IMPLEMENTAÇÃO DE ALGORITMOS SIMPLEX E PONTOS INTERIORES PARA PROGRAMAÇÃO LINEAR

LEIZER DE LIMA PINTO
MARCO ANTONIO FIGUEIREDO MENEZES

Resumo: *o objetivo deste trabalho é estudar e implementar algoritmos das famílias simplex, pontos interiores e ponto-interior-inviável para Programação Linear, contribuindo com a criação do Laboratório de Programação Linear (LabPL – http://www.ucg.br/Institutos/LabPL/Index.html).*

Palavras-chave: *Programação Linear, Algoritmo Simplex, Algoritmos de Pontos Interiores, Complexidade, Processo Ensino-Aprendizagem.*

Neste trabalho, estamos interessados na implementação dos algoritmos simplex tabular, de pontos interiores e de ponto-interior-inviável para Programação Linear (PL). O Problema de Programação Linear (PPL) é o problema de Otimização onde se deseja minimizar (ou maximizar) uma função linear (contínua) com restrições também lineares (contínuas).

Publicado em 1951 por Dantzig [7], o simplex foi o primeiro método efetivo desenvolvido para resolver um PPL. Com o início da análise de complexidade, surgiu a seguinte pergunta: quão bom é o algoritmo simplex? A resposta para esta pergunta veio em 1972 com Klee e Minty [15], demonstrando que o

algoritmo simplex tem complexidade exponencial (pior caso) para uma certa escolha na entrada da base, apesar de funcionar bem na prática. A partir daí, surgiram outras classes de algoritmos para PL com complexidade inferior à do simplex. Um deles, com complexidade polinomial e que funciona bem na prática, é o método de pontos interiores, publicado por Karmarkar [12] em 1984.

O que faremos aqui será reescrever o que já existe na literatura sobre PL com uma perspectiva para discutir a complexidade, em número de iterações, em algoritmos de ponto-interior-inviável. Dividiremos este trabalho da seguinte forma: na seção 2, apresentaremos os problemas de PL primal, dual e primal-dual; na seção 3, apresentaremos os problemas teste que resolveremos com nossas implementações; na seção 4, apresentaremos idéias básicas dos algoritmos simplex primal, dual e primal-dual, dos algoritmos de pontos interiores (primais-duais) de passos curtos e preditor-corretor, e dos algoritmos de ponto-interior-inviável (primais-duais) preditor-corretor e homogêneo e auto-dual para PL, a página do LabPL referente ao algoritmo simplex tabular primal fases 1 e 2, e o enunciado do algoritmo de ponto-interior-inviável homogêneo e auto-dual; na seção 5, apresentaremos algumas implementações e, finalmente, faremos algumas análises comparativas em torno destas implementações.

OS PROBLEMAS DE PROGRAMAÇÃO LINEAR

O PPL primal, no formato padrão, é o seguinte problema de Otimização:

$$\begin{aligned} (P) \quad & \text{minimizar} \quad c^T x \\ & \text{sujeito a}: \; Ax = b \\ & \qquad\qquad x \geq 0, \end{aligned}$$

onde são dados $A \in R^{m \times n}$, $b \in R^m$ e $c \in R^n$. Supomos, sem perda de generalidade, que $posto(A) = m$ e $0 < m < n$.

Associado a todo PPL existe um outro PPL chamado dual. E o dual de *(P)* é definido por:

$$\begin{aligned} (D) \quad & \text{maximizar} \quad b^T y \\ & \text{sujeito a}: \; A^T y + s = c \\ & s \geq 0, \end{aligned}$$

onde $y \in R^m$ é o vetor de variáveis duais e incluímos explicitamente um vetor com componentes não negativas $s \in R^n$ denominado folga dual.

Relacionando o par de problemas primal *(P)* e dual *(D)*, existe um outro PPL chamado primal-dual, que consiste em encontrar, se existir, uma solução para o seguinte sistema de equações e inequações:

$$\begin{aligned} & Ax = b \\ (PD) \quad & A^T y + s = c \\ & xs = 0 \\ & x, s \geq 0, \end{aligned}$$

onde $xs = 0$ significa $x_j s_j = 0, \;\; j = 1, \ldots, n.$

As condições de otimalidade para o par de problemas primal e dual coincidem com as condições de Karush-Kuhn-Tucker, a saber: *x* é uma solução ótima de *(P)* se, e somente se, existe um par *(y,s)* tal que o sistema *(PD)* é satisfeito.

Consideremos os problemas *(P)*, *(D)* e *(PD)*. O conjunto viável e o conjunto de pontos interiores viáveis do problema primal *(P)* são, respectivamente,

$$X = \{x \in R^n; Ax = b, x \geq 0\} \quad e \quad X^0 = \{x \in X; x > 0\}.$$

O conjunto viável e o conjunto de pontos interiores viáveis do problema dual *(D)* são, respectivamente,

$$S = \{(y, s) \in R^m \times R^n;\ A^T y + s = c,\ s \geq 0\} \quad e \quad S^0 = \{(y, s) \in S;\ s > 0\}.$$

O conjunto viável e o conjunto de pontos interiores viáveis do problema primal-dual *(PD)* são, respectivamente,

$$F = \{(x, s) \in R^n \times R^n;\ x \in X,\ s \in \tilde{S}\} \quad e \quad F^0 = \{(x, s) \in F;\ (x, s) > 0\},$$

$$\text{onde } \tilde{S} = \{s \in R^n; (y, s) \in S \text{ para algum } y \in R^m\}.$$

Apresentaremos a seguir, os problemas que resolveremos através de nossas implementações.

