versão impressa ISSN 0101-7438 / versão online ISSN 1678-5142

MÉTODOS DO TIPO DUAL SIMPLEX PARA PROBLEMAS DE OTIMIZAÇÃO LINEAR CANALIZADOS

Ricardo Silveira Sousa
Carla Taviane Lucke da Silva
Marcos Nereu Arenales *
Departamento de Matemática Aplicada e Estatística
Inst. de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)
São Carlos – SP
rsouza@icmc.usp.br
carla@icmc.usp.br
arenales@icmc.usp.br

* *Corresponding author* / autor para quem as correspondências devem ser encaminhadas

Recebido em 03/2004; aceito em 07/2005 após 1 revisão
Received March 2004; accepted July 2005 after one revision

Resumo

Neste artigo estudamos o problema de otimização linear canalizado (restrições e variáveis canalizadas, chamado formato geral) e desenvolvemos métodos do tipo dual simplex explorando o problema dual, o qual é linear por partes, num certo sentido não-linear. Várias alternativas de busca unidimensional foram examinadas. Experimentos computacionais revelam que a busca unidimensional exata na direção dual simplex apresenta melhor desempenho.

Palavras-chave: otimização linear; otimização linear por partes; método dual simplex.

Abstract

In this paper we study the linear optimization problem lower and upper constrained (i.e., there are lower and upper bounds on constraints and variables) and develop dual simplex methods that explore the dual problem, which is piecewise linear, in some sense nonlinear. Different one-dimensional searches were examined. Computational experiments showed that the exact one-dimensional search in the dual simplex direction has the best performance.

Keywords: linear optimization; linear piecewise optimization; simplex dual method.

1. Introdução

O clássico problema de otimização linear (formato padrão):

$$\begin{aligned} &\text{Minimizar } f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \\ &\text{Sujeito a: } \mathbf{Ax} = \mathbf{b} \\ &\qquad\qquad \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{1.1}$$

em que $\mathbf{A}$ é uma matriz $m \times n$, foi formalizado por George B. Dantzig em 1947 que, em seguida, desenvolveu também o *método primal simplex* para resolvê-lo. Desde então, um número grande de pesquisadores têm contribuído para o campo da otimização linear de diferentes maneiras, seja incluindo desenvolvimentos teóricos e computacionais, seja encontrando novas aplicações práticas.

Logo após a publicação do método primal simplex de Dantzig (1951), sua versão dual surgiu com Lemke (1954), que consiste numa especialização do método primal simplex para o problema dual:

$$\begin{aligned} &\text{Maximizar } \varphi(\boldsymbol{\lambda}) = \mathbf{b}^T\boldsymbol{\lambda} \\ &\text{Sujeito a: } \mathbf{A}^T\boldsymbol{\lambda} \leq \mathbf{c}. \end{aligned} \tag{1.2}$$

Segundo Maros (2003), o método dual simplex tem atraído considerável interesse, devido à importante aplicação nos métodos de otimização linear inteiro misto, os quais resolvem uma seqüência de problemas de otimização linear, com característica de que uma solução básica dual factível de boa qualidade é sempre disponível para o problema seguinte da seqüência. Segundo Bixby (2001), testes computacionais mostram que o desempenho do método dual simplex pode ser superior ao método primal simplex.

É importante observar que a evolução das implementações computacionais dos resolvedores lineares teve um papel fundamental no progresso da Otimização Linear. Um exemplo disto é o pacote CPLEX, que vem sendo melhorado desde sua primeira versão lançada em 1988. Bixby (2001) relatou que a implementação do método dual simplex, na primeira versão do pacote CPLEX 1.0 rodado numa estação de trabalho UltraSparc de 300 MHz, resolvia um problema de 16223 restrições e 28568 variáveis, com 88340 elementos não nulos em 1217.4 segundos e, o mesmo método na última versão CPLEX 7.1 na mesma máquina, resolvia o mesmo problema em 22.6 segundos. Ou seja, um mesmo método pode se tornar 50 vezes mais rápido dependendo de sua implementação. Vale observar que esta relação de desempenho também foi obtida por Karmarkar (1984), que afirmou que o método de pontos interiores era 50 vezes mais rápido do que o método simplex. Obviamente, estruturas de dados adequadas são fundamentais para um bom desempenho de uma implementação computacional. A despeito disto, um mesmo método pode permitir um número grande de variantes, com desempenhos bastante diversos, como por exemplo, a regra de Dantizg normalizada (conhecida na literatura de língua inglesa por *'steepest edge rule'*. Veja Forrest & Goldfarb, 1992).

O objetivo principal deste trabalho consiste em desenvolver métodos tipo dual simplex para um formato geral de problemas de otimização linear (assim chamado em Vanderbei, 1997, o qual chamamos problemas canalizados) explorando o problema dual linear por partes (em certo sentido, não-linear), com buscas unidimensionais, exatas e inexatas. Em Vanderbei (1997), um método dual simplex para esta classe de problemas foi apresentado, baseado em tableau-simplex, em que se explorasse a natureza linear por partes do problema dual. Em

trabalhos futuros examinaremos o efeito da regra de Dantizg normalizada e certas estruturas de dados, para problemas esparsos de médio e grande porte.

A maioria dos problemas práticos de otimização linear incluem diversos tipos de restrições (limitantes inferiores e/ou superiores, igualdades, variáveis livres, etc.). Para resolver esses problemas, necessita-se de uma versão do algoritmo dual simplex que possa tratar todos os tipos de restrições eficientemente (Maros, 2003). Este trabalho, apresenta o algoritmo dual simplex especializado em resolver problemas canalizados, baseado na característica da função dual simplex, que é linear por partes. A principal característica deste método é que em uma iteração ele pode fazer um progresso equivalente a muitas iterações do dual simplex padrão.

O artigo está organizado da seguinte forma: Na seção 2 relatamos sobre a eficiência e a complexidade computacional do método simplex, na seção 3 é descrito o algoritmo dual simplex linear por partes (Arenales, 1984), e na seção 4 descrevemos algumas modificações para busca unidimensional utilizada no algoritmo. A seção 5 apresenta os primeiros experimentos computacionais. Finalmente na seção 6, temos as conclusões.

2. A Eficiência e a Complexidade Computacional do Método Simplex

A cada iteração do algoritmo simplex, a tarefa mais importante consiste na resolução dos sistemas básicos, que pode ser feito em $O(m^3)$ operações elementares. Assim, o esforço necessário para resolver (1.1), pelo algoritmo simplex pode ser medido pelo número de iterações.

A questão da eficiência do método simplex sempre foi tema de pesquisa desde a sua publicação. No início, a convergência finita do método, garantida com regras que evitassem ciclos (repetições indefinidas de uma solução básica), era suficiente para os pesquisadores, mesmo que o número máximo de iterações pudesse ser muito grande. Um conjunto de relatos (a maioria informal e não publicado) sobre sua eficiência computacional para se resolver problemas práticos ou gerados de forma aleatória formou o chamado *folclore* do método simplex, o qual afirma que, na prática, o número de iterações requeridas é um polinômio de grau baixo em m (número de restrições). Veja Shamir (1987) para uma revisão deste folclore. Sousa (2000) realizou um estudo computacional para o problema de corte de peças em estoque, que envolve milhares de variáveis, e observou que o número de iterações é da ordem de m^2, fornecendo uma classe de problemas práticos que reforça o folclore simplex.

A experiência computacional com um número grande de problemas resolvidos em vários anos (Dantzig, 1963; Murty, 1983; Shamir, 1987; Bazaraa *et al*., 1992) indicou que o número médio de iterações é uma função linear de m, que parece ser menor do que $3m$.

No entanto, Klee & Minty (1972) apresentaram um exemplar, para o qual o método simplex com o critério de Dantzig para escolha da variável a entrar na base (i.e., menor custo relativo) necessita de $2^n - 1$ iterações (onde n é o número de variáveis), percorrendo todos os vértices da região factível do problema, a qual é definida como uma distorção do hipercubo m-dimensional e tem 2^n vértices. Assim, o algoritmo simplex não pertence à classe de algoritmos com tempo polinomial. Variações do método simplex, mais tarde, também se mostraram ineficientes, do ponto de vista do estudo do pior caso, necessitando um número exponencial de iterações (Bertsimas & Tsitsiklis, 1997).

Portanto, a questão contínua aberta quanto à possibilidade da construção de um método do tipo simplex que seja polinomial, ou a prova definitiva de que é impossível construir um algoritmo do tipo simplex com complexidade polinomial. Segundo Shamir (1987), Terlaky & Zhang (1993) está é a questão aberta mais desafiante na teoria da otimização linear.

Um resultado interessante foi provado recentemente por Fukuda & Terlaky (1998). Eles provaram que partindo de qualquer base, existe uma seqüência pequena de pivôs contendo no máximo *n* passos, levando a base ótima. Este resultado indica que algoritmos polinomial podem existir.

3. O Problema de Otimização Linear com Restrições Canalizadas

A classe de problemas de otimização linear com restrições canalizadas é de grande interesse prático, pois representa vários problemas reais, tais como problemas de mistura, planejamento, etc., cujas restrições, ou parte delas, são definidas por limitantes tanto superiores como inferiores, decorrentes de tolerâncias de especificações técnicas, demanda, etc.

3.1 O problema primal

O problema de otimização linear com restrições canalizadas (ou, formato geral, conforme Vanderbei, 1997) é definido por:

$$\begin{aligned} &\text{Minimizar } f(\mathbf{x}) = \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ &\text{Sujeito a: } \mathbf{d} \le \mathbf{A}\mathbf{x} \le \mathbf{e}, \end{aligned} \qquad (3.1)$$

em que $\mathbf{A} \in R^{m\times n}$; $\mathbf{d}, \mathbf{e} \in R^m$ com $d_i \le e_i$ $i=1,...,m$; $\mathbf{c}, \mathbf{x} \in R^n$.

Suporemos, que posto $(\mathbf{A}) = n$, isto é, suas colunas formam um conjunto de vetores de $\boldsymbol{R}^m$ linearmente independente. A dependência linear indicaria casos triviais de inexistência de soluções ótimas ($f(\mathbf{x}) \to -\infty$, admitindo-se que o problema seja factível), ou que o problema independe da variável cuja coluna é combinação linear das demais. Para ver isto, suponha que a *n-ésima* coluna seja combinação linear das demais: $\boldsymbol{a_n} = \sum_{j=1}^{n-1} \gamma_j a_j$. Considere a matriz elementar

$$E = \begin{pmatrix} 1 & & & -\gamma_1 \\ & \ddots & & \vdots \\ & & 1 & -\gamma_{n-1} \\ & & & 1 \end{pmatrix}.$$

Note que a matriz **AE** tem as *n-1* primeiras colunas iguais à matriz **A** e a última coluna nula. Assim, considerando a identidade: $\mathbf{Ax} = (\mathbf{AE})(\mathbf{E}^{-1}\mathbf{x})$ e, com a mudança de variável $\mathbf{y}=\mathbf{E}^{-1}\mathbf{x}$, as restrições em (3.1) independem de y_n. Com esta mudança de variável, a função objetivo $f(\mathbf{x}) = \mathbf{c}^{\mathrm{T}}\mathbf{x} = (\mathbf{c}^{\mathrm{T}}\mathbf{E})\mathbf{y}$ tem a coordenada de y_n dada por: $-\sum_{j=1}^{n-1} \gamma_j c_j + c_n$, que se for nula, então y_n pode assumir qualquer valor em toda solução ótima, ou se for não nula, então $f(\mathbf{x}) \to -\infty$. Como $x_n = y_n$, podemos concluir que o problema ou não tem solução ótima, ou independe da variável x_n.

Observe que o problema (3.1) contempla restrições que tipicamente ocorrem na prática, como por exemplo a condição de não negatividade ou a canalização das variáveis.. Consideramos, sem perda de generalidade, que as variáveis sejam canalizadas, isto é, existam restrições do tipo: $d_i \le x_i \le e_i$, de modo que a matriz **A** contém uma matriz identidade $n \times n$ e, portanto, posto(**A**)=n. Observe também que restrições de igualdade são representadas no formato geral (3.1) por: $d_i = e_i$.

O problema (3.1) pode ainda ser rescrito no formato padrão (com variáveis canalizadas), por se definir $\mathbf{y} = \mathbf{Ax}$, então (3.1) é equivalente a:

$$\begin{aligned} &\text{Minimizar} \quad f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \\ &\text{Sujeito a:} \quad \mathbf{Ax} - \mathbf{y} = \mathbf{0} \\ &\qquad\qquad \mathbf{d} \le \mathbf{y} \le \mathbf{e}. \end{aligned} \tag{3.2}$$

O formato equivalente (3.2) será empregado na construção do problema dual lagrangiano.

