INF05010 - Otimização combinatória
Notas de aula

Alysson M. Costa, Luciana Buriol, Marcus Ritt
{amcosta,buriol,mrpritt}@inf.ufrgs.br

23 de Abril de 2009

Universidade Federal do Rio Grande do Sul
Instituto de Informática
Departamento de Informática Teórica

Na parte I, as notas de aula seguem o livro "Linear programming: Foundations and extensions" do Robert J. Vanderbei, Universidade Princeton, disponível em http://www.princeton.edu/~rvdb/LPbook.

Fonte das imagens:
George Dantzig (16): INFORMS, Jean Baptiste Joseph Fourier (15): Wikipedia, Xadrez (84): Wikipedia, Mauricio G. C. Resende (150): Página pessoal, Fred Glover (153): Página pessoal, Pierre Hansen (157): Página pessoal, Pablo Moscato (167): Página pessoal.

Conteúdo

Parte I

Programação linear

Introdução

> If one would take statistics about which mathematical problem is using up most of the computer time in the world, then ... the answer would probably be linear programming. (Laszlo Lovasz)

1 Introdução

1.1 Exemplo

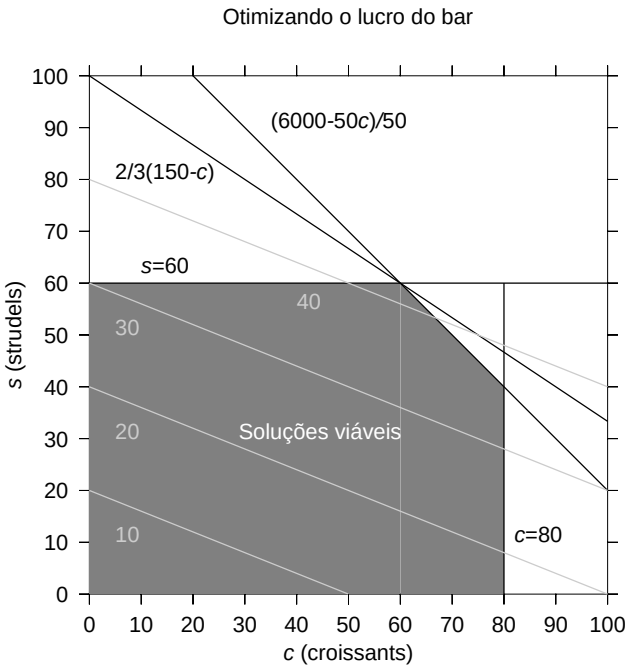
Exemplo 1.1 (No Ildo)
Antes da aula visito o Ildo para tomar um café e comer um Croissant. Ele me conta: "Estou especializado em Croissants e Strudels. Tenho um lucro de 20 centavos por Croissant e 50 centavos por Strudel. Diariamente até 80 clientes compram um Croissant e até 60 um Strudel. Mas infelizmente, o Ildo apenas disponibiliza de 150 ovos e 6 kg de açúcar por dia. Entre outros ingredientes, preciso um ovo e 50g de açúcar para cada Croissant e um ovo e meio e 50g de açúcar para cada Strudel. Agora, professor, quantas Croissants e Strudels devo produzir para obter o maior lucro?"

Sejam c e s o número de Croissants e Strudels, respectivamente. O lucro do Ildo em Reais é $0.2c+0.5s$. Seria ótimo produzir todos 80 Croissants e 60 Strudels, mas uma conta simples mostra que não temos ovos e açúcar suficientes. Para produzir os Croissants e Strudels precisamos $c + 1.5s$ ovos e $50c + 50sg$ de açúcar que não podem ultrapassar 150 ovos e 6000g. Com a condição óbvia que $c \geq 0$ e $s \geq 0$ chegamos no seguinte problema de otimização:

$$\begin{aligned} \textbf{maximiza} \quad & 0.2c + 0.5s \\ \textbf{sujeito a} \quad & c + 1.5s \leq 150 \\ & 50c + 50s \leq 6000 \\ & c \leq 80 \\ & s \leq 60 \\ & c, s \geq 0 \end{aligned}$$

Como resolver esse problema? Com duas variáveis podemos visualizar a situação num grafo com c no eixo x e s no eixo y

No Ildo



que nesse caso permite resolver o problema graficamente. Desenhando diversos *conjuntos de nível* (ingl. *level set*) com valor da função objetivo 10, 20, 30, 40 é fácil observar que o lucro máximo se encontra no ponto $c = s = 60$, e possui um valor de 42 reais.

$\Diamond$

Isso é um exemplo de um problema de otimização. A forma geral de um *problema de otimização* (ou de *programação matemática*) é

$$\begin{aligned} \textbf{opt} \quad & f(x) \\ \textbf{sujeito a} \quad & x \in V \end{aligned}$$

com

- um *objetivo* $\textbf{opt} \in \{\max, \min\}$,
- uma *função objetivo* (ou função critério) $f : V \to \mathbb{R}$,
- um conjunto de *soluções viáveis* (ou soluções candidatas) V.

Falamos de um *problema de otimização combinatória*, se V é discreto.
Nessa generalidade um problema de otimização é difícil de resolver. O exemplo 1.1 é um problema de *otimização linear* (ou *programação linear*):

- as variáveis da solução são $x_1, \ldots, x_n \in \mathbb{R}$
- a função de otimização é linear em $x_1, \ldots, x_n$:

$$f(x_1, \ldots, x_n) = c_1 x_1 + \cdots + c_n x_n \tag{1.1}$$

- as soluções viáveis são dadas implicitamente por m *restrições lineares*

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \bowtie_1 b_1 \tag{1.2}$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \bowtie_2 b_2 \tag{1.3}$$
$$\cdots \tag{1.4}$$
$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \bowtie_m b_m \tag{1.5}$$

com $\bowtie_i \in \{\leq, =, \geq\}$.

Exemplo 1.2 (O problema da dieta)
Suponha que temos uma tabela de nutrientes de diferentes tipos de alimentos. Sabendo o valor diário de referência (VDR) de cada nutriente (quantidade de nutriente que deve ser ingerido) e o preço de cada unidade de alimento, qual a dieta ótima, i.e. que contém ao menos o valor diário de referência, mas de menor custo?
Com m nutrientes e n alimentos, seja a_{ij} a quantidade do nutriente i no alimento j, r_i o valor diário de referência do nutriente i e c_j o preço do alimento j. Queremos saber as quantidades x_j de cada alimento que

$$\begin{aligned} \textbf{minimiza} \quad & c_1 x_1 + \cdots + c_n x_n \\ \textbf{sujeito a} \quad & a_{11}x_1 + \cdots + a_{1n}x_n \geq r_1 \\ & \cdots \\ & a_{m1}x_1 + \cdots + a_{mn}x_n \geq r_m \\ & x_1, \ldots, x_n \geq 0 \end{aligned}$$

$\Diamond$

Exemplo 1.3 (Problema de transporte)
Uma empresa agrária tem m depósitos, cada um com um estoque de a_i $(1 \leq i \leq m)$ toneladas de milho. Ela quer encaminhar b_j $(1 \leq j \leq n)$ toneladas de milho para n clientes diferentes. O transporte de uma tonelada do depósito i para cliente j custa R$ c_{ij}. Qual seria o esquema de transporte de menor custo?
Como problema de otimização linear, podemos introduzir como variáveis x_{ij} o peso dos produtos encaminhados pelo depósito i para cliente j, e queremos resolver

$$\begin{aligned}
\textbf{minimiza} \quad & \sum_{ij} c_{ij} x_{ij} \\
\textbf{sujeito a} \quad & \sum_{j} x_{ij} \leq a_i \qquad \text{para todo fornecedor } i \\
& \sum_{i} x_{ij} = b_j \qquad \text{para todo cliente } j \\
& x_{ij} \geq 0
\end{aligned}$$

Concretamente, suponha que temos a situação da figura 1.1. A figura mostra

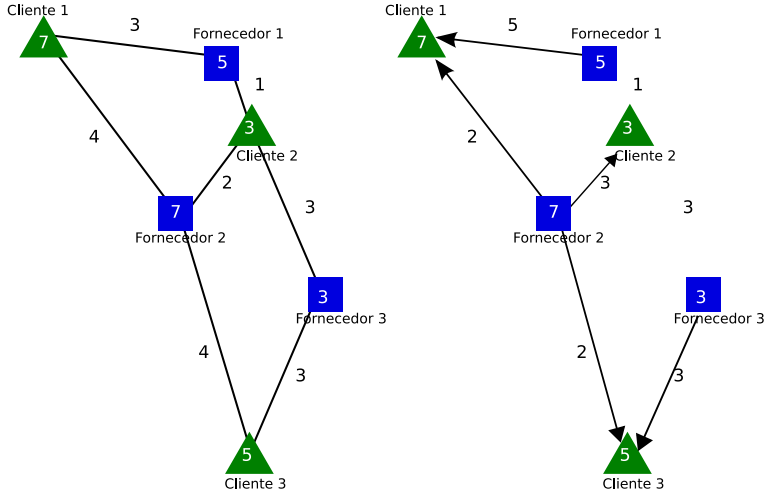


Figura 1.1: Esquerda: Instância do problema de transporte. Direita: Solução ótima dessa instância.

as toneladas disponíveis de cada fornecedor, a demanda (em toneladas) de cada cliente e as distâncias (em km) entre eles. O transporte custa R$ 1000

por km e tonelada. Observe que um transporte do fornecedor 1 para cliente 3 e fornecedor 3 para cliente 1 não é possível. Nós usaremos uma distância grande de 100km nesses casos (outra possibilidade seria usar restrições $x_{13} = x_{31} = 0$.

$$\begin{aligned}
\textbf{minimiza} \quad & 3x_{11} + x_{12} + 100x_{13} + 4x_{21} + 2x_{22} \\
& + 4x_{23} + 100x_{31} + 3x_{32} + 3x_{33} \\
\textbf{sujeito a} \quad & x_{11} + x_{12} + x_{13} \leq 5 \\
& x_{21} + x_{22} + x_{23} \leq 7 \\
& x_{31} + x_{32} + x_{33} \leq 3 \\
& x_{11} + x_{21} + x_{31} = 7 \\
& x_{12} + x_{22} + x_{32} = 3 \\
& x_{13} + x_{23} + x_{33} = 5 \\
& x_{ij} \geq 0
\end{aligned}$$

Qual seria a solução ótima? A figura da direita mostra o número ótimo de toneladas transportadas. O custo mínimo é 46 (em R$ 1000). ♢

Para simplificar a descrição, podemos usar matrizes e vetores. Como exemplo, considere o problema de otimização generalizado visto anteriormente (equações 1.1-1.5). Com $A := (a_{ij}) \in \mathbb{R}^{m \times n}$, $b := (b_i) \in \mathbb{R}^m$, $c := (c_i) \in \mathbb{R}^n$ e $x = (x_i) \in \mathbb{R}^n$ o sistema acima é equivalente a

$$\begin{aligned}
\textbf{opt} \quad & c^t x \\
\textbf{sujeito a} \quad & a_i x \bowtie_i b_i \qquad 1 \leq i \leq m
\end{aligned}$$

(Denotamos com a_i a i-ésima linha e como a^j a j-ésima coluna de A.)

1.2 Formas normais

Conversões

É possível converter

- um problema de minimização para um problema de maximização

$$\min c^t x \iff \max -c^t x$$

- uma restrição $\geq$ para uma restrição $\leq$

$$a_i x \geq b_i \iff -a_i x \leq -b_i$$

- uma igualdade para desigualdades

$$a_i x = b_i \iff a_i x \leq b_i \wedge a_i x \geq b_i$$

Conversões

- uma desigualdade para uma igualdade

$$a_i x \leq b \iff a_i x + x_{n+1} = b_i \wedge x_{n+1} \geq 0$$
$$a_i x \geq b \iff a_i x - x_{n+1} = b_i \wedge x_{n+1} \geq 0$$

  usando uma nova *variável de folga ou excesso* x_{n+1} (inglês: slack and surplus variables).

- uma variável x_i sem restrições para duas positivas

$$x_i^+ \geq 0 \wedge x_i^- \geq 0$$

  substituindo x_i por $x_i^+ - x_i^-$.

Essas transformações permitem descrever cada problema linear em uma *forma padrão*.

Forma padrão

$$\begin{array}{rl} \textbf{maximiza} & c^t x \\ \textbf{sujeito a} & Ax \leq b \\ & x \geq 0 \end{array}$$

As restrições $x \geq 0$ se chamam *triviais*.

Exemplo 1.4
Dado o problema

$$\begin{array}{rl} \textbf{minimiza} & 3x_1 - 5x_2 + x_3 \\ \textbf{sujeito a} & x_1 - x_2 - x_3 \geq 0 \\ & 5x_1 + 3x_2 + x_3 \leq 200 \\ & 2x_1 + 8x_2 + 2x_3 \leq 500 \\ & x_1, x_2 \geq 0 \end{array}$$

vamos substituir **minimiza** para **maximiza**, converter a primeira desigualdade de $\geq$ para $\leq$ e introduzir $x_3 = x_3^+ - x_3^-$ com duas variáveis positivas x_3^+

e x_3^- para obter a forma padrão

$$\begin{aligned} \textbf{maximiza} \quad & -3x_1 + 5x_2 - x_3^+ + x_3^- \\ \textbf{sujeito a} \quad & -x_1 + x_2 + x_3^+ - x_3^- \leq 0 \\ & 5x_1 + 3x_2 + x_3^+ - x_3^- \leq 200 \\ & 2x_1 + 8x_2 + 2x_3^+ - 2x_3^- \leq 500 \\ & x_1, x_2, x_3^+, x_3^- \geq 0 \end{aligned}$$

Em notação matricial temos

$$c = \begin{pmatrix} -3 \\ 5 \\ -1 \\ 1 \end{pmatrix}; b = \begin{pmatrix} 0 \\ 200 \\ 500 \end{pmatrix}; A = \begin{pmatrix} -1 & 1 & 1 & -1 \\ 5 & 3 & 1 & -1 \\ 2 & 8 & 2 & -2 \end{pmatrix}.$$

$\Diamond$

1.3 Notas históricas

História da programação linear

- Jean Baptiste Joseph Fourier (1826): Método de resolver um sistema de desigualdades (eliminação de Fourier-Motzkin) [5].
- Leonid Kantorovich (1939): Programação linear.
- George Bernard Dantzig (1948): Método Simplex.
- John von Neumann: Dualidade.
- Leonid Khachiyan (1979): Método de ellipsoides.
- Narendra Karmarkar (1984): Métodos de pontos interiores.

Jean Baptiste Joseph Fourier (*1768, +1830)

Pesquisa operacional, otimização e "programação"

- "The discipline of applying advanced analytical methods to help make better decisions" (INFORMS)
- A noção foi criada no segunda guerra mundial, para métodos científicos de análise e predição de problemas logísticos.
- Hoje se aplica para técnicas que ajudam decisões de execução e coordenação de operações em organizações.
- Os problemas da pesquisa operacional são problemas de otimização.
- "Programação" $\neq$ "Programação"
  - Não se refere à computação: a noção significa "planejamento" ou "agendamento".

George Bernard Dantzig (*1914, +2005)

Técnicas da pesquisa operacional

- Em geral: Técnicas algorítmicas conhecidas como
  - Modelagem matemática (equações, igualdades, desigualdades, modelos probabilísticos,...)
  - Algoritmos gulosos, randômicos, ...; programação dinâmica, linear, convexo, ...
  - Heurísticas e algoritmos de aproximação.
- Algumas dessas técnicas se aplicam para muitos problemas e por isso são mais comuns:
  - Exemplo: Programação linear.

2 O método Simplex

Graficamente, é difícil resolver sistemas com mais de três variáveis. Portanto é necessário achar métodos que permitam resolver sistemas grandes. Um método importante se chama Simplex. Nós vamos estudar esse método primeiramente através da aplicação a um exemplo.

2.1 Um exemplo

Começamos com o seguinte sistema em forma padrão:

Exemplo: Simplex

$$\begin{aligned} \textbf{maximiza} \quad & z = 6x_1 + 8x_2 + 5x_3 + 9x_4 \\ \textbf{sujeito a} \quad & 2x_1 + x_2 + x_3 + 3x_4 \leq 5 \\ & x_1 + 3x_2 + x_3 + 2x_4 \leq 3 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Introduzimos variáveis de folga e reescrevemos as equações:

Exemplo: Com variáveis de folga

$$\begin{aligned} \textbf{maximiza} \quad & z = 6x_1 + 8x_2 + 5x_3 + 9x_4 && (2.1) \\ \textbf{sujeito a} \quad & w_1 = 5 - 2x_1 - x_2 - x_3 - 3x_4 && (2.2) \\ & w_2 = 3 - x_1 - 3x_2 - x_3 - 2x_4 && (2.3) \\ & x_1, x_2, x_3, x_4, w_1, w_2 \geq 0 \end{aligned}$$

Nesse exemplo é fácil obter uma solução viável, escolhendo $x_1 = x_2 = x_3 = x_4 = 0$. Podemos verificar que $w_1 = 5$ e $w_2 = 3$ e todas as restrições são respeitadas. O valor da função objetivo seria 0. Uma outra solução viável é $x_1 = 1$, $x_2 = x_3 = x_4 = 0$, $w_1 = 3$, $w_2 = 2$ com valor $z = 6$.
Com 6 variáveis e duas equações independentes o espaço de soluções do sistema de equações lineares dado pelas restrições tem $6 - 2 = 4$ graus de liberdade.

Uma solução viável com no mínimo esse número de *variáveis nulas* (igual a 0) se chama uma *solução básica viável*. Logo nossa primeira solução acima é uma solução básica viável.
A idéia do método Simplex é percorrer soluções básicas viáveis, aumentando em cada passo o valor z da função objetivo.
Logo nosso próximo objetivo é aumentar o valor da função objetivo z. Para esse fim, podemos aumentar o valor das variáveis x_1, x_2, x_3 ou x_4, pois o coeficiente delas é positivo. Escolhemos x_4, porque essa variável tem o maior coeficiente. Não podemos aumentar x_4 arbitrariamente: Para respeitar as restrições $w_1, w_2 \geq 0$ temos os limites

Limites

$$w_1 = 5 - 3x_4 \geq 0 \Longleftrightarrow x_4 \leq 5/3$$
$$w_2 = 3 - 2x_4 \geq 0 \Longleftrightarrow x_4 \leq 3/2$$

ou seja $x_4 \leq 3/2$. Aumentando x_4 o máximo possível, obtemos $x_4 = 3/2$ e $w_2 = 0$. Os valores das demais variáveis não mudam. Essa solução respeita novamente todas as restrições, e portanto é *viável*. Ainda, como trocamos uma variável nula (x_4) com uma outra não-nula (w_2) temos uma nova solução básica viável

Solução básica viável

$$x_1 = x_2 = x_3 = 0; x_4 = 3/2; w_1 = 1/2; w_2 = 0$$

com valor da função objetivo $z = 13.5$.
O que facilitou esse primeiro passo foi a forma especial do sistema de equações. Escolhemos quatro variáveis independentes (x_1, x_2, x_3 e x_4) e duas variáveis dependentes (w_1 e w_2). Essas variáveis são chamadas *não-básicas* e *básicas*, respectivamente. Na nossa solução básica viável todas variáveis não-básicas são nulas. Logo, pode-se aumentar uma variável não-básica cujo coeficiente na função objetivo seja positivo (para aumentar o valor da função objetivo). Inicialmente tem-se as seguintes variáveis básicas e não-básicas

$$\mathcal{B} = \{w_1, w_2\}; \qquad \mathcal{N} = \{x_1, x_2, x_3, x_4\}.$$

Depois de aumentar x_4 (e consequentemente zerar w_2) podemos escolher

$$\mathcal{B} = \{w_1, x_4\}; \qquad \mathcal{N} = \{x_1, x_2, x_3, w_2\}.$$

A variável x_4 se chama *variável entrante*, porque ela entra no conjunto de variáveis básicas B. Analogamente w_2 se chama *variável sainte*.
Para continuar, podemos reescrever o sistema atual com essas novas variáveis básicas e não-básicas. A segunda restrição 2.3 é fácil de reescrever

$$w_2 = 3 - x_1 - 3x_2 - x_3 - 2x_4 \iff x_4 = 3/2 - 1/2x_1 - 3/2x_2 - 1/2x_3 - 1/2w_2$$

Além disso, temos que reescrever a primeira restrição 2.2, porque a variável básica w_1 depende de x_4 que agora é básica também. Nosso objetivo é escrever todas variáveis básicas em termos de variáveis não-básicas. Para esse fim, podemos usar combinações lineares da linhas, que eliminam as variáveis não-básicas. Em nosso exemplo, a combinação (2.2)$-3/2$(2.3) elimina x_4 e resulta em

$$w_1 - 3/2w_2 = 1/2 - 1/2x_1 + 7/2x_2 + 1/2x_3$$

e colocando a variável não-básica w_2 no lado direto obtemos

$$w_1 = 1/2 - 1/2x_1 + 7/2x_2 + 1/2x_3 + 3/2w_2.$$

Temos que aplicar uma operação semelhante à função objetivo que ainda depende da variável básica x_4. Escolhemos (2.1)$-9/2$(2.3) para obter

$$z = 27/2 + 3/2x_1 - 11/2x_2 + 1/2x_3 - 9/2w_2.$$

Novo sistema

$$\begin{aligned} \textbf{maximiza} \quad & z = 27/2 + 3/2x_1 - 11/2x_2 + 1/2x_3 - 9/2w_2 \\ \textbf{sujeito a} \quad & w_1 = 1/2 - 1/2x_1 + 7/2x_2 + 1/2x_3 + 3/2w_2 \\ & x_4 = 3/2 - 1/2x_1 - 3/2x_2 - 1/2x_3 - 1/2w_2 \\ & x_1, x_2, x_3, x_4, w_1, w_2 \geq 0 \end{aligned}$$

que obtemos após uma operação de trocar as variáveis x_4 e w_2. Essa operação se chama um *pivot*. Observe que no novo sistema é fácil recuperar toda informação atual: zerando as variáveis não-básicas obtemos diretamente a solução $x_1 = x_2 = x_3 = w_2 = 0$, $w_1 = 1/2$ e $x_4 = 3/2$ com função objetivo $z = 27/2$.
Antes de continuar "pivotando" introduzimos uma forma mais simples de escrever o sistema

Dicionário

$$\begin{array}{rlrrrr} & & \downarrow & & & \\ z & = 27/2 & +3/2x_1 & -11/2x_2 & +1/2x_3 & -9/2w_2 \\ \hline \leftarrow w_1 & = 1/2 & -1/2x_1 & +7/2x_2 & +1/2x_3 & +3/2w_2 \\ x_4 & = 3/2 & -1/2x_1 & -3/2x_2 & -1/2x_3 & -1/2w_2 \end{array}$$

que se chama *dicionário* (inglês: dictionary).
No próximo passo podemos aumentar somente x_1 ou x_3 porque somente elas têm coeficientes positivos. Aumentado x_1 temos que respeitar $x_1 \leq 1$ (da primeira restrição) e $x_1 \leq 3$ (da segunda). Logo a primeira restrição é mais forte, x_1 é a variável entrante, w_1 a variável sainte, e depois do pivot obtemos

Segundo passo

$$\begin{array}{rlrrrr} & & & \downarrow & & \\ z & = 15 & -3w_1 & +5x_2 & +2x_3 & \\ \hline x_1 & = 1 & -2w_1 & +7x_2 & +x_3 & +3w_2 \\ \leftarrow x_4 & = 1 & +w_1 & -5x_2 & -x_3 & -2w_2 \end{array}$$

No próximo pivot x_2 entra. A primeira restrição não fornece limite para x_2, porque o coeficiente de x_2 é positivo! Mas a segunda $x_2 \leq 1/5$ e x_4 sai da base. O resultado do pivot é

Terceiro passo

$$\begin{array}{rlrrrr} & & & & \downarrow & \\ z & = 16 & -2w_1 & -x_4 & +x_3 & -2w_2 \\ \hline x_1 & = 12/5 & -3/5w_1 & -7/5x_4 & -2/5x_3 & +1/5w_2 \\ \leftarrow x_2 & = 1/5 & +1/5w_1 & -1/5x_4 & -1/5x_3 & -2/5w_2 \end{array}$$

O próximo pivot: x_3 entra, x_2 sai:

Quarto passo

$$\begin{array}{rlrrrr} z & = 17 & -w_1 & -2x_4 & -5x_2 & -4w_2 \\ \hline x_1 & = 2 & -w_1 & -x_4 & +2x_2 & +w_2 \\ x_3 & = 1 & +w_1 & -x_4 & -5x_2 & -2w_2 \end{array}$$

Agora, todos coeficientes da função objetivo são negativos. Isso significa, que não podemos mais aumentar nenhuma variável não-básica. Como esse sistema

é equivalente ao sistema original, qualquer solução tem que ter um valor menor ou igual a 17, pois todas as variáveis são positivas. Logo chegamos no resultado final: a solução

$$w_1 = x_4 = x_2 = w_2 = 0; x_1 = 2; x_3 = 1$$

com valor objetivo 17, é ótima!
Concluímos esse exemplo com mais uma observação. O número de soluções básicas viáveis é limitado. Em nosso exemplo, se escolhemos um subconjunto de quatro variáveis nulas, as duas equações determinam as variáveis restantes. Logo temos no máximo $\binom{6}{4} = 15$ soluções básicas viáveis. Com m equações e n variáveis, uma solução básica viável possui $n-m$ variáveis independentes (que são nulas). Portanto, o número de soluções básicas viáveis é igual a $\binom{n}{n-m}$. Dessa forma, se aumentamos em cada pivot o valor da função objetivo, o método termina em no máximo $\binom{n}{n-m}$ passos.

Exemplo 2.1 (Solução do problema do Ildo)
Exemplo da solução do problema do Ildo na página 9.

$$\begin{array}{rcrr}
 & & & \downarrow \\
z = & 0/1 & +1/5c & +1/2s \\
w_1 = & 150/1 & -1/1c & -3/2s \\
w_2 = & 6000/1 & -50/1c & -50/1s \\
w_3 = & 80/1 & -1/1c & 0/1s \\
\leftarrow w_4 = & 60/1 & 0/1c & -1/1s
\end{array}$$

Pivô s–w_4

$$\begin{array}{rcrr}
 & & \downarrow & \\
z = & 30/1 & +1/5c & -1/2w_4 \\
\leftarrow w_1 = & 60/1 & -1/1c & +3/2w_4 \\
w_2 = & 3000/1 & -50/1c & +50/1w_4 \\
w_3 = & 80/1 & -1/1c & 0/1w_4 \\
s = & 60/1 & 0/1c & -1/1w_4
\end{array}$$

Pivô c–w_1

$$\begin{array}{rcrr}
z = & 42/1 & -1/5w_1 & -1/5w_4 \\
c = & 60/1 & -1/1w_1 & +3/2w_4 \\
w_2 = & 0/1 & +50/1w_1 & -25/1w_4 \\
w_3 = & 20/1 & +1/1w_1 & -3/2w_4 \\
s = & 60/1 & 0/1w_1 & -1/1w_4
\end{array}$$

Lucro total de R$ 42, com os seguintes valores de variáveis: $c = 60$, $s = 60$, $w_1 = 0$, $w_2 = 0$, $w_3 = 20$ e $w_4 = 0$.
Interpretação das variáveis:

- c: Número de Croissants produzidos.
- s: Número de Strudels produzidos.
- w_1: Número de ovos sobrando: 0.
- w_2: Quantidade de açúcar sobrando: 0 g.
- w_3: Croissants não produzidos (abaixo da demanda): 20.
- w_4: Strudels não produzidos: 0.

$\Diamond$

2.2 O método resumido

Considerando n variáveis e m restrições:

Sistema inicial

$$\begin{aligned} \textbf{maximiza} \quad & z = \sum_{1 \le j \le n} c_j x_j & \\ \textbf{sujeito a} \quad & \sum_{1 \le j \le n} a_{ij} x_j \le b_i & 1 \le i \le m \\ & x_j \ge 0 & 1 \le j \le n \end{aligned}$$

Preparação

Introduzimos variáveis de folga

$$\sum_{1 \le j \le n} a_{ij} x_j + x_{n+i} = b_i \qquad 1 \le i \le m$$

e escrevemos as variáveis de folga como dependentes das variáveis restantes

$$x_{n+i} = b_i - \sum_{1 \le j \le n} a_{ij} x_j \qquad 1 \le i \le m$$

Solução básica viável inicial

Se todos $b_i \geq 0$ (o caso contrário vamos tratar na próxima seção), temos uma solução básica inicial

$$\begin{aligned} x_{n+i} &= b_i \qquad 1 \leq i \leq m \\ x_j &= 0 \qquad 1 \leq j \leq n \end{aligned}$$

Índices das variáveis

Depois do primeiro passo, os conjuntos de variáveis básicas e não-básicas mudam. Seja $\mathcal{B}$ o conjunto dos índices das variáveis básicas (dependentes, em geral não-nulas) e $\mathcal{N}$ o conjunto das variáveis independentes (nulas). No começo temos

$$\mathcal{B} = \{n+1, n+2, \ldots, n+m\}; \qquad \mathcal{N} = \{1, 2, \ldots, n\}$$

A forma geral do sistema muda para

$$\begin{aligned} z &= \bar{z} + \sum_{j \in \mathcal{N}} \bar{c}_j x_j \\ x_i &= \bar{b}_i - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j \qquad i \in \mathcal{B} \end{aligned}$$

As barras em cima dos coeficientes enfatizam que eles mudam ao longo da aplicação do método.

Escolher variável entrante

Em cada passo do método Simplex, escolhemos uma variável não-básica x_k, com $k \in \mathcal{N}$ para aumentar o valor objetivo z. Isso somente é possível para os índices j tal que $\bar{c}_j > 0$, i.e.

$$\{j \in \mathcal{N} \mid \bar{c}_j > 0\}.$$

Escolhemos um k desse conjunto, e x_k é a variável entrante. Uma heurística simples é a *regra do maior coeficiente*, que escolhe

$$k = \operatorname{argmax}\{\bar{c}_j \mid \bar{c}_j > 0, j \in \mathcal{N}\}$$

Aumentar a variável entrante

Seja x_k a variável entrante. Se aumentamos x_k para um valor positivo, as variáveis básicas têm novos valores

$$x_i = \bar{b}_i - \bar{a}_{ik} x_k \qquad i \in \mathcal{B}.$$

Temos que respeitar $x_i \geq 0$ para $1 \leq i \leq n$. Cada equação com $\bar{a}_{ik} > 0$ fornece uma cota superior para x_k:

$$x_k \leq \bar{b}_i / \bar{a}_{ik}.$$

Logo podemos aumentar x_k ao máximo um valor

$$\alpha := \min_{\substack{i \in \mathcal{B} \\ \bar{a}_{ik} > 0}} \frac{\bar{b}_i}{\bar{a}_{ik}} > 0.$$

Podemos escolher a variável sainte entre os índices

$$\{i \in \mathcal{B} \mid \bar{b}_i / \bar{a}_{ik} = \alpha\}.$$

2.3 Interpretação geométrica

2.4 Sistemas ilimitados

Como pivotar?

- Considere o sistema

z	$= 24$	$-x_1$	$+2x_2$
x_3	$= 2$	$-x_1$	$+x_2$
x_4	$= 5$	$+x_1$	$+4x_2$

- Qual a próxima solução básica viável?

- A duas equações não restringem o aumento de x_2: existem soluções com valor ilimitado.

2.5 Encontrar uma solução inicial

Solução básica inicial

- Nosso problema inicial é

$$\begin{aligned} \textbf{maximiza} \quad & z = \sum_{1 \le j \le n} c_j x_j \\ \textbf{sujeito a} \quad & \sum_{1 \le j \le n} a_{ij} x_j \le b_i \qquad 1 \le i \le m \\ & x_i \ge 0 \qquad 1 \le i \le m \end{aligned}$$

- com dicionário inicial

$$\begin{aligned} z &= \sum_{1 \le j \le n} c_j x_j \\ x_{n+i} &= b_i - \sum_{1 \le j \le n} a_{ij} x_j \qquad 1 \le i \le m \end{aligned}$$

Solução básica inicial

- A solução básica inicial desse dicionário é

$$x = (0 \cdots 0\, b_1 \cdots b_m)^t$$

- O que acontece se existe um $b_i < 0$?
- A solução básica não é mais viável! Sabe-se disso porque pelo menos uma variável básica terá valor negativo.