Problemas Teste

O problema que apresentaremos a seguir é um PPL clássico na literatura da PL, o qual foi publicado por Beale [1] em 1955 para mostrar que o algoritmo simplex pode não convergir, dependendo das escolhas para a entrada e para a saída da base. Segue-se o exemplo de Beale:

$$\begin{aligned} \textit{(Be)} \quad \text{minimizar} \quad & -3/4\, x_1 + 20x_2 - 1/2\, x_3 + 6x_4 \\ \text{sujeito a :} \quad & 1/4\, x_1 - 8x_2 - x_3 + 9x_4 + x_5 = 0 \\ & 1/2\, x_1 - 12x_2 - 1/2\, x_3 + 3x_4 + x_6 = 0 \\ & x_3 + x_7 = 1 \\ & x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6,\ x_7 \geq 0. \end{aligned}$$

Outro PPL clássico na literatura da PL é o exemplo de Klee e Minty [15], o qual foi publicado em 1972 para mostrar que o algoritmo simplex tem complexidade exponencial (pior caso) para uma certa escolha na entrada da base. Segue-se o exemplo de Klee e Minty no formato padrão:

$$
\begin{aligned}
(KM)_P \quad & \text{minimizar } -x_n \\
& \text{sujeito a : } \quad x_1 - x_{n+1} = \varepsilon \\
& \qquad\qquad x_1 + x_{n+2} = 1 \\
& \qquad\qquad \varepsilon x_{i-1} - x_i + x_{n+i+1} = 0, \quad i = 2,\ldots,n \\
& \qquad\qquad \varepsilon x_{i-1} + x_i + x_{2n+i} = 1, \quad i = 2,\ldots,n \\
& \qquad\qquad x_i \geq 0, \qquad\qquad i = 1,\ldots,3n,
\end{aligned}
$$

onde $\varepsilon \in (0,\ 1/2)$. Denotamos por $(KM)_2$ o PPL $(KM)_P$ para $n = 2$, e usamos $\varepsilon = 1/4$.

Um problema prático e de pequeno porte (Müller [21]), consiste em decidir a quantidade (em hectare) para plantar de cada alimento (soja, milho, arroz e feijão), em cada uma das oito glebas (pedaço de terra) de uma área total de 350 hectares, de modo a maximizar o lucro.

Considere x_{ij} a quantidade a ser plantada do alimento j ($j = 1$, soja; $j = 2$, milho; $j = 3$, arroz; e $j = 4$, feijão) na gleba i ($i = 1,\ldots,8$). Segue-se o PPL:

$$
\begin{aligned}
(PA) \quad & \text{maximizar } \sum_{i=1}^{8}\sum_{j=1}^{4} c_{ij}x_{ij} \\
& \text{sujeito a : } \sum_{i=1}^{8}\sum_{j=1}^{4} x_{ij} \leq A_{total} \\
& \qquad \sum_{j=1}^{4} x_{ij} \leq g_i,\ i = 1,\ldots,8 \\
& \qquad x_{ij} \leq g_i,\ i = 1,\ldots,8 \ \ e \ \ j = 1,\ldots,4 \\
& \qquad 50x_{11} + 48x_{21} + 48x_{31} + 50x_{41} + 35x_{51} + 32x_{61} + 35x_{71} + 38x_{81} \geq S \\
& \qquad 130x_{12} + 120x_{22} + 140x_{32} + 100x_{42} + 70x_{52} + 65x_{62} + 68x_{72} + 95x_{82} \geq M \\
& \qquad x_{13} + x_{23} + x_{33} + x_{43} + x_{53} + x_{63} + x_{73} + x_{83} \geq A_a \\
& \qquad x_{14} + x_{24} + x_{34} + x_{44} + x_{54} + x_{64} + x_{74} + x_{84} \leq A_f \\
& \qquad x_{ij} \geq 0,\ i = 1,\ldots,8 \ \ e \ \ j = 1,\ldots,4,
\end{aligned}
$$

onde $c^T = [c_{11}\ c_{12}\ c_{13}\ c_{14}\ \ldots\ c_{81}\ c_{82}\ c_{83}\ c_{84}] = [\,1200$ 1040 240 1450 1080 910 300 3380 1065 1728 300 1890 1320 700 280 1220 365 -120 600 610 160 -380 595 280 360 -171 620 585 610 410 665 900 $]$, $[A_{total}\ S\ M\ A_a\ A_f] = [\,350\ \ 2500\ \ 3000\ \ 150\ \ 80\,]$ e $g^T = [10\ \ 18\ \ 22\ \ 49\ \ 51\ \ 54\ \ 77\ \ 69]$.

O problema *(PA)* pode ser simplificado, retirando-se a primeira restrição e as 32 restrições $x_{ij} \leq g_i\ \ (i = 1,\ldots,8\ \ e\ \ j = 1,\ldots,4)$. Denominaremos tal PPL por *(PA) simplificado*.

Os métodos simplex e de pontos interiores são os mais utilizados atualmente na prática da PL. A seguir, apresentaremos as idéias básicas de alguns algoritmos destas famílias.