3.2 O problema dual

Podemos determinar o problema dual de (3.1) usando o problema equivalente (3.2). Para isto, definimos a função lagrangiana:

$$\begin{aligned} &L(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}) = \mathbf{c}^T\mathbf{x} + \boldsymbol{\lambda}^T(\mathbf{y} - \mathbf{Ax}) \\ &L(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}) = (\mathbf{c}^T - \boldsymbol{\lambda}^T\mathbf{A})\mathbf{x} + \boldsymbol{\lambda}^T\mathbf{y}, \qquad \boldsymbol{\lambda} \in R^m. \end{aligned} \tag{3.3}$$

Considere o problema langrangiano:

$$h(\boldsymbol{\lambda}) = \min_{(x,y)} L(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}), \quad \mathbf{d} \le \mathbf{y} \le \mathbf{e}. \tag{3.4}$$

Segue diretamente da definição de (3.4) a clássica desigualdade: $f(\mathbf{x}) \ge h(\boldsymbol{\lambda})$, para todo **x** factível e para todo $\boldsymbol{\lambda} \in R^m$. Ou seja, $h(\boldsymbol{\lambda})$ fornece um limitante inferior para $f(\mathbf{x})$. Esta desigualdade motiva para o chamado *problema dual lagrangiano*, ou simplesmente *problema dual*, que consiste em determinar o melhor limitante inferior:

$$\text{Maximizar } h(\boldsymbol{\lambda}), \ \boldsymbol{\lambda} \in R^m.$$

Note que $\boldsymbol{\lambda}$ qualquer, como definido até agora, pode levar a limitantes inferiores triviais e sem interesse, isto é: $h(\boldsymbol{\lambda}) = -\infty$. Assim, restringimos $\boldsymbol{\lambda}$ tal que o mínimo em (3.4) exista. Como **x** é irrestrito de sinal e **y** é canalizado, analisando (3.3), devemos escolher $\boldsymbol{\lambda}$ tal que:

$$\mathbf{c}^T - \boldsymbol{\lambda}^T\mathbf{A} = \mathbf{0} \quad \text{ou} \quad \mathbf{A}^T\boldsymbol{\lambda} = \mathbf{c}. \tag{3.5}$$

Com $\boldsymbol{\lambda}$ satisfazendo o sistema de equações lineares em (3.5), é possível expressar $h(\boldsymbol{\lambda})$ por se resolver o problema de minimização em (3.4). Usando a restrição (3.5) em (3.3), temos:

$$L(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}) = \boldsymbol{\lambda}^T\mathbf{y} = \sum_{i=1}^{m} \lambda_i y_i.$$

Então,

$$h(\boldsymbol{\lambda}) = \min L(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}) = \sum_{i=1}^{m} \min_{d_i \le y_i \le e_i} \lambda_i y_i = \sum_{i/\lambda_i > 0} \lambda_i d_i + \sum_{i/\lambda_i < 0} \lambda_i e_i.$$

A soma de cada mínimo na expressão acima é válida, pois as variáveis y_i são independentes. Se $\lambda_i = 0$, então y_i pode assumir qualquer valor no intervalo $[d_i, e_i]$.

Assim, podemos explicitar a função objetivo dual como sendo:

$$h(\boldsymbol{\lambda}) = \sum_{i=1}^{m} h_i(\lambda_i)$$

em que

$$h_i(\lambda_i) = \begin{cases} e_i\lambda_i & \text{se } \lambda_i \leq 0 \\ d_i\lambda_i & \text{se } \lambda_i \geq 0. \end{cases}$$

Ou seja, o problema dual é um problema de otimização onde a função objetivo é côncava linear por partes ilustrada na Figura 1.

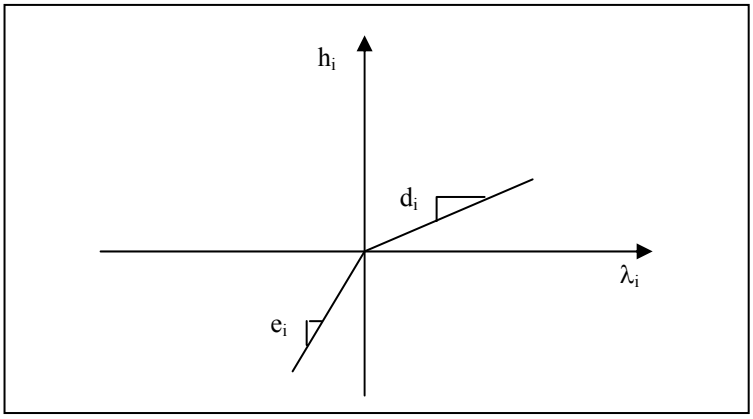

Figura 1 – Função objetivo dual linear por partes.

Desta forma, tendo resolvido o problema de minimização que surge na definição da função dual (3.4), podemos explicitar o problema dual:

$$\begin{aligned} &\text{Maximizar } h(\boldsymbol{\lambda}) = \sum_{i=1}^{m} h_i(\lambda_i) \\ &\text{Sujeito a: } \mathbf{A}^T\boldsymbol{\lambda} = \mathbf{c}. \end{aligned} \tag{3.6}$$

3.3 A estratégia dual simplex

Note que o problema dual sempre será factível, pois posto (**A**) = *n* e $\boldsymbol{\lambda}$ é irrestrito de sinal. Portanto, se o problema dual tiver solução ótima, então existirá uma solução básica ótima.

A definição de uma solução básica para um problema de otimização linear por partes, com a função objetivo separável como em (3.6), é uma simples extensão de definição de solução básica da otimização linear, bastando que as variáveis não básicas sejam fixadas em pontos de não-diferenciação (Cavichia & Arenales, 2000). No caso particular da função dual em (3.6), cada função h_i é não diferenciável apenas em λ_i=0. Observe também que, enquanto

uma solução básica factível é um vértice da região das soluções factíveis de um problema de otimização linear, uma solução básica de um problema de otimização linear por partes pode pertencer ao interior da região das soluções factíveis, de modo que a solução ótima pode ser um ponto interior. Em particular, a região das soluções factíveis de (3.6) consiste num subespaço afim de R^m (subespaço transladado) e, portanto, não tem vértices.

Para a construção de uma solução básica para o problema (3.6), considere uma partição básica qualquer nas colunas de $\mathbf{A}^T$: $\mathbf{A}^T = (\mathbf{B}^T, \mathbf{N}^T)$ (*i.e.*, selecione uma sub-matriz inversível $n \times n$ de $\mathbf{A}^T$, denotada por $\mathbf{B}^T$, e denote por $\mathbf{N}^T$ a sub-matriz $n \times (m-n)$ formada pelas colunas restantes de $\mathbf{A}^T$).

Esta partição nas colunas de $\mathbf{A}^T$ introduz uma partição no vetor de variáveis: $\boldsymbol{\lambda}=(\boldsymbol{\lambda}_B, \boldsymbol{\lambda}_N)$. O *i-ésimo* elemento de $\boldsymbol{\lambda}_B$ é denotado por λ_{B_i} (ou, também: λ_i, $i \in B$). A *i-ésima* coluna de $\mathbf{B}^T$ é denotada por $\mathbf{a}_{B_i}^T$ e a *k-ésima* coluna de $\mathbf{N}^T$ é denotada por $\mathbf{a}_{N_k}^T$.

Assim, podemos escrever a solução geral do sistema de equações em (3.5):

$$\mathbf{A}^T\boldsymbol{\lambda} = \mathbf{c} \Leftrightarrow \mathbf{B}^T\boldsymbol{\lambda}_B + \mathbf{N}^T\boldsymbol{\lambda}_N = \mathbf{c} \Leftrightarrow \boldsymbol{\lambda}_B = \left(\mathbf{B}^T\right)^{-1}\mathbf{c} - \left(\mathbf{B}^T\right)^{-1}\mathbf{N}^T\boldsymbol{\lambda}_N. \qquad (3.7)$$

Uma solução particular, chamada solução básica dual, associada à partição básica é dada por:

$$\hat{\boldsymbol{\lambda}} = \begin{bmatrix} \hat{\boldsymbol{\lambda}}_B \\ \hat{\boldsymbol{\lambda}}_N \end{bmatrix}, \text{ onde } \hat{\boldsymbol{\lambda}}_B^T = \mathbf{c}^T\mathbf{B}^{-1} \text{ e } \hat{\boldsymbol{\lambda}}_N^T = 0. \qquad (3.8)$$

Note que a solução definida em (3.8), independentemente da partição básica, satisfaz as restrições do problema dual e, portanto, é chamada *dual factível*.

Para a solução básica dual, $\hat{\lambda}_{B_i}$ pode ser positivo ou negativo (se nulo, é o caso de degeneração dual) e, portanto, para a avaliação da função dual $h(\boldsymbol{\lambda})$ o valor de y_{B_i} fica determinado por:

$$\hat{y}_{B_i} = \begin{cases} d_{B_i} & \text{se } \hat{\lambda}_{B_i} \geq 0 \\ e_{B_i} & \text{se } \hat{\lambda}_{B_i} \leq 0. \end{cases} \qquad (3.9)$$

Em caso de degeneração da solução dual, *i.e.*, $\hat{\lambda}_{B_i} = 0$, y_{B_i} pode, a princípio, assumir qualquer valor no intervalo $[d_{B_i}\ e_{B_i}]$ que, por razões práticas, escolheremos $\hat{y}_{B_i} = d_{B_i}$ ou $\hat{y}_{B_i} = e_{B_i}$. Deste modo, se escolhermos $\hat{y}_{B_i} = d_{B_i}$, então para efeito de desenvolvimentos futuros, a variável λ_{B_i} é vista como positiva (apesar de ter seu valor nulo). Assim, se a variável λ_{B_i} sofre um acréscimo, $\hat{y}_{B_i}$ não se altera conforme (3.8); caso contrário, um decréscimo em λ_{B_i} tornaria a variável negativa e, portanto, $\hat{y}_{B_i}$ deveria ser e_{B_i}.

Observe também que a partição básica fornece uma partição nas equações de (3.2):

$$\begin{bmatrix} \mathbf{B} \\ \mathbf{N} \end{bmatrix} x = \begin{bmatrix} \mathbf{y}_B \\ \mathbf{y}_N \end{bmatrix} \Leftrightarrow \begin{cases} \mathbf{B}\mathbf{x} = \mathbf{y}_B \\ \mathbf{N}\mathbf{x} = \mathbf{y}_N. \end{cases}$$

Da primeira equação acima, com $\mathbf{y}_B = \hat{\mathbf{y}}_B$, define-se a *solução básica primal*

$$\hat{\mathbf{x}} = \mathbf{B}^{-1}\hat{\mathbf{y}}_B. \tag{3.10}$$

Além disso, a segunda equação define o valor de $\mathbf{y}_N$, de modo que as restrições (exceto as canalizações) do problema primal estão satisfeitas:

$$\hat{\mathbf{y}}_N = \mathbf{N}\mathbf{B}^{-1}\hat{\mathbf{y}}_B. \tag{3.11}$$

Note que a solução $(\hat{\mathbf{x}},\hat{\mathbf{y}})$ dada por (3.9)-(3.11) satisfaz as restrições **Ax=y** e $\mathbf{d}_B \le \mathbf{y}_B \le \mathbf{e}_B$, enquanto que as restrições $\mathbf{d}_N \le \mathbf{y}_N \le \mathbf{e}_N$ podem estar violadas.

Definição 3.3.1: Considere uma partição básica sobre as colunas de $\mathbf{A}^T$. As soluções em (3.8) e (3.9)-(3.11) são chamadas soluções básicas primal dual e primal, respectivamente, associadas à partição básica. Se $\mathbf{d}_N \le \hat{\mathbf{y}}_N \le \mathbf{e}_N$, então a solução básica primal é factível, ou simplesmente, primal factível.

Observe que, para o par de soluções básicas, dual e primal, associadas a uma partição básica, temos: $h(\hat{\boldsymbol{\lambda}}) = f(\hat{\mathbf{x}})$, pois:

$$h(\hat{\boldsymbol{\lambda}}) = \sum_{i=1}^{m} h_i(\hat{\lambda}_i) = \sum_{i\in B} h_i(\hat{\lambda}_i) + \sum_{i\in N} h_i(\hat{\lambda}_i) = \sum_{i\in B} h_i(\hat{\lambda}_i) = \hat{\boldsymbol{\lambda}}_B^T \mathbf{y}_B = \mathbf{c}^T\mathbf{B}^{-1}\hat{\mathbf{y}}_B = \mathbf{c}^T\hat{\mathbf{x}} = f(\hat{\mathbf{x}}).$$

Como $h(\hat{\boldsymbol{\lambda}})$ é um limitante inferior para f(**x**), segue o teorema.

Teorema 3.3.2: Considere uma partição básica qualquer e as soluções básicas primal e dual associadas. Se a solução for factível (i,e., $\mathbf{d}_N \le \hat{\mathbf{y}}_N \le \mathbf{e}_N$) então as soluções básicas primal e dual são soluções ótimas dos problemas primal e dual respectivamente.

Em geral, para a resolução de um problema de otimização linear qualquer usando o método simplex, se faz necessário a fase I, que consiste em outro problema de otimização linear a ser resolvido. Entretanto, o método simplex aplicado ao problema dual (3.6) não requer a fase I e pode ser iniciado a partir de qualquer partição básica, a qual fornece sempre uma solução dual factível, conforme (3.8).

Como é comum em problemas práticos, restrições do tipo: $\mathbf{l} \le \mathbf{x} \le \mathbf{u}$ (implicitamente consideradas em $\mathbf{d} \le \mathbf{A}\mathbf{x} \le \mathbf{e}$), uma partição óbvia é disponível com **B=I**. Caso não haja limitantes naturais para uma variável x_i, pode-se considerar l_i = - M, ou u_i = M, onde M é um valor suficientemente grande, e a partição básica inicial segue válida.

Considere uma partição básica e suas soluções básicas associadas (3.8) e (3.9)-(3.11). Se as condições do teorema 3.3.2 não são satisfeitas, perturbamos a solução básica dual, de modo a aumentar a função objetivo dual $h(\boldsymbol{\lambda})$.

A *estratégia dual simplex* consiste na perturbação de uma variável dual não básica (uma componente de $\boldsymbol{\lambda}_N$). Temos dois casos a considerar. Por simplicidade, supomos a solução dual não degenerada, caso contrário, $\hat{\lambda}_{B_i} = 0$ para algum $i \in B$.