Sistema auxiliar

- Um método para resolver o problema: resolver outro programa linear
  - se o sistema original possui alguma solução viável, então garantidamente a solução do sistema auxiliar será básica viável para o sistema linear original, tal que podemos aplicar o método Simplex nesta solução

$$\begin{aligned} \textbf{maximiza} \quad & z = -x_0 \\ \textbf{sujeito a} \quad & \sum_{1 \le j \le n} a_{ij} x_j - x_0 \le b_i \qquad 0 \le i \le m \\ & x_i \ge 0 \qquad 0 \le i \le n \end{aligned}$$

Resolver o sistema auxiliar

- É fácil achar uma solução viável do sistema auxiliar: se o sistema original possui solução, a solução do método auxiliar se torna viável garantidamente depois do primeiro pivô.
- Escolha $x_i = 0$, para todos $1 \leq i \leq n$.
- Escolha x_0 suficientemente grande: $x_0 \geq \max_{1 \leq i \leq m} -b_i$.
- Isso corresponde a:
  - uma solução inicial não-viável $x_0 = x_1 = \ldots = x_n = 0$.
  - x_0 entra na base.
  - sai da base a variável com valor de b_i mais negativo.
- deve-se resolver o sistema auxiliar
- se o sistema auxiliar não possuir solução factível, ou sua solução ótima gera um valor $x_0 > 0$, então o sistema original é infactível (sem nenhuma solução factível)

Exemplo: Problema original

$$\begin{aligned} \textbf{maximiza} \quad & z = -2x_1 - x_2 \\ \textbf{sujeito a} \quad & -x_1 + x_2 \leq -1 \\ & -x_1 - 2x_2 \leq -2 \\ & x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Exemplo: Problema auxiliar

$$\begin{aligned} \textbf{maximiza} \quad & z = -x_0 \\ \textbf{sujeito a} \quad & -x_1 + x_2 - x_0 \leq -1 \\ & -x_1 - 2x_2 - x_0 \leq -2 \\ & x_2 - x_0 \leq 1 \\ & x_0, x_1, x_2 \geq 0 \end{aligned}$$

Exemplo: Dicionário inicial do problema auxiliar

$$\begin{array}{rcccc} & & & & \downarrow \\ z & = & & & -x_0 \\ \hline w_1 & = -1 & +x_1 & -x_2 & +x_0 \\ \leftarrow w_2 & = -2 & +x_1 & +2x_2 & +x_0 \\ w_3 & = 1 & & -x_2 & +x_0 \end{array}$$

- Observe que a solução básica não é viável.
- Para achar uma solução básica viável: fazemos um primeiro pivot com variável entrante x_0 e variável sainte w_2.

Exemplo: Dicionário inicial viável do sistema auxiliar

$$\begin{array}{rcccc} & & & \downarrow & \\ z & = -2 & +x_1 & +2x_2 & -w_2 \\ \hline \leftarrow w_1 & = 1 & & -3x_2 & +w_2 \\ x_0 & = 2 & -x_1 & -2x_2 & +w_2 \\ w_3 & = 3 & -x_1 & -3x_2 & +w_2 \end{array}$$

Primeiro pivot

$$\begin{array}{rcccc} & & \downarrow & & \\ z & = -4/3 & +x_1 & -2/3w_1 & -1/3w_2 \\ \hline x_2 & = 1/3 & & -1/3w_1 & +1/3w_2 \\ \leftarrow x_0 & = 4/3 & -x_1 & +2/3w_1 & +1/3w_2 \\ w_3 & = 2 & -x_1 & +w_1 & \end{array}$$

Segundo pivot

$$\begin{array}{rcccc} z & = 0 & -x_0 & & \\ \hline x_2 & = 1/3 & & -1/3w_1 & +1/3w_2 \\ x_1 & = 4/3 & -x_0 & +2/3w_1 & +1/3w_2 \\ w_3 & = 2/3 & +x_0 & +1/3w_1 & -1/3w_2 \end{array}$$

Solução ótima!

Solução do sistema auxiliar

- O que vale a solução do sistema auxiliar?
- Obviamente, se o sistema original tem solução, o sistema auxiliar também tem uma solução com $x_0 = 0$.
- Logo, após aplicar o método Simplex ao sistema auxiliar, temos os casos
  - $x_0 > 0$: O sistema original não tem solução.
  - $x_0 = 0$: O sistema original tem solução. Podemos descartar x_0 e continuar resolvendo o sistema original com a solução básica viável obtida.
- A solução do sistema auxiliar se chama *fase I*, a solução do sistema original *fase II*.

Sistema original

Reescreve-se a função objetivo original substituindo as variáveis básicas do sistema original pelas equações correspondentes do sistema auxiliar, de forma que a função objetivo z não contenha variáveis básicas. No exemplo, a função objetivo é rescrita como:

$$z = -2x_1 - x_2 = -3 - w_1 - w_2.$$

z	$= -3$	$-w_1$	$-w_2$
x_2	$= 1/3$	$-1/3w_1$	$+1/3w_2$
x_1	$= 4/3$	$+2/3w_1$	$+1/3w_2$
w_3	$= 2/3$	$+1/3w_1$	$-1/3w_2$

Nesse exemplo, o dicionário original já é ótimo!

2.6 Soluções degeneradas

Solução degenerada

- Um dicionário é *degenerado* se existe pelo menos um $\bar{b}_i = 0$.
- Qual o problema?
- Pode acontecer um pivot que não aumenta a variável entrante, e portanto não aumenta o valor da função objetivo.

Exemplo 1

- Nem sempre é um problema.

$$\begin{array}{rlrr} & & \downarrow & \\ z & = 5 & +x_3 & -x_1 \\ \hline \leftarrow x_2 & = 5 & -2x_3 & -3x_1 \\ x_1 & = 7 & & -4x_1 \\ w_3 & = & & +x_1 \end{array}$$

- x_2 é a variável sainte e o valor da função objetivo aumenta.

Exemplo 2

$$\begin{array}{rlrrr} & & & \downarrow & \\ z & = 3 & -1/2x_1 & +2x_2 & -3/2x_3 \\ \hline w_1 & = 1 & +1/2x_1 & & -1/2x_3 \\ \leftarrow w_2 & = & x_1 & -x_2 & +x_3 \end{array}$$

- Se a variável sainte é determinada pela equação com $\bar{b}_i = 0$, temos um *pivot degenerado.*

- Nesse caso, a variável entrante não aumenta: temos a mesma solução depois do pivot!

Exemplo 2: Primeiro pivot

$$\begin{array}{rlrrr} & & \downarrow & & \\ z & = 3 & +3/2x_1 & -2w_2 & +1/2x_3 \\ \hline \leftarrow w_1 & = 1 & -1/2x_1 & & -1/2x_3 \\ x_2 & = & x_1 & -w_2 & +x_3 \end{array}$$

- O valor da função objetivo não aumentou!

Exemplo 2: Segundo pivot

$$\begin{array}{rlrrr} z & = 6 & -3w_1 & -2w_2 & -x_3 \\ \hline x_1 & = 2 & -2w_1 & & -x_3 \\ x_2 & = 2 & -2w_1 & -w_2 & \end{array}$$

- A segunda iteração aumentou o valor da função objetivo!

Ciclos

- O pior caso seria, se entramos em ciclos.
- É possível? Depende da regra de seleção de variáveis entrantes e saintes.
- Nossas regras
  - Escolha a variável entrante com o maior coeficiente.
  - Escolha a variável sainte que restringe mais.
  - Em caso de empate, escolha a variável com o menor índice.
- Ciclos são possíveis: O seguinte sistema possui um ciclo de 6 pivôs: x_1–w_1, x_2–w_2, x_3–x_1, x_4–x_2, w_1–x_3, w_2–x_4.

z	$=$	$10x_1$	$-57x_2$	$-9x_3$	$-24x_4$
w_1	$=$	$-1/2x_1$	$+11/2x_2$	$+5/2x_3$	$-9x_4$
w_2	$=$	$-1/2x_1$	$+3/2x_2$	$+1/2x_3$	$-x_4$
w_3	$=1$	$-x_1$			

Soluções do problema

- Como resolver o problema?
- Duas propostas
  - Método lexicográfico.
  - Regra de Bland.

Método lexicográfico

- Idéia: O fato que existe um $\bar{b}_i = 0$ é por ocaso.
- Se introduzimos uma pequena perturbação $\epsilon \ll 1$
  - o problema desaparece
  - a solução será (praticamente) a mesma.

Método lexicográfico

- Ainda é possível que duas perturbações numéricas se cancelem.
- Para evitar isso: Trabalha-se simbolicamente.
- Introduzimos perturbações simbólicas

$$0 < \epsilon_1 \ll \epsilon_2 \ll \cdots \ll \epsilon_m$$

em cada equação.

- Característica: Todo ϵ_i é numa escala diferente dos outros tal que eles nunca se ancelem.

Exemplo

Sistema original degenerado e sistema perturbado

z	$= 4$	$+2x_1$	$-x_2$
$1 - 4w_1$	$= 1/2$		$-x_2$
w_2	$=$	$-2x_1$	$+4x_2$
w_3	$=$	x_1	$-3x_2$

z	$= 4$					$+2x_1$	$-x_2$
w_1	$= 1/2$	$+\epsilon_1$					$-x_2$
w_2	$=$		ϵ_2			$-2x_1$	$+4x_2$
w_3	$=$			ϵ_3		$+x_1$	$-3x_2$

Características

- Depois de chegar no valor ótimo, podemos retirar as perturbações ϵ_i.

Teorema 2.1
O método Simplex sempre termina escolhendo as variáveis saintes usando a regra lexicográfica.

Prova. É suficiente mostrar que o sistema nunca vai ser degenerado: assim o valor da função objetivo sempre cresce, e o método Simplex não entra em ciclo. A matriz de perturbações

$$\begin{pmatrix} \epsilon_1 & & & \\ & \epsilon_2 & & \\ & & \cdots & \\ & & & \epsilon_m \end{pmatrix}$$

inicialmente tem posto m (*rank* de uma matriz). As operações do método Simplex são operações lineares que não mudam o posto do matriz. Logo, em cada passo do método Simplex temos uma matriz de perturbações

$$\begin{pmatrix} e_{11}\epsilon_1 & e_{12} & \cdots & e_{1m}\epsilon_m \\ e_{21}\epsilon_1 & e_{22}\epsilon_2 & \cdots & e_{2m}\epsilon_m \\ \cdots & & \cdots & \\ e_{m1}\epsilon_1 & e_{m2}\epsilon_2 & \cdots & e_{mm}\epsilon_m \end{pmatrix}$$

que ainda tem posto m. Portanto, em cada linha i existe ao menos um $e_{ij} \neq 0$ e assim uma perturbação diferente de zero. Logo, o sistema não é degenerado. ■

Regra de Bland

- Outra solução do problema: A regra de Bland.
- Escolhe como variável entrante e sainte sempre a variável com o menor índice (caso tiver mais que um candidato).

Teorema 2.2
O método Simplex sempre termina se as variáveis entrantes e saintes são escolhidas através da regra de Bland.

Prova. Prova por contradição: Suponha que exista uma sequência de dicionários num ciclo $D_0, D_1, \ldots, D_{k-1}$ usando a regra do Bland. Nesse ciclo algumas variáveis, chamadas *inconstantes*, entram e saem novamente da base, outras permanecem sempre como básicas, ou como não-básicas. Seja x_t a variável inconstante com o maior índice. Sem perda de generalidade, seja x_t a variável sainte do primeiro dicionário D_0. Seja x_s a variável entrante no D_0. Observe que x_s também é inconstante e portanto $s < t$. Seja D^* o dicionário em que x_t entra na base. Temos a seguinte situação

$$\begin{array}{ccccccccc} x_s \text{ entra} & & & & & & x_t \text{ entra} & & \\ \downarrow & & & & & & \downarrow & & \\ D_0, & D_1, & D_2, & \cdots & & & D^*, & \cdots & D_{k-1} \\ \downarrow & & & & & & & & \\ x_t \text{ sai} & & & & & & & & \end{array}$$

com os sistemas correspondentes

$$
\begin{aligned}
&D_0: && D^*: \\
&z = z_0 + \sum_{j\in\mathcal{N}} c_j x_j && z = z^* + \sum_{j\in\mathcal{N}^*} c_j^* x_j \\
&x_i = b_i - \sum_{j\in\mathcal{N}} a_{ij} x_j \quad i \in \mathcal{B} && x_i = b_i^* - \sum_{j\in\mathcal{N}^*} a_{ij}^* x_j \quad i \in \mathcal{B}^*
\end{aligned}
$$

Como temos um ciclo, todas variáveis inconstantes são nulas e o valor da função objetivo é constante. Logo $z_0 = z^*$. Se aumentamos o valor do x_s, qual seria o novo valor da função objetivo?

$$
\begin{aligned}
&x_s = y \\
&x_j = 0 \qquad j \in \mathcal{N} \setminus \{s\} \\
&x_i = b_i - a_{is} y \qquad i \in \mathcal{B}
\end{aligned}
$$

Logo temos o novo valor

$$z = z_0 + c_s y \tag{2.4}$$

(no sistema D_1) mas também

$$z = z^* + \sum_{j\in\mathcal{N}^*} c_j^* x_j = z_0 + \sum_{j\in\mathcal{N}^*} c_j^* x_j.$$

Agora, podemos substituir as variáveis com índices em $\mathcal{N}^* \cap \mathcal{B}$ nessa equação. Para facilitar a notação, vamos definir $c_j^* := 0$ para $j \notin \mathcal{N}^*$, que permite substituir todas variáveis $x_j, j \in \mathcal{B}$ e assim obtemos

$$z = z_0 + \sum_{j\in[1,n+m]} c_j^* x_j = z_0 + c_s^* y + \sum_{j\in\mathcal{B}} c_j^* (b_j - a_{js} y). \tag{2.5}$$

Equações 2.4 e 2.5 representam o mesmo valor, portanto

$$\left(c_s - c_s^* + \sum_{j\in\mathcal{B}} c_j^* a_{js} \right) \cdot y = \sum_{j\in\mathcal{B}} c_j^* b_j$$

e como essa igualdade deve ser correta para qualquer aumento y

$$c_s - c_s^* + \sum_{j\in\mathcal{B}} c_j^* a_{js} = 0.$$

Como x_s entra em D_0 temos $c_s > 0$. Também, como $s < t$, mas x_t entra em D^*, $c^*_s \leq 0$. Logo,

$$\sum_{j \in \mathcal{B}} c^*_j a_{js} < 0$$

e deve existir um $r \in \mathcal{B}$ tal que $c^*_r a_{rs} < 0$. Isso tem uma série de consequências:

1. $c^*_r \neq 0$.
2. $r \in \mathcal{N}^*$, porque somente as variáveis não-básicas tem c^*_j em D^*.
3. x_r é inconstante, porque ela é básica em D_0, mas não-básica em D^*.
4. $r \leq t$, porque t foi a variável inconstante com o maior índice.
5. $r < t$, porque $c^*_t a_{ts} > 0$: x_t entra em D^*, logo $c^*_t > 0$, e x_t sai em D_0, logo $a_{ts} > 0$.
6. $c^*_r \leq 0$, senão r e não t entraria em D^* seguindo a regra de Bland.
7. $a_{rs} > 0$.
8. $b_r = 0$, porque x_r é inconstante, mas todos variáveis inconstantes são nulas no ciclo, e x_r é básica em D_0.

Os últimos dois itens mostram que x_r foi candidato ao sair em D_0 com índice $r < t$, uma contradição à regra de Bland. ■

Teorema fundamental

Teorema 2.3 (Teorema fundamental da programação linear)
Para qualquer programa linear temos:

1. Se não existe solução ótima, o problema é inviável ou ilimitado.
2. Se existe uma solução viável, existe uma solução básica viável.
3. Se existe uma solução ótima, existe uma solução ótima básica.

2.7 Complexidade do método Simplex

Complexidade pessimista

- Com a regra de Bland o método Simplex sempre termina.

- Com $n+m$ variáveis (de decisão e de folga) existem

$$\binom{n+m}{n} = \binom{n+m}{m}$$

soluções básicas possíveis.

- Logo: No pior caso o método Simplex termina depois desse número de pivots.

Complexidade pessimista

Se as restrições forem todas linearmente independente, e tivermos mais restrições que variáveis, então a resolução do sistema é trivial. O pior caso acontece quando o número de restrições é igual ao número de variáveis básicas.

- Com as equações linearmente independentes, temos

$$\binom{n+m}{m} \leq \binom{2n}{n}$$

- e os limites (exercício 5.11)

$$\frac{1}{2n}2^{2n} \leq \binom{2n}{n} \leq 2^{2n}.$$

- Logo, o número de passo no pior caso é *exponencial* no tamanho da entrada.

3 Dualidade

3.1 Introdução

Visão global

- *Dualidade*: Cada programa linear tem um programa linear associado, chamado de *dual*.
- Para enfatizar a diferença, o programa linear original chama-se de *primal*.
- Programas lineares duais tem várias aplicações como
  - Estimar a convergência.
  - Análise de sensibilidade e re-otimização de sistemas.
  - Solução mais simples ou eficiente com o Método Simplex dual.
- O programa dual também tem uma interpretação relevante.

Introdução

- Considere o programa linear

$$\begin{aligned} \textbf{maximiza} \quad & z = 4x_1 + x_2 + 3x_3 \\ \textbf{sujeito a} \quad & x_1 + 4x_2 \leq 1 \\ & 3x_1 - x_2 + x_3 \leq 3 \\ x_1, x_2, x_3 \geq 0 & \end{aligned}$$

- Cada solução viável fornece um limite inferior para o valor máximo.

$$\begin{aligned} x_1 = x_2 = x_3 = 0 &\Rightarrow z = 0 \\ x_1 = 3, x_2 = x_3 = 0 &\Rightarrow z = 4 \end{aligned}$$

- Qual a qualidade da solução atual?
- Não sabemos, sem limite superior.

Limites superiores

- Como obter um limite superior? Observe: $z = 4x_1 + x_2 + 3x + 3 \leq 10x_1 + x_2 + 3x_3 \leq 10$
- Podemos construir uma combinação linear das desigualdades, tal que o coeficiente de cada x_j ultrapasse o coeficiente da função objetivo.
- Nosso exemplo:

$$\begin{aligned} & (x_1 + 4x_2) + 3(3x_1 - x_2 + x_3) \leq 1 + 3 \cdot 3 = 10 \\ \Longleftrightarrow & 10x_1 + x_2 + 3x_3 \leq 10 \end{aligned}$$

- Como obter um limite superior para a função objetivo?
- Qual seria o menor limite superior que esse método fornece?

O menor limite superior

- Sejam $y_1, \ldots, y_n$ os coeficientes de cada linha. Observação: Eles devem ser ≥ 0 para manter a direção das desigualdades.
- Então queremos

$$\begin{aligned} \textbf{minimiza} \quad & \sum_i b_i y_i \\ \textbf{sujeito a} \quad & \sum_i a_{ij} y_i \geq c_j \qquad 1 \leq j \leq n \\ & y_i \geq 0 \end{aligned}$$

- Isto é o *problema dual*.

Dualidade: Características

- Em notação matricial

$$\begin{array}{rlrl} \textbf{maximiza} & c^t x & \textbf{minimiza} & b^t y \\ \textbf{sujeito a} & Ax \leq b & \textbf{sujeito a} & y^t A \geq c^t \\ & x \geq 0 & & y \geq 0 \end{array}$$

- O primeiro se chama *primal* e o segundo *dual*.
- Eles usam os mesmos parâmetros c_j, a_{ij}, b_i.

O dual do dual

- Observação: O dual do dual é o primal.
- Forma normal do dual:

$$\begin{array}{rl} \textbf{maximiza} & -b^t y \\ \textbf{sujeito a} & -y^t A \leq -c^t \\ & y \geq 0 \end{array} \quad = \quad \begin{array}{rl} \textbf{maximiza} & -b^t y \\ \textbf{sujeito a} & (-A^t)y \leq -c \\ & y \geq 0 \end{array}$$

- Dual do dual

$$\begin{array}{rl} \textbf{minimiza} & -c^t x \\ \textbf{sujeito a} & x^t(-A^t) \geq -b^t \\ & x \geq 0 \end{array} \quad = \quad \begin{array}{rl} \textbf{maximiza} & c^t x \\ \textbf{sujeito a} & Ax \leq b \\ & x \geq 0 \end{array}$$

3.2 Interpretação do dual

Exemplo: Dieta dual

- Problema da dieta: Minimiza custos de uma dieta x que alcance dados VDR mínimos.

$$\begin{array}{rl} \textbf{minimiza} & c^t x \\ \textbf{sujeito a} & Ax \geq r \\ & x \geq 0 \end{array}$$

- Unidades das variáveis e parâmetros
  - x: Quantidade do alimento $[g]$
  - c: R\$/alimento $[R\$/g]$
  - a_{ij}: Nutriente/Alimento $[g/g]$
  - r: Quantidade de nutriente $[g]$.

Exemplo: Dieta dual

- O problema dual é

$$\begin{array}{rl} \textbf{maximiza} & y^t r \\ \textbf{sujeito a} & y^t A \leq c^t \\ & y \geq 0 \end{array}$$

- Qual a unidade de y? Preço por nutriente $[R\$/g]$.
- Imagine uma empresa, que produz cápsulas que substituem os nutrientes.
- Para vender no mercado, a empresa tem que garantir que uma dieta baseado em cápsulas custa menos que os alimentos correspondentes:

$$\sum_i y_i a_{ij} \leq c_j$$

- Além disso, ela define preços por nutriente que maximizam o custo de uma dieta adequada, para maximizar o próprio lucro.

$$\textbf{maximiza} \quad y^t r$$

Interpretação do dual

- Outra interpretação: o valor de uma variável dual y_j é o *lucro marginal* de adicionar mais uma unidade b_j.

  Teorema 3.1
  Se um sistema tem ao menos uma solução básica viável não-degenerada, existe um ϵ tal que, se $|t_j| \leq \epsilon$ para $1 \leq j \leq m$,

$$\begin{aligned} \textbf{maximiza} \quad & c^t x \\ \textbf{sujeito a} \quad & Ax \leq b + t \\ & x \geq 0 \end{aligned}$$

  tem uma solução ótima com valor

$$z = z^* + {y^*}^t t$$

  (com z^* o valor ótimo do primal, é y^* a solução ótima do dual).

3.3 Características

Teorema da dualidade fraca

Teorema 3.2 (Dualidade fraca)
Se $x_1, \ldots, x_n$ é uma solução viável do sistema primal, e $y_1, \ldots, y_m$ uma solução viável do sistema dual, então

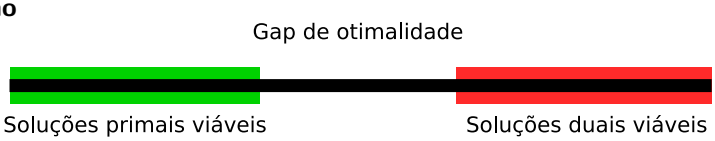
$$\sum_{1 \leq i \leq n} c_i x_i \leq \sum_{1 \leq j \leq m} b_j y_j.$$

Prova.

$$
\begin{aligned}
& c^t x && && (3.1)\\
\leq & (y^t A)x = y^t(Ax) && \text{pela restrição dual} && (3.2)\\
\leq & y^t b && \text{pela restrição primal} && (3.3)
\end{aligned}
$$

■

Situação



- Em aberto: Qual o tamanho desse intervalo em geral?

Teorema da da dualidade forte

Teorema 3.3
Se $x_1^*, \ldots, x_n^*$ é uma solução ótima do sistema primal, existe uma solução ótima $y_1^*, \ldots, y_m^*$ do sistema dual, e

$$\sum_{1 \leq i \leq n} c_i x_i^* = \sum_{1 \leq j \leq m} b_j y_j^*.$$

Prova. Seja x^* uma solução ótima do sistema primal, que obtemos pelo método Simplex. No início introduzimos variáveis de folga

$$x_{n+j} = b_j - \sum_{1 \leq i \leq n} a_{ji} x_i \qquad 1 \leq j \leq m$$

e a função objetivo final é

$$z = z^* + \sum_{1 \leq i \leq n+m} \bar{c}_i x_i$$

(supondo que $\bar{c}_i = 0$ para variáveis básicas). Temos que construir uma solução ótima dual y^*. Pela optimalidade, na função objetivo acima, todos $\overline{c_i}$ devem ser não-positivos. Afirmamos que $y_j^* = -\overline{c_{n+j}} \geq 0$ para $j \in [1, m]$ é uma solução dual ótima. Como z^* o valor ótimo do problema inicial, temos $z^* = \sum_{1 \leq i \leq n} c_i x_i^*$.

Reescrevendo a função objetivo temos

$$
\begin{aligned}
z & \\
&= \sum_{1\le i\le n} c_i x_i && \text{sistema inicial}\\
&= z^* + \sum_{1\le i\le n+m} \bar{c_i} x_i && \text{sistema final}\\
&= z^* + \sum_{1\le i\le n} \bar{c_i} x_i + \sum_{1\le j\le m} \overline{c_{n+j}} x_{n+j} && \text{separando índices}\\
&= z^* + \sum_{1\le i\le n} \bar{c_i} x_i - \sum_{1\le j\le m} y_j^* \left(b_j - \sum_{1\le i\le n} a_{ji} x_i \right) && \text{subst. solução e var. folga}\\
&= \left(z^* - \sum_{1\le j\le m} y_j^* b_j \right) + \sum_{1\le i\le n} \left(\bar{c_i} + \sum_{1\le j\le m} y_j^* a_{ji} \right) x_i && \text{agrupando}
\end{aligned}
$$

Essa derivação está válida para x_i qualquer, porque são duas expressões para a mesma função objetivo, portanto

$$z^* = \sum_{1\le j\le m} y_j^* b_j \quad \text{e} \quad c_i = \bar{c_i} + \sum_{1\le j\le m} y_j^* a_{ji} \qquad 1 \le i \le n.$$

Com isso sabemos que o primal e dual possuem o mesmo valor

$$\sum_{1\le j\le m} y_j^* b_j = z^* = \sum_{1\le i\le n} c_i x_i^*$$

e como $\bar{c_i} \le 0$ sabemos que a solução y^* satisfaz a restrições duais

$$
\begin{aligned}
c_i &\le \sum_{1\le j\le m} y_j^* a_{ji} \qquad 1 \le i \le n\\
y_i^* &\ge 0 \qquad 1 \le i \le m
\end{aligned}
$$

■

Consequências: Soluções primais e duais

- Com o teorema da dualidade forte, temos quatro possibilidades

Sistema primal	Sistema dual	Intervalo
Ótima	Ótima	Sem
Ilimitado	Inviável	Sem
Inviável	Ilimitado	Sem
Inviável	Inviável	Infinito

Exemplo 3.1
Pelo teorema da dualidade forte, não podemos concluir, que existe um caso que tanto o sistema primal qunato o sistema dual são inviáveis. O seguinte exemplo mostra que isso pode realmente acontecer. O sistema primal

$$\begin{aligned} \textbf{maximiza} \quad & x_1 \\ \textbf{sujeito a} \quad & +x_1 - x_2 \leq 0 \\ & -x_1 + x_2 \leq -1 \end{aligned}$$

possui sistema dual correspondete

$$\begin{aligned} \textbf{minimiza} \quad & -y_2 \\ \textbf{sujeito a} \quad & +y_1 - y_2 \geq 1 \\ & -y_1 + y_2 \geq 0 \end{aligned}$$

Os dois sistemas são inviáveis. ◊

Consequências

- Dado soluções primais e duais x^*, y^* tal que $c^t x^* = b^t y^*$ podemos concluir que ambas soluções são ótimas (x^*, y^* é um *certificado* da optimalidade)[1].
- A prova mostra: com o valor ótimo do sistema primal, sabemos também o valor ótima do sistema dual.
- Além disso: Podemos trocar livremente entre o sistema primal e dual. ⇒ Método Simplex dual.

Outra consequência do Teorema da dualidade forte é o

Teorema 3.4 (Teorema das folgas complementares)
Se x^*, y^* são soluções ótimas do sistema primal e dual, respectivamente, temos

$$y^{*t}(b - Ax) = 0 \tag{3.4}$$
$$(y^{*t}A - c^t)x^* = 0 \tag{3.5}$$

[1] Uma consequencia é que o problema de decisão correspondente, determinar se existe uma solução maior que um dado valor, possui um certificado que pode ser verificado em tempo polinomial tanto para uma resposta positiva quanto uma resposta negativa. Portanto, já antes da descoberta de um algoritmo polinomial para esse problema, foi claro que ele pertence a NP ∩ co-NP.

Prova. Pelo Teorema da dualidade forte as duas desigualdades 3.2 e 3.3 da prova do Teorema da dualidade fraca se tornam igualdades para soluções ótimas:

$$c^t x^* = y^{*t} A x^* = y^{*t} b$$

Reagrupando termos, o teorema segue. ■

As igualdades 3.4 e 3.5 são ainda válidas em cada componente, porque tanto as soluções ótimas x^*, y^* quanto as folgas primas e duais $b - Ax$ e $y^{*t}A - c^t$ sempre são positivos.

$$x_i > 0 \Rightarrow \sum_{1 \leq j \leq m} y_j a_{ji} = c_i \tag{3.6}$$

$$\sum_{1 \leq j \leq m} y_j a_{ji} > c_i \Rightarrow x_i = 0 \tag{3.7}$$

$$y_j > 0 \Rightarrow b_j = \sum_{1 \leq i \leq n} a_{ji} x_i \tag{3.8}$$

$$b_j > \sum_{1 \leq i \leq n} a_{ji} x_i \Rightarrow y_j = 0 \tag{3.9}$$

Como consequência, podemos ver que, por exemplo, caso uma igualdade primal não possui folga, a variável dual correspondente é positiva, e, contrariamente, caso uma igualdade primal possui folga, a variável dual correspondente é zero. As mesmas relações se aplicam para as desigualdades no sistema dual. Após a introdução da forma matricial no seção 3.6 vamos analisar a interpretação das variáveis duais com mais detalha no seção 3.7.

3.4 Método Simplex dual

Método Simplex dual

- Considere

$$\begin{aligned} \textbf{maximiza} \quad & -x_1 - x_2 \\ \textbf{sujeito a} \quad & -2x_1 - x_2 \leq 4 \\ & -2x_1 + 4x_2 \leq -8 \\ & -x_1 + 3x_2 \leq -7 \\ & x_1, x_2 \geq 0 \end{aligned}$$

- Qual o dual?

$$\begin{aligned} \textbf{minimiza} \quad & 4y_1 - 8y_2 - 7y_3 \\ \textbf{sujeito a} \quad & -2y_1 - 2y_2 - y_3 \geq -1 \\ & -y_1 + 4y_2 + 3y_2 \geq -1 \end{aligned}$$

Com dicionários

z	$=$	$-x_1$	$-x_2$	
w_1	$=4$	$+2x_1$	$+x_2$	
w_2	$=-8$	$+2x_1$	$-4x_2$	
w_3	$=-7$	$+x_1$	$-3x_2$	
$-w$	$=$	$-4y_1$	$+8y_2$	$+7y_3$
z_1	$=1$	$-2y_1$	$-2\mathbf{y_2}$	$-y_3$
z_2	$=1$	$-y_1$	$+4y_2$	$+3y_3$

- Observação: O primal não é viável, mas o dual é!
- Correspondência das variáveis:

	Variáveis	
	principais	de folga
Primal	$x_1, \ldots, x_n$	$w_1, \ldots, w_m$
Dual	$z_1, \ldots, z_n$,	$y_1, \ldots, y_m$
	de folga	principais

- Primeiro pivot: y_2 entra, z_1 sai. No primal: w_2 sai, x_1 entra.

Primeiro pivot

z	$=-4$	$-0.5w_2$	$-3x_2$	
w_1	$=12$	$+w_2$	$+5x_2$	
x_1	$=4$	$+0.5w_2$	$+2x_2$	
w_3	$=-3$	$+0.5w_2$	$-x_2$	
$-w$	$=4$	$-12y_1$	$-4z_1$	$+3y_3$
y_2	$=0.5$	$-y_1$	$-0.5z_1$	$-0.5\mathbf{y_3}$
z_2	$=3$	$-5y_1$	$-2z_1$	$+y_3$

- Segundo pivot: y_3 entra, y_2 sai. No primal: w_3 sai, w_2 entra.

Segundo pivot

z	$=-7$	$-w_3$	$-4x_2$
w_1	$=18$	$+2w_3$	$+7x_2$
x_1	$=7$	$+w_3$	$+3x_2$
w_2	$=6$	$+2w_3$	$+2x_2$

$$\begin{array}{llll} -w & = 7 & -18y_1 & -7z_1 & -6y_2 \\ \hline y_3 & = 1 & -2y_1 & -z_1 & -2y_2 \\ z_2 & = 4 & -7y_1 & -3z_1 & -2y_2 \end{array}$$

- Sistema dual é ótimo, e portanto o sistema primal também.

Método Simplex dual

- Observação: Não é necessário escrever o sistema dual. Ele é sempre o negativo transposto do sistema primal.

$$z = \bar{z} + \sum_{j \in \mathcal{N}} \bar{c}_j x_j$$
$$x_i = \bar{b}_i - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j \qquad i \in \mathcal{B}$$

- Mas é necessário modificar as regras para resolver o sistema dual.