ALGORITMOS PARA PL

Simplex

Publicado em 1951 por Dantzig [7], o simplex foi o primeiro método efetivo desenvolvido para resolver um PPL. Sua idéia geométrica consiste em caminhar de ponto extremo a ponto extremo adjacente do conjunto viável de um PPL, otimizando o valor da função objetivo com relação aos pontos extremos anteriormente visitados. Por outro lado, pelos fundamentos da PL (veja por exemplo, Bregalda, Oliveira e Bornstein [4]), temos que, quando não vazio, o conjunto viável de um PPL possui um número finito não nulo de pontos extremos; além disso, pelo Teorema Fundamental da PL, se o PPL admitir solução ótima, uma poderá ser encontrada em pelo menos um ponto extremo de seu conjunto viável. Desta forma, o método simplex se baseia fortemente nestes resultados.

Devido as relações entre os problemas primal *(P)* e dual *(D)*, obtidas pelos teoremas de dualidade (veja por exemplo, Saigal

[26]), podemos encontrar uma solução para *(P)* através da resolução de *(D)*. Assim, o que basicamente difere o algoritmo simplex primal (Dantzig [7]) dos algoritmos simplex dual (Lemke [16]) e simplex primal-dual (Dantzig, Ford e Fulkerson [8]), é que o simplex primal resolve diretamente o problema *(P)*, enquanto os algoritmos simplex dual e primal-dual obtêm uma solução para *(P)* através da resolução de *(D)*. Por outro lado, uma diferença básica entre o simplex dual e primal-dual é que no simplex dual necessitamos de uma solução básica viável inicial para o problema dual, enquanto que no simplex primal-dual basta uma solução dual viável. Deste modo, utilizamos o algoritmo simplex primal quando conhecemos uma solução básica viável para o problema *(P)*; o algoritmo simplex dual quando conhecemos uma solução básica viável para o problema *(D)*; e o algoritmo primal-dual quando conhecemos uma solução dual viável. Apesar disso, o método de duas fases busca viabilidade primal (problema fase 1) e otimalidade primal (problema fase 2), não necessitando de um ponto viável inicial. Todavia, os algoritmos simplex necessitam de uma regra anti-ciclagem. Aqui utilizaremos a regra do menor índice, conforme Bland [3].

Nossas implementações dos algoritmos simplex tabular primal, dual e primal-dual foram desenvolvidas na linguagem de programação PHP, a qual nos permite construir páginas dinâmicas para a *internet*. Utilizamos um computador pessoal com processador Pentium 800MHz e 320MB de memória RAM. E, também, utilizamos a versão 4.0 do PHP com um servidor Apache 1.3. Estas implementações estão contidas nas páginas do LabPL (veja [30]), de modo que os interessados em PL podem resolver seus problemas de pequeno porte *on-line*.

Consideremos o exemplo de Beale *(Be)* apresentado na seção anterior. Vejamos, conforme Figura 1, a resolução de *(Be)* através da página do LabPL referente ao algoritmo simplex tabular primal fases 1 e 2.

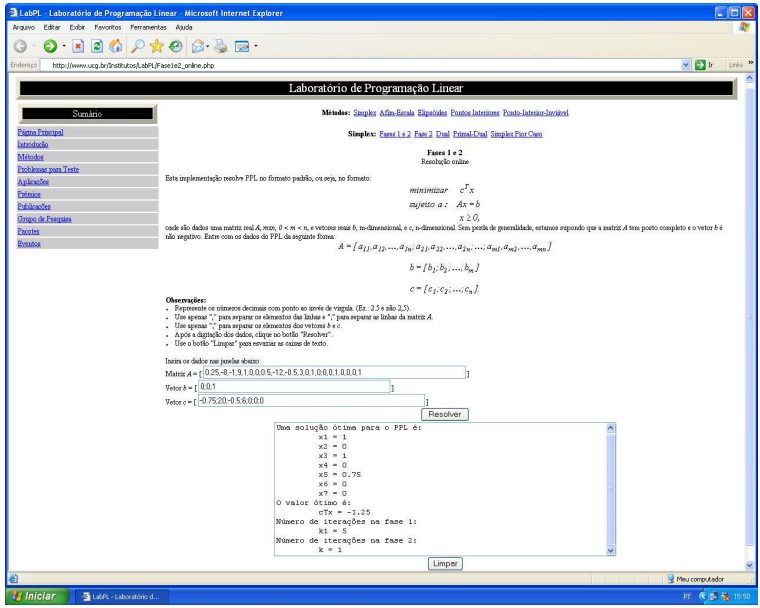

Figura 1: Página do algoritmo simplex primal fases 1 e 2 no site do LabPL.

Pontos Interiores

Publicado em 1984 por Karmarkar [12], o método de pontos interiores apresenta-se diferentemente do método simplex, uma vez que aqui é proibitivo caminhar pela fronteira do conjunto viável de um PPL. Os algoritmos de pontos interiores geram pontos interiores (pontos no primeiro ortante com coordenadas estritamente positivas) viáveis, 'próximos à trajetória central', até obter uma solução aproximadamente ótima. Estes algoritmos trabalham com a hipótese de que é dado um ponto interior viável inicial ('próximo à trajetória central'). Além disso, para se obter uma solução ótima, é necessário um procedimento chamado purificação (veja por exemplo, Gonzaga [9]). Aqui não utilizamos o algoritmo de purificação, quer dizer, consideraremos soluções aproximadamente ótimas. Finalmente, para a análise de

complexidade destes algoritmos, supõe-se que os dados iniciais são inteiros, conforme Khachiyan [13] e [14].

