Após construir a face, retornamos ao passo 2 e fixamos outro plano seguindo o mesmo processo, construindo face por face, até todos serem verificados. Ao analisar todas as restrições o poliedro está construído. É importante destacar que no $\mathbb{R}^2$ consideramos apenas um plano e deste modo não é necessário retornar ao segundo passo.

Além do comando *patch*, outras ferramentas do Matlab auxiliam na visualização de poliedros, como por exemplo a transparência e a rotação.

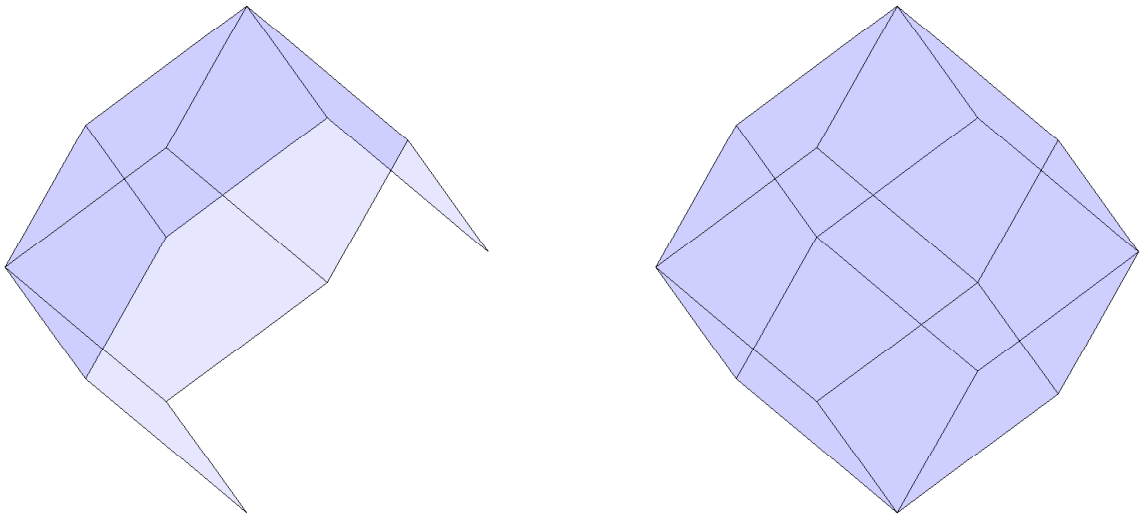


Figura 3.6: Poliedro sendo formado face a face.

3.5.1 Ordenando os vértices

Dada a matriz $V_{q\times 3}$ dos vértices de um polígono (faceta). A idéia é encontrar uma sequência finita de índices, representada pelo vetor s, que determine a ordem dos vértices do polígono. Primeiramente fixamos o vértice v_1, isto é, o coeficiente do primeiro vértice é a origem da sequência que estamos procurando

$$s = [1],$$

além disso, construímos todos os vetores *auxiliares* com origem nesse vértice e extremidade nos demais vértices, veja Figura (3.7).

Para ordenarmos os vértices do polígono, tomamos como exemplo um polígono com $q = 6$ vértices, ou seja, V e o poliedro são dados da seguinte forma:

$$V = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix}$$

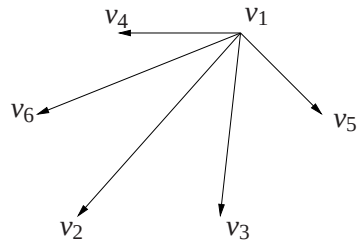


Figura 3.7: Construção dos vetores auxiliares

Depois de construídos os vetores auxiliares, fixamos o vetor com extremidade no segundo vértice v_2, denominado *vetor base*. Em seguida, calculamos os ângulos do vetor base com relação aos demais. Após, tomamos o máximo entre os ângulos e fixamos o vetor adjacente

a este ângulo como uma aresta do polígono. Assim, definimos o segundo elemento da nossa sequência, determinado por desta aresta. Retornando ao exemplo, na Figura (3.8), θ representa o máximo entre os ângulos, assim fixemos o índice 5 como o próximo ponto da sequência. Note que o segmento de v_1 a v_5 é uma aresta do polígono.

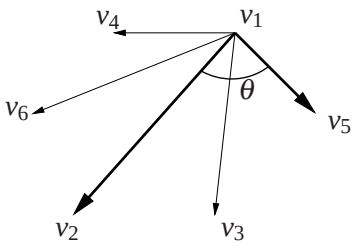
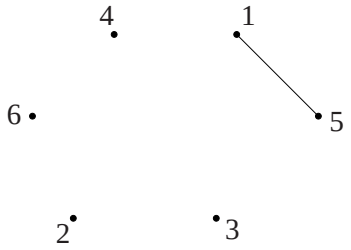


Figura 3.8: Máximo ângulo



$$s = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$

Figura 3.9: Construção da aresta

Observe que estamos construindo o poliedro no $\mathbb{R}^3$ e os ângulos calculados entre os vetores auxiliares são positivos, por tais razões, devemos definir primeiramente uma aresta da faceta.

Agora, definimos um novo vetor base com extremidades v_1 e v_5, e calculamos os ângulos com os demais vetores. Reordenando esses ângulos em ordem crescente, encontramos os demais índices que definem a nossa sequência s. Assim, obtemos o polígono com vértices orientados por s. No exemplo, s e o polígono são dados por:

$$s = \begin{bmatrix} 1 \\ 5 \\ 3 \\ 2 \\ 6 \\ 4 \end{bmatrix}$$

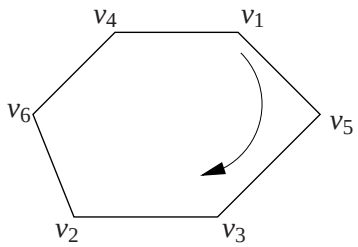


Figura 3.10: polígono definido pelos vértices em V, na orientação s.

Algoritmo 3.2: Ordem dos vértices

Dados: $V_{n\times 3}$, Matriz dos vértices do polígono.

Seja $s = [1]$;

– Construa os vetores auxiliares, determinando o *vetor base*;

– Calcule o *ângulo máximo* entre o *vetor base* e os demais vetores auxiliares. Assim, determinando uma aresta do polígono e a segunda componente do vetor s;

– Fixe essa aresta como sendo o novo *vetor base* e determine todos os ângulos com os vetores restantes;

– Ordene os ângulos de forma crescente, determinando o vetor s pelos índices das extremidades de cada vetor.

3.5.2 O programa `POLIEDRO_V1`

O programa `POLIEDRO_V1` foi implementado em Matlab, e tem o objetivo de auxiliar na compreensão geométrica de algoritmos, como o simplex e o branch-and-bound, construindo os conjuntos viáveis (poliedros) para que possam ser visualizados em duas ou três dimensões. O formato e a dimensão do poliedro são definidos pelas restrições. O `POLIEDRO_V1` constrói os poliedros utilizando apenas o formato canônico ou dual através do programa `poliedro.m`.

Os dados de entrada do programa `poliedro.m` são:

a) A matriz das restrições A e o vetor dos termos independentes b. Independentemente do formato do poliedro, canônico ou dual, A e b estão bem definidos.

b) O formato do poliedro pode ser canônico ou dual, e é representado pelos números 1 e 2, respectivamente.

Sabemos que o formato canônico considera a restrição de positividade $x \geq 0$, já o formato dual não necessariamente possui essa restrição. Veremos na próxima seção um exemplo, onde isso ocorre.

c) A cor do poliedro[4].

As principais cores são: amarelo, representado pela letra y, magenta por m, ciano c, vermelho r, verde g, azul b, branco w e preto k.

Considere o poliedro $P = \{x \in \mathbb{R}^3 \mid Ax \leq b, x \geq 0\}$. A linha de comando do Matlab, para construir o poliedro P no formato canônico, é definida por:

```
poliedro(A,b,1,'b');
```

esse comando reproduz o poliedro P, azul no $\mathbb{R}^3$. O programa trabalha com o formato canônico (1), e a cor azul (b) como *Default*. Então para construirmos o mesmo poliedro P, basta o comando `poliedro(A,b)`.

Por outro lado, se quisermos construir um poliedro $Q = \{x \in \mathbb{R}^2 \mid Dx \leq d\}$ na cor vermelha, o comando é:

```
poliedro(D,d,2,'r').
```

[4]Existem diversas formas de definir cores no Matlab, encontradas em qualquer manual do Matlab.

Algoritmo 3.3: Poliedro

Dados: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, formato, cor.

Se $n = 3$ *ou* $n = 2$, **então**

Se *formato é canônico (1),* **então**

– Definimos um novo sistema $Dx \leq d$,isto é,

$$\begin{bmatrix} A \\ -I \end{bmatrix} x \leq \begin{bmatrix} b \\ 0 \end{bmatrix},$$

em que I é a matriz identidade de ordem n.

$m = p + n$;

Senão

$D = A$, $d = b$ e $m = p$;

Fim-se

Note que $D \in \mathbb{R}^{m\times n}$ e $d \in \mathbb{R}^m$.

– Determine todos os vértices do poliedro, através do algoritmo (3.1);

Seja V a matriz desses vértices.

Para $i = 1, \ldots, m$

– Fixe o hiperplano, $D^i x = d_i$, em que D^i é a i-ésima linha da matriz D.

– Determine quais os vértices que pertencem a este hiperplano. **Se** *os vértices selecionados formam uma faceta (são em número maior ou igual a n),* **então**

– Determine o vetor s, de ordenação dos vértices pelo algoritmo (3.2).

– Através do comando *patch* construímos a faceta do poliedro.

```
patch('vertices',V,'faces',s,'FaceColor',cor);
```

Senão

A restrição é desconsiderada (*redundante*).

Fim-se

Fim

Senão

Não é possível construir o poliedro.

Fim-se

3.5.3 Detalhes do programa `Poliedro`

Formato canônico ou dual

Considere os poliedros $P = \{x \in \mathbb{R}^2 \mid Ax \leq b\}$ e $Q = \{x \in \mathbb{R}^2 \mid Ax \leq b,\ x \geq 0\}$, em que A e b são definidos por:

$$A = \begin{bmatrix} 1 & 1 \\ 3 & -5 \\ -2 & 1 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}.$$

Note que P esta no formato dual, sem a restrição de positividade. Então, aplicando o programa `poliedro`, obtemos P na Figura (3.11). Por outro lado, Q esta no formato canônico e é definido pelo programa `poliedro` na Figura (3.12)[5].

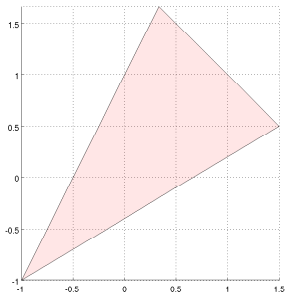


Figura 3.11: `poliedro(A,b,2,'r')`

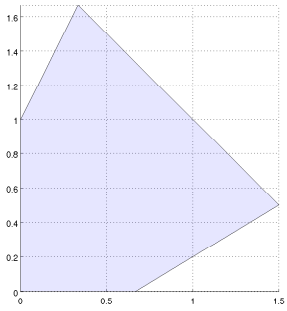


Figura 3.12: `poliedro(A,b)`

Observe que podemos transformar o poliedro Q no formato dual através do sistema $Dx \leq d$, em que

$$D = \begin{bmatrix} 1 & 1 \\ 3 & -5 \\ -2 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \quad \text{e} \quad d = \begin{bmatrix} 2 \\ 2 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

isto é, $Q = \{x \in \mathbb{R}^2 \mid Dx \leq d\}$.

Neste exemplo, podemos dizer que P é um "subsistema" de Q, isto é, $Q \subset P$. Além disso, a restrição de positividade $x \geq 0$ não é considerada em P, e não é redundante em Q. Daí, temos que $P \not\subset Q$. Portanto $Q \neq P$, veja Figura (3.13).

[5]Lembramos que o programa `poliedro` tem como padrão o formato canônico e a cor azul.

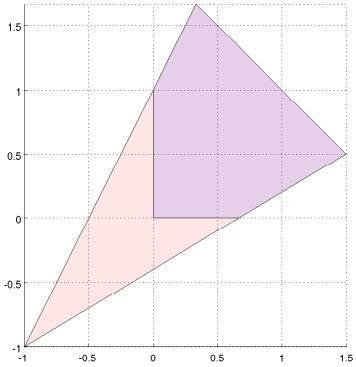


Figura 3.13: $Q \subset P$.

Precisão computacional

Ao computar os vértices e construir as facetas do poliedro exigimos uma precisão de 10^{-10}, isto é, se a distância entre dois vértices é menor que 10^{-10} conseideramos apenas um vértice. Por essa razão, o poliedro não é alterado em perturbações numa ordem menor que 10^{-10}. Veja no exemplo a seguir.

Considere o poliedro sem perturbação $P_1 = \{x \in \mathbb{R}^2 \mid Ax \leq b,\ x \geq 0\}$, em que

$$A = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 4 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 4 \\ 4 \end{bmatrix},$$

e o poliedro $P_2 = \{x \in \mathbb{R}^2 \mid Ax \leq \bar{b},\ x \geq 0\}$, com perturbação ε no termo independente $\bar{b}$ dada por:

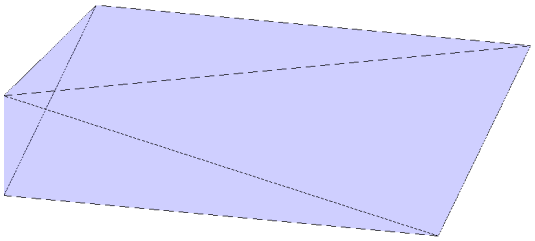


Figura 3.14: Poliedro sem perturbação.

$$\bar{b} = \begin{bmatrix} 4+\varepsilon \\ 4+\varepsilon^2 \end{bmatrix}.$$

Nas Figuras (3.14) e (3.15), apresentamos os poliedros P_1 e P_2, respectivamente, em que os valores de ε computados são 10^{-5} e 10^{-6}.

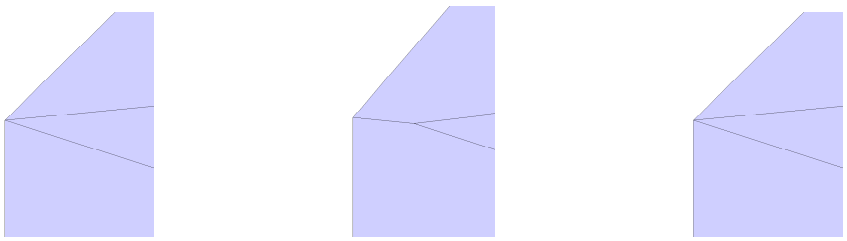


Figura 3.15: Zoom do poliedro sem perturbação e com perturbações de ordem 10^{-10} e 10^{-12}, respectivamente.

Elaboração de poliedros

Dado um politopo na forma algébrica, através do programa `POLIEDRO_V1` é fácil visualizá-lo na forma geométrica. Por outro lado, se é dado um politopo na forma geométrica, o trabalho para representá-lo algebricamente é muito dispendioso. O grau de dificuldade depende do número de restrições ou facetas do politopo, e de sua simetria. Por exemplo, na construção do dodecaedro pelo programa `poliedro` são necessárias 12 restrições, o que torna difícil sua construção. Por outro lado, o dodecaedro possui uma simetria bem definida, facilitando o cálculo dos vértices.

O dodecaedro é definido por $P = \{x \in \mathbb{R}^2 \mid Ax \leq b\}$, em que

$$A = \begin{bmatrix} r & -2 \\ 2-r & r-2 \\ 2 & -r \\ 2 & r \\ 2-r & 2-r \\ r & 2 \\ -r & 2 \\ r-2 & 2-r \\ -2 & r \\ -2 & -r \\ r-2 & r-2 \\ -r & -2 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 4r \\ r^2-8r+12 \\ -4r+16 \\ 4r+16 \\ r^2-16r+28 \\ 4r+16 \\ -4r+16 \\ r^2-8r+12 \\ 4r \\ -4r \\ r^2-4 \\ -4r \end{bmatrix},$$

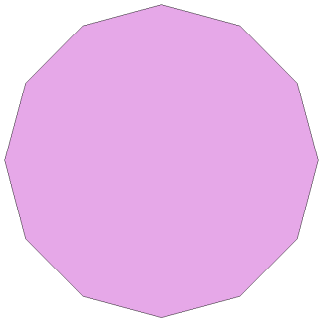
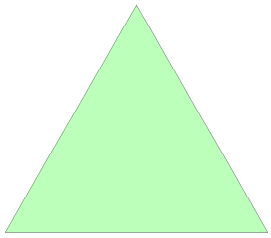
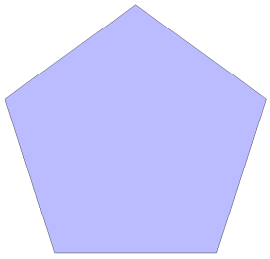


Figura 3.16: Dodecaedro

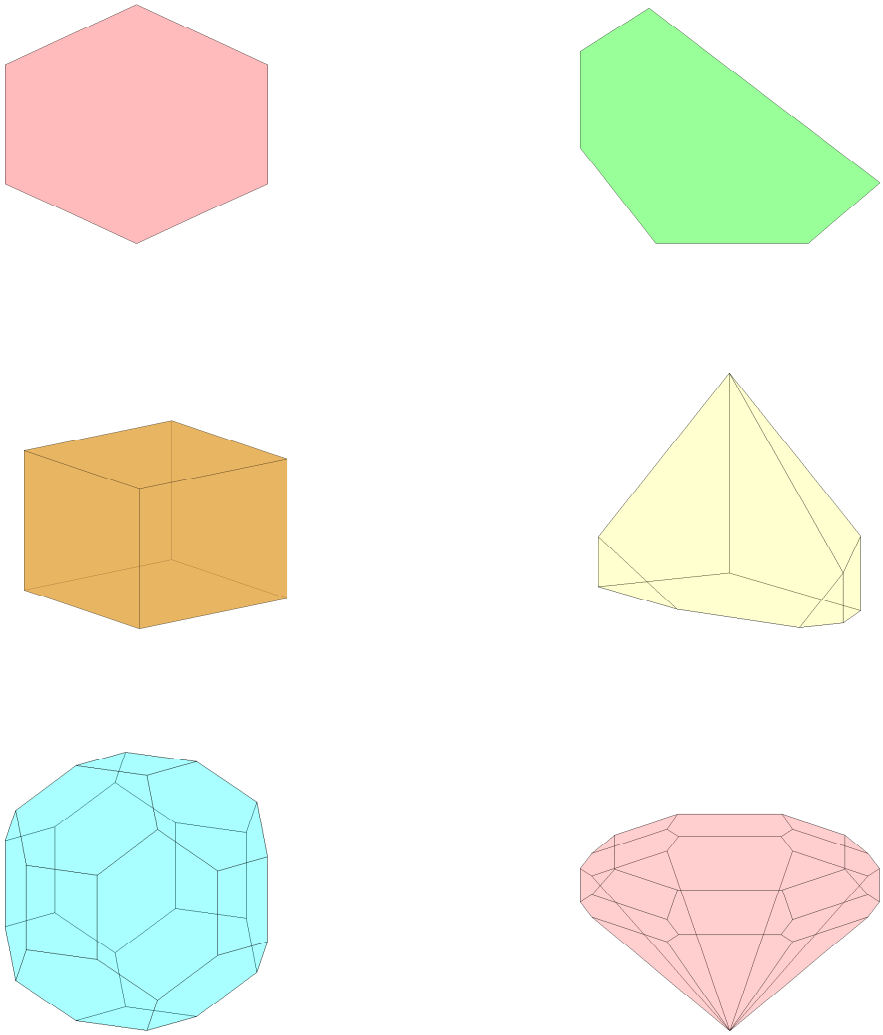
com $r = 8\left(\sin\left(\frac{\pi}{12}\right)\right)^2$.

Segue alguns exemplos de poliedros construídos pelo programa[6]:





[6] Anexo, no **Guia do Usuário**, estão os arquivos "`.m`" que definem os poliedros desta seção.



Poliedros ilimitados

O programa "`poliedro`" constrói poliedros limitados. Mas dado um poliedro na forma algébrica, como saber se ele é ou não limitado? Sabemos, por construção, que o programa analisa face por face, até que todas as facetas sejam construídas. Se o poliedro é um politopo $P \subset \mathbb{R}^3$, então o programa fornece duas regiões bem definidas, o interior do poliedro $\overset{\circ}{P}$ e seu complementar $\mathbb{R}^3 \setminus P$, em que $\overset{\circ}{P} \cap \{\mathbb{R}^3 \setminus P\} = \emptyset$.

Por outro lado, o programa não consegue definir para um poliedro Q ilimitado, o seu interior e nem o seu complementar $\mathbb{R}^3 \setminus Q$. Mas para abordar o estudo de poliedros ilimitados, devemos primeiramente definir direção de recessão.

Definição 44 (Direção de recessão) *Dizemos que $d \in \mathbb{R}^n$ é uma direção de recessão do con-*

junto convexo $\Omega \subset \mathbb{R}^n$ *quando*

$$x + td \in \Omega \quad \forall x \in \Omega, \forall t \in \mathbb{R}_+.$$

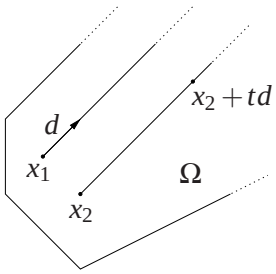


Figura 3.17: Direção de recessão

Denotamos por $\mathscr{R}_\Omega$ o conjunto de todas as direções de recessão do conjunto Ω. A proposição a seguir mostra que $\mathscr{R}_\Omega$ é um cone convexo (chamado de *cone de recessão*) e que ele contém direções não-nulas se, e somente se, o conjunto Ω é ilimitado.

Proposição 2 (Cone de recessão de conjunto convexo)[7] *Seja* $\Omega \subset \mathbb{R}^n$ *um conjunto convexo, fechado e não vazio. Então*

(a) $\mathscr{R}_\Omega$ *é um cone convexo, fechado e não vazio.*

(b) $d \in \mathscr{R}_\Omega$ *se, e somente se, existe* $x \in \Omega$ *tal que* $x + td \in \Omega$ *para todo* $t \in \mathbb{R}_+$.

(c) $\mathscr{R}_\Omega \neq \{0\}$ *se, e somente se,* Ω *é ilimitado.*

Proposição 3 *Sejam* $A \in \mathbb{R}^{m\times n}$ *e* $b \in \mathbb{R}^m$. *O cone de recessão do poliedro* $\Omega = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ *é dado por:*

$$\mathscr{R}_\Omega = \{d \in \mathbb{R}^n \mid Ad \leq 0\}.$$

Dem. $(\subseteq)$ Sejam $d \in \mathscr{R}_\Omega$ e $x \in \Omega$. Então pela definição (44), $x + td \in \Omega$ para todo $t \in \mathbb{R}_+$.

Assim, para todo $t \in \mathbb{R}_+$, temos que

$$\begin{aligned} A(x+td) &\leq b \\ Ax + tAd &\leq b \\ tAd &\leq b - Ax. \end{aligned}$$

Seja $M = b - Ax$. Como $Ax \leq b$, segue que $M \geq 0$. Então,

$$tAd \leq M, \quad \forall t \in \mathbb{R}_+.$$

[7] A demonstração da proposição (2) é encontrada em Izmailov e Solodov (2005)

Portanto, $Ad \leq 0$.

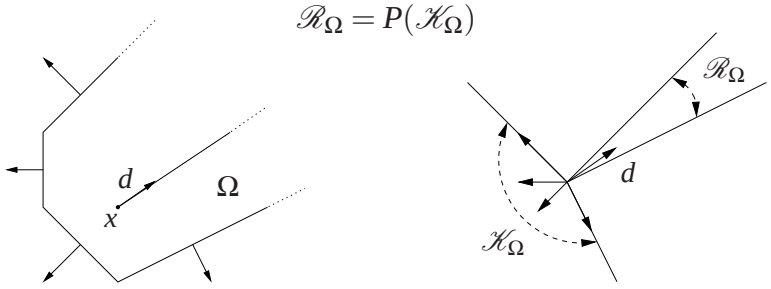
($\supseteq$) Sejam $x \in \Omega$, $t \in \mathbb{R}_+$ e $d \in \mathbb{R}^n$, tal que $Ad \leq 0$. Então

$$A(x+td) = Ax + tAd \leq Ax \leq b,$$

como $t \in \mathbb{R}_+$ é qualquer, temos que $x+td \in \Omega$, $\forall t \in \mathbb{R}_+$. ■

Considere o poliedro $\Omega = \{x \in \mathbb{R}^n \mid Ax \leq b\}$. Geometricamente, uma direção d será de recessão se esta não formar ângulo agudo com qualquer vetor normal dos semi-espaços que definem o poliedro Ω. Além disso, se denotarmos a envoltória cônica dos vetores normais aos semi-espaços por $\mathscr{K}_\Omega$, concluímos que o cone de recessão de Ω é igual ao polar de $\mathscr{K}_\Omega$.

$$\mathscr{R}_\Omega = P(\mathscr{K}_\Omega)$$



Exemplo 2: Considere o politopo $\Omega = \{x \in \mathbb{R}^3 \mid Ax \leq b\}$, em que

$$A = \begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & -1 \\ -1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & -1 & -1 \\ -8 & -8 & -11 \\ -8 & 8 & -11 \\ 8 & -8 & -11 \\ 8 & 8 & -11 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} -2 \\ 8 \\ -8 \\ 8 \\ -8 \\ -128 \\ 0 \\ 0 \\ 128 \\ 8 \end{bmatrix}.$$

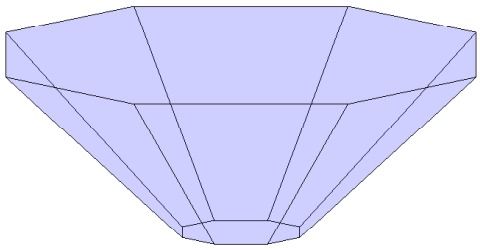


Figura 3.18: politopo Ω

Sabemos que o programa "`poliedro`" constrói facetas a partir dos vértices. Se não consideramos a última restrição do politopo Ω, que corresponde à faceta superior na Figura (3.18), definimos um poliedro $Q = \{x \in \mathbb{R}^3 \mid \bar{A}x \leq \bar{b}\}$ ilimitado, isto é, existe $d \in \mathbb{R}^3$ tal que $\bar{A}d \leq 0$.