Método Simplex dual: Viabilidade e otimalidade

- Pré-condição: O dicionário é *dualmente viável*, i.e. os coeficientes das variáveis não-básicas na função objetivo tem quer ser não-positivos.

$$\bar{c}_j \leq 0 \quad \text{para} \quad j \in \mathcal{N}.$$

- Otimalidade: Todos variáveis básicas primais positivas

$$\forall i \in \mathcal{B} : \bar{b}_i \geq 0$$

Método Simplex dual: Pivô

- Caso existe uma variável primal negativa: A solução dual não é ótima.
- Regra do maior coeficiente: A variável básica primal com menor valor (que é negativo) sai da base primal.

$$i = \operatorname*{argmin}_{i \in \mathcal{B}} \bar{b}_i$$

- A variável primal nula com fração $\bar{a}_{ij}/\bar{c}_j$ maior entra.

$$j = \operatorname*{argmin}_{\substack{j \in \mathcal{N} \\ \bar{a}_{ij} < 0}} \frac{\bar{c}_j}{\bar{a}_{ij}} = \operatorname*{argmax}_{\substack{j \in \mathcal{N} \\ \bar{a}_{ij} < 0}} \frac{\bar{a}_{ij}}{\bar{c}_j} = \operatorname*{argmax}_{j \in \mathcal{N}} \frac{\bar{a}_{ij}}{\bar{c}_j}$$

Método Simplex dual
Resumo:

- Dualmente viável: $\bar{c}_j \leq 0$ para $j \in \mathcal{N}$.
- Otimalidade: $\forall i \in \mathcal{B} : \bar{b}_i \geq 0$.
- Variável sainte: $i = \text{argmin}_{i \in \mathcal{B}} \bar{b}_i$
- Variável entrante: $j = \text{argmax}_{j \in \mathcal{N}} \frac{\bar{a}_{ij}}{\bar{c}_j}$.

Exemplo

$$\begin{aligned} \textbf{maximiza} \quad & z = -2x_1 - x_2 \\ \textbf{sujeito a} \quad & -x_1 + x_2 \leq -1 \\ & -x_1 - 2x_2 \leq -2 \\ & x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Exemplo: Dicionário inicial

$$\begin{array}{lrrr} z & = & -2x_1 & -x_2 \\ w_1 & = -1 & +x_1 & -x_2 \\ w_2 & = -2 & +x_1 & +2\mathbf{x_2} \\ w_3 & = 1 & & -x_2 \end{array}$$

- O dicionário primal não é viável, mais o dual é.

Exemplo: Primeiro pivot

$$\begin{array}{lrrr} z & = -1 & -3/2x_1 & -1/2w_2 \\ w_1 & = -2 & +3/2\mathbf{x_1} & -1/2w_2 \\ x_2 & = 1 & -1/2x_1 & +1/2w_2 \\ w_3 & = & +1/2x_1 & -1/2w_2 \end{array}$$

Exemplo: Terceiro privot

$$\begin{array}{lrrr} z & = -3 & -w_1 & -w_2 \\ x_1 & = 4/3 & +2/3w_1 & +1/3w_2 \\ x_2 & = 1/3 & -1/3w_1 & +1/3w_2 \\ w_3 & = 2/3 & +1/3w_1 & -1/3w_2 \end{array}$$

3.5 Dualidade em forma não-padrão

Dualidade em forma padrão

$$\begin{array}{rl} \textbf{maximiza} & c^t x \\ \textbf{sujeito a} & Ax \leq b \\ & x \geq 0 \end{array} \qquad \begin{array}{rl} \textbf{minimiza} & b^t y \\ \textbf{sujeito a} & y^t A \geq c^t \\ & y \geq 0 \end{array}$$

- O que acontece se o sistema não é em forma padrão?

Igualdades

- Caso de igualdades: Substituindo desigualdades..

$$\begin{array}{rl} \textbf{maximiza} & c^t x \\ \textbf{sujeito a} & Ax = b \\ & x \geq 0 \end{array} \qquad \begin{array}{rl} \textbf{maximiza} & c^t x \\ \textbf{sujeito a} & Ax \leq b \\ & Ax \geq b \\ & x \geq 0 \end{array}$$

- ... padronizar novamente, e formar o dual:

$$\begin{array}{rl} \textbf{maximiza} & c^t x \\ \textbf{sujeito a} & Ax \leq b \\ & -Ax \leq -b \\ & x \geq 0 \end{array} \qquad \begin{array}{rl} \textbf{minimiza} & b^t y^+ - b^t y^- \\ \textbf{sujeito a} & {y^+}^t A - {y^-}^t A \geq c \\ & y^+ \geq 0, y^- \geq 0 \\ & y^+ = (y_1^+, \ldots, y_m^+)^t \\ & y^- = (y_1^-, \ldots, y_m^-)^t \end{array}$$

Igualdades

- Equivalente, usando variáveis não-restritas $y = y^+ - y^-$

$$\begin{array}{rl} \textbf{minimiza} & b^t y \\ \textbf{sujeito a} & y^t A \geq c \\ & y^t \lesseqgtr 0 \end{array}$$

- Resumo

Primal	Dual
Igualdade	Variável dual livre
Desigualdade ($\leq$)	Variável dual não-negativa
Variável primal livre	Igualdade
Variável primal não-negativa	Desigualdade ($\geq$)

3.6 Os métodos em forma matricial

A forma matricial permite uma descrição mais compacto do método Simplex. A seguir vamos resumir os métodos Simplex primal e dual na forma matricial. Mais importante, nesse forma é possível expressar o dicionário correspondente com qualquer base em termos das dados inicias (A, c, b). Na próxima seção vamos usar essa forma para analisar a sensibilidade de uma solução ao pequenas perturbações dos dados (i.e. os coeficientes A,b, e c).

Sistema padrão

- O sistema padrão é

$$\begin{aligned} \textbf{maximiza} \quad & c^t x \\ \textbf{sujeito a} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

- Com variáveis de folga $x_{n+1}, \ldots, x_{n+m}$ e A,c,x *novo* (definição segue abaixo)

$$\begin{aligned} \textbf{maximiza} \quad & c^t x \\ \textbf{sujeito a} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Matrizes

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & 1 & & \\ a_{21} & a_{22} & \cdots & a_{2n} & & 1 & \\ \vdots & \vdots & & \vdots & & & \ddots \\ a_{m1} & a_{m2} & \dots & a_{mn} & & & 1 \end{pmatrix};$$

$$b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}; c = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \\ 0 \\ \vdots \\ 0 \end{pmatrix}; x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ x_{n+1} \\ \vdots \\ x_{n+m} \end{pmatrix}$$

Separação das variáveis

- Em cada iteração as variáveis estão separados em básicas e não-básicas.
- Conjuntos de índices correspondentes: $\mathcal{B} \dot\cup \mathcal{N} = [1, n+m]$.
- A componente i de Ax pode ser separado como

$$\sum_{1 \le j \le n+m} a_{ij} x_j = \sum_{j \in \mathcal{B}} a_{ij} x_j + \sum_{j \in \mathcal{N}} a_{ij} x_j$$

Separação das variáveis

- Para obter a mesma separação na forma matricial: Reordenamos as colunas e separamos as matrizes e vetores:

$$A = (B\ N); x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}; c = \begin{pmatrix} c_B \\ c_N \end{pmatrix}$$

- com $B \in \mathbb{R}^{m \times m}$, $N \in \mathbb{R}^{m \times n}$, $c \in \mathbb{R}^{n+m}$.

Forma matricial das equações

- Agora, $Ax = b$ é equivalente com

$$(B\ N) \begin{pmatrix} x_B \\ x_N \end{pmatrix} = Bx_B + Nx_N = b$$

- Numa solução básica, a matriz B tem posto m tal que as colunas de B formam uma *base* do $\mathbb{R}^m$. Logo B tem inversa e

$$x_B = B^{-1}(b - Nx_N) = B^{-1}b - B^{-1}Nx_N$$

Forma matricial da função objetivo

- A função objetivo é

$$z = c^t x = (c_B\ c_N) \begin{pmatrix} x_B \\ x_N \end{pmatrix} = c_B^t x_B + c_N^t x_N$$

- e usando $x_B = B^{-1}b - B^{-1}Nx_N$ obtemos

$$\begin{aligned} z &= c_B^t(B^{-1}b - B^{-1}Nx_N) + c_N^t x_N \\ &= c_B^t B^{-1}b - (c_B^t B^{-1}N - c_N^t)x_N \\ &= c_B^t B^{-1}b - ((B^{-1}N)^t c_B - c_N)^t x_N \end{aligned}$$

Dicionário em forma matricial

- Logo, o dicionário em forma matricial é

$$\begin{aligned} z &= c_B^t B^{-1}b - ((B^{-1}N)^t c_B - c_N)^t x_N \\ x_B &= B^{-1}b - B^{-1}Nx_N \end{aligned}$$

- Compare com a forma em componentes:

$$\begin{aligned} z &= \bar{z} + \sum_{j \in \mathcal{N}} \bar{c}_j x_j \\ x_i &= \bar{b}_i - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j \qquad i \in \mathcal{B} \end{aligned}$$

Dicionário em forma matricial

- Portanto, vamos identificar

$$\begin{aligned} \bar{z} &= c_B^t B^{-1}b; & \bar{c} &= -((B^{-1}N)^t c_B - c_N)^t \\ \bar{b} &= B^{-1}b; & \bar{A} &= (\bar{a}_{ij}) = B^{-1}N \end{aligned}$$

- para obter o dicionário

$$\begin{aligned} z &= \bar{z} + \bar{c}^t x_N \\ x_B &= \bar{b} - \bar{A}x_N \end{aligned}$$

Sistema dual

- As variáveis primais são

$$x = (\underbrace{x_1 \dots x_n}_{\text{original}} \underbrace{x_{n+1} \dots x_{n+m}}_{\text{folga}})^t$$

- Para manter índices correspondentes, escolhemos variáveis duais da forma

$$y = (\underbrace{y_1 \dots y_n}_{\text{folga}} \underbrace{y_{n+1} \dots y_{n+m}}_{\text{dual}})^t$$

- O dicionário do dual correspondente então é

Primal	Dual
$z = \bar{z} + \bar{c}^t x_N$	$-w = -\bar{z} - \bar{b}^t y_B$
$x_B = \bar{b} - \bar{A} x_N$	$y_N = -\bar{c} + \bar{A}^t y_B$

Primal e dual

- A solução básica do sistema primal é

$$x_N^* = 0; \quad x_B^* = \bar{b} = B^{-1} b$$

- A solução dual correspondente é

$$y_B^* = 0; \quad y_N^* = -\bar{c} = ((B^{-1}N)^t c_B - c_N)^t$$

- Com isso temos os dicionários

$$z = \bar{z} - (y_N^*)^t x_N \qquad -w = -\bar{z} - (x_B^*)^t y_B$$
$$x_B = x_B^* - (B^{-1}N) x_N \qquad y_N = y_N^* + (B^{-1}N)^t y_B$$

Método Simplex em forma matricial

- Começamos com uma partição $\mathcal{B} \dot{\cup} \mathcal{N} = [1, n+m]$.
- Em cada iteração selecionamos uma variável sainte $i \in \mathcal{B}$ e entrante $j \in \mathcal{N}$.
- Fazemos o pivot x_i com x_j.
- Depois a nova base é $\mathcal{B} \setminus \{i\} \cup \{j\}$.

Método Simplex em forma matricial

S1: Verifique solução ótima Se $y_N^* \geq 0$ a solução atual é ótima. Pare.

S2: Escolhe variável entrante Escolhe $j \in \mathcal{N}$ com $y_j^* < 0$. x_j é a variável entrante.

S3: Determine passo básico Aumentando x_j uma unidade temos novas variáveis não-básicas $x_N = x_N^* + \Delta x_N$ com $\Delta x_N = (0 \cdots 010 \cdots 0)^t = e_j$ e e_j o vetor nulo com somente 1 na posição correspondente com índice j. Como

$$x_B = x_B^* - B^{-1}Nx_N$$

a diminuição correspondente das variáveis básicas é $\Delta x_B = B^{-1}Ne_j$.

Método Simplex em forma matricial

S4: Determine aumento máximo O aumento máximo de x_j é limitado por $x_B \geq 0$, i.e.

$$x_B = x_B^* - t\Delta x_B \geq 0 \Longleftrightarrow x_B^* \geq t\Delta x_B.$$

Com $t, x_B^* \geq 0$ temos

$$t \leq t^* = \min_{\substack{i \in \mathcal{B} \\ \Delta x_i > 0}} \frac{x_i^*}{\Delta x_i}$$

S5: Escolhe variável sainte Escolhe um $i \in \mathcal{B}$ com $x_i^* = t^* \Delta x_i$.

Método Simplex em forma matricial

S5: Determine passo dual A variável entrante dual é y_i. Aumentando uma unidade, as variáveis y_N diminuem $\Delta y_N = -(B^{-1}N)^t e_i$.

S6: Determina aumento máximo Com variável sainte y_j, sabemos que y_i pode aumentar ao máximo

$$s = \frac{y_j^*}{\Delta y_j}.$$

S7: Atualiza solução

$$\begin{aligned} x_j^* &:= t & \qquad y_i^* &:= s \\ x_B^* &:= x_B^* - t\Delta x_B & \qquad y_N^* &:= y_N^* - s\Delta y_N \\ \mathcal{B} &:= \mathcal{B} \setminus \{i\} \cup \{j\} \end{aligned}$$

3.7 Análise de sensibilidade

Motivação

- Tratamos os parâmetros como ser fixados.
- Qual o efeito de uma perturbação

$$c := c + \Delta c; \quad b := b + \Delta b; \quad A := A + \Delta A?$$

(Imagina erros de medida, pequenas flutuações, etc.)

Análise de sensibilidade

- Após a solução de um sistema linear, temos o dicionário ótimo

$$\begin{aligned} z &= z^* - (y_N^*)^t x_N \\ x_B &= x_B^* - B^{-1} N x_N \end{aligned}$$

- com

$$\begin{aligned} x_B^* &= B^{-1} b \\ y_N^* &= ((B^{-1}N)^t c_B - c_N) \\ z^* &= c_B^t B^{-1} b \end{aligned}$$

Modificar c

- Mudarmos c para $\hat{c}$, mantendo a base $\mathcal{B}$.
- x_B^* não muda, mas temos que reavaliar y_N^* e z^*.
- Depois, x_B^* ainda é uma solução básica viável do sistema primal.
- Logo, podemos continuar aplicando o método Simplex primal.

Modificar b

- Da mesma forma, modificamos b para $\hat{b}$ (mantendo a base).
- y_N^* não muda, mas temos que reavaliar x_B^* e z^*.
- Depois, y_N^* ainda é uma solução básica viável do sistema dual.
- Logo, podemos continuar aplicando o método Simplex dual.

Vantagem dessa abordagem

- Nos dois casos, esperamos que a solução inicial já é perto da solução ótima.
- Experiência prática confirma isso.
- O que acontece se queremos modificar tanto b quanto c ou ainda A?
- A solução atual não necessariamente é viável no sistema primal ou dual.
- Mas: Mesmo assim, a convergência na prática é mais rápido.

Estimar intervalos

- Pergunta estendida: Qual o intervalo de $t \in R$ tal que o sistema com $\hat{c} = c + t\Delta c$ permanece ótimo?
- Para $t = 1$: $y_N^* = ((B^{-1}N)^t c_B - c_N)$ aumenta $\Delta y_N := (B^{-1}N)^t \Delta c_B - \Delta c_N$.
- Em geral: Aumento $t\Delta y_N$.
- Condição para manter a viabilidade dual:

$$y_N^* + t\Delta y_N \geq 0$$

- Para $t > 0$ temos

$$t \leq \min_{\substack{j \in \mathcal{N} \\ \Delta y_j < 0}} -\frac{y_j^*}{\Delta y_j}$$

- Para $t < 0$ temos

$$\max_{\substack{j \in \mathcal{N} \\ \Delta y_j > 0}} -\frac{y_j^*}{\Delta y_j} \leq t$$

Estimar intervalos

- Agora seja $\hat{b} = b + t\Delta b$.
- Para $t = 1$: $x_B^* = B^{-1}b$ aumenta $\Delta x_B := B^{-1}\Delta b$.
- Em geral: Aumento $t\Delta b$.

- Condição para manter a viabilidade primal:

$$x_B^* + t\Delta x_B \geq 0$$

- Para $t > 0$ temos

$$t \leq \min_{\substack{i \in \mathcal{B} \\ \Delta x_i < 0}} -\frac{x_i^*}{\Delta x_i}$$

- Para $t < 0$ temos

$$\max_{\substack{i \in \mathcal{B} \\ \Delta x_i > 0}} -\frac{x_i^*}{\Delta x_i} \leq t$$

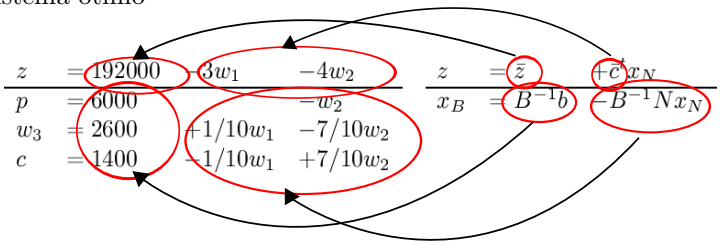
Exemplo 3.2
Considere o problema da empresa de aço (vista na aula prática, veja também execício 5.6).

$$\begin{aligned} \textbf{maximiza} \quad & 25p + 30c \\ \textbf{sujeito a} \quad & 7p + 10c \leq 56000 \\ & p \leq 6000 \\ & c \leq 4000 \end{aligned}$$

Qual o intervalo em que o valor do lucro das placas de $25R\$$ pode variar sem alterar a solução ótima?

Exemplo: Empresa de aço

- Sistema ótimo



- Base $\mathcal{B} = \{p, w_3, c\}$, variáveis não-básicas $\mathcal{N} = \{w_1, w_2\}$. (Observe: Usamos conjuntos de variáveis, ao invés de conjuntos de índices).

Exemplo: Variáveis

- Vetores c e Δc. Observe que reordenamos dos dados do sistema inicial de forma correspondente com a ordem das variáveis do sistema final.

$$c = \begin{pmatrix} 25 \\ 0 \\ 30 \\ 0 \\ 0 \end{pmatrix}; c_B = \begin{pmatrix} 25 \\ 0 \\ 30 \end{pmatrix}; c_N = \begin{pmatrix} 0 \\ 0 \end{pmatrix};$$

$$\Delta c = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}; \Delta c_B = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}; \Delta c_N = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Exemplo: Aumentos

- Aumento das variáveis duais

$$\Delta y_N = (B^{-1}N)^t \Delta c_B - \Delta c_N = (B^{-1}N)^t \Delta c_B$$

- com

$$B^{-1}N = \begin{pmatrix} 0 & 1 \\ -1/10 & 7/10 \\ 1/10 & -7/10 \end{pmatrix}$$

- temos

$$\Delta y_N = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Exemplo: Limites

- Limites em geral

$$\max_{\substack{j \in \mathcal{N} \\ \Delta y_j > 0}} -\frac{y_j^*}{\Delta y_j} \leq t \leq \min_{\substack{j \in \mathcal{N} \\ \Delta y_j < 0}} -\frac{y_j^*}{\Delta y_j}$$

- Logo

$$-4 \leq t \leq \infty.$$

- Uma variação do preço entre $25 + [-4, \infty] = [21, \infty]$ preserve a otimalidade da solução atual.

- O valor da função objetivo pode variar, mas os valores das variáveis p e c permanecem os mesmos.

$\Diamond$

Exemplo 3.3
Qual o intervalo em que o lucro das placas (R$ 25) e dos canos (R$ 30) podem variar sem que a solução ótima seja alterada?

Exemplo: Variação do lucro dos placas e canos

- Neste caso, os vetores c, c_B, c_N e Δc_N permanecem os mesmos do exemplo anterior. Enquanto que:

$$\Delta c = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}; \Delta c_B = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix};$$

- Neste caso, o valor de Δy_N é

$$\Delta y_N = (B^{-1}N)^t \Delta c_B = \begin{pmatrix} 0 & -1/10 & 1/10 \\ 1 & 7/10 & -7/10 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1/10 \\ 3/10 \end{pmatrix};$$

- Logo $-40/3 \leq t \leq \infty$
- Ou seja, uma variação do lucro das placas entre R$ 11.67 e ∞, e do lucro dos canos entre R$ 16.67 e ∞, não altera a solução ótima do sistema.

$\Diamond$

Exemplo: Modificação

- Qual o intervalo em que o lucro dos canos (R$ 30) podem variar sem que a solução ótima seja alterada?
- Neste caso, os vetores c, c_B, c_N e Δc_N permanecem os mesmos do exemplo anterior. Enquanto que:

$$\Delta c = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}; \Delta c_B = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix};$$

- Neste caso, o valor de Δy_N é:

$$\Delta c_B = \begin{pmatrix} 1/10 \\ -7/10 \end{pmatrix};$$

- Logo $-30 \leq t \leq 40/7$
- Ou seja, uma variação do lucro dos canos entre R$ 0 e R$ 35.71, não altera a solução ótima do sistema.

Exemplo 3.4
O que acontece se mudarmos o lucro das placas para R$ 20?

Exemplo: Placas com lucro R$ 20

- Novos vetores

$$\hat{c} = \begin{pmatrix} 20 \\ 0 \\ 30 \\ 0 \\ 0 \end{pmatrix}; \hat{c}_B = \begin{pmatrix} 20 \\ 0 \\ 30 \end{pmatrix}; \hat{c}_N = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

- Aumento

$$\begin{aligned} \hat{y}_N^* &= (B^{-1}N)^t \hat{c}_B - \hat{c}_N = (B^{-1}N)^t \hat{c}_B \\ &= \begin{pmatrix} 0 & -1/10 & 1/10 \\ 1 & 7/10 & -7/10 \end{pmatrix} \begin{pmatrix} 20 \\ 0 \\ 30 \end{pmatrix} = \begin{pmatrix} 3 \\ -1 \end{pmatrix} \end{aligned}$$

Novas variáveis

- Com

$$B^{-1}b = \begin{pmatrix} 6000 \\ 2600 \\ 1400 \end{pmatrix}$$

- Novo valor da função objetivo

$$\hat{z}^* = \hat{c}_B^t B^{-1} b = \begin{pmatrix} 20 & 0 & 30 \end{pmatrix} \begin{pmatrix} 6000 \\ 2600 \\ 1400 \end{pmatrix} = 162000$$

Exemplo: Novo dicionário

- Novo sistema primal viável, mas não ótimo:

z	$= 162000$	$-3w_1$	$+w_2$
p	$= 6000$		$-w_2$
w_3	$= 2600$	$+1/10w_1$	$-7/10\mathbf{w_2}$
c	$= 1400$	$-1/10w_1$	$+7/10w_2$

- Depois um pivot: Sistema ótimo.

z	$= 165714\,2/7$	$-20/7w_1$	$-10/7w_3$
p	$= 2285\,5/7$	$-1/7w_1$	$+10/7w_3$
w_2	$= 3714\,2/7$	$+1/7w_1$	$-10/7w_2$
c	$= 4000$		$-w_3$

$\Diamond$

Exemplo 3.5
O que acontece se mudarmos o lucro das placas de R$ 25 para R$ 35 e dos canos de R$ 30 para R$ 10?

Exemplo: Placas e canos com lucro R$ 35 e R$ 10

- Novos vetores

$$\hat{c} = \begin{pmatrix} 35 \\ 0 \\ 10 \\ 0 \\ 0 \end{pmatrix}; \hat{c}_B = \begin{pmatrix} 35 \\ 0 \\ 10 \end{pmatrix}; \hat{c}_N = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

- Aumento

$$\hat{y}^*_N = ((B^{-1}N)^t c_B - c_N) = \begin{pmatrix} 0 & -1/10 & 1/10 \\ 1 & 7/10 & -7/10 \end{pmatrix} \begin{pmatrix} 35 \\ 0 \\ 10 \end{pmatrix} = \begin{pmatrix} 1 \\ 28 \end{pmatrix} =$$

Novas variáveis e novo dicionário

- Novo valor da função objetivo

$$\hat{z}^* = \hat{c}^t_B B^{-1} b = \hat{c}^t_B x^*_B = \begin{pmatrix} 35 & 0 & 10 \end{pmatrix} \begin{pmatrix} 6000 \\ 2600 \\ 1400 \end{pmatrix} = 224000$$

- O novo sistema primal viável é

z	$= 224000$	$-1w_1$	$-28w_2$
p	$= 6000$		$-w_2$
w_3	$= 2600$	$+1/10w_1$	$-7/10w_2$
c	$= 1400$	$-1/10w_1$	$+7/10w_2$

- O sistema é ótimo.

$\Diamond$

4 Tópicos

- Formalização de problemas, modelagem de problemas.
- Aplicações.
- Técnicas: Maximização do mínimo ou máximo de um conjunto finito de valores.

4.1 Função objetivo linear por segmentos

A função objetivo de um programa linear deve ser linear. Caso o problema possui uma função objetivo de outra classe (por exemplo quadrática), em geral o método Simplex não se aplica mais e a solução se torna mais difícil. Porém, nessa capítulo vamos conhecer exeções importantes: *Funções lineares por segmentos* (ingl. piecewise linear functions) e *funções racionais*.
Uma função f linear por segmentos com n segmentos é definido por

$$f(x) = f_i(x) \qquad \text{para } v_{i-1} \leq x \leq v_i\text{n} \tag{4.1}$$

usando n funções lineares $f_1, \ldots, f_n$ e $n-1$ pontos de separação $v_1, \ldots, v_{n-1}$ tal que $f_i(v_1) = f_{i+1}(v_i)$ para $1 \leq i < n$ (com pontos auxiliares $v_0 = -\infty$ e $v_n = \infty$); veja figura 4.1.
Para avaliar $f(x)$ temos que saber qual o segmento "ativo" (o segmento que contém o valor atual). Uma ideía expressar $f(x)$ como função linear é separar a variável x em n variáveis x_i positivas (exceto x_1) que representam a contribuição do segmento i ao valor total do x, tal que

$$x = \sum_{1 \leq i \leq n} x_i$$

e otimizar a função objetivo

$$f_1(x_1) + (f_2(v_1 + x_2) - f_2(v_1)) + \cdots + (f_n(v_{n-1} + x_n) - f_n(v_{n-1})). \tag{4.2}$$

com restrições

$$\begin{aligned} x_1 &\leq v_1 \\ 0 \leq x_i &\leq v_i - v_{i-1} \qquad\qquad 2 \leq i \leq n \end{aligned}$$

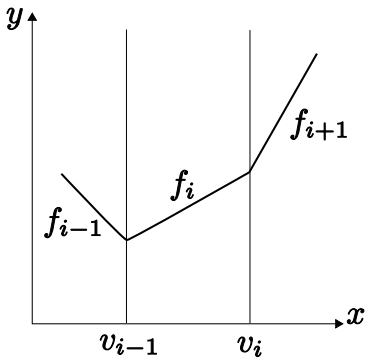


Figura 4.1: Representação de funções lineares por segmentos.

O problema com essa abordagem é garantir, que x é representado "da esquerda à direita", i.e. antes de atribuir um valor à variável x_i, todas variáveis x_j com $j < i$ devem receber o seu valor máximo. O exemplo seguinte mostra, que isso nem sempre é o caso.

Exemplo 4.1
Vamos supor que queremos maximizar a função $f(x) = |x|$ no intervalo $-1 \leq x \leq 1$, i.e. resolver o problema de otimização

$$\begin{aligned} \textbf{maximiza} \quad & |x| \\ \textbf{sujeito a} \quad & -1 \leq x \leq 1 \end{aligned}$$

O ponto $v_1 = 0$ e $f_1(x) = -x$ e $f_2(x) = x$ definam os segmentos lineares dessa função conforme equação 4.1.
A função objetivo linear conforme equação 4.2 é

$$f_1(x_1) + f_2(v_1 + x_2) - f_2(v_1) = -x_1 + (x_2 - 1)$$

e o sistema completa correspondente com variáveis x_1 e x_2 seria

$$\begin{aligned} \textbf{maximiza} \quad & -x_1 + x_2 - 1 && (4.3) \\ \textbf{sujeito a} \quad & x_1 \leq 0 && (4.4) \\ & -1 \leq x_1 + x_2 \leq 1 && (4.5) \\ & x_2 \geq 0 && (4.6) \end{aligned}$$

As soluções $-x_1 = x_2 = c$ com $c \to \infty$ mostram, que esse sistema é ilimitado.
$\Diamond$

O exemplo mostrou, que a nossa abordagem não funciona no caso geral.
TBD

- Motivate, that for concave functions, the maximization would be correct, and, similarly, for convex function the minimization would work.
- Discuss why solutions of the type $x_{i+1} \leq Mx_i$ would not work in general. How to choose M?
- Give a pointer to integer programming, which allows to handle arbitrary piecewise linear functions.

4.2 Eliminação de Fourier-Motzkin

A eliminação de Fourier-Motzkin foi a primeira técnica para resolver uma sistema de desigualdades. Fourier inventou o método em 1826. Ele foi redescoberta pelo Motzkin em 1936. O objetivo é achar um soluções do sistema

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &\leq b_2 \\ &\cdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &\leq b_m \end{aligned}$$

(brevemente $Ax \leq b$ em notação matricial). Começamos com um exemplo. Dado

$$\begin{aligned} 2x_1 - 3x_2 &\leq 5 \\ 4x_1 + 8x_2 &\leq 20 \end{aligned}$$

nos queremos eliminar a variável x_2. Reescrevendo as desigualdades, obtemos

$$\begin{aligned} 3x_2 \geq 2x_1 - 5 &\iff x_2 \geq 2/3x_1 - 5/3 \\ 8x_2 \leq 20 - 4x_1 &\iff x_2 \leq 5/2 - 1/2x_1 \end{aligned}$$

Combinado as duas desigualdades, temos

$$2/3x_1 - 5/3 \leq 5/2 - 1/2x_1 \iff x_1 \leq 20/7$$

A partir dessa desigualdade podemos concluir, que existem soluções viáveis. Para obter uma solução concreta, podemos escolher um valor arbitrário de x_1, por exemplo $x_1 = 2$. Substituindo de novo nas primeiras equações obtemos

$$-1/3 \leq x_2 \leq 3/2$$

e podemos escolher, por exemplo, o valor $x_2 = 1$.
Em geral, a idéia é repetidamente eliminar uma variável com essa técnica até chegar numa única variável. Se as equações restantes são viáveis, o sistema original também é, e com o método de substituição podemos produzir soluções. Particularmente, para eliminar a variável x_k separamos os coeficientes a_{ik} em $L = \{i|a_{ik} < 0\}$ e $U = \{i|a_{ik} > 0\}$. Cada equação em L gera uma cota inferior para x_k e cada equação em U uma cota superior da forma

$$a_{ik}^{-1}\left(\sum_{j\neq k} a_{ij}x_j - b_i\right) \leq x_k \qquad \text{para } i \in L$$

$$x_k \leq a_{ik}^{-1}\left(b_i - \sum_{j\neq k} a_{ij}x_j\right) \qquad \text{para } i \in U$$

Em seguida, podemos eliminar a variável x_k combinando cada par de cota inferior e cota superior, e ainda copiando as desigualdades que não contém x_k para o novo sistema:

$$a_{ik}^{-1}\left(\sum_{j\neq k} a_{ij}x_j - b_i\right) \leq a_{i'k}^{-1}\left(b'_i - \sum_{j\neq k} a_{i'j}x_j\right) \forall i \in L, i' \in U$$

(Observe que no caso $L = \emptyset$ ou $U = \emptyset$ não serão geradas novas desigualdades.) No pior caso nos temos $|L| = |U| = m/2$ e portanto $(m/2)^2$ equações depois uma eliminação de uma variável. Logo, em cada iteração o número de equações pode ser o quadrado do número anterior. Eliminando $n-1$ variáveis, o algoritmo possui uma complexidade pessimista de $O(m^{2^n})$.

4.3 Métodos de pontos interiores

- TBD: Karmarkar.

4.4 Relaxaçao Lagrangeana

Queremos resolver um problema de otimização na forma geral

$$\begin{aligned} \textbf{maximiza} \quad & f(x) \\ \textbf{sujeito a} \quad & g(x) = 0 \\ & x \in \mathbb{R}^n \end{aligned}$$

com funções arbitrárias $f(x)$ e $g(x)$. Uma abordagem para obter a solução ótima seria resolver a restrições $g(x) = 0$, i.e. $V = \{x \mid g(x) = 0\}$ e depois maximizar sobre $f(x)$ para $x \in V$.

Exemplo 4.2

$$\begin{aligned} \textbf{maximiza} \quad & 2x + y \\ \textbf{sujeito a} \quad & x^2 + y^2 = 2 \\ & x, y\mathbb{R} \end{aligned}$$

Nesse exemplo podemos usar a restrição para eliminar $y = \pm\sqrt{2 - x^2}$ na função objetivo e depois maximizar $h(x) = 2x \pm \sqrt{2 - x^2}$ para $x \in \mathbb{R}$. A derivada é $h'(x) = 2 \pm x(2 - x^2)^{-1/2}$ e a condição $h'(x) = 0$ leva a solução $x = \sqrt{8/5}$ e $y = \sqrt{2/5}$.
TBD: Gráfico simples. ◊

TBD: I tried to complicate the example, using

$$\begin{aligned} \textbf{maximiza} \quad & -x - y && (4.7) \\ \textbf{sujeito a} \quad & x^2 + y^2 = e^{cx+y} && (4.8) \\ & x, y \in \mathbb{R} && (4.9) \end{aligned}$$

but the Lagrangian approach makes the example not _much_more solvable.
TBD: Lagrangian approach.