Hipótese: Os algoritmos de pontos interiores que implementamos neste trabalho são algoritmos primais-duais: passos curtos e preditor-corretor. Assim, supomos que F^0 é não vazio. Esta hipótese garante as condições de otimalidade para o problema primal-dual *(PD)* (veja por exemplo, Gonzaga [10]).

Trajetória central: Definimos a trajetória central perturbando o problema *(PD)*, a saber: a função

$$\mu > 0 \mapsto (x(\mu), y(\mu), s(\mu))$$

satisfazendo o problema perturbado

$$(PD)_\mu \quad \begin{aligned} Ax &= b \\ A^T y + s &= c \\ xs &= \mu e \\ x, s &\geq 0, \end{aligned}$$

onde $e \in R^n$ é um vetor de uns, é uma curva diferenciável denominada trajetória central.

A trajetória central é a curva formada pelos pontos centrais primais-duais associados a $\mu > 0,$ cujo limite quando $\mu \to 0$ é uma solução ótima do problema primal-dual (veja por exemplo, Ye [27])

Medida de proximidade: Dado $\mu > 0,$ queremos encontrar o par $(x, s) \in F^0$ tal que $xs/\mu = e.$ A palavra 'próximo à trajetória central' se refere a alguma medida de proximidade que definimos agora:

$$(x,s,\mu)\in F^0\times R_{++}\mapsto \delta(x,s,\mu)=\left\|\frac{xs}{\mu}-e\right\|.$$

Desta forma, dado o parâmetro $\alpha\in(0,1)$, definimos a vizinhança da trajetória central por

$$V(\alpha)=\{(x,s,\mu)\in F^0\times R_{++};\ \delta(x,s,\mu)\le\alpha\}.$$

Passo de Newton: Idealmente, gostaríamos de encontrar

$$x^+=x+u,\quad y^+=y+w\quad e\quad s^+=s+v,$$

tais que $(x^+,s^+)\in F$ e $x^+s^+=\mu e$. A direção (u,w,v) deve ser viável, isto é, $A(x+u)=Ax$ e $A^T(y+w)+(s+v)=A^Ty+s$. Isto é equivalente a dizer que *u* pertence ao espaço nulo de *A* e *v* pertence ao espaço linha de *A*. O passo de Newton resolve isto aproximadamente linearizando o sistema perturbado $(PD)_\mu$, a saber:

$$(N)\qquad \begin{aligned} Au &= 0\\ A^Tw+v &= 0\\ su+xv &= -xs+\mu e. \end{aligned}$$

A linearização é básica e o sistema linearizado *(N)* possui uma única solução, conforme Mizuno, Todd e Ye [19].

O teorema fundamental de pontos interiores é o seguinte resultado conforme Gonzaga [10].

Teorema 4.1 *Dados* $(x,s,\mu)\in V(\alpha)$ *tal que* $\delta(x,s,\mu)=\alpha<1$, *seja* (x^+,s^+) *o resultado de um passo de Newton de (x, s). Então,*

a)

$$\delta(x^+,s^+,\mu) \leq \frac{\alpha^2}{\sqrt{8}(1-\alpha)}; \quad e$$

b) *para* $\alpha \leq 0{,}7, \quad (x^+,s^+) \in F^0$.

A seguir, vejamos as idéias básicas de alguns algoritmos de pontos interiores.

Passos Curtos: O algoritmo de trajetória central de passos curtos (veja Gonzaga [11]), apesar de ineficiente na prática, caracteriza bem a idéia dos métodos de pontos interiores. Cada iteração parte de um ponto interior viável na vizinhança da trajetória central (usualmente definida com $\alpha = 0{,}5$) e, então, calcula $\mu > 0$ reduzindo seu valor com relação à iteração anterior e se aproxima do novo ponto central $(x(\mu),s(\mu))$ através de um passo de Newton.

Preditor-Corretor: Este algoritmo trabalha com duas vizinhanças da trajetória central (veja Mizuno, Todd e Ye [18]). Cada iteração parte de um ponto interior viável na primeira vizinhança e, então, calcula uma direção gulosa, no sentido de tentar atingir diretamente uma solução ótima, e um tamanho de passo (veja Ye, Güler, Tapia e Zhang [28]) limitado pela segunda vizinhança obtendo um novo ponto: este é o passo preditor. Em seguida, após atualizar o parâmetro μ, o algoritmo tenta atingir o novo ponto central obtendo, através de um passo de Newton, um novo ponto na primeira vizinhança: este é o passo corretor.

Ponto-Interior-Inviável

Os algoritmos de ponto-interior-inviável primas-duais são algoritmos com pontos interiores tal que o ponto inicial não é restrito a ser um ponto viável. Da mesma forma que os algoritmos

de pontos interiores, para se obter uma solução ótima é necessário um procedimento de purificação e, para a análise de complexidade, supõe-se que os dados iniciais são inteiros. Aqui também supomos que F^0 é não vazio.

A seguir, vejamos as idéias básicas de alguns algoritmos de ponto-interior-inviável.

Preditor-Corretor: O algoritmo preditor-corretor de ponto-interior-inviável (veja, por exemplo, Menezes [17]) tem a mesma idéia do algoritmo preditor-corretor de pontos interiores. A diferença é que neste algoritmo caminhamos próximos a superfície de centros, conforme Mizuno, Todd e Ye [19], ao invés da trajetória central. Além disso, busca-se viabilidade e otimalidade em cada iteração, e não apenas otimalidade, como se faz nos algoritmos de pontos interiores.