Caso a) A variável λ_{N_K} é acrescida: $\begin{cases} \lambda_{N_K} = \delta, \text{ com } \delta \geq 0 \text{ e pequeno} \\ \lambda_{N_i} = 0, \quad i = 1,...,m-n,\ i \neq k. \end{cases}$

Ou seja:

$$\boldsymbol{\lambda}_N = \delta \tilde{\mathbf{e}}_k , \tag{3.12}$$

em que $\tilde{\mathbf{e}}_k$ é a *k*-ésima coluna da matriz identidade (*m*-*n*)×(*m*-*n*).

Com esta perturbação nas variáveis não básicas, devemos determinar a perturbação nas variáveis básicas de modo que o sistema $\mathbf{A}^T\boldsymbol{\lambda}=\mathbf{c}$ seja satisfeito. Para isto, substituindo (3.12) em (3.7) segue:

$$\boldsymbol{\lambda}_B = \left(\mathbf{B}^T\right)^{-1}\mathbf{c} - \delta\left(\mathbf{B}^T\right)^{-1}\mathbf{a}_{N_K}^T .$$

De (3.8), temos que:

$$\boldsymbol{\lambda}_B = \hat{\boldsymbol{\lambda}}_B + \delta\,\boldsymbol{\eta}_B, \tag{3.13}$$

em que $\boldsymbol{\eta}_B = -\left(\mathbf{B}^T\right)^{-1}\mathbf{a}_{N_K}^T$ é o vetor das componentes básicas da direção dual simplex.

Portanto, a nova solução dual pode ser escrita por:

$$\boldsymbol{\lambda} = \begin{pmatrix} \boldsymbol{\lambda}_B \\ \boldsymbol{\lambda}_N \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{\lambda}}_B \\ \hat{\boldsymbol{\lambda}}_N \end{pmatrix} + \delta \begin{pmatrix} -\left(\mathbf{B}^T\right)^{-1}\mathbf{a}_{N_K}^T \\ \tilde{\mathbf{e}}_k \end{pmatrix},$$

ou ainda,

$$\boldsymbol{\lambda} = \hat{\boldsymbol{\lambda}} + \delta\,\boldsymbol{\eta}, \tag{3.14}$$

em que $\boldsymbol{\eta} = \begin{pmatrix} -\left(\mathbf{B}^T\right)^{-1}\mathbf{a}_{N_K}^T \\ \tilde{\mathbf{e}}_k \end{pmatrix}$ é chamada *direção dual simplex*.

Escrevendo a função objetivo dual para a nova solução (3.14), considerando δ suficientemente pequeno, de modo que a função $h(\boldsymbol{\lambda})$ é linear, temos:

$$h(\boldsymbol{\lambda}) = h(\hat{\boldsymbol{\lambda}} + \delta\ \boldsymbol{\eta}) = \sum_{i=1}^{n} h_{B_i}(\hat{\lambda}_{B_i} + \delta\eta_{B_i}) + \sum_{i=1}^{m-n} h_{N_i}(\hat{\lambda}_{N_i} + \delta\eta_{N_i}).$$

Como supomos a solução básica não degenerada, *i.e.*, $\hat{\lambda}_{B_i} \neq 0, i = 1,...,n$ e lembrando que apenas a *k-ésima* variável não básica foi perturbada: $\lambda_{N_k} = \delta \geq 0$ (portanto o coeficiente de λ_{N_k} na função dual é d_{N_k}, veja Figura 1), segue:

$$h(\boldsymbol{\lambda}) = h(\hat{\boldsymbol{\lambda}} + \delta\,\boldsymbol{\eta}) = \sum_{i=1}^{n} \hat{y}_{B_i}(\hat{\lambda}_{B_i} + \delta\eta_{B_i}) + \delta d_{N_k} = \sum_{i=1}^{n} \hat{y}_{B_i}\hat{\lambda}_{B_i} + \delta\left(\sum_{i=1}^{n} \hat{y}_{B_i}\eta_{B_i} + d_{N_k}\right)$$

$$h(\boldsymbol{\lambda}) = h(\hat{\boldsymbol{\lambda}}) + \delta\left(\hat{\mathbf{y}}_B^T\boldsymbol{\eta}_B + d_{N_k}\right),$$

Substituindo na expressão acima as componentes básicas da direção dual simplex dadas em (3.13) e a solução básica primal em (3.10), vem que:

$$\begin{aligned} h(\boldsymbol{\lambda}) &= h(\hat{\boldsymbol{\lambda}}) + \delta\left(\hat{\mathbf{y}}_B^T(-\mathbf{B}^T)^{-1}\mathbf{a}_{N_K} + d_{N_k}\right) \\ h(\boldsymbol{\lambda}) &= h(\hat{\boldsymbol{\lambda}}) + \delta\left(-\mathbf{a}_{N_K}^T\hat{\mathbf{x}} + d_{N_k}\right). \end{aligned} \tag{3.15}$$

Em resumo, a perturbação da solução básica dual na direção dual simplex dada por (3.14), decorrente da estratégia dual simplex (3.12), promove uma variação da função objetivo dual dada por (3.15).

Portanto, se $\mathbf{d}_{N_K} - \mathbf{a}_{N_k}^T\hat{\mathbf{x}} > 0$, ou seja, se o limite inferior da *k*-ésima restrição primal for violado, a direção dual simplex é uma direção de subida.

A expressão (3.15) é valida, desde que λ_B em (3.13) não sofra mudança de sinal, pois caso contrário, o coeficiente da função objetivo dual é alterado. Se $\hat{\lambda}_{B_i}$ e η_{B_i} têm sinais opostos, então λ_{B_i} pode trocar de sinal com o crescimento de δ. Portanto, temos dois casos a considerar:

Caso (*i*) Se $\hat{\lambda}_{B_i} < 0$ e $\eta_{B_i} > 0$, então impondo que: $\hat{\lambda}_{B_i} + \delta\eta_{B_i} \le 0,$ segue que: $\delta \le -\dfrac{\hat{\lambda}_{B_i}}{\eta_{B_i}}$;

Caso (*ii*) Se $\hat{\lambda}_{B_i} > 0$ e $\eta_{B_i} < 0$, então impondo que: $\hat{\lambda}_{B_i} + \delta\eta_{B_i} \ge 0,$ segue que: $\delta \le -\dfrac{\hat{\lambda}_{B_i}}{\eta_{B_i}}$.

Assim, para que não haja mudança de sinal em λ_{B_i} devemos ter:

$$\delta \le \delta_i = -\frac{\hat{\lambda}_{B_i}}{\eta_{B_i}}, \text{ para todo } i=1,\ldots,n, \text{ tal que } \hat{\lambda}_{B_i} \text{ e } \eta_{B_i} \text{ tenham sinais opostos.}$$

Seja p_o um índice básico tal que:

$$\delta_{p_0} = -\frac{\hat{\lambda}_{B_{p0}}}{\eta_{Bp_0}} = \min\left\{-\frac{\hat{\lambda}_{B_i}}{\eta_{Bi}} \text{ tal que } -\frac{\hat{\lambda}_{B_i}}{\eta_{Bi}} \ge 0 \;\; i=1,\ldots,n\right\}. \tag{3.16}$$

Para $0 \le \delta \le \delta_{P_0}$ temos que: $h(\boldsymbol{\lambda}) = h(\hat{\boldsymbol{\lambda}}) + \delta\left(d_{N_K} - \hat{y}_{N_k}\right)$. Quando $\delta = \delta_{P_0}$ temos de (3.13) e (3.16) que p_0-ésima restrição básica torna-se nula: $\lambda_{B_{p_0}} = 0$, o que sugere uma redefinição da partição básica, em que $\lambda_{B_{p0}}$ torna-se não básica e $\lambda_{N_K} = \delta_{P_0} > 0$ torna-se básica. A Figura 2 ilustra a situação.

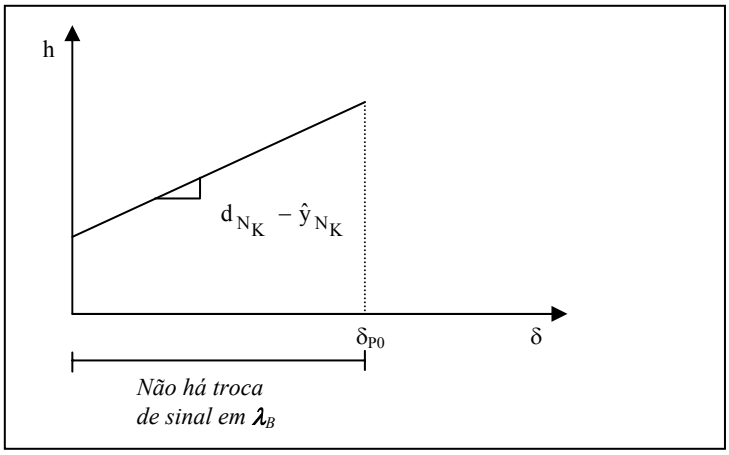

Figura 2 – Gráfico da função dual no intervalo [0, δ_{P0}].

Obviamente, como as variáveis duais são irrestritas de sinal, o crescimento de δ não precisa ser limitado por $h(\hat{\boldsymbol{\lambda}})$. Para valores de $\delta > \delta_{P0}$, a expressão da função dual se altera, mas a direção dual pode ainda ser de subida. Isto sugere uma busca na direção $\boldsymbol{\eta}$ para a escolha de δ. Esta busca é linear por partes, devido à característica da função objetivo dual.

Considere que: $h(\boldsymbol{\lambda}) = h(\hat{\boldsymbol{\lambda}}) + \delta h_0$, onde $h_0 = \left(d_{N_K} - \hat{y}_{N_K}\right)$.

Se dermos um pequeno acréscimo ao valor de δ_{P_0}, ou seja, $\delta = \delta_{P_0} + \varepsilon$, onde ε>0 e pequeno, então $\lambda_{B_{P0}}$ troca de sinal. (Observe que $\lambda_{B_{P0}} = 0$ para $\delta = \delta_{P_0}$).

Analisaremos somente o caso (*i*): $\hat{\lambda}_{B_{P_0}} < 0$ e $\eta_{B_i} > 0$, o outro caso ($\hat{\lambda}_{B_{P_0}} > 0$ e $\eta_{B_i} < 0$) segue de maneira análoga. Então, note que para $0 < \delta < \delta_{P0}$ a variável $\lambda_{B_{P_0}} < 0$ e que para $\delta > \delta_{P_0}$ a variável $\lambda_{B_{P_0}} > 0$. Substituindo $\delta = \delta_{P_0} + \varepsilon$ em (3.14) temos:

$$\boldsymbol{\lambda} = \hat{\boldsymbol{\lambda}} + (\delta_{P_0} + \varepsilon)\boldsymbol{\eta}$$
$$\boldsymbol{\lambda} = (\hat{\boldsymbol{\lambda}} + \delta_{P_0}\boldsymbol{\eta}) + \varepsilon\boldsymbol{\eta}$$
$$\boldsymbol{\lambda} = \boldsymbol{\lambda}^1 + \varepsilon\boldsymbol{\eta},$$

em que $\boldsymbol{\lambda}^1 = \hat{\boldsymbol{\lambda}} + \delta_{P_0}\boldsymbol{\eta}$. Observe que $\lambda_{N_K} = \delta_{p_0} + \varepsilon$ e $\lambda_{N_i} = 0 \quad i \neq k$.

Assim, a função objetivo dual pode ser expressa por:

$$h(\boldsymbol{\lambda}) = \sum_{i=1}^{n} h_{B_i}(\hat{\lambda}_{B_i}) + \sum_{i=1}^{m-n} h_{N_i}(\hat{\lambda}_{N_i})$$
$$h(\boldsymbol{\lambda}) = \sum_{i=1}^{n} h_{B_i}(\lambda^1_{B_i} + \varepsilon\eta_{B_i}) + d_{N_k}(\delta_{p_0} + \varepsilon).$$

Suporemos por simplicidade que $\lambda^1_{B_i} \neq 0$, $i \neq p_0$, ou seja, que o mínimo em (3.16) ocorre somente para o índice p_0 e apenas a variável $\lambda_{B_{p_0}}$ se anula quando $\delta = \delta_{p_0}$.