Abordagem Lagrangeana Supomos que x^* é uma solução ótima que satisfaz $g(x^*) = 0$. Logo, o normal das restrições $g(x)$ é na direção do gradiente da função objetivo $f(x)$, senão a função objetivo não é um máximo local. Isso permite obter a solução ótimo como solução do sistema

$$\begin{aligned} \nabla f(x) &= \lambda \nabla g(x) \\ g(x) &= 0 \\ \lambda &\geq 0 \end{aligned}$$

de $n + 1$ equações e inconhecidas. O *multiplicador Lagrangeana* λ representa a fração entre o tamanho de ∇f e ∇g.
TBD: Gráfico.
TBD: Isnt this just a necessary condition for a local minimum? Wheres the optimization?

Para escrever o sistema mais compacto, vamos introduzir a *função Lagrangeana*

$$L(x, \lambda) = f(x) - \lambda g(x).$$

TBD: Check the sign, Usually (see below with multiple constraints), its positive.
TBD: The penalty interpretation.
Com isso obtemos a condição simples

$$\nabla L(x, \lambda) = 0; \qquad \lambda \geq 0$$

TBD: Exemplo.
TBD: Talk about the advantages (1) Function g(x) must not be inverted! (2) Instead of optimizing a function in n vars with k constraints, after determining the λ's, we can optimize and unconstrained function in $n + k$ vars. Give examples, which show this!

Multiplas restrições No caso geral temos um sistema com k restrições

$$\begin{array}{ll} \textbf{optimiza} & f(x) \\ \textbf{sujeito a} & g_i(x) = 0 \qquad\qquad 1 \leq i \leq k \\ & x \in \mathbb{R}^n \end{array}$$

e a função Lagrageana

$$L(x, \lambda) = f(x) + \sum_{1 \leq i \leq k} \lambda_i g_i(x) \tag{4.10}$$

com multiplicadores Lagrangeanos $\lambda = (\lambda_1, \ldots, \lambda_k) \in \mathbb{R}^k$.

Aplicação à PL Dado o sistema

$$\begin{array}{ll} \textbf{maximiza} & c^t x \\ \textbf{sujeito a} & Ax = b \\ & x \geq 0 \end{array}$$

a função Lagrangeana é

$$L(x, \lambda) = c^t x + \sum_{1 \leq i \leq k} \lambda_i (a_i x - b_i)$$

e obtemos as condições

$$\begin{aligned} c_j + \sum_{1 \leq i \leq k} \lambda_i a_{ij} &= 0 && 1 \leq j \leq n \\ a_j x - b_j &= 0 && 1 \leq j \leq k \end{aligned}$$

ou equivalente

$$\begin{aligned} \lambda^t A &= -c^t \\ Ax &= b \end{aligned}$$

TBD: Relate it better to the dual of the LP. For now, we only know that x is a primal and y is a dual solution.

5 Exercícios

(Soluções a partir da página 181.)

Exercício 5.1
Na definição da programação linear permitimos restrições lineares da forma

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \bowtie_i b_i$$

com $\bowtie_i \in \{\leq, =, \geq\}$. Porque não permitimos $\bowtie_i \in \{<, >\}$ também? Discute.

Exercício 5.2
Procura a tabela nutricional de algum restaurante e resolve o problema da dieta (exemplo 1.2).

Exercício 5.3
Um investidor pode vender ações de suas duas empresas na bolsa de valores, mas está sujeito a um limite de 10.000 operações diárias (vendas por dia). Na cotação atual, as ações da empresa A valorizaram-se 10% e agora cada uma vale R$ 22. Já a empresa B teve valorização de 2% e cada ação vale R$ 51. Sabendo-se que o investidor possui 6.000 ações da Empresa A e 7.000 da empresa B, maximize seu lucro na BOVESPA e diga qual o lucro obtido.

Exercício 5.4
Dona Maria adora ver seus netinhos Marcos, Renato e Vinicius bem alimentados. Sempre na hora de cozinhar ela leva em conta o quanto eles gostam de cada prato para fazê-los comer o máximo possível. Marcos gosta da lasanha e comeria 3 pratos dela após um prato de sopa; Renato prefere lanches, e comeria 5 hambúrgueres, ignorando a sopa; Vinicius gosta muita da massa a bolonhesa, e comeria 2 pratos após tomar dois pratos de sopa. Para fazer a sopa, são necessários entre outros ingredientes, 70 gramas de queijo por prato e 30 gramas de carne. Para cada prato de lasanha, 200 gramas de queijo, e 100 gramas de carne. Para cada hambúrguer são necessários 100 gramas de carne, e 100 gramas de queijo. Para cada prato de massa a bolonhesa so necessários 100 gramas de carne e 30 gramas de queijo (ralado para por sobre a massa). Seus netos vieram visitá-la de surpresa, e tendo ela somente 800 gramas de carne e 1000 gramas de queijo em casa, como ela poderia fazê-los comer o maior número de pratos, garantindo que cada um deles comerá pelo menos dois pratos, e usando somente os ingredientes que ela possui?

Exercício 5.5
Uma cidade foi invadida por pombas, e o prefeito obteve licença para eliminar 35.000 pombas durante apenas um dia. Ele pode usar duas estratégias: ou contratar adultos que poderão portar uma arma de matar pombas, ou crianças com "bodoques". O prefeito tem orçamento para pagar no máximo 340 adultos, sendo que 200 se inscreveram para a missão, enquanto que 7000 crianças se inscreveram. Estima-se que um adulto consiga matar 40 pombas por dia, enquanto que uma criança apenas 5. Cada adulto receberá um valor de R$30 pelo dia de trabalho, enquanto que as crianças receberão R$10. Quantos adultos e quantas crianças devem ser contratados de forma que o serviço seja feito, e seja gasto o menor custo possível com a missão?

Exercício 5.6 ([4])
Formalize como problema de otimização linear e resolve graficamente.
Uma empresa de aço produz placas ou canos de ferro. As taxas de produção são 200t/h para placas e 140t/h para canos. O lucro desses produtos e 25$/t para placas e 30$/t para canos. Considerando a demanda atual, os limites de produção são 6000t de placas e 4000t de canos. Na semana atual são 40h de tempo de produção disponível. Quantas toneladas de placas e canos devem ser produzidas para maximizar o lucro?

Exercício 5.7 ([4])
Formalize como problema de otimização linear.
Uma pequena empresa aérea oferece um vôo de Pelotas, com escala em Porto Alegre para Torres. Logo tem três tipos de clientes que voam Pelotas–Porto Alegre, Pelotas–Torres e Porto Alegre–Torres. A linha também oferece três tipos de bilhetes:

- Tipo A: bilhete regular.
- Tipo B: sem cancelamento.
- Tipo C: sem cancelamento, pagamento três semanas antes de viajar.

Os preços (em R$) dos bilhetes são os seguintes

	Pelotas–Porto Alegre	Porto Alegre–Torres	Pelotas–Torres
A	600	320	720
B	440	260	560
C	200	160	280

Baseado em experiência com esse vôo, o marketing tem a seguinte predição de passageiros:

	Pelotas–Porto Alegre	Porto Alegre–Torres	Pelotas–Torres
A	4	8	3
B	8	13	10
C	22	20	18

O objetivo da empresa e determinar o número ótimo de bilhetes para vender de cada tipo, respeitando um limite de 30 passageiros em cada vôo e o limite dos passageiros previstos em cada categoria, que maximiza o lucro.

Exercício 5.8
Escreve em forma normal.

$$\begin{aligned} \textbf{minimiza} \quad & z = -5x_1 - 5x_2 - 5x_3 \\ \textbf{sujeito a} \quad & -6x_1 - 2x_2 - 9x_3 \leq 0 \\ & -9x_1 - 3x_2 + 3x_3 = 3 \\ & x_j \geq 0 \end{aligned}$$

$$\begin{aligned} \textbf{maximiza} \quad & z = -6x_1 - 2x_2 - 6x_3 + 4x_4 + 4x_5 \\ \textbf{sujeito a} \quad & -3x_1 - 8x_2 - 6x_3 - 7x_4 - 5x_5 = 3 \\ & 5x_1 - 7x_2 + 7x_3 + 7x_4 - 6x_5 \leq 6 \\ & 1x_1 - 9x_2 + 5x_3 + 7x_4 - 10x_5 = -6 \\ & x_j \geq 0 \end{aligned}$$

$$\begin{aligned} \textbf{maximiza} \quad & z = 7x_1 + 4x_2 + 8x_3 + 7x_4 - 9x_5 \\ \textbf{sujeito a} \quad & -4x_1 - 1x_2 - 7x_3 - 8x_4 + 6x_5 = -2 \\ & x_1 + 4x_2 + 2x_3 + 2x_4 - 7x_5 \geq -7 \\ & -8x_1 + 2x_2 + 8x_3 - 6x_4 - 7x_5 = -7 \\ & x_j \geq 0 \end{aligned}$$

$$\begin{aligned} \textbf{minimiza} \quad & z = -6x_1 + 5x_2 + 8x_3 + 7x_4 - 8x_5 \\ \textbf{sujeito a} \quad & -5x_1 - 2x_2 + x_3 - 9x_4 - 7x_5 = 9 \\ & 7x_1 + 7x_2 + 5x_3 - 3x_4 + x_5 = -8 \\ & -5x_1 - 3x_2 - 5x_3 + 9x_4 + 8x_5 \leq 0 \\ & x_j \geq 0 \end{aligned}$$

Exercício 5.9 ([3])
Resolve com o método Simplex.

$$\begin{aligned} \textbf{maximiza} \quad & z = 3x_1 + 5x_2 \\ \textbf{sujeito a} \quad & x_1 \leq 4 \\ & x_2 \leq 6 \\ & 3x_1 + 2x_2 \leq 18 \\ & x_j \geq 0 \end{aligned}$$

Exercício 5.10
Resolve o exercício 5.6 usando o método Simplex.

Exercício 5.11
Prova que

$$\frac{2^{2n}}{2n} \leq \binom{2n}{n} \leq 2^{2n}.$$

Exercício 5.12
Resolve o sistema degenerado

z	$=$	$10x_1$	$-57x_2$	$-9x_3$	$-24x_4$
w_1	$=$	$-1/2x_1$	$+11/2x_2$	$+5/2x_3$	$-9x_4$
w_2	$=$	$-1/2x_1$	$+3/2x_2$	$+1/2x_3$	$-x_4$
w_3	$=1$	$-x_1$			

usando o método lexicográfico e o regra de Bland.

Exercício 5.13
Dado o problema de otimização

$$\begin{aligned} \textbf{maximiza} \quad & x_1 + x_2 \\ \textbf{sujeito a} \quad & ax_1 + bx_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

determine condições suficientes e necessárias ao respeita de a e b tal que

1. existe ao menos uma solução ótima,
2. existe exatamente uma solução ótima,
3. existe nenhuma solução ótima,

4. o sistema é ilimitado.

ou demonstre que o caso não é possível.

Exercício 5.14
Sabe-se que o dicionário ótimo do problema

$$\begin{aligned} \textbf{maximiza} \quad & z = 3x_1 + x_2 \\ \textbf{sujeito a} \quad & 2x_1 + 3x_2 \leq 5 \\ & x_1 - x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

é

$$\begin{array}{llll} z^* & = 31 & -11w_2 & -4w_1 \\ \hline x_2 & = 7 & -2w_2 & -w_1 \\ x_1 & = 8 & -3w_2 & -w_1 \end{array}$$

- Se a função objetivo passar a $z = x_1 + 2x_2$, a solução continua ótima? No caso de resposta negativa, determine a nova solução ótima.
- Se a função objetivo passar a $z = x_1 - x_2$, a solução continua ótima? No caso de resposta negativa, determine a nova solução ótima.
- Se a função objetivo passar a $z = 2x_1 - 2x_2$, a solução continua ótima?No caso de resposta negativa, determine a nova solução ótima.
- Formular o dual e obter a solução dual ótima.

Parte II

Programação inteira

6 Introdução

6.1 Definições

Problema da dieta

- Problema da dieta

$$\begin{aligned} \textbf{minimiza} \quad & c^t x \\ \textbf{sujeito a} \quad & Ax \geq r \\ & x \geq 0 \end{aligned}$$

- com limites quantidade de comida x.
- Uma solução (laboratório): 5 McDuplos, 3 maçãs, 2 casquinhas mista para R$ 24.31
- Mentira! Solução correta: 5.05 McDuplos, 3.21 maças, 2.29 casquinhas mistas.
- Observação: Correto somente em média sobre várias refeições.

Como resolver?

- Única refeição? Como resolver?
- Restringe a variáveis x ao $\mathbb{Z}$.
- Será que metodo Simplex ainda funciona?
- Não. Pior: O problema torna-se NP-completo.

Problemas de otimização

- Forma geral

$$\begin{aligned} \textbf{optimiza} \quad & f(x) \\ \textbf{sujeito a} \quad & x \in V \end{aligned}$$

Programação inteira

- Programação linear (PL)

$$\begin{aligned} \textbf{maximiza} \quad & c^t x \\ \textbf{sujeito a} \quad & Ax \leq b \\ & x \in \mathbb{R}^n \geq 0 \end{aligned}$$

- Programação inteira pura (PI)

$$\begin{aligned} \textbf{maximiza} \quad & h^t y \\ \textbf{sujeito a} \quad & Gy \leq b \\ & y \in \mathbb{Z}^n \geq 0 \end{aligned}$$

Programação inteira

- Programção (inteira) mista (PIM)

$$\begin{aligned} \textbf{maximiza} \quad & c^t x + h^t y \\ \textbf{sujeito a} \quad & Ax + Gy \leq b \\ & x \in \mathbb{R}^n \geq 0, y \in \mathbb{Z}^n \geq 0 \end{aligned}$$

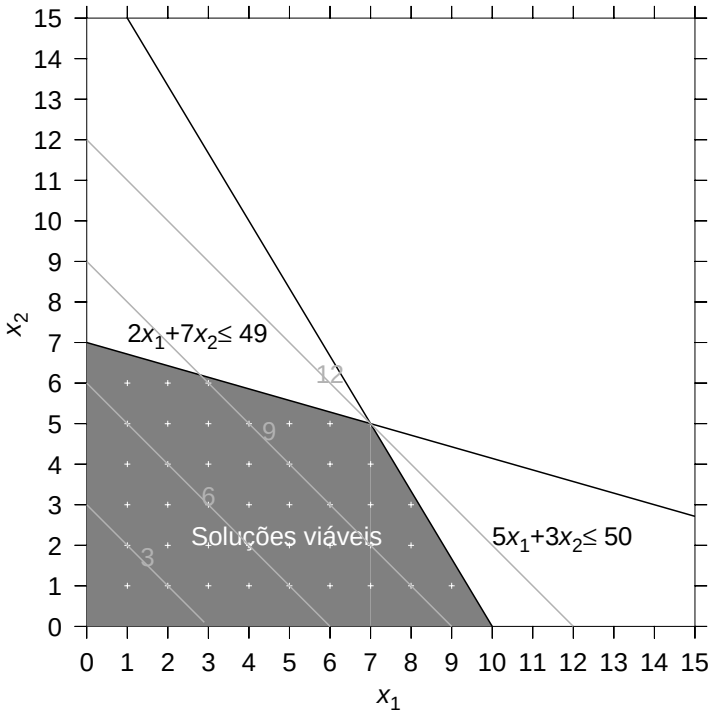
- Programação linear e inteira pura são casos particulares da programação mista.
- Outro caso particular: 0-1-PIM e 0-1-PI.

$$x \in \mathbb{B}^n$$

Exemplo

$$\begin{aligned} \textbf{maximiza} \quad & x_1 + x_2 \\ \textbf{sujeito a} \quad & 2x_1 + 7x_2 \leq 49 \\ & 5x_1 + 3x_2 \leq 50 \end{aligned}$$

Exemplo

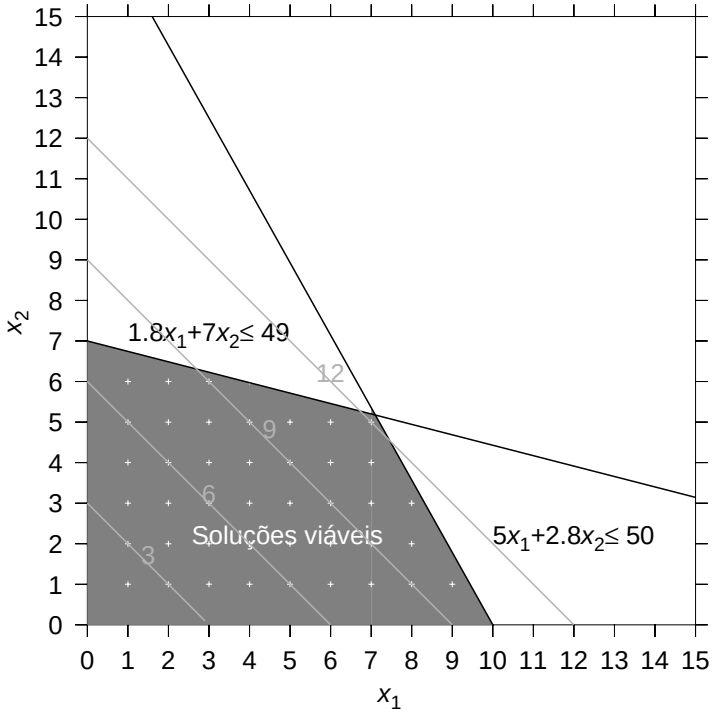


- Sorte: A solução ótima é inteira! $x_1 = 7$, $x_2 = 5$, $V = 12$.
- Observação: Se a solução ótima é inteira, um problema de PI(M) pode ser resolvido com o método Simplex.

Exemplo

$$\begin{aligned} \textbf{maximiza} \quad & x_1 + x_2 \\ \textbf{sujeito a} \quad & \mathbf{1.8}x_1 + 7x_2 \leq 49 \\ & 5x_1 + \mathbf{2.8}x_2 \leq 50 \end{aligned}$$

Exemplo

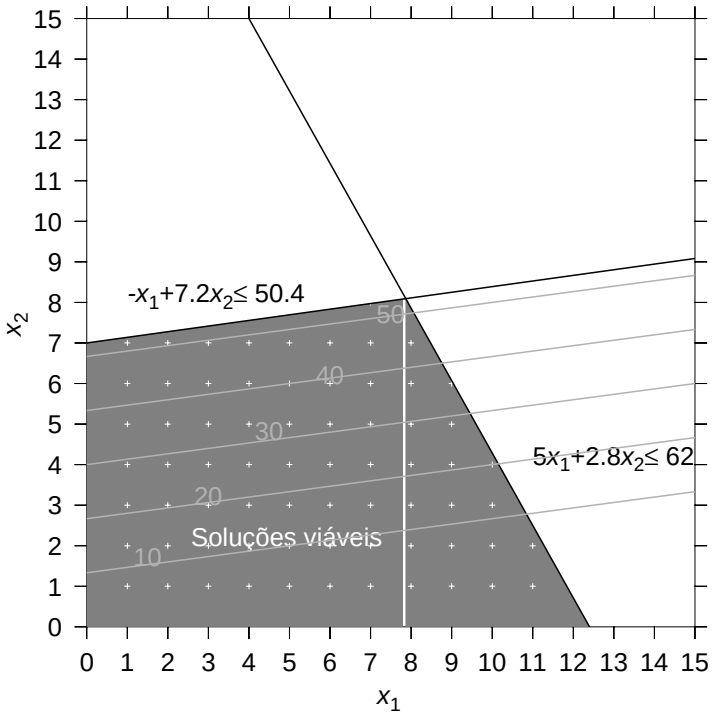


- Solução ótima agora: $x_1 \approx 7.10$, $x_2 \approx 5.17$, $V = 12.28$.
- Será que $\lfloor x_1 \rfloor, \lfloor x_2 \rfloor$ é a solução ótima do PI?

Exemplo

$$\begin{aligned} \textbf{maximiza} \quad & -x_1 + 7.5x_2 \\ \textbf{sujeito a} \quad & -x_1 + 7.2x_2 \leq 50.4 \\ & 5x_1 + 2.8x_2 \leq 62 \end{aligned}$$

Exemplo



- Solução ótima agora: $x_1 \approx 7.87$, $x_2 \approx 8.09$, $V = 52.83$.
- $\lfloor x_1 \rfloor = 7$, $\lfloor x_2 \rfloor = 8$.
- Solução ótima inteira: $x_1 = 0, x_2 = 7$!
- Infelizmente a solução ótima inteira pode ser arbitrairamente distante!

Métodos

- Prove que a solução da relaxação linear sempre é inteira.
- Insere cortes.
- Branch-and-bound.

Exemplo: 0-1-**Knapsack**

Problema da Mochila (Knapsack)

Instância Um conjunto de n items $I = \{i_1, \ldots, i_n\}$ com valores v_i e pesos w_i. Um limite de peso K do mochila.

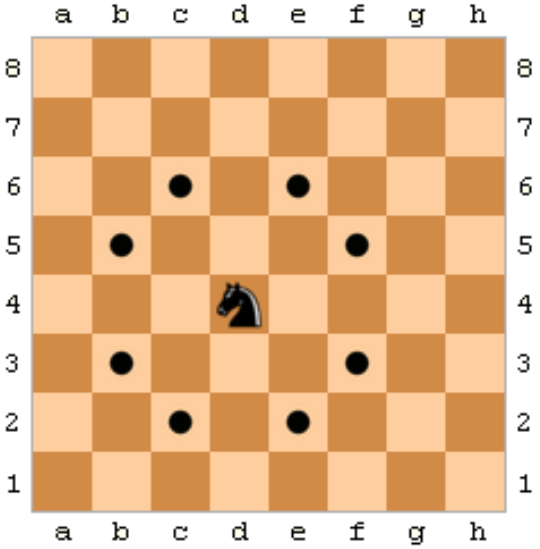
Solução Um conjunto $S \subseteq I$ de elements que cabem na mochila, i.e. $\sum_{i \in S} w_i \leq K$.

Objetivo Maximizar o valor $\sum_{i \in S} v_i$.

- Observação: Existe um solução com programação dinâmica que possui complexidade de tempo $O(Kn)$ (pseudo-polinomial) e de espaço $O(K)$.

Exemplo: Maximizar cavalos

- Qual o número máximo de cavalos que cabe num tabuleiro de xadrez, tal que nenhum ameaça um outro?



Exemplo 6.1
Formulação do problema da mochila, com variáveis indicadores x_i, $1 \leq i, j \leq$

8:

$$\begin{aligned} \textbf{maximiza} \quad & \sum_i v_i x_i \\ \textbf{sujeito a} \quad & \sum_i w_i x_i \leq L \\ & x_i \in \mathbb{B} \end{aligned}$$

Formulação do problema dos cavalos com variáveis indicadores x_{ij}:

$$\begin{aligned} \textbf{maximiza} \quad & \sum_{i,j} x_{ij} \\ \textbf{sujeito a} \quad & x_{ij} + x_{i-2,j+1} \leq 1 \qquad 3 \leq i \leq 8, 1 \leq j \leq 7 \\ & x_{ij} + x_{i-1,j+2} \leq 1 \qquad 2 \leq i \leq 8, 1 \leq j \leq 6 \\ & x_{ij} + x_{i+2,j+1} \leq 1 \qquad 1 \leq i \leq 6, 1 \leq j \leq 7 \\ & x_{ij} + x_{i+1,j+2} \leq 1 \qquad 1 \leq i \leq 7, 1 \leq j \leq 6 \end{aligned}$$

Soluções do problema dos cavaleiros (A030978)

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
k	1	4	5	8	13	18	25	32	41	50	61	72	85	98	113	128

$\Diamond$

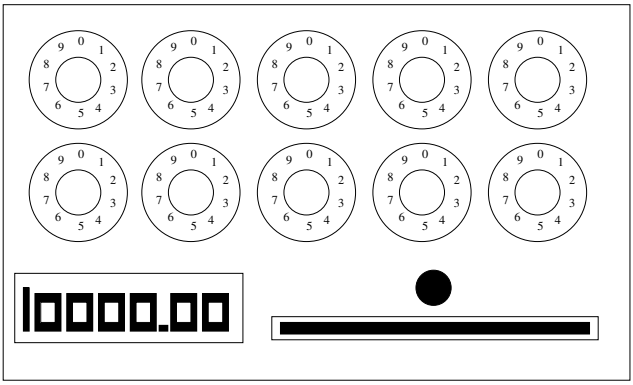
6.2 Motivação e exemplos

Motivação

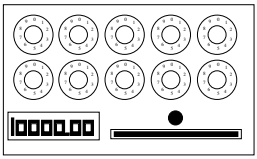
- Otimização combinatória é o ramo da ciência da computação que estuda problemas de otimização em conjuntos (wikipedia).
- "The discipline of applying advanced analytical methods to help make better decisions" (INFORMS)
- Tais problemas são extremamente frequentes e importantes.

Máquina de fazer dinheiro

- Imagine uma máquina com 10 botões, cada botão podendo ser ajustado em um número entre 0 e 9.



Máquina de fazer dinheiro



- há uma configuração que retorna R$ 10.000.
- total de combinações: 10^{10}.
- dez testes por segundo
- em um ano:$\Rightarrow 10 \times 60 \times 60 \times 24 \times 365 \cong 3 \times 10^8$

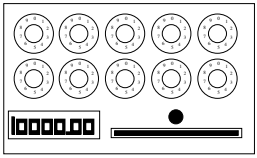
Explosão combinatória

Funções típicas:

n	$\log n$	$n^0.5$	n^2	2^n	$n!$
10	3.32	3.16	10^2	1.02×10^3	3.6×10^6
100	6.64	10.00	10^4	1.27×10^{30}	9.33×10^{157}
1000	9.97	31.62	10^6	1.07×10^{301}	4.02×10^{2567}

"Conclusões"

[1] retirado de Integer Programming - Wolsey (1998)



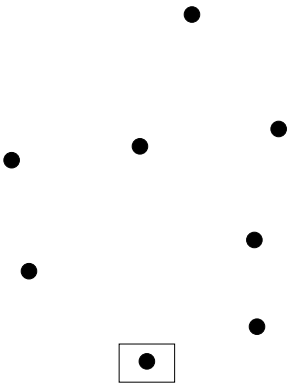
- Melhor não aceitar a máquina de dinheiro.
- Problemas combinatórios são difíceis.

6.3 Aplicações

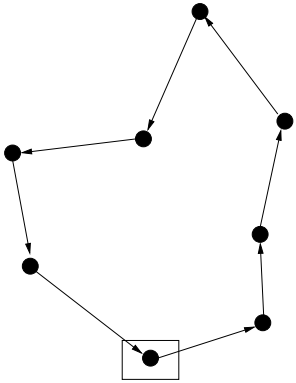
Apanhado de problemas de otimização combinatória

- Caixeiro viajante
- Roteamento
- Projeto de redes
- Alocação de horários
- Tabelas esportivas
- Gestão da produção
- etc.

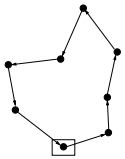
Caixeiro Viajante



Caixeiro Viajante



Caixeiro Viajante



- Humanos são capazes de produzir boas soluções em pouco tempo!
- Humanos ?

Caixeiro Viajante

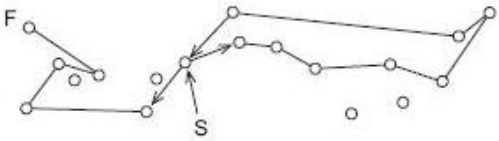


Figure 1.40 Chimpanzee tour (Bido).

Caixeiro Viajante

[1]Retirado de: "The Traveling Salesman Problem: A Computational Study" David L. Applegate, Robert E. Bixby, Vasek Chvátal & William J. Cook. Princeton University Press

Figure 1.41 Pigeon solving a TSP. Images courtesy of Brett Gibson.

Caixeiro Viajante

Der
Handlungsreisende
wie er sein soll
und was er zu thun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiß zu sein.

Von
einem alten Commis - Voyageur.

Mit einem Titelkupfer.

Ilmenau 1832,
Druck und Verlag von B. Fr. Voigt.

Caixeiro Viajante

- Business leads the traveling salesman here and there, and there is not a good tour for all occurring cases; but through an expedient choice division of the tour so much time can be won that we feel compelled to give guidelines about this. Everyone should use as much of the advice as he thinks useful for his application. We believe we can ensure as much that

[1]Retirado de: "The Traveling Salesman Problem: A Computational Study" David L. Applegate, Robert E. Bixby, Vasek Chvátal & William J. Cook. Princeton University Press

[1]Retirado de: "The Traveling Salesman Problem: A Computational Study" David L. Applegate, Robert E. Bixby, Vasek Chvátal & William J. Cook. Princeton University Press

> it will not be possible to plan the tours through Germany in consideration of the distances and the traveling back and fourth, which deserves the traveler's special attention, with more economy. The main thing to remember is always to visit as many localities as possible without having to touch them twice.

"Der Handlungsreisende wie er sein soll und was er zu tun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiss zu sein. Von einem alten Commis-Voyageur" (O caixeiro viajante, como ele deve ser e o que ele deve fazer para obter encomendas e garantir um sucesso feliz dos seus negócios. Por um caixeiro viajante experiente).

First brought to the attention of the TSP research community in 1983 by Heiner Muller-Merbach [410]. The title page of this small book is shown in Figure 1.1. The Commis-Voyageur [132] explicitly described the need for good tours in the following passage, translated from the German original by Linda Cook.

Caixeiro Viajante

Caixeiro Viajante

[1]Retirado de: "The Traveling Salesman Problem: A Computational Study" David L. Applegate, Robert E. Bixby, Vasek Chvátal & William J. Cook. Princeton University Press

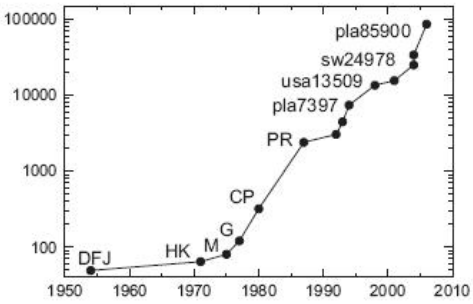
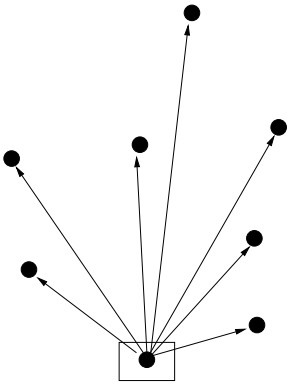


Figure 1.45 Further progress in the TSP, log scale.

Formulando matemáticamente o PCV

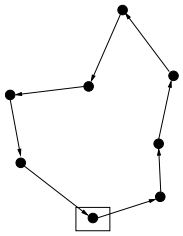
- Associar uma variável a cada possível decisão.



Formulando matemáticamente o PCV

- Associar uma variável a cada possível decisão.

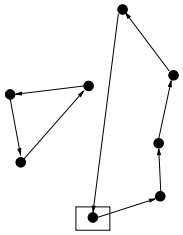
[1]Retirado de: "The Traveling Salesman Problem: A Computational Study" David L. Applegate, Robert E. Bixby, Vasek Chvátal & William J. Cook. Princeton University Press



$$
\begin{aligned}
\textbf{minimiza} \quad & c_{ij}y_{ij} \\
\textbf{sujeito a} \quad & \sum_{j\in N} x_{ij} + \sum_{j\in N} x_{ji} = 2, \qquad \forall i \in N \\
& x_{ij} \in \{0,1\}, \qquad \forall i,j \in N.
\end{aligned}
$$

Formulando matemáticamente o PCV

- Associar uma variável a cada possível decisão.



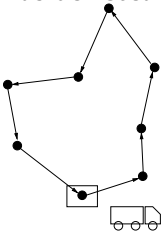
$$
\begin{aligned}
\textbf{minimiza} \quad & c_{ij}y_{ij} \\
\textbf{sujeito a} \quad & \sum_{j\in N} x_{ij} + \sum_{j\in N} x_{ji} = 2, \qquad \forall i \in N \\
& x_{ij} \in \{0,1\}, \qquad \forall i,j \in N.
\end{aligned}
$$

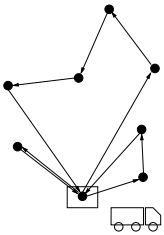
\+ restrições de eliminação de subciclos!

Apanhado de problemas de otimização combinatória

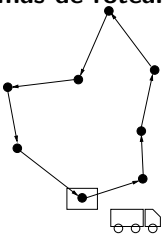
- Caixeiro viajante
- Roteamento
- Projeto de redes
- Alocação de horários
- Tabelas esportivas
- Gestão da produção
- etc.

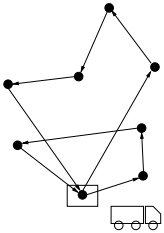
Problemas de roteamento

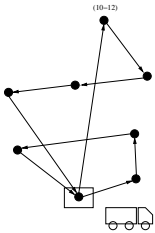


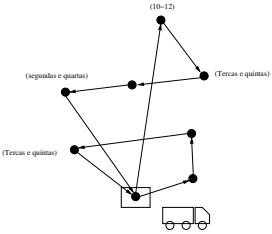


Problemas de roteamento







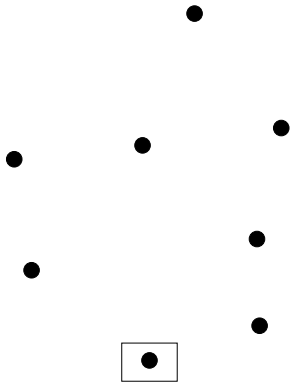


Etc.