Homogêneo e Auto-Dual: Ye, Todd e Mizuno [29] apresentam o problema de PL artificial homogêneo e auto-dual relacionado com o problema *(PD)*, a saber: dados $x^0 \in R^n_{++}$, $s^0 \in R^n_{++}$ e $y^0 \in R^m$,

$$
\begin{aligned}
(HLP) \quad \text{minimizar} \quad & ((x^0)^T s^0 + 1)\theta \\
\text{sujeito a:} \quad & Ax - b\tau + r_P^0\theta = 0 \\
& -A^T y + c\tau - r_D^0\theta \geq 0 \\
& b^T y - c^T x + \bar{z}\theta \geq 0 \\
& -(r_P^0)^T y + (r_D^0)^T x - \bar{z}\tau = -(x^0)^T s^0 - 1 \\
& x \geq 0, \quad \tau \geq 0,
\end{aligned}
$$

onde $r_P^0 = b - Ax^0$, $r_D^0 = c - A^T y^0 - s^0$ e $\bar{z} = c^T x^0 + 1 - b^T y^0$.

Aqui, a homogeneidade do problema significa que todo o lado direito das restrições é igual a zero, exceto para uma delas, freqüentemente chamada de restrição de normalização. Por outro lado, a auto-dualidade do problema significa que o dual é

equivalente ao primal, uma vez que os coeficientes do lado esquerdo das restrições em *(HLP)* formam uma matriz anti-simétrica. Denotamos

$$s = -A^T y + c\tau - r_D^0\theta \quad \text{e} \quad \kappa = b^T y - c^T x + \bar{z}\theta.$$

Consideremos G o espaço nulo da matriz dos coeficientes das restrições do PPL homogêneo e auto-dual no formato padrão. Enunciaremos, a seguir, o algoritmo homogêneo e auto-dual. Aqui, optamos pela idéia preditor-corretor; com o parâmetro μ sendo atualizado no final do passo corretor, uma vez que $\gamma = 1$ neste passo.

Algoritmo 4.2 *Homogêneo e Auto-Dual.*

Dados:

$\varepsilon := 2^{-L}$, $\beta := 0{,}25$, $\mu^0 := 1$ *e*

$(y^0, x^0, \tau^0, \theta^0, s^0, \kappa^0) := (0, e, 1, 1, e, 1) \in V(\beta)$.

$k := 0$.

REPITA

$y := y^k$, $x := x^k$, $\tau := \tau^k$, $\theta := \theta^k$, $s := s^k$, $\kappa := \kappa^k$, $\mu := \mu^k$.

Passo preditor:

Calcular $(dy, dx, d\tau, d\theta, ds, d\kappa) \in G$ *tal que*

$$\begin{pmatrix} xds + sdx \\ \tau d\kappa + \kappa d\tau \end{pmatrix} = \gamma\mu e - \begin{pmatrix} xs \\ \tau\kappa \end{pmatrix},$$

para $\gamma = 0.$

Obter o tamanho do passo

$$\bar{\lambda} := \max\{\lambda; (y + \lambda dy, x + \lambda dx, \tau + \lambda d\tau, \theta + \lambda d\theta, s + \lambda ds, \kappa + \lambda d\kappa) \in V(2\beta)\}.$$

Obter o resultado do passo preditor

$$(\bar{y}, \bar{x}, \bar{\tau}, \bar{\theta}, \bar{s}, \bar{\kappa}) := (y + \bar{\lambda} dy, x + \bar{\lambda} dx, \tau + \bar{\lambda} d\tau, \theta + \bar{\lambda} d\theta, s + \bar{\lambda} ds, \kappa + \bar{\lambda} d\kappa).$$

Passo corretor:

Calcular $(\bar{d}y, \bar{d}x, \bar{d}\tau, \bar{d}\theta, \bar{d}s, \bar{d}\kappa) \in G$ *tal que*

$$\begin{pmatrix} \bar{x}\bar{d}s + \bar{s}\bar{d}x \\ \bar{\tau}\bar{d}\kappa + \bar{\kappa}\bar{d}\tau \end{pmatrix} = \gamma\mu e - \begin{pmatrix} \bar{x}\bar{s} \\ \bar{\tau}\bar{\kappa} \end{pmatrix},$$

para $\gamma = 1.$

Obter o resultado do passo corretor

$$(y^{k+1}, x^{k+1}, \tau^{k+1}, \theta^{k+1}, s^{k+1}, \kappa^{k+1}) := (\bar{y} + \bar{d}y, \bar{x} + \bar{d}x, \bar{\tau} + \bar{d}\tau, \bar{\theta} + \bar{d}\theta, \bar{s} + \bar{d}s, \bar{\kappa} + \bar{d}\kappa).$$

Atualizar o parâmetro μ

$$\mu^{k+1} := \frac{(x^{k+1})^T s^{k+1} + \tau^{k+1}\kappa^{k+1}}{n+1}.$$

Atualizar o contador de iterações

$$k := k + 1.$$

ATÉ QUE $\mu^k < \varepsilon.$

A seguir apresentaremos os dois primeiros resultados: resolução de um problema prático pequeno usando os algoritmos de ponto-interior-inviável e o software LINDO, com uma análise comparativa em número de iterações; e implementações dos algoritmos primais-duais simplex tabular e pontos-interiores-

inviáveis com uma análise comparativa em número de iterações e tempo de execução.