Neste exemplo, as facetas ilimitadas do poliedro Q possuem no máximo dois vértices, então o programa ignora essas facetas e nos retorna um polígono, a base do politopo Ω. Veja Figura (3.19).

Figura 3.19: Base de Ω

Se acrescentarmos a condição de positividade no poliedro Q, definimos outro poliedro $R = \{x \in \mathbb{R}^3 \mid \bar{A}x \leq \bar{b},\ x \geq 0\}$. O programa "`poliedro`" considera novos vértices e constrói um poliedro aberto, não convexo (Figura 3.20).

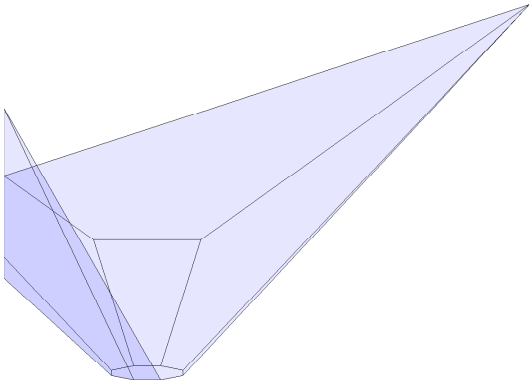


Figura 3.20: Poliedro aberto e não convexo

Ambos os poliedros ilimitados Q e R, o programa "`poliedro`" não consegue contruí-los. Mas utilizando outra abordagem, conseguimos uma "boa idéia" da estrutura desses poliedros ilimitados. Este processo é construído acrescentando restrições de caixa. O tamanho da caixa é analisado a partir de cada poliedro a ser construído. Veja Figura (3.21).

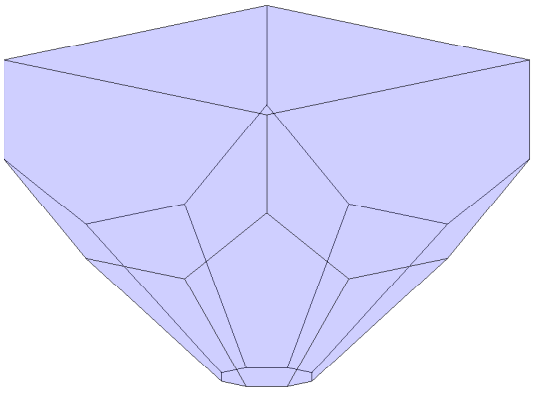


Figura 3.21: Poliedro ilimitado com o acréscimo de restrições de caixa

4 *Poliedros em Programação Linear, Método Simplex*

Programação linear pode ser vista como parte de um movimento revolucionário que deu à humanidade a capacidade de formular temas gerais e estabelecer uma trajetória de decisões específicas a serem tomadas a fim de "melhor" alcançar esses objetivos quando confrontados com situações práticas de grande complexidade. Nossas ferramentas para fazer isto são maneiras de formular problemas do mundo real em termos matemáticos detalhados (modelos), técnicas para resolver os modelos (algoritmos), e motores para executar os passos dos algoritmos (computadores e softwares).

George B. Dantzig

4.1 Pioneiros da Programação matemática

Historicamente, o que se sabe sobre a programação matemática é que em 1823, Fourier, e quase cem anos depois, em 1911, o famoso matemático belga C. J. de la Vallée Poussin, escreveram documentos a respeito desse assunto. Mas para a época, tais documentos não tinham significado prático. Já em 1939, o notável Leonid Kantorovich escreveu uma monografia sobre o tema, mas também foi negligenciada por razões ideológicas na URSS.

Durante o período da Segunda Guerra Mundial (1941-1945), George B. Dantzig, trabalhou no Pentágono tornando-se um especialista em programação utilizando métodos de planejamento. Em 1946, Dantzig era assessor matemático para o Controle da Força Aérea Americana no Pentágono, quando foi desafiado por seus colegas de trabalho, D. Hitchcock e M. Wood, a encontrar um processo de planejamento capaz de calcular em tempo hábil a implantação, treinamento e o programa de fornecimento logístico.

No verão de 1947, Dantzig propôs o método simplex. Mas, o método demorou quase um ano antes de Dantzig e seus colegas do Pentágono perceberem o quão poderoso ele realmente era. No mesmo ano, Dantzig consultou John von Neumann para obter sugestões técnicas que

poderiam contribuir nos resultados propostos na teoria do método simplex. Num recente trabalho feito por von Neumann e O. Morgenstern sobre teoria dos jogos, Dantzig descobriu o lema de Farkas e aprendeu sobre a dualidade pela primeira vez. Por tais descobertas e pela promessa da existência de computadores eletrônicos, o que seguiu foi um excitante período de desenvolvimento da programação matemática. Assim, logo tornou-se evidente que uma surpreendente gama de problemas aparentemente independentes poderiam ser formulados em termos de programação linear e, o mais importante, resolvidos pelo método simplex.

> *... a promessa de que existiriam computadores eletrônicos em breve, a exposição de matemáticos e economistas teóricos a problemas reais durante a guerra, o interesse em estruturar o processo de planejamento e, por último mas não menos importante, a disponibilidade de dinheiro aplicado para essa investigação, fizeram que todos convergissem para este tema durante o período 1947-1949. "The time was ripe". A investigação realizada em exatamente dois anos é, em minha opinião, um dos mais notáveis acontecimentos da história.*
>
> *... o rápido avanço da ciência de decisões, poucos lembraram das contribuições dos grandes pioneiros para que isso acontecesse. Alguns dos nomes são von Neumann, Kantorovich, Leontief e Koopmans.*
>
> *George B. Dantzig*

No início dos anos 50, muitas áreas que nós chamamos coletivamente programação matemática começaram a surgir. Programação não linear, programação inteira, otimização combinatória, programação estocástica, entre outras novas áreas cresceram rapidamente com a programação linear desempenhando um papel fundamental no desenvolvimento de cada uma delas.

Na década de 1970, por duas vezes, a programação linear chamou a atenção do público. Em 14 de outubro de 1975, a Royal Academy of Sciences da Suécia condecorou com o Prêmio Nobel em ciência econômica L. V. Kantorovich e T. C. Koopmans pelas suas contribuições para a teoria de locação ótima de recursos. (Como o leitor deve saber, não há Prêmio Nobel em matemática. Aparentemente, a Academia considerou o trabalho de G. B. Dantzig, que é universalmente reconhecido como o pai da programação linear, como sendo demasiado matemático.) Na foto ao lado estão, Koopmans, Dantzig e Kantorovich (Luxemburgo, 1976). O segundo evento foi ainda mais dramático. Desde a

invenção do método simplex, matemáticos estavam a procura de um algoritmo teoricamente satisfatório para resolver problemas em programação linear. Uma explicação para isso, estava nos critérios teóricos para avaliar a eficiência de algoritmos sendo diferentes das práticas existentes. Assim, um algoritmo como o método simplex, que é eminentemente satisfatório em aplicações práticas, pode ser considerado teoricamente insatisfatório. O inverso é também verdade: algoritmos teoricamente satisfatórios podem ser inúteis na prática.

A descoberta ocorreu em 1979, quando L. G. Khachiyan publicou, *''A polynomial algorithm in linear programming"*, um algoritmo de *elipsóides*, baseado em trabalhos anteriores de Shor, e de Yudin e Nemirovskii. Jornais de todo o mundo publicaram relatórios deste resultado, e em 1982, Khachiyan recebeu o Prêmio Fulkerson por este feito.

Figura 4.1: Dantzig e Khachiyan (Asilomar, 1990)

Programação não linear teve início por volta de 1951 com as condições de Karush-Kuhn-Tucker que estão relacionadas com as Condições Fritz-John (1948). Em 1954, Ragnar Frisch (que mais tarde recebeu o primeiro prêmio Nobel de Economia) propôs um método não linear de pontos interiores para resolver programas lineares. Propostas anteriores, como as de Von Neumann e Motzkin também podem ser encaradas como métodos de pontos interiores. Mais tarde na década de 1960, G. Zoutendijk, T. Rockafellar, P. Wolfe, M. Powell, Fiacco e McCormick, e outros desenvolveram a teoria da programação não linear e ampliaram o conceito de dualidade.

Já a programação inteira, que veremos no próximo capítulo, começou em 1958 por R. E. Gomory. Ele mostrou como fez para gerar sistematicamente os planos de "corte" (*cutting planes*). Os cortes adicionais são condições extras necessárias que quando adicionadas a um sistema existente de desigualdades garantem que a solução do problema de otimização seja inteira. Ellis Johnson da IBM, Egon Balas e muitos outros, expandiram as idéias de Gomory. Após, o amadurecimento dos resultados, "branch-and-bound" revelou-se uma das mais bem sucedidas maneiras de resolver problemas inteiros na prática. As técnicas mais eficientes parecem ser aquelas que combinam planos de corte e branch-and-bound.

Os relatos históricos acima mencionados foram baseados em Chvátal (1983), Bachem, Grötschel e Korte (1983), Dantzig (1991), Nocedal e Wright (1999) e Dantzig e Thapa (1997).

4.2 Resolução gráfica em programação linear

Problemas de programação linear, exceto em casos especiais, necessitam de apoio computacional para resolvê-los, pois sua complexidade varia de acordo com o número de restrições e variáveis envolvidas. Modelos de programação linear, na prática, podem ser muito grandes, alguns têm milhares de restrições e variáveis. Por exemplo, destacamos o trabalho do professor Jacek Gondzio com problemas que chegam a 352 milhões de restrições e 1 bilhão de variáveis (GONDZIO, 2007). Para abordar esse problemas de grande porte, além do apoio computacional, são necessárias técnicas complexas e bem elaboradas. Mas quando o problema linear tem exatamente duas (ou três) variáveis, sujeito a restrições de desigualdade (formato dual e canônico), é possível resolvê-lo graficamente.

Considere o seguinte problema de programação linear:

$$\begin{aligned} \text{minimizar} \quad & -x_1 - 3x_2 \\ \text{sujeito a} \quad & 3x_1 + 5x_2 \leq 15 \\ & -x_1 - x_2 \leq -1 \\ & -x_1 + 2x_2 \leq 4 \\ & 2x_1 - 3x_2 \leq 6 \\ & x \geq 0. \end{aligned} \tag{4.1}$$

Para resolvermos o problema graficamente, primeiro construímos a região viável, Figura (4.2). Neste caso, a região viável é o conjunto de pontos com coordenadas (x_1, x_2) que satisfazem todas as restrições. Isto é, o conjunto viável é representado como uma intersecção finita de semi-espaços fechados (Poliedro). Do capítulo 3, os semi-espaços fechados são representados pelas restrições do problema, que se encontram no formato canônico.

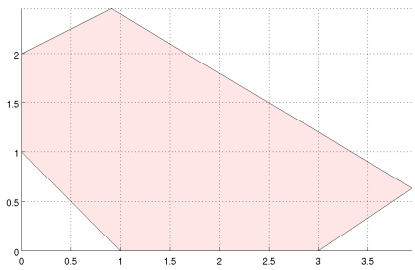


Figura 4.2: Região viável

Em seguida, construímos a função objetivo passando pela origem (hiperplano $-x_1 - 3x_2 = 0$). Note, Figura (4.2), que a origem não pertence ao conjunto viável, e portanto não é uma solução viável do problema. Aqui representamos o custo

pela região de custo constante (curva de nível). Veja Figura (4.3).

Representando graficamente a normal no hiperplano e avaliando a função objetivo na mesma direção da normal, mas com sentido contrário (direção de Cauchy), diminuímos o valor da função objetivo até encontrar um ponto viável. Neste exemplo, o hiperplano suporte $-x_1 - 3x_2 = -1$ intercepta o poliedro no ponto $(1,0)$, que não é um ponto ótimo, pois transladando o hiperplano no mesmo sentido, encontramos valores cada vez menores da função objetivo. Assim deslocamos o hiperplano até encontrar o seu valor mínimo na região viável com o ponto azul (i.e., até a curva de nível da função objetivo definir um hiperplano suporte $-x_1 - 3x_2 = \frac{-91}{11}$, que intercepta o poliedro no ponto ótimo $(\frac{10}{11}, \frac{27}{11})$ e tem como valor ótimo $\frac{-91}{11}$). Veja figura (4.3).

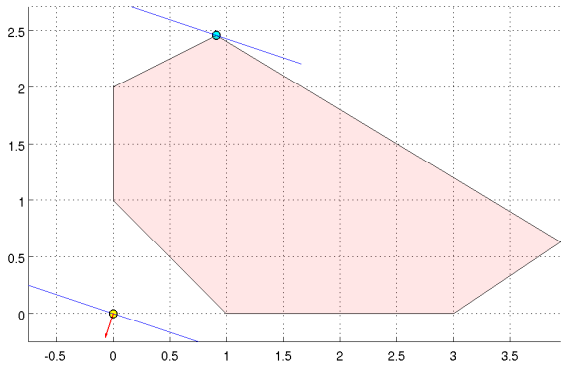


Figura 4.3: Solução gráfica em $\mathbb{R}^2$

A mesma idéia de resolução gráfica de um problema de programação linear bidimensional, pode ser expandida para resolver problemas tridimensionais.

A grande dificuldade na resolução desses problemas em $\mathbb{R}^3$ é construir a região viável. Mas se a região viável for um politopo, podemos utilizar o programa `POLIEDRO_V1`, e obter a visualização do "sólido"que determina essa regiões.

Analogamente ao caso bidimensional, resolveremos um problema tridimensional onde o custo será representado por um paralelogramo proporcional ao tamanho do poliedro, contido na região de custo constante (conjunto de nível).

Considere o problema:

$$\begin{aligned} \text{minimizar} \quad & x_1 - \tfrac{1}{2}x_2 - x_3 \\ \text{sujeito a} \quad & Ax \leq b, \\ & x \geq 0, \end{aligned} \tag{4.2}$$

em que,

$$A = \begin{bmatrix} 1 & 0 & 1 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & -1 & 1 \\ 1 & 1 & 0 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \\ 0 & -1 & -1 \\ -1 & 0 & -1 \\ 0 & -1 & -1 \\ -1 & -1 & 0 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 6 \\ 2 \\ 6 \\ 2 \\ 6 \\ 2 \\ 2 \\ 2 \\ 2 \\ 6 \\ 6 \\ 6 \\ -2 \\ -2 \\ -2 \end{bmatrix}$$

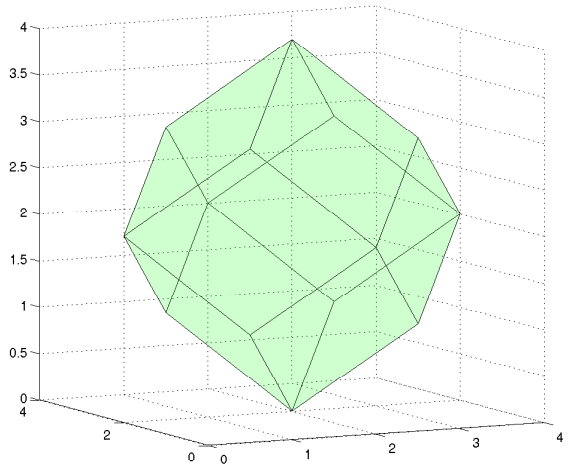


Figura 4.4: Região viável

Novamente, construímos o "custo" passando pela origem, hiperplano $x_1 - \frac{1}{2}x_2 - x_3 = 0$, e o transladamos na direção de Cauchy até tangenciar o poliedro no ponto azul. Assim, encontramos o ponto ótimo $\bar{x} = (1\ 3\ 3)^T$, que determina a solução ótima do problema igual a $-\frac{7}{2}$. Veja Figura (4.5).

Para uma melhor visualização da resolução do problema tridimensional, o programa `POLIEDRO_V1` utilizou ferramentas adicionais, do Matlab, como rotação, grade (coordenadas) e transparência.

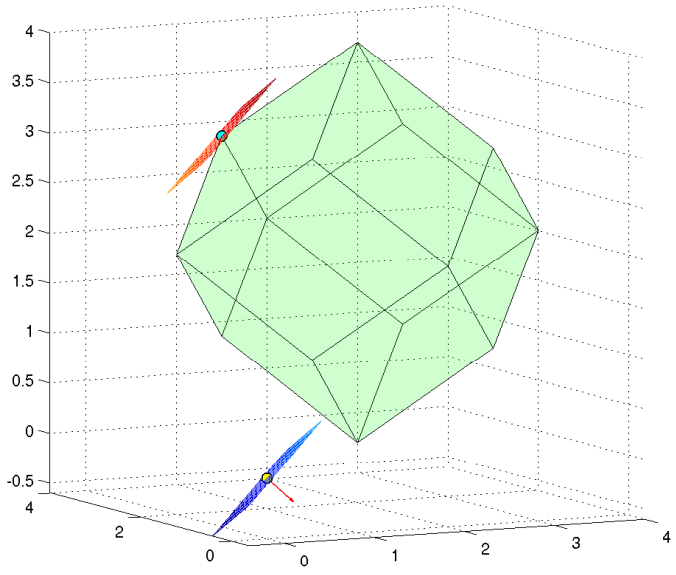


Figura 4.5: Solução gráfica em $\mathbb{R}^3$

4.3 Método Simplex

O objetivo deste capítulo é visualizar geometricamente a evolução do algoritmo Simplex num problema de Programação Linear, com região viável limitada. Mas para isso, antes estudaremos uma breve descrição do método simplex, mostrando a estrutura algébrica envolvida num problema geral de Programação Linear.

A teoria apresentada neste capítulo deriva de um curso básico de programação linear. Assim, nos limitamos apenas a algumas considerações importantes para o desenvolvimento do trabalho, já que o tratamento detalhado deste assunto foge do enfoque proposto a este texto. Por tais razões, alguns teoremas serão simplesmente enunciados, e indicado ao leitor textos onde são encontradas as demonstrações.

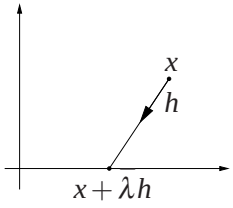
Teste da razão

Dados $x \in \mathbb{R}^n_+$ e $h \in \mathbb{R}^n$, o teste da razão se resume em encontrar $\bar{\lambda} \in \mathbb{R}$, tal que

$$\bar{\lambda} = \sup\{\lambda \mid x + \lambda h \geq 0\}.$$

Se $h \geq 0$, é fácil perceber que $\bar{\lambda} = +\infty$. Caso contrário, podemos associar a cada componente $i = 1, \ldots, n$ tal que $h_i < 0$, um passo máximo determinado por

$$\lambda_i = -\frac{x_i}{h_i}.$$



Note que $x_i + \lambda_i h_i = 0$. Para garantir $x + \lambda h \geq 0$, escolhemos o menor dos λ_i, isto é,

$$\bar{\lambda} = \min\Big\{ -\frac{x_i}{h_i} \mid h_i < 0 \Big\}.$$

4.3.1 Solução básica viável

Considere o poliedro Ω definido por:

$$\begin{aligned} Ax &= b \\ x &\geq 0, \end{aligned} \tag{4.3}$$

em que, $b \in \mathbb{R}^m$, A é $m \times n$, $m \leq n$ e posto$(A) = m$.

Dos teoremas (21) e (22), sabemos que Ω possui vértice, e que $v \in \Omega$, é um vértice se, e somente se, as colunas de $A_{I_+}(v)$ são linearmente independentes (L.I.). Assim, podemos verificar se um ponto x é vértice de Ω quando satisfizer:

(i) $x \geq 0$;

(ii) $Ax = b$;

(iii) $\{A_i \mid i \in I_+(x)\}$ é L.I.

Os itens (i) e (ii), caracterizam um ponto viável.

Definição 45 (Base de A) *Uma submatriz A_β é uma base de A se, e somente se, as colunas de A_β são base de $\mathbb{R}^m$. Isto é, β tem m elementos, $\{A_i \mid i \in \beta\}$, L.I.*

Dada uma base A_β, podemos resolver $A_\beta x_\beta = b$ e portanto o vetor $\bar{x} = \begin{bmatrix} x_\beta \\ 0 \end{bmatrix} \in \mathbb{R}^n$ (reordenando os índices se necessário), satisfaz

$$A\bar{x} = A_\beta x_\beta = b.$$

Se $\bar{x} \geq 0$, então $\bar{x}$ é solução viável.

Definição 46 *Chamamos de solução básica viável ao par* (x,β)*, em que* $x \in \mathbb{R}^n$ *e* $\beta \subset \{1,\ldots,n\}$*, com*

$$\begin{aligned} &A_\beta \ : \ \textit{base de } \mathbb{R}^m; \\ &x_\beta \geq 0; \\ &x_N = 0, \ \textit{em que, } N = \{1,\ldots,n\} \setminus \beta; \\ &Ax = b. \end{aligned}$$

Além disso, $x_i, \ i \in \beta$ *são denominadas as variáveis básicas e* $x_i, \ i \in N$ *as variáveis não básicas.*

Lema 17 *Considere o poliedro (4.3).*

1. *Se* (x,β) *é solução básica viável, então x é um vértice.*
2. *Se x é vértice de* Ω*, então existem um ou mais conjuntos* $\beta \subset \{1,\ldots,n\}$*, tal que* (x,β) *é solução básica viável.*

A demonstração do lema acima pode ser encontrada em Chvátal (1983), Nocedal e Wright (1999) e Gonzaga (2005b). Mais a frente, visualizaremos um exemplo em que um vértice possui mais de uma solução básica viável (vértice degenerado).

4.3.2 O Método

Considere o problema de programação linear

$$\begin{aligned} \text{minimizar} \ \ & c^T x \\ \text{sujeito a} \ \ & Ax = b, \\ & x \geq 0, \end{aligned} \tag{4.4}$$

em que, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times n}$ com $m < n$ e posto$(A) = m$.

Vimos nos capítulos anteriores que a região viável

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

é um poliedro, e se existir um conjunto ótimo $\widehat{\Omega}$, do problema (4.4), então este conjunto é uma face de Ω. Note que, tanto o problema (4.4), quanto o poliedro estão no formato primal. Observe também, que a face ótima $\widehat{\Omega}$, por si só, é um poliedro com o mesmo formato, e portanto contém pelo menos um vértice.

Inicialmente, vamos supor que um vértice inicial x^* é dado (i.e., existem um ou mais conjuntos $\beta \subset \{1,\ldots,n\}$ tal que (x^*,β) é uma solução básica viável). Nosso propósito é minimizar

a função objetivo, $z = c^T x$, a partir deste vértice. Isto é, encontrar uma direção de descida que seja viável e em seguida determinar o próximo ponto x^1, tal que $z(x^1) \leq z(x^*)$. Caso não seja possível determinar x^1, temos duas possibilidade: ou x^* é a solução ótima, ou o problema é ilimitado.

Geometricamente, o método simplex determina a direção a partir do vértice inicial x^*, escolhendo entre as arestas (faces de dimensão 1) adjacentes ao vértice x^* e que leva a uma redução do custo. Caso exista tal aresta, o método desloca-se sobre ela até encontrar o próximo vértice x^1. Repetimos este processo até obter uma solução ótima, ou determinar que o problema é ilimitado. Exemplos descritos geometricamente serão apresentados no final deste capítulo.

Sabemos que cada vértice é perfeitamente caracterizado por uma ou mais bases. Algebricamente, o método simplex trabalha com essas bases. Cada iteração inicia com uma solução básica viável (x^k, β^k), e termina quando: x^k é solução ótima, ou detecta-se que o problema não tem solução, ou encontra-se (x^{k+1}, β^{k+1}) solução básica viável de custo $c^T x^{k+1} \leq c^T x^k$.

Na iteração que consiste de um movimento por uma aresta adjacente ao vértice x^k a um vértice x^{k+1} (possivelmente $x^{k+1} = x^k$), a base β^k difere em exatamente uma componente da base β^{k+1}. O processo de percorrer uma aresta descrito geometricamente, se faz, algebricamente, permitindo que uma variável não básica $(i \notin \beta^k)$ aumente, deixando de ser nula. Faz-se um tese da razão para obter um novo vértice (x^{k+1}).

A cada iteração (mas não todas), o valor da função objetivo diminui. Isto não acontece se já estamos em uma solução ótima, ou o problema é degenerado, mantendo o custo constante (faremos um breve comentário sobre problemas degenerados mais adiante). Outro tipo de situação ocorre quando o problema é ilimitado, isto é, o passo que reduz a função move-se ao longo de uma aresta indefinidamente, sem encontrar outro vértice. Assim, as grandes questões em cada iteração do método simplex são: decidir qual aresta escolher (determinar qual o índice que entra e qual sai da base), e decidir se o passo é ilimitado. Para obter dicas sobre como estas decisões são tomadas no algoritmo simplex, relacionamos os dados de cada iteração (vértice x^k), com as condições de KKT. Veremos resultados teóricos destes fatos na próxima seção.

4.4 Algoritmo simplex

4.4.1 Problemas equivalentes

Considere os problemas (P_1) e (P_2) com o mesmo conjunto viável Ω:

$$(P_1)\quad \begin{array}{ll}\text{minimizar} & c_1^T x \\ \text{sujeito a} & Ax = b, \\ & x \geq 0.\end{array} \qquad \text{e} \qquad (P_2)\quad \begin{array}{ll}\text{minimizar} & c_2^T x, \\ \text{sujeito a} & Ax = b, \\ & x \geq 0,\end{array}$$

em que $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A_{m\times n}$ e $s \in \mathbb{R}^n$.