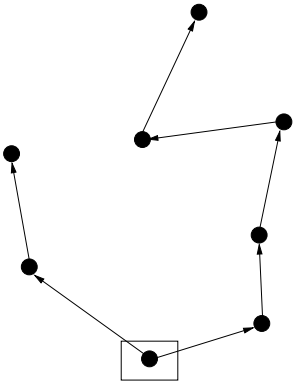
Apanhado de problemas de otimização combinatória

- Caixeiro viajante
- Roteamento
- Projeto de redes
- Alocação de horários
- Tabelas esportivas
- Gestão da produção
- etc.

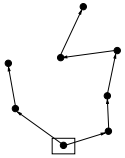
Problemas em árvores



Problemas em árvores



Problemas em árvores - aplicações



- Telecomunicações
- Redes de acesso local
- Engenharias elétrica, civil, etc..

Apanhado de problemas de otimização combinatória

- Caixeiro viajante
- Roteamento
- Projeto de redes
- Alocação de horários
- Tabelas esportivas
- Gestão da produção
- etc.

Alocação de tripulações

Apanhado de problemas de otimização combinatória

- Caixeiro viajante
- Roteamento
- Projeto de redes
- Alocação de horários
- Tabelas esportivas
- Gestão da produção
- etc.

Tabelas esportivas

Proximos Adversários

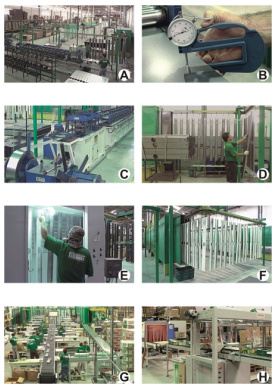
Fla	Vasco	Paysandu	Criciuma	Vitória
JUVENTUDE	Ponte	Coritiba	**GALO**	**CORINTHIANS**
Guarani	**CRUZEIRO**	**PALMEIRAS**	Santos	Juventude
GALO	São Paulo	Paraná	**FURACÃO**	**GUARANI**
Botafogo	**GOIÁS**	**CRICIÚMA**	Paysandu	Grêmio
PALMEIRAS	Juventude	Santos	**PONTE**	**COXA**
Coritiba	**CORINTHIANS**	**GALO**	Paraná	São Paulo
S. PAULO	Furacão	Guarani	**PALMEIRAS**	**CRUZEIRO**
Cruzeiro	**SANTOS**	**JUVENTUDE**	Coxa	Ponte

Botafogo	Galo	Paraná	Grêmio	Guarani
Cruzeiro	Criciúma	**S.CAETANO**	Palmeiras	Goiás
S. PAULO	**GOIÁS**	Grêmio	**PARANÁ**	**FLA**
Coxa	Fla	**PAYSANDU**	Ponte	Vitória
FLA	**PARANÁ**	Galo	**VITÓRIA**	**PALMEIRAS**
Guarani	**FIGUEIRA**	Goiás	Furacão	**BOTAFOGO**
JUVENTUDE	Paysandu	**CRICIÚMA**	**SANTOS**	Figueira
Corinthians	**GRÊMIO**	Flu	Galo	**PAYSANDU**
FURACÃO	S. Caetano	**INTER**	**GUARANI**	Grêmio

Apanhado de problemas de otimização combinatória

- Caixeiro viajante
- Roteamento
- Projeto de redes
- Alocação de horários
- Tabelas esportivas
- Gestão da produção
- etc.

Gestão da produção



Etc.

- programação de projetos
- rotação de plantações
- alocação de facilidades (escolas, centros de comércio, ambulâncias...)
- projeto de circuitos integrados
- portfolio de ações
- etc, etc, etc, etc...

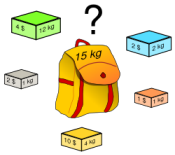
7 Formulação

7.1 Exemplos

"Regras de formulação"

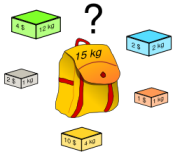
- Criar (boas) formulações é uma arte.
- Algumas diretivas básicas:
  - escolha das variáveis **de decisão**.
  - escolha do objetivo.
  - ajuste das restrições.

Formulação - Problema da mochila



- itens $N = \{1, 2, ...n\}$
- peso de cada ítem: p_i, valor de cada ítem: v_i
- Levar o maior valor possível, dada a restrição de peso.
- Variáveis de decisão ?

Formulação - Problema da mochila



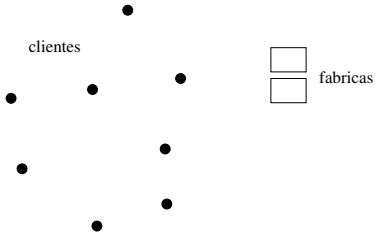
$$\text{Max } v_i x_i$$

s.t.

$$\sum_{i \in N} p_i x_i \leq P$$
$$x_i \in \{0, 1\}$$

Formulação - Problema de locação de facilidades não-capacitado

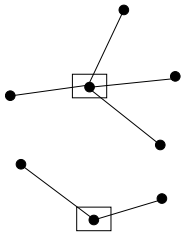
- Alocar fábricas a cidades, de modo a minimizar o custo total de instalação das fábricas e custo de transporte do produto até o cliente



- Cada ponto $i = \{1, 2, ...n\}$ apresenta um custo de instalação da fábrica f_i
- Entre cada par de cidade, (i, j), o custo de transporte é dado por c_{ij}

Formulação - Problema de locação de facilidades não-capacitado

- Exemplo:



- Variáveis de decisão ?

Para formulação escolhemos variáveis de decisão $x_{ij} \in \mathbb{B}$, que indicam se o cliente i for atendido pela fábrica em j.

Formulação - Problema de locação de facilidades não-capacitado

$$
\begin{array}{lll}
\textbf{minimiza} & \displaystyle\sum_{1\le j\le n} f_j y_j + \sum_{1\le i,j\le n} c_{ij} x_{ij} & \\
\textbf{sujeito a} & \displaystyle\sum_{1\le j\le n} x_{ij} = 1, & 1 \le i \le n \quad \text{(só uma fábrica atende)} \\
& \displaystyle\sum_{1\le j\le n} y_j, \le m & \text{(no máximo } m \text{ fábricas)} \\
& x_{ij} \le y_j, & 1 \le i,j \le n \quad \text{(só fábricas existentes atendem)} \\
& x_{ij} \in \mathbb{B}, & 1 \le i,j \le n \\
& y_j \in \mathbb{B}, & 1 \le j \le n.
\end{array}
$$

Alternativas:

- Para instalar exatamente m fábricas: $\sum y_j = m$.

7.2 Técnicas

Formulação: Indicadores

- Variáveis indicadores $x \in \mathbb{B}$: Seleção de um objeto.
- Implicação (limitada): Se x for selecionado, então y deve ser selecionado

$$x \le y \qquad x, y \in \mathbb{B}$$

- Ou:

$$x + y \ge 1 \qquad x, y \in \mathbb{B}$$

- Ou-exlusivo:

$$x + y = 1 \qquad x, y \in \mathbb{B}$$

- Em geral: Seleciona n de m itens $x_1, \ldots, x_m \in \mathbb{B}$

$$\sum_i x_i \left\{ \begin{matrix} = \\ \ge \end{matrix} \right\} n$$

Formulação: Indicadores
Para $x, y, z \in \mathbb{B}$

- Conjunção $x = yz = y \wedge z$

$$\begin{aligned} x &\leq (y+z)/2 \\ x &\geq y + z - 1 \end{aligned}$$

- Disjunção $x = y \vee z$

$$\begin{aligned} x &\geq (y+z)/2 \\ x &\leq y + z \end{aligned}$$

- Negação $x = \neg y$

$$x = 1 - y$$

Formulação: Função objetivo não-linear

- Queremos minimizar custos, com uma "entrada" fixa c

$$f(x) = \begin{cases} 0 & x = 0 \\ c + l(x) & 0 < x \leq \bar{x} \end{cases}$$

  com $l(x)$ linear.

- Solução?

$$\begin{aligned} f(x) &= cy + l(x) \\ x &\leq \bar{x}y \\ x &\in R, y \in \mathbb{B} \end{aligned}$$

- Disjunção de equações: Queremos que aplica-se uma das equações

$$\begin{aligned} f_1 &\leq f_2 \\ g_1 &\leq g_2 \end{aligned}$$

- Solução, com constante M suficientemente grande

$$\begin{aligned} f_1 &\leq f_2 + Mx \\ g_1 &\leq g_2 + M(1-x) \\ x &\in \mathbb{B} \end{aligned}$$

Exemplo
Planejamento de produção (ingl. uncapacitated lot sizing)

- Objetivo: Planejar a futura produção no próximos n semanas.
- Parametros: Para cada semana i
  - Custo fixo f_i para produzir,
  - Custo p_i para produzir uma unidade,
  - Custo h_i por unidade para armazenar,
  - Demanda d_i

Exemplo
Seja

- x_i a quantidade produzido,
- s_i a quantidade no estoque no final da semana i,
- $y_i = 1$ sem tem produção na semana i, 0 senão.

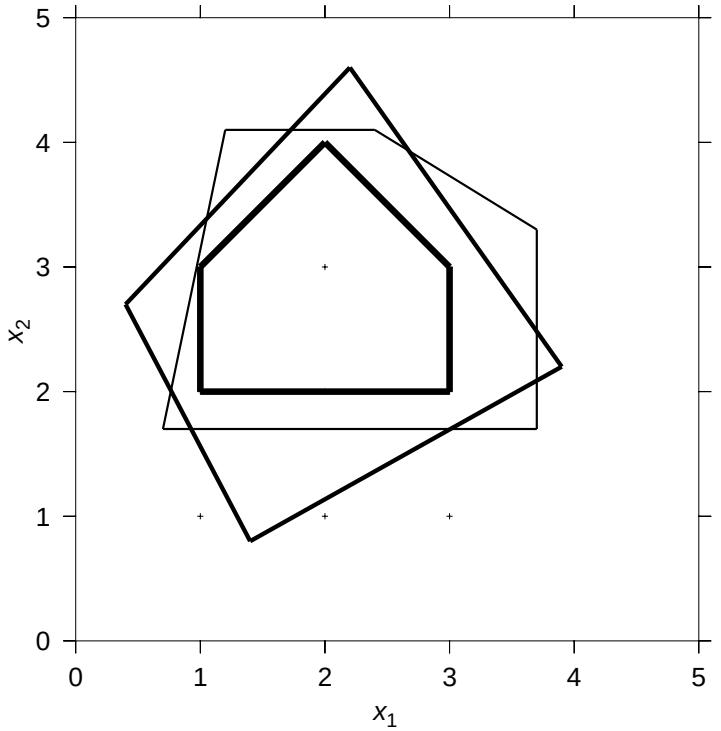
Problema:

- Função objetivo tem custos fixos, mas x_i não tem limite.
- Determina ou estima um valor limite M.

Exemplo

$$
\begin{array}{lll}
\textbf{minimiza} & \sum_i p_i x_i + \sum_i h_i s_i + \sum_i f_i y_i & \\
\textbf{sujeito a} & s_i = s_{i-1} + x_i - d_i, & 1 \leq i \leq n \\
& s_0 = 0 & \\
& x_i \leq M y_i, & 1 \leq i \leq n \\
& x \in \mathbb{R}^n, y \in \mathbb{B}^n. &
\end{array}
$$

Formulações diferentes



As formulações acima são limitadas. Por exemplo, podemos escrever $x \leq y$ para formular uma implicação $x \to y$ para duas variáveis booleanas. Isso não tem generalização simples para outros casos. Por exemplo qual seria uma formulação de

$$x_1 \to x_2 + x_3 \leq 4$$
$$x_1 \in \mathbb{B}, x_2, x_3 \in \mathbb{R},$$

ou seja, a equação $x_2 + x_3 \leq 4$ só precisa ser satisfeita, quando a variável booleana x_1 é verdadeira? Uma abordagem natural é

$$x_1(x_2 + x_3) \leq 4 \Leftrightarrow x_1x_2 + x_1x_3 \leq 4$$

mas, infelizmente, gera uma restrição *quadrática*, porque contém produtos de variáveis como x_1x_3. Como programas quadráticas são consideravelmente mais complicados de resolver, o nosso objetivo é formular problemas *lineares*. Logo, a formulação acima não serve.

Caso conhecemos limites superiores

$$x_2 \leq u_2$$
$$x_3 \leq u_3$$

sabemos também que $x_2 + x_3 \leq u_2 + u_3$. TBD TBD

$$x_2 + x_3 \leq 4 + (1 - x_1)M$$

com $M := u_2 + u_3$.

8 Técnicas de solução

8.1 Introdução

Limites

- Exemplo: Problema de maximização.
- Limite inferior (limite primal): Cada solução viável.
  - Qualquer técnica construtiva, p.ex. algoritmos gulosos, heurísticas etc.
- Limite superior (limite dual): Essentialmente usando uma relaxação
  - Menos restrições $\Rightarrow$ conjunto maior de solução viáveis.
  - Nova função objetivo que é maior ou igual.
- Importante: Relaxação linear: $x \in \mathbb{Z} \Rightarrow x \in \mathbb{R}$.

8.2 Problemas com solução eficiente

Relaxação inteira

- Solução simples: A relaxação linear possui solução ótima inteira.
- Como garantir?
- Com base B temos a solução $x = (x_B\ x_N)^t = (B^{-1}b, 0)^t$.
- Observação: Se $b \in \mathbb{Z}^m$ e $|\det(B)| = 1$ para a base ótima, então o PL resolve o PI.

Lembrança: Determinante usando Laplace

$$\det(A) = \sum_{1 \le i \le n} (-1)^{i+j} a_{ij} \det(A_{ij}) = \sum_{1 \le j \le n} (-1)^{i+j} a_{ij} \det(A_{ij})$$

com A_{ij} a submatriz sem linha i e coluna j.

Relaxação inteira

- Para ver isso: Regra de Cramer.

- A solução de $Ax = b$ é

$$x_i = \frac{\det(A_i)}{\det(A)}$$

com A_i a matriz resultante da substituição da i-gésima coluna de A por b.

Prova. Seja U_i a matriz identidade com a i-gésima coluna substituído por x, i.e.

$$\begin{pmatrix} 1 & & & x_1 & & \\ & 1 & & x_2 & & \\ & & \ddots & \vdots & & \\ & & & x_{n-1} & \ddots & \\ & & & x_n & & 1 \end{pmatrix}$$

É simples verificar que $AU_i = A_i$. Com $\det(U_i) = x_i$ e $\det(A)\det(U_i) = \det(A_i)$ temos o resultado. ■

Exemplo: Regra de Cramer

$$\begin{pmatrix} 3 & 2 & 1 \\ 5 & 0 & 2 \\ 2 & 1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Exemplo: Regra de Cramer

$$\begin{vmatrix} 3 & 2 & 1 \\ 5 & 0 & 2 \\ 2 & 1 & 2 \end{vmatrix} = -13$$

$$\begin{vmatrix} 1 & 2 & 1 \\ 1 & 0 & 2 \\ 1 & 1 & 2 \end{vmatrix} = -1$$

$$\begin{vmatrix} 3 & 1 & 1 \\ 5 & 1 & 2 \\ 2 & 1 & 2 \end{vmatrix} = -3$$

$$\begin{vmatrix} 3 & 2 & 1 \\ 5 & 0 & 1 \\ 2 & 1 & 1 \end{vmatrix} = -4$$

Logo $x_1 = 1/13; x_2 = 3/13; x_3 = 4/13$.

Aplicação da regra de Cramer

- Como garantir que $x = B^{-1}b$ é inteiro?
- Cramer:
$$x_i = \frac{\det(B_i)}{\det(B)}$$
- Condição possível: (a) $\det(B_i)$ inteiro, (b) $\det(B) \in \{-1, 1\}$.
- Garantir (a): $A \in \mathbb{Z}^{m \times n}$ e $b \in \mathbb{Z}^m$.
- Garantir (b): Toda submatriz quadrática não-singular de A tem determinante $\{-1, 1\}$.

Exemplo 8.1
Observe que essas condições são suficientes, mas não necessárias. É possível que $Bx = b$ possui solução inteira sem essas condições ser satisfeitas. Por exemplo

$$\begin{pmatrix} 2 & 2 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

tem a solução inteira $(x_1 x_2) = (10)$, mesmo que $\det(A) = -2$. ◊

A relaxação é inteira

Definição 8.1
Uma matriz quadrática inteira $A \in \mathbb{R}^{n \times n}$ é *unimodular* se $|\det(A)| = 1$. Uma matriz arbitrária A é *totalmente unimodular* (TU) se cada submatriz quadrada não-singular A' de A é modular, i.e. $\det(A') \in \{0, 1, -1\}$.

Uma consequência imediata dessa definição: $a_{ij} \in \{-1, 0, 1\}$.

Exemplo
Quais matrizes são totalmente unimodular?

$$\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}; \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & -1 & -1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{pmatrix}; \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Critérios

Proposição 8.1
Se A é TU então

1. A^t é TU.
2. $(A\ I)$ com matriz de identidade I é TU.
3. Uma matriz B que é uma permutação das linhas ou colunas de A é TU.
4. Multiplicando uma linha ou coluna com -1 resulta numa matriz TU.

Prova. (i) Qualquer submatriz quadrada B^t de A^t e uma submatriz B de A também. Com $\det(B) = \det(B^t)$, segue que A^t é totalmente unimodular. (ii) Qualquer submatriz de (AI) tem a forma $(A'I')$ com A' submatriz de A e I' submatriz de I. Com $|\det(A'I')| = |\det(A')|$ segue que (AI) é TU. (iii) Cada submatriz de B é uma submatriz de A. (iv) A determinate troca no máximo o sinal. ■

Critérios

Proposição 8.2
Uma matriz A é totalmente unimodular se

1. $a_{ij} \in \{+1, -1, 0\}$
2. Cada coluna contém no máximo dois coeficientes não-nulos.
3. Existe uma partição de linhas $M_1 \dot{\cup} M_2 = [1, m]$ tal que cada coluna *com dois coeficientes não-nulos* satisfaz

$$\sum_{i \in M_1} a_{ij} - \sum_{i \in M_2} a_{ij} = 0$$

Observe que esse critério é suficiente, mas não necessário.

Exemplo

$$\begin{pmatrix} 1 & -1 & -1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{pmatrix}$$

- Coeficientes $\in \{-1, 0, 1\}$: Sim.
- Cada coluna no máximo dois coeficientes não-nulos: Sim.
- Partição M_1,M_2? Sim, escolhe $M_1 = [1, 3], M_2 = \emptyset$.

Exemplo

$$A = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

TU?
Não: $\det(A) = 2$.

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

TU?

Não: $\det(A) = 2$.

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

TU? Sim. Mas nossa regra não se aplica!

Prova. (Proposição 8.2). Prova por contradição. Seja A uma matriz que satisfaz os critérios da proposição 8.2, e seja B o menor submatriz quadrada de A tal que $\det(B) \notin \{0, +1, -1\}$. B não contém uma coluna com um único coeficiente não-nula: seria uma contradição com a minimalidade do B (removendo a linha e a coluna que contém esse coeficiente, obtemos uma matriz quadrada menor B^*, que ainda satisfaz $\det(B^*) \notin \{0, +1, -1\}$). Logo, B contém dois coeficientes não-nulos em cada coluna. Aplicando a condição (3) acima, subtraindo as linhas com índice em M_1 das linhas com índice em M_2 podemos ver as linhas do B são linearmente dependentes e portanto temos $\det(B) = 0$, uma contradição. ■

Consequências

Teorema 8.1 (Hoffman,Kruskal)
Se a matriz A de um programa linear é totalmente unimodular e o vetor b é inteiro, todas soluções básicas são inteiras. Em particular as regiões

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

tem pontos extremos inteiros.

Prova. Considerações acima. ■

Exemplo 8.2 (Caminhos mais curtos)

Exemplo: Caminhos mais curtos

- Dado um grafo não-direcionado $G = (V, A)$ com custos $c : A \to \mathbb{Z}$ nos arcos.
- Qual o caminho mais curto entre dois nós $s, t \in V$?

Exemplo: Caminhos mais curtos

$$
\begin{aligned}
\textbf{minimiza} \quad & \sum_{a \in A} c_a x_a \\
\textbf{sujeito a} \quad & \sum_{a \in N^+(s)} x_a - \sum_{a \in N^-(s)} x_a = 1 \\
& \sum_{a \in N^+(v)} x_a - \sum_{a \in N^-(a)} x_a = 0, \qquad \forall v \in V \setminus \{s, t\} \\
& \sum_{a \in N^+(t)} x_a - \sum_{a \in N^-(t)} x_a = -1 \\
& x_a \in \mathbb{B}, \qquad \forall a \in A.
\end{aligned}
$$

A matriz do sistema acima de forma explicita:

$$
\begin{array}{c}
s \\ \\ \vdots \\ \\ t
\end{array}
\begin{pmatrix}
1 & \cdots & & \cdots & -1 \\
 & & 1 & & \\
\vdots & & & & \\
 & & -1 & & -1 \\
-1 & \cdots & & &
\end{pmatrix}
\begin{pmatrix}
x_{a_1} \\ \\ \vdots \\ \\ x_{a_m}
\end{pmatrix}
=
\begin{pmatrix}
1 \\ 0 \\ \vdots \\ 0 \\ 1
\end{pmatrix}
$$

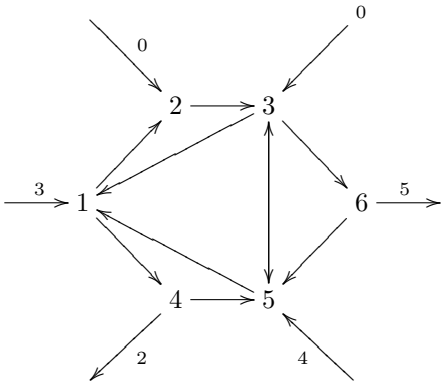
Como cada arco é adjacente ao no máximo dois vértices, e cada coluna contém um coeficiente 1 e -1, a Proposição 8.2 é satisfeito com a partição trivial. $\Diamond$

Exemplo 8.3 (Fluxo em redes)

Exemplo: Fluxo em redes

- Dado: Um grafo direcionado $G = (V, A)$
  - com arcos de capacidade limitada $l : A \to \mathbb{Z}^+$,

  - demandas $d : V \to \mathbb{Z}$ dos vértices,
  - (com $d_v < 0$ para destino e $d_v > 0$ nos fonte)
  - e custos $c : A \to \mathbb{R}$ por unidade de fluxo nos arcos.
- Qual o fluxo com custo mínimo?



Exemplo: Fluxo em redes

$$\begin{aligned}
\textbf{minimiza} \quad & \sum_{a \in A} c_a x_a \\
\textbf{sujeito a} \quad & \sum_{a \in N^+(v)} x_a - \sum_{a \in N^-(v)} x_a = d_v, && \forall v \in V \\
& 0 \leq x_a \leq l_a, && \forall a \in A.
\end{aligned}$$

com conjunto de arcos entrantes $N^-(v)$ e arcos saintes $N^+(v)$.

Exemplo: Fluxo

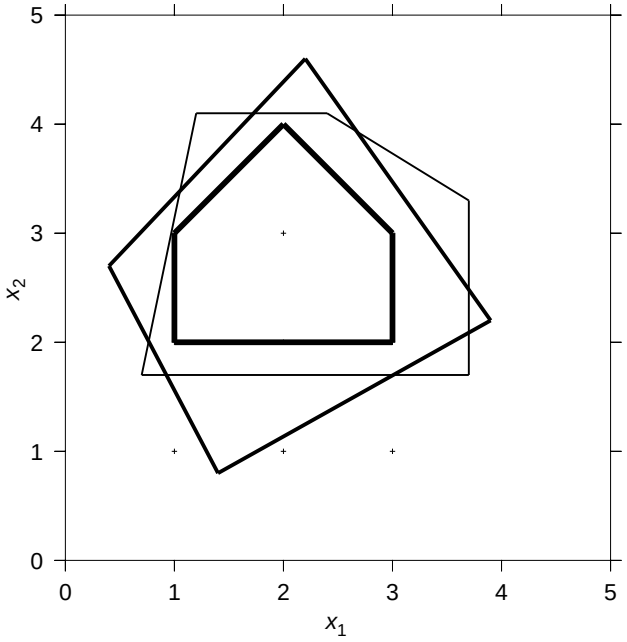
- A matriz que define um problema de fluxo é totalmente unimodular.
- Consequências
  - Cada ponto extremo da região víavel é inteira.
  - A relaxação PL resolve o problema.
- Existem vários subproblemas de fluxo mínimo que podem ser resolvidos também, p.ex. fluxo máximo entre dois vértices.

$\Diamond$

8.3 Desigualdades válidas

Desigualdades válidas

- Problema inteiro
$$\max\{c^t x | Ax \leq b, x \in Z_+^n\}$$
- Relaxação linear
$$\max\{c^t x | Ax \leq b, x \in R_+^n\}$$



Desigualdades válidas

Definição 8.2
Uma desigualdade $\pi x \leq \pi_0$ é *válida* para um conjunto P, se $\forall x \in P : \pi x \leq \pi_0$.

- Como achar desigualdades (restrições) válidas para o conjunto da soluções viáveis $\{x | Ax \leq b, x \in Z_+^n\}$ de um problema inteiro?
  - Técnicas de construção (p.ex. método de Chvátal-Gomory)
  - Observar e formalizar características específicas do problema.

– "The determination of families of strong valid inequalities is more of an art than a formal methodology" [6, p. 259]

Exemplo 8.4 (Locação de facilidades não-capacitado)

$$\textbf{minimiza} \quad \sum_{1\le j\le n} f_j y_j + \sum_{1\le i,j\le n} c_{ij}x_{ij} \tag{8.1}$$

$$\textbf{sujeito a} \quad \sum_{1\le j\le n} x_{ij} = 1, \qquad \forall i = 1...n \tag{8.2}$$

$$x_{ij} \le y_j, \qquad \forall i,j = 1...n \tag{8.3}$$

$$x_{ij} \in \mathbb{B}, \qquad i,j = 1,...,n \tag{8.4}$$

$$y_j \in \mathbb{B}, \qquad j = 1,...,n. \tag{8.5}$$

Ao invés de

$$x_{ij} \le y_j \tag{8.6}$$

podemos pensar em

$$\sum_{1\le i\le n} x_{ij} \le n y_j. \tag{8.7}$$

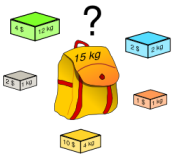
Essa formulação ainda é correto, mas usa n restrições ao invés de n^2. Entretanto, a qualidade da relação linear é diferente. É simples ver que podemos obter (8.7) somando (8.6) sobre todos i. Portanto, qualquer solução que satisfaz (8.6) satisfaz (8.7) também, e dizemos que (8.6) *domina* (8.7).
Que o contrário não é verdadeiro, podemos ver no seguinte exemplo: Com custos de instalação $f_j = 1$, de transporte $c_{ij} = 5$ para $i \neq j$ e $c_{ii} = 0$, duas cidades e uma fábrica obtemos as duas formulações (sem restrições de integralidade)

minimiza	$y_1 + y_2 + 5c_{12} + 5c_{21}$	$y_1 + y_2 + 5c_{12} + 5c_{21}$
sujeito a	$x_{11} + x_{12} = 1$	$x_{11} + x_{12} = 1$
	$x_{21} + x_{22} = 1$	$x_{21} + x_{22} = 1$
	$y_1 + y_2 \le 1$	$y_1 + y_2 \le 1$
	$x_{11} \le y_1$	$x_{11} + x_{21} \le 2y_1$
	$x_{12} \le y_2$	
	$x_{21} \le y_1$	$x_{21} + x_{22} \le 2y_2$
	$x_{22} \le y_2$	

A solução ótima da primeira é $y_1 = 1, x_{11} = x_{21} = 1$ com valor 6, que é a solução ótima inteira. Do outro lado, a solução ótima da segunda formulação é $y_1 = y_2 = 0.5$ com $x_{11} = x_{22} = 1$, com valor 1, i.e. ficam instaladas duas "meia-fábricas" nas duas cidades!

$\Diamond$

Exemplo: 0-1-**Knapsack**



$$\begin{aligned} \textbf{maximiza} \quad & \sum_{1 \le i \le n} v_i x_i \\ \textbf{sujeito a} \quad & \sum_{1 \le i \le n} p_i x_i \le P \\ & x_i \in \mathbb{B} \end{aligned}$$

Exemplo: $79x_1 + 53x_2 + 53x_3 + 45x_4 + 45x_5 \le 178$.

Exemplo: 0-1-**Knapsack**

- Observação: Para um subconjunto $S \subset [1, n]$: Se $\sum_S p_i > P$ então $\sum_S x_i \le |S| - 1$.
- Exemplos:

$$\begin{aligned} x_1 + x_2 + x_3 &\le 2 \\ x_1 + x_2 + x_4 + x_5 &\le 3 \\ x_1 + x_3 + x_4 + x_5 &\le 3 \\ x_2 + x_3 + x_4 + x_5 &\le 3 \end{aligned}$$

Exemplo: Casamento

- Dado um grafo $G = (V, A)$ procuramos um *casamento* máximo, i.e. um subconjunto $C \subseteq A$ tal que $\delta(v) \le 1$ para $v \in V$.

- Programa inteiro

$$\begin{aligned} \textbf{maximiza} \quad & \sum_{A} x_a & \\ \textbf{sujeito a} \quad & \sum_{u \in N(v)} x_{(u,v)} \leq 1, & \forall v \in V \\ & x_a \in \mathbb{B}, & \forall a \in A. \end{aligned}$$

Exemplo: Casamento

- Escolhe um subconjunto de nós $U \subseteq V$ arbitrário.
- Observação: O número de arestas internas é $\leq \lfloor |U|/2 \rfloor$.
- Portanto:

$$\sum_{a \in U^2 \cap A} x_a \leq \lfloor |U|/2 \rfloor$$

  é uma desigualdade válida.

Método de Chvátal-Gomory

Dado

$$\sum_i a_i x_i \leq b$$

também temos, para $u \in \mathbb{R}, u > 0$ as restrições válidas

$$\begin{aligned} \sum_i u a_i x_i &\leq ub && \text{(multiplicação)} \\ \sum_i \lfloor u a_i \rfloor \, x_i &\leq ub && \lfloor y \rfloor \leq y, 0 \leq x_i \\ \sum_i \lfloor u a_i \rfloor \, x_i &\leq \lfloor ub \rfloor && \text{Lado esquerda é inteira.} \end{aligned}$$

Método de Chvátal-Gomory

Teorema 8.2
Todas desigualdades válidadas pode ser construída através de um número finito de aplicações do método de Chvátal-Gomory.

Exemplo: Casamento

- Para um $U \subseteq V$ podemos somar as desigualdades

$$\sum_{u \in N(v)} x_{(u,v)} \leq 1 \qquad \forall v \in V$$

  com peso 1/2, obtendo

$$\sum_{a \in U^2 \cap A} x_a + \frac{1}{2} \sum_{a \in N(U)} x_a \leq \frac{1}{2}|U|$$

- Também temos

$$\frac{1}{2} \sum_{a \in N(U)} x_a \geq 0$$

- Portanto

$$\sum_{a \in U^2 \cap A} x_a \leq \frac{1}{2}|U|$$

$$\sum_{a \in U^2 \cap A} x_a \leq \left\lfloor \frac{1}{2}|U| \right\rfloor \qquad \text{Lado esquerdo inteiro}$$

8.4 Planos de corte

Como usar restrições válidas?

- Adicionar à formulação antes de resolver.
  - Vantagens: Resolução com ferramentas padrão.
  - Desvantagens: Número de restrições pode ser grande ou demais.
- Adicionar ao problema se necessário: Algoritmos de plano de corte.
  - Vantagens: Somente cortes que ajudam na solução da instância são usados.

Planos de corte

Problema inteiro

$$\max\{c^t x | Ax \leq b, x \in Z_+^n\}$$

- O que fazer, caso a relaxação linear não produz soluções ótimas?
- Um método: Introduzir *planos de corte*.

Definição 8.3
Um plano de corte (ingl. cutting plane) é uma restrição válida (ingl. valid inequality) que todas soluções inteiras satisfazem.

Algoritmo de planos de corte

PLANOS DE CORTE

Entrada Programa inteiro $\max\{c^t x | Ax \leq b, x \in Z_+^n\}$.