Implementações

As implementações que apresentaremos nesta seção foram desenvolvidas em MATLAB, versões 4.0 e 6.0. A implementação do algoritmo de ponto-interior-inviável preditor-corretor para o LabPL se refere ao trabalho de Bueno e Menezes [5]. Utilizamos um computador pessoal com processador Duron 997MHz e memória RAM de 120MB.

Considere o exemplo de Klee e Minty apresentado na seção 3. Veremos a resolução de $(KM)_2$ através de nossas implementações dos algoritmos preditor-corretor de pontos interiores e homogêneo e auto-dual de ponto-interior-inviável, respectivamente. Estas implementações resolvem o PPL primal-dual, porém, mostraremos apenas os pontos primais gerados pelos algoritmos.

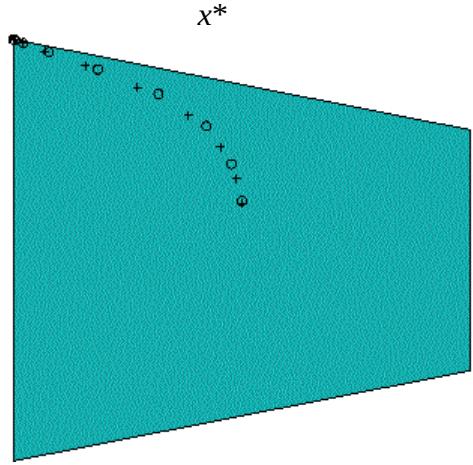

Figura 2: Algoritmo preditor-corretor para $(KM)_2$.

Observemos a Figura 2, que se refere à implementação do algoritmo preditor-corretor para $(KM)_2$. Em 'o' estão representados os pontos preditores e em '+' os pontos corretores

gerados pelo algoritmo. A última solução (primal) obtida pelo algoritmo é aproximadamente o ponto extremo 'x^*', que é uma solução ótima para $(KM)_2$.

Este algoritmo trabalha com duas vizinhanças da trajetória central e funciona bem na prática. Uma observação é que, enquanto o algoritmo preditor-corretor levou 10 iterações para resolver $(KM)_2$, o algoritmo de passos curtos levou 187 iterações, ambos partindo do mesmo ponto.

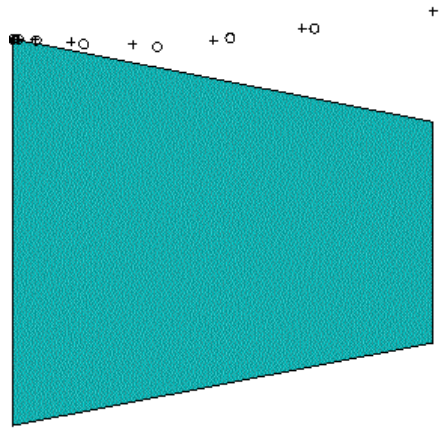

Figura 3: Algoritmo homogêneo e auto-dual para $(KM)_2$.

Observemos, agora, a Figura 3, que se refere à implementação do algoritmo homogêneo e auto-dual para $(KM)_2$. Em 'o' estão representados os pontos preditores e em '+' os pontos corretores gerados pelo algoritmo. A última solução (primal) obtida pelo algoritmo é aproximadamente o ponto extremo 'x^*', que é uma solução ótima para $(KM)_2$.

Este algoritmo também funciona bem na prática e não necessita de um ponto viável inicial para a resolução dos problemas de PL. Além disso, ele utiliza um PPL artificial para obter uma solução para o PPL *(P)*.

Comparações. Faremos, a seguir, uma análise comparativa simples, no sentido de que ela se refere a apenas dois exemplos, apesar de podermos, para um deles (Klee e Minty), aumentar o número de variáveis $(n = 2,\ldots,15)$ e, para o outro (problema agrícola), tomar um problema prático disponível.

Primeiramente, apresentaremos uma comparação, em número de iterações, entre uma implementação com o software LINDO (veja por exemplo, Moré e Wright [20]), que utiliza o algoritmo simplex e o método de conjunto ativo, e nossas implementações dos algoritmos de ponto-interior-inviável para a resolução dos problemas *(PA)* e *(PA) simplificado*, apresentados na seção 3.

Tabela 1: Um Problema Agrícola para PL

	Software LINDO	***Algoritmo* Preditor-Corretor***	***Algoritmo* Homogêneo e Auto-Dual***
PPL	iterações	iterações	iterações
(PA)	24	53	25
(PA) simplificado	11	30	22

*Precisão: 10e-06.

Na Tabela 1, podemos observar que a implementação utilizando o software LINDO é superior às demais em ambos os problemas. Todavia, a superioridade é reduzida quando consideramos o problema *(PA)*, o qual possui mais variáveis para o formato padrão e possui degeneração.

É importante lembrar que o software LINDO é uma ferramenta muito utilizada por profissionais de vários países e, portanto, é um produto bastante testado e qualificado. Além disso, não aplicamos o procedimento de purificação nas soluções obtidas pelos algoritmos de ponto-interior-inviável, que foram soluções aproximadas à solução ótima obtida pelo LINDO. Ainda, para as nossas implementações, não primamos pelo uso de Análise Numérica (veja, por exemplo, [6] e [25]).

A seguir, vejamos na Tabela 2 uma comparação, em número de iterações e tempo de execução, referente às nossas implementações dos algoritmos primais-duais simplex e de ponto-

interior-inviável para a resolução de $(KM)_P$, também apresentado na seção 3.