Definição 47 *Dizemos que (P_1) e (P_2) são equivalentes se têm o mesmo conjunto de soluções.*

Lema 18 *Se $c_1^T d = c_2^T d$ para toda direção d pertencente ao núcleo de A, $ker(A)$, então (P_1) e (P_2) são equivalentes.*

Dem. Dados dois pontos viáveis $x^1, x^2 \in \Omega$, temos $x^1 - x^2 \in \ker(A)$ e então $c_1^T x^1 - c_1^T x^2 = c_2^T x^1 - c_2^T x^2$. Daí se conclui imediatamente que $\bar{x}$ é solução ótima de (P_1) se e somente se é solução ótima de (P_2). ■

Lema 19 *Se $c_2 = c_1 + A^T y$ para algum $y \in \mathbb{R}^m$, então (P_1) e (P_2) são equivalentes.*

Dem. Para todo $d \in \ker(A)$,

$$c_2^T d = c_1^T d + d^T A^T y = c_1^T d + (Ad)^T y = c_1^T d,$$

pois $Ad = 0$. ■

4.4.2 O algoritmo

As condições de otimalidade do problema (4.4) são dadas por: $y \in \mathbb{R}^m$, $s \in \mathbb{R}^n$

$$A^T y + s = c, \tag{4.5}$$

$$Ax = b, \tag{4.6}$$

$$x,\ s \geq 0, \tag{4.7}$$

$$x_i s_i = 0, \qquad i = 1, 2, \ldots, n. \tag{4.8}$$

Se $s \in \mathbb{R}^n$ satisfaz a equação (4.5) com algum $y \in \mathbb{R}^m$, s é chamada "*folga dual*"[1]. Se $s \geq 0$, é denominada "*foga dual viável*".

[1] Vimos no capítulo 2, que a equação (4.5) corresponde ao problema dual do problema (4.4), e por essa razão s é denominada folga dual.

Inicialmente, supomos que nos é fornecida uma solução básica viável (x, β). Relembrando da definição (46), obtemos os conjuntos de índices $\beta \subset \{1,\ldots,n\}$ e $N = \{1,\ldots,n\} \setminus \beta$, juntamente com as submatrizes A_β e A_N, as quais correspondem às colunas de A com índices $i \in \beta$ e $i \in N$, respectivamente. Também particionamos as componentes de x, s e c, segundo os conjuntos de índices β e N, usando a notação

$$x_\beta = [x_i]_{i\in\beta}, \quad x_N = [x_i]_{i\in N}, \quad s_\beta = [s_i]_{i\in\beta}, \quad x_N = [s_i]_{i\in N}, \quad c_\beta = [c_i]_{i\in\beta} \quad \text{e} \quad c_N = [c_i]_{i\in N}.$$

As variáveis com índices $i \in \beta$ e $i \in N$ são denominadas *básicas* e *não básicas*, respectivamente.

Cada iteração do algoritmo inicia com uma solução básica viável. Das condições de KKT (4.5), (4.6) e (4.7), podemos extrair valores não apenas para a variável primal x, mas também para as variáveis duais y e s. Se ao calcularmos s, obtemos $s \geq 0$, então encontramos uma solução ótima. Senão, obtém-se outra solução básica viável.

Para constatarmos esses resultados, usaremos o fato de que c e s são custos equivalentes quando s é folga dual, isto é,

$$\begin{array}{rl} \text{minimizar} & c^T x \\ \text{sujeito a} & Ax = b, \\ & x \geq 0. \end{array} \qquad \text{equivalente a} \qquad \begin{array}{rl} \text{minimizar} & s^T x, \\ \text{sujeito a} & Ax = b, \\ & x \geq 0, \end{array}$$

como visto na seção (4.4.1), pois $s = c - A^T y$.

Vamos escrever uma iteração do algoritmo:

- Dados iniciais: solução básica viável (x, β), $\beta \subset \{1,\ldots,n\}$, $N = \{1,\ldots,n\} \setminus \beta$, x_β e x_N.
- Definimos $B = A_\beta$ e $\mathtt{N} = A_N$. Da condição de KKT (4.5), $Ax = Bx_\beta + \mathtt{N}x_N = b$ e portanto, $Bx_\beta = b$, pois $x_N = 0$.
- Calculamos y e s satisfazendo as condições de KKT (4.5), (4.6) e (4.5).

  Da condição de dualidade (4.5), temos que:

  $$\begin{bmatrix} B^T \\ \mathtt{N}^T \end{bmatrix} y + \begin{bmatrix} s_\beta \\ s_N \end{bmatrix} = \begin{bmatrix} c_\beta \\ c_N \end{bmatrix}$$

  Já a condição de complementariedade (4.7) nos induz a considerar $s_\beta = 0$ e sabendo que B é não singular, concluímos que y é a solução única de

  $$B^T y = c_\beta.$$

Consequentemente, temos que

$$s_N = c_N - \mathtt{N}^T y = c_N - (B^{-1}\mathtt{N})^T c_\beta .$$

- Variável que entra na base:

  Se $s \geq 0$, então x é solução ótima. Senão, escolhemos um índice i tal que $s_i < 0$. Como s e c são custos equivalentes para o problema (4.4), temos: se x_i aumentar (mantendo nulas as outras variáveis não básicas), então $s^T x$ diminui, pois $s_\beta = 0$ e $s^T x_N = s_i x_i$.

  Construindo a direção $d = \begin{bmatrix} d_\beta \\ d_N \end{bmatrix}$, em que d_N é igual a 1 na i-ésima coordenada e zero nas demais, e calculamos d_β para satisfazer $Ad = 0$, temos que:

$$Bd_\beta + \mathtt{N}d_N = 0,$$

  mas $\mathtt{N}d_N = A_i$. Portanto, d é obtido por $Bd_\beta = -A_i$, isto é,

$$d = \begin{bmatrix} -B^{-1}A_i \\ d_N \end{bmatrix}. \tag{4.9}$$

- Variável que sai da base:

  Caminhamos ao longo de $x + \lambda d$ até que alguma variável x_j, $j \in \beta$ se anule.

  Se $d \geq 0$, o problema é ilimitado. Senão, obtemos o passo λ, através do teste de razão, com $j \in \beta$ tal que $x_j + \lambda d_j = 0$.

- Fim da iteração: O novo ponto é dado por $x^+ = x + \lambda d$ e a nova base $B^+ = \{i\} \cup B \setminus \{j\}$.

Vamos mostrar que de fato B^+ é uma base, e portanto x^+ é um vértice. Sabemos por construção que $Bd_\beta = -A_i$, pelo teste da razão, $d_j < 0$. Devemos mostrar que A_i é linearmente independente das colunas A_k para $k \in \beta,\ k \neq j$, isto é, as colunas de A_{β^*} são L.I.. Este fato é consequência direta do seguinte lema:

Lema 20 *Dados vetores $u^1, u^2 \ldots, u^n \in \mathbb{R}^n$ linearmente independentes e $v = \lambda_1 u^1 + \lambda_2 u^2 + \cdots + \lambda_n u^n$. Se $\lambda_1 \neq 0$ então $v, u^2, \ldots, u^n$ são L.I..*

Dem. Suponha que $v = \alpha_2 u^2 + \cdots + \alpha_n u^n$, temos $\lambda_1 u^1 = (\alpha_2 - \lambda_2)u^2 + \cdots + (\alpha_n - \lambda_n)u^n$, com $\lambda_1 \neq 0$, o que contradiz a independência linear de $u^1, u^2 \ldots, u^n$. ■

Algoritmo 4.1: Um passo do algoritmo Simplex

Dados: $c, A \in \mathbb{R}^{m\times n}$, $b, x \in \mathbb{R}^n$, β vetor de índices das variáveis básicas, N vetor de índices das variáveis não básicas, com $x_\beta \geq 0$ e $x_N = 0$.

Definimos B, $\mathtt{N}$, c_β e c_N;

Calculamos as folgas duais:

- $y = B^{-1}c_\beta$;
- $s_N = c_N - \mathtt{N}^T y$;

Variável que sai da base: calcular $j \in N$ tal que

$s_j = \min(s_N)$;

Se $s_j \geq 0$ **então**
Solução ótima encontrada.

Senão
$i = N(j)$ (x_i entra na base);

Fim-se

Direção de decréscimo:

- $d_\beta = -B^{-1}A_i$;
- $d_i = 1$, os demais termos que compõem d_N, são zeros;

Variável que sai da base:

Se $d \geq 0$ **então**
Problema ilimitado.

Senão
Encontrar o passo λ utilizando o teste da razão e $j \in \beta$ tal que $(x_j + \lambda d_j = 0)$.

Fim-se

Definimos a nova base, trocando os índices j por i:

- $\beta = \{i\} \cup \beta \setminus \{j\}$;
- $N = \{j\} \cup N \setminus \{i\}$;

$x = x + \lambda d$.

4.4.3 Detalhes sobre o algoritmo simplex

1. Obtenção de uma solução básica inicial.

   Obter uma solução básica inicial é tão difícil quanto resolver o problema de P.L. a partir de um vértice dado. O método é dividido em duas fases:

Fase I: Cria-se um novo problema utilizando m variáveis "artificiais", $x_{n+1}, \ldots, x_{n+m}$ e a solução deste problema é uma solução básica para o problema original. A Fase I

consiste em resolver o seguinte problema:

$$\begin{aligned} &\text{minimizar } x^*_{n+1} + \cdots + x^*_{n+m} \\ &\text{sujeito a } \quad A^*x^* = b, \\ &\qquad\qquad x^* \geq 0, \end{aligned} \tag{4.10}$$

em que $A^* = [A\ E]$, com E diagonal $\begin{cases} E_{jj} = +1 \text{ se } b_j \geq 0 \\ E_{jj} = -1 \text{ se } b_j < 0 \end{cases}$.

O ponto inicial é dado por:

$$\bar{x} = \begin{bmatrix} 0 \\ |b_j| \end{bmatrix},$$

em que as variáveis nulas correspondem às primeiras n variáveis originais, e $|b_j| = \bar{x}_{n+j}$, com $j = 1, \ldots, m$ correspondem as variáveis artificiais.

Note que $\bar{x}$ satisfaz $A^*x^* = b$ e que para a base $\beta = \{n+1, \ldots, n+m\}$, B é não singular. A Fase I usa o método simplex e termina ao minimizar o problema acima com custo zero.

Fase II: Partindo da solução básica encontrada pela Fase I, devemos minimizar o problema original com um acréscimo de variáveis no custo, isto é,

$$c^* = \begin{bmatrix} c \\ 0 \end{bmatrix},$$

em que a parte nula de c^* representa um vetor nulo com m entradas.

As variáveis artificiais devem permanecer sempre nulas. Para isto, basta desconsiderá-las ao escolher a variável que entra na base em cada iteração. Em geral, podem-se eliminar as variáveis artificiais na Fase II. No entanto, pode ocorrer que alguma variável artificial (com valor nulo) pertença à base obtida na Fase I. Essas variáveis devem ser mantidas nulas na Fase II, isto é, as variáveis artificiais que pertencem à base não podem alterar seus valores.

2. Degeneração.

   Considere uma solução básica viável (x, β). Se $x_\beta > 0$, então a solução é chamada *não degenerada*. Neste caso é fácil perceber que o passo λ, na iteração do simplex a partir da solução básica, é positivo ($\lambda > 0$), pois uma das variáveis básicas se anula. O passo é "útil" e o custo diminui. A base muda e o algoritmo nunca volta ao ponto x.

   Se $x_\beta \not> 0$, então a solução básica é *degenerada*. O vértice tem menos de m variáveis positivas, e pode ser associado a várias bases diferentes. Pode ocorrer que, para alguma escolha da variável que entra na base, o passo λ resultante seja nulo. Neste caso o custo se mantém constante, o ponto não muda, mas a base muda nesta iteração. Na seção (4.5.1), visualizaremos o desenvolvimento do algoritmo simplex em que este fato ocorre.

   Muito raramente, é possível que ocorra um ciclo: após alguns passos nulos, repete-se a base β, o algoritmo entra em *loop* e falha. Isto depende das escolhas de quem entra e sai da base. Um exemplo deste fato é apresentado em Chvátal (1983).

4.4.4 Teoria por trás do método simplex

Esta seção tem como objetivo apresentar alguns resultados teóricos importantes que sustentam as afirmações apresentadas pelo método simplex. As demonstrações destes resultados estão em Dantzig e Thapa (1997), Nocedal e Wright (1999), Izmalov e Solodov (2007) e Chvátal (1983).

Para os teorema a seguir, consideremos um problema de programação linear geral, no formato primal, definido por (4.4), em que as condições de KKT são dadas por (4.5)–(4.8).

Teorema 23 (Teste de Otimalidade) *Dada uma solução básica viável $(\bar{x}, \bar{\beta})$, $\bar{x}$ é uma solução ótima com custo $z = c^T \bar{x}$, se s for folga dual viável, isto é,*

$$s_i \geq 0, \quad \textit{para } i = 1, \ldots, n.$$

Teorema 24 (Múltiplos mínimos) *Dada uma solução básica viável $(\bar{x}, \bar{\beta})$, em que $\bar{x}$ é solução ótima do problema, qualquer outra solução viável (x, β), não necessariamente básica, com a propriedade que $x_i = 0$ para todo $\bar{s_i} > 0$, é também uma solução ótima. Por outro lado, qualquer outra solução viável (x, β), com a propriedade que $x_i > 0$ e $\bar{s_i} > 0$ para algum i, não pode ser uma solução ótima.*

Corolário 1 (Solução única) *A solução básica viável é a única solução ótima, se $\bar{s_i} > 0$ para todas as variáveis não básicas.*

Teorema 25 (Problema linear ilimitado) *Seja s folga dual associada a* (x, β)*, tal que* $s_i < 0$ *para algum i, e considere a direção d definida em (4.9). Se* $d \geq 0$ *então o problema é ilimitado.*

Teorema 26 (Processo finito sobre problema não degenerado) *Assumindo que cada iteração é não degenerada, o algoritmo simplex termina em um número finito de iterações.*

4.5 Visualização espacial do método simplex

Descrevemos anteriormente o processo geométrico do algoritmo simplex, o qual decide que direção tomar a partir de um vértice inicial. A direção é escolhida entre as arestas adjacentes a este vértice, tal que reduza o custo e determine o próximo vértice da iteração (quando possível).

A maioria dos textos descrevem o método simplex através de uma técnica denominada *tableau*, que determina uma sequência de vértices através do pivoteamento de uma tabela numérica. Esta técnica foi apresentado por Dantzig. Na sequência Chvátal (1983), desenvolveu um processo similar denominado *dicionário*. Neste texto não apresentamos tais técnicas[2], priorizamos a visualização do método simplex.

Nesta seção visualizaremos três exemplos que exibem o desenvolvimento do método simplex para problemas bidimensionais, tridimensionais, degenerados e não degenerados.

4.5.1 Visualização

Considere o problema bidimensional

$$\begin{aligned} \text{minimizar} \quad & -2x_1 - x_2 \\ \text{sujeito a} \quad & 3x_1 + 5x_2 \leq 15 \\ & -x_1 - x_2 \leq -1 \\ & -x_1 + 2x_2 \leq 4 \\ & 2x_1 - 3x_2 \leq 6 \\ & x \geq 0. \end{aligned} \tag{4.11}$$

Note que o conjunto viável está no formato dual. Já o método simplex trabalha com problemas no formato primal, portanto temos que introduzir as variáveis de folga para termos os dados de entrada para o algoritmo simplex. Por outro lado, não é possível visualizar a região viável no

[2]Estas técnicas são encontradas em Chvátal (1983), Vanderbei (1996), Dantzig e Thapa (1997), e em qualquer texto básico de programação linear.

formato primal, por tais razões, resolveremos o problema pelo simplex numa dimensão superior ao plano, e visualizaremos o resultado bidimensional.

Observe também que não foi apresentada uma solução básica viável, portanto o próprio algoritmo simplex acrescenta as variáveis artificiais (aumentando ainda mais a dimensão do problema), e calcula um vértice inicial. A seguir, apresentamos o desenvolvimento do algoritmo simplex na fase 2 para o problema bidimensional através de figuras.

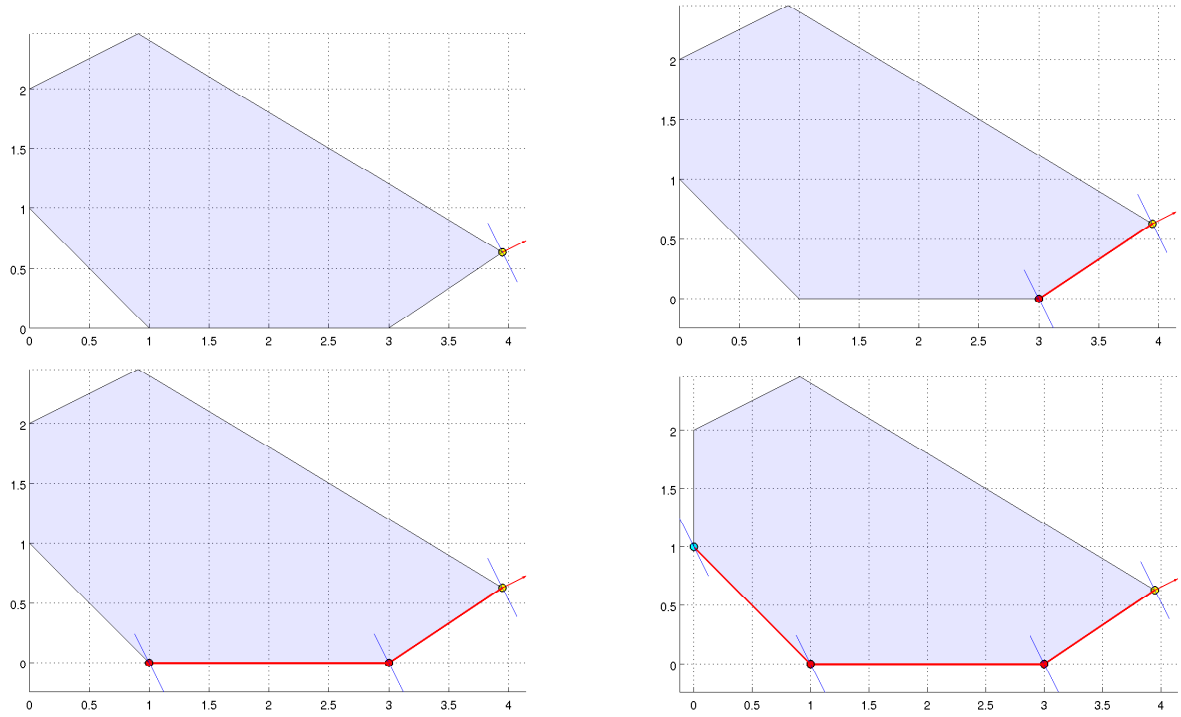


Figura 4.6: Representação passo a passo do algoritmo simplex num problema bidimensional.

Em todas as figuras, a região viável é representada pelo politopo azul. Na primeira figura destacamos o vértice inicial x^0, em amarelo, determinado pela fase 1, juntamente, com o hiperplano $c^T x = c^T x^0$ representado por um segmento de reta azul passando pelo ponto inicial e com vetor normal destacado em vermelho. Já na segunda figura, o algoritmo simplex decide, entre as duas arestas adjacentes ao vértice inicial, qual reduz mais o custo e através desta aresta encontra o próximo vértice $(3,0)$. Na terceira figura, só existe uma escolha da direção que determina o vértice $(1,0)$. Note que no passo 2 e 3, o hiperplano não é suporte ao politopo, portanto os vértices não podem ser soluções do problema. Na quarta figura, o simplex determina o vértice $(0,1)$, que é a intersecção do hiperplano suporte com o poliedro com o menor custo encontrado, isto é, o método simplex encontrou a solução ótima do problema. No processo cada iteração do método simplex resulta em um passo positivo (aresta em vermelho), que determina um novo vértice com custo inferior ao antecessor.

As cores dos pontos que são encontrados pelo método Simplex representam a seguinte relação:

Cor	Representação
●	Vértice inicial;
●	Vértice intermediário;
●	Vértice ótimo.

Analogamente, considere o problema tridimensional

$$\begin{aligned} \text{minimizar} \quad & \tfrac{1}{10}x_1 - \tfrac{1}{2}x_2 - x_3 \\ \text{sujeito a} \quad & Ax \leq b, \\ & x \geq 0, \end{aligned} \tag{4.12}$$

em que A e b são definidas por:

$$A = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & -1 \\ -1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & -1 & -1 \\ -8 & -8 & -11 \\ -8 & 8 & -11 \\ 8 & -8 & -11 \\ 8 & 8 & -11 \\ 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ -1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & -1 & 1 \\ 0 & 1 & 1 \\ 8 & 8 & 11 \\ 8 & -8 & 11 \\ -8 & 8 & 11 \\ -8 & -8 & 11 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 8 \\ 8 \\ -8 \\ 8 \\ -8 \\ -128 \\ 0 \\ 0 \\ 128 \\ 14 \\ -2 \\ 14 \\ -2 \\ 5 \\ 21 \\ 5 \\ 21 \\ 271 \\ 143 \\ 143 \\ 15 \end{bmatrix}$$

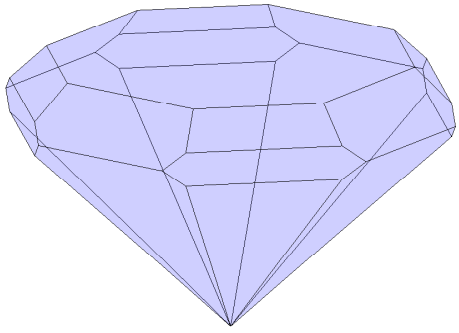
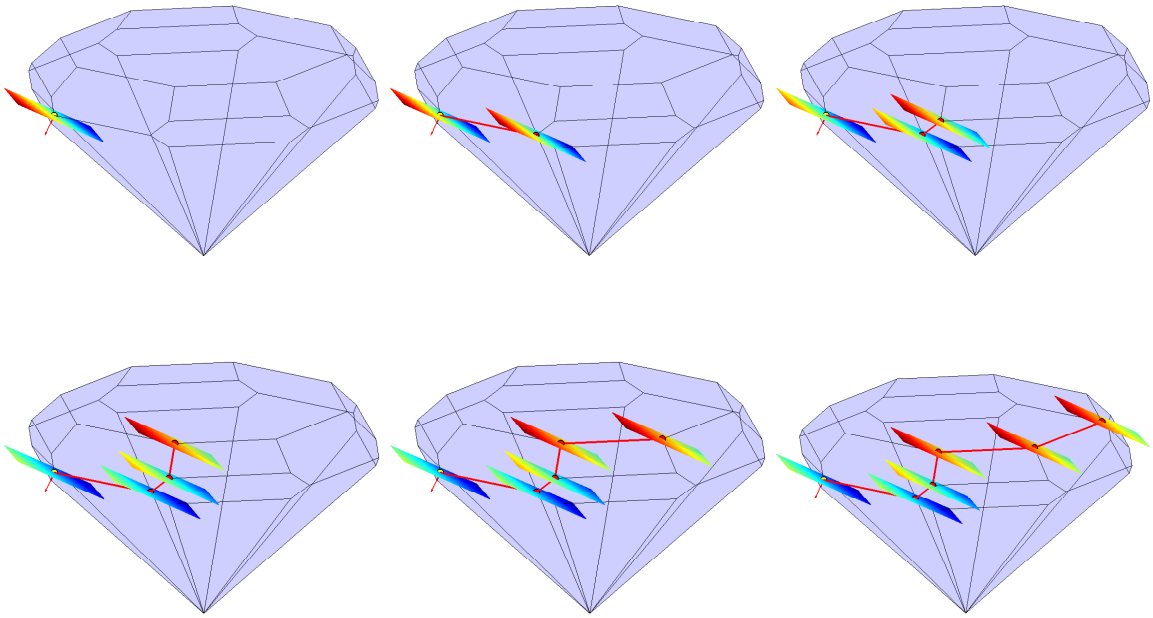


Figura 4.7: Diamante

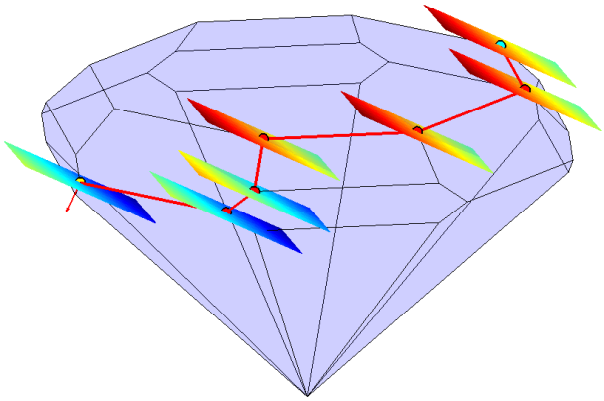
Este problema tridimensional é denomidado *diamante*, e foi baseado em Vanderbei (1996)

que apresenta na capa um diamante e uma trajetória sobre as arestas, representando os passos do método simplex.

A seguir, apresentamos o desenvolvimento passo a passo do método simplex na fase 2 para este problema. O custo será representado por um paralelogramo com vetor normal em vermelho centrado na solução básica inicial (ponto amarelo) determinada pela fase I do algoritmo simplex. Os demais passos não apresentam o vetor normal. Neste exemplo, cada iteração corresponde a uma figura, em que os vértices, juntamente com o custo, são plotados em sequência.



Da primeira à sexta figura, observe que os hiperplanos não são suportes, já na próxima figura, o método simplex encontra a solução do problema destacada pelo hiperplano suporte passando pelo ponto azul.