Lembrando que $\lambda^1_{B_{p_0}} = \hat{\lambda}_{B_{p_0}} + \delta_{p0}\eta_{B_{p_0}} = 0$ e $\varepsilon > 0$ e pequeno, temos que:

$$h(\boldsymbol{\lambda}) = \sum_{\substack{i=1 \\ i \neq p_0}}^{n} \hat{y}_{B_i}(\hat{\lambda}^1_{B_i} + \varepsilon\eta_{B_i}) + h_{B_{p_0}}(\varepsilon\eta_{B_{p_0}}) + (\delta_{p_0} + \varepsilon)d_{N_K}.$$

Como $\eta_{B_{p0}} > 0$, pois estamos analisando o caso (i), temos que $h_{B_{p_0}}(\varepsilon\eta_{B_{p_0}}) = \varepsilon d_{B_{p_0}}\eta_{B_{p_0}}$. Logo,

$$h(\boldsymbol{\lambda}) = \sum_{i=1}^{n} \hat{y}_{B_i}(\lambda^1_{B_i}) + \delta_{p_0} d_{N_K} + \varepsilon\left(\sum_{\substack{i=1 \\ i \neq p_0}}^{n} y_{B_i}\eta_{B_i} + d_{B_{p_0}}\eta_{B_{p_0}} + d_{N_k}\right) \qquad (3.17)$$

As seguintes igualdades são asseguradas:

I) $h(\boldsymbol{\lambda}^1) = h(\hat{\boldsymbol{\lambda}} + \delta_{p_0}\boldsymbol{\eta}) = \sum_{i=1}^{n} h_{B_i}(\hat{\lambda}_{B_i} + \delta_{p_0}\eta_{B_i}) + \sum_{i=1}^{m-n} h_{N_i}(\hat{\lambda}_{N_i} + \delta_{p_0}\eta_{N_i})$

$h(\boldsymbol{\lambda}^1) = \sum_{i=1}^{n} h_{B_i}(\lambda^1_{B_i}) + \delta_{p_0} d_{N_{Ki}}$;

II) $\sum_{i=1}^{n} \hat{y}_{B_i}\eta_{B_i} = \sum_{i=1}^{n} \hat{y}_{B_i}\eta_{B_i} + \hat{y}_{B_{p0}}\eta_{B_{p_0}} - \hat{y}_{B_{p_0}}\eta_{B_{p_0}}$

$\sum_{\substack{i=1 \\ i \neq p_0}}^{n} \hat{y}_{B_i}\eta_{B_i} = \sum_{i=1}^{n} \hat{y}_{B_i}\eta_{B_i} - e_{B_{p0}}\eta_{B_{p_0}}$;

III) $\sum_{i=1}^{n} \hat{y}_{B_i}\eta_{B_i} = \hat{\mathbf{y}}_B^T\boldsymbol{\eta}_B = \hat{\mathbf{y}}_B^T\left(-\left(\mathbf{B}^T\right)^{-1}\mathbf{a}_K\right) = -\mathbf{a}_{N_K}^T\left(\mathbf{B}^{-1}\hat{\mathbf{y}}_B\right) = -\mathbf{a}_{N_K}^T\hat{\mathbf{x}}$.

Portanto, a expressão (3.17) pode ser reescrita usando (*I*), (*II*) e (*III*):

$$h(\boldsymbol{\lambda}) = h(\boldsymbol{\lambda}^1) + \varepsilon(h_0 - (e_{B_{p_0}} - d_{B_{p_0}})\eta_{B_{p_0}}),$$

em que $h_0 = -\mathbf{a}_{N_K}^T\hat{\mathbf{x}} + d_{N_K}$.

As expressões descritas anteriormente para a função dual são válidas desde que $\varepsilon > 0$ e suficientemente pequeno. Analisaremos agora, qual o maior valor para o incremento ε, ou seja, determinaremos o tamanho do passo de modo que $\hat{\lambda}_{B_i} + (\delta_{p_0} + \varepsilon)\eta_{B_i}$ não mude de sinal.

Note que $\hat{\lambda}_{B_i}$ tem o mesmo sinal de $\hat{\lambda}_{B_i} + (\delta_{p_0} + \varepsilon)\eta_{B_i}$, $i \neq p_0$. Assim, os valores de ε que promovem mudança de sinal em λ_{B_i} são dados por:

$$\varepsilon_i = -\frac{\hat{\lambda}_{B_i} + \delta_{p_0}\eta_{B_i}}{\eta_{B_i}} = -\frac{\hat{\lambda}_{B_i}}{\eta_{B_i}} - \delta_{p_0} > 0,\ i \neq p_0.$$

Chamando $\delta_i = \varepsilon_i + \delta_{p_0} = -\dfrac{\hat{\lambda}_{B_i}}{\eta_{B_i}}$, o valor máximo de δ que evita a segunda mudança de sinal em λ_{B_i} é dado por:

$$\delta_{p_1} = -\frac{\hat{\lambda}_{B_{p_1}}}{\eta_{Bp_1}} = \min\left\{-\frac{\hat{\lambda}_{B_i}}{\eta_{B_i}} > 0, \quad i \neq p_0\right\}.$$

Observe que, no cálculo do índice p_0 (veja (3.16)) as mesmas razões já haviam sido calculadas e p_0 foi o índice que forneceu a menor razão. Agora p_1 é o índice da segunda menor razão. A Figura 3 representa o gráfico da função objetivo dual.

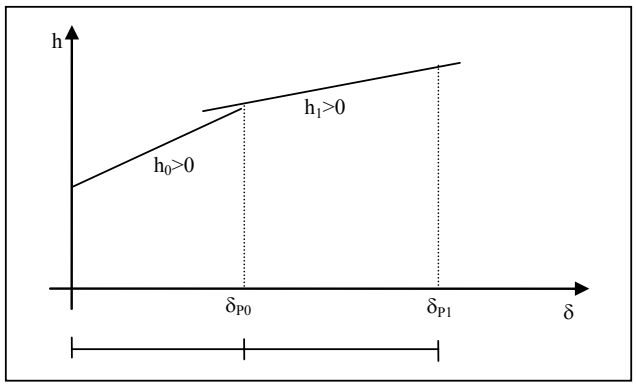

Figura 3 – Gráfico da função dual após a mudança de sinal de uma variável básica.

Podemos examinar o passo dual além de $\delta_{p_1} : \delta > \delta_{p_1}$ de maneira análoga ao que foi feito até aqui. Em geral temos: $h_{q+1} = h_q - (e_{B_{pq}} - d_{B_{pq}})\eta_{B_{pq}}$. A Figura 4 ilustra a função dual e como uma busca na direção dual simplex pode ser feita para encontrar: o max $h(\boldsymbol{\lambda} + \delta\boldsymbol{\eta})$.

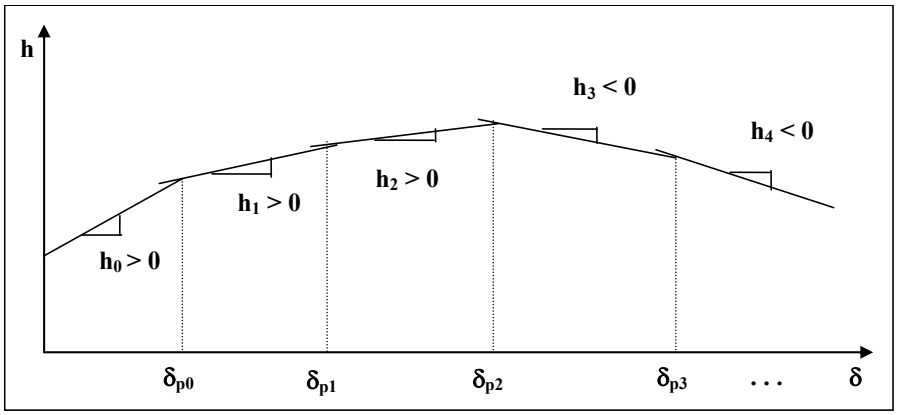

Figura 4 – Gráfico da função dual: δ_{p2} é o ponto de máximo.

A busca unidimensional na direção dual simplex consiste em escolher um índice ℓ tal que:

$$h_\ell > 0 \text{ e } h_{\ell+1} \le 0 .$$

Caso b) A variável λ_{N_K} é decrescida: $\begin{cases} \lambda_{N_K} = -\delta, \text{ com } \delta \ge 0 \text{ e pequeno} \\ \lambda_{N_i} = 0, \quad i = 1,...,m-n, \; i \ne k. \end{cases}$

Ou seja:

$$\boldsymbol{\lambda}_N = -\delta \tilde{\mathbf{e}}_k . \tag{3.18}$$

O procedimento é análogo ao caso a), notando agora que:

$$\boldsymbol{\lambda}_B = \hat{\boldsymbol{\lambda}}_B - \delta \boldsymbol{\eta}_B ,$$

em que $\boldsymbol{\eta}_B = -\left(\mathbf{B}^T\right)^{-1} \mathbf{a}_{N_K}^T$ é o vetor das componentes básicas da direção dual simplex definido anteriormente. Portanto, a nova solução dual pode ser escrita por:

$$\boldsymbol{\lambda} = \begin{pmatrix} \boldsymbol{\lambda}_B \\ \boldsymbol{\lambda}_N \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{\lambda}}_B \\ \hat{\boldsymbol{\lambda}}_N \end{pmatrix} - \delta \begin{pmatrix} -\left(\mathbf{B}^T\right)^{-1} \mathbf{a}_{N_K}^T \\ \tilde{\mathbf{e}}_k \end{pmatrix}$$

ou ainda,

$$\boldsymbol{\lambda} = \hat{\boldsymbol{\lambda}} - \delta \boldsymbol{\eta}. \tag{3.19}$$

Fazendo o mesmo desenvolvimento da função objetivo dual para a nova solução (3.19) e considerando as devidas mudanças, segue que:

$$h(\boldsymbol{\lambda}) = h(\hat{\boldsymbol{\lambda}}) + \delta\left(\mathbf{a}_{N_k}^T \hat{\mathbf{x}} - e_{N_K}\right). \tag{3.20}$$

Portanto, se $\mathbf{a}_{N_k}^T \hat{\mathbf{x}} - e_{N_K} > 0$, ou seja, se o limite superior da *k*-ésima restrição primal for violado, a direção dual simplex é direção de subida.

A expressão (3.20) é válida desde que λ_B não sofra mudança de sinal. Logo se $\hat{\boldsymbol{\lambda}}_B$ e $\boldsymbol{\eta}_B$ têm o mesmo sinal, então λ_B pode trocar de sinal com o crescimento de δ. Portanto, de maneira análoga ao caso a), para que não haja mudança de sinal em λ_{B_i} devemos ter:

$$\delta \le \delta_i = \frac{\hat{\boldsymbol{\lambda}}_{Bi}}{\boldsymbol{\eta}_{Bi}},$$

seja p_o um índice básico tal que:

$$\delta_{p_0} = \frac{\hat{\boldsymbol{\lambda}}_{B_{Po}}}{\boldsymbol{\eta}_{B p_0}} = \min\left\{ \frac{\hat{\boldsymbol{\lambda}}_{B_i}}{\boldsymbol{\eta}_{B_i}} \quad \text{tal que} \quad \frac{\hat{\boldsymbol{\lambda}}_{B_i}}{\boldsymbol{\eta}_{B_i}} < 0 \quad i = 1,...,m \right\}$$

Para $\hat{\mathbf{x}} = \mathbf{B}^{-1}\hat{\mathbf{y}}_B$ temos que: $h(\boldsymbol{\lambda}) = h(\hat{\boldsymbol{\lambda}}) + \delta\left(\hat{\mathbf{y}}_{N_K} - \mathbf{e}_{N_K}\right)$. Quando $\delta = \delta_{P_0}$ temos que a p_0-ésima restrição básica torna-se nula, enquanto que $\lambda_{N_k} = -\delta \leq 0$, o que sugere uma redefinição da partição básica.

Como já observado no caso a), o valor de δ pode ser maior do que δ_{P0} pois as variáveis duais não têm restrição de sinal. Para analisar a função dual para valores além de δ_{P0} consideramos $\delta = \delta_{P_0} + \varepsilon,\ \varepsilon > 0$ e suficientemente pequeno. Logo a nova solução dual é dada por:

$$\boldsymbol{\lambda} = \begin{bmatrix} \hat{\boldsymbol{\lambda}}_B \\ \hat{\boldsymbol{\lambda}}_N \end{bmatrix} - (\delta_{P_0} + \varepsilon)\begin{bmatrix} \boldsymbol{\eta}_B \\ \boldsymbol{\eta}_N \end{bmatrix}.$$

Analogamente ao caso a), obtemos a seguinte expressão para a função dual:

$$h(\boldsymbol{\lambda}) = h(\hat{\boldsymbol{\lambda}} - \delta_{P_0}\boldsymbol{\eta}) + \varepsilon h_1, \quad \text{onde} \quad h_1 = h_0 - (e_{B_{P_0}} - d_{B_{P_0}})\eta_{B_{P_0}} \quad \text{e} \quad h_0 = \mathbf{a}_{N_K}^T\hat{\mathbf{x}} - e_{N_K}.$$

A busca unidimensional é realizada da mesma forma que foi feita para o caso (*a*). Escolhido os índices para as variáveis que entra e sai da base, atualizamos **B** e **N** e repetimos o processo. Esta busca foi considerada na implementação do algoritmo, para a qual fizemos algumas alterações, que serão apresentadas na seção 4.

Embora Maros (2003) tenha desenvolvido o algoritmo dual simplex para problemas com variáveis canalizadas, baseada na característica da função dual (linear por partes), o autor não considera a canalização nas restrições como fizemos, explorando as particularidades deste caso.

3.4 Algoritmo dual simplex com busca linear por partes

Resumimos o método dual simplex com busca unidimensional linear por partes exata especializado para problemas no formato geral dado por (3.1):

Passo 0: ***INICIALIZAÇÃO***

Determine uma partição básica nas colunas de $\mathbf{A}^T = (\mathbf{B}^T, \mathbf{N}^T)$.

Calcule a solução dual: $\hat{\boldsymbol{\lambda}} = \begin{bmatrix} \hat{\boldsymbol{\lambda}}_B \\ \hat{\boldsymbol{\lambda}}_N \end{bmatrix}$ com $\hat{\boldsymbol{\lambda}}_B^T = \mathbf{c}^T\mathbf{B}^{-1}$ e $\hat{\boldsymbol{\lambda}}_N^T = 0$.

Calcule a solução primal: $\hat{\mathbf{x}} = \mathbf{B}^{-1}\hat{\mathbf{y}}_B$, com $\hat{\mathbf{y}}_{B_i} = \begin{cases} d_{B_i} & \text{se } \hat{\boldsymbol{\lambda}}_{B_i} \geq 0 \\ e_{B_i} & \text{se } \hat{\boldsymbol{\lambda}}_{B_i} \leq 0. \end{cases}$

IT **= 0**.