Tabela 2: Número de Iterações e Tempo de Execução (em segundos)

nº de variáveis	*Algoritmo Simplex*		*Algoritmo* Preditor-Corretor*		*Algoritmo* Homogêneo e Auto-Dual*	
n	iterações	tempo	iterações	tempo	iterações	tempo
2	8	0.0032	11	0.0199	9	0.0163
3	12	0.0050	15	0.0362	12	0.0348
4	16	0.0073	19	0.0615	15	0.0437
5	20	0.0095	22	0.0973	17	0.0840
6	24	0.0118	25	0.1309	19	0.1076
7	28	0.0151	28	0.1826	20	0.1468
8	32	0.0180	31	0.2462	20	0.2008
9	36	0.0214	34	0.3821	20	0.2340
10	40	0.0286	37	0.4817	20	0.2828
11	44	0.0355	40	0.7254	20	0.3979
12	48	0.0387	43	0.8911	20	0.4578
13	53	0.0394	45	1.3058	21	0.5713
14	57	0.0504	48	1.5957	21	0.6359
15	61	0.0507	51	1.9677	21	0.7319

*Precisão: 10e-06.

Para a obtenção do tempo de execução, rodamos cada problema $(n = 2,\ldots,15)$ 250 vezes, de 50 em 50 vezes, onde, em cada vez, calculamos, através da função *cputime* do MATLAB, o tempo em que o problema ficou em execução na CPU. Em cada uma dessas 50 vezes, calculamos a média e, então, obtivemos a média das 5 médias.

Na Tabela 2, podemos observar que a partir de $n = 4$ o algoritmo homogêneo e auto-dual supera os demais em número de iterações, enquanto que o algoritmo preditor-corretor supera o algoritmo simplex a partir de $n = 8$. Em tempo de execução, o algoritmo simplex supera os algoritmos de ponto-interior-inviável. Lembramos que o algoritmo homogêneo e auto-dual tem complexidade, em iterações, $O(\sqrt{n}L)$ e o algoritmo preditor-corretor $O(nL)$. Também, uma desvantagem do algoritmo

simplex primal-dual em relação aos algoritmos de ponto-interior-inviável, é que ele necessita de um ponto dual viável para resolver o PPL *(P)*.

CONSIDERAÇÕES FINAIS

Neste ponto podemos concluir sobre os terceiro e quarto resultados, isto é, viabilização do processo ensino-aprendizagem, em que desenvolvemos os enunciados dos algoritmos como facilitador para aqueles interessados na disciplina PL e, também, com a utilização da internet; e na contribuição para a criação do LabPL, através do estudo e implementação de alguns algoritmos das famílias simplex e pontos interiores.

Através da comparação que fizemos entre os algoritmos simplex e de ponto-interior-inviável para instâncias do exemplo de Klee e Minty, foi possível observar que no simplex primal-dual gastamos mais iterações e menos tempo de execução do que nos algoritmos de ponto-interior-inviável. Acreditamos que isto se deve ao fato de que o número de iterações do simplex está relacionado com o número combinatorial de soluções básicas viáveis e, nos algoritmos de ponto-interior-inviável resolvemos em cada iteração um sistema com ordem $2n+m$. Pensamos que trabalhos explorando comparações computacionais mais aprofundadas em torno destas duas famílias de algoritmos sejam interessantes para o desenvolvimento, por exemplo, de algoritmos híbridos, conforme Bixby, Gregory, Lustig, Marsten e Shanno [2].

Acreditamos que a visualização gráfica para resoluções de problemas de PL facilita o entendimento da idéia do método utilizado. Por outro lado, a internet é uma ferramenta de longo alcance e, também, é de fácil acesso atualmente. Deste modo, pensando ainda na contribuição para o processo ensino-aprendizagem, sugerimos implementações de algoritmos de PL na internet que possibilitem, em cada iteração da resolução dos problemas, a visualização gráfica pelo usuário.

Referências

BEALE, E. M. L. Cycling in the dual simplex algorithm. *Naval Research Logistics Quarterly*, 1955. p. 269-275.

BIXBY, R. E. et al. Very large-scale linear programming: a case study in combining interior point and simplex methods. *Operations Research*, v. 40, n. 5, p. 885-897, 1992.

BLAND, R. G. New finite pivoting rules for simplex method. *Mathematics of Operations Research*, v. 2, n. 2, p. 103-107, 1977.

BREGALDA, P. F.; DE OLIVEIRA, A. A. F.; BORNSTEIN, C. T. *Introdução à Programação Linear*. 3. ed. Campus, 1988.

BUENO, E. F.; MENEZES, M. A. F. Implementação de Algoritmos das Famílias Simplex, Elipsóides e Pontos Interiores para Programação Linear. *Revista Eletrônica de Iniciação Científica da SBC*, v. 3, n. 3, 2003.

CAMPOS FILHO, F. F. *Algoritmos Numéricos*. LTC, 2001.

DANTZIG, G. B. Maximization of a linear function of variables subject to linear inequalities. *Activity Analysis of Production and Allocation*, (Ed.). KOOPMANS, T. C. New York, John Wiley, p. 339-347, 1951.

DANTZIG, G. B.; Ford, L. R.; FULKERSON, D. R. A primal-dual algorithm for linear programs. In: KUHN, H. W.; TUCKER, A. W. (Ed.). *Linear Inequalities and Related Systems*, Princeton: Princeton University Press, p. 171-181, 1956.

GONZAGA, C. C. *Algoritmos de Pontos Interiores para Programação Linear*. 17º COLÓQUIO BRASILEIRO DE MATEMÁTICA (Minicurso), IMPA/CNPq, 1989.