Problema degenerado

Considere o seguinte problema

$$\begin{array}{ll} \text{minimizar} & x_1 - \frac{1}{2}x_2 - x_3 \\ \text{sujeito a} & Ax \leq b, \\ & x \geq 0, \end{array} \tag{4.13}$$

em que A e b são definidas por:

$$A = \begin{bmatrix} 1 & 0 & 1 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & -1 & 1 \\ 1 & 1 & 0 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \\ 0 & -1 & -1 \\ -1 & 0 & -1 \\ 0 & -1 & -1 \\ -1 & -1 & 0 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 6 \\ 2 \\ 6 \\ 2 \\ 6 \\ 2 \\ 2 \\ 2 \\ 2 \\ 6 \\ 6 \\ 6 \\ -2 \\ -2 \\ -2 \end{bmatrix}$$

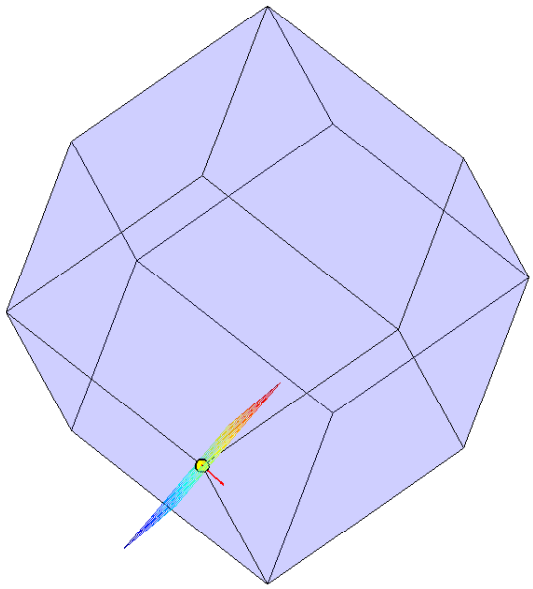
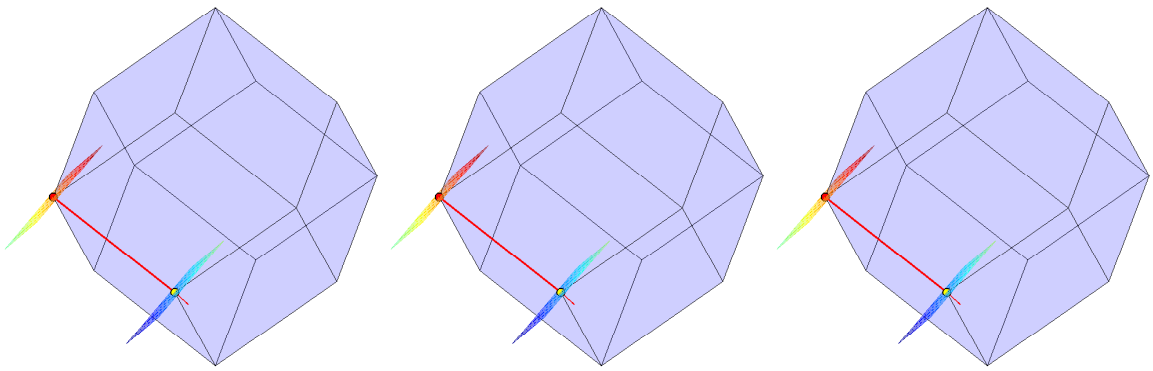


Figura 4.8: Região viável com o ponto inicial determinado pela fase I.

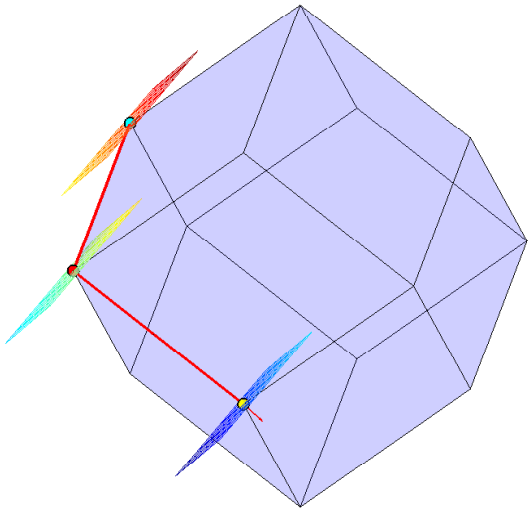
Após determinados a região viável e o ponto inicial, o método simplex efetua uma iteração determinando o segundo vértice, figura (a).



(a) (b) (c)

Nas próximas duas iterações, determinadas pelas figuras (b) e (c), respectivamente, o custo não se altera e o vértice no formato dual permanece o mesmo. Já no formato primal, as bases e os vértices são alterados mas o custo se mantém. Este fato ocorre, pois o vértice é degenerado. Utilizando a técnica *dicionário*, Chvátal (1983) apresenta exemplos algébricos de problemas degenerados (problemas que um mesmo vértice possui mais de uma base) e um exemplo especial, em que o método simplex gera um ciclo.

Na figura a seguir, o simplex determina a solução ótima do problema representado pelo ponto azul.



4.5.2 Programa `poliedro_simplex`

Nesta seção apresentaremos o programa que fez possível visualizar o desenvolvimento do método simplex dos exemplos propostos na seção 4.5.1.

Considere um problema de programação linear, constituído de um politopo como região viável, caracterizado por A, b e c no formato dual. O programa `poliedro_simplex` é composto basicamente pelos programas `poliedro` e `simplex`, que efetua dois passos principais:

O primeiro passo é descobrir todos os vértices que o algoritmo simplex percorre para definir a solução ótima do problema. Para isso, basta acrescentar uma matriz V aos dados de saída do programa simplex, correspondente a estes vértices. Note que os dados correspondentes ao problema estão no formato dual, assim antes de introduzirmos os dados de entrada no simplex devemos alterar o formato do problema (fomato primal).

O segundo passo é utilizar o programa `poliedro` para construir a região viável.

Note que no programa `poliedro_simplex`, o primeiro passo é feito separadamente do segundo. Assim, os vértices são simplesmente plotados numa ordem determinada pelo simplex, juntamente com os hiperplanos e a normal. Para plotar os hiperplanos e seu vetor normal fixamos uma proporção com relação ao tamanho do poliedro. Aqui, não entraremos em detalhes de como foram construídos os hiperplanos nem o vetor normal. Foram simplesmente utilizadas técnicas de programação em Matlab e geometria analítica.

O comando de chamada do programa `poliedro_simplex` é definido por:

```
poliedro_simplex(c,A,b);
```

Algoritmo 4.2: poliedro_simplex

Dados: $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$.

Definimos os vértices encontrados pelo simplex:

[V]=simplex(c,A,b);

Se $V = \emptyset$. **então**

Não é possível construir o poliedro, problema ilimitado ou mal posto.

Senão

Construção do politopo:

poliedro(A,b);

Seja k o número de vértices definidos em V.

Para $i = 1,\ldots,k$

Se V_i *é o vértice inicial.* **então**

plotar V_i, juntamente com o hiperplano e o vetor normal.

Senão

Construir a aresta, em vermelho, entre V_i e o vértice antecessor.

plotar V_i e o hiperplano.

Fim-se

Fim

Fim-se

5 *Poliedros em Programação Inteira, Método Branch and Bound*

O início da Programação Matemática é vinculado a Dantzig com a apresentação do Método Simplex. Já a Programação Inteira surge com o Algoritmo Plano de Corte proposto pelo engenheiro e matemático Ralph E. Gomory em 1958.

Gomory, após seu doutorado em Princeton, ingressou na Repartição de Física do Instituto de Investigação Naval da Marinha dos Estados Unidos, em Washington no ano de 1955. Além de seus afazeres, Gomory juntou-se ao Grupo de Investigação de Operações da marinha. Lá teve o primeiro contato com a Programação Linear, num curso de investigação dado por Alan Goldman.

Em 1957, Gomory voltou a Princeton como professor de matemática, onde conheceu o professor A. W. Tucker, então chefe do departamento e organizador de um grupo interessado na teoria dos jogos. Este grupo incluía Harold Kuhn e Martin Beale.

Gomory ainda mantinha contato com a Marinha dos Estados Unidos, como consultor. Em uma destas viagens de consultoria, um grupo apresentou um modelo de programação linear de uma força tarefa da marinha. Um dos apresentadores comentou que seria bom ter resultados inteiros ao invés de 1,3 porta-aviões. Pensando na observação levantada, Gomory, sentiu-se determinado a inventar um método que produzisse resultados inteiros, e associou imediatamente a seu conhecimento de programação linear.

Após algumas investigações, concluiu que o método deveria ter duas etapas. Na primeira o resultado do problema de Programação Linear é um limitante para o problema de Programação Linear Inteira. A segunda etapa consistia em produzir resultados inteiros, com a adição de restrições lineares suplementares.

No mesmo período, Martin Beale entra em contato com Gomory para que ministrasse um seminário em Princeton. Já haviam apresentado seminários sobre teoria de jogos e programação linear. Gomory aceitou o pedido e disse que teria prazer de dar uma palestra sobre a resolução de

problemas de programação linear inteira. Ouvindo o assunto do seminário, Martin respondeu: “mas isso é impossível”. Essa foi a primeira indicação para Gomory que outros tinham pensado sobre o problema.

Durante as emocionantes semanas que se seguiram, Gomory finalmente consegui elaborar uma prova e com o auxílio da teoria proposta por Dantzig, no início de 1958, Gomory faz a primeira apresentação pública do algoritmo plano de corte, dando origem à Programação Inteira.

Gomory ressalta ainda a colaboração do professor Tucker, que foi extremamente útil durante aquele período, como durante toda sua estada em Princeton. No verão de 1958, ao escrever o algoritmo em Fortran para o computador IBM 704, reuniram-se pela primeira vez na Rand Corporation os matemáticos Gomory, Lloyd Shapley, Herb Scarf, Dantzig, Richard Bellman e Phillip Wolfe. O programa Fortran apresentou falhas em alguns modelos, em que os maiores tinham de dez a quinze variáveis. A maior parte destes problemas chegaram na solução rapidamente, mas um desses modelos nunca chegava a uma resposta definitiva. Gomory pensou que talvez houvesse deixado “bugs”no programa, mas, na verdade, este foi o primeiro indício de problemas computacionais que viriam pela frente.

O relato histórico acima apresentado é encontrado em Gomory (1991). As coincidências entre os precursores da programação linear (Dantzig) e da programação inteira (Gomory) ficam claras quando nos colocamos no ambiente de desenvolvimento educacional e tecnológico da época, juntamente com o incentivo proposto pelas Forças Armadas Americanas. Mas o fato que merece maior destaque é a perspicácia dos matemáticos envolvidos na época, e seus círculos de amizades que tanto contribuíram à matemática de hoje.

5.1 Programação Linear Inteira

Caracterizamos um problema de programação linear como sendo um problema contínuo, em que a solução (caso exista) é real. Já em programação inteira, colocamos outra restrição sobre as variáveis, exigimos que sejam inteiras, o que caracteriza a Programação Inteira como sendo um problema discreto. Muitos problemas do mundo real podem ser modelados como problemas de programação linear, em que algumas ou todas as variáveis sejam inteiras. A necessidade de formular e resolver problemas de Programação Linear Inteira (PLI), veio inicialmente a partir do fato de que, em algumas das aplicações de Programação Linear (PL), soluções fracionárias eram indesejáveis - imaginar um pedido em que x_j é, por exemplo, o número de porta-aviões atribuídos à rota j.

Poderíamos pensar que estes problemas não seriam muito mais difíceis do que problemas

de programação linear, mas, como veremos neste capítulo, problemas de programação inteira geralmente são muito mais difíceis de resolver.

Uma abordagem natural para os problemas PLI, é resolver o correspondente problema de PL, caso exista solução, encontrar na vizinhança o ponto mais próximo com componentes inteiras. Esta é certamente uma estratégia plausível em muitos casos, especialmente quando a solução x esperada contém valores inteiros "grandes" e, sendo por conseguinte insensível aos arredondamentos. Há, no entanto, muitas limitações sérias para essa abordagem: O arredondamento para uma solução viável inteira pode não ser aceitável (solução não viável), além disso, a maioria das aplicações de PLI não são apenas PL, mas problemas que possuem a propriedade que as variáveis inteiras sejam apenas um dos dois valores, ou seja, zero ou um. Mais freqüentemente, um problema de PLI é o resultado da utilização de modelos combinatórios com restrições inteiras ou não lineares de diferentes tipos.

No entanto, existem outros problemas de programação inteira, em que as variáveis inteiras podem assumir qualquer valor inteiro não negativo. Por isso, consideramos o problema padrão de programação inteira da seguinte forma: Sejam $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$ e $b \in \mathbb{R}^m$.

$$\begin{aligned} \text{minimizar} \quad & c^T x \\ \text{sujeito a} \quad & Ax = b, \\ & x \geq 0, \\ & x \in \mathbb{Z}^n. \end{aligned} \tag{5.1}$$

Neste capítulo, discutiremos uma técnica para resolver problemas desta classe, chamado Branch-and-bound. A abordagem destes problemas através da Programação Linear, podem ser encontradas em Vanderbei (1996) e Papadimitriou e Steiglitz (1998).

5.2 Método Branch-and-Bound

O método Branch-and-bound envolve a resolução de um grande número de subproblemas lineares, na busca de uma solução ótima inteira. O branch em branch-and-bound refere-se ao processo de particionamento do problema original em sucessivos subproblemas de PL. O bound refere-se aos limites inferiores que são utilizados para a construção da prova de otimalidade sem pesquisas exaustivas.

O Algoritmo começa com a seguinte abordagem: Primeiramente ignoramos a restrição $x_i \in \mathbb{Z},\ i = 1,\ldots,n$, do problema (5.1), resultando num problema de Programação Linear deno-

minado *Problema relaxado*.

$$\begin{aligned} &\text{minimizar} && c^T x \\ &\text{sujeito a} && Ax = b, \\ & && x \geq 0. \end{aligned} \tag{5.2}$$

Vimos no capítulo anterior que o método Simplex resolve o problema relaxado (5.2) e obtem uma solução (caso exista) $\bar{x}$, que em geral não é inteira. Se a solução $\bar{x}$ é inteira, então o problema esta solucionado. Por outro lado, se $\bar{x}$ não é inteira, associamos o custo $c(\bar{x})$ desta solução ao menor valor da função objetivo a ser alcançada no problema relaxado, isto é, se calcularmos em qualquer solução viável inteira x^* temos que $c(\bar{x}) \leq c(x^*)$. Para encontrarmos uma solução viável inteira, teríamos agora que adicionar uma restrição (plano de corte) para o problema relaxado, de tal forma que não excluísse uma possível solução inteira. Aqui no entanto, vamos dividir o problema em dois subproblemas, acrescentando duas restrições lineares. Suponha que $\bar{x_i}$ é uma componente não inteira de $\bar{x}$, por exemplo. Assim, definimos os seguintes subproblemas.

$$\begin{aligned} (P_1)\quad &\text{minimizar} && c^T x \\ &\text{sujeito a} && Ax = b, \\ & && x \geq 0, \\ & && x_i \leq \lfloor \bar{x_i} \rfloor \end{aligned} \qquad \text{e} \qquad \begin{aligned} (P_2)\quad &\text{minimizar} && c^T x \\ &\text{sujeito a} && Ax = b, \\ & && x \geq 0, \\ & && x_i \geq \lfloor \bar{x_i} \rfloor + 1, \end{aligned} \tag{5.3}$$

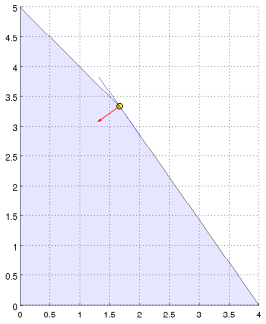
em que $\lfloor \bar{x_i} \rfloor$ é o maior inteiro menor que $\bar{x_i}$.

Para melhor compreender o processo, classificamos os subproblemas por níveis, isto é, o problema (5.2) representa o nível 0, já os problemas definidos em (5.3) pertencem ao nível 1, e assim sucessivamente[1]. O método Branch-and-bound torna-se claro ao resolvermos o seguinte exemplo. Este problema é encontrado em Vanderbei (1996), no entanto, como um problema de maximização.

Exemplo 1: Considere o seguinte problema:

$$\begin{aligned} &\text{minimizar} && -17x_1 - 12x_2 \\ &\text{sujeito a} && 10x_1 + 7x_2 \leq 40 \\ & && x_1 + x_2 \leq 5 \\ & && x \geq 0 \\ & && x \in \mathbb{Z}^2. \end{aligned} \tag{5.4}$$

[1]Branch-and-Bound pode ser interpretado através da teoria de grafos, em que o método Branch-and-Bound "varem" (busca em profundidade) uma árvore binária, a partir do nível 0 (raiz) até determinar uma solução inteira de custo mínimo (folha).



Definimos o problema relaxado p_0 (nível 0), desconsiderando a restrição inteira $x \in \mathbb{Z}$ e resolvemos o problema PL pelo Simplex determinado a solução $x^0 : x_1 = 1,67\ x_2 = 3,33$, com custo $c(x^0) = -68.33$. Veja figura ao lado. Note que, tanto x_1, quanto x_2 são fracionárias. Assim, para definirmos os subproblemas do nível 1, devemos escolher uma das variáveis para adicionar os planos de corte (Branch). A técnica adotada neste texto é escolher um índice j tal que

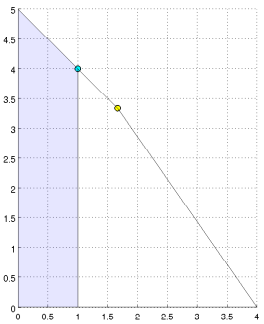
$$M_j = \max(M_i)_{i=1,\ldots,n},$$

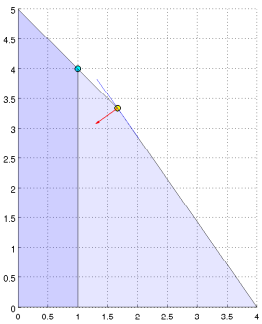
com $M_i = |x_i^0 - \text{round}(x_i^0)|,\ i = 1,\ldots,n$.

Neste exemplo, o nível 0 apresenta $M_1 = M_2$, por essa razão, escolhemos a variável x_1. Assim adicionamos os planos de corte, $x_1 \leq 1$ ou $x_1 \geq 2$, considerando esses dois casos separadamente. Ao definir os subproblemas relaxados, resolvemos primeiramente o problema da esquerda (busca em profundidade), deixando o problema da direita em aberto. Sejam os subproblemas do nível 1 o problema da esquerda (p_1) e o problema da direita, denotados por:

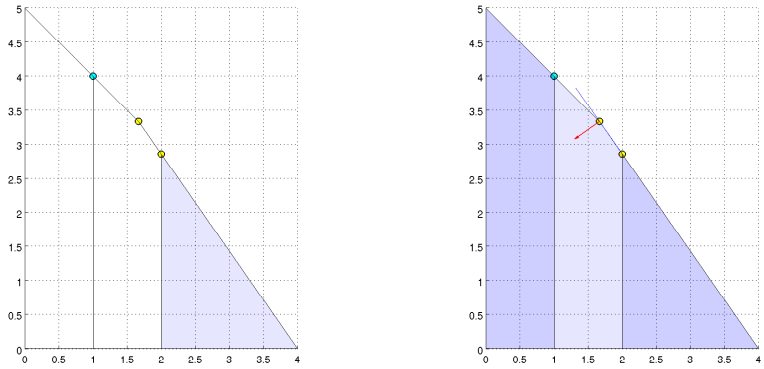
$$\begin{array}{lrcl} (p_1) & \text{minimizar} & -17x_1 - 12x_2 & \text{e} \\ & \text{sujeito a} & 10x_1 + 7x_2 \leq 40 & \\ & & x_1 + x_2 \leq 5 & \\ & & x_1 \leq 1 & \\ & & x \geq 0 & \end{array} \qquad \begin{array}{lr} \text{minimizar} & -17x_1 - 12x_2 \\ \text{sujeito a} & 10x_1 + 7x_2 \leq 40 \\ & x_1 + x_2 \leq 5 \\ & x_1 \geq 2 \\ & x \geq 0 \end{array} \tag{5.5}$$

Utilizando novamente o método Simplex, encontramos como solução $x^1 : x_1 = 1\ x_2 = 4$, com custo $c(x^1) = -65$. O algoritmo encontrou a primeira solução viável para o problema de Programação Linear Inteira, tal solução é tida como um limitante para os demais problemas (Bound), isto é, se a partir de agora um subproblema relaxado encontrar um custo superior a $c(x^1)$ o subproblema é desconsiderado. Nas figuras a seguir, uma representa o subproblema p_1 separadamente, e a outra os subproblemas p_0 e p_1 sobrepostos.





Aqui o "bound", $c(x^1)$, é representado pelo ponto $x^1 = (1,4)$ que está na cor azul. Além disso, como a solução é inteira, o problema p_1 não será mais particionado, por essa razão, passamos a resolver, com o auxílio do algoritmo Simplex, o problema da direita em (5.5) do nível 1, que definimos por p_2. A solução encontrada é dada por $x^2 : x_1 = 2\ x_2 = 2,86$, com custo $c(x^2) = -68,29$. Neste caso, a solução x^2 não é inteira e a variável x_2 é a única escolha



possível. Seja, p_3 e o problema da direita do nível 2 em aberto, os subproblemas denotados por:

$$
\begin{array}{lrcl}
(p_3) \quad \text{minimizar} & -17x_1 - 12x_2 & \text{e} & \text{minimizar} \quad -17x_1 - 12x_2 \\
\text{sujeito a} & 10x_1 + 7x_2 \leq 40 & & \text{sujeito a} \quad 10x_1 + 7x_2 \leq 40 \\
& x_1 + x_2 \leq 5 & & x_1 + x_2 \leq 5 \\
& x_1 \geq 2 & & x_1 \geq 2 \\
& x_2 \leq 2 & & x_2 \geq 2 \\
& x \geq 0 & & x \geq 0
\end{array}
\tag{5.6}
$$

Analogamente, resolvemos primeiro o problema da esquerda p_3, encontrando a solução $x^3 : x_1 = 2,6\ x_2 = 2$, com custo $c(x^3) = -68,2$.

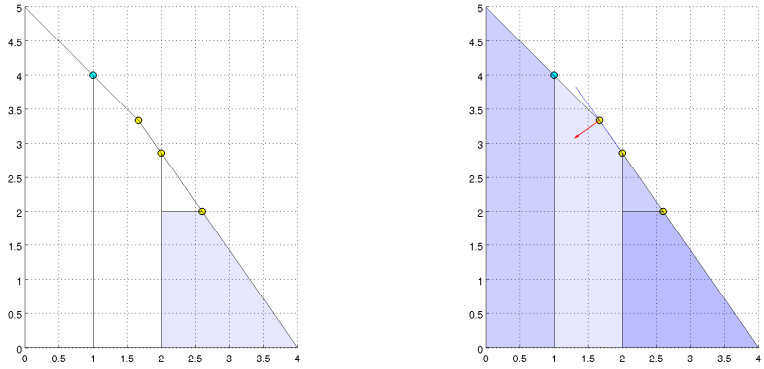


Figura 5.1: Subproblema p_3 e os demais subproblemas sobrepostos.

A solução x^3 não é inteira, assim escolhemos a variável x_1 para efetuar a partição dos subproblemas, com o acréscimo das restrições $x_1 \leq 2$ ou $x_1 \geq 3$. Considere os seguintes subproblemas do nível 3, p_4 e o problema à direita aberto:

$$
\begin{array}{lrcl}
(p_4) \quad \text{minimizar} & -17x_1 - 12x_2 & \text{e} & \text{minimizar} \quad -17x_1 - 12x_2 \\
\text{sujeito a} & 10x_1 + 7x_2 \leq 40 & & \text{sujeito a} \quad 10x_1 + 7x_2 \leq 40 \\
& x_1 + x_2 \leq 5 & & x_1 + x_2 \leq 5 \\
& x_1 \geq 2 & & x_1 \geq 2 \\
& x_2 \leq 2 & & x_2 \geq 2 \\
& x_1 \leq 2 & & x_1 \geq 3 \\
& x \geq 0 & & x \geq 0
\end{array}
\tag{5.7}
$$

Podemos representar os passos do método Branch-and-Bound até o nível 3, pelo seguinte diagrama.

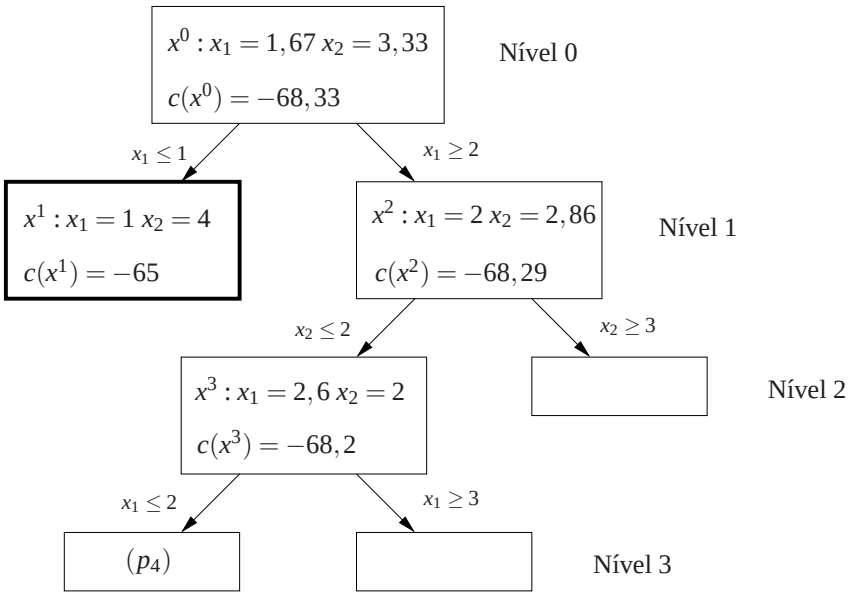


Figura 5.2: Árvore representando o desenvolvimento do método Branch-and-Bound até o nível 4.