Passo 1: ***OTIMALIDADE***

Escolha um índice k, $1 \leq k \leq m\text{-}n$ que fornece a maior violação, ou seja,

caso a): $d_{N_K} - \mathbf{a}_{N_k}^T \hat{\mathbf{x}} > 0$ *(limite inferior violado),* $h_0 = d_{N_K} - \mathbf{a}_{N_k}^T \hat{\mathbf{x}}$

ou

caso b): $\mathbf{a}_{N_k}^T \hat{\mathbf{x}} - e_{N_K} > 0$ *(limite superior violado),* $h_0 = \mathbf{a}_{N_k}^T \hat{\mathbf{x}} - e_{N_K}$

Se não existe tal índice, isto é, $\mathbf{d}_N \le \mathbf{a}_N^T \hat{\mathbf{x}} \le \mathbf{e}_N$, então **pare**
(a solução atual é ótima obtida na iteração ***IT***.)

Passo 2: ***DIREÇÃO DUAL SIMPLEX***

Determine as componentes básicas da direção dual simplex: $\boldsymbol{\eta}_B = -\left(\mathbf{B}^T\right)^{-1} \mathbf{a}_{N_K}^T$.

Passo 3: ***TAMANHO DO PASSO***

Passo 3.1: ***PONTOS DE NÃO-DIFERENCIAÇÃO***

Encontre δ_i por:

Caso a): $\delta_i = \left\{ -\dfrac{\hat{\lambda}_{B_i}}{\eta_{B_i}} \quad \text{tal que} \quad -\dfrac{\hat{\lambda}_{B_i}}{\eta_{B_i}} \ge 0, \quad i = 1,...,n \right\}$

ou

Caso b): $\delta_i = \left\{ \dfrac{\hat{\lambda}_{B_i}}{\eta_{B_i}} \quad \text{tal que} \quad \dfrac{\hat{\lambda}_{B_i}}{\eta_{B_i}} \ge 0, \quad i = 1,...,n \right\}$

(suponha que são r valores de δ_i).

Se r=0 então **pare**
(ou seja, não existe nenhum delta, o dual não tem solução ótima finita e, portanto, o primal é infactível.)

Senão

Passo 3.2: ***BUSCA UNIDIMENSIONAL***

Ordene o vetor de δ_i de forma a obter: $\delta_{p_0} \le \delta_{p_1} \le ... \le \delta_{p_r}$.

Determine $\ell \le r$ tal que: $h_\ell > 0$ e $h_{\ell+1} \le 0$,

em que $h_0 > 0$ é dado no passo 1.

$h_{\ell+1} = h_\ell - (e_{B_{P\ell}} - d_{B_{P\ell}}) \left| \eta_{B_{p\ell}} \right|$.

faça $\delta = \delta_{p_\ell}$.

Se $h_{r+1} > 0$ então **pare** ($h(\hat{\boldsymbol{\lambda}} + \delta\eta) \to \infty$, isto é, o problema dual não tem solução ótima, logo o primal é infactível.)

Passo 4: **ATUALIZAÇÃO**

Troque o p_ℓ-ésimo índice básico pelo *k*-ésimo índice não básico e atualize as soluções primal e dual $B_{p_\ell} \leftrightarrow N_k$.

IT $\leftarrow$ ***IT+1***, repita o passo 1.

Observe que o *passo 1* do algoritmo, caso a otimalidade não seja encontrada, determina a variável que entra na base (λ_{N_k}) e o *passo 3.2*, determina a variável que sai da base ($\lambda_{B_{p_\ell}}$), caso o problema primal não seja infactível. Note que quando se fixa $\ell = 1$, o método não faz a busca unidimensional e, assim temos o método dual simplex padrão ou usual, que iremos denotar por SB.

A este procedimento de busca unidimensional exata, definido no passo 3.2, denominamos busca completa com ordenação preliminar (BC_1).

4. Algumas Extensões da Busca Unidimensional

Os métodos de busca para resolução de problemas de otimização, por exemplo,

$$\text{Maximizar } \Phi(\mathbf{x})$$
$$\mathbf{x} \in \Omega,$$

têm dois passos fundamentais: a partir de uma solução $\hat{\mathbf{x}} \in \Omega$ *i*) determinação da direção de busca em $\hat{\mathbf{x}}$: **d** (normalmente direção de subida em $\hat{\mathbf{x}}$) e, *ii*) determinação do tamanho do passo: $\hat{\delta}$ tal que $\Phi(\hat{\mathbf{x}} + \hat{\delta}\mathbf{d}) > \Phi(\hat{\mathbf{x}})$. Para a determinação do tamanho do passo, muitas vezes é colocado o sub-problema:

$$\text{Maximizar } \varphi(\delta) = \Phi(\hat{\mathbf{x}} + \delta\mathbf{d}),\ \delta \geq 0.$$

Entretanto, o esforço computacional para a resolução deste sub-problema pode ser grande e desnecessário, pois a direção de busca **d** é importante na solução $\hat{\mathbf{x}}$ e pode ser equivocada longe de $\hat{\mathbf{x}}$, sugerindo que uma nova direção de busca seja determinada. Embora o sub-problema para a função dual linear por partes seja de fácil resolução (passo 3.2 do algoritmo dual simplex com busca unidimensional exata), resta ainda saber se uma busca inexata (i.e., o abandono da direção de busca antes que sub-problema seja resolvido) pode ser efetiva. Investigamos três procedimentos de busca. O primeiro é busca inexata, que é baseado na estrutura linear por partes da função objetivo, somente uma parcela dos pontos de não-diferenciação são examinados, o segundo é um procedimento de busca exata, avaliando os valores de h_i para cada δ_i e o último, busca inexata, adaptamos a clássica regra de Armijo, a qual apresenta um teste para evitar passos 'muito grandes'.

4.1 Primeira modificação no procedimento de busca – Busca Parcial (BP)

Este procedimento é feito realizando uma alteração muito simples no passo 3.2, que faz a ordenação do deltas, no algoritmo da secção 3.4. Neste caso escolheremos uma fração dos deltas ordenados, digamos αn, onde α é um parâmetro ajustável. Assim, determinamos o

índice ℓ (que corresponde ao índice da variável que sai da base) utilizando somente esta fração. O procedimento é dado por :

Passo 3.2.1.: $D=\{\varnothing\}$;
$E=\{1,2,..., r\}$;
Para i=1 até p *(em que p é o maior inteiro menor igual a αn)*;
Determine $\delta_i = \min\{\delta_j,\ j \in E\}$;
Faça $D=D+\{\delta_i\}$ e $E=E-\{i\}$.

Observe que D é o conjunto que ordena somente uma parte do número de deltas. Se para esta ordenação, acontecer do índice ℓ alcançar o máximo αn, isto indica que embora a direção dual simplex seja ainda de subida no ponto $\boldsymbol{\lambda}=\hat{\boldsymbol{\lambda}}+\delta_{p_\ell}\boldsymbol{\eta}$, a busca nesta direção será interrompida.

A esta modificação chamamos busca unidimensional parcial, pois não utiliza todos os deltas.

4.2 Segunda modificação no procedimento de busca – Busca Completa (BC_2)

Este procedimento é feito realizando uma pequena modificação no passo 3.2 do algoritmo, que pode ser descrito pelo seguinte sub-algoritmo:

Passo 0: Determine p_0 tal que: $\delta_{p_0} = \min\{\delta_i,\ i=1,...,r\}$
Seja $h_0 = d_{N_k} - \hat{y}_{N_k}$ ou $\hat{y}_{N_k} - e_{N_k}$ conforme o caso a) ou b),
$\ell = 0$, cont=0 e $E=\{p_0\}$.

Passo 1: Determine $p_{\ell+1}$ tal que: $\delta_{p_{\ell+1}} = \min\{\delta_i,\ i=1,...,\ r,\ i \notin E\}$ e
$h_{\ell+1} = h_\ell - (e_{B_{p_\ell}} - d_{B_{p_\ell}})\,|\,\eta_{B_{p_\ell}}\,|$
cont=cont+1, $E=E+\{p_{cont}\}$

Passo 2: Se $h_{\ell+1} \le 0$ **pare**.
Senão
Se $\ell \le r$-1 então $\ell = \ell+1$ e repita o *passo 1*;
Senão ($h_r > 0$) então **pare**.

(Note que o critério de parada no passo 2 ($h_r > 0$), indica que o problema dual não tem solução ótima e, portanto, o primal é infactível).

Este procedimento faz a busca exata sem que seja necessário a ordenação completa do vetor de pontos de não diferenciação, o que pode ser vantajoso caso ℓ seja muito menor que r.

4.3 Um método de busca unidimensional inexata – Regra de Armijo (AR)

Descreveremos brevemente aqui, a regra de Armijo (Luenberger, 1984) especializada à função objetivo dual linear por partes.

Em essência, a regra de Armijo evita passos muito grandes.

Para que o tamanho do passo não seja muito grande, devemos ter:

$$h(\hat{\boldsymbol{\lambda}} + \delta\boldsymbol{\eta}) \geq h(\hat{\boldsymbol{\lambda}}) + \delta(\varepsilon h_0),$$

em que $\varepsilon \in (0,1)$. Então, enquanto:

$$h(\hat{\boldsymbol{\lambda}} + \delta_{p_i}\boldsymbol{\eta}) \geq h(\hat{\boldsymbol{\lambda}}) + \delta_{p_i}(\varepsilon h_0),$$

a busca continua.

Inicialmente começamos com $\delta=\delta_{p_0}$, depois $\delta=\delta_{p_1}$ e assim sucessivamente até que:

$h_{\ell+1} \leq 0$ (i.e., a busca exata foi concluída), ou

$h(\hat{\boldsymbol{\lambda}} + \delta_{p_{\ell+1}}\boldsymbol{\eta}) \leq h(\hat{\boldsymbol{\lambda}}) + \delta_{p_{\ell+1}}(\varepsilon h_0)$, (i.e., o passo $\delta_{p_{\ell+1}}$ é grande). Adota-se $\delta=\delta_{p_\ell}$.

Note que o menor passo para a regra de Armijo, vai ser sempre $\delta=\delta_{p_0}$. A Figura 5 ilustra a situação.

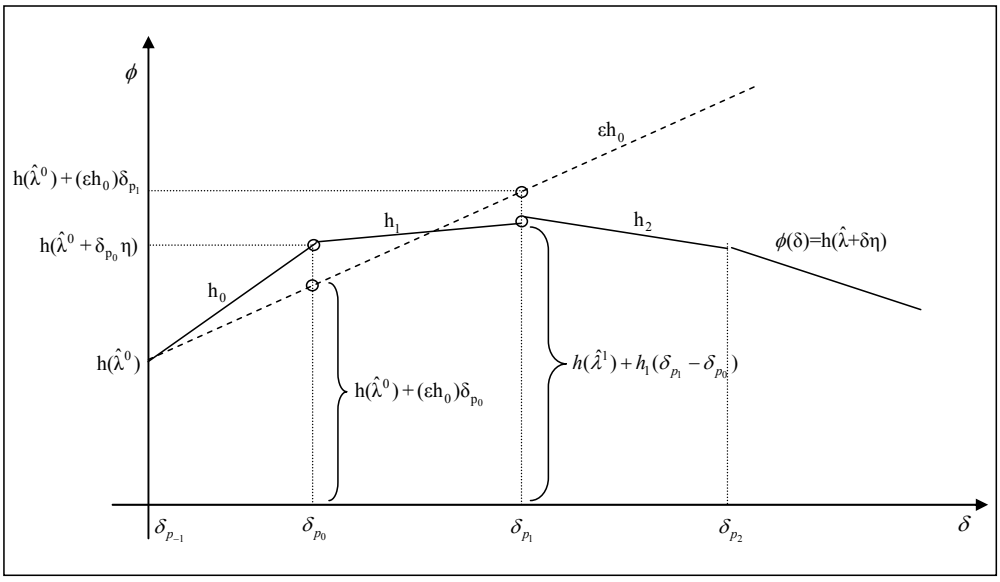

Figura 5 – Regra de Armijo para a função dual linear por partes.

Os resultados para estas buscas unidimensionais são dados na próxima seção.

5. Experimentos Computacionais e Conclusões

Daremos a seguir os resultados computacionais obtidos com o método dual simplex linear por partes com as modificações propostas na seção anterior e para o método dual simplex padrão, quando algumas comparações são realizadas para 3 tipos de estrutura na matriz **A**: densa, escada com 4 blocos e escada com 20 blocos.

O algoritmo dual simplex linear por partes (seção 3.4) foi implementado utilizando a forma produto da inversa para resolver problemas no formato (3.1). Para cada tamanho (dimensão) do problema, que define um exemplar com *m'* restrições e *n* variáveis (veja Figura 7), foram resolvidos 20 exemplares com os dados gerados uniformemente aleatórios nos seguintes intervalos: $c_j \in [-6, 0]$, $a_{ij} \in [-1, 5]$ (para gerar exemplares factíveis, uma solução factível é gerada: $\hat{x}_j \in [0, 10]$), $d_i = \sum_{j=1}^{n} a_{ij}\hat{x}_j - \sigma_i$, $e_i = \sum_{j=1}^{n} a_{ij}\hat{x}_j + \sigma_i$, com $\sigma_i \in [0\ 8]$. Em média, 10% dos valores de σ_i foram gerados nulos, de modo que $d_i = e_i$, isto é, restrições de igualdade (outros parâmetros foram utilizados sem que alterassem as conclusões). Os testes foram realizados em um Notebook Toshiba Satellite Celeron 650 MHz com 312 Mbytes de RAM. A linguagem C de programação foi utilizada, versão 5.01, com estrutura de dados alocada estaticamente.