Saida Solução inteira ótima ou "Não existe corte.".

```
1  V := {x|Ax ≤ b}                          { região viável }
2  x* := argmax{c^t x|x ∈ V}  { resolve relaxação }
3  while (x* ∉ Z_+^n) do
4     if (existe corte a^t x ≤ d com a^t x* > d) then
5        V := V ∩ {x|a^t x ≤ d}  { nova região viável }
6        x* := argmax{c^t x|x ∈ V}  { nova solução ótima }
7     else
8        return "Não existe corte."
9     end if
10 end while
```

Método de Gomory

- Como achar um novo corte na linha 4 do algoritmo?
- A solução ótima atual é representado pelo dicionário

$$z = \bar{z} + \sum_j \bar{c}_j x_j$$

$$x_i = \bar{b}_i - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j \qquad i \in \mathcal{B}$$

- Se a solução não é inteira, existe um índice i tal que $x_i \notin \mathbb{Z}_+$, i.e. $\bar{b}_i \notin \mathbb{Z}_+$.

Cortes de Chvátal-Gomory

$$x_i = \bar{b}_i - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j \qquad \text{Linha fracionária} \qquad (8.8)$$

$$x_i \leq \bar{b}_i - \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j \qquad \text{Definição de } \lfloor \cdot \rfloor \qquad (8.9)$$

$$x_i \leq \lfloor \bar{b}_i \rfloor - \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j \qquad \text{Integralidade de } x \qquad (8.10)$$

$$0 \geq \{\bar{b}_i\} - \sum_{j \in \mathcal{N}} \{\bar{a}_{ij}\} x_j \qquad (8.8) - (8.10) \qquad (8.11)$$

$$x_{n+1} = -\{\bar{b}_i\} + \sum_{j \in \mathcal{N}} \{\bar{a}_{ij}\} x_j \qquad \text{Nova variável} \qquad (8.12)$$

$$x_{n+1} \in \mathbb{Z}_+ \qquad (8.13)$$

(Para soluções inteiras, a diferença do lado esquerdo e do lado direito na equação (8.10) é inteira. Portanto x_{n+1} também é inteira.)
A solução básica atual não satisfaz (8.11), porque com $x_j = 0, j \in \mathcal{N}$ temos que satisfazer

$$\{\bar{b}_i\} \leq 0,$$

uma contradição com a definição de $\{\cdot\}$ e o fato que $\bar{b}_i$ é fracionário. Portanto, provamos

Proposição 8.3
O corte (8.11) satisfaz os critérios da linha 4 do algoritmo Planos de corte. Em particular, sempre existe um corte e o caso da linha 8 nunca se aplica.

Exemplo 8.5
Queremos resolver o problema

$$\begin{aligned} \textbf{maximiza} \quad & x_1 + x_2 \\ \textbf{sujeito a} \quad & -x_1 + 3x_2 \leq 9 \\ & 10x_1 \leq 27 \\ & x_1, x_2 \in \mathbb{Z}_+ \end{aligned}$$

A solução da relaxação linear produz a série de dicionários

$$\begin{array}{llllll} (1) & z & = & x_1 & +x_2 & \\ & w_1 & = 9 & +x_1 & -3\mathbf{x_2} & \\ & w_2 & = 27 & -10x_1 & & \end{array} \qquad \begin{array}{llllll} (2) & z & = 3 & +4/3x_1 & -1/3w_1 & \\ & x_2 & = 3 & +1/3x_1 & -1/3w_1 & \\ & w_2 & = 27 & -10\mathbf{x_1} & & \end{array}$$

$$
\begin{array}{llll}
(3) & z & = 6.6 & -4/30w_2 \quad -1/3w_1 \\
 & x_2 & = 3.9 & -1/30w_2 \quad -1/3w_1 \\
 & x_1 & = 2.7 & -1/10w_2
\end{array}
$$

A solução ótima $x_1 = 2.7$, $x_2 = 3.9$ é fracionária. Correspondendo com a segunda linha

$$x_2 \quad = 3.9 \quad -1/30w_2 \quad -1/3w_1$$

temos o corte

$$w_3 \quad = -0.9 \quad +1/30w_2 \quad +1/3w_1$$

e o novo sistema é

$$
\begin{array}{llll}
(4) & z & = 6.6 & -4/30w_2 \quad -1/3w_1 \\
 & x_2 & = 3.9 & -1/30w_2 \quad -1/3w_1 \\
 & x_1 & = 2.7 & -1/10w_2 \\
 & w_3 & = -0.9 & +1/30w_2 \quad +1/3\mathbf{w_1}
\end{array}
$$

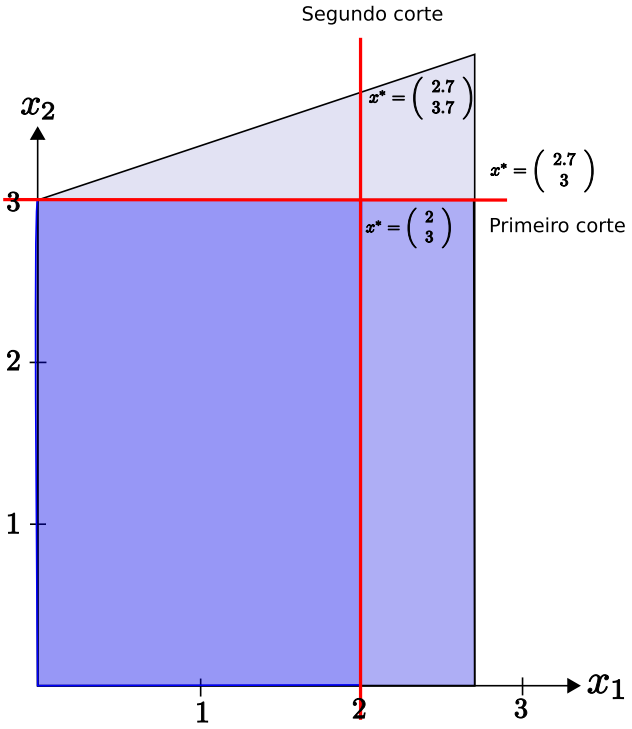
Esse sistema não é mais ótimo, e temos que re-otimizar. Pior, a solução básica atual não é viável! Mas como a na função objetivo todos coeficientes ainda são negativos, podemos aplicar o método Simplex dual. Um pivot dual gera a nova solução ótima

$$
\begin{array}{llll}
(5) & z & = 5.7 & -1/10w_2 \quad -w_3 \\
 & x_2 & = 3 & -w_3 \\
 & x_1 & = 2.7 & -1/10w_2 \\
 & w_1 & = 2.7 & -1/10w_2 \quad +3w_3
\end{array}
$$

com $x_2 = 3$ inteiro agora, mas x_1 ainda fracionário. O próximo corte, que corresponde com x_1 é

$$
\begin{array}{llll}
(5) & z & = 5.7 & -1/10w_2 \quad -w_3 \\
 & x_2 & = 3 & -w_3 \\
 & x_1 & = 2.7 & -1/10w_2 \\
 & w_1 & = 2.7 & -1/10w_2 \quad +3w_3 \\
 & w_4 & = -0.7 & +\mathbf{1/10w_2}
\end{array}
\qquad
\begin{array}{llll}
(6) & z & = 5 & -w_4 \quad -w_3 \\
 & x_2 & = 3 & -w_3 \\
 & x_1 & = 2 & -w_4 \\
 & w_1 & = 2 & -w_4 \quad +3w_3 \\
 & w_2 & = 7 & +10w_4
\end{array}
$$

cuja solução é inteira e ótima. ◊



Resumo: Algoritmos de planos de corte

- O algoritmo de planos de corte, usando os cortes de Gomory termina sempre, i.e. é correto.
- O algoritmos pode ser modificado para programas mistos.
- A técnica *pura* é considerado inferior ao algoritmos de branch-and-bound.
- Mas: Planos de corte em combinação com branch-and-bound é uma técnica poderosa: Branch-and-cut.

8.5 Branch-and-bound

Branch-and-bound
Ramifica-e-limite (ingl. branch-and-bound)

- Técnica geral para problemas combinatoriais.

> Branch and Bound is by far the most widely used tool for solving large scale NP-hard combinatorial optimization problems. [2]

- Idéia básica:
  - Particiona um problema recursivamente em subproblemas disjuntas e procura soluções.
  - Evite percorrer toda árvore de busca, calculando limites e cortando sub-árvores.
- Particularmente efetivo para programas inteiras: a relaxação linear fornece os limites.

Branch-and-bound

- Problema PI (puro): $\{\max c^t x | x \in S, x \in \mathbb{Z}_+^n\}$.
- Resolve a relaxação linear.
- Solução inteira? Problema resolvido.
- Senão: Escolhe uma variável inteira x_i, cuja solução atual v_i é fracionário.
- Tipico: Variável mais fracionário com $\text{argmin}_i |\{x_i\} - 0.5|$.
- Particione o problema $S = S_1 \dot\cup S_2$ tal que

$$S_1 = S \cap \{x | x_i \leq \lfloor v_i \rfloor\}; \quad S_2 = S \cap \{x | x_i \geq \lceil v_i \rceil\}$$

- Em particular com variáveis $x_i \in \mathbb{B}$:

$$S_1 = S \cap \{x | x_i = 0\}; \quad S_2 = S \cap \{x | x_i = 1\}$$

Limitar

- Para cada sub-árvore mantemos um limite inferior e um limite superior.
  - Limite inferior: Valor de uma solução encontrada na sub-árvore.
  - Limite superior: Valor da relaxação linear.
- Observação: A eficiencia do método depende crucialmente da qualidade do limite superior.
- Preferimos formulações mais “rigidos”.

Cortar sub-árvores

1. Corte por inviabilidade: Sub-problema é inviável.
2. Corte por limite: Limite superior da sub-árvore $\overline{z_i}$ menor que limite inferior global $\underline{z}$ (o valor da melhor solução encontrada).
3. Corte por otimalidade: Limite superior $\overline{z_i}$ igual limite inferior $\underline{z_i}$ da sub-árvore.
4. Observação: Como os cortes dependem do limite $\underline{z}$, uma boa solução inicial pode reduzir a busca consideravelmente.

Ramificar

- Não tem como cortar mais? Escolhe um nó e particiona.
- Qual a melhor ordem de busca?
- Busca pro profundidade
  - V: Limite superior encontrado mais rápido.
  - V: Pouco memória ($O(\delta d)$, para δ subproblemas e profundidade d).
  - V: Re-otimização eficiente do pai (método Simplex dual)
  - D: Custo alto, se solução ótima encontrada tarde.
- Melhor solução primeiro ("best-bound rule")
  - V: Procura ramos com maior potencial.
  - V: Depois encontrar solução ótima, não produz ramificações superfluas.
- Busca por largura? Demanda de memória é impraticável.

Algoritmos B&B

B&B

Instância Programa inteiro $P = \max\{c^t x | Ax \leq b, x \in Z_+^n\}$.

Saida Solução inteira ótima.

```
1  { usando função $\overline{z}$ para estimar limite superior }
2  $\underline{z}$:=$-\infty$          { limite inferior }
3  $A$:= $\{(P, g(P))\}$          { nós ativos }
4  while $A \neq \emptyset$ do
5     Escolhe: $(P, g(P) \in A$; $A := A \setminus (P, g(P))$
6     Ramifique: Gera subproblemas $P_1, \ldots, P_n$.
7     for all $P_i$, $1 \leq i \leq n$ do
8       { adiciona, se permite melhor solução }
9       if $\overline{z}(P_i) > \underline{z}$ then
10        $A := A \cup \{(P_i, \overline{z}(P_i))\}$
11      end if
12      { atualize melhor solução }
13      if (solução $\overline{z}(P_i)$ é viável) then
14        $\underline{z} := \overline{z}(P_i)$
15      end if
16    end for
17 end while
```

9 Tópicos

Teorema 9.1 (Lenstra)
The integer programming feasibility problem can be solved with $O(p^{9p/2}L)$ arithmetic operations with integers of $O(p^{2p}L)$ bits in size, where p is the number of ILP variables and L is the number of bits in the input.

Observação: IPF is FPT.

Outras técnicas

- Branch-and-cut.

  Começa com menos restrições (relaxação) e insere restrições (cortes) nos sub-problemas da busca com branch-and-bound.

- Branch-and-price.

  Começa com menos variáveis e insere variáveis ("geração de colunas") nos sub-problemas da busca com branch-and-bound.

10 Exercícios

(Soluções a partir da página 185.)

Exercício 10.1 (Formulação)
A empresa "Festa fulminante" organiza festas. Nos próximos n dias, ela precisa p_i pratos, $1 \leq i \leq n$. No começo de cada dia gerente tem os seguintes opções:

- Comprar um prato para um preço de c reais.
- Mandar lavar um prato devagarmente em d_1 dias, por um preço de l_1 reais.
- Mandar lavar um prato rapidamente em $d_2 < d_1$ dias, por um preço de $l_2 > l_1$ reais.

O gerente quer minimizar os custos dos pratos. Formalize como programa inteira.

Exercício 10.2 (Planos de corte)
Resolve

$$\begin{aligned} \textbf{maximiza} \quad & x_1 + 3x_2 \\ \textbf{sujeito a} \quad & -x_1 \leq -2 \\ & x_2 \leq 3 \\ & -x_1 - x_2 \leq -4 \\ & 3x_1 + x_2 \leq 12 \\ & x_i \in \mathbb{Z}_+ \end{aligned}$$

e

$$\begin{aligned} \textbf{maximiza} \quad & x_1 - 2x_2 \\ \textbf{sujeito a} \quad & -11x_1 + 15x_2 \leq 60 \\ & 4x_1 + 3x_2 \leq 24 \\ & 10x_1 - 5x_2 \leq 49 \\ & x_1, x_2 \in \mathbb{Z}_+ \end{aligned}$$

com o algoritmo de planos de corte using cortes de Chvátal-Gomory.

Exercício 10.3 (Formulação)
Para os problemas abaixo, acha uma formulação como programa inteira.

Conjunto independente máximo

Instância Um grafo não-direcionado $G = (V, A)$.

Solução Um *conjunto independente* I, i.e. $I \subseteq V$ tal que para vértices $v_1, v_2 \in I$, $\{v_1, v_2\} \notin A$.

Objetivo Maximiza $|I|$.

Casamento perfeito com peso máximo

Instância Um grafo não-direcionado bi-partido $G = (V_1 \dot\cup V_2, A)$ (a fato de ser bi-partido significa que $A \subseteq V_1 \times V_2$) com pesos $p : A \to \mathbb{R}$ nos arcos.

Solução Um *casamento perfeito*, i.e. um conjunto de arcos $C \subseteq A$ tal que todos nós no sub-grafo $G[C] = (V_1 \cup V_2, C)$ tem grau 1.

Objetivo Maximiza o peso total $\sum_{c \in C} p(c)$ do casamento.

Problema de transporte

Instância n depósitos, cada um com um estoque de p_i $(1 \leq i \leq n)$ produtos, e m clientes, cada um com uma demanda de d_j $(1 \leq j \leq m)$ produtos. Custos de transporte a_{ij} de cada depósito para cada cliente.

Solução Um decisão quantos produtos x_{ij} devem ser transportados do depósito i ao cliente j, que satisfaz (i) Cada depósito manda todo seu estoque (ii) Cada cliente recebe exatamente a sua demanda. (Observe que o número de produtos transportados deve ser integral.)

Objetivo Minimizar os custos de transporte $\sum_{i,j} a_{ij} x_{ij}$.

Conjunto dominante

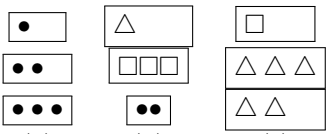
Instância Um grafo não-direcionado $G = (V, A)$.

Solução Um *conjunto dominante*, i.e. um conjunto $D \subseteq V$, tal que $\forall v \in V : v \in D \vee (\exists u \in D : \{u, v\} \in A)$ (cada vértice faz parte do conjunto dominante ou tem um vizinho no conjunto dominante).

Objetivo Minimizar o tamanho do conjunto dominante $|D|$.

Exercício 10.4 (Formulação: Apagando e ganhando)
Juliano é fã do programa de auditório Apagando e Ganhando, um programa no qual os participantes são selecionados atráves de um sorteio e recebem pr6emios em dinheiro por participarem. No programa, o apresentador escreve um número de N dígitos em uma lousa. O participante então deve apagar exatamente D dígitos do número que está na lousa; o número formado pelos dígitos que restaram é então o prêmio do participante. Juliano finalmente foi selecionado para participar do programa, e pediu que você escrevesse um programa inteira que, dados o número que o apresentador escreveu na lousa, e quantos dígitos Juliano tem que apagar, determina o valor do maior prêmio que Juliano pode ganhar.
(Fonte: Maratona de programação regional 2008, RS)

Exercício 10.5 (Formulação: Set)
Set é um jogo jogado com um baralho no qual cada carta pode ter uma, duas ou três figuras. Todas as figuras em uma carta são iguais, e podem ser círculos, quadrados ou triângulos. Um set é um conjunto de três cartas em que, para cada característica (número e figura), u ou as três cartas são iguais, ou as três cartas são diferentes. Por exemplo, na figura abaixo, (a) é um set válido, já que todas as cartas têm o mesmo tipo de figura e todas elas têm números diferentes de figuras. Em (b), tanto as figuras quanto os números são diferentes para cada carta. Por outro lado, (c) nào é um set, já que as duas ultimas cartas têm a mesma figura, mas esta é diferente da figura da primeira carta.



O objetivo do jogo é formar o maior número de sets com as cartas que estão na mesa; cada vez que um set é formado, as três cartas correspondentes são removidas de jogo. Quando há poucas cartas na mesa, é fácil determinar o maior número de sets que podem ser formados; no entanto, quando há muitas cartas há muitas combinações possíveis. Seu colega quer treinar para o campeonato mundial de Set, e por isso pediu que você fizesse um programa inteira e que calcula o maior número de sets que podem ser formados com um determinado conjunto de cartas.
(Fonte: Maratona de programação regional 2008, RS)

Exercício 10.6 (Matrizes totalmente unimodulares)
Para cada um dos problemas do exercício 10.3 decide, se a matriz de coeficientes é totalmento unimodular.

Exercício 10.7 (Formulação)
Para os problemas abaixo, acha uma formulação como programa inteira.

Cobertura por arcos

Instância Um grafo não-direcionado $G = (V, E)$ com pesos $c : E \to \mathbb{Q}$ nos arcos.

Solução Uma cobertura por arcos, i.e. um subconjunto $E' \subseteq E$ dos arcos tal que todo vértice faz parte de ao menos um arco selecionado.

Objetivo Minimiza o custo total dos arcos selecionados em E'.

Conjunto dominante de arcos

Instância Um grafo não-direcionado $G = (V, E)$ com pesos $c : E \to \mathbb{Q}$ nos arcos.

Solução Um conjunto dominante de arcos, i.e. um subconjunto $E' \subseteq E$ dos arcos tal que todo arco compartilha um vértico com ao menos um arco em E'.

Objetivo Minimiza o custo total dos arcos selecionados em E'.

Coloração de grafos

Instância Um grafo não-direcionado $G = (V, E)$.

Solução Uma coloração do grafo, i.e. uma atribuição de cores nas vértices $c : V \to \mathbb{Z}]$ tal que cada par de vértices ligando por um arco recebe uma cor diferente.

Objetivo Minimiza o número de cores diferentes.

Clique mínimo ponderado

Instância Um grafo não-direcionado $G = (V, E)$ com pesos $c : V \to \mathbb{Q}$ nos vértices.

Solução Uma *clique*, i.e. um subconjunto $V' \subseteq V$ de vértices tal que existe um arco entre todo par de vértices em V'.

Objetivo Minimiza o peso total dos vértices selecionados V'.

Subgrafo cúbico

Instância Um grafo não-direcionado $G = (V, E)$.

Solução Uma subgrafo cúbico, i.e. uma seleção $E' \subseteq E$ dos arcos, tal que cada vértice em $G' = (V, E')$ possui grau 0 ou 3.

Objetivo Minimiza o número de arcos selecionados $|E'|$.

Exercício 10.8 (Formulação e implementação: Investimento)
Uma empresa tem que decidir quais de sete investimentos devem ser feitos. Cada investimento pode ser feito somente uma única vez. Os investimentos tem lucros (ao longo prazo) e custos iniciais diferentes como segue

	Investimento						
	1	2	3	4	5	6	7
Lucro estimado [MR$]	17	10	15	19	7	13	9
Custos iniciais [MR$]	43	28	34	48	17	32	23

A empresa tem 100 MR$ capital disponível. Como maximizar o lucro total (ao longo prazo, não considerando os investimentos atuais), respeitando que os investimentos $1, 2$ e $3, 4$ são mutualmente exclusivas, e nem o investimento 3 nem o investimento 4 pode ser feita, sem ao menos um investimento em 1 ou 2 (as outros investimentos não tem restrições).

Exercício 10.9 (Formulação e implementação: Brinquedos)
Um produtor de brinquedos projetou dois novos brinquedos para Natal. A preparação de uma fábrica para produzir custaria 50000 R$ para a primeiro brinquedo e 80000 R$ para o segundo. Após esse investimento inicial, o primeiro brinquedo rende 10 R$ por unidade e o segundo 15 R$.
O produtor tem duas fábricas disponíveis mas pretende usar somente uma, para evitar custos de preparação duplos. Se a decisão for tomada de produzir os dois brinquedos, a mesma fábrica seria usada.
Por hora, a fábrica 1 é capaz de produzir 50 unidades do brinquedo 1 e 40 unidades do brinquedo 2 e tem 500 horas de produção disponível antes de Natal. A fábrica 2 é capaz de produzir 40 unidades do brinquedo 1 e 25 unidades do brinquedo 2 por hora, e tem 700 horas de produção disponível antes de Natal.
Como não sabemos se os brinquedos serão continuados depois Natal, a problema é determinar quantas unidades de cada brinquedo deve ser produzido até Natal (incluindo o caso que um brinquedo não é produzido) de forma que maximiza o lucro total.

Exercício 10.10 (Formulação e implementação: aviões)
Uma empresa produz pequenos aviões para gerentes. Os gerentes frequentemente precisam um avião com características específicas que gera custos inicias altos no começo da produção.
A empresa recebeu encomendas para três aviões, mas como ela está com capacidade de produção limitada, ela tem que decidir quais das três aviões ela vai produzir. Os seguintes dados são relevantes

Aviões produzidas	Cliente 1	2	3
Custo inicial [MR$]	3	2	0
Lucro [MR$/avião]	2	3	0.8
Capacidade usada [%/avião]	20%	40%	20%
Demanda máxima [aviões]	3	2	5

Os clientes aceitam qualquer número de aviões até a demanda máxima. A empresa tem quer decidir quais e quantas aviões ela vai produzir. As aviões serão produzidos em paralelo.

Parte III

Heurísticas

11 Introdução

Resolução de Problemas

- Problemas Polinomiais
  1. Programação Dinâmica
  2. Divisão e Conquista
  3. Algoritmos Gulosos
- Problemas Combinatórios
  - Técnicas Exatas: Programação Dinâmica, Divisão e Conquista backtracking, branch & bound
  - Programação não-linear: Programação semi-definida, etc.
  - Algoritmos de aproximação: garantem solução aproximada
  - Heurísticas e metaheurísticas: raramente provêem aproximação

Heurísticas

- O que é uma heurística?

  Practice is when it works and nobody knows why.
- Grego *heurísko*: eu acho, eu descubro.
- Qualquer procedimento que resolve um problema
  - bom em média
  - bom na prática (p.ex. Simplex)
  - não necessáriamente comprovadamente.
- Nosso foco
  - Heurísticas construtivas: Criam soluções.
  - Heurísticas de busca: Procumra soluções.

Heurísticas de Construção

- Constróem uma solução, escolhendo um elemento a ser inserido na solução a cada passo.
- Geralmente são algoritmos gulosos.
- Podem gerar soluções infactíveis.
  - Solução infactível: não satisfaz todas as restrições do problema.
  - Solução factível: satisfaz todas as restrições do problema, mas não é necessariamente a ótima.

Exemplo: Heurística construtiva

- Problema do Caixeiro Viajante (PCV) – Heurística do vizinho mais próximo.

HVizMaisProx

Entrada Matriz de distâncias completa $D = (d_{ij})$, número de cidades n.

Saída Uma solução factível do PCV: Ciclo Hamiltaneo C com custo c.

```
1  HVizMaisProx(D,n)=
2    { cidade inicial randômica }
3    u := seleciona uniformemente de [1,n]
4    w := u
5    { representação de camnihos: sequência de vértices }
6    C := u        { ciclo inicial }
7    c := 0        { custo do ciclo }
8    repeat n-1 vezes
9      seleciona v ∉ C com distância mínima de u
10     C := C v
11     c := c + d_uv
12     u := v
13   end repeat
14   C := C w  { fechar ciclo }
15   c := c + d_uw
16   return (C,c)
```

Meta-heurísticas

- Heurísticas genéricas: *meta-heurísticas*.

Motivação: quando considera-se a possibilidade de usar heurísticas

- Para gerar i,a solução factível num tempo pequeno, muito menor que uma solução exata pudesse ser fornecida.
- Para aumentar o desempenho de métodos exatos. Exemplo: um limitante superior de um Branch-and-Bound pode ser fornecido por uma heurística.

Desvantagens do uso de heurísticas

- No caso de metaheurísticas, não há como saber o quão distante do ótimo a solução está
- Não há garantia de convergência
- Dependendo do problema e instância, não há nem como garantir uma solução ótima

Problema de otimização em geral

- Um problema de otimização pode ser representado por uma quádrupla

$$(I, S, f, obj)$$

  - I é o conjunto de possíveis instâncias.
  - $S(i)$ é o conjunto de soluções factíveis (espaço de soluções factíveis) para a instância i.
  - Uma função objetivo (ou *fitness*) $f(\cdot)$ avalia a qualidade de uma dada solução.
  - Um objetivo obj = min ou max: $s^* \in S$ para o qual $f(s^*)$ seja mínimo ou máximo.

- Alternativa

$$\begin{aligned} \textbf{optimiza} \quad & f(x) \\ \textbf{sujeito a} \quad & x \in S \end{aligned}$$

- S discreto: problema combinatorial.

Técnicas de solução

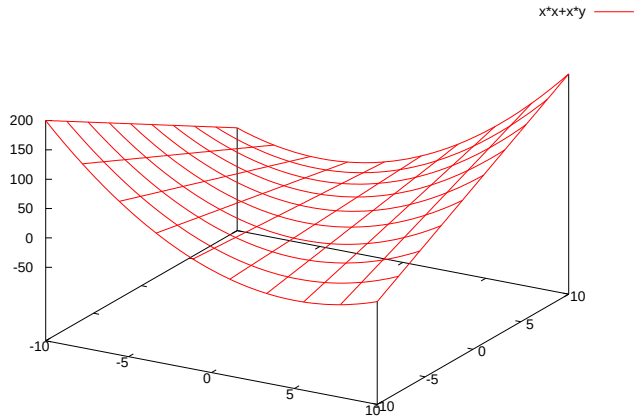
- Resolver o problema nessa geralidade: enumeração.
- Frequentemente: Uma solução $x \in S$ possiu uma *estrutura*.
- Exemplo: x é um tuplo, um grafo, etc.
- Permite uma enumeração por componente: branch-and-bound.

12 Heurísticas baseados em Busca local

12.1 Busca local

Busca Local

- Frequentemente: O espaço de soluções possui uma *topologia*.
- Exemplo da otimização (contínua): $\max\{x^2 + xy \mid x, y \in \mathbb{R}\}$



- Espaço euclidiano de duas dimensões.
- Isso podemos aproveitar: Busca localmente!

Vizinhanças

- O que fazer se não existe uma topologia natural?
- Exemplo: No caso do TSP, qual o vizinho de um ciclo Hamiltaneo?
- Temos que definir uma vizinhança.

- Notação: Para $x \in S$

$$\mathcal{N}(x)$$

  denota o conjunto de soluções vizinhos.

- Uma vizinhança defina a *paisagem de otimização* (ingl. optimization landscape): Espaço de soluções com valor de cada solução.

Relação de vizinhança entre soluções

- Uma solução s' é obtida por uma pequena modificação na solução s.
- Enquanto que $\mathcal{S}$ e f são fornecidos pela especificação do problema, o projeto da vizinhança é livre.

Busca Local k-change e inserção

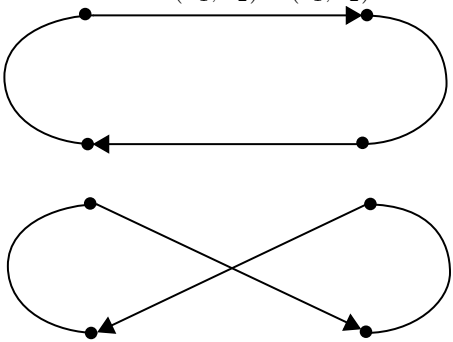
- k-change: mudança de k componentes da solução.
- Cada solução possui vizinhança de tamanho $O(n^k)$.
- Exemplo: 2-change, 3-change.
- TSP: 2-change (inversão).
- Inserção/remoção: inserção de um componente da solução, seguido da factibilização da solução
- Vertex cover: 1-change + remoção.

Exemplo: Vizinhança mais elementar

- Suponha um problema que possue como soluções factíveis $S = \mathbb{B}^n$ (por exemplo, uma instância do problema de particionamento de conjuntos).
- Então, para $n = 3$ e s_0=$\{0,1,0\}$, para uma busca local 1-flip, $N(s_0) = \{(1,1,0),(0,0,0),(0,1,1)\}$.

Exemplo: Vizinhanças para TSP

- 2-opt: Para cada par de arcos (u_1, v_1) e (u_2, v_2) não consecutivos, remova-os da rota, e insira os arcos (u_1, u_2) e (v_1, v_2).



- Para uma solução s e uma busca k-opt $|\mathcal{N}(s)| \in \mathrm{O}(n^k)$.

Características de vizinhanças

É desejável que uma vizinhança é

- *simétrica* (ou *reversível*)

$$y \in \mathcal{N}(x) \Rightarrow x \in \mathcal{N}(y)$$

- *conectada* (ou *completa*)

$$\begin{aligned} \forall x, y \in S \, \exists z_1, \ldots, z_k \in S \quad & z_1 \in \mathcal{N}(x) \\ & z_{i+1} \in \mathcal{N}(z_i) \qquad 1 \leq i < k \\ & y \in \mathcal{N}(z_k) \end{aligned}$$

Busca Local: Ideía

- Inicia a partir de uma solução s_0
- Se move para soluções vizinhas melhores no espaço de busca.
- Para, se não tem soluções melhores na vizinhança.
- Mas: Repetindo uma busca local com soluções inicias randômicas, achamos o mínimo global com probabilidade 1.

Busca local – Caso contínuo

Busca local contínua

Entrada Solução inicial $s_0 \in \mathbb{R}^n$, tamanho incial α de um passo.

Saída Solução $s \in \mathbb{R}^n$ tal que $f(s) \leq f(s_0)$.

Nome Gradient descent.

```
1  BuscaLocal(s_0 ,α)=
2    s := s_0
3    while ∇f(x) ≠ 0 do
4      s' := s − α∇f(s)
5      if f(s') < f(s) then
6        s := s'
7      else
8        diminui α
9      end if
10   end while
11   return s
```

Busca local – Caso contínuo

- Gradiente
$$\nabla f(x) = \left(\frac{\delta f}{\delta x_1}(x), \ldots, \frac{\delta f}{\delta x_n}(x) \right)^t$$
sempre aponta na direção do crescimento mais alto de f (Cauchy).
- Necessário: A função objetivo f é diferenciável.
- Diversas técnicas para diminuir (aumentar) α.
- Opção: Line search na direção $-\nabla f(x)$ para diminuir o número de gradientes a computar.

Busca Local – Best Improvement

Busca Local BI

Entrada Solução inicial s_0.

Saída Solução s tal que $f(s) \leq f(s_0)$.

Nomes Steepest descent, steepest ascent.

```
1  BuscaLocal(s_0)=
2     s := s_0
3     while true
4        s' := argmin_y{f(y) | y ∈ N(s)}
5        if f(s') < f(s) then s := s'
6        else break
7     end while
8     return s
```

Busca Local – First Improvement

Busca Local FI (s)

Entrada Solução inicial s_0.

Saída Solução s' tal que $f(s') \leq f(s)$.

Nomes Hill descent, hill climbing.

```
1  BuscaLocal(s_0)=
2     s := s_0
3     repeat
4        Select any s' ∈ N(s) not yet considered
5        if f(s') < f(s) then s := s'
6     until all solutions in N(s) have been visited
7     return s
```

Projeto de uma busca local

- Como gerar uma solução inicial? Aleatória, via método construtivo, etc.
- Quantas soluções inicias devem ser geradas?
- Importante: Definição da função de vizinhança $\mathcal{N}$.
- Vizinhança grande ou pequena? (grande= muito tempo e pequena=menos vizinhos)
- Estratégia de seleção de novas soluções
  - examine todas as soluções vizinhas e escolha a melhor
  - assim que uma solução melhor for encontrada, reinicie a busca. Neste caso, qual a sequência de soluções examinar?
- Importante: Método eficiente para avaliar a função objetivo de vizinhos.

Exemplo: 2-change TSP

- Vizinhança: Tamanho $O(n^2)$.
- Avaliação de uma solução: $O(n)$ (somar n distâncias).
- Atualizando a valor da solução atual: $O(1)$ (somar 4 distâncias)
- Portanto: Custo por iteração de "best improvement"
  - $O(n^3)$ sem avaliação diferential.
  - $O(n^2)$ com avaliação diferential.