Resolvendo o subproblema p_4 obtemos $x^4 : x_1 = 2\ x_2 = 2$, com custo $c(x^4) = -58$. O método encontrou outra solução inteira, mas se compararmos ao "bound" $c(x^1)$, observamos que o custo $c(x^4)$ é caro, isto é, $c(x^1) \leq c(x^4)$, por essa razão, x^4 é desconsiderado. Graficamente representamos o custo "caro" pelo ponto x^4 em vermelho. Observe também que o poliedro que representa o problema de programação linear relaxado é unidimensional. Veja Figura (5.3).

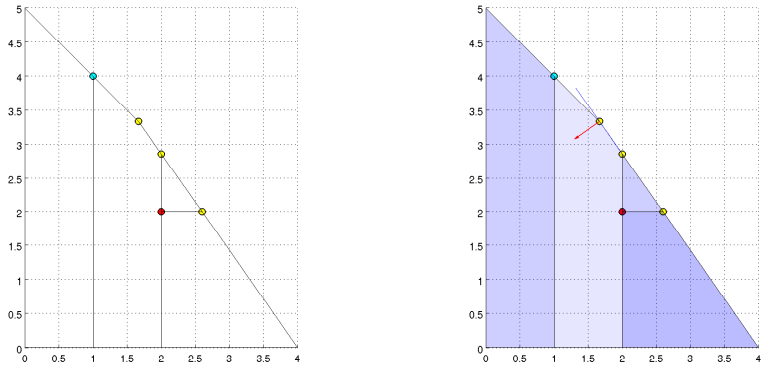
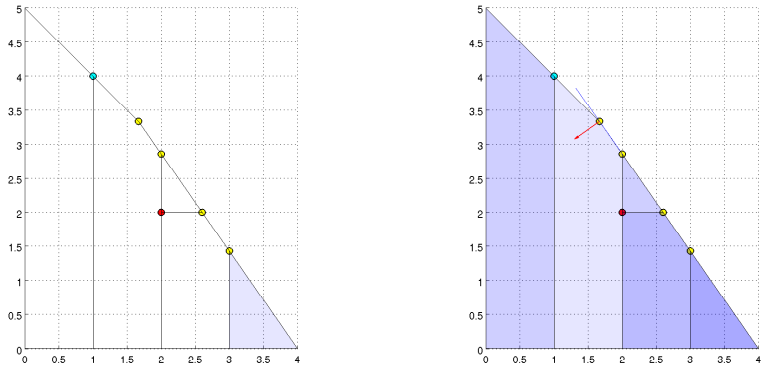


Figura 5.3: Representação gráfica desenvolvida até o subproblema p_4.

Analogamente ao nível 1, permanecemos no nível 3 e resolvemos o problema à direita definido em (5.7). Definimos esse problema de programação linear relaxado por p_5. A solução de p_5 é dada por $x^5 : x_1 = 3\, x_2 = 1,43$, com custo $c(x^5) = -68,14$.

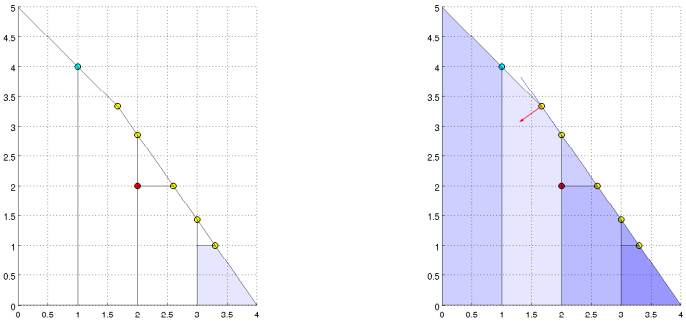


Os planos de corte são efetuados na variável x_2 pelas restrições $x_2 \leq 1$ ou $x_2 \geq 2$, e desta forma definimos os subproblemas do nível 4.

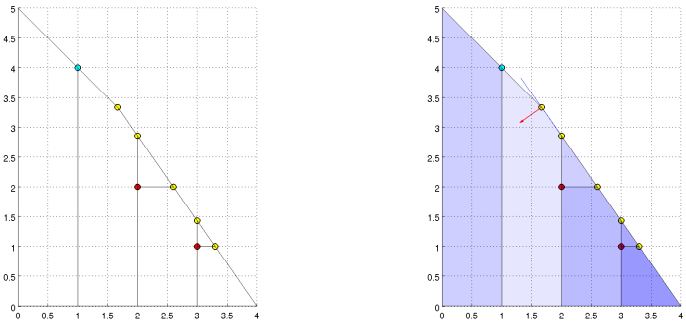
$$
\begin{array}{lrcl}
(p_6) \quad \text{minimizar} & -17x_1 - 12x_2 & \text{e} & \text{minimizar} \quad -17x_1 - 12x_2 \\
\text{sujeito a} & 10x_1 + 7x_2 \leq\ 40 & & \text{sujeito a} \quad 10x_1 + 7x_2 \leq\ 40 \\
& x_1 + x_2 \leq 5 & & x_1 + x_2 \leq 5 \\
& x_1 \geq 2 & & x_1 \geq 2 \\
& x_2 \leq 2 & & x_2 \geq 2 \\
& x_1 \leq 2 & & x_1 \geq 3 \\
& x_2 \leq 1 & & x_2 \geq 2 \\
& x \geq 0 & & x \geq\ 0
\end{array}
\tag{5.8}
$$

Aqui, definimos o subproblema à esquerda p_6 e deixamos o problema à direita aberto. Note, que ao adicionarmos os planos de corte, as restrições lineares suplementares fazem com que

o subproblema no nível antecessor tenha restrições redundantes, mas não as descartamos. Do Simplex segue que $x^6 : x_1 = 3,3\ x_2 = 1$, com o custo $c(x^6) = -68.1$.



No problema p_6, x_1 é a variável escolhida para se fazer o "branch", $x_1 \leq 3$ ou $x_1 \geq 4$. Definimos assim o problema à esquerda p_7, com solução $x^7 : x_1 = 3\ x_2 = 1$, com o custo $c(x^7) = -63$. Da mesma forma que em p_4 no nível 4, $c(x^7)$ é considerado "caro", e assim descartado.

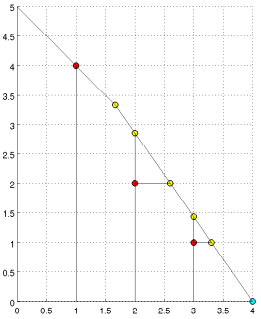


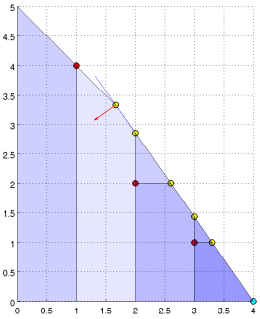
Passamos então para o problema da direita, que é definido por p_8, com o acréscimo da restrição $x_1 \geq 4$.

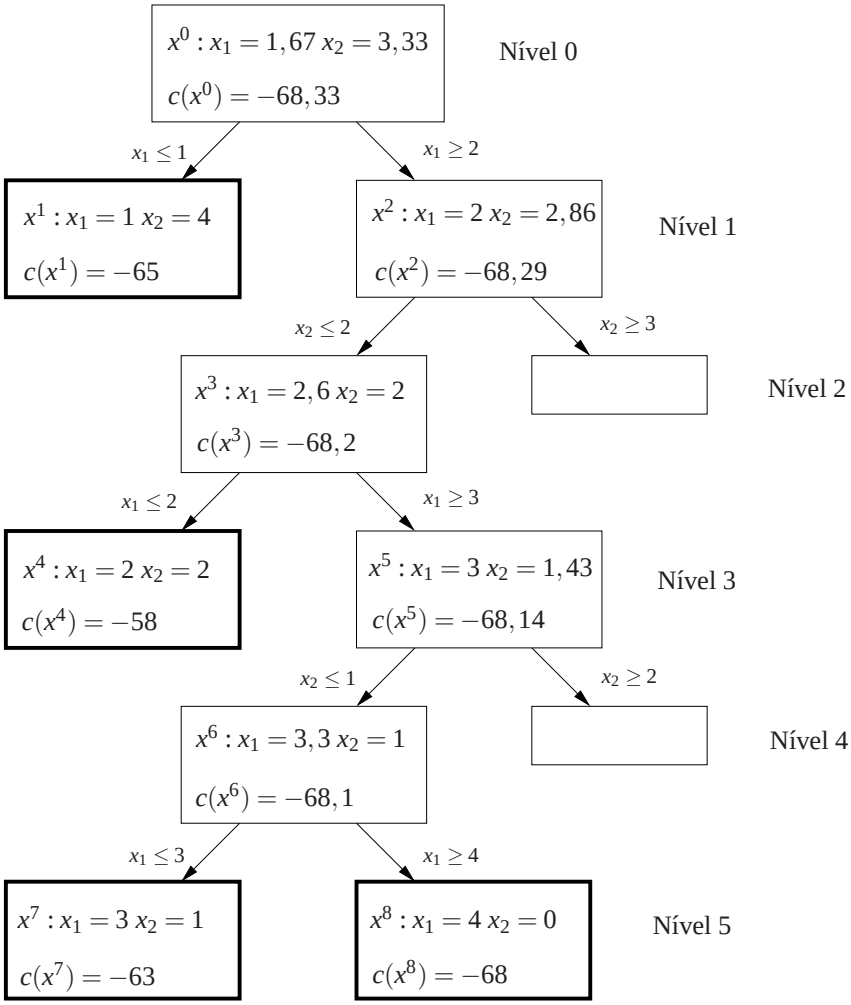
$$
\begin{aligned}
(p_8) \quad \text{minimizar} \quad & -17x_1 - 12x_2 \\
\text{sujeito a} \quad & 10x_1 + 7x_2 \leq 40 \\
& x_1 + x_2 \leq 5 \\
& x_1 \geq 2 \\
& x_2 \geq 2 \\
& x_1 \geq 3 \\
& x_2 \geq 2 \\
& x_1 \geq 4 \\
& x \geq 0
\end{aligned}
\tag{5.9}
$$

Resolvendo o problema p_8, encontramos a solução $x^8 : x_1 = 4\ x_2 = 0$, com custo $c(x^8) = -68$. Comparando o resultado com o "bound" $c(x^1)$, obtemos que $c(x^8) \leq c(x^1)$. Portanto

descartamos $c(x^1)$, agora "caro" e definimos o novo "bound" $c(x^8)$. Graficamente, o ponto x^1 é representado pela cor vermelho e o ponto x^8 como sendo o novo ponto azul. Observe também que o poliedro definido pelo problema (5.9) possui dimensão zero, isto é, o poliedro é formado por um único ponto.







A árvore binária acima, representa o desenvolvimento do método até o nível 5. Observe que tanto o problema da esquerda quanto o da direita no nível 5, obtiveram solução inteiras, assim voltamos ao nível 4 para resolver o problema à direita p_9, definido em (5.8). Mas ao

adicionar a restrição $x_2 \geq 2$ a região viável definida pelo subproblema p_9 é vazia, isto é, não existe ponto (x_1, x_2) do poliedro determinado pelo nível 0, tal que $(x_1, x_2) \geq (3, 2)$. Portanto, dizemos que o problema p_9 é inviável. Analogamente, analisamos o problema à direita p_{10}, no nível 2, definido por (5.6), e constatamos que ao acrescentar a restrição $x_2 \geq 3$ o problema torna-se inviável. Como não há mais subproblemas para serem analisados, determinamos a solução ótima inteira do problema dada por $x^* = (4, 0)$, com custo $c(x^*) = -68$. Note que a solução ótima inteira do problema não é um solução viável próxima da solução do problema relaxado no nível 0, isto é, se arredondássemos a solução do PL relaxado no nível 0, não encontraríamos a melhor solução. A seguir apresentamos a "árvore" completa do problema de PLI.

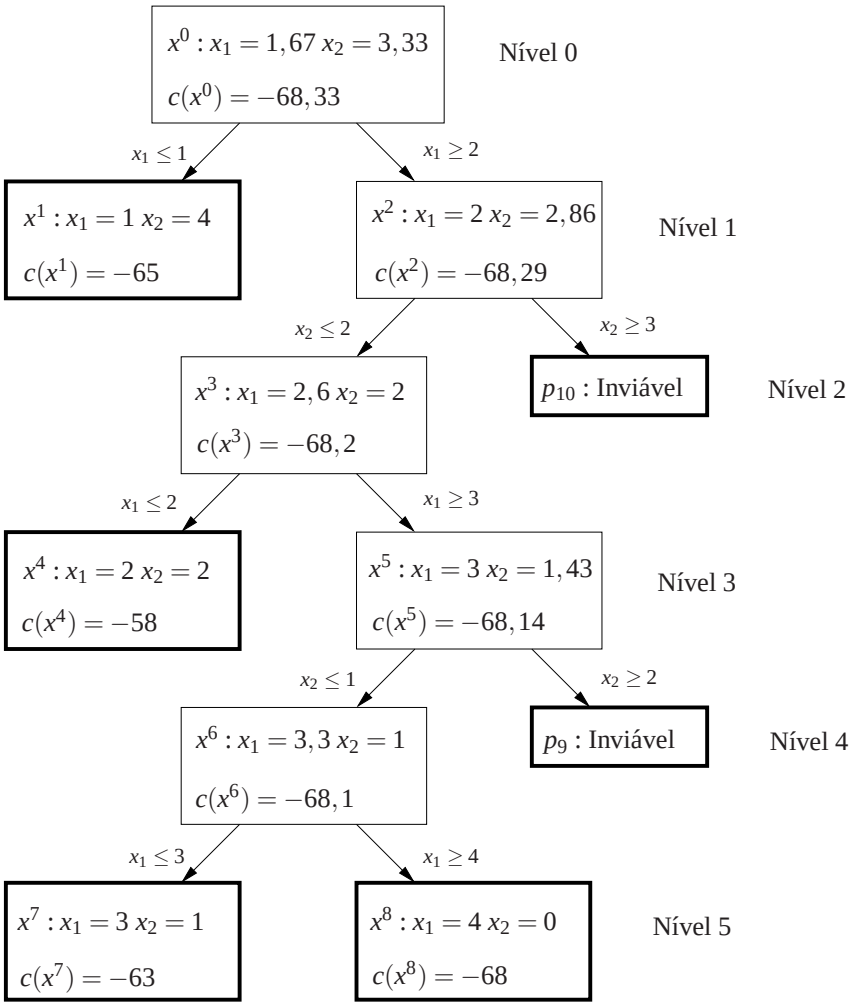


Figura 5.4: Árvore completa

No exemplo 1, apresentamos graficamente um problema de Programação Linear Inteira bidimensional. Na próxima seção, abordamos um exemplo de PLI tridimensional, utilizando o método Branch-and-Bound. É importante ressaltar que o número de subproblemas aumenta significativamente ao acrescentarmos uma dimensão. Portanto, para que o número de iterações não seja "elevado", mostraremos exemplos simples resolvidos graficamente em 9 iterações.

5.2.1 Algoritmo Branch-and-Bound

Vimos no desenvolvimento do exemplo 1, que o algoritmo branch-and-bound constrói subproblemas de programaçao linear e que esses problemas são resolvidos pelo método simplex. Basicamente, o algoritmo se detém em construir os subproblemas e avaliar as soluções encontradas pelo simplex para determinar se essas soluções são inteiras.

Sejam $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$ e $b \in \mathbb{R}^m$. Considere o seguinte problema de Programação Linear Inteira:

$$\begin{aligned} \text{minimizar} \quad & c^T x \\ \text{sujeito a} \quad & Ax = b, \\ & x \geq 0, \\ & x \in \mathbb{Z}^n. \end{aligned} \tag{5.10}$$

Dados $bound = \infty$ e nível $= 0$, a linha de comando do Matlab é definida por:

```
[xopt, bound]=brach_bound(c, A, b, bound, nível)
```

em que os dados de saída `xopt` e `bound` representam o ponto e o valor ótimo do problema de PLI, respectivamente.

O algoritmo Branch-and-Bound proposto neste texto é baseado em Gonzaga (2005a) e utiliza o processo de recursividade, isto é, o Branch-and-Bound é uma rotina em si mesmo. Veja o algoritmo (5.1).

5.2.2 Visualização Gráfica do Método Branch-and-Bound tridimensional

Na seção anterior, abordamos um exemplo bidimensional com o auxílio gráfico. Aqui apresentaremos o desenvolvimento do método Branch-and-Bound graficamente, com a utilização do programa "`poliedro_bb`" que será apresentado na próxima seção.

Exemplo 2: Considere o seguinte problema de PLI:

$$\begin{aligned} \text{minimizar} \quad & -x_1 - 0,5x_2 - 0,5x_3 \\ \text{sujeito a} \quad & x_1 + x_2 + 0,9x_3 \leq 4,5 \\ & x_3 \leq 3,5 \\ & x \geq 0 \\ & x \in \mathbb{Z}^3. \end{aligned} \tag{5.11}$$

Algoritmo 5.1: Branch-and-Bound

Dados: $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, *bound* e o nível.

Resolução do problema relaxado utilizando o algoritmo Simplex:

$[\bar{x},\ \bar{c}] = \text{simplex}(c,A,b)$;

nível=nível+1.

Se $\bar{x} = \emptyset$ **então**
Problema inviável.

Fim-se

Se $\bar{c} \geq bound$ **então**
O custo é considerado caro e a solução é desconsiderada.

Fim-se

Escolha da variável j fracionária para definir os planos de corte:

$M = |\bar{x} - \text{round}(\bar{x})|$;

Se $M = 0$ **então**
$x_{opt} = \bar{x}$.

Se $\bar{c} < bound$ **então**
$bound = \bar{c}$.

Fim-se

Fim-se

Escolha o índice j tal que $M_j = \max(M_i)_{i=1,\ldots,n}$.

– Construção do problema à esquerda:

Acrescentar a restrição $x_j \leq \lfloor \bar{x}_j \rfloor$, definindo $\bar{A}$ e $\bar{b}$.

Utilizando o processo de recursividade, chamamos a rotina branch-and-bound com os parâmetros de saída x_{opt} e *bound*

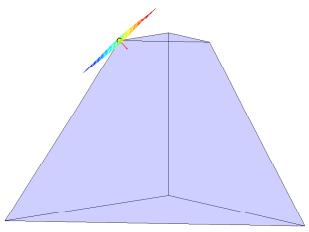
$[x_{opt}, bound] = \text{brach-and-bound}(c,\bar{A},\bar{b},bound,\text{nível})$;

– Construção problema à direita:

Acrescentar a restrição $x_j \geq \lfloor \bar{x}_j \rfloor + 1$, nos dados de entrada A e b, redefinimos $\bar{A}$ e $\bar{b}$.

$[x_{opt}, bound] = \text{brach-and-bound}(c,\bar{A},\bar{b},bound,\text{nível})$.

Primeiramente, desconsideramos a restrição inteira $x \in \mathbb{Z}^n$, definindo o problema PL relaxado do nível 0. Utilizando o programa "`poliedro_simplex`" obtemos a seguinte solução $x^0 = [1,35\ 0\ 3,5]^T$, com custo $c(x^0) = -6,6$. Analogamente, ao exemplo anterior, a escolha do índice é dada pela maior variável de M, tal que $M = |x^0 - \text{round}(\bar{x})|$, isto é, escolher j tal que



$$M_j = \max(M_i)_{i=1,\ldots,3}$$

Portanto, $j = 3$. Assim adicionamos o plano de corte $x_3 \leq 3$, definindo o problema p_1 à esquerda no nível 1. Solução $x^1 = [1{,}8\ 0\ 3]^T$, com custo $c(x^1) = -6{,}3$.

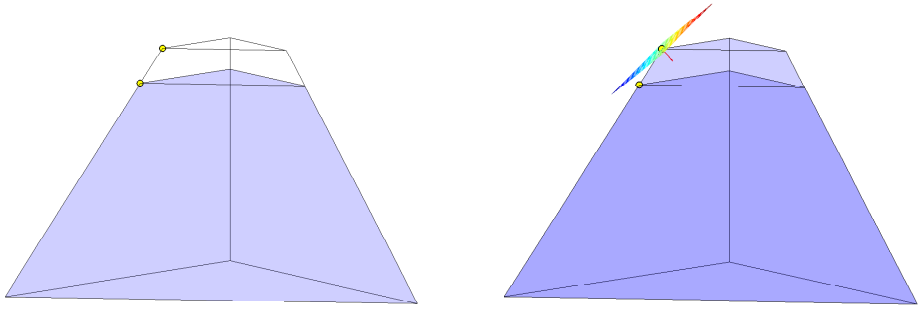


Figura 5.5: Problema à esquerda p_1 no nível 1

A variável x_1 é a única fracionária, assim definimos o problema p_2 à esquerda no nível 2, adicionando a restrição $x_1 \leq 1$. Solução $x^2 = [1\ 0\ 3]^T$, com custo $c(x^2) = -5{,}5$. O método determina a primeira solução viável inteira, isto é, definimos o "bound" por $c(x^2) = -5{,}5$.

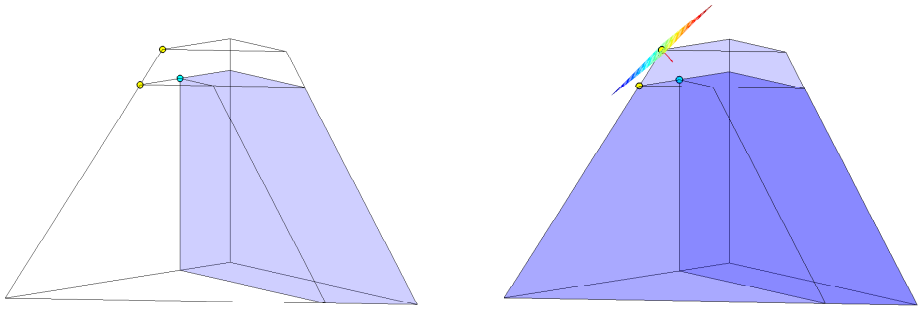


Figura 5.6: Problema à esquerda p_2 no nível 2

A solução x^2 é inteira, portanto definimos o problema à direita p_3 no nível 2, com o acréscimo da restrição $x_1 \geq 2$. Solução $x^3 = [2\ 0\ 2{,}78]^T$, com custo $c(x^3) = -6{,}167$.

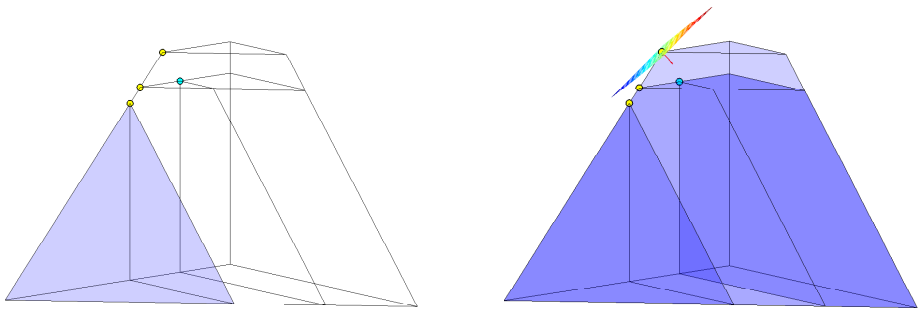


Figura 5.7: Problema à direita p_3 no nível 2

Definimos o problema à esquerda p_4 no nível 3, com o acréscimo do plano de corte $x_3 \leq 2$. Solução $x^4 = [2{,}7\ 0\ 2]^T$, com custo $c(x^4) = -5{,}7$. Veja Figura (5.8).

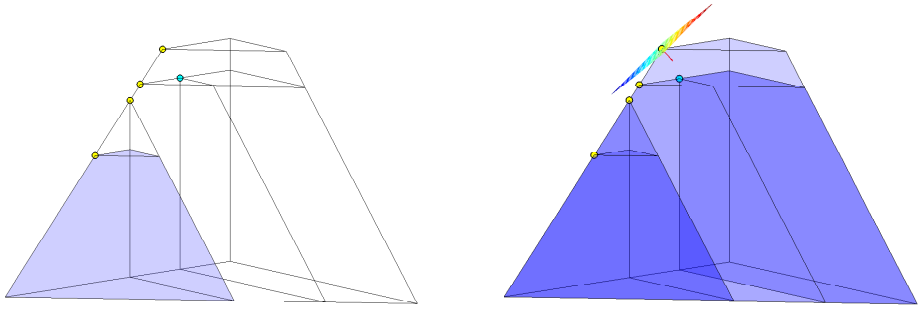


Figura 5.8: Problema à esquerda p_4 no nível 3

Analogamente, definimos o problema à esquerda p_5 no nível 4, adicionando a restrição $x_1 \leq 2$. Solução $x^5 = [2\ 0\ 2]^T$, com custo $c(x^5) = -5$. Solução inteira viável, mas com custo "caro".