O tempo (em segundos) e o número de iterações, reportados a seguir, são as médias dos resultados da resolução dos 20 exemplares para cada tamanho do problema. Para a apresentação dos resultados, considere que BC_1 representa o método dual simplex com busca exata (completa) com ordenação preliminar, SB o método dual simplex padrão (usual), BP método dual simplex com busca inexata (parcial), BC_2 o método dual simplex com busca exata com ordenação em processo, AR o método dual simplex com a regra de Armijo (busca inexata) e *m-n* representa o número de restrições e *n* o número de variáveis. Para a regra de Armijo, utilizamos $\varepsilon = 0.2$ (veja seção 4.3) que é um valor usual. Testamos outros valores, mas o desempenho foi pior.

Lembre-se que os valores de δ devem ser ordenados, em ordem crescente, e indicam os pontos de não-diferenciabilidade de $h(\hat{\boldsymbol{\lambda}} + \delta\boldsymbol{\eta})$. A primeira questão que surge, é sobre a grandeza de *r* (comparada com *n*), pois isto pode levar a ordenações de vetores de dimensões elevadas para problemas de grande porte. Como a grandeza de *r* é influenciada pela esparsidade da matriz de restrições? Uma outra questão é sobre a grandeza de ℓ (comparada com *r*). Será necessário ordenar todo o vetor de pontos de não diferenciação? E, por último, compensa a busca unidimensional exata? Isto é, após um avanço na direção dual simplex, não seria melhor mudar de direção? Para responder estas questões, apresentamos os resultados.

5.1 Problemas densos

Nesta seção apresentamos os resultados obtidos para problemas com quase 100% dos elementos das (*m-n*) restrições diferentes de zero. A Tabela 5.1 apresenta os resultados para o método dual simplex linear por partes com busca completa (BC_1) e método dual simplex padrão (SB).

Tabela 5.1 – Busca Completa_1 × Sem Busca – Problemas Densos.

		Iterações		Tempo	
Exemplar	***(m-n)×n***	**BC_1**	**SB**	**BC_1**	**SB**
1	***100×100***	*133.2*	*419.3*	*0.08*	*0.22*
2	***200×100***	*205.0*	*653.7*	*0.19*	*0.52*
3	***300×100***	*235.5*	*785.8*	*0.32*	*1.01*
4	***200×200***	*301.3*	*914.7*	*1.08*	*3.02*
5	***300×200***	*389.6*	*1200.6*	*1.72*	*5.04*
6	***20×400***	*63.5*	*432.7*	*0.59*	*3.65*
7	***100×200***	*175.0*	*642.7*	*0.50*	*1.67*
8	***100×400***	*241.1*	*1150.3*	*2.40*	*10.03*
9	***200×400***	*401.2*	*1618.1*	*4.41*	*17.86*
10	***400×200***	*475.2*	*1481.4*	*2.49*	*7.71*
11	***400×400***	*622.8*	*1980.1*	*8.30*	*25.97*
	Média	***294.8***	***1025.4***	***2.00***	***6.97***

Da Tabela 5.1, observamos que a busca unidimensional completa faz reduzir consideravelmente o número de iterações com relação ao método dual simplex usual, em torno de 3.5. Observe que o tempo por iteração em média foi praticamente o mesmo para os dois métodos.

No exemplar 6, em que o número de variáveis é muito maior do que o número de restrições, observamos a maior diferença no número de iterações. Portanto, testamos mais 4 exemplares com esta característica. Os resultados estão na Tabela 5.2.

Tabela 5.2 – Busca Completa_1 × Sem Busca – Problemas Densos.

		Iterações		Tempo	
Exemplar	***(m-n)×n***	**BC_1**	**SB**	**BC_1**	**SB**
12	***10×400***	*35.2*	*300.8*	*0.30*	*1.98*
6	***20×400***	*63.5*	*432.7*	*0.54*	*3.05*
13	***30×400***	*89.3*	*567.1*	*0.78*	*4.10*
14	***40×400***	*109.1*	*661.8*	*0.96*	*4.95*
15	***50×400***	*132.0*	*790.8*	*1.18*	*6.08*
	Média	***85.8***	***550.6***	***0.75***	***4.03***

Notamos da Tabela 5.2, que a maior diferença foi para o exemplar 12, em que o número de variáveis é 40 vezes o número de restrições. Esta diferença no número de iterações diminui à medida que o número de restrições aumenta.

Apresentamos agora na Tabela 5.3 os resultados obtidos para o método com a busca unidimensional inexata (regra de Armijo – AR) descrito na seção 4.3, comparando com BC_1.

Tabela 5.3 – Busca Completa_1 × Regra de Armijo – Problemas Densos.

		Iterações		Tempo	
Exemplar	***(m-n)×n***	**BC_1**	**AR**	**BC_1**	**AR**
1	***100×100***	*133.2*	*171.2*	*0.08*	*0.09*
2	***200×100***	*205.0*	*285.3*	*0.19*	*0.25*
3	***300×100***	*235.5*	*341.2*	*0.32*	*0.44*
4	***200×200***	*301.3*	*494.9*	*1.08*	*1.64*
5	***300×200***	*389.6*	*751.4*	*1.72*	*3.13*
6	***20×400***	*63.5*	*107.6*	*0.59*	*0.85*
7	***100×200***	*175.0*	*225.8*	*0.50*	*0.57*
8	***100×400***	*241.1*	*342.3*	*2.40*	*3.09*
9	***200×400***	*401.2*	*562.0*	*4.41*	*5.85*
10	***400×200***	*475.2*	*924.5*	*2.49*	*4.72*
11	***400×400***	*622.8*	*1351.6*	*8.30*	*13.2*
	Média	***294.8***	***505.2***	***2.00***	***3.07***

Notamos que a clássica regra de Armijo foi muito inferior a busca completa, fazendo em média 70% mais iterações e gastando aproximadamente 53% mais tempo. No entanto a regra de Armijo apresenta desempenho melhor em relação ao método dual simplex padrão (veja Tabela 5.1), reduzindo, em média, o número de iterações pela metade.

No método dual simplex com busca linear por partes, calculados os deltas, estes precisam ser ordenados. Este procedimento de ordenação pode consumir muito tempo, assim relatamos o tempo de ordenação para o método BC_1, apresentados na Tabela 5.4, em que TT indica o tempo de resolução e TO o tempo de ordenação para todas as iterações.

Tabela 5.4 – Tempo Total × Tempo de Ordenação.

		Tempo	
Exemplar	***(m-n)×n***	**TT**	**TO**
1	***100×100***	*0.08*	*0.00*
2	***200×100***	*0.19*	*0.01*
3	***300×100***	*0.32*	*0.01*
4	***200×200***	*1.08*	*0.06*
5	***300×200***	*1.72*	*0.09*
6	***20×400***	*0.59*	*0.06*
7	***100×200***	*0.50*	*0.05*
8	***100×400***	*2.40*	*0.20*
9	***200×400***	*4.41*	*0.33*
10	***400×200***	*2.49*	*0.14*
11	***400×400***	*8.30*	*0.23*
	Média	***2.00***	***0.10***

Podemos observar que o custo de ordenação não é muito elevado pois, em média, corresponde a 5% do tempo total. Mas para os exemplares 6,7 e 8 esta porcentagem foi de 10%, caso em que o número de variáveis é sensivelmente maior do que o número de restrições. Isto pode ser um indicativo que para problemas maiores o tempo de ordenação pode ser significativo.

Reportamos agora o número de deltas (r) em função de n para as primeiras iterações e o valor de ℓ/r para as duas primeiras iterações, já que para as outras, os valores diminuem suavemente. Na Tabela 5.5 são apresentados os resultados.

Tabela 5.5 – Número de Deltas e Número de Buscas – Problemas Densos.

		r/n	*ℓ/r Iteração*	
Exemplar	***(m-n)×n***		**1ª**	**2ª**
1	***100×100***	*0.86*	*0.36*	*0.18*
2	***200×100***	*0.86*	*0.36*	*0.18*
3	***300×100***	*0.86*	*0.36*	*0.18*
4	***200×200***	*0.86*	*0.36*	*0.18*
5	***300×200***	*0.87*	*0.33*	*0.18*
6	***20×400***	*0.81*	*0.35*	*0.18*
7	***100×200***	*0.88*	*0.34*	*0.17*
8	***100×400***	*0.84*	*0.33*	*0.17*
9	***200×400***	*0.84*	*0.36*	*0.17*
10	***400×200***	*0.87*	*0.59*	*0.58*
11	***400×400***	*0.86*	*0.56*	*0.59*
	Média	***0.85***	***0.39***	***0.25***

Observarmos da Tabela 5.5, que $r \approx 0.85n$. Em nossos experimentos notamos que este valor, tende a $0.5n$ independente do número de variáveis, após as primeiras iterações. Com relação à razão ℓ/r, a média foi a mesma para a primeira e a segunda iteração nos exemplares 1-9 e nos exemplares 10 e 11 os valores foram bem diferentes. Ou seja, para estes problemas (densos) o número de deltas é muito alto, sendo em média 85% do número de variáveis e que para determinar o índice ℓ (pela busca) é apenas de 39% de r e tendem a diminuir com as iterações.

Diante dos resultados apresentados nas Tabelas 5.4 e 5.5, investigamos as modificações abordadas na seção 4. As Tabelas 5.6 e 5.7 fornecem os resultados.

Para a busca parcial (BP), utilizamos o parâmetro α (veja seção 4) sendo 0.35, já que pelos resultados da Tabela 5.5 a busca foi feita utilizando em média $0.39r$ ou $0.33n$.

Tabela 5.6 – Busca Completa_1 × Busca Parcial – Problemas Densos.

		Iterações		Tempo	
Exemplar	***(m-n)×n***	**BC_1**	**BP**	**BC_1**	**BP**
1	***100×100***	*133.2*	*133.2*	*0.08*	*0.07*
2	***200×100***	*205.0*	*205.0*	*0.19*	*0.18*
3	***300×100***	*235.5*	*235.5*	*0.32*	*0.30*
4	***200×200***	*301.3*	*301.3*	*1.08*	*1.03*
5	***300×200***	*389.6*	*389.6*	*1.72*	*1.64*
6	***20×400***	*63.5*	*63.5*	*0.59*	*0.53*
7	***100×200***	*175.0*	*175.0*	*0.50*	*0.45*
8	***100×400***	*241.1*	*241.1*	*2.40*	*2.37*
9	***200×400***	*401.2*	*401.2*	*4.41*	*4.20*
10	***400×200***	*475.2*	*475.2*	*2.49*	*2.39*
11	***400×400***	*622.8*	*630.0*	*8.30*	*9.14*
	Média	***294.8***	***295.5***	***2.00***	***2.02***

Neste caso, a busca parcial (BP) não fez praticamente nenhuma diferença no tempo computacional. No exemplar 11, a busca parcial (inexata) foi pior, pois fez mais iterações.

Fizemos testes para diferentes valores de alfa e não observamos nenhuma melhora significativa, por isso não apresentaremos aqui.

Na Tabela 5.7, reportamos os resultados para a segunda modificação (busca exata) realizada na busca unidimensional, descrita pelo segundo procedimento de busca na seção 4.

Tabela 5.7 – Busca Completa_1× Busca Completa_2.

		Tempo	
Exemplar	***(m-n)×n***	**BC_1**	**BC_2**
1	***100×100***	*0.08*	*0.07*
2	***200×100***	*0.19*	*0.18*
3	***300×100***	*0.32*	*0.29*
4	***200×200***	*1.08*	*0.99*
5	***300×200***	*1.72*	*1.59*
6	***20×400***	*0.59*	*0.48*
7	***100×200***	*0.50*	*0.42*
8	***100×400***	*2.40*	*2.06*
9	***200×400***	*4.41*	*3.92*
10	***400×200***	*2.49*	*2.30*
11	***400×400***	*8.30*	*8.46*
	Média	***2.00***	***1.88***

Da Tabela 5.7, observamos para estes problemas densos, uma suave melhora no tempo computacional, para BC_2 em relação a BC_1, em que conseguimos uma redução no tempo (em média) em torno de 5%. Note que não apresentamos o número de iterações para BC_1, que é o mesmo para BC_2, pois ambas são buscas exatas.

Lembre-se que o que muda da busca completa 1 para a busca completa 2, é o método de ordenação. No 2º caso é o método '*bolha*' que parece ser melhor, pois ℓ é bem menor que r e não precisa ordenar todo o vetor.

A seguir, a Figura 6 apresenta o desempenho computacional (tempo) dos quatro procedimentos de busca e do método dual simplex padrão, para os 11 exemplares de problemas densos que trabalhamos.