Avaliação de buscas locais

Como avaliar a busca local proposta?

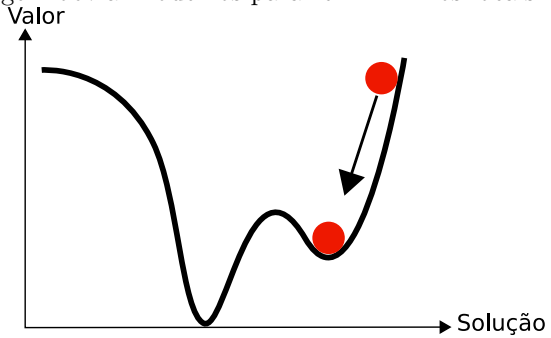
- Poucos resultados teóricos.
- Difícil de saber a qualidade da solução resultante.
- Depende de experimentos.

Problema Difícil

- É fácil de gerar uma solução aleatória para o TSP, bem como testar sua factibilidade
- Isso não é verdade para todos os problemas
- Exemplo difícil: Atribuição de pesos a uma rede OSPF

Busca local

- Desvantagem obvia: Podemos parar em mínimos locais.



- Exceto: Função objetivo convexa (caso minimização) ou concava (caso maximização).
- Técnicas para superar isso baseadas em busca local
  - Multi-Start
  - Busca Tabu
  - Algoritmos Metropolis e Simlated Annealing
  - Variable neighborhood search

Multi-Start Metaheuristic

- Gera uma solução aleatória inicial e aplique busca local nesta solução.
- Repita este procedimento por n vezes.
- Retorne a melhor solução encontrada.
- Problema: soluções aleatoriamente geradas em geral possuem baixa qualidade.

Multi-Start

MULTI-START

Entrada Número de repetições n.

Saída Solução s.

```
1  Multi_Start(n) :=
2    s* := ∅
3    f* := ∞
4    repeat n vezes
5      gera solução randômica s
6      s := BuscaLocal(s)
7      if f(s) < f* then
8        s* := s
9        f* := f(s)
10     end if
11   end repeat
12   return s*
```

Cobrimento de Vértices

- Definição de vizinhança
- grafo sem vértices
- grafo estrela
- clique bipartido $K_{i,j}$
- grafo linha

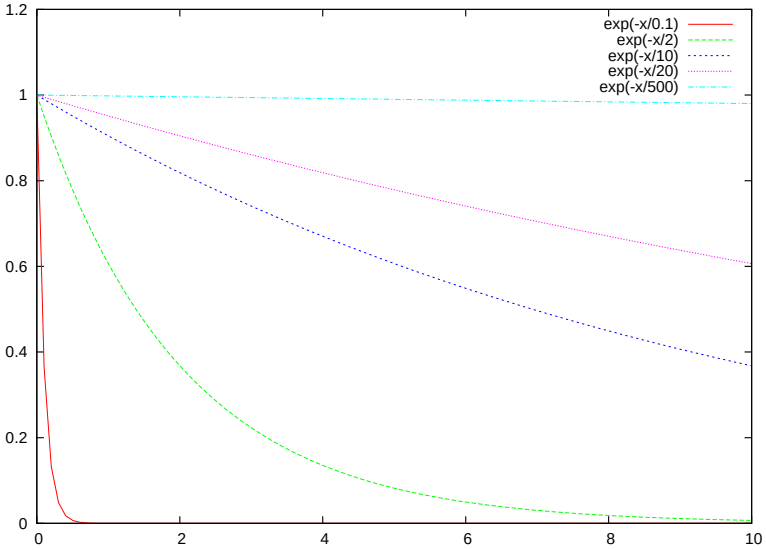
12.2 Metropolis e Simulated Annealing

O algoritmo Metropolis

- Proposto em 1953 por Metropolis, Rosenbluth, Rosenbluth, Teller e Teller
- simula o comportamento de um sistema físico de acordo com a mecânica estatística
- supõe temperatura constante
  - Um modelo básico define que a probabilidade de obter um sistema num estado com energia E é proporcional à função $e^{-\frac{E}{kT}}$ de Gibbs-Boltzmann, onde $T > 0$ é a temperatura, e $k > 0$ uma constante

- a função é monotônica decrescente em E: maior probabilidade de estar em um sistema de baixa energia
- para T pequeno, a probabilidade de um sistema em estado de baixa energia é maior que um em estado de alta energia
- para T grande, a probabilidade de passar para outra configuração qualquer do sistema é grande

A distribuição de Boltzmann



Algoritmo Metropolis

- Estados do sistema são soluções candidatas
- A energia do sistema é representada pelo custo da solução
- Gere uma perturbação na solução s gerando uma solução s'.
- Se $E(s') \leq E(s)$ atualize a nova solução para s'.
- Caso contrário, $\triangle E = E(s') - E(s) > 0$.
- A solução s' passa ser a solução atual com probabilidade $e^{-\frac{\triangle E}{kT}}$
- Característica marcante: permite movimentos de melhora e, com baixa probabilidade, também de piora

Metropolis

METROPOLIS

Entrada Solução inicial s, uma temperatura T, uma constante k.

Saída Solução $s' : c(s') \leq c(s)$

```
1  Metropolis(s, T, k)=
2    while STOP1 times do
3      Select any unvisited s' ∈ N(s)
4      if c(s') ≤ c(s) then update s := s'
5      else
6        with probability e^{-(c(s')-c(s))/kT} update s := s'
7      end while
8    return s
```

Considerações sobre o algoritmo

- O algoritmo Metropolis pode resolver problemas que o gradiente descendent não conseguia
- Mas em muitos casos o comportamento deste algoritmo não é desejado (vertex cover para grafo sem arcos)
- Alta probabilidade de saltos quando próximo de um mínimo local
- T pode ser manipulada: se T for alta, o algoritmo Metropolis funciona de forma similar a um *random walk* e se T for baixa (próxima a 0), o algoritmo Metropolis funciona de forma similar ao gradiente descendente.

Simulated Annealing

- Simula um processo de annealing.
- Annealing: processo da física que aquece um material a uma temperatura bem alta e resfria aos poucos, dando tempo para o material alcançar seu estado de equilíbrio
- Simulated annealing: parte de uma alta temperatura e baixa gradualmente. Para cada temperatura, permite um número máximo de saltos (dois loops encadeados)

Simulated Annealing

SIMULATED ANNEALING

Entrada Solução inicial s, temperatura T, constante k, fator de esfriamento $r \in [0,1]$, dois números inteiros STOP1, STOP2.

Saída Solução s' tal que $f(s') \leq f(s)$.

```
1  SimulatedAnnealing(s, T, k, r, STOP1, STOP2) :=
2    repeat STOP2 vezes
3      repeat STOP1 vezes
4        seleciona s' ∈ N(s) que ainda não foi visitado
5          if f(s') ≤ f(s) then
6            s := s'
7          else
8            Com probabilidade e^{-(f(s')-f(s))/kT}: s := s'
9          end fi
10     end repeat
11     T := T × r
12   end repeat
13 return s
```

12.3 GRASP

GRASP

- GRASP: greedy randomized adaptive search procedure
- Proposto por Mauricio Resende e Thomas Feo (1989).
- Mauricio Resende: Pesquisador da AT&T por 20 anos, Departamento de Algoritmos e Otimização

Mauricio G. C. Resende

GRASP

- Método multi-start, em cada iteração
  1. Gera soluções com um procedimento guloso-randomizado.
  2. Otimiza as soluções geradas com busca local.

GRASP

Entrada Solução inicial s, parametro α.

Saída Solução $s' : c(s') \leq c(s)$

```
1  GRASP(s0 , α , ...) =
2     s := s0
3     do
4        s' := greedy_randomized_solution(α)
5        s' := BuscalLocal(s')
6        s := s' if f(s') < f(s)
7     until a stopping criterion is satisfied
8     return s
```

Construção gulosa-randomizada

- Motivação: Um algoritmo guloso gera boas soluções inicias.
- Problema: Um algoritmo determinístico produz sempre a mesma solução.
- Logo: Aplica um algoritmo guloso, que não escolhe *o melhor* elemento, mas escolhe randomicamente entre os $\alpha\%$ *melhores* candidatos.
- O conjunto desses candidatos se chama *restricted candidate list* (RCL).

Construção gulosa-randomizada: Algoritmo guloso

```
1  Guloso () :=
2    S := ()
3
4    while S = (s1, ..., si) com i < n do
5      entre todos candidatos C para si+1:
6        escolhe o melhor s ∈ C
7      S := (s1, ..., si, s)
8    end while
```

Construção gulosa-randomizada: Algoritmo guloso

```
1  Guloso−Randomizado(α) :=
2    S := ()
3
4    while S = (s_1, ..., s_i) com i < n do
5      entre todos candidatos C para s_{i+1}:
6        forma a RCL com os α\% melhores candidatos em C
7        escolhe randomicamente um s ∈ RCL
8      S := (s_1, ..., s_i, s)
9    end while
```

GRASP

GRASP

Entrada Solução inicial s, parametro α.

Saída Solução $s' : c(s') \leq c(s)$

```
1  GRASP(s_0 , α , ...) =
2     x := s_0
3     do
4        y := greedy_randomized_solution(α)
5        y := BuscalLocal(y)
6        atualiza x caso y é solução melhor
7     until a stopping criterion is satisfied
8     return s
```

GRASP: Variações

- *long term memory*: hash table (para evitar otimizar soluções já vistas)
- Parâmetros: s_0, $\mathcal{N}(x)$, $\alpha \in [0, 1]$ (para randomização), tamanho das listas (conj. elite, rcl, hash table), número de iterações,

GRASP com memória

- O GRASP original não havia mecanismo de memória de iterações passadas

- Atualmente toda implementação de GRASP usa conjunto de soluções elite e religação por caminhos (*path relinking*)
- Conjunto de soluções elite: conjunto de soluções diversas e de boa qualidade
  - uma solução somente é inserida se for melhor que a melhor do conjunto ou se for melhor que a pior do conjunto e diversa das demais
  - a solução a ser removida é a de pior qualidade
- Religação por Caminhos: a partir de uma solução inicial, modifique um elemento por vez até que se obtenha uma solução alvo (do conjunto elite)
- soluções intermediárias podem ser usadas como soluções de partida

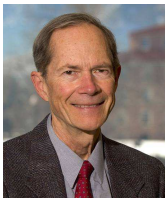
Comparação entre as metaheurísticas apresentadas

- Metaheurísticas: Simulated annealing (SA), Multi-Start Search (MS), GRASP
- SA tem apenas um ponto de partida, enquanto que os outros dois métodos testa diversos
- SA permite movimento de piora, enquanto que os outros dois métodos não
- SA é baseado em um processo da natureza, enquanto que os outros dois não

12.4 Busca Tabu

Busca Tabu (Tabu Search)

- Proposto por Fred Glover em 1986 (princípios básicos do método foram propostos por Glover ainda em 1977)
- Professor da Universidade do Colorado, EUA



Fred Glover

Busca Tabu (BT)

- Assim como em simulated anneling (SA) e VNS, TB é baseada inteiramente no processo de busca local, movendo-se sempre de uma solução s para uma solução s'
- Assim com em SA, também permite movimentos de piora
- Diferente de SA que permite movimento de piora por randomização, tal movimento na BT é determinístico
- A base do funcionamento de Busca Tabu é o uso de memória segundo algumas regras
- O nome *Tabu* tem origem na proibição de alguns movimentos durante a busca

Busca Tabu (BT)

- Mantém uma lista T de movimentos tabu
- A cada iteração se move para o melhor vizinho, desde que não faça movimentos tabus
- Permite piora da solução: o melhor vizinho pode ser pior que o vizinho atual!
- São inseridos na lista tabu elementos que provavelmente não direcionam a busca para o ótimo local desejado. Ex: último movimento executado
- o tamanho da lista tabu é um importante parâmetro do algoritmo
- Critérios de parada: quando todos movimentos são tabus ou se x movimentos foram feitos sem melhora

Busca Tabu: Conceitos Básicos e notação

- s: solução atual
- s^*: melhor solução
- f^*: valor de s*
- $\mathcal{N}(s)$: Vizinhança de s.
- $\tilde{\mathcal{N}}(s) \subset \mathcal{N}(s)$: possíveis (não tabu) soluções vizinhas a serem visitadas

- Soluções: inicial, atual e melhor
- Movimentos: atributos, valor
- Vizinhança: original, modificada (reduzida ou expandida)

Movimentos Tabu

- Um movimento é classificado como *tabu* ou *não tabu* pelas *regras de ativação tabu*
- em geral, as regras de ativação tabu classificam um movimento como tabu se o movimento foi recentemente realizado
- Memória de curta duração (MCD) - também chamada de *lista tabu*: usada para armazenar os movimentos tabu
- duração tabu (*tabu tenure*) é o número de iterações em que o movimento permanecerá tabu
- dependendo do tamanho da MCD um movimento pode deixar de ser tabu antes da duração tabu estabelecida
- A MCD em geral é implementada como uma lista circular
- O objetivo principal da MCD é evitar ciclagem e retorno a soluções já visitadas
- os movimentos tabu também colaboram para a busca se mover para outra parte do espaço de soluções, em direção a um outro mínimo local

Busca Tabu

BuscaTabu

Entrada uma solução s

Saída uma solução $s' : f(s') \leq f(s)$

```
1  BuscaTabu()=
2     Inicialização:
3        s := S_0 ;  f* := f(s_0) ;  s* := s_0  ;  T := ∅
4     while not STOP
```

```
5    s' := select s' ∈ Ñ(s) com min f(s)
6    if f(s) < f* then
7       f* := f(s); s* := s
8    insira movimento em T (a lista tabu)
9  end while
```

Busca Tabu (BT)

- critérios de parada:
  - número de iterações (N_{max})
  - número interações sem melhora
  - quando $s*$ atinge um certo valor mínimo (máximo) estabelecido
- Um movimento não é executado se for tabu, ou seja, se possuir um ou mais atributos tabu-ativos
- Pode ser estabelecida uma regra de uso de um movimento tabu (critério de aspiração)
  - Critério de aspiração por objetivo: se o movimento gerar uma solução melhor que $s*$, permite uso do movimento tabu
  - Critério de aspiração por direção: o movimento tabu é liberado se for na direção da busca (de melhora ou piora)

Busca Tabu: mecanismos auxiliares

- intensificação: a idéia é gastar mais "esforço" em regiões do espaço de busca que parece mais promissores. Isso pode ser feito de diversas maneiras (exemplo, guardar o número de interações com melhora consecutiva). Nem sempre este a intensificação traz benefícios.
- Diversificação: recursos algorítmicos que forçam a busca para um espaço de soluções ainda não explorados.
  - uso de memória de longo prazo (exemplo, número de vezes que a inserção de um elemento provocou melhora da solução)
  - Estratégia básica: forçar a inserção de alguns poucos movimentos pouco executados e reiniciar a busca daquele ponto
  - Estratégia usada para alguns problemas: permiter soluções infactíveis durante algumas interações

Busca Tabu: variações

- Várias listas tabus podem ser utilizadas (com tamanhos, duração, e regras diferentes)
- BT probabilístico: os movimentos são avaliados para um conjunto selecionado aleatoriamente $N'(s) \in \tilde{N}(s)$. Permite usar uma lista tabu menor, acontece menos ciclagem.
- A duração tabu pode variar durante a execução

Comparação entre as metaheurísticas apresentadas até então

- Metaheurísticas: Simulated annealing (SA), Multi-Start Search (MSS), GRASP, BT
- SA e BT têm apenas um ponto de partida, enquanto que os outros dois métodos testa diversos
- SA e BT permitem movimentos de piora, enquanto que os outros dois métodos não
- SA é baseado em um processo da natureza, enquanto que os outros métodos não

Parâmetros e decisões das metaheurísticas

- SA:
  - Parâmetros: temperatura inicial, critério de parada, variável de resfriamento
  - Decisões: vizinhança, solução inicial
- GRASP:
  - Parâmetros: s_0, $\mathbb{N}(x)$, $\alpha \in [0,1]$ (para randomização), tamanho das listas (conj. elite, rcl, hash table), critério de parada
  - Decisões: vizinhança, solução inicial (s_0), randomização da s_0, atualizações do conjunto elite
- BT:
  - Parâmetros: tamanho da lista tabu, critério de parada
  - Decisões: vizinhaça, critérios para classificar movimento tabu

12.5 Variable Neighborhood Search

Variable Neighborhood Search

- Pierre Hansen e Mladenović, 1997
- Hansen é Professor na HEC Montréal, Canadá

Pierre Hansen

Variable Neighborhood Search

- Método multi-start que explora mais de uma vizinhaça.
- Explora sistematicamente as seguintes propriedades:
  - O mínimo local de uma vizinhança não é necessariamente mínimo para outra vizinhança
  - Um mínimo global é um mínimo local com respeito a todas as vizinhanças
  - Para muitos problemas, os mínimos locais estão localizados relativamente próximos no espaço de busca para todas as vizinhanças

Variable Neighborhood Search

VNS

Entrada Solução inicial s_0, um conjunto de vizinhanças $\mathcal{N}_i$, $1 \leq i \leq m$.

Saída uma solução $s : f(s) \leq f(s_0)$

```
1  VNS(s_0, {N_i})=
2     x := s_0
3     do (até chegar a um mínimo local
4        para todas as buscas locais)
5        k := 1
6        while k < m do
7           escolhe y ∈ N_k(x) randomicamente
```

```
 8          y := BuscaLocal(y)
 9          if f(y) < f(x) then
10            x := y
11            k := 1
12          else
13            k := k + 1
14          end if
15        end while
16      end do
17      return x
```

13 Heurísticas inspirados da natureza

13.1 Algoritmos Genéticos e meméticos

Algoritmos Genéticos

- Proposto na década de 60 por Henry Holland.
- Professor da Faculdade de Engenharia Elétrica e de Computação da Universidade de Michigan/EUA.
- Seu livro: Adaptation in Natural and Artificial Systems (1975).

John Henry Holland (+1929)

Algoritmos genéticos

- Foi proposto com o objetivo de projetar software de sistemas artificiais que reproduzem processos naturais.
- Baseados na evolução natural das espécies.
- Por Darwin: indivíduos mais aptos têm mais chances de perpetuar a espécie.
- Mantém uma população de soluções e não uma única solução por vez.
- Usa regras de transição probabilísticas, e não determinísticas.
- Procedimentos: avaliação, seleção, geração de novos indivíduos (recombinação), mutação.
- Parada: número x de gerações total, número y de gerações sem melhora.

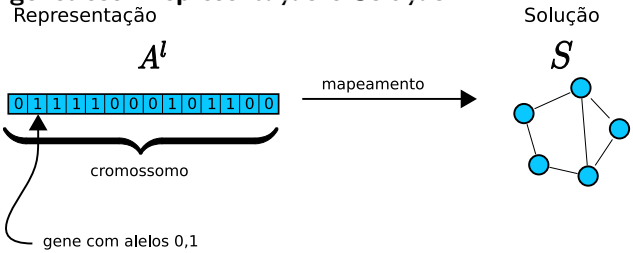
Algoritmos genéticos: Características

- Varias soluções ("população").
- Operações novas: Recombinação e mutação.
- Separação da representação ("genótipo") e formulação "natural" (fenótipo).

Algoritmos Genéticos: Noções

- Genes: Representação de um elemento (binário, inteiro, real, arco, etc) que determine uma característica da solução.
- Alelo: Instância de uma gene.
- Cromossomo: Uma string de genes que compõem uma solução.
- Genótipo: Representação genética da solução (cromossomos).
- Fenótipo: Representação "física" da solução.
- População: Conjunto de cromossomos.

Algorítmos genéticos: Representação e Solução



Algoritmos Genéticos: exemplos

- Problema de partição de conjuntos

  Gens: 0 ou 1

  Cromossomo: 000110101010111110110
- Problema do Caixeiro viajante

  Gens: valores inteiros entre 1 e n

  Cromossomo: 1 5 3 6 8 2 4 7

Procedimentos dos Algoritmos Genéticos

- Codificação: genes e cromossomos.
- Initialização: geração da população inicial.
- Função de Avaliação (fitness): função que avalia a qualidade de uma solução.

- Seleção de pais: seleção dos indivíduos para crossover.
- Operadores genéticos: crossover, mutação
- Parâmetros: tamanho da população, percentagem de mutação, critério de parada

Algoritmos Genéticos

AlgoritmoGenético

Entrada Parâmetros do algoritmo.

Saída Melhor solução encontrada para o problema.

```
1  Inicialização e avalição inicial
2  while (critério de parada não satisfeito) do
3    repeat
4      if (critério para recombinação) then
5        selecione pais
6        recombina e gera um filho
7      end if
8      if (critério para mutação) then
9        aplica mutação
10     end if
11   until (descendentes suficientes)
12   selecione nova população
13 end while
```

População Inicial: geração

- Soluções aleatórias.
- Método construtivo (ex: vizinho mais próximo com diferentes cidades de partida).
- Heurística construtiva com perturbações da solução.
- Pode ser uma mistura das opções acima.

População inicial: tamanho

- População maior: Custo alto por iteração.
- Populaçao menor: Cobertura baixa do espaço de busca.
- Critério de Reeves: Para alfabeto binário, população randômica: Cada ponto do espaço de busca deve ser alcancável através de recombinações.
- Consequencia: Probabilidade que cada alelo é presente no gene i: $1 - 2^{1-n}$.
- Probabilidade que alelo é presente em todos gene: $(1 - 2^{1-n})^l$.
- Exemplo: Com $l = 50$, para garantir cobertura com probabilidade 0.999:

$$n \geq 1 - \log_2 \left(1 - \sqrt[50]{0.999}\right) \approx 16.61$$

Terminação

- Tempo.
- Número de avaliações.
- Diversidade. Exemplo: Cada gene é dominado por um alelo, i.e. 90% dos indivíduos tem o mesmo alelo.

Próxima Geração

- Gerada por recombinação e mutação (soluções aleatórias ou da população anterior podem fazer parte da próxima geração).
- Estratégias:
  - Recombinação *e* mutação.
  - Recombinação *ou* mutação.
- Regras podem ser randomizadas.
- Exemplo: Taxa de recombinação e taxa de mutação.
- Exemplo: Número de genes mutados.

Mutação

- Objetivo: Introduzir elementos diversificados na população e com isso possibilitar a exploração de uma outra parte do espaçõ de busca.
- Exemplo para representação binária: flip de k bits.
- Exemplo para o PCV: troca de posição entre duas cidades.

Recombinação

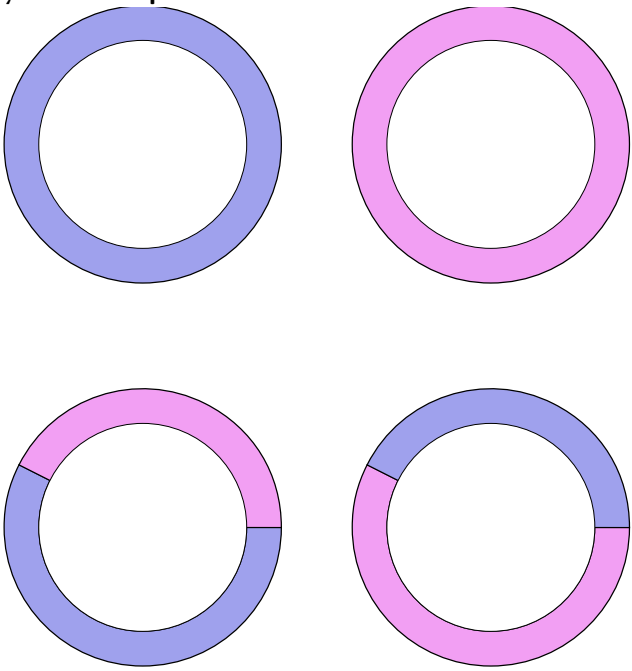
- Recombinação (ingl. crossover): combinar características de duas soluções para prover uma nova solução potencialmente com melhor fitness.
- Explora o espaço entre soluções.
- Crossover clássicos: one-point recombinação e two-points recombinação.

One-point crossover

Escolha um número aleatório k entre 1 e n. Gere um filho com os primeiros k bits do pai A e com os últimos $n - k$ bits do pai B

- Problema de particação: aplicação direta do conceito
- Problema do Caixeiro Viajante: copie os primeiros k elementos do pai A e as demais $n-k$ posições preenche com as cidades faltantes, segundo a ordem em que elas aparecem no pai B

Recombinação de dois pontos



Exemplo: Strategic Arc Crossover

- Selecione todos os pedaçõs de rotas (string) com 2 ou mais cidades que são iguais nas duas soluções
- Forme uma rota através do algoritmo de vizinho mais próximo entre os pontos extremos dos strings

Recombinação: Seleção dos pais

- A probabilidade de uma solução ser pai num processo de crossover deve depender do seu fitness.
- Variações:
  - Probabilidade proporcional com fitness.
  - Probabilidade proporcional com ordem.

Estratégia adotada pelos operadores
Inúmeros operadores podem ser propostos para cada problema. O ideal é combinar características do operador usado, com outros operadores (mutação, busca local) usados no GA. Basicamente um crossover é projetado da seguinte forma:

- Encontre similaridades entre A e B e insira $S = A \cap B$ no filho.
- Defina conjuntos S_{in} e S_{out} de características desejáveis e não desejáveis.
- Projete um operador que mantenha ao máximo elementos de S e S_{in}, minimizando o uso de elementos de S_{out}.

Nova População

- Todos os elementos podem ser novos.
- Alguns elementos podem ser herdados da população anterior.
- Elementos novos podem ser gerados.
- Exemplos, com população de tamanho λ que gera μ filhos. (λ, μ) Seleciona os λ melhores dos filhos. $(\lambda + \mu)$ Seleciona os λ melhores em toda população.

Estrutura da População
Em geral, população estruturada garante melhores resultados. A estrutura da população permite selecionar pais para crossover de forma mais criteriosa. Algumas estruturas conhecidas

- Divisão em Castas: 3 partições A, B e C (com tamanhos diferentes), sendo que os melhores indivíduos estão em A e os piores em C.
- Ilhas: a população é particionada em subpopulações que evoluem em separado, mas trocam indivíduos a cada período de número de gerações.
- População organizada como uma árvore.

Exemplo: População em castas

- Recombinação: Somente entre individuos da casta A e B ou C para manter diversidade.
- Nova população: Manter casta "elite" A, re-popular casta B com filhos, substituir casta C com soluções randômicas.

Exemplo: População em árvore

- Considere uma árvore ternária completa, em que cada nó possui duas soluções (pocket e current).
- A solução current é a solução atual armazenada naquela posição da árvore.
- A solução pocket é a melhor já tida naquela posição desde a primeira geração.
- A cada solução aplique *exchange* (se a solução current for melhor que a pocket, troque-as de posição)
- Se a solução pocket de um filho for melhor que a do seu pai, troque o nó de posição.

Algoritmos Meméticos

- Proposto por Pablo Moscato, Newcastle, Austrália.
- Ideía: Informação "cultural" pode ser adicionada a um indivíduo, gerando um algoritmo memético.
- Meme: unidade de informação cultural.

Pablo Moscato

Algoritmos Meméticos

- Um procedimento de busca local pode inserir informação de boa qualidade, e não genética (memes).
- Faz uso de um procedimento de busca local (em geral aplicado à solução gerada pelo procedimento de recombinação).
- Geralmente trabalha com populações menores.

Comparação entre as Metaheurísticas Apresentadas

- Quais que dependem de randomização? SA, GRASP, GA
- Quais que geram apenas uma solução inicial em todo processo? BT, SA

- Quais mantêm um conjunto de soluções, em vez de considerar apenas uma? GA
- Quais são inspiradas em processos da natureza? GA, BT
- Qual gera os melhores resultados?

Existem outras Metaheurísticas

Handbook of Metaheuristics, por Fred W. Glover (Editor), Gary A. Kochenberger (Editor) Kluwer 2002.

Considerações Finais

- O desempenho de uma metaheurística depende muito de cada implementação
- As metaheurísticas podem ser usadas de forma hibridizada
- Técnicas de otimização multiobjetivo tratam os casos de problemas com mais de um objetivo (Curva de pareto)

Exercício

- Problema de alocação: atender n clientes por m postos de atendimento (um posto é instalado no local onde se encontra umum cliente)
- Entrada: distâncias entre cada par de clientes
- Problema: Determinar em que locais instalar os postos, de forma a minimizar a soma das distâncias de cada cliente a um ponto de atendimento

- Propor uma heurística construtiva e uma busca local.

Comparação entre as Metaheurísticas

- Quais que permitem movimento de piora? BT, SA
- Quais que não dependem de randomização? BT
- Quais que geram apenas uma solução inicial em todo processo? BT, SA
- Quais mantêm um conjunto de soluções, em vez de considerar apenas uma?
- Qual gera os melhores resultados?

TBD

- Aplicação da programação linear na aproximação (TK 11.6,11.7, CA 2.4.1,2.4.2).
- ϵ-cortes no branch-and-bound: corta sub-árvores que são pouco mais que o valor atual.

Parte IV

Appéndice

A Conceitos matemáticos

$\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ e $\mathbb{R}$ denotam os conjuntos dos números naturais sem 0, inteiros, racionais e reais, respectivamente. Escrevemos também $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$, e para um dos conjuntos C acima, $C_+ := \{x \in C | x > 0\}$ e $C_- := \{x \in C | x < 0\}$. Por exemplo

$$\mathbb{R}_+ = \{x \in \mathbb{R} | x > 0\}.$$

Para um conjunto finito S, $\mathcal{P}(S)$ denota o conjunto de todos subconjuntos de S.
$A = (a_{ij}) \in F^{m \times n}$ denota uma *matriz* de m linhas e n colunas com elementos em F, a_i, com $a_i^t \in F^n$ a i-ésigma linha e $a^j \in F^m$ a j-ésima coluna de A.

- Vetores linearmente independentes.
- Poesto (linha,coluna) de uma matriz.

Definição A.1
Uma função $f : \mathbb{R} \to \mathbb{R}$ é *linear* se

1. $\forall a \in \mathbb{R}\ f(ax) = af(x)$
2. $f(x + y) = f(x) + f(y)$

Definição A.2 (Pisos e tetos)
Para $x \in \mathbb{R}$ o *piso* $\lfloor x \rfloor$ é o maior número inteiro menor que x e o *teto* $\lceil x \rceil$ é o menor número inteiro maior que x. Formalmente

$$\lfloor x \rfloor = \max\{y \in \mathbb{Z} | y \leq x\}$$
$$\lceil x \rceil = \min\{y \in \mathbb{Z} | y \geq x\}$$

O *parte fracionário* de x é $\{x\} = x - \lfloor x \rfloor$.

Observe que o parte fracionário sempre é positivo, por exemplo $\{-0.3\} = 0.7$.

Proposição A.1 (Regras para pisos e tetos)
Pisos e tetos satisfazem

$$x \leq \lceil x \rceil < x + 1 \tag{A.1}$$
$$x - 1 < \lfloor x \rfloor \leq x \tag{A.2}$$

B Formatos

Essa capítulo contém um breve resumo de dois formatos usados para descrever problemas de otimização linear. CPLEX LP é um formato simples, enquanto AMPL (A modeling language for mathematical programming) é uma linguagem completa para definir problemas de otimização, com elementos de programação, comandos interativos e um interface para diferentes "solvers" de problemas.
CPLEX LP serve bom para experimentos rápidos. Aprender AMPL precisa mais investimento, que rende em aplicações maiores. AMPL tem o apoio da maioria das ferramentas disponíveis.
Vários outros formatos são em uso, a maioria deles comerciais. Exemplos são MPS (Mathematical programming system, um formato antigo e pouco usável do IBM), LINGO, ILOG, GAMS e ZIMPL.

B.1 CPLEX LP

Uma gramática simplificada[1] do formato CPLEX LP é

⟨*specification*⟩ ::= ⟨*objective*⟩
⟨*restrictions*⟩?
⟨*bounds*⟩
⟨*general*⟩?
⟨*binary*⟩?
'`End`'

⟨*objective*⟩ ::= ⟨*goal*⟩ ⟨*name*⟩? ⟨*linear expression*⟩

⟨*goal*⟩ ::= '`MINIMIZE`' | '`MAXIMIZE`' | '`MIN`' | '`MAX`'

⟨*restrictions*⟩ ::= '`SUBJECT TO`' ⟨*restriction*⟩+

⟨*restriction*⟩ ::= ⟨*name*⟩? ⟨*linear expression*⟩ ⟨*cmp*⟩ ⟨*number*⟩

⟨*cmp*⟩ ::= '`<`' | '`<=`' | '`=`' | '`>`' | '`>=`'

[1] A gramática não contém as especificações "semi-continuous" e "SOS".