GONZAGA, C. C. Path-Following Methods for Linear Programming. *SIAM Review*, vol. 34, n. 2, 167-224, 1992.

GONZAGA, C. C. On the complexity of linear programming. *Resenhas IME-USP*, vol. 2, n. 2, 197-207, 1995.

KARMARKAR, N. A new polynomial time algorithm for linear programming. *Combinatorica*, p. 373-395, 1984.

KHACHIYAN, L. G. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady,* v. 20, 191-194, 1979.

KHACHIYAN, L. G. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics,* v. 20, p. 53-72, 1980.

KLEE, V.; MINTY, G. How good is the simplex algorithm? *Inequalities III* . In: SISHA, O. (Ed.). New York: Academic Press, 1972.

LEMKE, C. E. The dual method for solving the linear programming problem. *Naval Research Logistic Quarterly*, 978-981, 1954.

MENEZES, M. A. F. *Um algoritmo de ponto-interior-inviável com complexidade* $O(\sqrt{n}L)$ *iterações para programação linear*. Tese (Doutorado) – COPPE/UFRJ, defendida sob a orientação do professor Clóvis C. Gonzaga em 1998.

MIZUNO, S.; TODD, M. J.; YE, Y. On Adaptive-step primal-dual interior-point algorithms for linear programming. *Mathematics of Operations Research*, v. 18, n. 4, 1993.

MIZUNO, S.; TODD, M. J.; YE, Y. A surface of analytic centers and primal-dual infeasible-interior-point algorithms for linear programming. *Mathematics of Operations Research*, v. 20, n. 1, 1995.

MORÉ, J. J.; WRIGHT, S. J. *Optimization software guide. Society for Industrial and Applied Mathematics*. Philadelphia, 1993.

MÜLLER, V. M. *Um modelo de programação linear para um problema agrícola*. Trabalho (Conclusão de Curso - Graduação em Ciência da Computação) – Universidade Católica de Goiás. Orientador: Marco Antonio Figueiredo Menezes, 2004.

PINTO, L. L. *Estudo e implementação de algoritmos simplex e de pontos interiores para programação linear*. Trabalho (Conclusão de Curso – Graduação em Ciência da Computação) - Universidade Católica de Goiás. Orientador: Marco Antonio Figueiredo Menezes, 2004.

PINTO, L. L.; MENEZES, M. A. F. *Algoritmos simplex tabular para o LabPL*. Anais do XXVII CNMAC, Porto Alegre, 2004. p. 304-304.

PINTO, L. L.; MENEZES, M. A. F. *Uma implementação do algoritmo simplex tabular primal, dual e primal-dual para programação linear*. Anais do XXVI CNMAC, São José do Rio Preto, 2003. p. 359-359.

RUGGIERO, M. A. G.;e LOPES, V. L. da R. *Cálculo Numérico: aspectos teóricos e computacionais*. 2. ed. MAKRON Books, 1996.

SAIGAL, R. *Linear Programming - A Modern Integrated Analysis*. Kluwer Academic Publishers, 1995.

YE, Y. *Interior Point Algorithms: theory and analysis*. Wiley-Interscience Series in Discrete Mathematics and Optimization. New York: John Wiley, 1997.

YE, Y.; GÜLER, O.; TAPIA, R. A.; ZHANG, Y. A quadratically convergent $O(\sqrt{n}L)$-iteration algorithm for linear programming. *Mathematical Programming*, v. 59, p. 151-162, 1993.

YE, Y.; TODD, M. J.; MIZUNO, S. An $O(\sqrt{n}L)$ - Iteration Homogeneous and Self-Dual Linear Programming Algorithm. *Mathematics of Operations Research*, v. 19, n. 1, 1994.

Site: <http://www.ucg.br/Institutos/LabPL/PaginaPrincipal.html>.

Abstract: *the objective of this work is to study and to implement the simplex, interior point, infeasible-interior-point families' algorithms for Linear Programming, contributing with the development process of the Laboratory of Linear Programming (LabPL - http://www.ucg.br/Institutos/LabPL/Index.html).*

Key words: *Linear Programming, Simplex Algorithms, Interior Point Algorithms, Complexity, Teaching-Learning Process.*

Este trabalho é inspirado em nossa Monografia de Projeto Final de Curso [22], defendida em 18/12/2004 no Departamento de Computação da UCG como parte dos pré-requisitos para obtenção do grau de Bacharel em Ciência da Computação e, também, em [23] e [24].

Agradecimentos
Aos amigos Elivelton, pelas atenciosas correções e sugestões e, ao Vianei, pela contribuição na implementação dos algoritmos simplex na linguagem de programação PHP; à UCG, pelo apoio para a realização do projeto de construção do LabPL, o qual nos proporcionou dois trabalhos de Iniciação Científica; e, à OVG (Organização das Voluntárias de Goiás), pela bolsa concedida durante nossos trabalhos de Iniciação Científica e Projeto Final de Curso.

LEIZER DE LIMA PINTO
Universidade Federal do Rio de Janeiro – UFRJ. Coordenação dos Programas de Pós-Graduação em Engenharias (COPPE). *E-mail*: *leizer@cos.ufrj.br*

MARCO ANTONIO FIGUEIREDO MENEZES
Universidade Católica de Goiás – UCG. Centro Técnico-Científico – Departamento de Computação. *E-mail*: marco@ucg.br