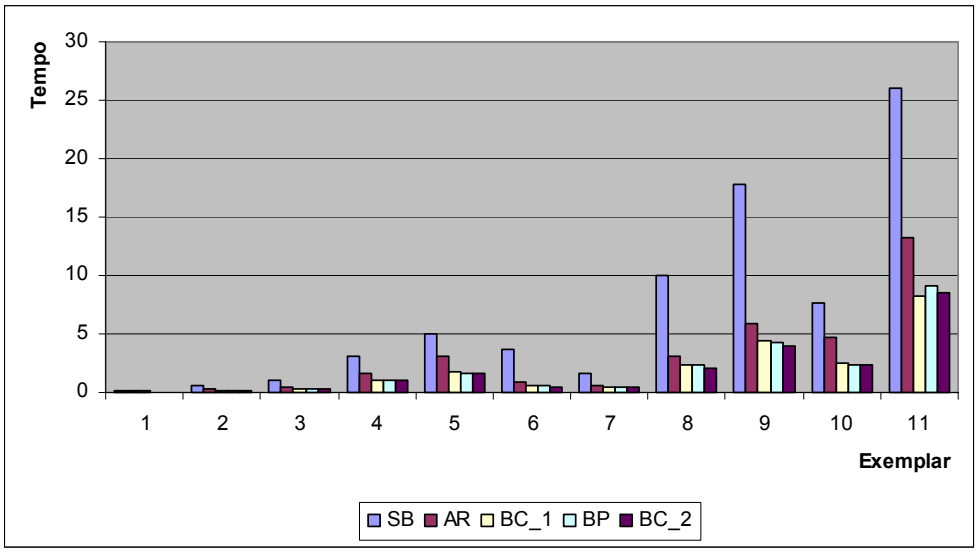

Figura 6 – Tempo de Resolução – Problemas Densos.

Observando a Figura 6, podemos dizer que de um modo geral, a busca completa com ordenação em processo (BC_2) foi um pouco melhor que a busca parcial (BP), que a busca completa com ordenação preliminar (BC_1) e principalmente, que a busca com a regra de Armijo (AR), ou seja, o tempo computacional foi favorável ao método dual simplex com busca linear por partes, utilizando o segundo procedimento de ordenação. E fica evidente a superioridade de se introduzir algum tipo de busca, pois o método dual simplex usual foi extremamente inferior em relação ao outros.

5.2 Problemas esparsos

Nesta seção apresentamos os resultados obtidos para uma classe esparsa de problemas, para os quais a matriz de restrição é da forma escada. (Figura 7). Os coeficientes não nulos da matriz **A** estão confinados nos blocos (Figura 7 ilustra 4 blocos) e foram gerados conforme descrito na seção anterior, assim como os demais dados do problema.

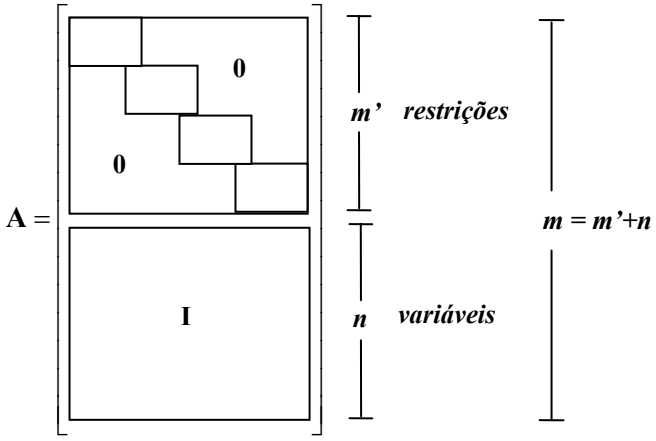

Figura 7 – Matriz **A** com 4 blocos.

Nas Tabelas 5.8 e 5.10 estão os resultados obtidos para o método dual simplex com busca completa com ordenação preliminar (BC_1) e para o método dual simplex usual (SB) e nas Tabelas 5.9 e 5.11 estão os resultados para BC_1 e para a regra de Armijo (AR), com a matriz de restrições tendo 4 blocos e 20 blocos, respectivamente. O símbolo %NZ indica a porcentagem de elementos diferente de zero na matriz de restrições definida pelos blocos e CC indica o número de colunas em comum para cada 2 blocos, aproximadamente 20%.

Tabela 5.8 – Busca Completa_1 × Sem Busca – Problemas Esparsos 4 Blocos.

				Iterações		Tempo	
Exemplar	**CC**	**%NZ**	***(m-n)×n***	**BC_1**	**SB**	**BC_1**	**SB**
1	5	28	***100×101***	*128.9*	*374.2*	*0.09*	*0.19*
2	5	28	***200×101***	*209.5*	*615.1*	*0.21*	*0.54*
3	5	28	***300×101***	*259.6*	*793.1*	*0.37*	*1.06*
4	11	29	***200×203***	*343.2*	*1295.0*	*1.35*	*4.55*
5	11	29	***300×203***	*428.4*	*1747.6*	*2.06*	*7.71*
6	23	29	***20×403***	*20.2*	*101.8*	*0.21*	*0.93*
7	11	29	***100×203***	*93.2*	*248.7*	*0.41*	*0.96*
8	23	29	***100×403***	*95.6*	*335.7*	*1.27*	*4.72*
9	23	29	***200×403***	*221.6*	*666.8*	*5.53*	*18.44*
10	11	29	***400×203***	*522.8*	*2180.2*	*3.03*	*12.09*
11	23	29	***400×403***	*908.3*	*4284.8*	*15.56*	*65.66*
			Média	***293.7***	***1149.3***	***2.73***	***10.60***

Podemos observar da Tabela 5.8, que na média o método dual simplex usual realizou 3.91 mais iterações do que o método dual simplex com a busca unidimensional completa e gastando 3.88 mais tempo. Note que a maior diferença para este caso, é observada para o exemplar 6, em que o número de variáveis é muito maior do que o número de restrições. Ou seja, mesmo para esta classe de problemas esparsos a busca unidimensional, faz uma diferença muito grande, apresentando um desempenho ainda melhor do que em problemas densos (veja Tabela 5.1).

Observa-se com relação ao caso denso (veja Tabela 5.1), para os dois métodos, que se $(m\text{-}n) > n$, então o número de iterações cresce, caso contrário, decresce. Isto talvez seja explicado, pois podem ocorrer várias restrições redundantes para o caso denso, o que não deve acontecer para a estrutura escada.

Note também nas Tabelas 5.1 e 5.7, como o número de restrições (m-n) tem maior impacto no número de iterações. Veja, por exemplo, os exemplares 9 e 10 da Tabela 5.1 (200×400 e 400×200, respectivamente). Entretanto, o tempo computacional é bem menor para o exemplar 10, pois as matrizes básicas são da ordem de 200×200 enquanto para o exemplar 9 as matrizes básicas são da ordem 400×400 (obviamente, essas matrizes 400×400 têm uma estrutura particular que poderia ser explorada).

Na Tabela 5.9 estão os resultados para a busca completa (BC_1) e para a regra de Armijo (AR).

Tabela 5.9 – Busca Completa_1 × Regra de Armijo – Problemas Esparsos 4 Blocos.

				Iterações		Tempo	
Exemplar	**CC**	**%NZ**	***(m-n)×n***	**BC_1**	**AR**	**BC_1**	**AR**
1	5	28	***100×101***	*128.9*	*173.7*	*0.09*	*0.11*
2	5	28	***200×101***	*209.5*	*269.9*	*0.21*	*0.24*
3	5	28	***300×101***	*259.6*	*360.7*	*0.37*	*0.49*
4	11	29	***200×203***	*343.2*	*504.2*	*1.35*	*1.95*
5	11	29	***300×203***	*428.4*	*709.9*	*2.06*	*3.37*
6	23	29	***20×403***	*20.2*	*27.4*	*0.21*	*0.24*
7	11	29	***100×203***	*93.2*	*103.4*	*0.41*	*0.46*
8	23	29	***100×403***	*95.6*	*117.8*	*1.27*	*1.50*
9	23	29	***200×403***	*221.6*	*278.1*	*5.53*	*7.06*
10	11	29	***400×203***	*522.8*	*910.3*	*3.03*	*5.20*
11	23	29	***400×403***	*908.3*	*1639.5*	*15.56*	*25.59*
			Média	***293.7***	***463.1***	***2.73***	***4.20***

Observamos que a busca completa (BC_1) foi ainda muito superior à regra de Armijo, apesar da diferença ter diminuído em relação ao caso denso (veja Tabela 5.3). A busca fez em média 35% menos iterações do que a regra de Armijo.

A Tabela 5.10 apresenta os resultados para a matriz de restrições tendo agora 20 blocos (problemas mais esparsos).

Tabela 5.10 – Busca Completa_1 × Sem Busca – Problemas Esparsos 20 Blocos.

				Iterações		Tempo	
Exemplar	**CC**	**%NZ**	***(m-n)×n***	**BC_1**	**SB**	**BC_1**	**SB**
1	1	5	***100×101***	*87.1*	*145.6*	*0.06*	*0.10*
2	1	5	***200×101***	*217.7*	*550.7*	*0.23*	*0.54*
3	1	5	***300×101***	*272.9*	*735.0*	*0.41*	*0.98*
4	2	5	***200×202***	*207.1*	*430.1*	*1.45*	*2.12*
5	2	5	***300×202***	*452.5*	*1618.5*	*2.32*	*7.81*
6	5	6	***20×405***	*3.5*	*8.2*	*0.04*	*0.06*
7	2	5	***100×202***	*15.3*	*26.5*	*0.05*	*0.20*
8	5	6	***100×405***	*14.2*	*32.0*	*0.20*	*0.44*
9	5	6	***200×405***	*35.6*	*67.0*	*0.70*	*1.27*
10	2	5	***400×202***	*584.8*	*2121.8*	*3.6*	*12.22*
11	5	5	***400×405***	*582.3*	*1779.0*	*12.93*	*29.65*
			Média	***224.8***	***683.1***	***1.99***	***5.03***

Note que neste caso, muito mais esparso do que o caso com 4 blocos (veja Tabela 5.8), o número de iterações caiu significantemente para os dois métodos, o que indica que para problemas muito esparsos e com a matriz de restrições tendo vários blocos pequenos, o número de iterações tende a não passar de 1.5 do número de restrições para o método dual simplex com a busca e 5.4 para o método sem a busca. De uma maneira geral, para este caso, o método dual simplex usual foi também muito inferior em relação ao método dual simplex linear por partes.

A Tabela 5.11 apresenta agora os resultados para a regra de Armijo e para a busca completa (BC_1).

Tabela 5.11 – Busca Completa_1 × Regra de Armijo – Problemas Esparsos 20 Blocos.

				Iterações		Tempo	
Exemplar	**CC**	**%NZ**	***(m-n)×n***	**BC_1**	**AR**	**BC_1**	**AR**
1	1	5	***100×101***	*87.1*	*90.5*	*0.06*	*0.06*
2	1	5	***200×101***	*217.7*	*276.1*	*0.23*	*0.28*
3	1	5	***300×101***	*272.9*	*356.7*	*0.41*	*0.52*
4	2	5	***200×202***	*207.1*	*224.5*	*1.45*	*1.52*
5	2	5	***300×202***	*452.5*	*701.2*	*2.32*	*3.41*
6	5	6	***20×405***	*3.5*	*4.0*	*0.04*	*0.03*
7	2	5	***100×202***	*15.3*	*15.3*	*0.05*	*0.06*
8	5	6	***100×405***	*14.2*	*15.4*	*0.20*	*0.21*
9	5	6	***200×405***	*35.6*	*34.5*	*0.70*	*0.65*
10	2	5	***400×202***	*584.8*	*867.9*	*3.6*	*5.76*
11	5	5	***400×405***	*582.3*	*722.3*	*12.93*	*14.82*
			Média	***224.8***	***300.76***	***1.99***	***2.48***

Observamos que para este caso muito mais esparso, a regra de Armijo diminuiu consideravelmente a desvantagem em relação a BC_1 (veja Tabela 5.9), conseguindo em

alguns casos o mesmo desempenho que a busca completa, veja os exemplares 1,7,8 e 9. Contudo, a busca completa foi ainda superior, fazendo em média 25% menos iterações.

A seguir nas Tabelas 5.12 e 5.13 apresentamos os resultados para o tempo de resolução e ordenação (com busca preliminar – BC_1) para a matriz com 4 e 20 blocos respectivamente, onde **TT** indica o tempo de resolução e **TO** o tempo de ordenação para todas as iterações.

Tabela 5.12 – Tempo Total × Tempo de Ordenação.

				Tempo	
Exemplar	**CC**	**%NZ**	***(m-n)×n***	**TT**	**TO**
1	5	28	***100×101***	*0.09*	*0.00*
2	5	28	***200×101***	*0.21*	*0.01*
3	5	28	***300×101***	*0.37*	*0.02*
4	11	29	***200×203***	*1.35*	*0.07*
5	11	29	***300×203***	*2.06*	*0.05*
6	23	29	***20×403***	*0.21*	*0.00*
7	11	29	***100×203***	*0.41*	*0.00*
8	23	29	***100×403***	*1.27*	*0.01*
9	23	29	***200×403***	*5.53*	*0.02*
10	11	29	***400×203***	*3.03*	*0.01*
11	23	29	***400×403***	*15.56*	*0.14*
			Média	***2.73***	***0.03***

Podemos observar que para o caso da matriz de restrições com 4 blocos, o tempo de ordenação foi inferior a 1% do tempo total, isto é, para problemas esparsos o tempo de ordenação é irrelevante para o desempenho do método dual simplex linear por partes.

Na Tabela 5.13 são exibidos os resultados dos tempos de ordenação e resolução para o caso da matriz de restrições tendo agora 20 blocos.

Tabela 5.13 – Tempo Total × Tempo de Ordenação.