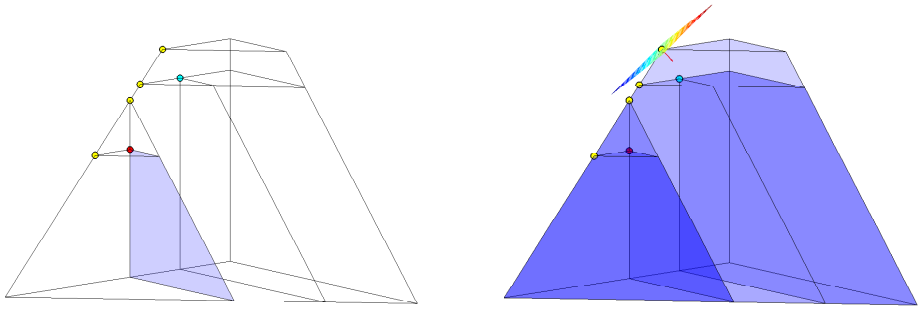


Figura 5.9: Problema à esquerda p_5 no nível 4, com custo caro.

Problema à direita p_6 no nível 4, com a restrição $x_1 \geq 3$. Solução $x^6 = [3\ 0\ 1,67]^T$, com custo $c(x^6) = -5,5$. Solução fracionária viável, mas com custo "caro", isto é, $c(x^2) \leq c(x^6)$ implica que o problema p_6 não precisa ser particionado (problema descartado).

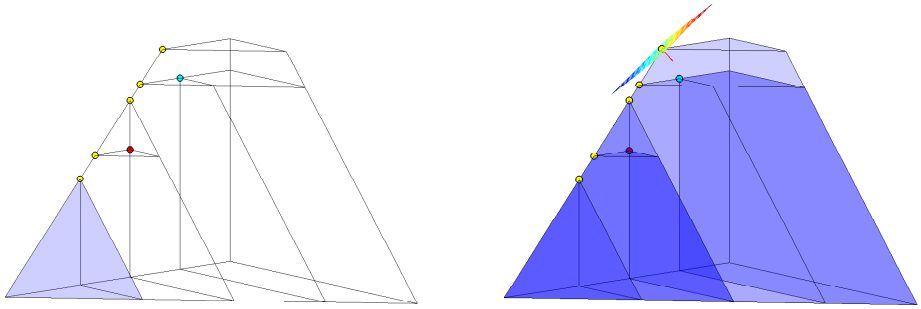


Figura 5.10: Problema à direita p_6 no nível 4, com custo caro.

Não é necessária a partição em subproblemas no nível 4, portanto, voltamos ao nível 3 e definimos o problema p_7. Solução inviável. Retornamos ao problema à direita do nível 1, definindo o problema relaxado p_8. Novamente, utilizando o Simplex, concluímos que a solução

é inviável. Assim, todos os subproblemas possíveis foram analisados com solução do problema de PLI dada por $x^* = [1\ 0\ 3]$, com custo $c(x^*) = -5,5$. Veja Figura (5.11).

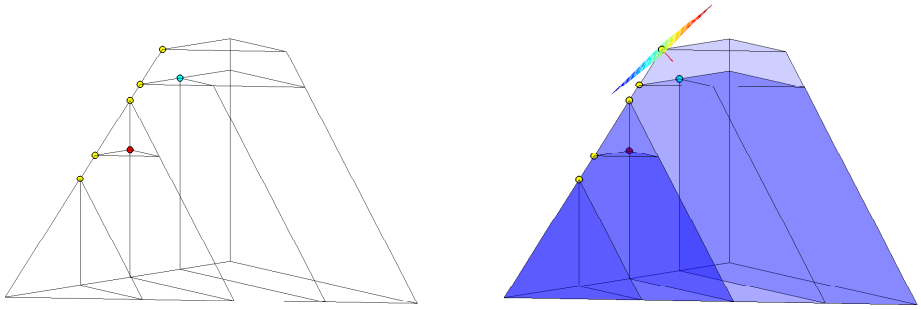


Figura 5.11: Solução do problema de PLI

A seguir apresentamos a árvore binária relacionada ao desenvolvimento do método Branch-and-Bound.

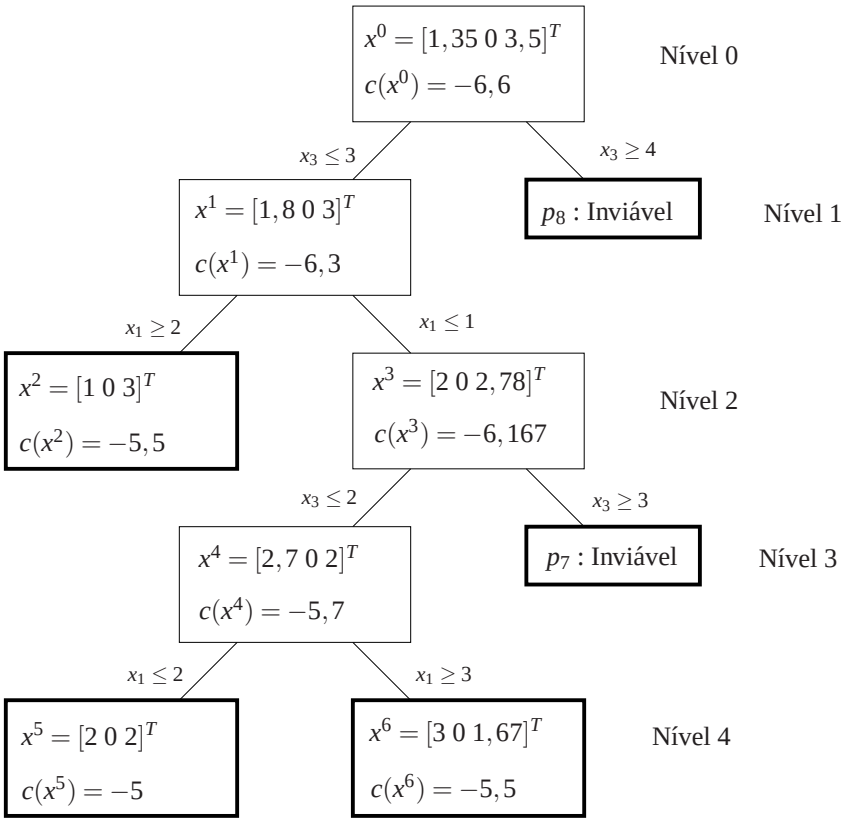


Figura 5.12: Árvore completa

5.3 O Programa `poliedro_bb`

Nesta seção apresentaremos o programa que fez possível visualizar o desenvolvimento do método branch-and-bound dos exemplos propostos nas seções anteriores. Sabemos que o programa "`poliedro`" constrói politopos, por essa razão, nos ateremos a exemplos com região viável neste formato.

Considere o problema de PLI definido em (5.10).

$$\begin{aligned} \text{minimizar} \quad & c^T x \\ \text{sujeito a} \quad & Ax = b, \\ & x \geq 0, \\ & x \in \mathbb{Z}^n. \end{aligned}$$

Dados

$$bound = \infty \quad \text{e} \quad \text{nível} = 0,$$

a linha de comando do programa `poliedro_bb` no Matlab é definida por:

```
poliedro_bb(c, A, b, bound, nível),
```

este programa repassa como dados de saída os gráficos e a solução, representada por:

- `xopt`: solução ótima do problema PLI;
- `bound`: custo relacionado ao `xopt`.

O programa `poliedro_bb` é o algoritmo Branch-and-Bound com as rotinas de representação gráfica, que utiliza o algoritmo Simplex e o programa "`poliedro`".

Primeiramente, utilizamos o Simplex para determinar a solução do problema relaxado. Após, o programa "`poliedro`" constrói a região viável, para determinar a solução do subproblema. A cada iteração o `poliedro_bb` constrói dois politopos: um representa o subproblema a ser resolvido separadamente e o outro é sobreposto aos demais politopos (caso não seja o subproblema do nível 0). Veja Figura (5.13)

No capítulo anterior, o programa `poliedro_simplex` trabalhava com os algoritmos "`poliedro`" e Simplex separadamente. Aqui, a recursividade do algoritmo branch-and-bound faz com que os poliedros sejam construídos a cada iteração, isto é, o programa "`poliedro`" é uma rotina do branch-and-bound (veja algoritmo (5.2)). Além disso, o ponto que caracte-

riza o bound, *xbound* (caso exista), é repassado para a iteração sucessiva auxiliando apenas na construção gráfica do problema.

A visualização gráfica do desenvolvimento do branch-and-bound ocorre após determinarmos o politopo correspondente ao subproblema. Mas, se o subproblema for inviável o programa `poliedro_bb` simplesmente o ignora.

No programa `poliedro_bb` as cores possuem fundamental importância, pois determinamos a solução do problema pela cor azul, além disso, a cor vermelha representa se o subproblema é considerado caro, e a cor amarela distingue as soluções fracionárias das inteiras. As cores dos pontos que são encontrados pelo método Branch-and-Bound representam a seguinte relação:

Cor	Representação
●	Ponto fracionário;
●	Ponto com custo “caro”;
●	Bound ou solução do problema.

Nas imagens a seguir, as mesmas ferramentas do Matlab que auxiliavam na visualização no `poliedro_simplex` são empregadas.

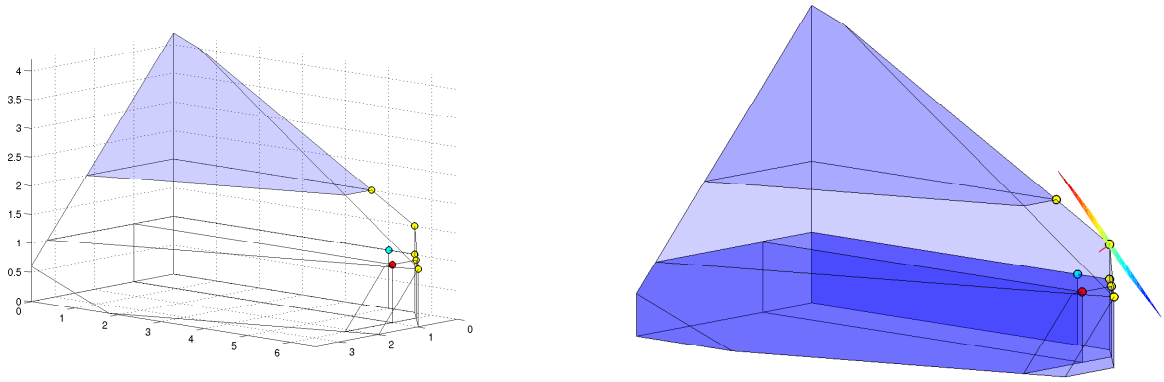


Figura 5.13: Subproblema e subproblemas sobrepostos construídos pelo `poliedro_bb`.

Algoritmo 5.2: Poliedro_bb

Dados: $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, *bound*, *xbound* e o nível.

Resolução do problema relaxado utilizando o algoritmo Simplex:

$[\bar{x},\ \bar{c}] = \text{simplex}(c, A, b)$;

Construção dos politopos:

Figura 1: poliedro(A, b) e plotar $\bar{x}$ em amarelo.

Se *nível*$= 0$ **então**

Figura 2: poliedro(A, b) e a construção da região de nível passando por $\bar{x}$. Plotar $\bar{x}$ em amarelo.

Senão

Figura 2: poliedro(A, b) e plotar $\bar{x}$ em amarelo.

Fim-se

nível=nível+1.

Se $\bar{x} = \emptyset$ **então**

Problema inviável.

Fim-se

Se $\bar{c} \geq$ *bound* **então**

O custo é considerado caro. Plotar $\bar{x}$ em vermelho.

Fim-se

Escolha da variável j fracionária para definir os planos de corte:

$M = |\bar{x} - \text{round}(\bar{x})|$;

Se $M = 0$ **então**

$x_{opt} = \bar{x}$. Plotar $\bar{x}$ em vermelho.

Se $\bar{c} <$ *bound* **então**

bound $= \bar{c}$, plotar $\bar{x}$ em azul e *xbound* em vermelho.

xbound $= \bar{x}$.

Fim-se

Fim-se

Escolha o índice j tal que $M_j = \max(M_i)_{i=1,\ldots,n}$.

– Construção do problema à esquerda:

Acrescentar a restrição $x_j \leq \lfloor \bar{x}_j \rfloor$, definindo $\bar{A}$ e $\bar{b}$.

Utilizando o processo de recursividade:

$[x_{opt}, bound, xbound] = \texttt{poliedro\_bb}(c, \bar{A}, \bar{b}, bound, xbound, \text{nível})$;

– Construção do problema à direita:

Acrescentar a restrição $x_j \geq \lfloor \bar{x}_j \rfloor + 1$, nos dados de entrada A e b, redefinimos $\bar{A}$ e $\bar{b}$.

$[x_{opt}, bound, xbound] = \texttt{poliedro\_bb}(c, \bar{A}, \bar{b}, bound, xbound, \text{nível})$.

6 *Considerações Finais*

No trabalho desenvolvemos e implementamos programas para visualizar geometricamente a evolução "passo a passo" dos algoritmos Simplex (Programação Linear) e Branch-and-Bound (Programação Linear Inteira). Esta proposta se caracterizou ao apresentarmos os algoritmos "*poliedro-simplex*" e "*poliedro-bb*", através de exemplos bidimensionais e tridimensionais. Outro fato importante foi a compreensão da teoria e dos algoritmos alcançada na elaboração dos programas.

A importância didática dos programas de visualização ficou clara na facilidade de apresentar geometricamente a resolução dos problemas de otimização propostos neste texto, através do tratamento dos poliedros e das cores nas imagens a cada iteração dos algoritmos Simplex e Branch-and-Bound.

Por fim, a troca de idéias e informações importantes durante a elaboração deste trabalho e no decorrer dos últimos três anos trabalhando com o professor Clóvis e seus orientandos possibilitou-me o amadurecimento matemático dos conceitos básicos nas áreas de Programação Linear, Não Linear e Inteira.

6.1 Trabalhos futuros

Desenvolver e implementar os programas propostos neste trabalho em softwares livres. Além disso, elaborar uma apostila (manual) para auxiliar no uso dos programas.

Desenvolver um algoritmo para visualizar geometricamente o desenvolvimento do algoritmo de regiões de confiança baseado em trajetória central e sua utilização em métodos de filtro apresentado na dissertação de mestrado de Ana Maria Basei em 2007. A teoria deste algoritmo foi desenvolvido pelo professor Clóvis Gonzaga em conjunto com Roger Behling, Ana Maria Basei, entre outros. Sua aplicação em problemas de Programação Não Linear (PNL) consiste num método intermediário, com regiões de confiança que evoluem continuamente de pequenas bolas e tendem a uma caixa completa. As regiões são definidas pela função barreira

logarítmica e a minimização da quadrática é feita através de uma busca de Armijo curvilínea sobre a trajetória central da caixa. Veja figure (6.1).

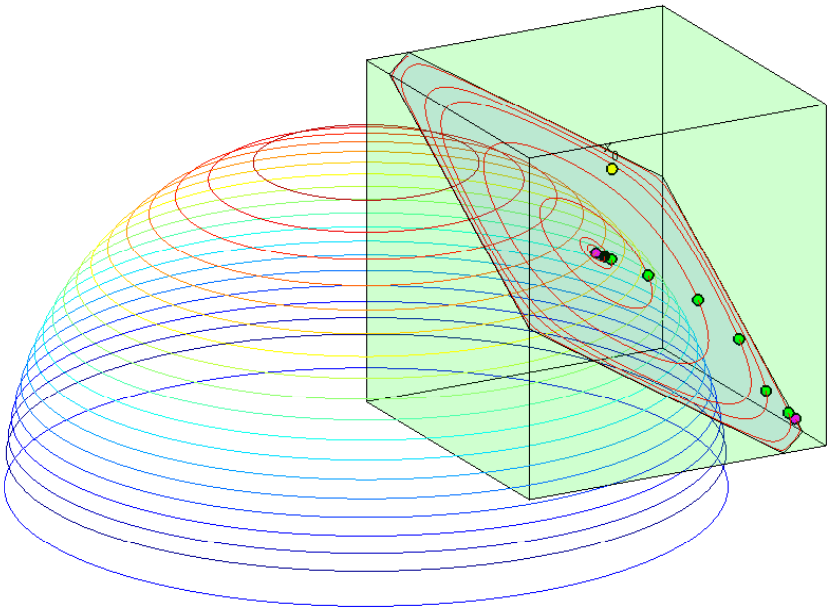


Figura 6.1: Trajetória Central é descrita no núcleo da matriz jacobiana (plano de corte) com restrição de caixa.

Referências Bibliográficas

BACHEM, A.; GRöTSCHEL, M.; KORTE, B. *Mathematical Programming, The State of the Art, Bonn 1982*. [S.l.]: Springer – Verlag, 1983.

BAZARAA, M. S.; SHERALI, H. D.; SHETTY, C. M. *Nonlinear Programming - Theory and Algorithms*. Second. [S.l.]: John Wiley & Sons, 1993.

BENNATON, J. F. *Fermat e o início da História dos problemas de otimização*. Abril 2001. Disponível em: <http://www.lps.usp.br/neo/jocelyn/historia_jocelyn.htm>.

BERTSEKAS, D. P. *Nonlinear Programming*. Belmont, Massachusetts: Athena Scientific, 1995.

CHVáTAL, V. *Linear Programming*. [S.l.]: W. H. Freeman and Company - New York, 1983.

COLLI, E. *Poliedros*. 2003. Disponível em: <http://matemateca.incubadora.fapesp.br/portal-/textos/matemateca/poliedros/Poliedros.pdf>.

DANTZIG, G. B. History of mathematical programming - a collection of personal reminiscences. In: ____. [S.l.]: North-Holland, 1991. cap. Linear Programming, p. 19–31.

DANTZIG, G. B.; THAPA, M. N. *Linear programming 1: introduction*. [S.l.]: Springer Series in Operations Research, 1997.

EVES, H. *Introdução à história da matemática*. [S.l.]: Editora da UNICAMP, 2004.

GOMORY, R. E. History of mathematical programming - a collection of personal reminiscences. In: ____. [S.l.]: North-Holland, 1991. cap. Early Integer Programming, p. 55–61.

GONDZIO, J. *Large Scale Optimization with Interior-Point Methods*. Agosto 2007. Disponível em: <http://www.maths.ed.ac.uk/~gondzio/>.

GONZAGA, C. C. Notas de aula de pesquisa operacional - não publicadas. Universidade Federal de Santa Catarina. 2005a.

GONZAGA, C. C. Notas de aula de progamação linear - não publicadas. Universidade Federal de Santa Catarina. 2005b.

GONZAGA, C. C. Notas de aula de progamação não linear - não publicadas. Universidade Federal de Santa Catarina. 2006.

IZMAILOV, A.; SOLODOV, M. *Otimização - volume 1 - Condições de Otimalidade, Elementos de Análise Convexa e de Dualidade*. [S.l.]: impa, 2005.

IZMALOV, A.; SOLODOV, M. *Otimalidade - volume 2 - Métodos Computacionais*. [S.l.]: impa, 2007.

NOCEDAL, J.; WRIGHT, S. J. *Numerical Optimization*. [S.l.]: Springer-Verlag, 1999. (Springer Series in Operations Research).

PAPADIMITRIOU, C. H.; STEIGLITZ, K. *Combinatorial Optimization - Algorithms and Complexity*. [S.l.]: Dover Publications, INC., 1998.

SCHRIJVER, A. Theory of linear and integer programming. In: ____. New York: John Wiley & Sons, 1986. cap. 15.1 : Karmarkar's polynomial–time algorithm for linear programming, p. 190–194.

VANDERBEI, R. J. *Linear Programming - Foundations and Extensions*. [S.l.]: Kluwer Academic Publishers - Boston/London/Dordrecht, 1996.

APÊNDICE A – POLIEDRO_V1: Guia do Usuário

A.1 Introdução

Este guia foi elaborado a pedido de meu orientador Clóvis Caesar Gonzaga e pelos professores da banca de mestrado Elizabeth Wegner Karas, Juliano de Bem Francisco e Danilo Royer.

O objetivo deste guia é incentivar professores e alunos a utilizarem o programa POLIEDRO_V1 em cursos básicos de Otimização Matemática. Além disso, o guia mostra como utilizar o programa POLIEDRO_V1 através de comandos básicos a serem processados pelo programa Matlab.

O programa POLIEDRO_V1 foi criado durante os cursos de Programação Linear e Pesquisa Operacional ministradas pelo professor Clóvis na Universidade Federal de Santa Catarina no segundo semestre de 2005. Este programa constrói poliedros limitados (politopos) que servem de auxílio na compreensão geométrica de algoritmos em problemas de otimização que utilizam como região viável poliedros. Inicialmente, o programa POLIEDRO_V1 apresenta a visualização do desenvolvimento dos algoritmos simplex e branch-and-bound, mas o programa pode ser reestruturado para problemas de Programação Não Linear com restrições de caixa.

A visualização geométrica dos algoritmos implica que os problemas abordados pelo programa POLIEDRO_V1 possuem duas ou três dimensões, em que o formato e a dimensão dos problemas são definidos pelas restrições (poliedro). Além disso, algumas definições e nomeclaturas são vinculadas aos conceitos teóricos de Otimização abordados na Dissertação de Mestrado em Matemática da Universidade Federal de Santa Catarina, *Visualização de Poliedros em Algoritmos de Programação Linear e Inteira*, apresentada em maio de 2008. Por essa razão, esta dissertação serve como auxílio na compreensão deste *Guia do Usuário*.

A.2 Requisitos do Programa

- O POLIEDRO_V1 foi implementado em Matlab, assim é necessário o software Matlab na versão 6 ou superior. Além disso o usuário deve ter um conhecimento básico da linguagem Matlab. Caso o usuário não esteja familiarizado com o Matlab recomendamos o uso do `help` do Matlab.

- A visualização de poliedros requer um computador com processador gráfico de boa qualidade.

- Algumas sub-rotinas do programa POLIEDRO_V1 utilizam os três botões do *mouse*, assim para uma melhor utilização do programa recomendamos o uso do *mouse* de três botões.

A.3 Instalação

A versão do POLIEDRO_V1 é fornecida em forma compacta (.zip ou .tar.gz) e precisa ser desconpactada antes de sua utilização. Para baixar o programa POLIEDRO_V1, basta acessar os sites:

`www.mtm.ufsc.br/~clovis/POLIEDRO_V1.zip`
`www.mtm.ufsc.br/~gilberto/POLIEDRO_V1.zip`

A.4 Conteúdo do Programa POLIEDRO_V1

Após a instalação, do diretório POLIEDRO_V1 terá os seguintes arquivos e subdiretórios contendo os diversos componentes do programa POLIEDRO_V1:

\BRANCH_BOUND	Diretório de programação inteira.
\DATA	Dados de problemas.
\DATA\R2	Dados de problemas bidimensionais.
\DATA\R3	Dados de problemas tridimensionais.
\POLIEDRO	Diretório com os programas de construção de poliedros.
\SIMPLEX	Diretório de programação linear.
startup.m	Arquivo caminho (orienta o Matlab na busca de diretórios e arquivos).

A.5 Aprendendo a Usar o POLIEDRO_V1

Agora que você instalou o programa POLIEDRO_V1, vamos ver como utilizamos esta ferramenta didática na construção de poliedros e na execução dos algoritmos "Simplex"e "Branch-and-Bound".

A.5.1 Inicialização

O programa Matlab deve ser executado no diretório POLIEDRO_V1, onde se encontra o arquivo `startup.m` que indicara ao Matlab em que subdiretórios estão os arquivos necessários para executar o programa POLIEDRO_V1.

Importante: Dependendo da versão do Matlab, o arquivo `startup.m` não é executado automaticamente. Portanto, antes de utilizar o programa, é recomendada a execução deste arquivo pelo comando:

```
>> startup
```

A.5.2 Poliedros

O programa POLIEDRO_V1 constrói apenas poliedros limitados (politopos) em formatos canônico ou dual[1], com dimesão menor ou igual a 3.

A construção do poliedro tem como base as definições de poliedros encontradas na disseração de mestrado, *Visualização de Poliedros em Algoritmos de Programação Linear e Inteira – 2008*, em que um poliedro pode ser representado como uma intersecção finita de semi-espaços fechados. A representação desse poliedro P é descrita por

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

Para construirmos a matriz A, devemos primeiramente definir os semi-espaços que definem o poliedro. As linhas da matriz A são formadas pelas normais desses semi-espaços.

Outro fator importante é a utilização do `help` do Matlab para visualizar os dados dos pro-

[1]O formato dual é constituído de um sistema de desigualdades, em que cada desigualdade representa um semi-espaço fechado, $Ax \leq b$. Por outro lado, o formato canônico é um caso particular do formato dual, em que as restrições de possitividades $x \geq 0$ são separadas das demais, isto é:

$$\begin{aligned} Ax &\leq b \\ x &\geq 0 \end{aligned}$$

gramas a serem executados. Por exemplo, para visualizar os dados do programa `poliedro`, basta executar o comando `help poliedro`:

```
>>help poliedro
  function [ ]=poliedro(A,b,formato,cor)
  saida := Desenha o poliedro do problema em R2 ou R3;
  entrada := A matriz das restricoes;
             b termos independentes;
             formato:  1 => canonico; Ax<=b e x>=0; Default;
                       2 => dual; Ax<=b;
             cor:  Default 'blue'; string.
  Programa auxiliar:  ordem.m para ordenar os vertices.
```

Construção de Poliedros

O programa `POLIEDRO_V1` possui um banco de dados no diretório `\DATA`[2]. Nos subdiretórios `\DATA\R2` e `\DATA\R3`, encontramos dados de entrada que geram poliedros bidimensionais e tridimensionais respectivamente.

Os dados são apresentados pela matriz A e o vetor b. Por exemplo no subdiretório `\DATA\R2` encontramos o arquivo `pentagono.m`, correspondente a um polígono bidimensional.

Após inicializado o Matlab no diretório `POLIEDRO_V1` basta digitar o comando `pentagono` que você gera a matriz A e o vetor b.

```
>> pentagono
>> A
A =
         0   -1.0000
    4.4721   -1.4531
    2.7639    3.8042
   -2.7639    3.8042
   -4.4721   -1.4531

>> b
b =
```

[2]Quando a dimensão do problema é maior que três, o programa ignora a construção do poliedro. Neste caso o problema é resolvido com o método simplex ou pelo método branch-and-bound, sem a visualização geométrica.