⟨*linear expression*⟩ ::= ⟨*number*⟩ ⟨*variable*⟩ (('+' | '-') ⟨*number*⟩ ⟨*variable*⟩)*

⟨*bounds*⟩ ::= '`BOUNDS`' ⟨*bound*⟩+

⟨*bound*⟩ ::= ⟨*name*⟩? (⟨*limit*⟩ '`<=`' ⟨*variable*⟩ '`<=`' ⟨*limit*⟩
| ⟨*limit*⟩ '`<=`' ⟨*variable*⟩
| ⟨*variable*⟩ '`<=`' ⟨*limit*⟩
| ⟨*variable*⟩ '`=`' ⟨*number*⟩
| ⟨*variable*⟩ '`free`')

⟨*limit*⟩ ::= '`infinity`' | '`-infinity`' | ⟨*number*⟩

⟨*general*⟩ ::= '`GENERAL`' ⟨*variable*⟩+

⟨*binary*⟩ ::= '`BINARY`' ⟨*variable*⟩+

Todas variáveis x tem a restrição padrão $0 \leq x \leq +\infty$. Caso outras limites são necessárias, eles devem ser informados na seção "BOUNDS". A seções "GENERAL" e "BINARY" permitem restringir variáveis para $\mathbb{Z}$ e $\{0, 1\}$, respectivamente.
As palvaras-chaves também podem ser escritem com letras minúscolas: o formato permite algumas abreviações não listadas acima (por exemplo, escrever "s.t" ao invés de "subject to").

Exemplo B.1
Problema de mochila 0-1 com 11 itens em formato CPLEX LP.

```
1  max 19x1+87x2+97x3+22x4+47x5+22x6+30x7+5x8+32x9+54x10+75x11
2  s.t
3  1x1+96x2+67x3+90x4+13x5+74x6+22x7+86x8+23x9+63x10+89x11<= 624
4  binary x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11
5  end
```

◊

B.2 AMPL

Objetos de modelagem

- Um modelo em AMPL consiste em
  - parâmetros,
  - variáveis,

  - restrições, e
  - objetiovos

- AMPL usa *conjuntos* (ou arrays de multiplas dimensões)

$$A : I \to D$$

  mapeam um conjunto de índices $I = I_1 \times \cdots \times I_n$ para valores D.

Formato

- Parte do modelo

```
<s1>
...
<sn>
end;
```

  com s_i é um comando ou uma declaração.

- Parte de dados

```
data
<d1>
...
<dn>
end;
```

Tipo de dados

- Números: `2.0,-4`
- Strings: `'Comida'`
- Conjuntos: `{2,3,4}`

Expressões numéricas

- Operações básicas: +,−,∗,/,div,mod,less,∗∗
  Exemplo: x less y
- Funções: abs, ceil , floor ,exp
  Exemplo: abs(−3)
- Condicional: **if** x>y **then** x **else** y

Expressões sobre strings

- AMPL converte números automaticamente em strings
- Concatenação de strings: &
  Exemplo: x & ' unidades'

Expressões para conjuntos de índices

- Única dimensão
  - t **in** S: variável "dummy" t, conjunto S
  - (t1 ,... tn) **in** S: para conjuntos de tuplos
  - S: sem nomear a variável
- Multiplas dimensões
  - {e1 ,..., en} com e_i uma dimensão (acima).
- Variáveis dummy servem para referenciar e modificar.
  Exemplo: (i−1) **in** S

Conjuntos

- Conjunto básico: {v1 ,..., vn}
- Valores: Considerados como conjuntos com conjunto de índices de dimensão 0
- Índices: [i1 ,..., **in**]
- Sequências: n1 ... n2 by d ou n1 ... n2
- Construção: setof I e: $\{e(i_1, \ldots, i_n) \mid (i_1, \ldots, i_n) \in I\}$
  Exemplo: `setof {j in A} abs(j)`

Operações de conjuntos

- X union Y: União $X \cup Y$
- X diff Y: Diferença $X \setminus Y$
- X symdiff Y: Diferença simétrica $(X \setminus Y) \cup (Y \setminus X)$
- X inter Y: Intersecção $X \cap Y$
- X cross Y: Produto cartesiano $X \times Y$

Expressões lógicas

- Interpretação de números: n vale "v", sse $n \neq 0$.
- Comparações simples <,<=,= ou ==,>=,>,<> ou !=
- Pertinencia x **in** Y, x not **in** Y, x !**in** Y
- Subconjunto X within Y, X !within Y, X not within Y
- Operadores lógicos: && ou and, || ou or, ! ou not
- Quantificação: com índices I, expressão booleana b
  forall I b: $\bigwedge_{(i_1,\ldots,i_n)\in I} b(i_1,\ldots,i_n)$
  exists I b $\bigvee_{(i_1,\ldots,i_n)\in I} b(i_1,\ldots,i_n)$

Declarações: Conjuntos
set N I [dimen n] [within S] [default e1] [:= e2]
param N I [**in** S] [<=,>=,!=,... n] [default e1] [:= e2]

- Nome N
- Conjunto de índices I (opcional)
- Conjunto de valores S
- Valor default e_1
- Valor inicial e_2

Declarações: Restrições e objetivos
subject to N I : e1 = e2 | e1 <= e2, e1 >= e2
minimize [I] : e
maximize [I] : e

Comandos

- solve: Resolve o sistema.
- check [I] : b: Valida expressão booleana b, erro caso falso.
- display [I] : e1 ,... en: Imprime expressões $e_1,\ldots,e_n$.
- printf [I] : fmt,e1 ,..., en: Imprime expressões $e-1,\ldots,e_n$ usando formato fmt.
- for I : c, `for I : {c1 ... cn}`: Laços.

Dados: Conjuntos
set N r1 ,... rn
Com nome N e records $r_1, \ldots, r_n$, cada record

- um tuplo: $v_1, \ldots, v|n$ Exemplo: 1 2, 1 3, 2 2, 2 7
- a definição de uma fatia $(v_1|*, v_2|*, \ldots, v_n|*)$: depois basta de listar os elementos com $*$. Exemplo: (1 *) 2 3, (2 *) 2 7
- uma matriz

Dados: Parâmetros
param N r1,...rn
Com nome N e records $r_1, \ldots, r_n$, cada record

- um valor $i_1, \ldots, i_n, v$
- a definição de uma fatia $[i_1|*, i_2|*, \ldots, i_n|*)$: depois basta definir índices com $*$.
- uma matriz
- uma tabela

Exemplo B.2 (Exemplo 1.1 em AMPL)

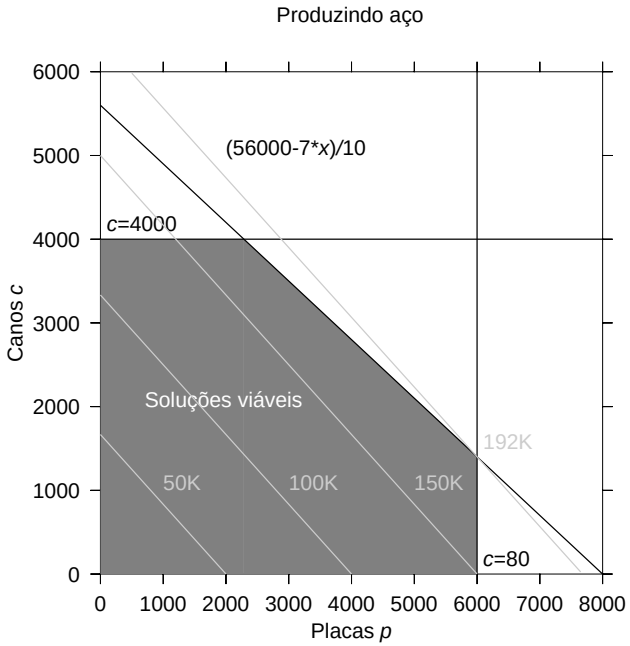
```
1 var c; # número de croissants
2 var s; # número de strudels
3 param lucro_croissant; # o lucro por croissant
4 param lucro_strudel; # o lucro por strudel
5 maximize lucro: lucro_croissant*c+lucro_strudel*s;
6 subject to ovo: c+1.5*s <= 150;
7 subject to acucar: 50*c+50*s <= 6000:
8 subject to croissant: c <= 80;
9 subject to strudel: s <= 60;
```

◊

C Soluções dos exercícios

Solução do exercício 5.6.

$$
\begin{aligned}
\textbf{maximiza} \quad & 25p + 30c \\
\textbf{sujeito a} \quad & p/200 + c/140 \leq 40 \iff 7p + 10c \leq 56000 \\
& p \leq 6000 \\
& c \leq 4000 \\
& c, p \geq 0
\end{aligned}
$$



A solução ótima é $p = 6000$, $c = 1400$ com valor 192000.

Solução do exercício 5.3.

$$\begin{array}{rl} \textbf{maximiza} & 2A + B \\ \textbf{sujeito a} & A <= 6000 \\ & B <= 7000 \\ & A + B <= 10000 \end{array}$$

Resposta: A=6000 e B=4000 e Z=16000

Solução do exercício 5.5.
São necessárias cinco variáveis:

- x1: número de pratos de lasanha comidos por Marcio
- x2: número de pratos de sopa comidos por Marcio
- x3: número de pratos de hambúrgueres comidos por Renato
- x4: número de pratos de massa comidos por vini
- x5: números de pratos de sopa comidos por vini

Formulação:

$$\begin{array}{rl} \textbf{maximiza} & x1 + x2 + x3 + x4 + x5 \\ \textbf{sujeito a} & 4 \geq x1 + x2 \geq 2 \\ & 5 \geq x3 \geq 2 \\ & 4 \geq x4 + x5 \geq 2 \\ & 70(x2 + x5) + 200x1 + 100x3 + 30x4 \leq 1000 \\ & 30(x2 + x5) + 100x1 + 100x3 + 100x4 \leq 800 \end{array}$$

Solução do exercício 5.7.
Usamos índices 1, 2 e 3 para os vôos Pelotas–Porto Alegre, Porto Alegre–Torres e Pelotas–Torres e variáveis a_1, a_2, a_3 para a categoria A, b_1, b_2, b_3 para categoria B e $c - 1, c_2, c_3$ para categoria C. A função objetivo é maximizar o lucro

$$z = 600a_1 + 320a_2 + 720a_3 + 440b_1 + 260b_2 + 560b_3 + 200c_1 + 160c_2 + 280c_3.$$

Temos que respeitar os limites de capacidade

$$a_1 + b_1 + c_1 + a_3 + b_3 + c_3 \leq 30$$
$$a_2 + b_2 + c_2 + a_3 + b_3 + c_3 \leq 30$$

e os limites da predição

$$\begin{array}{lll} a_1 \leq 4; & a_2 \leq 8; & a_3 \leq 3 \\ b_1 \leq 8; & b_2 \leq 13; & b_3 \leq 10 \\ c_1 \leq 22; & c_2 \leq 20; & c_3 \leq 18 \end{array}$$

Obviamente, todas variáveis também devem ser positivos.

Solução do exercício 5.8.

$$\begin{array}{rl} \textbf{maximiza} & z = 5x_1 + 5x_2 + 5x_3 \\ \textbf{sujeito a} & -6x_1 - 2x_2 - 9x_3 \leq 0 \\ & -9x_1 - 3x_2 + 3x_3 \leq 0 \\ & 9x_1 + 3x_2 - 3x_3 \leq 0 \\ & x_j \geq 0 \end{array}$$

$$\begin{array}{rl} \textbf{maximiza} & z = -6x_1 - 2x_2 - 6x_3 + 4x_4 + 4x_5 \\ \textbf{sujeito a} & -3x_1 - 8x_2 - 6x_3 - 7x_4 - 5x_5 \leq 3 \\ & 3x_1 + 8x_2 + 6x_3 + 7x_4 + 5x_5 \leq -3 \\ & 5x_1 - 7x_2 + 7x_3 + 7x_4 - 6x_5 \leq 6 \\ & x_1 - 9x_2 + 5x_3 + 7x_4 - 10x_5 \leq -6 \\ & -x_1 + 9x_2 - 5x_3 - 7x_4 + 10x_5 \leq 6 \\ & x_j \geq 0 \end{array}$$

$$\begin{array}{rl} \textbf{maximiza} & z = 7x_1 + 4x_2 + 8x_3 + 7x_4 - 9x_5 \\ \textbf{sujeito a} & -4x_1 - 1x_2 - 7x_3 - 8x_4 + 6x_5 \leq -2 \\ & 4x_1 + x_2 + 7x_3 + 8x_4 - 6x_5 \leq 2 \\ & -x_1 - 4x_2 - 2x_3 - 2x_4 + 7x_5 \leq 7 \\ & -8x_1 + 2x_2 + 8x_3 - 6x_4 - 7x_5 \leq -7 \\ & 8x_1 - 2x_2 - 8x_3 + 6x_4 + 7x_5 \leq 7 \\ & x_j \geq 0 \end{array}$$

$$
\begin{aligned}
\textbf{maximiza} \quad & z = 6x_1 - 5x_2 - 8x_3 - 7x_4 + 8x_5 \\
\textbf{sujeito a} \quad & -5x_1 - 2x_2 + x_3 - 9x_4 - 7x_5 \leq 9 \\
& 5x_1 + 2x_2 - x_3 + 9x_4 + 7x_5 \leq -9 \\
& 7x_1 + 7x_2 + 5x_3 - 3x_4 + x_5 \leq -8 \\
& -7x_1 - 7x_2 - 5x_3 + 3x_4 - x_5 \leq 8 \\
& -5x_1 - 3x_2 - 5x_3 + 9x_4 + 8x_5 \leq 0 \\
& x_j \geq 0
\end{aligned}
$$

Solução do exercício 5.9.
Solução com método Simplex, escolhendo como variável entrante sempre aquela com o maior coeficiente positivo (em negrito):

z	$=$	$25p$	$+30c$
w_1	$= 56000$	$-7p$	$-10c$
w_2	$= 6000$	$-p$	
w_3	$= 4000$		$\mathbf{-c}$

z	$= 120000$	$+25p$	$-30w_3$
w_1	$= 16000$	$\mathbf{-7p}$	$+10w_3$
w_2	$= 6000$	$-p$	
c	$= 4000$		$-w_3$

z	$= 1240000/7$	$-25/7p$	$+40/7w_3$
p	$= 16000/7$	$-1/7w_1$	$+10/7w_3$
w_2	$= 26000/7$	$+1/7w_1$	$-10/7\mathbf{w_3}$
c	$= 4000$		$-w_3$

z	$= 192000$	$-3w_1$	$-4w_2$
p	$= 6000$		$-w_2$
w_3	$= 2600$	$+1/10w_1$	$-7/10w_2$
c	$= 1400$	$-1/10w_1$	$+7/10w_2$

Solução do exercício 5.11.
Temos

$$\binom{2(n+1)}{n+1} = \binom{2n}{n}\frac{(2n+2)(2n+1)}{(n+1)^2} = \binom{2n}{n}\frac{2(2n+1)}{n+1}$$

e logo

$$\frac{2^2 n}{n+1}\binom{2n}{n} \leq \binom{2(n+1)}{n+1} \leq 2^2\binom{2n}{n}.$$

Logo, por indução $(1/2n)2^{2n} \leq \binom{2n}{n} \leq 2^{2n}$.

Solução do exercício 10.2.
O sistema inicial

$$\begin{array}{rrrr} z = & & x_1 & +3x_2 \\ w_1 = & -2 & +x_1 & \\ w_2 = & 3 & & -x_2 \\ w_3 = & -4 & +x_1 & +x_2 \\ w_4 = & 12 & -3x_1 & -x_2 \end{array}$$

não é primalmente nem dualmente viável. Aplicando a fase I (pivots x_0–w_3, x_0–x_1) e depois fase II (pivots x_2–w_1, w_3–w_2, w_1–w_4) gera o dicionário final

$$\begin{array}{rrrr} z = & 12 & -8/3w_2 & -1/3w_4 \\ x_2 = & 3 & -w_2 & \\ w_3 = & 2 & -2/3w_2 & -1/3w_4 \\ x_1 = & 3 & +1/3w_2 & -1/3w_4 \\ w_1 = & 1 & +1/3w_2 & -1/3w_4 \end{array}$$

cuja solução $x_1 = 3$, $x_2 = 3$ já é inteira.
No segundo sistema começamos com o dicionário

$$\begin{array}{rrrr} z = & & x_1 & -2x_2 \\ w_1 = & 60 & +11x_1 & -15x_2 \\ w_2 = & 24 & -4x_1 & -3x_2 \\ w_3 = & 59 & -10x_1 & +5x_2 \end{array}$$

e um pivot x_1–w_3 gera a solução ótimo fracionária

$$\begin{array}{rrrr} z = & 4.9 & -0.1w_3 & -1.5x_2 \\ w_1 = & 113.9 & -1.1w_3 & -9.5x_2 \\ w_2 = & 4.4 & +0.4w_3 & -5x_2 \\ x_1 = & 4.9 & -0.1w_3 & +0.5x_2 \end{array}$$

e a linha terceira linha (x_1) gera o corte

$$w_4 = \quad -0.9 \quad +0.1w_3 \quad +0.5x_2$$

Com o pivot w_4–w_3 obtemos a solução ótima inteira

$$\begin{aligned} z &= 4 - w_4 - x_2 \\ w_1 &= 104 - 11w_4 - 4x_2 \\ w_2 &= 8 + 4w_4 - 7x_2 \\ x_1 &= 4 - w_4 + 1x_2 \\ w_3 &= 9 + 10w_4 - 5x_2 \end{aligned}$$

Solução do exercício 10.3.

Conjunto independente máximo Com variáveis indicadores x_v, $v \in V$ temos o programa inteiro

$$\begin{aligned} \textbf{maximiza} \quad & \sum_{v \in V} x_v \\ \textbf{sujeito a} \quad & x_u + x_v \leq 1, && \forall \{u, v\} \in A \\ & x_v \in \mathbb{B}, && \forall v \in V. \end{aligned} \tag{C.1}$$

A equação C.1 garante que cada aresta possui no máximo um nó incidente.

Casamento perfeito com peso máximo Sejam x_a, $a \in A$ variáveis indicadores para a seleção de cada aresta. Com isso, obtemos o programa inteiro

$$\begin{aligned} \textbf{maximiza} \quad & \sum_{a \in A} p(a) x_a \\ \textbf{sujeito a} \quad & \sum_{u \in N(v)} x_{\{u,v\}} = 1, && \forall v \in V \\ & x_a \in \mathbb{B}, && \forall v \in V. \end{aligned} \tag{C.2}$$

A equação C.2 garante que cada nó possui exatamente um vizinho.

Problema de transporte Sejam x_{ij} variáveis inteiras, que correspondem com o número de produtos transportados do depósito i para cliente j. Então

$$\begin{aligned}
\textbf{minimiza} \quad & \sum_{\substack{1\le i\le n\\ 1\le j\le m}} c_{ij}x_{ij} \\
\textbf{sujeito a} \quad & \sum_{1\le j\le m} x_{ij} = p_i, \quad \forall 1 \le i \le n \quad \text{cada depósito manda todo estoque} \\
& \sum_{1\le i\le n} x_{ij} = d_j, \quad \forall 1 \le j \le m \quad \text{cada cliente recebe a sua demanda} \\
& x_{ij} \in \mathbb{Z}^+.
\end{aligned}$$

Conjunto dominante Sejam x_v, $v \in V$ variáveis indicadores para seleção de vértices. Temos o programa inteiro

$$\begin{aligned}
\textbf{minimiza} \quad & \sum_{v\in V} x_v \\
\textbf{sujeito a} \quad & x_v + \sum_{u\in N(v)} x_u \ge 1, \quad \forall v \in V \quad \text{nó ou vizinho selecionado} \\
& x_v \in \mathbb{B}, \qquad\qquad \forall v \in V.
\end{aligned}$$

Solução do exercício 10.4.
Seja $d_1 d_2 \dots d_n$ a entrada, e o objetivo selecionar $m \le n$ dígitos da entrada. Seja $x_{ij} \in \mathbb{B}$ um indicador que o dígito i da entrada seria selecionado como dígito j da saida, $1 \le i \le n$, $1 \le j \le m$. Então

$$\begin{aligned}
\textbf{maximiza} \quad & \sum_{i,j} x_{ij} d_i 10^{m-j} \\
\textbf{sujeito a} \quad & \sum_{i} x_{ij} = 1, && \forall j && \text{(C.3)} \\
& \sum_{j} x_{ij} \le 1, && \forall i && \text{(C.4)} \\
& x_{ij} = 0, && \forall j > i, && \text{(C.5)} \\
& x_{kl} \le 1 - x_{ij}, && \forall k > i, l < j. && \text{(C.6)}
\end{aligned}$$

A função das equações é a seguinte:

- Equação C.3 garante que tem exatamente um dígito em cada posição.

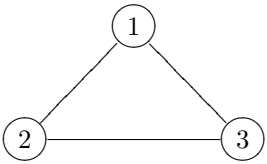
- Equação C.4 garante que cada dígito é selecionado no máximo uma vez.
- Equação C.5 garante que dígito i aparece somente a partir da posição j.
- Equação C.4 proibe inversões.

Solução do exercício 10.5.
Existem 21 sets diferentes, cada um com consumo diferente das 9 cartas. Seja $A\mathbb{R}^{9\times 21}$ uma matriz, que contém em cada das 21 coluna o número de cartas de cada set. Além disso, seja $b \in \mathbb{R}^9$ o número de cartas disponíveis. Usando variáveis inteiros $x \in \mathbb{Z}^{21}$ que representam o número de sets formandos de cada tipo de set diferentes, temos a formulação

$$\begin{aligned} \textbf{maximiza} \quad & \sum_{1\leq i\leq 21} x_i \\ \textbf{sujeito a} \quad & Ax \leq b \\ & x \geq 0. \end{aligned}$$

Solução do exercício 10.6.

Conjunto independente máximo A matriz de coeficientes contém dois coeficientes igual 1 em cada linha, que correspondem com uma aresta, mas geralmente não é totalmente unimodular. Por exemplo, o grafo completo com três vértices K_3



gera a matriz de coeficientes

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

cuja determinante é -2. A solução ótima da relaxação inteira $0 \leq x_i \leq 1$ é $x_1 = x_2 = x_3 = 1/2$ com valor $3/2$. (Observação: A transposta dessa matriz satisfaz os critérios (i) e (ii) da nossa proposição, e caso o grafo é bi-partido, também o critério (iii). Portanto *Conjunto independente máximo* pode ser resolvido em tempo polinomial em grafos bi-partidos).

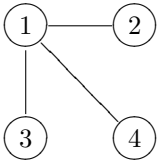
Casamento perfeito com peso máximo A matriz de coeficientes satisfaz critério (i). Ela tem uma linha para cada vértice e uma coluna para cada aresta do grafo. Como cada aresta é incidente a exatamente dois vértices, ela também satisfaz (ii). Finalmente, a bi-partição $V_1 \dot\cup V_2$ do grafo gera uma bi-partição das linhas que satisfaz (iii). Portanto, a matriz é TU, e o *Casamento perfeito com peso máximo* pode ser resolvido em tempo polinomial usando a relaxação linear.

Problema de transporte A matriz de coeficientes satisfaz critério (i). Podemos representar o problema como grafo bi-partido completo $K_{n,m}$ entre os depósitos e os clientes. Desta forma, com o mesmo argumento que no último problema, podemos ver, que os critérios (ii) e (iii) são satisfeitos.

Conjunto dominante A matriz de coeficientes satisfaz critério (i), mas não critério (ii): cada linha e coluna correspondente com vértice v contém $|N(v)|+1$ coeficientes não-nulos. Mas, não é obviou se a matriz mesmo assim não é TU (lembra que o critério é suficiente, mas não necessário). O K_3 acima, por exemplo, gera a matriz

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

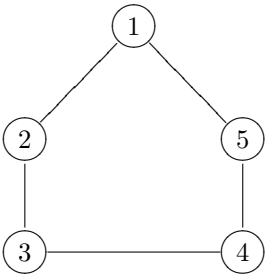
que *é* TU. Um contra-exemplo seria o grafo bi-partido $K_{1,3}$



que gera a matriz de coeficientes

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

com determinante -2. Isso não prova ainda que a relaxação linear não produz resultados inteiros ótimos. De fato, nesse exemplo a solução ótima da relaxação inteira é a solução ótima inteira $D = \{1\}$.
Um verdadeiro contra-exemplo é um ciclo com cinco vértices C_5



com matriz

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

(cuja determinante é 3). A relaxação linear desse sistema tem a solução ótimo $x_1 = x_2 = x_3 = x_4 = x_5 = 1/3$ com valor $5/3$ que não é inteira.

Solução do exercício 10.7.

Cobertura por arcos

$$\begin{aligned} \textbf{maximiza} \quad & \sum_{e \in E} c_e x_e \\ \textbf{sujeito a} \quad & \sum_{u \in N(v)} x_{uv} \geq 1, \qquad \forall v \in V \\ & x_e \in \mathbb{B}. \end{aligned}$$

Observe que esse problema é redutível a um emparalhamento perfeito máximo e portanto possui solução em tempo polinomial.

Conjunto dominante de arcos

$$\begin{aligned} \textbf{maximiza} \quad & \sum_{e \in E} c_e x_e \\ \textbf{sujeito a} \quad & \sum_{\substack{e' \in E \\ e \cap e' \neq \emptyset}} x_{e'} \geq 1, \qquad \forall e \in E \\ & x_e \in \mathbb{B}. \end{aligned}$$

Coloração de grafos Seja $n = |V|$.

$$\begin{aligned}
\textbf{minimiza} \quad & \sum_{1 \le j \le n} c_j \\
\textbf{sujeito a} \quad & \sum_{1 \le j \le n} x_{vj} = 1, && \forall v \in V && \text{(C.7)} \\
& x_{ui} + x_{vi} \le 1, && \forall \{u, v\} \in E, 1 \le i \le n && \text{(C.8)} \\
& nc_j \ge \sum_{v \in V} x_{vj}, && \forall 1 \le j \le n && \text{(C.9)} \\
& x_{vi}, c_j \in \mathbb{B}.
\end{aligned}$$

- Equação C.7 garante que todo vértice recebe exatamente uma cor.
- Equação C.8 garante que vértices adjacentes recebem cores diferentes.
- Equação C.9 garante que $c_j = 1$ caso cor j for usada.

Clique mínimo ponderado

$$\begin{aligned}
\textbf{minimiza} \quad & \sum_{v \in V} c_v x_v \\
\textbf{sujeito a} \quad & x_u + x_v \le 1, && \forall \{u, v\} \notin E && \text{(C.10)} \\
& x_v \in \mathbb{B}.
\end{aligned}$$

Equação C.10 garante que não existe um par de vértices selecionados que não são vizinhos.

Subgrafo cúbico x_e indica se o arco e é selecionado, e y_e indica se ele possui grau 0 (caso contrário grau 3).

$$\begin{aligned}
\textbf{minimiza} \quad & \sum_{e \in E} x_e \\
\textbf{sujeito a} \quad & \sum_{e \in N(v)} x_e \le 0 + |E|(1 - y_e) \\
& \sum_{e \in N(v)} x_e \le 3 + |E| y_e \\
& - \sum_{e \in N(v)} x_e \le -3 + 3y_e
\end{aligned}$$

Observe que o grau de cada vértice é limitado por $|E|$.

Solução do exercício 10.8.
Sejam $x_i \in \mathbb{B}, 1 \leq i \leq 7$ variáveis que definem a escolha do projeto i. Então temos

maximiza	$17x_1 + 10x_2 + 15x_3$	
	$+ 19x_4 + 7x_5 + 13x_6 + 9x_7$	
sujeito a	$43x_1 + 28x_2 + 34x_3 + 48x_4$	
	$+ 17x_5 + 32x_6 + 23x_7 \leq 100$	Limite do capital
	$x_1 + x_2 \leq 1$	Projetos 1,2 mutualmente exclusivos
	$x_3 + x_4 \leq 1$	Projetos 3,4 mutualmente exclusivos
	$x_3 + x_4 \leq x_1 + x_2$	Projeto 3 ou 4 somente se 1 ou 2

http://www.inf.ufrgs.br/~mrpritt/e6q2.mod

```
1  set projetos := 1 .. 7;
2  param lucro { projetos };
3  param custo { projetos };
4
5  var fazer { projetos } binary;
6
7  maximize M: sum { i in projetos } lucro[i]*fazer[i];
8  subject to S1:
9    sum { i in projetos } custo[i]*fazer[i] <= 100;
10 subject to S2: fazer[1]+fazer[2] <= 1;
11 subject to S3: fazer[3]+fazer[4] <= 1;
12 subject to S4: fazer[3]+fazer[4] <= fazer[1]+fazer[2];
13
14 data;
15 param lucro :=  1 17 2 10 3 15 4 19 5 7  6 13 7 9;
16 param custo :=  1 43 2 28 3 34 4 48 5 17 6 32 7 23;
17 end;
```

Solução: Selecionar projetos 1,3,7 com lucro de 41MR$.

Solução do exercício 10.9.
Seja $f \in \mathbb{B}$ uma variável que determina qual fábrica vai ser usada (fábrica 1, caso $f = 0$, fábrica 2, caso $f = 1$), $b_i \in \mathbb{B}$ uma variável binária que determina,

se brinquedo i vai ser produzido e $u_i \in \mathbb{Z}$ as unidades produzidas de brinquedo i (sempre com $1 \leq i \leq 2$).

maximiza	$10u_1 + 15u_2 - 50000b_1 - 80000b_2$	
sujeito a	$u_i \leq Mb_i$	Permitir unidades somente se tem pro
	$u_1/50 + u_2/40 \leq 500 + fM$	Limite fábrica 1, se selecionada
	$u_1/40 + u_2/25 \leq 700 + (1-f)M$	Limite fábrica 2, se selecionada

A constante M deve ser suficientemente grande tal que ela efetivamente não restringe as unidades. Dessa forma, se a fábrica 1 está selecionada, a terceira restrição (da fábrica 2) não se aplica e vice versa.

http://www.inf.ufrgs.br/~mrpritt/e6q3.mod

```
1  var f binary;
2  var b { brinquedos } binary;
3  var u { brinquedos } integer, >= 0;
4  param inicial { brinquedos };
5  param lucro { brinquedos };
6  param prodfab1 { brinquedos };
7  param prodfab2 { brinquedos };
8  param M := 35000;
9
10 maximize Lucro:
11   sum { i in brinquedos } u[i]*lucro[i]
12   - ( sum { i in brinquedos } inicial[i]*b[i] );
13 subject to PermitirProducao { i in brinquedos }:
14   u[i] <= M*b[i];
15 subject to LimiteFab1 :
16   sum { i in brinquedos }
17     u[i]*prodfab1[i] <= 500 + f*M;
18 subject to LimiteFab2 :
19   sum { i in brinquedos }
20     u[i]*prodfab2[i] <= 700 + (1-f)*M;
21
22 data;
23 param inicial := 1 50000 2 80000;
24 param lucro := 1 10 2 15;
25 param prodfab1 := 1 0.020 2 0.025;
```

```
26 param prodfab2 := 1 0.025 2 0.040;
```

Solução: Produzir 28000 unidades do brinquedo 1 na fábrica 2, com lucro 230KR$.

Solução do exercício 10.10.
Sejam $a_i \in \mathbb{B}$ uma variável que determina se avião i vai ser produzido e $u_i \in \mathbb{Z}$ as unidadas produzidas.

maximiza	$2u_1 + 3u_2 + 0.2u_3 - 3a_1 - 2a_2$	
sujeito a	$0.2u_1 + 0.4u_3 + 0.2u_3 \leq 1$	Limite de capacidade
	$u_i \leq 5b_i$	Permitir unidades somente se for p
	$u_1 \leq 3$	Limite avião 1
	$u_2 \leq 2$	Limite avião 2
	$u_3 \leq 5$	Limite avião 3

http://www.inf.ufrgs.br/~mrpritt/e6q4.mod

```
27 param custo { avioes };
28 param lucro { avioes };
29 param capacidade { avioes };
30 param demanda { avioes };
31 var produzir { avioes } binary;
32 var unidades { avioes } integer, >= 0;
33
34 maximize Lucro:
35   sum { i in avioes }
36     (lucro[i]*unidades[i]-custo[i]*produzir[i]);
37 subject to LimiteCapacidade:
38   sum { i in avioes } unidades[i]*capacidade[i] <= 1;
39 subject to PermitirProducao { i in avioes }:
40   unidades[i] <= 5*produzir[i];
41 subject to LimiteDemanda { i in avioes }:
42   unidades[i] <= demanda[i];
43
44 data;
45 param : custo lucro capacidade demanda :=
```

```
46  1  3  2     0.2  3
47  2  2  3     0.4  2
48  3  0  0.8   0.2  5
49  ;
```

Solução: Produzir dois aviões para cliente 2, e um para cliente 3, com lucro 4.8 MR$.

Bibliografia

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and approximation – Combinatorial Optimization Problems and their Approximability Properties.* Springer-Verlag, 1999. INF 510.5 C737.

[2] J. Clausen. Branch and bound algorithms – principles and examples, 1999.

[3] N. Maculan and M. H. C. Fampa. *Otimização linear.* Editora UnB, 2006. INF 65.012.122 M175o.

[4] R. J. Vanderbei. *Linear programming: Foundations and Extensions.* Kluwer, 2nd edition, 2001.

[5] H. P. Williams. Fourier's method of linear programming and its dual. *The American Mathematical Monthly*, 93(9):681–695, 1986.

[6] L. A. Wolsey and G. L. Nemhauser. *Integer and Combinatorial Optimization.* Wiley, 1999.

Índice