				Tempo	
Exemplar	**CC**	**%NZ**	***(m-n)×n***	**TT**	**TO**
1	1	5	***100×101***	*0.06*	*0.00*
2	1	5	***200×101***	*0.23*	*0.01*
3	1	5	***300×101***	*0.41*	*0.01*
4	2	5	***200×202***	*1.45*	*0.00*
5	2	5	***300×202***	*2.32*	*0.04*
6	5	6	***20×405***	*0.04*	*0.00*
7	2	5	***100×202***	*0.05*	*0.00*
8	5	6	***100×405***	*0.20*	*0.00*
9	5	6	***200×405***	*0.70*	*0.00*
10	2	5	***400×202***	*3.6*	*0.03*
11	5	5	***400×405***	*12.93*	*0.05*
			Média	***1.99***	***0.01***

O mesmo pode ser observado da Tabela 5.13 para o caso anterior (veja Tabela 5.12). Ou seja, para os dois casos esparsos (matrizes com 4 e 20 blocos) o tempo de ordenação foi insignificante em relação ao tempo total.

Agora, nas Tabelas 5.14 e 5.15 são apresentados os resultados da busca completa com ordenação preliminar (BC_1) para as razões: r/n e ℓ/r, para os casos de 4 e 20 blocos da matriz de restrições, respectivamente.

Tabela 5.14 – Número de Deltas e Número de Buscas – 4 blocos.

Exemplar	CC	%NZ	$(m-n)\times n$	r/n	ℓ/r
1	5	28	***100×101***	*0.51*	*0.22*
2	5	28	***200×101***	*0.49*	*0.27*
3	5	28	***300×101***	*0.54*	*0.22*
4	11	29	***200×203***	*0.49*	*0.22*
5	11	29	***300×203***	*0.50*	*0.24*
6	23	29	***20×403***	*0.13*	*0.62*
7	11	29	***100×203***	*0.17*	*0.65*
8	23	29	***100×403***	*0.17*	*0.53*
9	23	29	***200×403***	*0.17*	*0.53*
10	11	29	***400×203***	*0.49*	*0.98*
11	23	29	***400×403***	*0.50*	*0.98*
			Média	***0.37***	***0.49***

Observarmos da Tabela 5.14, que o número de deltas (r) diminuiu em relação ao caso denso (veja Tabela 5.5) e os testes indicam que esta fração tende a diminuir gradativamente.

Para o caso em que o número de variáveis é maior do que o número de restrições, notamos que o número de deltas diminuiu significantemente. Observamos ainda que, para os exemplares maiores 10 e 11 a razão ℓ/r (observada somente para a primeira iteração) foi muito alta, ou seja, o número de buscas cresceu em relação ao caso denso.

Tabela 5.15 – Número de Deltas e Número de Buscas – 20 blocos.

Exemplar	CC	%NZ	$(m-n)\times n$	r/n	ℓ/r
1	1	5	***100×101***	*0.06*	*0.33*
2	1	5	***200×101***	*0.49*	*0.43*
3	1	5	***300×101***	*0.50*	*0.46*
4	2	5	***200×202***	*0.05*	*0.20*
5	2	5	***300×202***	*0.46*	*0.43*
6	5	6	***20×405***	*0.03*	*0.30*
7	2	5	***100×202***	*0.03*	*0.33*
8	5	6	***100×405***	*0.04*	*0.25*
9	5	6	***200×405***	*0.04*	*0.25*
10	2	5	***400×202***	*0.50*	*0.86*
11	5	5	***400×405***	*0.06*	*0.11*
			Média	***0.20***	***0.35***

Da Tabela 5.15, notamos que o número de deltas diminuiu em relação ao caso de 4 blocos (Tabela 5.14), o que era de se esperar, pois este caso é muito mais esparso. Observa-se ainda que o número de deltas foi próximo de 50% do valor de n somente para os casos em que o número de restrições é maior do que o número de variáveis. Em relação à fração ℓ/r, observamos que foi muito pequena para os exemplares 6, 8 e 9, tendo em vista que o número de deltas foi muito pequeno. Em relação ao caso denso, temos que o número de deltas (i.e., r) diminuiu assim como a fração que determina a quantidade de deltas utilizados na busca (i.e., ℓ).

Diante destes resultados (Tabelas 5.14 e 5.15), utilizamos apenas a segunda modificação (veja Seção 4) na busca unidimensional para o caso de 4 e 20 blocos da matriz de restrições, já que o número de deltas foi pequeno. Os resultados são apresentados nas Tabelas 5.16 e 5.17.

Tabela 5.16 – Busca Completa_1 × Busca Completa_2 – Problemas Esparsos 4 Blocos.

				Tempo	
Exemplar	**CC**	**%NZ**	***(m-n)×n***	**BC_1**	**BC_2**
1	5	28	***100×101***	*0.09*	*0.08*
2	5	28	***200×101***	*0.21*	*0.19*
3	5	28	***300×101***	*0.37*	*0.35*
4	11	29	***200×203***	*1.34*	*1.25*
5	11	29	***300×203***	*2.13*	*2.02*
6	23	29	***20×403***	*0.21*	*0.22*
7	11	29	***100×203***	*0.42*	*0.41*
8	23	29	***100×403***	*1.30*	*1.30*
9	23	29	***200×403***	*5.58*	*5.59*
10	11	29	***400×203***	*3.07*	*2.96*
11	23	29	***400×403***	*15.05*	*14.44*
			Média	***2.70***	***2.61***

Neste caso o método dual simplex (BC_2) foi ligeiramente melhor do que o método dual simplex (BC_1). Note que a maior diferença é observada para o exemplar 11. Logo, é de se esperar que o procedimento (BC_2) tenha um desempenho mais promissor para problemas de grande porte.

Notamos da Tabela 5.17 que na média houve um empate para os dois procedimentos de busca exata. Com uma ligeira vantagem para BC_2 na maioria dos exemplares, mas nada significante.

Contudo, podemos observar que o procedimento BC_2 teve um bom desempenho para problemas densos e para problemas com a matriz **A** tendo 4 Blocos, no entanto esta vantagem não foi verificada para o caso mais esparso, ou seja, a matriz **A** com 20 blocos e com densidade em torno de 6%.

Tabela 5.17 – Busca Completa_1 × Busca Completa_2 – Problemas Esparsos 20 Blocos.

				Tempo	
Exemplar	**CC**	**%NZ**	***(m-n)×n***	**BC_1**	**BC_2**
1	1	5	***100×101***	0.06	*0.05*
2	1	5	***200×101***	*0.22*	*0.20*
3	1	5	***300×101***	*0.43*	*0.40*
4	2	5	***200×202***	*1.51*	*1.50*
5	2	5	***300×202***	*2.36*	*2.30*
6	5	6	***20×405***	*0.03*	*0.03*
7	2	5	***100×202***	*0.06*	*0.06*
8	5	6	***100×405***	*0.20*	*0.22*
9	5	6	***200×405***	*0.58*	*0.59*
10	2	5	***400×202***	*3.45*	*3.36*
11	5	5	***400×405***	*13.67*	*13.93*
			Média	***2.05***	***2.05***

Destacamos aqui, que há grandes possibilidades de tornar o método dual simplex linear por partes mais eficiente em termos do tempo computacional para problemas de grande porte, investindo-se na eficiência da resolução dos sistemas básicos utilizando a decomposição LU (Bartels & Goloub, 1969) com heurística de Markowitz (1957) e atualização da base (Reid, 1982), procedimentos muito utilizados na prática para resolução de problemas de grande porte e esparsos. O trabalho de Silva (2002) aborda detalhadamente estas técnicas.

6. Conclusões

Neste artigo foi estudada a classe de problemas de otimização linear com restrições canalizadas (formato geral) sob a ótica da dualidade. O problema dual, embora possa ser linearizado, tem a função objetivo linear por partes. Métodos do tipo simplex (i.e., baseados na clássica estratégia simplex) foram desenvolvidos explorando a não-linearidade da função dual e chamados métodos tipo dual simplex linear por partes. Como nos métodos da otimização não-linear, surgiu a questão da busca unidimensional a ser feita na direção dual simplex. Experimentos computacionais revelam a importância da busca unidimensional, podendo reduzir o esforço computacional a 25% do critério usual de troca de bases do algoritmo dual simplex. A simplicidade da busca unidimensional faz também com que a busca exata apresente melhor desempenho do que buscas inexatas, como por exemplo, a regra de Armijo.

O método dual simplex linear por partes com busca exata (o qual se mostrou mais eficiente) foi comparado com o pacote CPLEX 4.0 e CPLEX 7.5 (Sousa, 2000 e Silva, 2002). O número de iterações, quando comparado com o CPLEX 4.0, era da ordem de 46% menor. Entretanto, quando comparado com o CPLEX 7.5, os desempenhos em termos do número de iterações se equivalem.

Sobre a importância de técnicas de ordenação mais elaboradas (ordenação de vetores surge na busca unidimensional), os experimentos computacionais revelam que é baixo o impacto

do tempo de busca sobre o tempo total. Resta ainda investigar se critérios alternativos para determinação da direção de busca (direção dual simplex) podem reduzir o número de iterações. Por exemplo, Forrest & Goldfarb (1992) reduzem de forma significativa o número de iterações e tempo computacional do método simplex usando a regra de Dantzig normalizada (*steepest edge rule*) para alguns problemas da Netlib. Além disso, resta a investigação de técnicas apropriadas para a resolução dos sistemas básicos que exploram a esparsidade de problemas de grande porte.

Reconhecimento

Os autores são gratos aos três revisores anônimos deste artigo, que contribuíram com sugestões valiosas. Este trabalho teve apoio da FAPESP e CNPq.

Referências Bibliográficas

(1) Arenales, M.N. (1984). Otimização Linear: Novos Métodos e Alguns Problemas Particulares. Tese de Doutorado, FEC-UNICAMP.

(2) Bartels, R.H. & Golub, G.H. (1969). The Simplex Method of Linear Programming Using LU-Decomposition. *Communications of the ACM*, **12**, 266-268.

(3) Bazaraa, M.S.; Jarvis, J.J. & Sherali, H.D. (1990). *Linear Programming and Network Flows.* 2nd Ed., Wiley, New York.

(4) Bertsimas, D. & Tsitsiklis, J.N. (1997). *Introduction to Linear Optimization.* 2nd Ed., Athena Scientific, Massachusetts.

(5) Bixby, R.E. (2001). *Solving Real-World Linear Programs: A Decade and More of Progress*. ILOG, Inc and Rice University. <http://www.caam.rice.edu/~bixby/default.htm>.

(6) Cavichia, M.C. & Arenales, M.N. (2000). Piecewise Linear Programming via Interior Points. *Computers & Operations Research*, **27**(13), 1303-1324.

(7) Dantizg, G.B. (1951). Maximization of a Linear Function of Variables Subject to Linear Inequalities. **In**: *Activity Analysis of Production and Allocation* [edited by T.C. Koopmans], John Wiley & Sons, New York, 339-347.

(8) Dantizg, G.B. (1963). *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ.

(9) Dantizg, G.B. & Thapa, M.N. (1997). *Linear Programming.* Ed. Springer Verlag.

(10) Dickson, J.C. & Frederick, F.P. (1960). A Decision Rule for Improvement Efficiency in Solving Linear Programming Problems with the Simplex Algorithm. *Comm. ACM*, **3**, 509-512.

(11) Forrest, J.J.H. & Goldfarb, D. (1992). Steepest-Edge Simplex Algorithms for Linear Programming. *Math. Programming*, **57**, 341-374.

(12) Fukuda, K. & Terlaky, T. (1998). On the Existence of Short Admissible Pivot Sequences. Technical Report 98-48, Delft University of Technology, The Netherlands.

(13) Goldfarb, D. & Reid, J.K. (1977). A Practicable Steepest-Edge Simplex Algorithm. *Math. Programming*, **12**, 361-371.

(14) Klee, V. & Minty, G.J. (1972). How Good is the Simplex Algorithm? **In**: *Inequalities* [edited by O. Shisha], AP, New York, 159-175.

(15) Kuhn, H.W. & Quandt, R.E. (1963). An Experimental Study of the Simplex Method. **In**: *Proc. 15th Sympos. Appl. Math*., Amer. Math. Soc. [edited by Metropolis *et al.*], Providence, R.I., 107-124.

(16) Lemke, C.E. (1954). The Dual Method of Solving the Linear Programming Problem. *Naval Research Logistics Quarterly*, **1**, 36-47.

(17) Luenberger, D.G. (1984). *Linear and Nonlinear Programming*. Addison-Wesley.

(18) Markowitz, H.M. (1957). The Elimination Form of the Inverse and its Applications to Linear Programming. *Management Science*, **3**, 255-269.

(19) Maros, I. (2003). A Generalized Dual Phase-2 Simplex Algorithm. *European Journal Operational Research*, **149**, 1-16.

(20) Murty, K.G. (1983). *Linear Programming*. John Wiley & Sons.

(21) Reid, J.K. (1982). A Sparsity-Exploiting Variant of the Bartels-Golub Decomposition for Linear Programming Bases. *Mathematical Programming*, **24**, 55-69.

(22) Shamir, R. (1987). The Efficiency of the Simplex Method: A Survey. *Management Science*, **3**, 301-334.

(23) Silva, C.T.L. (2002). Problemas de Otimização Linear Canalizados e Esparsos. Dissertação de Mestrado, ICMC-USP.

(24) Sousa, R.S. (2000). Estudos em Otimização Linear. Dissertação de Mestrado, ICMC-USP.

(25) Terlaky, T. & Zhang, S. (1993). Pivot Rules for Linear Programming: A Survey on Recent Theorical Developments. *Annals of Operation Research*, **46**, 203-233.

(26) Vanderbei, R.J. (1997). *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers.

(27) Wolfe, P. & Cutler, L. (1963). Experiments in Linear Programming. **In**: *Recent Advances in Mathematical Programming* [edited by Graves and Wolfe], McGraw Hill, New York, 177-200.