−0.7639
27.2931
41.4895
19.3781
−8.4840

Observe que a matriz *A* possui duas colunas, o que caracteriza um problema bidimensional. Da mesma forma, podemos gerar poliedros tridimensionais encontrados no subdiretório \DATA\R3. Por exemplo, ao digitar o comando cubo, temos:

```
>> cubo
>> A
A =
    1   0   0
    0   1   0
    0   0   1

>> b
b =
    2
    2
    2
```

Visualização de Poliedros

A visualização de poliedros no programa POLIEDRO_V1 é executada pelo arquivo poliedro.m. Vimos anteriormente pelo comando help os dados do programa poliedro, agora analizaremos cada dado separadamente:

a) A matriz das restrições *A* e o vetor dos termos independentes *b*:

- A matriz *A* deve ter duas ou três colunas, determinando a dimensão do poliedro.
- O número de linhas da matriz *A* deve ter a dimensão do vetor *b*. Assim, independentemente do formato do poliedro, canônico ou dual, A e b estão bem definidos.
- O sistema de desigualdades (restrições) devem formar um politopo[3].

b) O formato do poliedro pode ser canônico ou dual, e é representado pelos números 1 e 2,

[3]Caso as restrições não definam um politopo, veja na subseção **Poliedros ilimitados** da seção (3.5.3) no capítulo 3 da dissertação.

respectivamente. Sabemos que o formato canônico considera a restrição de positividade $x \geq 0$, já o formato dual não necessariamente possui essa restrição.

c) A cor do poliedro[4].

As principais cores são: amarelo, representado pela letra y, magenta por m, ciano c, vermelho r, verde g, azul b, branco w e preto k.

Considere o poliedro $P = \{x \in \mathbb{R}^3 \mid Ax \leq b, x \geq 0\}$. Após ter gerado os dados iniciais (A e b), a linha de comando do Matlab, para construir o poliedro P no formato canônico, é definida por:

```
>>poliedro(A,b,1,'b');
```

Esse comando reproduz o poliedro P, azul no $\mathbb{R}^3$. O programa trabalha com o formato canônico (1), e a cor azul (b) como *Default*. Então para construirmos o mesmo poliedro P, basta o comando

```
>>poliedro(A,b).
```

Por outro lado, se quisermos construir um poliedro $Q = \{x \in \mathbb{R}^2 \mid Dx \leq d\}$ na cor vermelha, o comando é:

```
>>poliedro(D,d,2,'r').
```

Exemplos de poliedros bi e tridimensionais

Poliedro Bidimensional: Após ter inicializado o programa com o comando `startup`, construímos o poliedro `pentagono` pelos comandos:

```
>> pentagono
>> poliedro(A,b)
```

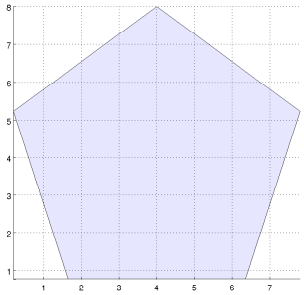


Figura A.1: Pentágono

[4]Existem diversas formas de definir cores no Matlab, encontradas em qualquer manual do Matlab.

Poliedro Tridimensional: Construímos o poliedro `cubo` pelos comandos:

```
>> cubo
>> poliedro(A,b)
```

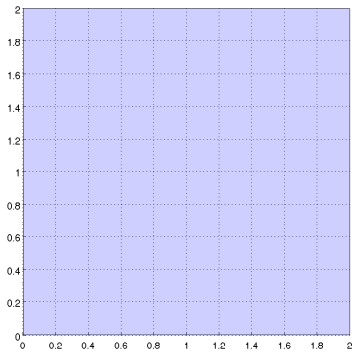


Figura A.2: Cubo

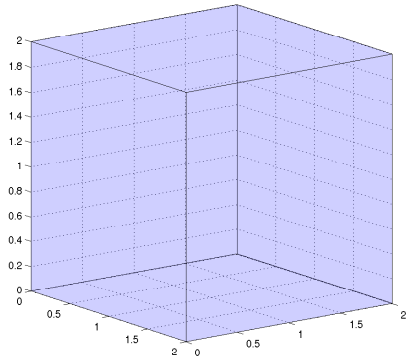


Figura A.3: Cubo, após rotação no ambiente Matlab.

Rotação no ambiente Matlab

Há duas formas de rotacionar figuras no ambiente Matlab. Primeiramente, selecione o link e em seguida, utilize o mouse ou o teclado da seguinte forma:

1. Rotação pelo mouse: Selecione a figura com o botão esquerdo e rotacione com o mouse;

2. Rotação pelo teclado:

Tecla	Ação
$\uparrow$	rotaciona a figura para cima;
$\downarrow$	rotaciona a figura para baixo;
$\leftarrow$	rotação horária;
$\rightarrow$	rotação antihorária.

Construindo seu próprio poliedro

Desejamos visualizar a região viável do seguinte problema:

$$\begin{aligned} \text{minimizar} \quad & f(x,y) \\ \text{sujeito a} \quad & x+y \leq 1, \\ & x,y \geq 0. \end{aligned}$$

Os semi-espaços do $\mathbb{R}^2$ que definem o poliedro são representados pelas seguintes desigual-

dades:

$$\begin{cases} x+y \leq 1 \\ -x \leq 0 \\ -y \leq 0 \end{cases}$$

A matriz A e o vetro b deste sistema são dados por:

$$A = \begin{bmatrix} 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

Essa matriz A e o vetor b podem ser salvos em um arquivo `.m` no subdiretório `\DATA\R2`, visto que o poliedro gerado é bidimensional.

Após criar a matrix A e o vetor b, no Matlab, basta executar o comando:

$>>$ `poliedro(A,b)`

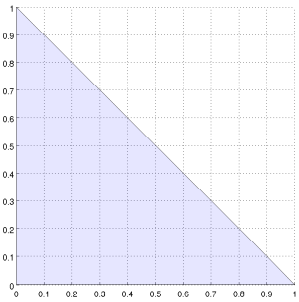


Figura A.4: Poliedro definido por A e b.

Note que definimos A e b no formato dual com a restrição de positividade. Para definirmos o mesmo poliedro no formato canônico bastava definir a matris A e o vetor b da seguinte forma:

$$A = \begin{bmatrix} 1 & 1 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 1 \end{bmatrix}.$$

A.5.3 Sub-rotina `click1.m`

A sub-rotina `click1.m` é usada pelos programas `poliedro_simplex` e `poliedro_bb`, que serão abordados nas seções (A.5.4) e (A.5.5). Ela controla rotações e transparência nas figuras. A seguir, as tabelas mostram como é executado o programa `click1.m`:

1. **Rotação:** A rotação da figura é executada pelo teclado da seguinte forma:

Tecla	Tecla	Ação
↑	w	rotaciona a figura para cima;
↓	s	rotaciona a figura para baixo;
←	a	rotaciona a figura à esquerda;
→	d	rotaciona a figura à direita.

2. **Transparência e execução do passo:** A transparência e a execução do próximo passo da figura são executados pelo teclado ou pelo `mouse` da seguinte forma:

Tecla	Botão	Ação
−	da direita	diminui a transparência da figura;
=	da esquerda	aumenta a transparência da figura;
Enter	central	executa o próximo passo.

A.5.4 Visualização de Poliedros no Método SIMPLEX

A visualização de poliedros no desenvolvimento do método Simplex utilizando o programa `POLIEDRO_V1` é executada pelo programa `poliedro_simplex`. Através do comando `help`, temos que:

```
>> help poliedro_simplex
  fuction [ ]=poliedro_simplex(c,A,b,Ae,be)
  Programa de visualizacao do metodo simplex em R2 ou R3
  Programas auxiliares:  poliedro.m
                         simplex_grafico.m
                         click1.m
                         corte.m
                         join2vetorpatch.m
                         join3vetorpatch.m
                         imprimeponto.m
  Problema:  min c'x
           s.a.  Ax<=b formato canonico
                Aex==be
                  x>=0
  Dados de entrada:
  c:= vetor coluna custo
  A:= Matriz das restricoes (de desigualdade)
```

```
b:= Termos independentes de A
Ae:= Matriz das restricoes (de igualdade) somente em R3
be:= termos independentes de Ae

O programa corte.m eh utilizado na construcao das restricoes
de igualdade.
```

Elaboração do programa `poliedro_simplex.m`

O objetivo desta subseção é apresentar a maneira como o programam `poliedro_simplex` foi elaborado a partir de um problema de Programação Linear dado e na composição dos programas `poliedro` e `simplex`.

Considere um problema de programação linear, constituído de um politopo como região viável, caracterizado por A, b e c.

$$\begin{aligned} \text{minimizar} \quad & c^T x \\ \text{sujeito a} \quad & Ax \leq b, \\ & x \geq 0. \end{aligned}$$

O programa `poliedro_simplex` é composto basicamente pelos programas `poliedro` e `simplex`, que efetua dois passos principais:

Primeiramente, o programa descobre todos os vértices que o algoritmo simplex percorre para definir a solução do problema. Para isso, constrói uma matriz V nos dados de saída do programa simplex, correspondente a estes vértices. Note que os dados correspondentes ao problema estão no formato dual, assim antes de introduzirmos os dados de entrada no simplex, o próprio programa altera o formato do problema (fomato primal).

O segundo passo é utilizar o programa poliedro para construir a região viável.

Note que no programa `poliedro_simplex`, o primeiro passo é feito separadamente do segundo. Assim, os vértices são simplesmente plotados numa ordem determinada pelo simplex, juntamente com os hiperplanos e a normal. Para plotar os hiperplanos e seu vetor normal fixamos uma proporção com relação ao tamanho do poliedro. Aqui, não entraremos em detalhes de como foram construídos os hiperplanos nem o vetor normal. Foram simplesmente utilizadas técnicas de programação em Matlab e geometria analítica.

Exemplo bi e tridimensional utilizando o `poliedro_simplex`

Visualização bidimensional do método Simplex

Considere o problema de Programação linear definido por:

$$\begin{array}{rl} \text{minimizar} & 2x+y \\ \text{sujeito a} & -x+2y \leq 4 \\ & 3x+5y \leq 15 \\ & 2x-3y \leq 6 \\ & -x-y \leq -1 \\ & x,y \geq 0. \end{array} \tag{A.1}$$

Os dados A, b e c que definem o problema (A.1), estão definidos no arquivo `barco.m` do banco de dados `\DATA\R2`. Assim para resolver este problema, basta executar o programa `POLIEDRO_V1` através dos comandos:

```
>> barco
>> poliedro_simplex(c,A,b)
```

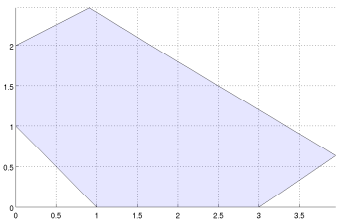


Figura A.5: Região viável gerada pelo programa `POLIEDRO_V1`.

Após o programa geral a figura (A.5), o usuário pode interagir com a figura de duas formas. Primeiramente, coloque o cursor na figura (A.5) gerada pelo Matlab e, em seguida, tecla `Enter` ou utilize o botão central do mouse para executar os próximos passos.

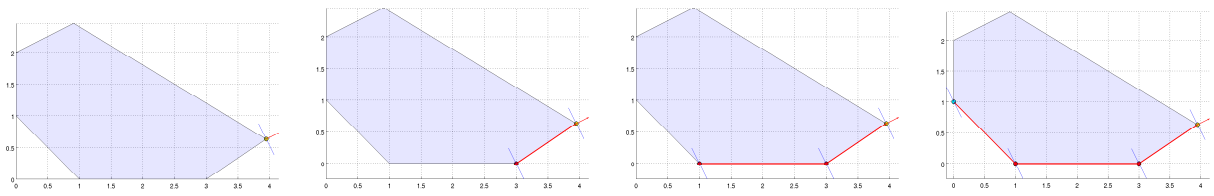


Figura A.6: Representação passo a passo do algoritmo simplex num problema bidimensional.

A sequência de figuras da esquerda à direita, obsevamos a evolução do método simplex do problema (A.1). A peimeira figura, apresenta o ponto inicial da segunda fase do método simplex, caracterizado pelo ponto amarelo. A segunda e a terceira figura correspondem aos

pontos intermediários, caracterizados pelos pontos em vermelho. Já a última figura representa a solução do problema caracterizada pelo ponto azul. Note que o caminho percorrido pelo método Simplex é destacado pelas arestas em vermelho.

Cor	Representação
●	Ponto inicial da segunda fase;
●	Pontos intermediários;
●	Ponto ótimo do problema.

Visualização tridimensional do método Simplex

Considere o arquivo `diamante.m`. Este arquivo é um problema linear do banco de dados `\DATA\R3` e gera a matriz A e os vetores b e c, em que c é o custo. Utilizando o comando `help`, temos:

```
>> help diamante
  diamante.m
  Exemplo de problema para o poliedro_simplex e poliedro_bb(branch
  and bound) com dimensao 3
  min c'x subject to Ax<=b, x>=0, em que
  A:= matriz das restricoes (desigualdades);
  b:= termos independentes (desigualdades);
  c:= custo relacionado;
```

Após a inicialização do programa `POLIEDRO_V1`, execute os comandos:

```
>> diamante
>> poliedro_simplex(c,A,b)
```

O comando `poliedro_simplex` gera a figura composta pela região viável (poliedro).

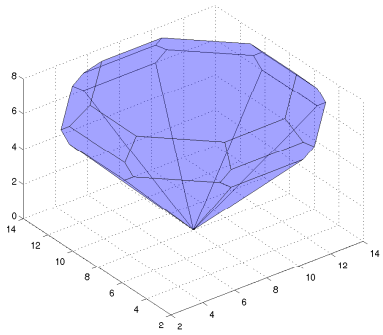


Figura A.7: Diamante

Em seguida, coloque o cursor sobre a Figura gerada pelo Matlab, e utilize o botão central do mouse ou a tecla `Enter` para gerar o ponto inicial da segunda fase do algoritmo Simplex. Veja a Figura (A.8).

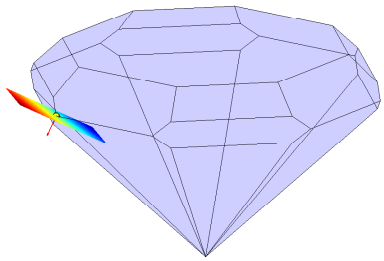


Figura A.8: Diamante com o ponto inicial da fase 2, após rotação utilizando a subrotina `click1`.

Para rotacionar a Figura (A.8), coloque o cursor sobre a Figura gerada pelo Matlab, e a rotacione pelas teclas `w`, `s`, `a` ou `d`. Veja tabela na seção (A.5.3). Para efetuar o próximo passo, basta clicar, novamente, com o mouse no botão central ou a tecla `Enter` na figura gerada pelo Matlab e obtemos a Figura (A.9).

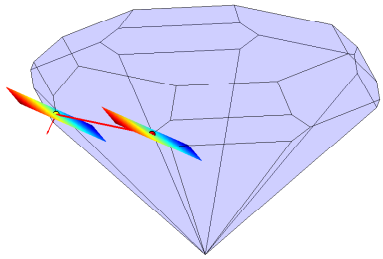


Figura A.9: Próximo passo da iteração no Simplex.

A aresta em vermelho representa o caminho percorrido pelo método de um vértice a outro. Já as cores dos pontos representam o mesmo que no caso bidimensional, isto é:

Cor	Representação
●	Ponto inicial da segunda fase;
●	Pontos intermediários;
●	Ponto ótimo do problema.

Vimos anteriormente que a transparência da figura pode ser alterada da seguinte forma:

Tecla	Mouse	Ação
–	Botão da direita	diminui a transparência da figura;
=	Botão da esquerda	aumenta a transparência da figura;

Quando a solução é encontrada não é mais possível executar a sub-rotina `click1.m`, a figura é "fixada" pelo Matlab e as ferramentas de visualização gráficas do Matlab são liberadas, isto é, a rotação é executada como se tivessemos no programa `poliedro` da seção (A.5.2). Veja Figura (A.10)

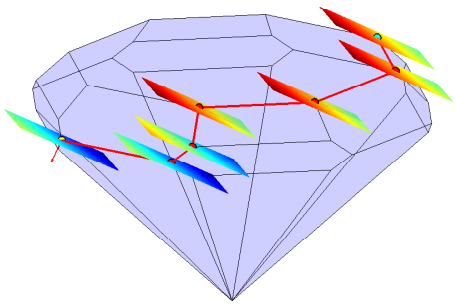


Figura A.10: Solução encontrada no ponto azul.

Obsevação: Se executarmos o comando `click1` no Matlab, acionamos novamente a sub-rotina `click1` na Figura (A.10).

A.5.5 Visualização de Poliedros no Método BRANCH AND BOUND

Nesta seção apresentaremos o programa `poliedro_bb` que faz possível visualizar o desenvolvimento do método branch-and-bound dos exemplos propostos em `\DATA\R2` e `\DATA\R3`. Sabemos que o programa "*poliedro*" constrói politopos, por essa razão, nos ateremos a exemplos com região viável neste formato. Considere o problema de PLI definido por:

$$\begin{aligned} \text{minimizar} \quad & c^T x \\ \text{sujeito a} \quad & Ax \leq b, \\ & x \geq 0, \\ & x \in \mathbb{Z}^n. \end{aligned} \tag{A.2}$$

Análogamente ao programa `poliedro_simplex`, o programa `poliedro_bb` necessita analisar os dados de entrada. Através do comando `help` do Matlab, temos:

```
>> help poliedro_bb
  function []=poliedro_bb(c,A,b,bound,nivel)
   metodo branch_and_bound para problemas lineares e inteiros, com cons-
   trucao de graficos em r2 e r3:
```

minimizar $c^T x$
sujeito a $Ax \leq b$,
$x \geq 0$,
$x \in \mathbb{Z}^n$.

```
   O formato dual possui todas as restricoes como desigualdades !!!
   isto e, o formato dual nao deve possuir resticoes de igualdade.
   Os programas auxiliares:  bb_graf.m e click1.m
   dados de entrada em data:
   p.ex:  no r2 vanderbei.m e no r3 diamante.m
   dados:  c, A, b, bound, nivel.
   bound:= inf, Default
   nivel:= 0, Default
   O programa polidero_bb.m nao possui opcao de cores.  A alteracao de
   cores deve ser feita diretamente no programa poliedro.m
   Dados de saida:  Graficos;
```

Observe que os dados de entrada `bound` e `nivel`, não são necessários na implementação, pois o programa `poliedro_bb` define estes dados por `inf` e `zero`, respectivamente como "Default".

Elaboração do programa `poliedro_bb.m`

O programa `poliedro_bb` é o algoritmo Branch-and-Bound com as rotinas de representação gráfica, que utiliza o algoritmo Simplex e o programa "`poliedro`".

Primeiramente, o programa utiliza o Simplex para determinar a solução do problema relaxado. Após, o programa "*poliedro*" constrói a região viável, para determinar a solução do subproblema. A cada iteração o `poliedro_bb` constrói dois politopos: um representa o subproblema a ser resolvido separadamente e o outro é sobreposto aos demais politopos (caso não seja o subproblema do nível 0).

Para uma melhor compreenção do método Branch-and-Bound, indicamos o capítulo (5) da dissertação. Essa passagem é representada no exemplo a seguir.

Visualização bidimensional do método Branch-and-Bound

Considere o problema `vanderbei2` definido no diretório `\DATA\R2`. Este problema gera a matriz *A*, e os vetores *b* e *c* do problema (A.2) de Programação Interira (PPI). Após a inicialização do programa `POLIEDRO_V1` execute os comandos:

`>> vanderbei2`

`>> poliedro_bb(c,A,b)`

Este programa gera duas figuras que devem ser posicionadas separadamente para uma melhor visualização.

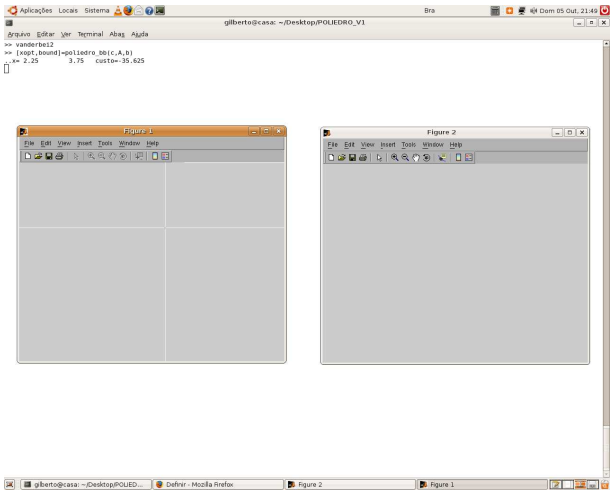


Figura A.11: Figura 1 e Figura 2 geradas pelo programa `POLIEDRO_V1`.

Para executar o próximo passo, basta colocar o cursor sobre a Figura 1, gerada pelo Matlab e clicar com o botão central do mouse ou usar a tecla `Enter`.

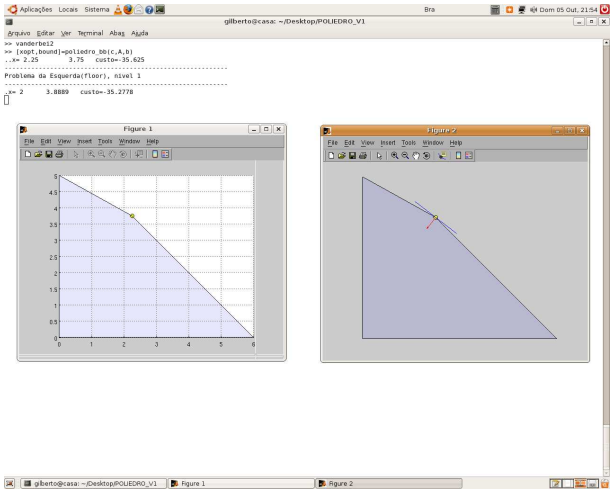


Figura A.12: Ponto ótimo determinado no nível 0.

A Figura 1 representa o subproblema que está sendo resolvido no nível, já a Figura 2 representa os subproblemas sobrepostos. A seguir o próximo passo do algoritmo Branch-and-Bound.

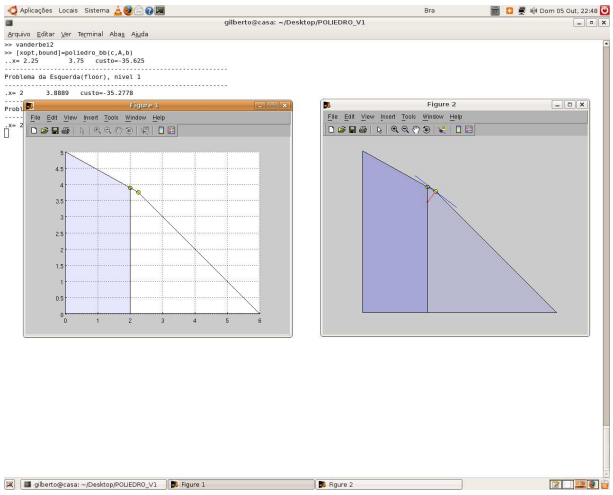
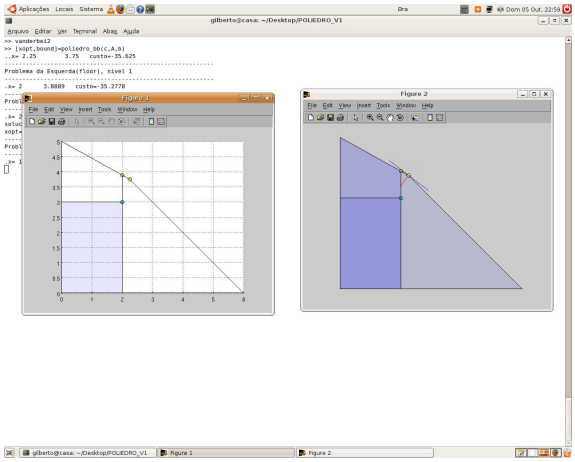


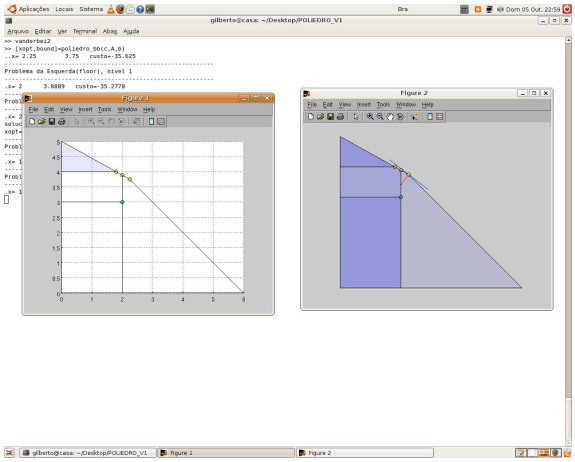
Figura A.13: Na Figura 1, o subproblema à esquerda. Na Figura 2, subproblemas sobrepostos.

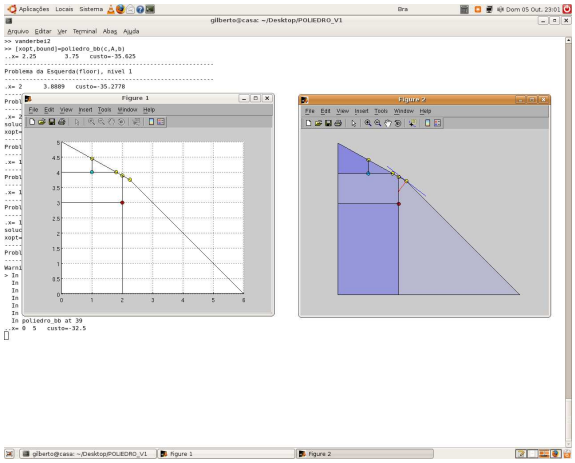
Note que a cada iteração o programa `poliedro_bb` repassa os dados de saída representados por:

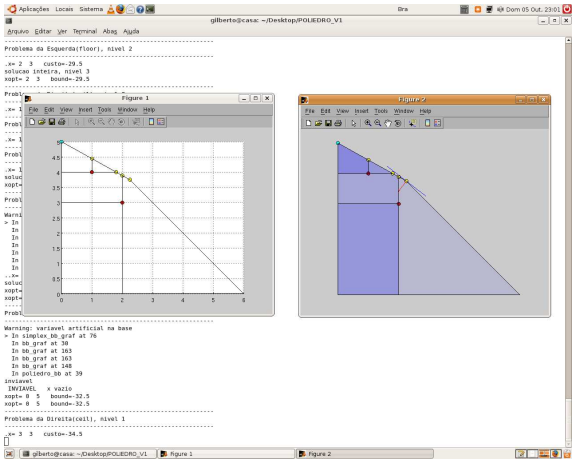
- x_{opt}: solução ótima do subproblema de Programação Linear;
- *bound*: custo relacionado ao x_{opt};

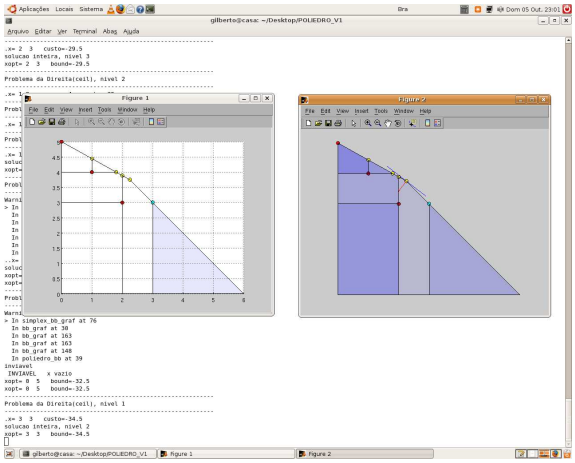
Seguindo o mesmo procedimento determinamos todos os pontos dos subproblemas até encontrar um ponto ótimo do problema de Programação Inteira. Veja na página a seguir.

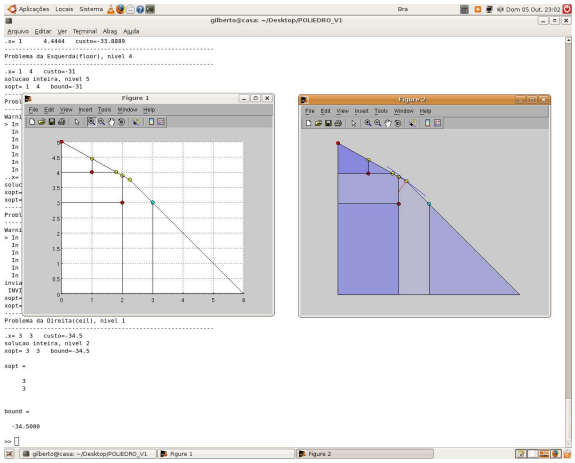












No programa `poliedro_bb` as cores possuem fundamental importância, pois determinamos a solução do problema pela cor azul, além disso, a cor vermelha representa se o subproblema é considerado caro, e a cor amarela distingue as soluções fracionárias das inteiras. Aqui, as mesmas ferramentas do Matlab que auxiliavam na visualização no `poliedro_simplex` são empregadas.

As cores dos pontos que são encontrados pelo método Branch-and-Bound representam a seguinte relação: (Veja a Figura (A.14))

Cor	Representação
●	Ponto fracionário;
●	Ponto com custo “caro”;
●	Bound ou solução do problema.

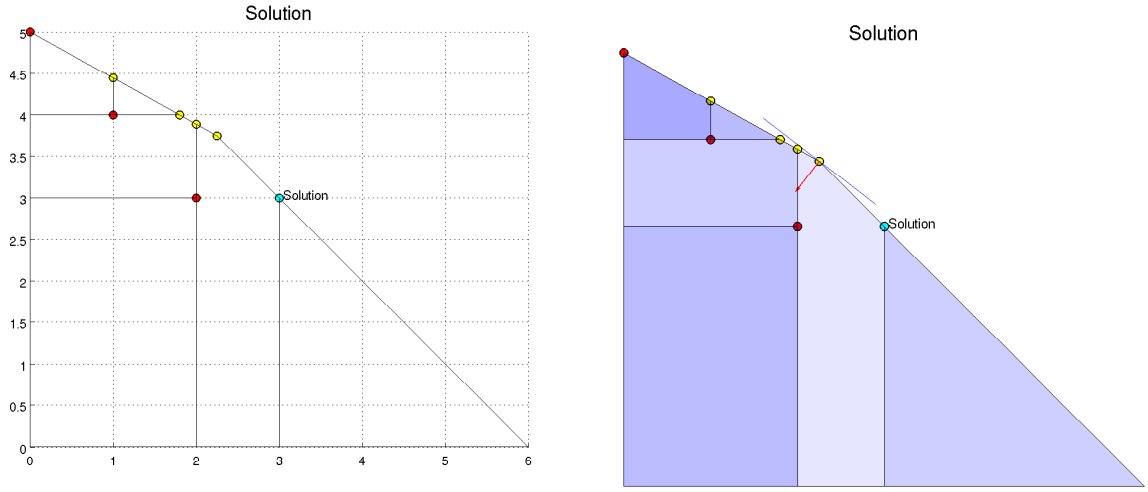


Figura A.14: Solução do problema de Programação Inteira

Algumas considerações:

Na seção anterior, o programa `poliedro_simplex` trabalhava com os algoritmos “*poliedro*” e Simplex separadamente. Aqui, a recursividade do algoritmo branch-and-bound faz com que os poliedros sejam construídos a cada iteração, isto é, o programa “*poliedro*” é uma subrotina do branch-and-bound. Além disso, o ponto que caracteriza o bound, *xbound* (caso exista), é repassado para a iteração sucessiva.

A visualização gráfica do desenvolvimento do branch-and-bound ocorre após determinarmos o politopo correspondente ao subproblema. Mas, se o subproblema for inviável (região viável vazia) o programa `poliedro_bb` simplesmente o ignora.

Visualização tridimensional do método Branch-and-Bound

A visualização tridimensional do método Branch-and-Bound se diferencia da visualização bidimensional analogamente ao método Simplex, isto é, para visualizarmos devemos primeiramente rotacionar as figuras geradas pelo programa `POLIEDRO_V1`. Mostraremos este fato no exemplo a seguir.

Considere o problema `tenda.m` localizado no diretório `\DATA\R3`. Este problema gera a matriz A, e os vetores b e c do problema (A.2) de Programação Interira (PPI). Após a inicialização do programa `POLIEDRO_V1` execute o comando:

```
>> help tenda
   tenda.m
   Problema proposto para trabalhar com simplex ou branch_and_bound em R3
   Problema de programação linear interira
   tipo= dual:  restrições de desigualdade com variáveis positivas.
   min c'x subject to Ax<=b, x>=0, em que
   A:= matriz das restricoes (desigualdades);
   b:= termos independentes (desigualdades);
   c:= custo relacionado;
```

O comando `help` mostra os dados gerados pelo arquivo `tenda.m`. Para executá-lo use os comandos:

```
>> tenda
>> poliedro_bb(c,A,b)
```

Este programa gera duas figuras que devem ser posicionadas separadamente para uma melhor visualização.

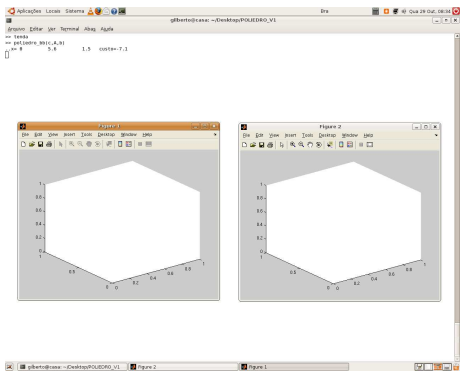


Figura A.15: Figura 1 e Figura 2 geradas pelo programa `POLIEDRO_V1`.

Para executar o próximo passo, basta colocar o cursor sobre a Figura 1, gerada pelo Matlab e clicar com o botão central do mouse ou usar a teclar `Enter`.

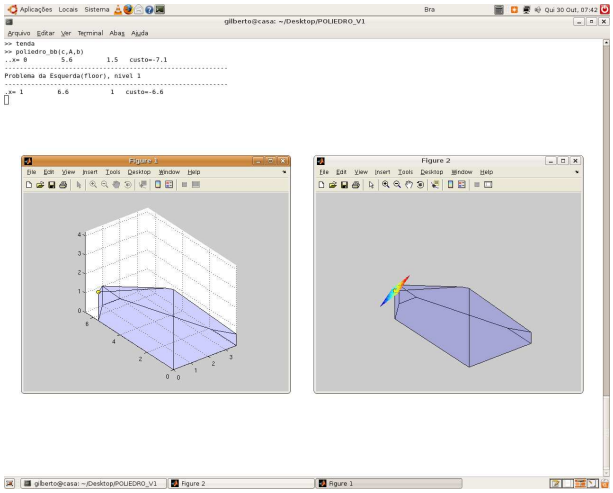


Figura A.16: Ponto ótimo determinado no nível 0.

Analogamente ao processo bidimensional, a Figura 1 representa o subproblema que esta sendo resolvido no nível, já a Figura 2 representa os subproblemas sobrepostos. Para uma melhor visualização dos poliedros, Figura 1 e Figura 2, podemos rotacioná-los.

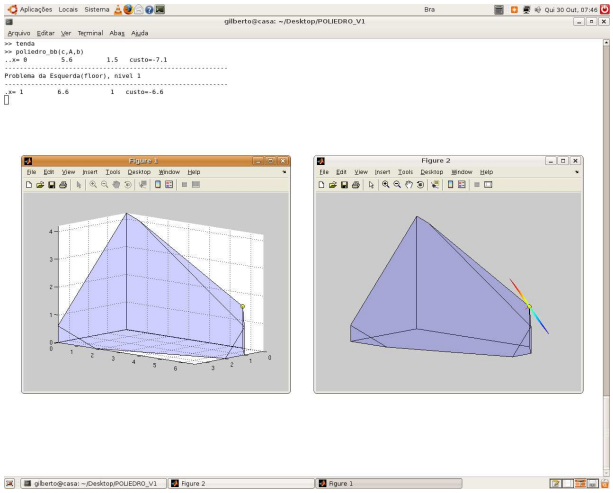


Figura A.17: Figuras após rotacioná-las.

A rotação das figuras são simultâneas e executada com o auxílio da subrotina `click1`, isto é, podemosrotacionar as figuras da seguinte forma:

Primeiramente, colocamos o cursos sobre a Figura 1, em seguida, utilizamos o teclado como na tabela a seguir:

Tecla	Tecla	Ação
↑	w	rotaciona a figura para cima;
↓	s	rotaciona a figura para baixo;
←	a	rotação horária;
→	d	rotação antihorária.

Para executar o próximo passo do algoritmo Branch-and-Bound, basta colocar o cursor sobre a Figura 1 e, em seguida, teclar `Enter` ou o botão central do mouse. Seguindo esse processo, encontramos a solução ótima do problema de Programação Inteira.

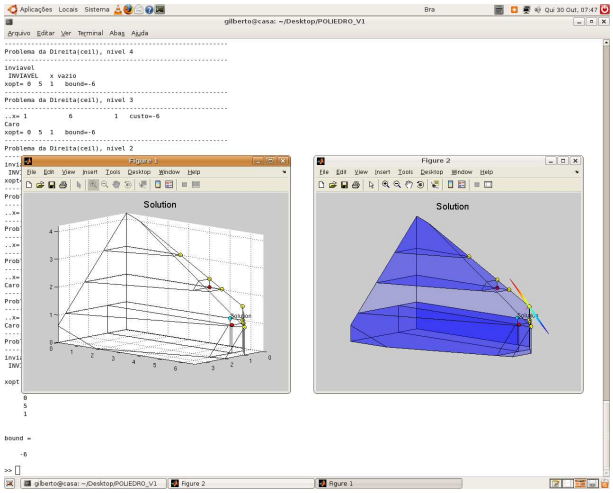


Figura A.18: Solução do problema `tenda`.

Analogamente, ao programa `poliedro_simplex`, o programa `poliedro_bb` utiliza as cores para determinar a solução ou o bound do problema pela cor azul, além disso, a cor vermelha representa se o subproblema é considerado caro, e a cor amarela distingue as soluções fracionárias das inteiras. Aqui, as mesmas ferramentas do Matlab que auxiliavam na visualização no `poliedro_simplex` são empregadas.

Novamente, relacionamos as cores dos pontos que são encontrados pelo método Branch-and-Bound: (Veja a Figura (A.19))

Cor	Representação
●	Ponto fracionário;
●	Ponto com custo “caro”;
●	Bound ou solução do problema.

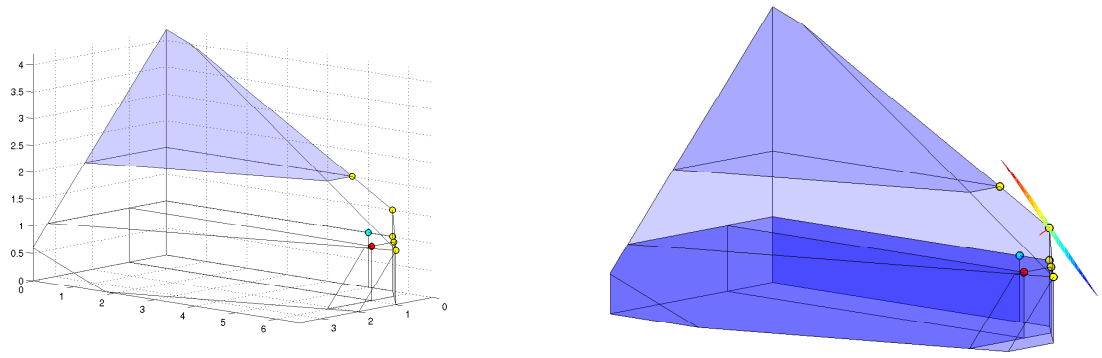


Figura A.19: Pontos calculados pelo programa `poliedro_bb`.

A.6 Problemas Extras

Nesta seção, selecionamos 5 problemas de Programação Linear e Inteira, e utilizamos o `POLIEDRO_V1` para resolvê-los. Os problemas são encontrados nos diretórios `\DATA`, `\DATA\R2` e `\DATA\R3`. Após o processo de inicialização do programa `POLIEDRO_V1` resovemos os seguintes problemas.

Problema 1: Este problema é definido pelo arquivo `exercicioA2b.m`.

```
>> help exercicioA2d
   exercicioA2d.m -- problema 1 bidimensional
   Um comerciante pretende obter uma quantidade nao superior a 5 toneladas
   de um certo produto.  Esta quantidade pode ser encomendada a duas fabri-
   cas A e B. A fabrica A garante um lucro de 4 mil Reais por tonelada mas
   nao pode fornecer mais de 3 toneladas.  A fabrica B garante um lucro de
   3,5 mil Reais por tonelada e pode fornecer qualquer quantidade.  Qual o
   esquema de encomenda que origina um lucro maximo?
   variavel de decisao:
   x:= quantidade do produto encomendada da fabrica A
   y:= quantidade do produto encomendada da fabrica B
    max 4x+3,5y     ou     min -4x-3,5y
   s.a.  x+y<=5           s.a.  x+y<=5
           x<=3                   x<=3
```

Para resolvermos este problema utilizando o método Simplex, basta executar os comandos:

```
>> exercicioA2d
>> poliedro_simplex(c,A,b)
```

Problema 2: Este problema é definido pelo arquivo `exercicioB2d.m`.

```
>> help exercicioB2d
   exercicioB2d.m -- problema 2 bidimensional
   Uma firma produz e comercializa dois tipos de tapetes.  Para a produ-
   cao de tapetes sao utilizadas duas maquinas A e B que funcionam res-
   pectivamente 12 e 14 horas por dia.  Para fabricar 100m2 do primeiro
   tipo de tapete sao necessarias 3 horas de trabalho na maquina A e 7
   horas na maquina B. A mesma quantidade de tapetes do tipo 2 requer 4
   horas e 2 horas respectivamente nas maquinas A e B. Sabendo que o
   lucro obtido com a producao de 100m2 do primeiro tipo de tapete eh de
   4 u.m.  e o correspondente valor para o tipo 2 eh de 3 u.m., qual deve
   ser a utilizacao diaria das maquinas de modo a garantir um lucro ma-
   ximo para a producao de tapetes?
   variavel de decisao:
   x:= quantidade de tapete do tipo 1 produzido pela firma;
   y:= quantidade de tapete do tipo 2 produzido pela firma.
    max     4x+3y       ou       min     -4x-3y
   s.a.   3x+4y<=12            s.a.   3x+4y<=12
          7x+2y<=14                   7x+2y<=14
```

Para resolvermos este problema utilizando o método Branch-and-Bound, basta executar os comandos:

```
>> exercicioB2d
>> poliedro_bb(c,A,b)
```

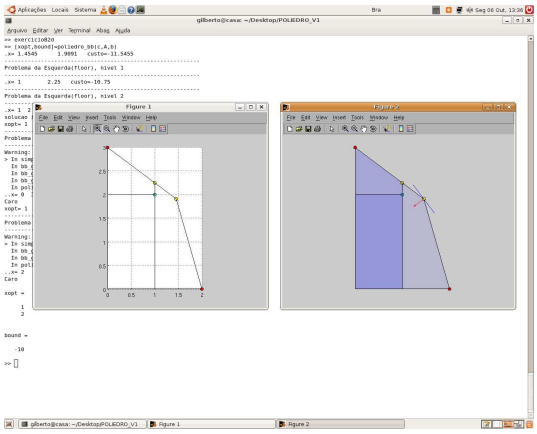


Figura A.20: Solução do problema 2.

Problema 3: Este problema é definido pelo arquivo `exercicioA3d.m`.

Uma empresa tem uma ocupação descontínua de certa linha de produção e a administração está a estudar a hipótese de destinar esse excesso de capacidade à produção de um ou mais de três produtos designados por A, B e C. A capacidade disponível das máquinas que poderão limitar a produção é a seguinte:

Maquina	Tempo disponivel (horas/semana)
1	200
2	150
3	50

O número de horas-máquina necessário para cada unidade dos referidos produtos é:

Maquina	Produto A	Produto B	Produto C
1	8	2	3
2	4	3	—
3	2	—	1

O departamento de vendas informa que as vendas potenciais dos produtos A e B ultrapassam a capacidade máxima de produção e que a venda potencial do produto C é de 20 unidades semanais. Os lucros unitários são 20, 6 e 8 u.m., respectivamente para os produtos A, B e C. Pretende-se determinar o plano ótimo de produção.

Variável de decisão:

x:= quantidade produzida do produto A pela empresa;

y:= quantidade produzida do produto B pela empresa;

z:= quantidade produzida do produto C pela empresa.

$$
\begin{aligned}
\text{minimizar} \quad & -20x - 6y - 8z \\
\text{sujeito a} \quad & 8x + 2y + 3z \leq 200 \\
& 4x + 3y \leq 150 \\
& 2x + z \leq 50 \\
& z \leq 20 \\
& x, y, z \geq 0, \\
& x \in \mathbb{Z}^n.
\end{aligned}
$$

Problema 4: Este problema é definido pelo arquivo `exercicioB3d.m`.

Uma fábrica produz três bens A, B e C. O lucro médio que obtem com A é de 40 u.m./ton, com B é de 30 u.m./ton e com C é de 34 u.m./ton. A unidade de produção compõe-se de 3 seções: a de corte, a de mistura e a de embalagem, que podem ser utilizadas durante 8 horas por dia. O processo de produção caracteriza-se por:

1. O bem A é primeiro cortado e depois embalado. Cada tonelada consome 1/2 hora na seção de corte e 2/3 hora na seção de embalagem.

2. O bem B é primeiro misturado e depois embalado. Cada tonelada consome 1 hora na seção de mistura e 2/3 hora na seção de embalagem.

3. O bem C é primeiro cortado, depois misturado e por fim embalado. Cada tonelada consome 1/4 hora na seção de corte, 1/5 hora na seção de mistura e 1/4 hora na seção de embalagem.

Qual a combinação de produção que a empresa deve realizar diariamente a fim de maximizar o lucro?

Variável de decisão:
x:= quantidade produzida do produto A pela fábrica;
y:= quantidade produzida do produto B pela fábrica;
z:= quantidade produzida do produto C pela fábrica.

$$
\begin{aligned}
\text{minimizar} \quad & -30x - 40y - 34z \\
\text{sujeito a} \quad & 2x + z \leq 32 \\
& 3y + 2z \leq 16 \\
& 8x + 2y + 3z \leq 96 \\
& x \leq 48/7 \\
& y \leq 48/7 \\
& z \leq 48/7 \\
& x, y, z \geq 0 \\
& x \in \mathbb{Z}^n.
\end{aligned}
$$

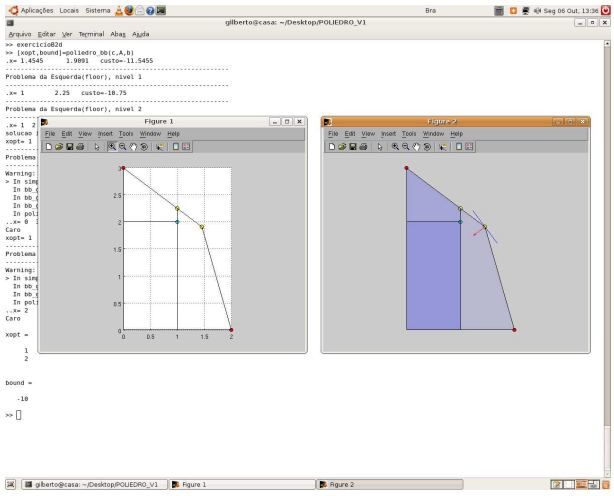


Figura A.21: Solução do problema 4.

Problema 5: Este problema é definido pelo arquivo `correio.m`.

Uma estação de Correios emprega um número diferente de trabalhadores em cada dia da semana:

dom	seg	ter	qua	qui	sex	sab
11	17	13	15	19	14	16

O acordo sindical obriga a que cada trabalhador trabalhe 5 dias consecutivos e depois tenha 2 dias de folga seguidos. Pretende-se saber qual o número mínimo de empregados que satisfaz as necessidades diárias.
Variável de decisão:
x_i:= número de pessoas que começam a trabalhar no dia da semana i, em que $i = 1, \ldots, 7$.

$$\begin{array}{ll} \text{minimizar} & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \\ \text{sujeito a} & Ax \leq b \\ & x \geq 0, \\ & x \in \mathbb{Z}^n. \end{array}$$

O problema `correio` possui dimensão 7, portanto não é possível representá-lo graficamente. Por essa razão o programa `POLIEDRO_V1` ignora a representação gráfica e repassa somente os dados de saída, o ponto ótimo e a solução do problema. Para resolvermos este problema de Programação Linear Inteira através do programa `POLIEDRO_V1` executamos os comandos:

```
>> correio
>> poliedro_bb(c,A,b)
```

$xopt =$

3
7
1
3
0
7
2

$bound =$

23.0000

em que, x é o ponto ótimo do problema e s a solução.

A.7 Diretório DATA

Nesta seção apresentamos os arquivos "`.m`" e subdiretórios que compõe o banco de dados original do programa `POLIEDRO_V1`. Os problemas pré-formulados estão no diretório `DATA`, classificados da seguinte forma: Nos subdiretórios `R2` e `R3` estão os problemas bidimensionais e tridimensionais, respectivamente. Já os problemas com dimensão superior a três estão no diretóiro `DATA`. Veja a tabela abaixo.

Diretório DATA

Arquivo	Subdiretório R2	Subdiretório R3
correio.m	barco.m	balao.m
	dodecaedro.m	balao2.m
	dual.m	chvatal281.m
	exercicioA2d.m	cubo.m
	exercicioB2d.m	cubo2.m
	hexagono.m	diamante.m
	pentagono.m	exercicioA3d.m
	triangulo.m	exercicioB3d.m
	vanderbei.m	tenda.m
	vanderbei2.m	tenda2.m
		vaso.m

Os arquivos "`.m`" do banco de dados definem o poliedro através da matriz A e do vetor b. Por outro lado, se estamos interessados em resolver um problema de Programação Linear, o custo c deve ser gerado. Somente alguns problemas já possuem o custo pré-definido. Para verificar os dados dos arquivos, basta executar o comando `help`. Por exemplo, para visualizar os dados do arquivo `pentagono.m`, executamos o comando:

```
>> help pentagono
  pentagono.m
  Problema proposta para trabalhar com simplex e branch_and_bound em R2
  tipo= dual:  restrições de desigualdade com variáveis positivas.
  min c'x subject to Ax<=b, x>=0, em que
  A:= matriz das restricoes (desigualdades);
  b:= termos independentes (desigualdades);
  c:= custo relacionado;
```

A.8 Considerações Finais

Esta é a primeira versão do **Guia do Usuário** para utilização do programa POLIEDRO_V1 e foi elaborado para usuários com um conhecimento básico em Matlab e em Programação Linear. Por tais razões, aceito sugestões e comentários para o aprimoramento deste documento.

Finalmente, agradeço a todos os colegas do Departamento de Matemática da Universidade Federal de Santa Catarina que contribuiram diretamente ou indiretamente na elaboração deste guia e do programa POLIEDRO_V1.

Gilberto Souto

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo