COMPLEMENTARIDADE LINEAR E APLICAÇÃO EM OPTIMIZAÇÃO GLOBAL

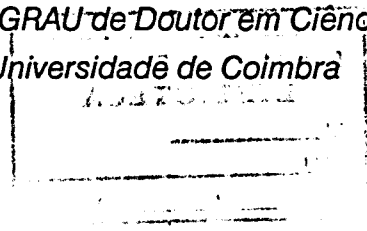
Ana Maria Ferreira Alves Faustino

Fevereiro 1992

COMPLEMENTARIDADE LINEAR E APLICAÇÃO EM OPTIMIZAÇÃO GLOBAL

Ana Maria Ferreira Alves Faustino

Dissertação para obtenção de GRAU de Doutor em Ciências, na especialidade de Investigação Operacional, na Universidade de Coimbra



Fevereiro 1992

Parcialmente financiado pelo INIC

AGRADECIMENTOS

O desenvolvimento deste trabalho só foi possível graças à colaboração e ajuda de um grande número de professores, colegas, amigos e familiares. Na impossibilidade de aqui os referir exaustivamente, devo no entanto destacar aqueles cujo contributo foi mais decisivo, expressando-lhes deste modo a minha gratidão.

Ao Professor Doutor Joaquim João Júdice, que orientou desde o início o desenvolvimento deste trabalho, assim como a demais investigação científica entretanto produzida. Sempre incansável na orientação e no acompanhamento de todos os trabalhos afins que se iam publicando, o Professor Júdice juntou à sua elevada competência científica, os favores da sua paciência, encorajamento e amizade.

Ao Professor Doutor José Manuel Ferreira Lemos, presidente do Departamento de Engenharia Civil, pela confiança demonstrada e o apoio junto do Conselho Científico da FEUP.

Ao Professor Doutor Francisco Calheiros, coordenador da secção de Matemática e Física da FEUP, pelas facilidades concedidas quanto ao trabalho de docência que me cabe na secção.

Aos Professores Doutores António Adão da Fonseca e Afonso Serra Neves pelo inestimável apoio prestado no âmbito da linha de investigação do INIC em que estou inserida.

Ao Professor Doutor Manuel Azeredo que, enquanto director do Departamento, tudo fez para que este trabalho fosse por diante.

Ao Professor Doutor Rui Guimarães por me ter proporcionado um contacto providencial com o Professor Júdice, numa altura difícil da minha carreira.

À colega e amiga Dra. Fernanda Marília Pires, pelo "software" cedido para os problemas de natureza similar à convexa e pelas frutuosas trocas de impressões durante o tempo de trabalho em que nos acompanhámos.

Aos colegas da secção de Matemática e Física, em geral, pelo interesse e apoio sempre prestado, e em particular ao Victor pelo incentivo constante, a colaboração quotidiana e dedicada e a partilha de todos os bons e maus momentos.

Às senhoras D. Maria Otília Torres e D. Paula Nunes pelo brio profissional e dedicação no trabalho de dactilografia.

À Eliana e ao Pedro que com a simpatia e amor que manifestam me fazem sentir menos culpada do tempo que não lhes dispensei.

Finalmente aos meus pais e aos meus sogros, pelo apoio dado durante todos estes anos e pelo amor e dedicação com que me ajudaram na educação dos meus filhos.

ÍNDICE

Pág.

NOTAÇÃO

R^n – espaço vectorial euclideano de dimensão n

R^n_+ – quadrante positivo do espaço vectorial euclideano R^n

Letras maiúsculas – Algumas letras maiúsculas romanas (A, B,...) são usadas para designar matrizes. Outras, romanas ou gregas (F, G, I,..., Ω) são usadas para designar conjuntos definidos em cada secção ou capítulo.

Letras minúsculas – Algumas letras minúsculas, quer romanas, quer gregas, (a, b,..., α, β,...) designam vectores coluna. Outras (i, j,..., ε, λ,...) referem-se a elementos de IN ou R. Uma letra minúscula seguida de ( ), (f(x),..., φ(x)) representa uma função de R^n em R.

Sobreíndices – Sendo números representam subvectores ou submatrizes dum vector ou duma matriz dada no capítulo ou secção. Dum modo geral representam elementos duma sucessão de vectores ou conjuntos.

∞ – infinito

$|\mu|$ – valor absoluto do número real μ

$|F|$ – cardinal do conjunto F = número de elementos distintos de F

$\subset$ – inclusão entre conjuntos

$\cap, -, \cup$ – as operações usuais entre conjuntos: intersecção, diferença e reunião

$\bigcup_{k \in K}$ – reunião estendida aos conjuntos cujos índices pertencem ao conjunto K

$\emptyset$ – conjunto vazio

{ } – conjunto definido em extensão ou compreensão

min{ } – o menor valor entre os elementos do conjunto. max{ } é similar

mínimo local
mínimo global – Para S subconjunto de R^n e f(x) função real definida em S, o valor mínimo global de f(x) em S é $f(\bar{x})$ tal que $\bar{x} \in S$ e $f(\bar{x}) \leq f(x)$ para todo o $x \in S$. Se $f(\bar{x}) \leq f(x)$ para todo o $x \in S$ tal que $\|x - \bar{x}\| < \varepsilon$ então o mínimo diz-se local

$\geq, >, \leq, <$ – operadores de relação de maior ou igual, maior, menor ou igual e menor respectivamente

♦ – fim de demonstração

A^{-1} – matriz inversa de A

A^{T} – matriz transposta de A

A^{-T} – matriz transposta da inversa de A

$A_{i.}$ – linha i da matriz A (vector linha)

$A_{.j}$ – coluna j da matriz A (vector coluna)

A_{iF} – vector linha formado pelos elementos de A da linha i e colunas $j \in F$

A_{Fj} – vector coluna formado pelos elementos de A da coluna j e das linhas $i \in F$

a_{ij} – elemento da matriz A que ocupa a linha i e a coluna j

A_{FT} – submatriz de A formada pelos elementos a_{ij} para $i \in F$ e $j \in T$

$(A|A_{FF})$ – Complemento de Schur de A_{FF} em A

a^{T} – vector linha

a_i – componente i do vector a

a_i^+, a_i^- – $a_i^+ = \max\{a_i, 0\}$; $a_i^- = \min\{a_i, 0\}$

$\sum_{j \in J} a_j$ – soma dos termos a_j para todo j do conjunto J

$\sum_{i=1}^{n} a_i$ – soma dos n primeiros elementos a_i

e – vector coluna com todas as componentes iguais a um

f_ℓ, f_u – limite inferior e superior da função f respectivamente

I_n – matriz identidade de ordem n

E_i – matriz elementar que apenas difere da matriz identidade num elemento não diagonal

P – matriz formada pelos vectores próprios p^i de uma matriz quadrada ou matriz de permutação

P_1, P_2 – matrizes de permutação

v_i – elementos de uma família de vectores ortogonais

(V, A) – rede cujos nós (v_i) pertencentes a V são ligados por arcos pertencentes a A

$(S, \overline{S})$ – corte, numa rede, que separa os nós v_0 e v_{n+1} ($v_0 \in S$, $v_{n+1} \in \overline{S}$)

$C(S, \overline{S})$ – capacidade do corte $(S, \overline{S})$

f_{ij} – fluxo através do arco orientado de v_i para v_j

c_{ij}^d – capacidade do arco orientado de v_i para v_j

c_{ij}^u – capacidade do arco não orientado que liga v_i a v_j

Δ_F^i, Δ_F^i – quantidades associadas respectivamente às variáveis x_i, $i \in F$ ou $i \in T$

NP-completo – é a designação dada a uma classe de problemas para os quais não existe nenhum algoritmo que os resolva em tempo polinomial, isto é, na pior das hipóteses o esforço computacional para os resolver cresce exponencialmente com a dimensão do problema

LCP – Problema Linear Complementar (Linear Complementarity Problem)

LP – Programa Linear (Linear Program)

NSD – negativa semidefinida (matriz)

PD – positiva definida (matriz)

PSD – positiva semidefinida (matriz)

QP – Programa Quadrático (Quadratic Program)

QP 0 – 1 - Programa Quadrático binário

SPD – simétrica positiva definida (matriz)

SLCP - Sequential LCP (método)

LBTF – forma triangular inferior por blocos (lower block triangular form)

UBTF – forma triangular superior por blocos (upper block triangular form)

BCQP - Programa Quadrático Côncavo com apenas limites nos valores das variáveis (Box Concave Quadratic Program)

BLCP – Problema Linear Complementar com limites superiores (Box Linear Complementarity Problem)

BLP – Problema de Programação Bilinear (Bilinear Program)

GSBLP – Problema de Programação Bilinear no Simplex Generalizado (Generalized Simplex Bilinear Program)

BP – Problema de Programação de dois níveis (Bilevel Program)

BLLP – Problema de Programação de dois níveis lineares

BLQP – Problema de Programação de dois níveis em que o problema do 2º nível é Quadrático Convexo e o do 1º nível é Linear

CQP – Problema de Programação Quadrática Convexa (Convex Quadratic Program)

GLCP – Problema Linear Complementar Generalizado (Generalized Linear Complementarity Problem)

LVI – Problema de Desigualdades Variacionais Lineares (Linear Variational Inequality Problem)

MLCP – Problema Linear Complementar Mínimo (Minimum Linear Complementarity Problem)

RLP – Programa Linear Relaxado (Relaxed Linear Program)

INTRODUÇÃO

O Problema Linear Complementar (LCP) tem vindo a merecer um enorme interesse nos últimos vinte anos, não só devido às suas grandes aplicações nas áreas da economia, ciência e engenharia, mas também como meio de resolução de problemas da optimização. Nesse sentido é conhecida a importância da complementaridade linear na resolução de problemas de programação linear, quadrática convexa, desigualdades variacionais lineares e da teoria dos jogos. Os LCPs dessas aplicações têm normalmente propriedades especiais que tornaram possível o desenvolvimento de algoritmos directos e iterativos para as suas soluções. A par dessa procura de técnicas eficientes para o LCP, toda uma teoria foi construída em que as classes de matrizes têm um papel fundamental. Várias generalizações do LCP têm também vindo a ser consideradas, conjuntamente com algoritmos para as suas resoluções.

À parte os casos especiais mencionados, o LCP é um problema NP-completo e por isso apenas um método enumerativo é capaz de o resolver em todos os casos. Os LCPs mais difíceis e algumas generalizações têm um papel fundamental na resolução de problemas de optimização global, nomeadamente nos programas bilineares, de dois níveis e quadráticos não convexos.

Nesta tese procurámos desenvolver processos para a resolução do LCP e algumas generalizações nos casos mais difíceis e ao mesmo tempo aplicar essas técnicas na resolução dos problemas de optimização referidos anteriormente. Assim desenvolvemos um método enumerativo híbrido capaz de resolver o LCP em todos os casos. Esse processo torna-se mais eficiente quando a matriz é negativa semi-definida (LCP côncavo). Estudámos teoricamente o LCP neste caso e propusemos um método polinomial para um caso especial. Debruçámo-nos ainda sobre a extensão de alguns métodos directos para a resolução do problema linear complementar com limites superiores (BLCP).

Como referimos anteriormente, alguns problemas de optimização global podem ser resolvidos usando uma generalização do LCP, em que é minimizada uma função linear sujeita a restrições lineares e a uma condição de complementaridade (MLCP). Neste trabalho estudámos a resolução do MLCP, propondo um algoritmo sequencial LCP (SLCP), em que é resolvida uma sucessão de LCPs (ou generalizações) até à obtenção da solução óptima do problema. Este SLCP, que incorpora o algoritmo enumerativo híbrido para a resolução dos vários LCPs, foi depois usado com algum sucesso na resolução de programas bilineares, de dois níveis e quadráticos não convexos. Finalmente estudámos o programa quadrático côncavo com apenas limites nos valores das variáveis, propondo

uma técnica que reduz o esforço computacional na obtenção de um mínimo global para esse problema.

No nosso trabalho tivemos a preocupação de testar convenientemente os nossos algoritmos com problemas de alguma dimensão. Para isso foi necessário desenvolver técnicas de implementação para os vários métodos.

Ao longo dos nove capítulos desta tese procurámos tratar os vários assuntos de uma maneira simples e rigorosa. Seguidamente apresentamos uma descrição muito sumária dos vários assuntos discutidos em cada capítulo. Assim no capítulo 1 é apresentado o LCP e são discutidas sumariamente as técnicas mais importantes para a sua resolução. São ainda introduzidas neste capítulo algumas extensões do LCP (GLCP, BLCP e MLCP). O método enumerativo híbrido é apresentado no capítulo 2. São apresentadas várias técnicas heurísticas para aumentar a eficiência do método, bem como a simplificação do algoritmo em alguns casos especiais. A solução do LCP côncavo é discutida no capítulo 3, onde é introduzido um método polinomial para um caso especial. No capítulo 4 é estudado o BLCP, sendo apresentadas extensões dos métodos de Lemke e Keller e aplicações em programação quadrática convexa e côncava. São também desenvolvidos dois algoritmos pivotais principais para a resolução do BLCP côncavo com limites finitos. Finalmente é analisado um possível método enumerativo para o BLCP.

No capítulo 5 é discutida a resolução do MLCP, sendo apresentados dois processos de resolução, nomeadamente um método de pesquisa em árvore com limites (branch-and--bound) e um método SLCP. Nos restantes capítulos desta tese é discutida a eficiência destas técnicas na resolução de alguns problemas de optimização global, designadamente programação bilinear (capítulo 6), programação de dois níveis (capítulo 7) e programação quadrática não convexa (capítulo 8). Uma aplicação da programação quadrática não convexa a um problema de gestão integrada de mini-hídricas é também apresentada neste último capítulo. Finalmente no capítulo 9 é estudado um caso especial de programação quadrática côncava com apenas restrições nos limites das variáveis.

CAPÍTULO 1

PROBLEMAS LINEARES COMPLEMENTARES

1 - Problema Linear Complementar

O Problema Linear Complementar (LCP) consiste em encontrar vectores $z \in R^n$ e $w \in R^n$ tais que

$$\begin{aligned} & w = q + Mz \\ & z \geq 0, w \geq 0 \\ & z^T w = 0 \end{aligned} \qquad (1.1)$$

onde $M \in R^{n \times n}$ e $q \in R^n$ são uma matriz e um vector dados. A notação LCP (q, M) é muitas vezes usada para representar um LCP com vector q e matriz M.

Este problema surgiu inicialmente como um processo de solução do jogo de duas matrizes e do problema de programação quadrática convexa [CODA68]. Desde então tem sido estudado intensamente e inúmeras aplicações têm surgido nas áreas de engenharia e economia [BAFA80, PAKAHA79, MAT85, JUFA86]. Há ainda uma grande diversidade de problemas de optimização importantes que podem ser reformulados como um LCP [JUMI88b].

Na definição deste problema devemos distinguir as relações lineares que definem o conjunto admissível

$$S = \{(z, w) : w = q + Mz \, , \, z \geq 0, \, w \geq 0\} \qquad (1.2)$$

e a relação complementar $z^T w = 0$. Como os vectores z e w são não negativos, esta última relação é equivalente a

$$z_i \, w_i = 0 \qquad i = 1, 2, \ldots, n \qquad (1.3)$$

Tal como em programação linear, uma solução (z, w) que satisfaça as restrições lineares diz-se **admissível**. Uma solução (z, w) que verifique as relações (1.3) diz-se **complementar**. Note-se que para cada i pelo menos uma das variáveis z_i ou w_i é igual a zero. As variáveis z_i e w_i dizem-se complementares uma da outra.

Um LCP é admissível se existir pelo menos uma solução admissível e é não admissível no caso contrário. Se o LCP é não admissível, então não tem solução. Além disso toda a solução do LCP é uma solução complementar admissível.

Na resolução de um LCP (q, M) podem surgir quatro casos:

i) LCP com uma solução única.

ii) LCP com mais do que uma solução.

iii) LCP não admissível.

iv) LCP admissível e sem solução.

Quatro tipos de metodologias têm sido propostos para a resolução do LCP, utilizando algoritmos directos, iterativos e enumerativos e a redução a problemas de optimização não convexa. Os métodos directos são baseados em conceitos semelhantes aos do método simplex para programação linear e procuram obter a solução exacta do LCP num número finito de iterações. Os métodos iterativos são normalmente extensões para o LCP de algoritmos existentes para a resolução de sistemas de equações lineares ou optimização convexa. Contrariamente aos métodos directos, estes processos procuram obter uma solução aproximada do LCP. Muito recentemente têm sido desenvolvidos algoritmos de pontos-interiores para o LCP. Esses processos podem ser considerados como iterativos em termos de funcionamento e directos em relação ao fim em vista. Todos esses algoritmos apenas podem resolver o LCP quando a matriz M pertence a certas classes especiais.

Os métodos enumerativos procuram explorar o carácter combinatório do LCP. São processos mais complexos que procuram obter uma solução do LCP por uma pesquisa implícita de uma árvore binária. Finalmente têm sido apresentadas algumas propostas de resolução do LCP usando optimização não convexa. Tais processos consistem em substituir a relação de complementaridade $z^T w = 0$ por uma restrição ou função equivalente e resolver o problema de optimização obtido.

Neste capítulo iremos discutir brevemente os processos mais importantes de cada uma das diferentes categorias. A limitação do uso dos métodos directos e iterativos a certas classes de matrizes levou a apresentar uma secção contendo alguns resultados referentes à existência de solução do LCP para algumas classes de matrizes consideradas mais relevantes para este trabalho.

2 - Classes de Matrizes

Os principais resultados teóricos relativos à existência e unicidade da solução do LCP são apresentados em termos de classes de matrizes. Um estudo exaustivo das principais classes aparece descrito em [MU88]. Neste trabalho limitar-nos-emos a apresentar alguns desses resultados. Assim as classes mais gerais em termos de existência de solução do LCP são definidas dos seguintes modos:

$M \in Q \Leftrightarrow$ LCP (q, M) tem uma solução para qualquer $q \in R^n$

$M \in Q_0 \Leftrightarrow$ LCP (q, M) tem uma solução para cada q tal que $S \neq \emptyset$

onde S é o conjunto admissível do LCP.

O estabelecimento de condições necessárias e suficientes para estas classes tem sido objecto de grande investigação. Algumas condições suficientes são apresentadas em termos de outras classes de matrizes [JU83, MU88]. Neste trabalho vão apenas considerar-se as seguintes classes:

$M \in P \Leftrightarrow$ todos os menores principais são positivos

$M \in P_0 \Leftrightarrow$ todos os menores principais são não negativos

$M \in Z \Leftrightarrow m_{ij} \leq 0$ para todo $i \neq j$

$M \in K \Leftrightarrow M \in Z$ e $M \in P$

$M \in PSD \Leftrightarrow x^T M x \geq 0$ para todo $x \in R^n$

$M \in PD \Leftrightarrow x^T M x > 0$ para todo $x \in R^n - \{0\}$

$M \in NSD \Leftrightarrow x^T M x \leq 0$ para todo $x \in R^n$

As relações de inclusão entre as diversas classes de matrizes são apresentadas no diagrama seguinte através de segmentos orientados. As justificações dessas inclusões podem encontrar-se em [JU83, MU88].

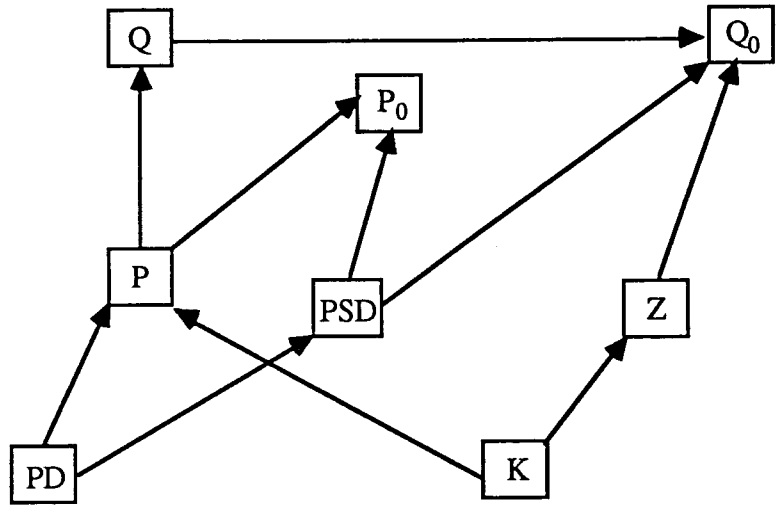


Figura 1.1

Além disso $NSD \cap Q_0 \neq \emptyset$ e $NSD \cap Q = \emptyset$ como veremos no capítulo 3 desta tese.

Em relação à unicidade da solução é possível provar [MU88, pp. 213] que, se $M \in P$, o LCP (q, M) tem solução única para cada q.

Para terminar esta secção note-se que o LCP é equivalente ao seguinte programa quadrático:

$$\begin{aligned} &\text{minimizar} \quad q^T z + \frac{1}{2} z^T (M + M^T) z \\ &\text{sujeito a} \quad Mz + q \geq 0 \qquad (1.4) \\ &\qquad\qquad z \geq 0 \end{aligned}$$

É conhecido que este programa quadrático é Convexo, Côncavo ou Não Convexo dependendo da matriz $(M + M^T)$ ser PSD, NSD ou Indefinida (isto é, não pertencer a PSD nem a NSD). Como para qualquer matriz M se tem

$$M \in \text{PSD (NSD)} \Leftrightarrow (M + M^T) \in \text{PSD (NSD)}$$

então pode estabelecer-se a seguinte classificação dos LCPs:

i) LCP convexo $\Leftrightarrow$ $M \in$ PSD

ii) LCP côncavo $\Leftrightarrow$ $M \in$ NSD

iii) LCP não convexo $\Leftrightarrow$ M é indefinida

Um outro assunto que tem merecido bastante investigação é o estudo da complexidade do LCP. Assim existem métodos polinomiais para resolver o LCP convexo [MU88, KOMIYO89]. LCPs côncavos e não convexos são problemas NP-completos [MU88], mas existem alguns casos especiais que podem ser resolvidos por métodos polinomiais [CHD70, AL89, MAN79].

3 - Métodos Directos para o Problema Linear Complementar

Estes métodos têm a propriedade de, sob determinadas condições, obterem a solução pretendida num número finito de operações. São métodos pivotais, isto é, apenas usam soluções básicas do sistema $w = q + Mz$ e operações pivotais. Seguidamente apresenta-se uma discussão breve desses conceitos. Se designarmos por C a matriz $[I_n : -M]$ e por x o vector $(w, z)^T$, então o sistema $w = q + Mz$ pode ser escrito na forma

$$Cx = q \tag{1.5}$$

Tal como em programação linear, $x = (w, z)^T$ é uma solução básica do LCP se for possível encontrar uma partição da matriz C em duas submatrizes B e N, tais que B é não singular e as componentes de x associadas às colunas de N são iguais a zero. Assim, para uma solução básica do sistema (1.5), tem-se $x = (x_B, x_N)^T$ com

$$B\,x_B = q\,, \quad x_N = 0 \tag{1.6}$$

A matriz B diz-se Base da solução básica x e as componentes dos vectores x_B e x_N designam-se por variáveis básicas e não básicas respectivamente.

Dada uma solução básica (1.6), o sistema (1.5) pode ser escrito na forma

$$x_B = B^{-1}\,q + B^{-1}\,N\,x_N \tag{1.7}$$

Se se considerar a solução básica associada à base B, então a igualdade (1.7) pode ser representada pelo quadro contraído

$$y = \begin{array}{c|c} & u \\ \hline b & A \end{array} \tag{1.8}$$

onde y e u são os vectores das variáveis básicas e não básicas da solução básica associada à base B, $b = B^{-1}q$ e $A = B^{-1}N$.

Dado o sistema (1.5) e uma solução básica x, uma operação pivotal consiste em transformar essa solução numa outra solução básica $\bar{x}$ por troca de uma variável básica x_t com uma variável não básica x_ℓ de x. Se B e $\bar{B}$ são as bases associadas às soluções básicas x e $\bar{x}$ respectivamente, $\bar{B}$ é obtida de B por troca da coluna da matriz C associada à variável x_t com a coluna de C associada à variável x_ℓ. Em termos de quadros contraídos, essa operação consiste em trocar a variável básica $y_s = x_t$ com a variável não básica $u_r = x_\ell$ de acordo com o seguinte processo:

i) resolver a equação de ordem s do sistema (1.8) em relação à variável u_r;

ii) substituir esta expressão nas outras equações.

Note-se que essa operação só é possível se a_{sr} é diferente de zero. Esse elemento diz-se o pivot da operação. Após essa operação obtém-se um novo quadro contraído da forma

$$\bar{y} = \begin{array}{|c|c|} \hline & \bar{u} \\ \hline \bar{b} & \bar{A} \\ \hline \end{array}$$

onde $\bar{y}_i = y_i$ se $i \neq s$, $\bar{y}_s = u_r$, $\bar{u}_j = u_j$ se $j \neq r$ e $\bar{u}_r = y_s$.

Além disso os elementos de $\bar{A}$ e $\bar{b}$ vêm dados por:

$$\bar{a}_{sr} = \frac{1}{a_{sr}}, \quad \bar{a}_{ir} = \frac{a_{ir}}{a_{sr}} \ (\text{se } i \neq s), \quad \bar{a}_{sj} = -\frac{a_{sj}}{a_{sr}} \ (\text{se } j \neq r)$$

$$\bar{a}_{ij} = a_{ij} - a_{ir}\,\frac{a_{sj}}{a_{sr}} \qquad (\text{para } i \neq s, j \neq r) \qquad (1.9)$$

$$\bar{b}_s = -\frac{b_s}{a_{sr}}, \quad \bar{b}_i = b_i - a_{ir}\,\frac{b_s}{a_{sr}} \qquad (\text{se } i \neq s)$$

Se A é uma matriz quadrada e se s = r, então o pivot a_{rr} é um elemento da diagonal de A e a operação diz-se **Principal**. Note-se que essas operações mantêm a complementaridade das soluções básicas.

É possível estender o conceito de operação pivotal para pivots que sejam submatrizes de A. Assim, se {F, T} é uma partição do conjunto {1,..., n}, o quadro (1.8) pode ser escrito na forma

$$\begin{array}{c|c|cc} & & u_F & u_T \\ \hline y_F = & b_F & A_{FF} & A_{FT} \\ y_T = & b_T & A_{TF} & A_{TT} \end{array} \qquad (1.10)$$

Se a submatriz principal A_{FF} é não singular, uma **operação bloco pivotal principal** com pivot A_{FF} consiste em transformar o sistema (1.10) num sistema equivalente, trocando as variáveis básicas y_F pelas variáveis não básicas u_F. O quadro obtido é do tipo

$$\begin{array}{c|c|cc} & & y_F & u_T \\ \hline u_F = & -A_{FF}^{-1} b_F & A_{FF}^{-1} & -A_{FF}^{-1} A_{FT} \\ y_T = & b_T - A_{TF} A_{FF}^{-1} b_F & A_{TF} A_{FF}^{-1} & (A|A_{FF}) \end{array} \qquad (1.11)$$

onde $(A|A_{FF}) = A_{TT} - A_{TF} A_{FF}^{-1} A_{FT}$ é o chamado **Complemento de Schur** de A_{FF} em A [CO74].

A operação pivotal por bloco atrás referida é equivalente a $|F|$ operações pivotais simples, principais ou não, que trocam sucessivamente um elemento de y_F com um elemento de u_F. Um caso particular importante é aquele em que

$$A_{FF} = \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix} \qquad (|F| = 2)$$

com $a_{ij} \neq 0$, $a_{ji} \neq 0$ e pelo menos um dos elementos da diagonal de A_{FF} é nulo. Então A_{FF} é não singular e a operação bloco pivotal é equivalente a duas operações pivotais principais simples com pivots a_{ij} e a_{ji}. À transformação que consiste nas duas sucessivas operações pivotais simples dá-se a designação de **operação pivotal dupla**, com pivots a_{ij} e a_{ji}.

Os métodos directos para a resolução do LCP começam com a solução complementar básica ($w = q$, $z = 0$) e usando operações pivotais percorrem apenas soluções básicas complementares até encontrarem uma solução admissível que é a solução do LCP.

Como só são percorridas soluções básicas e o número total dessas soluções é finito, a convergência do método fica estabelecida desde que, por imposição de certas regras, não se permita que haja repetição de soluções básicas. Os métodos diferem nas regras impostas para que se atinja a solução admissível, havendo a distinguir os principais e não

principais consoante apenas usem operações pivotais principais ou autorizem quaisquer tipos de operações pivotais.

De todos os métodos directos aquele que se pode aplicar a uma classe mais lata de matrizes é devido a Lemke [LE68]. Neste algoritmo uma variável artificial z_0 e um vector p não negativo são introduzidos de modo a que

$$\bar{q} = q + z_0 p \geq 0$$

para um certo valor positivo de z_0. Um sistema alargado da forma

$$w = q + z_0 p + Mz \qquad (1.12)$$

é assim obtido. O método de Lemke é um algoritmo pivotal não principal em que apenas soluções complementares admissíveis do sistema (1.12) são usadas, isto é, tem-se

$$w \geq 0 \, , \, z \geq 0 \, , \, z_0 \geq 0 \, , \, z^T w = 0$$

em cada iteração. Uma solução básica nessas condições diz-se **Quase-Complementar**. Para manter esse tipo de soluções básicas, a coluna do pivot é escolhida como sendo a da variável complementar da que passou a não básica na iteração anterior e a linha do pivot é calculada usando o critério do quociente mínimo usado no método simplex. Tal como no método simplex, se a linha pivotal não puder ser escolhida, então o método termina numa aresta ilimitada. Os passos do algoritmo são os seguintes:

ALGORITMO DE LEMKE

Passo 1 - Se $q \geq 0$, então $w = q$, $z = 0$ é solução do LCP. Caso contrário determine r a partir de

$$-\frac{q_r}{p_r} = \max \{-\frac{q_i}{p_i} : q_i < 0\}$$

Efectue uma operação pivotal com pivot p_r (z_0 passa a básica por troca com w_r) obtendo-se um quadro da forma (1.8).

Passo 2 - Seja u_r a variável complementar da variável que passou a não básica na iteração anterior. Seja

$$\mu_0 = \begin{cases} +\infty & \text{se } a_{ir} \geq 0 \text{ para todo } i = 1,\ldots,n \\ \min & \{\frac{b_i}{-a_{ir}} : a_{ir} < 0\} \end{cases}$$

Se $\mu_0 = +\infty$ o método termina em aresta ilimitada. De outro modo efectue uma operação pivotal com pivot a_{sr}, sendo s a primeira linha onde μ_0 é atingido.

Passo 3 - Se $z_0 = 0$, então obtém-se uma solução do LCP. De outro modo vá para o Passo 2.

Se todas as soluções básicas são não degeneradas, isto é, se $b_i>0$ para todo i=1,..., n, então a regra de escolha da variável a entrar na base garante que o algoritmo termina num número finito de iterações. De facto, cada solução básica é intersecção de duas arestas quase–complementares, chegando-se à solução básica por uma dessas arestas e abandonando-a pela outra aresta.

Se a solução básica for degenerada, poderá ser encontro de mais do que duas arestas quase-complementares e por isso a conclusão anterior deixa de ser válida. No entanto, o critério do menor índice na determinação de μ_0 no Passo 2 normalmente evita a entrada em ciclo no caso de algumas soluções básicas serem degeneradas. É mesmo possível provar [CHG79] que não é possível ocorrer um ciclo se a matriz M é PSD.

Como se pode verificar pela descrição dos passos do algoritmo, há duas possíveis terminações:

i) Terminação numa solução do LCP ($z_0 = 0$).

ii) Terminação numa aresta ilimitada, com $z_0 > 0$.

Se M é uma matriz qualquer, a terminação **ii)** não implica necessariamente a não existência de solução do LCP. Por esta razão, o método de Lemke não resolve o LCP em todos os casos. Contudo, se M é PSD ou Z, a terminação **ii)** implica que o LCP é não admissível [MU88]. Se M é P então o método de Lemke termina sempre numa solução do LCP. Como PD $\subset$ P, o mesmo acontece quando M é uma matriz PD.

Como é perfeitamente conhecido, todo o programa quadrático convexo com apenas desigualdades é equivalente a um LCP com uma matriz PSD [CODA68]. Então o método de Lemke pode ser usado nesse caso para obter um mínimo global de um programa quadrático convexo. No capítulo 4 apresenta-se uma extensão do método de Lemke capaz de resolver programas quadráticos com quaisquer tipos de restrições. Nessa altura discutiremos também uma implementação do método de Lemke para resolução de LCPs e de programas quadráticos convexos de grandes dimensões e estrutura esparsa.

Existem vários métodos pivotais principais capazes de resolver o LCP eficientemente quando a matriz M é P ou PSD. Para uma descrição desse tipo de processos sugerimos [MU88]. Em relação ao tipo de trabalho que propomos fazer nesta tese apenas o método de Keller [KE73] tem algum interesse. Seguidamente vai apresentar-se uma descrição desse processo.

O método de Keller é um processo capaz de resolver programas quadráticos convexos ou de obter um mínimo local de um programa quadrático côncavo ou não convexo com apenas restrições de desigualdade da forma

$$\begin{aligned} &\text{minimizar} \quad c^T x + \frac{1}{2} x^T Q x \\ &\text{sujeito a} \quad A x \le b \\ &\qquad\qquad x \ge 0 \end{aligned} \tag{1.13}$$

sendo Q uma matriz simétrica de ordem ℓ, $A \in R^{m \times \ell}$, $x, c \in R^{\ell}$ e $b \in R^m$. As condições de Kuhn-Tucker de primeira ordem desse programa são do tipo

$$\begin{aligned} &y = c + Q x + A^T u \\ &v = b - A x \\ &y, v, x, u \ge 0, \quad x^T y = v^T u = 0 \end{aligned}$$

A matriz deste LCP tem a forma

$$M = \begin{bmatrix} Q & A^T \\ -A & 0 \end{bmatrix}$$

e satisfaz por isso a propriedade da bissimetria. Recordemos que uma matriz M de ordem n é bissimétrica se existe uma partição {I, J} de { 1, 2,..., n } tal que

$$P^T M P = \begin{bmatrix} M_{II} & M_{IJ} \\ M_{JI} & M_{JJ} \end{bmatrix}$$

com M_{II} e M_{JJ} simétricas, $M_{IJ} = -M_{JI}^T$ e P uma matriz de permutação.

Consideremos o LCP (q, M) com uma matriz bissimétrica. Então podemos escrever

$$\begin{array}{c|c|cc} & & z_I & z_J \\ \hline w_I = & q_I & M_{II} & M_{IJ} \\ w_J = & q_J & M_{JI} & M_{JJ} \end{array} \qquad (1.14)$$

O método de Keller assume que $q_J \geq 0$, isto é, $x = 0$ é solução admissível do programa quadrático (1.13). Se tal não acontecer, uma modificação da Fase 1 do método simplex é suficiente para obter uma solução básica com $q_J \geq 0$ [KE73]. Além disso $M_{JJ} = 0$ inicialmente e portanto $M_{JJ} \in PSD$. Estas duas propriedades são mantidas em cada iteração do algoritmo. Se (1.14) representar o quadro contraído associado a uma iteração qualquer, então os passos do algoritmo nessa iteração são os seguintes:

ALGORITMO DE KELLER

Passo 1 – Se $q_I \geq 0$, faça TERM = 1 e vá para EXIT. De outro modo seja

$$r = \min\{i \in I : q_i < 0\} \qquad (1.15)$$

Passo 2 – Calcule

$$\mu_1 = \begin{cases} -\dfrac{q_r}{m_{rr}} & \text{se } m_{rr} > 0 \\ +\infty & \text{se } m_{rr} \leq 0 \end{cases}$$

$$\mu_2 = \begin{cases} \min\left\{\dfrac{q_i}{-m_{ir}} : m_{ir} < 0,\ i \in J\right\} \\ +\infty \quad \text{se } m_{ir} \geq 0 \text{ para todo } i \in J \end{cases} \qquad (1.16)$$

e seja $\mu_0 = \min\{\mu_1, \mu_2\}$

Passo 3 – Se $\mu_0 = +\infty$, faça TERM = 2 e vá para EXIT. Se $\mu_0 = \mu_2$ vá para Passo 4. De outro modo, efectue uma operação pivotal principal com pivot m_{rr}. Obtém-se assim um novo quadro da forma (1.14) com $I = I - \{r\}$ e $J = J \cup \{r\}$. Vá para Passo 1.

Passo 4 – Seja s o índice da linha onde μ_2 é atingido.

(i) Se $m_{ss} > 0$, efectue uma operação pivotal com pivot m_{ss}. Então obtém-se um novo quadro da forma (1.14) com $I = I \cup \{s\}$ e $J = J - \{s\}$. Vá para Passo 2.

(ii) Se $m_{ss} = 0$, efectue uma operação pivotal dupla com pivots m_{sr} e m_{rs}. Obtém-se um novo quadro da forma (1.14) sem alteração dos conjuntos I e J. Vá para Passo 1.

EXIT – Se TERM = 1, $(w = q, z = 0)$ é solução do LCP. Se TERM = 2 , a função quadrática é ilimitada no conjunto de restrições.

É possível mostrar que as operações pivotais usadas pelo algoritmo mantêm a bissimetria da matriz M e que além disso $q_J \geq 0$ e $M_{JJ} \in PSD$ em cada iteração [KE73].

A prova de que o método termina num número finito de operações pivotais baseia-se no facto de o valor da função quadrática decrescer de iteração para iteração quando as soluções forem sempre não degeneradas. No caso de algumas soluções serem degeneradas é possível provar que a regra de Bland [BL77], aplicada aos critérios de escolha de r (1.15) e s (1.16), é suficiente para garantir que o algoritmo termine num número finito de iterações [CHCO80].

4 - Métodos Iterativos para o Problema Linear Complementar

Dando um ponto inicial $z^0 \geq 0$, os métodos iterativos geram uma sucessão de pontos $z^1, z^2 \ldots$ obtidos por uma fórmula de recorrência do tipo

$$z^{r+1} = g(z^r)$$

que em certas condições converge para a solução do LCP. Se a escolha do ponto z^0 é determinante na obtenção da solução, a convergência diz-se local. De outro modo a convergência é global.

Estes métodos têm a vantagem de serem extremamente simples de implementar, podendo tornar-se bastante eficientes para resolver problemas de grandes dimensões com estrutura esparsa, desde que a matriz do LCP seja bem condicionada. A maior parte dos métodos iterativos baseia-se na equivalência de um LCP ao programa quadrático

$$\begin{aligned} &\text{minimizar} \quad q^T z + \frac{1}{2} z^T M z \\ &\text{sujeito a} \quad z \geq 0 \end{aligned} \tag{1.17}$$

Esta equivalência apenas se verifica se a matriz M é simétrica e a função quadrática é limitada no conjunto

$$R_+^n = \{z \in R^n : z \geq 0\}$$

No entanto, para a maior parte dos problemas que são tratados neste trabalho, a matriz M não satisfaz essas propriedades.

No caso de M ser não simétrica é ainda possível estabelecer a convergência dos métodos iterativos quando a matriz satisfaz uma condição de dominância diagonal generalizada [AH83]. Contudo essa propriedade não é satisfeita nos LCPs que estudaremos neste trabalho.

Recentemente foram desenvolvidos alguns métodos de pontos-interiores para a resolução do LCP [KOMIYO89, PAYEHAKA90, DILI91]. Esses métodos são iterativos na concepção, mas, tal como anteriormente, a sua convergência está apenas assegurada quando a matriz do LCP for PSD. Apesar destes métodos poderem ser úteis em alguns dos LCPs a estudar neste trabalho, são ainda muito recentes para os incluirmos nos processos de solução que discutiremos nesta tese.

5 - Resolução do Problema Linear Complementar por Optimização Não Convexa

Neste capítulo serão revistos alguns dos processos que são capazes de resolver o LCP sem qualquer restrição na classe da matriz M. Esses processos baseiam-se em equivalências entre o LCP e alguns problemas de optimização não convexa. O primeiro desse tipo de processos baseia-se no facto de a condição de complementaridade do LCP (1.1) se poder traduzir por restrições disjuntivas do tipo $z_i = 0$ ou $w_i = 0$ ($i = 1,\ldots, n$). Como as variáveis z e w são não negativas, tais restrições podem ser substituídas por $z_i \leq 0$ ou $w_i \leq 0$ ($i = 1,\ldots, n$). Portanto o LCP é um problema particular de programação disjuntiva do tipo

$$\begin{aligned} &\text{minimizar} && f(x) \\ &\text{sujeito a} && x \in S \\ & && x \in \bigcup_{k \in K} S_k \end{aligned}$$

onde S é dado por (1.2), f(x) é a função nula, $S_k = \{x : A^k x \geq e^k, x \geq 0\}$, com e^k vector de uns, e K o conjunto que numera os diferentes conjuntos disjuntivos S_k.

Este tipo de problemas é resolvido relaxando as restrições disjuntivas e acrescentando planos de corte do tipo $\pi^T x \geq \pi_0$, tais que

$$x \in \bigcup_{k \in K} S_k \Rightarrow \pi^T x \geq \pi_0$$

O objectivo é criar cortes o mais profundos possível, de modo a que a solução complementar seja rapidamente encontrada. Métodos para obter estes planos de corte podem ser encontrados em [JE78, OW73, RASH84], mas apenas no último destes artigos é apresentada alguma experiência computacional. No entanto, este tipo de métodos levanta problemas de estabilidade e de capacidade de armazenamento [RASH84] que não permitem resolver eficientemente LCPs de média ou grande dimensão.

Bard e Falk [BAFA82b] usam a equivalência do LCP com o problema côncavo [MAN79]:

$$\min_{z \in S} \sum_{i=1}^{n} \{\min\{0, M_{i.} z - z_i + q_i\} + z_i\}$$

onde $M_{i.}$ corresponde à linha i da matriz M e S é dado por

$$S = \{z \in R^n : q + Mz \geq 0, z \geq 0\} \qquad (1.18)$$

Para o efeito, utilizam um algoritmo de pesquisa em árvore com limites (branch-and--bound) que, na pior das hipóteses, resolve $2^{n+1}-1$ subproblemas, mas que na prática necessita de resolver um número bastante menor. Cada um desses subproblemas consiste em resolver um programa linear do tipo

$$\min_{z \in S} \sum_{i=1}^{n} (1 - \delta_i)\, z_i + \delta_i\, M_{i.}\, z$$

onde $\delta_i = \frac{1}{2}$ inicialmente, e depois toma os valores 0 ou 1 para cada $i = 1,\ldots, n$.

Em alguns casos, o algoritmo obtém a solução na fase inicial, não necessitando de ramificar a árvore. Tais casos correspondem a situações muito especiais em que o LCP se pode resolver a partir de um programa linear. Para uma descrição das classes de LCP que se podem resolver a partir de programação linear sugerimos [MAN79, AL89]. No caso geral é necessário recorrer a um método de pesquisa em árvore com limites que parece ser bastante ineficiente. Aliás os autores não apresentam quaisquer resultados computacionais que atestem a eficiência do algoritmo.

Em [AL86a] Al-Khayyal propôs o tratamento do LCP como um programa bilinear (BLP) com restrições não separadas da forma

$$\begin{aligned} \text{minimizar} \quad & z^T w \\ \text{sujeito a} \quad & w = q + Mz \\ & w \geq 0,\ z \geq 0 \end{aligned}$$

Para resolver este problema é introduzido um hiper-rectângulo

$$\Omega = \{(z, w) : 0 \leq z \leq u_z,\ 0 \leq w \leq u_w\}$$

que é sucessivamente subdividido em outros hiper-rectângulos até se obter uma solução do LCP. Essas partições são obtidas resolvendo programas lineares. É ainda possível encontrar processos de eliminar alguns hiper-rectângulos dessa partição, quando se verifica que nenhuma solução do LCP pode aí existir. Deste modo obtém-se um método de pesquisa em árvore com limites que, de acordo com a experiência computacional apresentada em [AL86a], se revela menos eficiente que um método tipo enumerativo híbrido que será objecto de estudo neste trabalho. Daí não nos debruçarmos sobre este processo de solução do LCP a partir de programação bilinear com restrições não separadas.

É ainda possível provar [AL86b] que o LCP é equivalente a um programa bilinear com restrições separadas da seguinte forma

$$\begin{aligned} \text{minimizar} \quad & g(z, y) = e^T z + q^T y + y^T (M - I_n) z \\ \text{sujeito a} \quad & Mz + q \geq 0,\ y \leq e,\ z \geq 0,\ y \geq 0 \end{aligned} \tag{1.19}$$

onde $e = (1,\ldots, 1)^T \in R^n$ e I_n é a matriz identidade de ordem n. Além disso, z é solução do LCP (q, M) se e só se (z, y) é o mínimo global do BLP(1.19) e $g(z, y) = 0$. O facto de ser conhecido o valor da função no mínimo global torna este tipo de processo mais

recomendável para a resolução do LCP. Esse assunto será discutido no capítulo 6 desta tese.

Entre as reduções a problemas não convexos, a proposta de Pardalos e Rosen [PAROS88] é a única que apresenta alguma experiência computacional com problemas de dimensão média. Esse algoritmo resolve, de forma não global, o programa quadrático não convexo

$$\begin{aligned} &\text{minimizar} && q^T z + \frac{1}{2} z^T (M + M^T) z \\ &\text{sujeito a} && Mz + q \geq 0,\ z \geq 0 \end{aligned} \tag{1.20}$$

para vários pontos iniciais z^i em princípio próximos da solução. Esses pontos são escolhidos entre as soluções de 2n programas lineares, cujas funções objectivo são do tipo $v_i^T z$ e $-v_i^T z$, com v_i (i = 1,..., n) constituindo n direcções ortogonais.

Se entretanto a solução não for encontrada por este processo recorre-se à execução de um programa 0 – 1 misto da forma:

$$\begin{aligned} &\text{maximizar} && \rho \\ &\text{sujeito a} && 0 \leq (My)_i + q_i \rho \leq 1 - \delta_i && i = 1,\ldots, n \\ & && 0 \leq y_i \leq \delta_i && i = 1,\ldots, n \\ & && \delta_i \in \{0,\ 1\} && i = 1,\ldots, n \\ & && 0 \leq \rho \leq 1 \end{aligned}$$

Se a solução $\bar{\rho}$ deste problema for maior do que zero, $z = y / \bar{\rho}$ é solução do LCP(1.1). Se $\bar{\rho} = 0$, o LCP não tem solução [PAROS88].

É difícil comparar a eficiência deste método com a de outros algoritmos, pois apenas são indicados tempos de execução em [PAROS88]. Contudo, parece ser um método que exige um elevado trabalho e a redução a um problema de programação inteira mista não parece ser a aposta mais conveniente para resolver o LCP. Um outro processo de resolução do LCP usando programação inteira 0 – 1 mista foi proposto anteriormente por Ibaraki [IB71]. Se u_w e u_z são limites superiores nos valores das variáveis w e z, então o LCP é equivalente ao programa 0 – 1 misto

$$w = q + Mz$$

$$\left.\begin{array}{l} 0 \le w_j \le \delta_j \, u_{w_j} \\ 0 \le z_j \le (1-\delta_j) \, u_{z_j} \\ \delta_j \in \{0, 1\} \end{array}\right\} \quad j = 1,\ldots, n$$

Alguma experiência computacional com este tipo de processo é apresentada em [KAHA78]. Esses resultados revelam que resolver LCPs usando este tipo de processo é menos eficiente do que usar um método enumerativo mesmo quando este último algoritmo é o menos sofisticado possível.

6 - Métodos Enumerativos para o Problema Linear Complementar

Este tipo de métodos baseia-se no carácter combinatório da definição do LCP, que sugere uma pesquisa em árvore para encontrar uma sua solução.

Estes métodos surgiram inicialmente com o objectivo de enumerar todas as soluções do LCP [MU70, GALE71, MITJA79, JUMI88a]. No entanto, o esforço computacional para obter todas as soluções do LCP é proibitivo mesmo para LCPs de pequena dimensão. Por isso estes algoritmos passaram a ser desenvolvidos para determinar uma só solução [KAHA78, AL87, JUFA88a].

Para encontrar uma ou todas as soluções complementares é preciso encontrar um ou todos os conjuntos $\Delta \subset \{1,\ldots, n,\ldots, 2n\}$ tais que $i \in \Delta$ se e só se $i+n \notin \Delta$ e para cada um desses conjuntos Δ formar a matriz A tal que $A_{j.} = I_{j.}$ se $j+n \in \Delta$ ou $A_{j.} = -M_{j.}$ se $j \in \Delta$ e verificar se $A^{-1} q \ge 0$. O método de Murty [MU70] procura determinar um ou vários conjuntos Δ usando para isso a resolução de uma sequência de problemas de cobertura. Nenhuma experiência computacional é apresentada. Contudo, o método não parece ser eficiente, dado que se tem de resolver problemas de cobertura e além disso as dimensões desses problemas vão aumentando passo a passo.

Os métodos descritos em [KAHA78, GALE71, MITJA79, JUMI88a] usam em cada iteração soluções admissíveis e terminam quando uma solução complementar é obtida. De entre esses processos apenas o algoritmo descrito em [JUMI88a] incorpora heurísticas que o tornam mais atractivo. Os autores apresentam ainda alguma experiência computacional para problemas de pequena dimensão, que mostram a validade da

$$w = q + Mz$$

$$\left.\begin{array}{l} 0 \le w_j \le \delta_j u_{w_j} \\ 0 \le z_j \le (1-\delta_j) u_{z_j} \\ \delta_j \in \{0, 1\} \end{array}\right\} \quad j = 1,\ldots, n$$

Alguma experiência computacional com este tipo de processo é apresentada em [KAHA78]. Esses resultados revelam que resolver LCPs usando este tipo de processo é menos eficiente do que usar um método enumerativo mesmo quando este último algoritmo é o menos sofisticado possível.

6 - Métodos Enumerativos para o Problema Linear Complementar

Este tipo de métodos baseia-se no carácter combinatório da definição do LCP, que sugere uma pesquisa em árvore para encontrar uma sua solução.

Estes métodos surgiram inicialmente com o objectivo de enumerar todas as soluções do LCP [MU70, GALE71, MITJA79, JUMI88a]. No entanto, o esforço computacional para obter todas as soluções do LCP é proibitivo mesmo para LCPs de pequena dimensão. Por isso estes algoritmos passaram a ser desenvolvidos para determinar uma só solução [KAHA78, AL87, JUFA88a].

Para encontrar uma ou todas as soluções complementares é preciso encontrar um ou todos os conjuntos $\Delta \subset \{1,\ldots, n,\ldots, 2n\}$ tais que $i \in \Delta$ se e só se $i+n \notin \Delta$ e para cada um desses conjuntos Δ formar a matriz A tal que $A_{j.} = I_{j.}$ se $j+n \in \Delta$ ou $A_{j.} = -M_{j.}$ se $j \in \Delta$ e verificar se $A^{-1} q \ge 0$. O método de Murty [MU70] procura determinar um ou vários conjuntos Δ usando para isso a resolução de uma sequência de problemas de cobertura. Nenhuma experiência computacional é apresentada. Contudo, o método não parece ser eficiente, dado que se tem de resolver problemas de cobertura e além disso as dimensões desses problemas vão aumentando passo a passo.

Os métodos descritos em [KAHA78, GALE71, MITJA79, JUMI88a] usam em cada iteração soluções admissíveis e terminam quando uma solução complementar é obtida. De entre esses processos apenas o algoritmo descrito em [JUMI88a] incorpora heurísticas que o tornam mais atractivo. Os autores apresentam ainda alguma experiência computacional para problemas de pequena dimensão, que mostram a validade da

proposta. As heurísticas que o algoritmo incorpora permitem fixar o valor de algumas variáveis z_i e w_i em zero, reduzindo assim a pesquisa na árvore. Além disso essas heurísticas são baseadas em algoritmos do tipo simplex, o que torna fácil a sua implementação para LCPs de média e grande dimensão. No capítulo 2 vai discutir-se esse método enumerativo e desenvolver outros procedimentos heurísticos que irão aumentar a sua eficiência.

Em [AL87], Al-Khayyal sugeriu o uso de um método de gradiente reduzido modificado para diminuir a pesquisa de uma solução do LCP. No capítulo 2 vai demonstrar-se que tal técnica é bastante útil e permitirá o desenvolvimento de um algoritmo enumerativo híbrido, que se mostrará eficiente para a resolução de LCPs de média e grande dimensão de estrutura esparsa.

7 - Algumas Generalizações do Problema Linear Complementar

Em muitas aplicações de economia e engenharia existem situações em que as variáveis estão relacionadas entre si por relações lineares, mas a condição de complementaridade apenas ocorre entre algumas dessas variáveis. Assim aparece o problema linear complementar generalizado (GLCP) da forma

$$\begin{aligned} & w = q + Mz + Ny \\ & w \geq 0,\ z \geq 0,\ y \geq 0 \\ & w_i z_i = 0 \qquad i = 1,\ldots, n \end{aligned} \tag{1.21}$$

com $w \in R^m$, $z \in R^n$, $y \in R^t$, $M \in R^{m \times n}$, $N \in R^{m \times t}$, $q \in R^m$ e $m \geq n$. Existem ainda problemas de optimização em que uma função linear é minimizada num conjunto de restrições que constituem um LCP ou GLCP. Esse problema será denotado por MLCP e é normalmente definido da seguinte forma

$$\begin{aligned} \text{minimizar} \quad & c^T z + d^T y \\ \text{sujeito a} \quad & w = q + Mz + Ny \\ & w \geq 0,\ z \geq 0,\ y \geq 0 \\ & w_i z_i = 0 \qquad i = 1,\ldots, n \end{aligned} \tag{1.22}$$

Apesar da importância destas duas generalizações do LCP, não têm sido desenvolvidos algoritmos capazes de resolver estes problemas eficientemente. Nesta tese iremos estudar

estes problemas com algum detalhe, discutindo não só algoritmos para a sua solução, como também o seu uso na resolução de alguns problemas de optimização global.

Uma outra generalização do LCP, que ocorre bastante em aplicações e na resolução de problemas de optimização usando a complementaridade linear, considera limites superiores b_i nas variáveis z_i e condições de complementaridade referentes aos limites inferiores e superiores. Esse problema é denotado por BLCP e é definido do seguinte modo:

$$
\begin{aligned}
& w = q + Mz \\
& \left.\begin{aligned}
& a_i \le z_i \le b_i \\
& z_i = a_i \Rightarrow w_i \ge 0 \\
& z_i = b_i \Rightarrow w_i \le 0 \\
& a_i < z_i < b_i \Rightarrow w_i = 0
\end{aligned}\right\} \qquad i = 1,\ldots, n \qquad (1.23)
\end{aligned}
$$

com $-\infty \le a_i < b_i \le +\infty$ para todo $i = 1,\ldots, n$.

Tal como no caso do LCP, iremos distinguir GLCPs, MLCPs e BLCPs Convexos, Côncavos e Não Convexos, dependendo da matriz M_1 associada às variáveis complementares ser PSD, NSD ou indefinida respectivamente. Note-se que $M_1 = M$ no caso do BLCP e M_1 é uma submatriz de M nas definições dos GLCP e MLCP.

Devido à sua importância, vários algoritmos directos e iterativos têm sido propostos para a resolução do BLCP [JUPI89a, MOTO91, PAYEHA90]. Em particular BLCPs convexos podem ser resolvidos por métodos polinomiais. Por outro lado BLCPs côncavos e não convexos são problemas NP-completos. No capítulo 4 estudaremos com algum detalhe o BLCP.

CAPÍTULO 2

MÉTODOS ENUMERATIVOS PARA PROBLEMAS LINEARES COMPLEMENTARES

1 - Método Enumerativo Simples para LCP e GLCP

Tal como foi referido anteriormente, o problema linear complementar LCP consiste em encontrar vectores $z \in R^n$ e $w \in R^n$ tais que

$$w = q + Mz, \quad z \geq 0, \quad w \geq 0, \quad z^T w = 0 \qquad (2.1)$$

Nestas restrições há que distinguir as lineares que definem o conjunto admissível

$$S = \{(z, w) : w = q + Mz \, , \, z \geq 0, \, w \geq 0\} \qquad (2.2)$$

e a condição de complementaridade $z^T w = 0$. Como as variáveis z_i e w_i são não negativas, esta última condição é equivalente a

$$z_i \, w_i = 0 \qquad i = 1, 2, \ldots, n \qquad (2.3)$$

Uma maneira simples e geral de resolver o LCP consiste em obter inicialmente uma solução admissível (isto é, um elemento do conjunto S) e, usando sempre soluções admissíveis, explorar uma árvore binária da forma

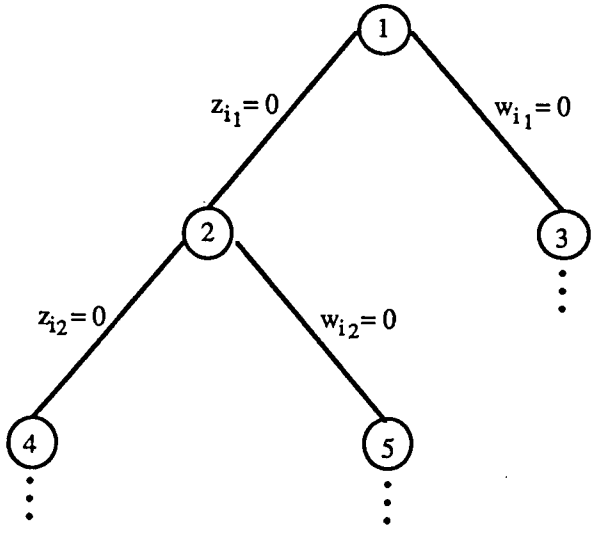


Fig. 2.1

com $i_1, i_2, \ldots$ inteiros pertencentes ao conjunto $\{1, \ldots, n\}$, até se obter uma solução complementar. Como essa solução é complementar e admissível, então é uma solução do LCP.

No nó inicial há que obter uma primeira solução admissível. Tal é conseguido resolvendo o programa linear

$$\begin{aligned} &\text{minimizar} && z_0 \\ &\text{sujeito a} && w = q + z_0 p + Mz \\ & && w, z, z_0 \geq 0 \end{aligned} \qquad (2.4)$$

com z_0 uma variável artificial e p um vector não negativo satisfazendo $p_i > 0$ para todo i tal que $q_i < 0$. Neste capítulo iremos discutir vários processos de resolver este programa linear.

Cada um dos outros nós é gerado resolvendo um subproblema que consiste em minimizar uma variável z_{i_k} ou w_{i_k}, sujeita às restrições lineares do LCP e restrições da forma $z_\ell = 0$ ou $w_\ell = 0$, correspondentes às variáveis marcadas (isto é, fixas a zero) no caminho que define o ramo que está a ser explorado.

Assim, por exemplo, para se gerar o nó 4 da figura 2.1, resolve-se o problema de programação linear

$$\begin{aligned} &\text{minimizar} && z_{i_2} \\ &\text{sujeito a} && w = q + Mz,\ z \geq 0,\ w \geq 0,\ z_{i_1} = 0 \end{aligned}$$

Para resolver este programa linear, usa-se uma modificação da Fase 2 do método simplex que será discutida mais adiante. Dois casos podem ocorrer:

i) a variável minimizada tem valor igual a zero, e então é marcada de modo a permanecer igual a zero em todos os caminhos descendentes da árvore;

ii) se o mínimo da variável é positivo, então o ramo é podado, pois não existe nenhuma solução do LCP que satisfaça as condições de complementaridade impostas ao longo do caminho já existente.

O método enumerativo procura resolver o LCP gerando sucessivos nós da árvore de acordo com o processo anterior, até que se obtenha uma solução complementar ou se

estabeleça que o LCP não tem solução. O algoritmo consiste genericamente nos seguintes passos:

Passo 1 – Procura determinar-se uma solução inicial admissível. Se tal não for possível, o LCP é não admissível e não tem solução.

Passo 2 – Se a solução é complementar, então é uma solução do LCP e o processo termina. De outro modo, segue para Passo 3.

Passo 3 – Escolhem-se duas variáveis complementares z_i, w_i básicas e positivas e geram-se dois novos nós resolvendo dois programas lineares de minimização de z_i e w_i. Para cada um dos problemas, se o mínimo da variável é positivo o respectivo ramo é podado. De outro modo, a variável com mínimo zero é marcada.

Passo 4 – Se não houver nós por explorar, o LCP não tem solução. Caso contrário, escolhe-se um nó para explorar e volta-se ao Passo 2.

Este algoritmo só é eficiente se houver poucos nós a serem visitados antes de se obter a solução ou de se provar que ela não existe. Por isso é necessário estabelecer processos heurísticos que acelerem a pesquisa dessa solução. Esses processos dizem respeito à escolha do par (z_i, w_i) no Passo 3 e à escolha do nó no Passo 4. Também se constatou que é importante obter uma "boa" solução inicial no Passo 1 e encontrar uma "boa" técnica para gerar os nós da árvore. Estas heurísticas basearam-se numa vasta experiência computacional. Por isso a sua explicação será acompanhada com a apresentação de resultados da solução de alguns problemas-teste. A definição desses problemas é apresentada na última secção deste capítulo.

I - Escolha do Nó

Em [AL87], Al-Khayyal sugeriu a escolha do nó, no Passo 4, como aquele para o qual o produto $P = z^T w$ é mínimo. A nossa experiência computacional mostrou que a incorporação de tal critério normalmente aumenta a eficiência do método enumerativo relativamente a outros processos de escolha de nós. De facto, apesar de obrigar muitas vezes a repor informação referente à exploração de um nó anteriormente abandonado, obtém-se mais rapidamente a solução, pois não se corre o risco de estar a explorar completamente ramos que não forneceriam uma solução.

Outro critério [JUFA88a], que a nossa experiência computacional mostrou ser importante, consiste na escolha do nó para o qual é mínimo o número NCP de pares de variáveis complementares básicas da correspondente solução básica. Como os dois critérios são muitas vezes conflituosos entre si, tentou obter-se um compromisso entre os dois critérios, tendo-se optado pela escolha do nó que minimiza a quantidade

$$NCP + \frac{P}{\rho}$$

onde ρ é um valor previamente fixado.

No quadro 2.1 é apresentada a eficiência destas técnicas de escolha de nós da árvore. Nesse quadro T, NI e ND representam respectivamente o tempo CPU em segundos, o número de operações pivotais e o número de nós gerados de que o método enumerativo necessita para resolver alguns dos problemas-teste do quadro 2.1. Ainda nesse quadro o símbolo > significa que o método enumerativo foi incapaz de obter a solução ao fim do tempo que é indicado. Nestes casos, o tempo de referência resulta da informação de outros tempos já obtidos para a resolução do mesmo problema com outras versões. É também importante referir que o mesmo problema-teste aparece várias vezes com resultados bastante diferentes, devido ao facto de se terem escolhido heurísticas distintas em outros passos do método enumerativo.

	min P			min NCP			$\min(NCP + \frac{P}{\rho})$					
							$\rho = 10$			$\rho = 100$		
Prob.	T	ND	NI	T	ND	NI	T	ND	NI	T	ND	NI
R1	0.9	11	22	2.0	23	45	0.9	11	22	1.1	13	26
R2	26.0	73	321	14.7	45	182	26.0	71	319	14.7	45	182
R3	56.5	69	376	56.6	69	376	56.6	69	376	56.6	69	376
R3	227.0	229	1538	81.0	77	504	96.0	95	578	81.0	77	504
R4	75.0	81	360	64.0	81	310	69.0	83	330	69.0	83	330
R4	260.0	217	1822	>420.0	>335	>3038	266.0	223	1858	313.0	265	2185
R4	85.0	71	486	>300.0	>189	>1860	72.0	61	404	128.0	97	709

Quadro 2.1

Para a maior parte dos problemas testados, o valor $\rho = 10$ mostrou fornecer um bom compromisso entre os dois critérios de minimização das quantidades P e NCP referidas anteriormente. Este novo critério é normalmente o mais consistente entre aqueles que foram analisados, conforme se verifica das indicações do quadro 2.1. Neste quadro

também se apresentam resultados obtidos para $\rho = 100$, que confere um maior peso à influência de NCP. Esses resultados mostram que a escolha desse valor para ρ é menos conveniente do que a anterior.

II - Geração dos Nós

Como foi referido anteriormente, para se gerar um novo nó no Passo 3 é necessário resolver um programa linear, que consiste em minimizar uma variável x_i sujeita às restrições lineares do LCP e de restrições $z_i = 0$ ou $w_i = 0$ correspondentes às variáveis que foram marcadas anteriormente.

O primeiro algoritmo utilizado é uma pequena variação à Fase 2 do método simplex. Esse processo será denotado por MINVAR e o seu objectivo é minimizar uma variável x_i partindo de uma solução admissível em que x_i é básica.

O programa linear a resolver é do tipo

$$\begin{aligned} &\text{minimizar} \quad x_i \\ &\text{sujeito a} \quad Ax = q \\ &\qquad\qquad x \geq 0 \ \text{ e } \ x_j = 0 \ \text{ para todo } j \in \bar{J} \end{aligned} \tag{2.5}$$

onde $\bar{J}$ é o conjunto dos índices das variáveis marcadas em níveis anteriores, x_i é uma variável z_{i_k} ou w_{i_k} e $A = [I_n : -M]$. Como apenas soluções básicas admissíveis são usadas pelo algoritmo, só é necessário aplicar a Fase 2 do método simplex para minimizar a variável x_i. Seja t a linha da variável x_i. Se designarmos por $\bar{q}_i$ e $\bar{a}_{ij}$ os elementos actualizados do quadro contraído correspondente à solução básica corrente, então vários casos podem acontecer e são discutidos a seguir.

i) Se $\bar{q}_t = 0$ e

$$r = \min \{ j : \bar{a}_{tj} \neq 0 \ \text{ com } \ j \notin \bar{J} \} \tag{2.6}$$

então efectua-se uma operação pivotal com pivot $\bar{a}_{tr}$ tornando a variável x_i não básica que assim pode ser marcada. Se $\bar{a}_{tj} = 0$ para todas as colunas não marcadas $j \notin \bar{J}$, a variável x_i continua básica, mas pode ser marcada porque nunca tomará valores diferentes de zero.

ii) Se $\bar{q}_t > 0$ e $\bar{a}_{tj} \geq 0$ para todas as colunas j não marcadas, então x_i tem mínimo positivo e o ramo pode ser podado. De outro modo, seja

$$r = \min \left\{ j : \bar{a}_{tj} = \min_{k} \{ \bar{a}_{tk} < 0 \text{ e } k \notin \bar{J} \} \right\} \qquad (2.7)$$

Então determina-se s tal que

$$s = \min \left\{ i : \frac{\bar{q}_i}{-\bar{a}_{ir}} = \mu_0 = \min_{j} \{ \frac{\bar{q}_j}{-\bar{a}_{jr}} : \bar{a}_{jr} < 0 \} \right\} \qquad (2.8)$$

e faz-se uma operação pivotal com pivot $\bar{a}_{sr}$. Se s = t então a variável x_i torna-se não básica e é marcada. Caso contrário, uma nova iteração é efectuada.

Para evitar possíveis entradas em ciclo, em casos degenerados, usa-se a regra de Bland [BL77], passando nesse caso o critério (2.7) a ter a forma

$$r = \min \{ j : \bar{a}_{tj} < 0 \quad \text{e} \quad j \notin \bar{J} \} \qquad (2.9)$$

Este processo tem a desvantagem de não controlar o número de pares de variáveis complementares positivas. Nesse sentido propusemos em [JUFA88a] uma modificação do método BRES (Basis Restricted Entry Simplex) desenvolvido por Wolfe [WO59] para a resolução de programas quadráticos convexos. No algoritmo BRES uma variável não básica z_i(ou w_i) com um custo reduzido negativo só pode passar a básica se a sua complementar for não básica. Este algoritmo BRES pode ter três terminações possíveis:

TERM = 1 – Uma solução básica com $\min x_i = 0$.

TERM = 2 – Uma solução básica óptima do programa (2.5) em que $\min x_i > 0$.

TERM = 3 – Uma solução básica não óptima do programa linear (2.5), mas sem haver variáveis não básicas candidatas a passar a básicas.

O algoritmo MBRES proposto em [JUFA88a] consiste essencialmente em continuar o processo de minimização no caso em que a terminação TERM = 3 ocorre. Os passos do algoritmo são apresentados a seguir.

ALGORITMO MBRES para gerar um nó k > 1

Passo 0 – Seja NCP o número de pares de variáveis complementares (z_i, w_i), com z_i e w_i ambas básicas na solução básica.

Passo 1 – Aplique o algoritmo BRES. Se uma solução básica complementar for encontrada quando o algoritmo BRES está a ser usado, faça TERM = 1 e NCP = 0 e vá para EXIT. Se o algoritmo BRES terminar com TERM = 1 ou TERM = 2, vá para EXIT. Caso contrário (TERM = 3), vá para Passo 2.

Passo 2 – Faça NCP = NCP + 1 e execute uma operação pivotal com pivot a pertencer a uma coluna com custo reduzido negativo e vá para Passo 1.

EXIT – Se TERM = 1 e NCP = 0, uma solução do LCP foi encontrada e o método enumerativo pára. Se TERM = 1 e NCP > 0, o nó k foi gerado. Se TERM = 2, o nó k não pode ser gerado e o ramo é podado.

Como no algoritmo MBRES as variáveis não básicas a entrar na base têm que satisfazer um critério adicional, este processo requer geralmente mais iterações do que o algoritmo MINVAR. Contudo, o processo MBRES não só controla a quantidade NCP, como também controla a quantidade P, o que se vai traduzir muitas vezes numa redução do esforço computacional do método enumerativo, conforme se pode ver no quadro 2.3.

Em certos casos, porém, o uso do processo MBRES obtém nós para os quais NCP e P são bastante pequenos, mas não conduzem a qualquer solução complementar. Se tal acontecer o método enumerativo gasta muito tempo a pesquisar nós que não necessitariam de ser explorados. É esta a razão dos alguns maus resultados da variante MBRES apresentados no quadro 2.3.

A experiência computacional revelou que em bastantes casos uma solução complementar era obtida durante a execução dos processos MINVAR ou MBRES. Essa solução seria no entanto perdida pelo processo enumerativo, dado que não seria a última solução obtida pelo processo. Daí se fazer a verificação de todas as soluções básicas usadas pelos processos MINVAR ou MBRES quanto à sua complementaridade. Como esta verificação é bastante rápida (ver secção 7), esta modificação não aumenta muito o tempo de execução e justifica-se plenamente pela grande frequência com que uma solução complementar é encontrada durante as iterações intermédias desses processos.

III - Solução Inicial

Como foi referido anteriormente, o Passo 1 do método enumerativo consiste em encontrar uma solução admissível para as restrições lineares do LCP. Em [JUFA88a] são apresentados vários processos de obter essa solução que se discutem a seguir.

Como as restrições lineares do LCP são constituídas pelas desigualdades $z \geq 0$, $Mz + q \geq 0$, então o primeiro processo consiste simplesmente em

$$\begin{aligned} &\text{minimizar} \quad e^T z \\ &\text{sujeito a} \quad Mz + q \geq 0,\ z \geq 0 \end{aligned} \tag{2.10}$$

com $e = (1,\ldots, 1)^T$, usando o método dual simplex (MÉTODO 1). Com efeito, como esse programa linear tem uma solução dual admissível, o uso do método dual simplex é recomendável e a solução óptima do programa linear (2.10) é uma solução admissível do LCP.

Um outro processo igualmente simples de obter uma solução admissível do LCP consiste em usar a Fase 1 do método simplex com uma única variável artificial [MU76, pp.136-138]. Introduz-se assim uma variável artificial z_0 e o vector $e = (1,\ldots, 1)^T$ nas restrições lineares do LCP e resolve-se o programa linear

$$\begin{aligned} &\text{minimizar} \quad z_0 \\ &\text{sujeito a} \quad Mz + z_0 e + q \geq 0,\ z \geq 0,\ z_0 \geq 0 \end{aligned} \tag{2.11}$$

Se $q \geq 0$ a solução $z = 0$ é uma solução complementar do LCP. Seja então

$$q_r = \min\{q_i : q_i < 0,\ i = 1,\ldots, n\}$$

A variável básica na linha r é então substituída pela variável artificial z_0, obtendo-se uma solução admissível de (2.11) com $z_0 > 0$. Para minimizar z_0 pode aplicar-se o algoritmo MINVAR (MÉTODO 2) ou MBRES (MÉTODO 3). Note-se que no MÉTODO 3 se tenta controlar os parâmetros NCP e P que são importantes no desenvolvimento do processo enumerativo.

Uma outra hipótese a explorar será a de usar o método de Lemke para tentar resolver o LCP. Se o método terminar com uma solução do LCP, então o método enumerativo não chega a ser usado. Se se terminar numa aresta ilimitada, um dos algoritmos MINVAR ou MBRES é aplicado para minimizar a variável z_0, obtendo-se deste modo uma solução inicial básica admissível para o método enumerativo. Estes processos híbridos são designados por MÉTODO 4 e MÉTODO 5, conforme se usa MINVAR ou MBRES respectivamente.

No quadro 2.2 é apresentada a eficiência dos cinco métodos discutidos nesta secção na resolução de alguns dos problemas-teste introduzidos na secção 8. Neste quadro NI e T

representam o número de operações pivotais e o tempo CPU em segundos de cada um dos processos e NCP e P são as quantidades referidas na secção anterior.

MÉTODO		PROBLEMA						
		R3	R4	R5	R6	R7	R8	R9
1	NCP	10	9	14	22	25	21	21
	P	361	526	959	343	766	879	1161
	NI	47	29	53	27	46	81	104
	T	3.2	1.5	4.9	1.7	3.1	9.3	23
2	NCP	8	12	13	22	23	27	27
	P	180	1217	849	343	590	1261	1839
	NI	38	20	38	27	57	61	57
	T	2.6	0.8	2.7	1.8	4.7	6.4	8.1
3	NCP	5	5	3	20	21	20	11
	P	597	268	402	350	604	1893	1537
	NI	34	48	65	26	55	70	102
	T	1.5	3.3	6.1	1.8	4.3	6.7	15.5
4	NCP	9	10	8	22	21	25	23
	P	412	724	573	243	568	1307	2143
	NI	42	73	37	33	56	65	96
	T	2.7	2.7	2.8	2.1	4.2	6.3	17.9
5	NCP	5	5	5	18	20	14	6
	P	371	731	481	350	553	942	704
	NI	47	93	74	32	52	77	223
	T	2.0	4.5	6.7	2.0	4.2	5.6	47.7

Quadro 2.2

Estes resultados revelam que, em geral, os MÉTODOS 1 e 2 requerem um reduzido número de iterações, mas os MÉTODOS 3 e 5 fornecem um melhor controle das quantidades NCP e P. Uma outra conclusão que se pode retirar do estudo computacional é a de que a incorporação do método de Lemke não prejudica significativamente a eficiência do processo de obtenção de uma solução admissível.

Assim, e dada a importância do controle de NCP e P no método enumerativo, recomenda-se a utilização dos MÉTODOS 3 ou 5 no Passo 1. Além disso, os resultados do quadro 2.3 mostram que a escolha desses métodos é particularmente importante se o processo MBRES for usado na geração dos nós.

MÉTODO	Problema	MINVAR			MBRES		
		T	ND	NI	T	ND	NI
1	R1	3.2	11	35	2.7	9	32
	R2	54.1	105	506	109.4	193	1222
	R3	136.9	149	868	65.2	59	518
	R4	72.3	61	404	> 420.0	> 265	> 2698
2	R1	3.2	11	36	2.7	9	33
	R2	53.9	103	515	123.0	215	1335
	R3	64.4	69	430	48.2	39	336
	R4	> 410.0	> 307	> 2851	103.5	73	641
3	R1	3.3	11	36	2.7	9	33
	R2	64.5	109	645	28.5	47	265
	R3	52.6	59	367	14.8	11	120
	R4	49.3	39	294	> 420.0	> 267	> 3019
4	R1	3.3	11	37	2.7	9	34
	R2	38.7	77	369	34.6	61	344
	R3	46.8	39	346	29.5	23	223
	R4	93.5	81	557	142.0	93	838
5	R1	3.3	11	37	2.7	9	34
	R2	29.3	55	292	30.9	53	324
	R3	109.6	127	721	68.2	55	512
	R4	78.1	55	442	53.4	39	385

Quadro 2.3

IV - Escolha do Par de Variáveis Complementares

Para a escolha do par de variáveis complementares no Passo 3, Al-Khayyal [AL87] sugere que se considere aquele par para o qual o produto $z_i w_i$ seja máximo. A nossa experiência computacional confirmou o acerto dessa escolha. Essa experiência também mostrou que o critério que escolhe o par de variáveis (z_i, w_i) para o qual o produto $z_i w_i$ é mínimo tem tendência a gerar nós que não são podados e que pouco alteram o valor de P. Consequentemente a solução é obtida só depois de uma grande busca. Por outro lado, a simples escolha do primeiro par de variáveis complementares cujo produto é diferente de zero torna o método pouco robusto.

V - O Algoritmo Enumerativo Simples

Baseados nas conclusões apresentadas nesta secção, elaborámos um método enumerativo que incorpora as heurísticas mais eficientes. Assim, o algoritmo tem as seguintes características:

(i) O MÉTODO 3 é usado para gerar o nó inicial.

(ii) O algoritmo MBRES é usado para a geração de nós.

(iii) A escolha do nó no Passo 4 é feita a partir do critério que utiliza a quantidade $\text{NCP} + \frac{P}{\rho}$ com $\rho = 10$.

(iv) No Passo 3 é escolhido o par (z_i, w_i) que maximiza o produto $z_i w_i$.

Este algoritmo enumerativo pode ser aplicado sem modificações significativas na resolução do GLCP. Na secção 7 iremos descrever sumariamente uma implementação deste processo para a resolução de LCPs e GLCPs de médias e grandes dimensões e estrutura esparsa.

Para aumentar a eficiência deste algoritmo foi incorporado no Passo 3 um algoritmo heurístico de [JU82] que consiste em minimizar simultaneamente as duas variáveis z_i e w_i ou, pelo menos, em minimizar uma delas sem aumentar o valor da outra. Este processo tem a vantagem de permitir marcar variáveis sem criar novos nós, reduzindo bastante o esforço de pesquisa da solução.

Assim, só se efectuam pivotagens quando um dos seguintes casos ocorrer:

(i) uma das variáveis é nula, considerando-se então a linha associada como linha pivot e para coluna pivot a correspondente a um elemento diferente de zero dessa linha;

(ii) ambas as variáveis são positivas e a coluna pivot tem elementos negativos ou, pelo menos, não positivos com um deles negativo nas linhas associadas a essas variáveis.

Quando tal não for possível, pode acontecer que uma das variáveis satisfaça uma das propriedades:

(P.1) $\bar{q}_i = 0$ e $\bar{a}_{ij} = 0$ para todo o $j \notin \bar{J}$

(P.2) $\bar{q}_i > 0$ e $\bar{a}_{ij} \geq 0$ para todo o $j \notin \bar{J}$

onde $\bar{q}_i$ e $\bar{a}_{ij}$ são os elementos actualizados do quadro contraído correspondente à solução básica corrente. Se uma das variáveis satisfizer (P.1) então pode ser marcada, dado que nunca mais toma valores diferentes de zero. Se uma das variáveis satisfizer (P.2), o seu valor mínimo é positivo e, portanto, a sua complementar vai ser marcada usando o algoritmo MBRES, sem necessidade de, explicitamente, definir novo nó. Se ambas as variáveis satisfizerem (P.2), o ramo é podado.

Se todas as colunas não marcadas tiverem elementos de sinais contrários nas linhas associadas a essas variáveis, usa-se o processo normal já descrito no Passo 3, gerando-se dois novos nós.

Na secção 8 deste capítulo apresentamos alguma experiência computacional com este método na resolução de alguns LCPs de média dimensão com matrizes indefinidas. Tal como seria de esperar, os resultados mostram um aumento significativo do esforço computacional à medida que a dimensão do LCP aumenta. Para minorar essa desvantagem Al-Khayyal sugeriu em [AL87] uma modificação do método de gradiente reduzido que é aplicado em cada nó. Este processo será discutido na próxima secção.

2 - Algoritmo de Gradiente Reduzido Modificado de Al–Khayyal

Como foi referido anteriormente, o LCP é equivalente ao programa quadrático

$$\begin{aligned} &\text{minimizar} \quad g(z, w) = z^T w \\ &\text{sujeito a} \quad (z, w) \in S \end{aligned} \tag{2.12}$$

sendo S o conjunto admissível do LCP definido por (2.2). Devido à definição do LCP é fácil provar, usando os argumentos apresentados em [MU76], que ou o LCP não tem solução ou então existe pelo menos uma solução que é um ponto extremo de S (isto é, uma solução básica admissível). Por isso apenas basta procurar uma solução do LCP entre os pontos extremos de S. Baseado nesta propriedade, Al-Khayyal [AL87] desenvolveu um algoritmo de gradiente reduzido modificado que procura obter um ponto de Kuhn-Tucker da função objectivo do programa (2.12) ou pelo menos um ponto extremo $(\bar{z}, \bar{w})$ de S tal que

$$g(\bar{z}, \bar{w}) \leq g(z, w) \quad \text{para todos os pontos extremos de S adjacentes a } (\bar{z}, \bar{w}).$$

Tal ponto extremo será denotado por LSM (Local Star Minimum) em conformidade com a notação usada por Al-Khayyal.

Seja $(\bar{z}, \bar{w})$ um ponto extremo de S e consideremos a aproximação linear $\varphi(z, w)$ da função $g(z, w)$ em $(\bar{z}, \bar{w})$ dada por

$$\varphi(z, w) = \bar{w}^T z + \bar{z}^T w + (\bar{z}^T \bar{w}) \tag{2.13}$$

Considere-se ainda o programa linear:

$$\begin{aligned} &\text{minimizar} \quad \varphi(z, w) \\ &\text{sujeito a} \quad (z, w) \in S \end{aligned} \tag{2.14}$$

Então verifica-se o seguinte teorema:

Teorema 2.1 – $(\bar{z}, \bar{w})$ é uma solução óptima do programa linear (2.14) se e só se $(\bar{z}, \bar{w})$ é um ponto de Kuhn-Tucker do programa quadrático (2.12).

Demonstração: Considere o programa dual de (2.14)

$$\begin{aligned} &\underset{v \in R^n}{\text{maximizar}} \quad q^T v \\ &\text{sujeito a} \quad \begin{bmatrix} -M^T \\ I_n \end{bmatrix} v \leq \begin{bmatrix} \bar{w} \\ \bar{z} \end{bmatrix} \end{aligned} \tag{2.15}$$

Se $(\bar{z}, \bar{w})$ é uma solução óptima do programa linear (2.14), então o dual tem solução óptima $\bar{v}$ e além disso $(\bar{z}, \bar{w}, \bar{v})$ satisfaz a condição de complementaridade das variáveis de folga. Portanto tem-se

$$\begin{bmatrix} 0 & I_n & M^T \\ I_n & 0 & -I_n \\ -M & I_n & 0 \end{bmatrix} \begin{bmatrix} \bar{z} \\ \bar{w} \\ \bar{v} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ q \end{bmatrix} = \begin{bmatrix} \bar{\alpha} \\ \bar{\beta} \\ 0 \end{bmatrix} \tag{2.16}$$

$$\bar{z}, \bar{w}, \bar{\alpha}, \bar{\beta} \geq 0 \,, \; \bar{z}^T \bar{\alpha} = \bar{\beta}^T \bar{w} = 0$$

Mas (2.16) são exactamente as condições de Kuhn-Tucker do programa quadrático (2.12). Donde $(\bar{z}, \bar{w})$ é um ponto de Kuhn-Tucker do programa (2.12). De igual modo se prova a implicação inversa. ♦

Deste modo, o programa linear (2.14) permite identificar facilmente pontos de Kuhn–Tucker do programa quadrático (2.12).

Se o ponto $(\bar{z}, \bar{w})$ não for ponto de Kuhn-Tucker de (2.12), existirá pelo menos uma aresta incidente em $(\bar{z}, \bar{w})$ tal que o valor de $z^T w$ decresce ao longo desta, sendo esse decréscimo tanto mais rápido quanto mais negativo for o seu custo reduzido. Como interessa determinar pontos extremos LSM de $g(z, w) = z^T w$, teremos que analisar o que acontece a esta função no outro extremo da aresta.

Se designarmos por (dz, dw) a direcção que caracteriza essa aresta, teremos

$$(z^*, w^*) = (\bar{z}, \bar{w}) + \mu_0 \, (dz, dw) \tag{2.17}$$

onde μ_0 é obtido pelo critério do quociente mínimo usado no método simplex. Por outro lado o valor de $z^T w$ ao longo dessa aresta é dado por

$$f(\mu) = dz^T dw \, \mu^2 + (dz^T \bar{z} + dz^T \bar{w}) \, \mu + \bar{z}^T \bar{w} \tag{2.18}$$

onde μ é uma variável com valor compreendido entre zero e μ_0. Se reordenarmos as componentes da direcção (dz, dw) segundo as variáveis básicas e não básicas (x_B, x_N) do ponto ($\bar{z}$, $\bar{w}$), essa direcção terá todas as componentes nulas, com excepção da correspondente à variável não básica x_{ℓ}, que é incrementada ao longo da aresta, e das variáveis básicas. O seu valor é dado por:

$$dx_B = -B^{-1}N_{.\ell} \quad \text{e} \quad dx_N = I_{.\ell} \tag{2.19}$$

onde $N_{.\ell}$ é uma coluna de – M ou de I_n associada à variável não básica x_{ℓ}, consoante essa variável seja z ou w, e $I_{.\ell}$ é a coluna ℓ da matriz identidade.

Então a quantidade $(\bar{z}^T dw + \bar{w}^T dz)$ é exactamente o custo reduzido $\bar{c}_{\ell}$ associado à variável não básica x_{ℓ}. Para encontrarmos direcções descendentes para a função g(z, w) do programa (2.12) temos que analisar o custo reduzido e também o produto $dz^T dw$. Assim podem acontecer os seguintes casos:

(i) Se $\bar{c}_{\ell} = \bar{z}^T dw + \bar{w}^T dz < 0$ e $dz^T dw \leq 0$, então o valor da função $g(z, w) = z^T w$ no ponto extremo adjacente

$$(\bar{z}, \bar{w}) + \mu_0 (dz, dw)$$

vem dado por

$$\bar{z}^T \bar{w} + \bar{c}_{\ell}\, \mu_0 + (dz^T dw)\, \mu_0^2 < \bar{z}^T \bar{w}$$

(ii) Se $\bar{c}_{\ell} < 0$ e $dz^T dw > 0$, então a função $z^T w$ decresce de valor no ponto extremo adjacente se

$$\mu_0 (dz^T dw) < -\bar{c}_{\ell} \tag{2.20}$$

(iii) Se $\bar{c}_{\ell} = 0$ e $dz^T dw < 0$, então a função $z^T w$ decresce de valor no ponto extremo adjacente. Contudo a função não muda de valor se $dz^T dw = 0$ e aumenta de valor se $dz^T dw > 0$. Portanto há redução de valor se a condição (2.20) se verificar.

(iv) Se $\bar{c}_{\ell} > 0$, então $dz^T dw < 0$ para haver decréscimo do valor da função. Além disso tal acontece apenas se a condição (2.20) é verdadeira.

Portanto em todos os casos há a necessidade de verificar se a condição (2.20) é verdadeira. O algoritmo de gradiente reduzido modificado de Al-Khayyal é um processo

que apenas visita pontos extremos do conjunto S procurando em cada iteração uma direcção descendente satisfazendo (2.20).

Assim, e tal como no método simplex, determinam-se os custos reduzidos $\bar{c}_{\ell}$ do programa linear (2.14). Se forem todos não negativos, então obteve-se um ponto de Kuhn-Tucker e o método termina. Se existir pelo menos um $\bar{c}_{\ell} < 0$, as variáveis não básicas são ordenadas por ordem crescente dos custos reduzidos e por essa ordem faz-se uma busca de direcções descendentes, começando, tal como no método simplex, por determinar a coluna actualizada dx_B a partir de (2.19) e calcular

$$\mu_0 = \min_{i \in F} \{ -\frac{x_i}{dx_i} : dx_i < 0 \} \qquad (2.21)$$

com F conjunto dos índices das variáveis básicas.

Se μ_0 não existir, estaremos numa aresta ilimitada que não pode ter pontos onde a função $z^T w$ reduz de valor, pois a função $z^T w$ é limitada inferiormente. Se μ_0 existir, determina-se $dw^T dz$ e verifica-se se a condição (2.20) é verdadeira. Se tal acontecer, então uma direcção descendente foi encontrada e efectua-se uma operação pivotal tornando básica a variável x_{ℓ} por troca com a variável x_s, em que s é a primeira linha em que ocorre o valor μ_0. Passa a ter-se nova solução básica e por isso um novo programa linear de forma (2.14). O processo é então repetido.

Se se não encontrar nenhuma direcção descendente adjacente, então o processo termina com um ponto LSM.

A ocorrência de soluções degeneradas provoca com frequência mudanças de base, sem no entanto mudar de ponto extremo. Por isso, e para que o processo não entre em ciclo, Al-Khayyal [AL87] evita usar o seu algoritmo quando a solução básica é degenerada. Contudo, à medida que a dimensão do LCP vai aumentando, a ocorrência de soluções básicas degeneradas é muito mais frequente. É por isso justificado o uso desse algoritmo de gradiente reduzido modificado mesmo em casos degenerados. Para tal, se um pivot degenerado ocorre ($\mu_0 = 0$), faz-se a escolha da variável a entrar para a base pela regra de Bland [BL77], escolhendo o primeiro coeficiente de custo negativo para determinar essa variável. Esta regra é seguida até que um pivot não degenerado ocorra, passando então a usar-se o processo original descrito por Al-Khayyal. Usando esta extensão, o algoritmo de gradiente reduzido modificado (MRG) pode ser usado em todos os casos, terminando num ponto de Kuhn-Tucker ou num ponto LSM. Este processo

pode ser usado em qualquer nó da árvore. Nesse caso a pesquisa de direcções descendentes é apenas feita para as variáveis não marcadas.

Os passos do algoritmo MRG são apresentados a seguir.

ALGORITMO MRG

Passo 0 – Seja x = (z, w) uma solução básica com Base B, T o conjunto das variáveis não básicas não marcadas e | T | o número de elementos desse conjunto. Faça deg = 0.

Passo 1 – Calcule os custos reduzidos $\bar{c}_j$ ($j \in T$) associados à função do programa linear (2.14). Se $\bar{c}_j \geq 0$ para todo $j \in T$, então x = (z, w) é um ponto de Kuhn-Tucker do programa (2.12) com as variáveis não básicas x_j ($j \notin T$) marcadas e páre. De outro modo vá para Passo 2.

Passo 2 – Se deg = 0, ordene as colunas das variáveis não básicas por ordem crescente dos valores de $\bar{c}_j$. Caso contrário mantenha a ordenação anterior. Faça k = 1 e vá para Passo 3.

Passo 3 – Se k > | T | pára. Um ponto LSM foi encontrado. Caso contrário, seja x_{jk} a variável não básica candidata a entrar na Base. Determine d = (dz, dw) e μ_0 de acordo com as fórmulas (2.19) e (2.21) respectivamente. Se $\mu_0 = 0$ vá para Passo 4. De outro modo ($\mu_0 > 0$), vá para Passo 5.

Passo 4 – Se $\bar{c}_{jk} < 0$ efectue uma operação pivotal, faça deg = 1 e vá para Passo 1. Se $\bar{c}_{jk} \geq 0$ faça k = k + 1 e vá para Passo 3.

Passo 5 – Determine $PD = dz^T dw$. Se $\mu_0 PD < -\bar{c}_{jk}$, efectue uma operação pivotal, faça deg = 0 e vá para Passo 1. De outro modo faça k = k + 1 e vá para Passo 3.

3 - Equivalência entre os Métodos de Al-Khayyal e de Keller

Consideremos um nó qualquer da árvore da figura 2.1 e sejam K e L os conjuntos das variáveis z_i e w_i respectivamente não marcadas no caminho que vai da raiz da árvore até ao nó em discussão. Se $J = K \cap L$, então o método MRG procura resolver o seguinte programa:

$$\begin{aligned} &\text{minimizar} \quad z_J^T w_J \\ &\text{sujeito a} \quad I_n w - Mz + q = 0 \\ &\qquad\qquad z_K \geq 0\,, \; w_L \geq 0 \\ &\qquad\qquad \begin{cases} z_i = 0 & \text{se } i \notin K \\ w_i = 0 & \text{se } i \notin L \end{cases} \end{aligned} \tag{2.22}$$

Sejam $x = (w, z)$, $A = [I_n : -M]$ e

$$H = \begin{bmatrix} 0 & E_n \\ E_n & 0 \end{bmatrix}$$

com E_n uma matriz diagonal de elementos diagonais e_{ii} tais que

$$e_{ii} = \begin{cases} 1 & \text{se } i \in J \\ 0 & \text{se } i \notin J \end{cases}$$

Então o programa quadrático (2.22) pode ser escrito na forma

$$\begin{aligned} &\text{minimizar} \quad g(x) = x^T H x \\ &\text{sujeito a} \quad Ax + q = 0 \\ &\qquad\qquad x \geq 0 \end{aligned} \tag{2.23}$$

Nesta secção iremos apresentar uma versão do método de Keller para a resolução deste programa quadrático (2.23) em que apenas são usadas soluções básicas das restrições $Ax + q = 0$, isto é, pontos extremos do conjunto admissível S do LCP. Em cada iteração podemos escrever:

$$A = [B : N]$$

com B matriz Base associada à solução básica corrente. Se F e T são os conjuntos de índices das colunas de A que pertencem a B e a N respectivamente, então podemos escrever as condições de Kuhn-Tucker do programa quadrático (2.23) na seguinte forma:

$$
\begin{array}{c|c|ccc}
 & & x_F & x_T & \theta \\
\hline
y_F = & p_F & H_{FF} & H_{FT} & -B^T \\
y_T = & p_T & H_{TF} & H_{TT} & -N^T \\
\psi = & q & B & N & 0
\end{array}
\qquad (2.24)
$$

$$x \geq 0\,,\; y \geq 0\,,\; x^T y = 0$$

com ψ um vector de variáveis artificiais ($\psi = 0$) e $p_F = p_T = 0$. Então a solução básica referida anteriormente é obtida do quadro (2.24) por uma operação pivotal principal com pivot

$$\begin{bmatrix} H_{FF} & -B^T \\ B & 0 \end{bmatrix}$$

e tem associada a si um quadro da forma

$$
\begin{array}{c|c|ccc}
 & & x_T & \psi & y_F \\
\hline
y_T = & \bar{p}_T & \bar{H}_{TT} & \bar{H}_{TF} & -\bar{N}^T \\
\theta = & \bar{p}_F & \bar{H}_{FT} & \bar{H}_{FF} & -\bar{B}^T \\
x_F = & \bar{q} & \bar{N} & \bar{B} & 0
\end{array}
$$

Como θ é um vector de variáveis sem restrição de sinal, então $\bar{x}$ é um ponto de Kuhn–Tucker se $\bar{q} \geq 0$ e $\bar{p}_T \geq 0$. Além disso, usando as regras das operações pivotais principais, é fácil de ver que os vectores $\bar{p}_F$, $\bar{p}_T$ e $\bar{q}$ são dados por

$$\bar{q} = -B^{-1} q$$

$$\bar{p}_F = B^{-T} (H_{FF}\, \bar{q})$$

$$\bar{p}_T = H_{TF}\, \bar{q} - N^T \bar{p}_F$$

Devido à definição da matriz H, uma linha j de H_{FF} só será não nula se a variável complementar da variável x_j for também básica. Além disso, se $k \in F$ é a linha dessa variável complementar, então tem-se $h_{jk} = h_{kj} = 1$. Assim, se $x_j = z_i$ então $x_k = w_i$ e portanto $H_{FF}\, \bar{q}$ representa o vector dos coeficientes de custo das variáveis básicas no método MRG. Do mesmo modo a linha j de H_{TF} só será não nula se a variável

complementar de x_j for básica numa linha $k \in F$. Nesse caso $h_{jk} = 1$ e portanto $H_{TF}\,\bar{q}$ é exactamente o coeficiente de custo da função linear usada pelo método MRG. Então se c é o vector dos coeficientes de custo da função linear usada pelo método MRG, os vectores $\bar{q}$, $\bar{p}_F$ e $\bar{p}_T$ do método de Keller satisfazem as igualdades

$$\bar{q} = \bar{x}_F$$
$$\bar{p}_F = B^{-T} c_F = \pi$$
$$\bar{p}_T = c_T - \pi^T N = \bar{c}_T$$

onde $\bar{x}_F$ é o vector das variáveis básicas, π é o vector das variáveis duais e $\bar{c}_T$ é o vector dos custos reduzidos usados pelo método MRG. Portanto a escolha da direcção descendente é feita de modo igual nos dois métodos.

Se uma variável não básica x_r for escolhida no método de Keller para entrar na base e de μ_0 é o valor do quociente mínimo, então o valor da função quadrática após a operação pivotal dupla vem dado por

$$g(x) = g(\bar{x}) + 2\mu_0\,\bar{p}_r + \bar{h}_{rr}\,\mu_0^2$$

com $g(\bar{x})$ o valor da função antes da operação pivotal ser efectuada. Portanto a operação só deve ser efectuada se

$$2\bar{p}_r + \bar{h}_{rr}\,\mu_0 < 0 \qquad (2.25)$$

Mas

$$\bar{h}_{rr} = H_{rF}\,\bar{N}_{.r} - N_{r.}^T\,B^{-T}(H_{Fr} + H_{FF}\,\bar{N}_{.r})$$

Como H é simétrica, então

$$-N_{r.}^T\,B^{-T} = \bar{N}_{.r}^T$$

Portanto

$$\bar{h}_{rr} = H_{rF}\,\bar{N}_{.r} + H_{rF}\,\bar{N}_{.r} + \bar{N}_{.r}^T\,H_{FF}\,\bar{N}_{.r} = 2dw^T dz$$

tendo em conta a definição da matriz H e a definição de d = (dw, dz) apresentada na secção anterior. Como $\bar{p}_r$ é o coeficiente de custo reduzido da função linear usada no método MRG então a condição (2.25) é exactamente igual à condição (2.20) usada no método MRG. Portanto este último algoritmo e a versão do método de Keller que apenas usa pontos extremos de S seguem exactamente o mesmo caminho e são portanto equivalentes.

4 - Método Enumerativo Híbrido para LCP e GLCP

Este algoritmo é essencialmente o método enumerativo simples incorporando o algoritmo MRG introduzido na secção 2. Os passos do método enumerativo híbrido são os seguintes:

MÉTODO ENUMERATIVO HÍBRIDO

Passo 0 – Faça NODE = NNODE = 1, onde NNODE é o número total de nós a ser investigados e NODE o nó corrente. Faça STZ (NODE) = STW (NODE) = 0, onde STZ (NODE) e STW (NODE) são os conjuntos de índices das variáveis z e w marcadas no nó NODE respectivamente.

Passo 1 – Geração do nó 1 (primeira solução admissível)

(i) Aplique o algoritmo MBRES para resolver o programa linear (2.11). Se TERM = 1 e NCP = 0, vá para EXIT. Se TERM = 2 vá para EXIT. Caso contrário (TERM = 1 e NCP > 0) vá para **(ii)**.

(ii) Aplique o algoritmo MRG para encontrar um ponto LSM($\overline{z}$, $\overline{w}$) da função $g(z, w) = z^T w$. Se $g(\overline{z}, \overline{w}) = 0$, faça TERM = 1 e vá para EXIT. Caso contrário faça NCP (NODE) = NCP e vá para o Passo 2.

Passo 2 – Ramificação

Faça NCP = NCP (NODE), STZ = STZ (NODE) e STW = STW (NODE). Escolha o par de variáveis complementares ($\overline{z}_s$, $\overline{w}_s$) no nó NODE tal que

$$\overline{z}_s \, \overline{w}_s = \max \{\overline{z}_{i_k} \, \overline{w}_{i_k} : k = 1,\ldots, NCP\}$$

O nó NODE está explorado. Vá para o Passo 3.

Passo 3 – Geração do nó NNODE + 1

(i) Aplique o algoritmo MBRES para resolver o programa linear

$$\text{minimizar} \quad z_s$$

$$\text{sujeito a} \quad w = q + Mz \ , \ z, w \geq 0$$
$$z_j = 0 \ , \ w_\ell = 0 \ , \ j \in STZ \ , \ \ell \in STW$$

Se TERM = 1 e NCP = 0 vá para EXIT. Se TERM = 2 o nó NODE está explorado, o ramo é podado e vá para o Passo 4. Caso contrário, faça STZ (NODE) = STZ ∪ {s} e vá para **(ii)**.

(ii) Aplique o algoritmo MRG. Se $g(\bar{z}, \bar{w}) = 0$, faça TERM = 1 e vá para EXIT. Caso contrário faça NCP (NODE) = NCP, g(NODE) = $= \sum_{k=1}^{NCP} \bar{w}_{i_k} \bar{z}_{i_k}$ e vá para o Passo 4.

Passo 4 – Geração do nó NNODE + 2
Este é o Passo 3 com z_s, STZ e Passo 4 substituídos por w_s, STW e Passo 5, respectivamente.

Passo 5 – Escolha do nó
Se todos os nós 1,..., NNODE estão explorados, faça TERM = 2 e vá para EXIT. Caso contrário, escolha o nó NODE a partir do seguinte critério:

$$NCP(NODE) + \frac{g(NODE)}{10} =$$
$$= \min\{NCP(j) + \frac{g(j)}{10} : j=1,\ldots, NNODE \text{ e } j \text{ por explorar}\}$$

Vá para o Passo 2.

EXIT – Se TERM = 1, a solução básica complementar $(\bar{z}, \bar{w})$ é uma solução do LCP. Se TERM = 2, o LCP não tem solução.

Este algoritmo pode ser aplicado à resolução do GLCP. A implementação deste processo para a solução de LCPs e GLCPs de média e grande dimensão e estrutura esparsa será discutida na secção 7. Finalmente na secção 8 apresentamos experiência computacional da resolução de LCPs, que atesta a superioridade deste processo em relação ao método enumerativo simples, particularmente quando a dimensão do LCP aumenta.

5 - Aplicação a LCP e GLCP com Estrutura Separada nas Variáveis Complementares

Como veremos em capítulos posteriores aparecem em algumas aplicações LCPs da forma:

$$\begin{bmatrix} w^1 \\ w^2 \end{bmatrix} = \begin{bmatrix} q^1 \\ q^2 \end{bmatrix} + \begin{bmatrix} 0 & M_{12} \\ M_{21} & 0 \end{bmatrix} \begin{bmatrix} z^1 \\ z^2 \end{bmatrix} \qquad (2.26)$$

$$w^1, w^2, z^1, z^2 \geq 0 \qquad w_i z_i = 0 \,, \; i = 1,\ldots, n_1+n_2$$

com $w^i, z^i \in R^{n_i}$, $i = 1, 2$ e $M_{12} \in R^{n_1 x n_2}$ e $M_{21} \in R^{n_2 x n_1}$. Em tais casos existem matrizes de permutação P_1 e P_2 tais que

$$P_1 \, [\, I_n : -M \,] \, P_2 = \begin{bmatrix} E & 0 \\ 0 & D \end{bmatrix} \qquad (2.27)$$

onde $E \in R^{n_1 x \, (n_1+n_2)}$ e $D \in R^{n_2 \, x \, (n_1+n_2)}$ são matrizes tais que E contém as colunas de w^1 e z^2 e D as colunas de w^2 e z^1. Portanto, o LCP tem uma estrutura separada nas variáveis complementares e para qualquer Base B associada a uma solução básica do LCP tem-se

$$P_1 \, B \, P_2 = \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix} \qquad (2.28)$$

com $B_1 \in R^{n_1 x \, n_1}$ e $B_2 \in R^{n_2 x \, n_2}$ matrizes não singulares.

Seguidamente iremos mostrar que o método MRG se simplifica bastante neste caso. Para isso consideremos os seguintes conjuntos

$$J_1 = \{1,\ldots, n_1\} \;\; , \;\; J_2 = \{n_1 + 1,\ldots, n_2\}$$

Seja $x = (x_B, x_N)$ uma solução básica de base B satisfazendo (2.28) e F e T os conjuntos de índices associados às variáveis básicas e não básicas respectivamente. Se

x_ℓ, $\ell \in T$, é uma variável não básica escolhida a entrar na base, então a direcção d é definida por:

$$d_\ell = 1$$

$$d_j = 0 \quad \text{para todo } j \in T - \{\ell\}$$

$$d_{F_1} = \begin{cases} -B_1^{-1} E_{.\ell} & \text{se } \ell \in J_1 \\ 0 & \text{se } \ell \in J_2 \end{cases}$$

$$d_{F_2} = \begin{cases} 0 & \text{se } \ell \in J_1 \\ -B_2^{-1} D_{.\ell} & \text{se } \ell \in J_2 \end{cases}$$

Portanto

$$dw_i = 0 \text{ ou } dz_i = 0 \quad \text{para todo} \quad i = 1,\ldots, n \qquad (2.29)$$

Então a condição (2.20) é verdadeira se e só se é negativo o custo reduzido $\bar{c}_\ell$ da variável não básica associado à função objectivo do programa linear (2.14). Assim, o método MRG reduz-se a um processo tipo simplex em que a direcção descendente é escolhida apenas pelo sinal dos coeficientes de custo reduzido associados à função do programa linear (2.14).

Tendo em conta as conclusões obtidas, podemos apresentar os passos do algoritmo MRG simplificado (SMRG) na seguinte forma:

ALGORITMO SMRG

Passo 0 – Sejam x = (z, w) uma solução básica com Base B e T o conjunto dos índices das variáveis não básicas não marcadas. Faça deg = 0.

Passo 1 – Calcule os custos reduzidos $\bar{c}_j$, $j \in T$, associados à função linear do programa (2.14). Se $\bar{c}_j \geq 0$ para todo $j \in T$, então x é um ponto de Kuhn–Tucker e termine. De outro modo vá para Passo 2.

Passo 2 – Calcule

$$\ell = \begin{cases} \min \{i : \bar{c}_i = \min \bar{c}_j\} & \text{se deg} = 0 \\ \min \{i : \bar{c}_i < 0\} & \text{se deg} = 1 \end{cases} \qquad (2.30)$$

e seja x_ℓ a variável não básica a entrar na base.

Passo 3 – Calcule o quociente mínimo μ_0. Se $\mu_0 = 0$ ($\mu_0 > 0$) faça deg = 1 (deg = 0). Efectue uma operação pivotal e obtenha uma nova solução básica admissível do LCP. Vá para Passo 1.

A incorporação desta versão simplificada do algoritmo MRG no método enumerativo aumenta evidentemente a eficácia do processo. No próximo capítulo apresentamos alguma experiência computacional com LCPs de médias e grandes dimensões que atestam a validade desta afirmação.

É ainda importante notar que o algoritmo SMRG pode também ser usado na resolução de GLCPs com estrutura separada nas variáveis complementares, isto é, GLCPs da forma

$$\begin{bmatrix} w^1 \\ w^2 \\ v \end{bmatrix} = \begin{bmatrix} q^1 \\ q^2 \\ b \end{bmatrix} + \begin{bmatrix} 0 & M_{12} \\ M_{21} & 0 \\ H_1 & H_2 \end{bmatrix} \begin{bmatrix} z^1 \\ z^2 \end{bmatrix} + Ny \qquad (2.31)$$

$$w^i, z^i \geq 0 \ , \quad i = 1, 2 \quad y \geq 0$$

$$z_i \ w_i = 0 \ , \quad i = 1, \ldots, n$$

com $v, b \in R^m$, $H_i \in R^{mxn_i}$, $i = 1, 2$ e $N \in R^{(m+n_1+n_2)xt}$ e t um inteiro positivo, tendo $H = [H_1 : H_2]$ e N também estrutura separada em J_1 e J_2. Este tipo de GLCP aparece muito frequentemente em aplicações da complementaridade a problemas de optimização global.

6 - Aplicação ao GLCP Convexo

Tal como se referiu anteriormente, se a matriz do LCP é PSD, existem vários métodos eficientes para o resolver. No entanto, isso já não se verifica no caso do GLCP convexo. Como vimos, o algoritmo MRG pode ser aplicado para resolver o GLCP. Seguidamente iremos provar que o algoritmo MRG pode ser simplificado neste caso. Consideremos o GLCP (1.21) e seja:

$$g(z, w, y) = \sum_{i=1}^{n} w_i \, z_i$$

Se $(\bar{z}, \bar{w}, \bar{y})$ e (z^*, w^*, y^*) são dois pontos extremos adjacentes de S então tem-se

$$(z^*, w^*, y^*) = (\bar{z}, \bar{w}, \bar{y}) + \mu_0 \, (dz, dw, dy)$$

onde μ_0 é o quociente mínimo do método simplex e d = (dz, dw, dy) é uma direcção admissível que caracteriza a aresta definida pelos dois pontos extremos (z^*, w^*, y^*) e $(\bar{z}, \bar{w}, \bar{y})$. Então:

$$\begin{aligned} g(z^*, w^*, y^*) &= (\bar{z} + \mu_0\, dz)^T (\bar{w} + \mu_0\, dw) = \\ &= \bar{z}^T \bar{w} + \mu_0^2\, dz^T dw + \mu_0(\bar{z}^T dw + \bar{w}^T dz) \\ &= g(\bar{z}, \bar{w}, \bar{y}) + \mu_0\,(\mu_0\, dz^T dw + (\bar{z}^T dw + \bar{w}^T dz)) \end{aligned}$$

Portanto só haverá redução de $g(\bar{z}, \bar{w}, \bar{y})$ se $\mu_0 > 0$ e

$$\mu_0\, dz^T dw + (\bar{z}^T dw + \bar{w}^T dz) < 0 \qquad (2.32)$$

Por outro lado, como ambos os pontos (z^*, w^*, y^*) e $(\bar{z}, \bar{w}, \bar{y})$ são soluções admissíveis do GLCP, tem-se

$$\bar{w} + \mu_0\, dw = q + M(\bar{z} + \mu_0\, dz) + N(\bar{y} + \mu_0\, dy)$$

e portanto

$$dw = M\, dz + N\, dy$$

Donde

$$dz^T dw = dz^T M\, dz + dz^T N\, dy$$

Como M é PSD, então $dz^T M\, dz \geq 0$ e é por isso pouco provável que (2.32) se verifique quando o custo reduzido satisfaz a

$$\bar{c}_{\ell} = (\bar{z}^T dw + \bar{w}^T dz) \geq 0$$

Portanto, é recomendável neste caso procurar apenas direcções descendentes em colunas cujo custo reduzido seja negativo. Com efeito, o tempo que se perde a fazer buscas em outras direcções não é compensado devido à pouca frequência com que direcções descendentes ocorrem para variáveis com coeficientes de custo reduzido não negativos. É de notar que não há garantia neste caso de o algoritmo MRG terminar num ponto de Kuhn-Tucker ou LSM. Contudo, o algoritmo MRG é usado para reduzir o valor da função $z^T w$ e isso é conseguido com esta forma simplificada. No capítulo 7 iremos verificar a importância que esta simplificação do algoritmo MRG tem na eficiência do método enumerativo híbrido para a resolução de GLCPs associados a problemas de optimização global.

7 - Implementação dos Métodos Enumerativos para LCP e GLCP Esparsos

Como foi referido anteriormente, os métodos enumerativos simples e híbrido baseiam-se em operações pivotais e procuram obter uma solução complementar (isto é, uma solução satisfazendo $z_i\, w_i = 0$ para todo $i = 1,\ldots, n$) usando sempre soluções básicas admissíveis. Nesta conformidade, uma implementação de um método enumerativo para a resolução de um LCP ou GLCP esparso deve incorporar as técnicas usadas para implementar o método simplex para programas lineares com esse tipo de estrutura. Esses processos serão discutidos mais adiante. A implementação deve ainda conter algumas estruturas de dados que permitam o funcionamento do método enumerativo. Assim, deverá haver antes de mais uma estrutura de dados que permita representar eficientemente os vários tipos de variáveis que um LCP ou GLCP pode conter. Nessa estrutura as variáveis complementares associadas a matrizes diferentes da matriz identidade (variáveis z) serão numeradas de 1 até n. Se existirem variáveis sem complementar associadas a matrizes diferentes da identidade (variáveis y), elas serão numeradas a seguir, de n+1 até MAC. As variáveis complementares das variáveis z (variáveis w) serão representadas pelos inteiros $MAC + 1,\ldots, MAC + n$. Se existirem ainda variáveis sem complementar associadas à matriz identidade (variáveis v), estas aparecerão no fim, numeradas de $MAC + n + 1$ até $MAC + n + \ell = NV$.

Deste modo, pela simples numeração das variáveis, podemos detectar pares de variáveis complementares (i e MAC + i, para $i \leq n$). Além disso, para não haver necessidade de guardar as matrizes identidade, existe ainda uma quantidade MA, tal que todas as variáveis com numeração $j > MA$ estão associadas a colunas da matriz identidade. Desta forma $j - MA$ é o índice da linha ocupada pelo elemento igual a 1.

Para todos os exemplos de GLCPs apresentados nesta tese, verifica-se MA = MAC, sendo, no caso particular da resolução de um LCP, MAC = MA = n e NV = 2 n. No entanto, preferimos separar as duas quantidades, pois podem ocorrer problemas passíveis de resolução pelo método enumerativo híbrido em que aquela igualdade não se verifique.

Como em cada iteração se usam soluções básicas, há necessidade de guardar a informação da solução básica, assim como das variáveis marcadas e não marcadas. Para isso usa-se um vector X que guarda os valores numéricos das variáveis básicas dessa solução, e um vector de inteiros IVAR de dimensão igual a NV. Essa vector é definido do seguinte modo:

(i) as primeiras m componentes de IVAR contêm inteiros que representam as m variáveis básicas.

(ii) os inteiros correspondentes às variáveis não básicas não marcadas são guardados nas componentes m+1 até m+NC de IVAR.

(iii) as variáveis marcadas estão representadas nas componentes m+NC+1 até NV.

Exemplo : Consideremos um LCP (q, M) com n = 3. A numeração das variáveis $\{z_1, z_2, z_3, w_1, w_2, w_3\}$ será $\{1, 2, 3, 4, 5, 6\}$ respectivamente. Se estivermos perante uma solução básica não complementar $x_B = (z_2, w_1, w_2)$, $x_N = (z_1, z_3, w_3)$, associada a um nó com as variáveis z_1 e w_3 marcadas, virá

$$IVAR = \{2, 4, 5, 3, 1, 6\}$$
$$m = 3$$
$$NC = 1$$
$$NV = 6$$

Se X = (2.5, 0.5, 1.25) então a solução básica é

$$z_2 = 2.5$$
$$w_1 = 0.5$$
$$w_2 = 1.25$$
$$z_1 = z_3 = w_3 = 0$$

Para se conhecer a localização de uma variável em IVAR, consideramos um vector IPT, tal que IPT(i) indica a componente de IVAR que guarda a variável i. Estes vectores IVAR e IPT são actualizados em cada iteração do método enumerativo e são bastante úteis para analisar se uma variável é básica, não básica ou marcada, detectar soluções básicas complementares e definir a função objectivo do algoritmo MRG.

Em determinadas situações são ainda utilizados vectores auxiliares que permitem o acesso rápido a uma dada informação. Por exemplo, no método MBRES usa-se uma lista ligada para mais rapidamente serem percorridas as variáveis não básicas com complementar também não básica ou marcada.

Em cada iteração, para além da informação anterior, é necessário conhecer o valor numérico associado à solução básica x_B, a parte básica dx_B da direcção admissível d e ainda, em alguns casos, os custos reduzidos associados a uma função objectivo. Como

$$x_B = -B^{-1} q \quad , \quad dx_B = B^{-1} N_{.\ell}$$

x_B e dx_B podem ser obtidos a partir da solução de dois sistemas de equações lineares com matriz B. Para se obterem os custos reduzidos das variáveis não básicas tem que se calcular $y = c_B^T B^{-1}$, onde c_B é o vector dos coeficientes de custo da função objectivo correspondentes às variáveis básicas. Esse vector y pode assim ser obtido a partir da solução do sistema $B^T y = c_B$.

Para resolver estes sistemas, usa-se a decomposição LU da matriz Base B. Tal decomposição é apresentada na forma [RE82]

$$L^{-1} B = P_1 U P_2$$

onde P_1 e P_2 são matrizes de permutação armazenadas em dois vectores de dimensão m, U é uma matriz esparsa armazenada por linhas e L^{-1} é armazenada na forma produto do tipo

$$L^{-1} = E_k E_{k-1} \ldots E_1$$

Nesta representação E_i é uma matriz elementar, que apenas difere da matriz identidade num elemento não diagonal que é aliás o único elemento real a ser armazenado na estrutura que armazena a forma produto.

Deste modo, para resolver o sistema

$$Bx = h$$

tem-se

$$L P_1 U P_2 x = h$$

ou seja

$$P_1 U P_2 x = L^{-1} h = E_k E_{k-1} \ldots E_1 h = h^k$$

Portanto, para obter x, resolve-se o sistema triangular superior

$$U P_2 x = P_1^T h^k$$

Por outro lado, para a resolução de

$$B^T y = c_B$$

tem-se

$$P_2^T U^T P_1^T L^T y = c_B$$

Assim resolve-se o sistema triangular inferior

$$U^T P_1^T v = P_2 c_B$$

e calcula-se

$$y = L^{-T} v = E_1^T \ldots E_{k-1}^T E_k^T v$$

usando as matrizes elementares armazenadas na forma produto.

Esta decomposição LU é actualizada em cada iteração de acordo com o esquema sugerido por [RE82], provocando enchimentos em U e acrescentando novos elementos a L^{-1}. Tal como em outros processos deste tipo, são efectuadas reinversões periódicas a partir do quadro inicial. Essa reinversão consta de uma Fase de Análise na qual a matriz é permutada de modo a provocar poucos enchimentos e de uma Fase de Factorização onde os pivots obtidos na primeira Fase podem ser alterados para manter a estabilidade do processo

Na Fase da Análise é usada uma técnica semelhante à desenvolvida em [HERA71, HERA72]. No processo usual de Hellerman-Rarick, a matriz é reordenada na forma triangular inferior por blocos (LBTF) de modo a que, por pivotagem, só sejam gerados acima da diagonal elementos diferentes de zero em poucas colunas (normalmente designadas por "spikes").

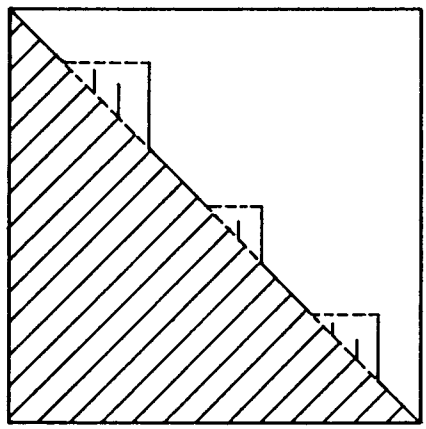


Figura 2.2 - Forma LBTF de Hellerman-Rarick

Como alguns dos pivots podem ser nulos, esta reordenação nem sempre permite uma factorização estável [ERGRLEPO85] e há necessidade de, na Fase de Factorização, prever a troca de "spikes". Por outro lado, como a matriz U está ordenada por linhas e como na Fase de Factorização é usada a escolha limite de pivot [RE82], preferimos a reordenação que transforma a matriz na forma triangular superior por blocos (UBTF) (ver figura 2.3).

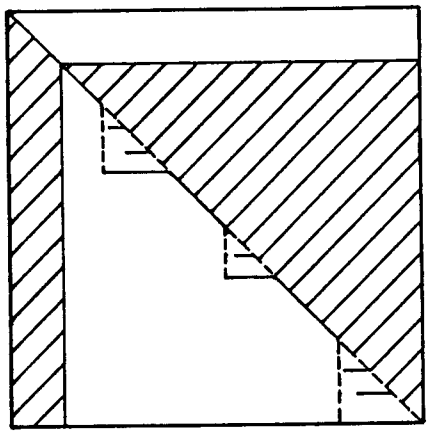


Figura 2.3 - Forma UBTF

Assim, a escolha limite de pivot só faz a troca de linhas "spikes" não alterando muito a estrutura prevista na Fase de Análise. Além disso, pelo modo como a actualização provoca enchimentos na decomposição LU, a reordenação UBTF, da figura 2.3, tem vantagens relativamente à LBTF da figura 2.2, pois começa com L^{-1} mais esparso e com U mais cheio na zona em que tem mais tendência a encher.

Para manter a estabilidade é necessário que os pivots, encontrados ao longo de todo o processo, satisfaçam a certas condições. Para isso, introduziram-se duas tolerâncias, a Tolerância Absoluta TOL2 e Tolerância Relativa TOLR, exigindo-se que o valor absoluto do pivot seja maior que a tolerância absoluta e superior ao produto do maior valor absoluto da coluna abaixo da diagonal pela tolerância relativa (escolha limite de pivot).

Se o valor absoluto do pivot for inferior à tolerância absoluta durante a actualização da inversa, faz-se a reinversão da matriz B a partir do quadro original. Se durante a reinversão se voltar a confirmar essa propriedade, então a matriz Base é singular, não se chegando a calcular a sua inversa. Deste modo o algoritmo termina podendo pensar-se em executá-lo de novo com outros valores para as tolerâncias.

Se o valor absoluto do pivot não satisfizer a condição de limite, então efectua-se uma troca de linhas de forma a que o elemento com maior valor absoluto passe a ser o pivot. Com esta troca, a decomposição fica mais estável, apesar de ocorrerem mais enchimentos. Para controlar esses novos enchimentos, é conveniente usar um valor de TOLR não muito grande (TOLR = 0.1 é normalmente recomendado [TO72]).

As reinversões são efectuadas devido a razões de estabilidade e de eficiência. Assim, como critério de estabilidade é feita uma análise da solução, usando um critério sugerido por [TO72], sempre que o elemento máximo em valor absoluto da decomposição LU é grande ($> 10^3$). Tal critério consiste em comparar o resíduo r_i de

$$r = h - Bx$$

com a quantidade

$$s_i = \sum_j | b_{ij} x_j |$$

onde x é a solução obtida pela decomposição actual. Sempre que $| r_i | > TOL\ s_i$ (TOL é normalmente igual a 10^{-6}), a decomposição é rejeitada e é efectuada uma reinversão para obter uma nova decomposição da mesma matriz B.

Como critério de eficiência, obriga-se a que uma nova reinversão seja executada sempre que se tenham executado pelo menos NACT actualizações (onde NACT é um parâmetro fornecido pelo utilizador, normalmente igual a 40) e que o número de elementos da última actualização ultrapasse o dobro dos elementos da última reinversão. Por outro lado, como os elementos de U podem ter as suas linhas dispostas desordenadamente e com elementos nulos entre elas, é necessário algum espaço livre para que sejam efectuadas actualizações sem que sejam necessárias muitas compressões da estrutura de dados que armazena U. Assim, na prática é conveniente também executar uma reinversão quando 2/3 do espaço livre da última reinversão já esteja ocupado.

Quando um nó é gerado, é necessário guardar, em memória auxiliar num ficheiro de acesso directo, alguma informação acerca do nó corrente e da solução básica associada. Essa informação inclui os vectores IVAR e IPT referidos anteriormente. Assim, no Passo 5 do método enumerativo híbrido, sempre que a escolha recai num nó diferente do último explorado, toda essa informação é lida do registo que corresponde a esse nó, sendo necessário repor, por reinversão, a decomposição LU correspondente àquela solução básica.

Como os nós são percorridos segundo uma certa ordem (normalmente por ordem crescente da quantidade NCP + P / ρ) usa-se uma lista ligada que consiste num vector NNV e numa variável ponteiro LPI. A variável LPI aponta para a componente de NNV associada ao primeiro nó segundo a ordenação pretendida e NNV (i) aponta para a componente de NNV associada ao nó seguinte segundo essa ordenação. O último elemento da lista é identificado por NNV (i) = 0. Os índices desse vector NNV

representam os números dos registos do ficheiro directo que guardam a informação necessária para identificar o nó. Depois de explorado o nó, fica livre um desses registos. Assim, para que o ficheiro directo não fique muito extenso, o mesmo vector NNV, com outra variável ponteiro LPL, é usado para ligar os registos cuja informação já não é necessária. Apenas no caso de LPL ser igual a zero (não existem registos livres) é que é incrementado o número de registos utilizados. Com este tipo de implementação, o processo de escolha do nó a ser explorado é extremamente simples e recorre sempre a um ficheiro com o menor número de registos possível.

Finalmente é bastante importante em problemas de grande dimensão ordenar as colunas não básicas por ordem crescente dos custos reduzidos, sempre que o método MRG faz busca de direcções descendentes. Para isso, na primeira iteração, usa-se o algoritmo "Quicksort" [WI76], dado que é o mais rápido quando o vector está desordenado. Nas iterações seguintes, como essa ordenação não é geralmente muito alterada, usa-se o algoritmo "linear", que é mais eficiente para estes casos [WI76].

8 - Experiência Computacional

Nesta secção apresenta-se alguma da experiência computacional usada para testar os métodos enumerativos na resolução de LCPs em que a matriz M é indefinida. Experiência computacional com LCPs e GLCPs com M$\in$ NSD ou com estrutura de variáveis separadas será descrita em capítulos posteriores.

Os resultados foram obtidos usando um computador CYBER CDC 720 da Universidade do Porto. Os problemas–teste foram gerados de modo a que o LCP tenha pelo menos uma solução complementar. Os elementos da matriz M e as componentes do vector q foram gerados aleatoriamente de acordo com uma técnica descrita em [RASH 84], que é explicada a seguir:

a) Os elementos não nulos da matriz $M \in R^{nxn}$ são números inteiros uniformemente distribuídos entre – 50 e +50.

b) Os valores de q são gerados de modo a que o LCP tenha pelo menos uma solução (z, w), com as primeiras k = n / 5 componentes de z iguais a zero, e as restantes n – k componentes iguais a um inteiro uniformemente distribuído entre 2 e 5, inclusive. Os valores de w são tais que:

$$w_i = \begin{cases} 2 & \text{se } M_{.i}\, z > 0 \\ 0 & \text{se } M_{.i}\, z = 0 \\ 1 & \text{se } M_{.i}\, z < 0 \end{cases} \quad \text{para } i = 1, \ldots, k$$

e $w_i = 0$ para $i = k+1, \ldots, n$. O vector q é então univocamente determinado por $q = w - Mz$

Usando esta técnica, foram gerados dois tipos de problemas esparsos. No primeiro grupo de problemas foi introduzida uma quantidade NZ que representa o número de elementos não nulos da matriz M, sendo esses elementos não nulos distribuídos aleatoriamente pela matriz. Foram gerados doze problemas deste tipo, identificados pela letra R. A dimensão n e número de não zeros NZ de cada um desses problemas são apresentados no quadro 2.4.

Problema	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
n	40	40	40	40	50	100	100	100	100	150	200	300
NZ	69	135	215	263	340	158	263	442	871	405	682	747

Quadro 2.4

No segundo grupo de problemas usámos as técnicas indicadas em [PAROS88], para gerar matrizes tridiagonais e pentadiagonais, identificadas respectivamente pelas letras T e P. A solução é gerada de tal modo que, aleatoriamente, uma das componentes w_i ou z_i seja nula e a outra tome um valor inteiro aleatório no intervalo [0, 2]. As dimensões n desses problemas são as indicadas no quadro 2.5, tendo-se gerado, para alguns deles, dois valores diferentes para as componentes do vector q.

Problema	T1	T2	T3	T4	T5	T6	T7	T8	T9	P1	P2	P3
n	100	200	200	300	300	400	400	500	500	100	200	300

Quadro 2.5

É importante notar que os 24 problemas apresentados têm matrizes não simétricas indefinidas com cerca de metade dos elementos gerados negativos, e que o vector q também tem um número significativo de valores negativos. Além disso, o método de Lemke [LE68], com vector $e = (1, \ldots, 1)^T$, termina em aresta ilimitada em todos estes 24 problemas-teste.

Nas experiências procurámos testar a eficiência do método enumerativo híbrido, que é identificado por ALKHYBEN, e do método enumerativo simples SIMEN. Os quadros

2.6 e 2.7 contêm a experiência computacional com estas duas versões do método enumerativo na resolução dos problemas-teste descritos anteriormente.

Problema	SIMEN			ALKHYBEN		
	T	ND	NI	T	ND	NI
R1	2.7	9	33	0.8	1	22
R2	28.5	47	265	4.8	3	69
R3	14.8	11	120	11.1	9	114
R4	> 420.0	> 267	> 3019	14.9	5	114
R5	141.8	63	570	29.8	5	148
R6	5.6	7	61	6.8	7	51
R7	38.3	51	280	21.9	5	130
R8	> 760.0	> 79	2397	57.9	1	218
R9		*		160.2	3	264
R10	> 1260.0	> 969	5941	77.0	17	241
R11		*		193.7	3	490
R12		*		405.5	19	481

Quadro 2.6

Problema	SIMEN			ALKHYBEN		
	T	ND	NI	T	ND	NI
T1	7.9	11	68	13.8	5	78
T2	28.1	31	137	42.8	3	128
T3	29.6	27	149	49.9	3	150
T4	64.3	33	227	164.6	7	221
T5	116.0	71	353	291.9	9	262
T6	132.1	59	336	299.6	7	292
T7	278.1	137	656	317.0	11	330
T8	> 1460.0	> 530	> 2592	694.8	17	385
T9		*		1080.0	13	441
P1	139.5	119	872	37.6	9	204
P2	> 400.0	> 113	>913	172.5	7	320
P3		*		997.4	37	797

Quadro 2.7

Tal como anteriormente, nestes quadros os parâmetros T, ND e NI representam respectivamente o tempo de CPU em segundos, o número de nós e o número de operações pivotais. O símbolo > significa que o método enumerativo foi incapaz de

encontrar a solução complementar durante o tempo T que se lhe segue. O símbolo * significa que o problema-teste correspondente não chegou a ser resolvido, pois a sua execução seria bastante ineficiente.

Os quadros 2.6 e 2.7 revelam que a versão simples do método enumerativo, descrita na secção 1.V, tem um bom comportamento para problemas bastante esparsos. Este facto é particularmente evidente para a maior parte dos problemas-teste com matrizes tridiagonais, onde a versão simples se comporta melhor que a híbrida. Contudo, quando a dimensão e a densidade do problema aumentam, esta versão simples piora. A versão híbrida é eficiente para todos os testes e não é tão sensível a um aumento de densidade e dimensão da matriz do LCP.

É também importante notar que em todos os problemas-teste o algoritmo ALKHYBEN requer muito poucos nós para encontrar uma solução complementar. O número de iterações é normalmente menor do que na versão simples, embora os tempos sejam piores em alguns casos. Este aumento significativo do tempo de execução é explicado pela necessidade que o método MRG tem em pesquisar mais do que uma direcção em cada iteração do processo para esses problemas mais difíceis.

CAPÍTULO 3

PROBLEMAS LINEARES COMPLEMENTARES CÔNCAVOS

1 - Propriedades do LCP Côncavo

Como é perfeitamente conhecido, uma matriz M quadrada de ordem n é Negativa Semi-Definida ($M \in NSD$) se e só se $x^T Mx \leq 0$ para todo $x \in R^n$. Neste capítulo estudaremos a resolução do LCP quando a sua matriz M pertence a essa classe. Iremos ainda considerar uma subclasse NSD_+ das matrizes NSD em que os elementos não diagonais são não negativos. Essa classe é portanto definida de acordo com a seguinte equivalência:

$$M \in NSD_+ \Leftrightarrow M \in NSD \quad e \quad m_{ij} \geq 0 \quad \text{para todo } i \neq j$$

Começamos por recordar os conceitos de Transformada Principal e Complemento de Schur de uma matriz que terão um papel fundamental no estabelecimento dos principais resultados deste capítulo. Seja M uma matriz quadrada de ordem n e F um subconjunto de $\{1,\ldots, n\}$. Então existe uma matriz de permutação P tal que

$$P^T M P = \begin{bmatrix} M_{FF} & M_{FT} \\ M_{TF} & M_{TT} \end{bmatrix}$$

com $T = \{1,\ldots, n\} - F$. Se M_{FF} é não singular, a transformada principal de M com pivot M_{FF} é a matriz $\bar{M}$ dada por

$$\bar{M} = \begin{bmatrix} M_{FF}^{-1} & -M_{FF}^{-1} M_{FT} \\ M_{TF} M_{FF}^{-1} & (M|M_{FF}) \end{bmatrix} \tag{3.1}$$

com

$$(M \mid M_{FF}) = M_{TT} - M_{TF}\ M_{FF}^{-1}\ M_{FT} \tag{3.2}$$

Esta última matriz é denominada Complemento de Schur de M_{FF} em M.

Usando demonstrações semelhantes às apresentadas em [MU88], é possível estabelecer um número importante de propriedades que estão agregadas no seguinte lema:

Lema 3.1: Se M é uma matriz quadrada de ordem n, então

i) $M \in NSD(NSD_+) \Rightarrow M_{FF} \in NSD(NSD_+)$ para qualquer $F \subset \{1,\ldots, n\}$

ii) $M \in NSD(NSD_+) \Rightarrow m_{ii} \leq 0$ para todo $i = 1,\ldots, n$

iii) $M \in NSD$ e $m_{ii} = 0 \Rightarrow m_{ij} = -m_{ji}$ para quaisquer $j \neq i$

$M \in NSD_+$ e $m_{ii} = 0 \Rightarrow m_{ij} = m_{ji} = 0$ para quaisquer $j \neq i$

iv) $M \in NSD(NSD_+) \Rightarrow P^T MP \in NSD(NSD_+)$ onde P é uma matriz de permutação

v) se $\bar{M}$ é a matriz dada por (3.1), então

$$M \in NSD \Rightarrow \bar{M} \in NSD$$

vi) se M_{FF} é não singular e $(M \mid M_{FF})$ é a matriz dada por (3.2), então

$$M \in NSD(NSD_+) \Rightarrow (M \mid M_{FF}) \in NSD(NSD_+)$$

Como consequência do lema anterior podemos estabelecer a seguinte propriedade:

Lema 3.2 - Se $M \in NSD_+$ e não singular, então $M^{-1} \leq 0$

Demonstração: Iremos demonstrar o teorema usando o método de indução sobre a ordem n da matriz M. Se n = 1, então $M = [m_{11}] < 0$ e $M^{-1} = [1/m_{11}] < 0$. Suponhamos agora que o lema é válido para todas as matrizes $M \in NSD_+$ de ordem $k \leq n - 1$ e demonstremos que também é verdadeiro quando M tem ordem n. Podemos escrever

$$M = \begin{bmatrix} M_1 & b \\ a^T & m_{nn} \end{bmatrix}$$

onde a e b são vectores de ordem n – 1 e M_1 é uma matriz quadrada de ordem n – 1. Se M_1 é singular, existe um conjunto $F \subseteq \{1,\ldots, n - 1\}$ tal que M_{FF} é a submatriz não singular de M_1 de maior ordem. Então pelo Lema 3.1 tem-se

$$P^T\bar{M}P = \begin{bmatrix} M_{FF}^{-1} & -M_{FF}^{-1}M_{FT} & -M_{FF}^{-1}b_F \\ M_{TF}\,M_{FF}^{-1} & 0 & 0 \\ a_F^T M_{FF}^{-1} & 0 & \bar{m}_{nn} \end{bmatrix} \qquad (3.3)$$

com $T = \{1,\ldots, n-1\} - F$. Portanto pela Fórmula de Schur [CRHA69] tem-se:

$$\det(M) = \det(M_{FF}) \det \begin{bmatrix} 0 & 0 \\ 0 & \bar{m}_{nn} \end{bmatrix} = 0$$

Mas isto é impossível, porque M é não singular. Então M_1 é não singular e podemos escrever em (3.3) $M_{FF} = M_1$ e $T = \emptyset$. Além disso pela Fórmula de Schur

$$\bar{m}_{nn} = \frac{\det(M)}{\det(M_1)} \neq 0$$

e o sinal de $\bar{m}_{nn}$ é negativo pelo lema 3.1. Então a inversa de M pode ser obtida da matriz (3.3) por uma operação pivotal com pivot $\bar{m}_{nn}$ e tem-se

$$P^T M^{-1} P = \begin{bmatrix} M_1^{-1} + \frac{1}{\bar{m}_{nn}} M_1^{-1} b.a^T M_1^{-1} & -\frac{1}{\bar{m}_{nn}} M_1^{-1} b \\ -\frac{1}{\bar{m}_{nn}} a^T M_1^{-1} & \frac{1}{\bar{m}_{nn}} \end{bmatrix}$$

Como $M \in NSD_+$ então $a \geq 0, b \geq 0$ e $\bar{m}_{nn} < 0$. Além disso $M_1^{-1} \leq 0$ pela hipótese de indução e portanto $M^{-1} \leq 0$. ♦

Estamos agora em condições de estabelecer o primeiro resultado referente à existência de solução do LCP côncavo [JUFA91a].

Teorema 3.1 – **(i)** se $M \in NSD_+$ e $q < 0$, então o LCP é não admissível

(ii) $Q \cap NSD_+ = \emptyset$

Demonstração: Basta demonstrar (i), pois (ii) é uma consequência de (i) e da definição de matriz Q apresentada no capítulo 1. Dois casos podem ocorrer e são discutidos a seguir:

a) Se M é não singular, o LCP é equivalente a

$$z = -M^{-1}q + M^{-1}w$$

$$z \geq 0,\ w \geq 0,\ z^T w = 0$$

Como $0 \neq M^{-1} \leq 0$, $w \geq 0$ e $q < 0$, então $M^{-1}q > 0$ e $M^{-1}w \leq 0$. Portanto o LCP é não admissível.

b) Se M é singular, existe um conjunto $F \subset \{1,\ldots, n\}$ e uma matriz de permutação P tal que

$$P^T M P = \begin{bmatrix} M_{FF} & b_F & M_{FT} \\ a_F^T & m_{rr} & a_T^T \\ M_{TF} & b_T & M_{TT} \end{bmatrix}$$

M_{FF} é não singular, $T = \{1,\ldots, n\} - (F \cup \{r\})$ e $\bar{m}_{rr} = m_{rr} - a_F^T M_{FF}^{-1} b_F = 0$. Como M_{FF} é não singular, então efectuando uma operação pivotal principal com pivot M_{FF} nas restrições $w = q + Mz$ obtém-se o seguinte sistema equivalente:

$$\begin{bmatrix} z_F \\ w_r \\ w_T \end{bmatrix} = \begin{bmatrix} \bar{q}_F \\ \bar{q}_r \\ \bar{q}_T \end{bmatrix} + \begin{bmatrix} M_{FF}^{-1} & -M_{FF}^{-1}b_F & -M_{FF}^{-1}M_{FT} \\ a_F^T M_{FF}^{-1} & 0 & 0 \\ M_{TF}M_{FF}^{-1} & 0 & \bar{M}_{TT} \end{bmatrix} \begin{bmatrix} w_F \\ z_r \\ z_T \end{bmatrix} \qquad (3.4)$$

Como $a \geq 0$, $q < 0$ e $M_{FF}^{-1} \leq 0$, então

$$\bar{q}_r = q_r - a_F^T M_{FF}^{-1} q_F < 0, \quad a_F^T M_{FF}^{-1} \leq 0$$

Portanto não existem vectores $z \geq 0$ e $w \geq 0$ satisfazendo o sistema (3.4) e o LCP é não admissível. ♦

Recentemente Cottle [CO90] estabeleceu que o segundo resultado (ii) do último teorema é também válido para qualquer matriz NSD. Dada a sua simplicidade, apresenta-se a seguir a respectiva demonstração:

Teorema 3.2 – $NSD \cap Q = \emptyset$

Demonstração: Se $M \in Q$, então existe um vector $z > 0$, tal que $Mz > 0$ [LE70]. Portanto existe um vector $z > 0$ tal que $z^T Mz > 0$ e $M \notin NSD$. ♦

Se $M \in Q$, então existe um vector $z > 0$, tal que $Mz > 0$ [LE70]. Portanto existe um vector $z > 0$ tal que $z^T Mz > 0$ e $M \notin NSD$. ♦

Seguidamente iremos investigar a intersecção $Q_0 \cap NSD$. Essa intersecção é não vazia, pois segundo [JU90]:

Teorema 3.3 – Se $n = 2$ então $NSD \subset Q_0$

Contudo esse resultado não é verdadeiro para matrizes de ordens superiores a 2, pois podem existir matrizes NSD que não sejam Q_0. Mais adiante mostraremos que $NSD_+ \subset Q_0$.

Em relação ao desenvolvimento de metodologias para resolver o LCP côncavo começaremos por mostrar que o método de Lemke não pode ser usado para resolver o LCP quando $M \in NSD_+$.

Teorema 3.4 – Se $M \in NSD_+$, o método de Lemke termina sempre em aresta ilimitada.

Demonstração: Tal como foi referido anteriormente este algoritmo considera um vector $p \geq 0$, com $p_i > 0$ para todo i tal que $q_i < 0$, de modo a obter o sistema adicional

$$w = q + z_0 p + Mz$$

com z_0 uma variável artificial. No passo inicial é efectuada uma operação pivotal que troca a variável não básica z_0 com a variável básica w_r de tal modo que $q + \bar{z}_0 p \geq 0$ para certo valor positivo $\bar{z}_0$ de z_0. Então z_r é a variável não básica a entrar na base na iteração a seguir. Actualizando a coluna que lhe está associada, tem-se

$$\bar{m}_{rr} = -\frac{m_{rr}}{p_r} \geq 0$$

$$\bar{m}_{ir} = m_{ir} - \frac{m_{rr}}{p_r} p_i \geq 0, \text{ para todo } i \neq r$$

Assim foi encontrada uma aresta ilimitada e isso demonstra o teorema. ♦

Este resultado não é verdadeiro para matrizes NSD gerais. Com efeito é possível determinar um ponto de Kuhn-Tucker de um programa quadrático côncavo resolvendo um LCP côncavo com o método de Lemke [MU88]. Contudo a nossa experiência computacional mostra que em geral o método de Lemke não consegue resolver LCP côncavos.

Um outro assunto de extrema importância teórica e algorítmica diz respeito à complexidade do LCP côncavo. É possível demonstrar que o LCP côncavo é um problema NP-completo [MU88]. Neste capítulo iremos ainda mostrar que se $M \in NSD_+$ então o LCP côncavo pode ser resolvido em tempo polinomial.

2 - Método Enumerativo para LCP Côncavo

Se o LCP é côncavo, o algoritmo de gradiente reduzido modificado de Al–Khayyal pode ser extremamente simplificado. Tal como vimos anteriormente este método procura em cada nó encontrar um ponto LSM ou um ponto de Kuhn-Tucker da função

$$g(z, w) = \sum_{i \in J} z_i w_i$$

com J conjunto dos pares de variáveis complementares não marcadas. Sejam $(\bar{z}, \bar{w})$ e (z^*, w^*) dois pontos extremos adjacentes do conjunto admissível S do LCP. Então

$$(z^*, w^*) = (\bar{z}, \bar{w}) + \mu_0 (dz, dw)$$

com μ_0 o quociente mínimo do método simplex e $d = (dz, dw)$ a direcção que define a aresta que une esses dois pontos extremos. Mas

$$g(z^*, w^*) = (\bar{z} + \mu_0 dz)^T (\bar{w} + \mu_0 dw) =$$

$$= \bar{z}^T \bar{w} + \mu_0 (\bar{w}^T dz + \bar{z}^T dw) + \mu_0^2 dz^T dw$$

Como $(z^*, w^*) \in S$, então

$$\bar{w} + \mu_0 dw = q + M(\bar{z} + \mu_0 dz) = q + M\bar{z} + \mu_0 M dz$$

e portanto

$$dw = M dz$$

Como $M \in NSD$, então

$$dz^T dw = dz^T M dz \leq 0$$

Assim tem-se

$$g(z^*, w^*) - (\bar{z}, \bar{w}) \leq \mu_0 (\bar{w}^T dz + \bar{z}^T dw)$$

Mas $\bar{w}^T dz + \bar{z}^T dw$ é o custo reduzido associado à função linear $\varphi(z, w) = \bar{z}^T w + \bar{w}^T z$ da variável não básica que é incrementada para gerar a aresta definida por d. Portanto cada custo reduzido negativo corresponde a uma direcção descendente e o algoritmo MRG pode ser substituído pelo método SMRG. É ainda de notar que o algoritmo SMRG também pode ser usado para resolver GLCPs côncavos da forma

$$\begin{aligned} w &= q + Mz \\ v &= b + Hz \\ v, w, z &\geq 0, \; z^T w = 0 \end{aligned} \qquad (3.5)$$

Esse tipo de GLCP irá aparecer em alguns problemas de optimização global.

3 - Método Polinomial para um Caso Especial do LCP Côncavo

Consideremos o conjunto de restrições

$$w = q + Mz \qquad (3.6)$$

do LCP. Se apenas operações pivotais principais forem usadas para obter soluções básicas de (3.6), então a condição de complementaridade $z^T w = 0$ é sempre satisfeita, obtendo-se em cada iteração um sistema da forma

$$\bar{w} = \bar{q} + \bar{M}\bar{z} \qquad (3.7)$$

onde $(\bar{w}, \bar{z})$ é uma permutação de (w, z) e $\bar{M}$ é uma transformada principal de M dada por (3.1). Além disso $\bar{M} \in$ NSD pelo lema 3.1. Como apenas operações pivotais principais são efectuadas, a solução do LCP ocorre quando se encontrar um sistema da forma (3.7) com $\bar{q} \geq 0$.

O processo a discutir nesta secção é um método pivotal principal. Seguidamente apresentamos os seus passos para depois estabelecer a polinomialidade do processo quando $M \in NSD_+$.

ALGORITMO POLINOMIAL [JUFA91a]

Passo 0 – Seja $I = \{i : q_i < 0\}$, $J = \{1,\ldots, n\} - I$, $F = \emptyset$ e $\bar{M} = M$

Passo 1 – Seja

$$r = \min\{i \in I\}$$

e considere-se o conjunto

$$L_0 = \{j \in J - F : \bar{m}_{rj} > 0\}$$

Se $L_0 \neq \emptyset$ vá para Passo 3. De outro modo vá para o Passo 2.

Passo 2 – Seja $I = I - \{r\}$. Se $I \neq \emptyset$ vá para o Passo 1. De outro modo, o LCP é não admissível e páre.

Passo 3 – Seja $s \in L_0$. Faça $F = F \cup \{s\}$ e efectue uma operação pivotal principal com o pivot $\bar{m}_{ss}$. Vá para o Passo 4.

Passo 4 – Considere $I = \{i : \bar{q}_i < 0\}$ e $J = \{1,\ldots, n\} - I$. Se $I = \emptyset$, o vector z definido por

$$z_i = \begin{cases} \bar{q}_i & \text{se } i \in F \\ 0 & \text{se } i \notin F \end{cases}$$

é solução do LCP. Caso contrário, vá para o Passo 1.

Seguidamente iremos estabelecer que se $M \in NSD_+$ então este algoritmo termina no máximo em n operações pivotais. Nesse sentido começaremos por provar que a operação pivotal no Passo 3 é sempre possível. Como os pivots $\bar{m}_{ss}$ pertencem ao Complemento de Schur $\bar{M}_{TT}$ da matriz M_{FF} em M, então, pelo Lema 3.1, $\bar{m}_{ss} \leq 0$. Mas, se $m_{ss} = 0$, todos os elementos dessa linha e coluna são iguais a zero ($M_{Ts} = \bar{M}_{sT} = 0$), o que tornaria vazio o conjunto L_0 do Passo 2. Logo $m_{ss} < 0$.

Provemos agora por indução que $\bar{q}_F \geq 0$ em cada iteração. Como apenas operações pivotais principais são efectuadas, em cada iteração do algoritmo podemos considerar um quadro da forma

	1	w_F	z_T
$z_F =$	$\bar{q}_F$	$\bar{M}_{FF}$	$\bar{M}_{FT}$
$w_T =$	$\bar{q}_T$	$\bar{M}_{TF}$	$\bar{M}_{TT}$

(3.8)

com

$$\bar{q}_F = -M_{FF}^{-1} q_F \quad \text{e} \quad \bar{q}_T = q_T + M_{TF}\,\bar{q}_F$$

Suponhamos então que $\bar{q}_F \geq 0$ numa iteração k. Além disso $M_{FF}^{-1} \leq 0$, pelo Lema 3.2, e $M_{Fs} \geq 0$ e portanto $\bar{M}_{Fs} = -M_{FF}^{-1} M_{Fs} \geq 0$. Se efectuarmos uma operação pivotal com pivot $\bar{m}_{ss}$, então $\bar{q}_s \geq 0$, $\bar{m}_{ss} < 0$. Como F é acrescido da componente s, as novas componentes do transformado de q_F vêm dadas por

$$\bar{q}_F - \frac{\bar{q}_s}{\bar{m}_{ss}} . \bar{M}_{Fs} \quad \text{e} \quad -\frac{\bar{q}_s}{\bar{m}_{ss}}$$

e são portanto todas não negativas.

Assim, quando uma variável z_i, $i \in T$, se torna básica (i torna-se um elemento de F), i nunca mais pode pertencer a I. Isto mostra que, na pior das hipóteses, são necessárias n operações pivotais principais e o algoritmo é polinomial.

Para estabelecer a convergência do método, basta agora demonstrar que, caso não se encontre um pivot em L = J – F, o LCP é não admissível. Para isso, vamos escrever o quadro (3.8) na seguinte forma:

	1	w_F	z_I	z_L
$z_F =$	$\bar{q}_F$	$\bar{M}_{FF}$	$\bar{M}_{FI}$	$\bar{M}_{FL}$
$w_I =$	$\bar{q}_I$	$\bar{M}_{IF}$	$\bar{M}_{II}$	$\bar{M}_{IL}$
$w_L =$	$\bar{q}_L$	$\bar{M}_{LF}$	$\bar{M}_{LI}$	$\bar{M}_{LL}$

onde

$$\bar{q}_F \geq 0,\ \bar{q}_I < 0,\ \bar{q}_L \geq 0,\ \bar{M}_{FF} = M_{FF}^{-1} \leq 0$$

$$\left[\, \bar{M}_{FI} : \bar{M}_{FL} \,\right] = -M_{FF}^{-1} \left[\, M_{FI} : M_{FL} \,\right] \geq 0$$

$$\begin{bmatrix} \bar{M}_{IF} \\ \bar{M}_{LF} \end{bmatrix} = \begin{bmatrix} M_{IF} \\ M_{LF} \end{bmatrix} M_{FF}^{-1} \leq 0$$

e

$$\begin{bmatrix} \bar{M}_{II} & \bar{M}_{IL} \\ \bar{M}_{LI} & \bar{M}_{LL} \end{bmatrix} = (M \mid M_{FF}) \in NSD_+$$

Se o algoritmo não puder efectuar uma pivotagem no Passo 3, então $\bar{M}_{IL} = 0$. Como $\bar{M}_{IF} \leq 0$, $\bar{q}_I \leq 0$ e $\bar{M}_{II} \in NSD_+$, estamos na situação do teorema 3.1 e o LCP é não admissível.

Assim, o algoritmo é polinomial, e ou termina com uma solução do LCP ou com a informação de que o LCP é não admissível. Portanto, $NSD_+ \subset Q_0$. Esta inclusão é obviamente estrita, pois existe um grande número de subclasses das matrizes Q_0 que não são NSD_+ [MU88].

4 - Implementação do Algoritmo Polinomial para LCP de Estrutura Esparsa

Na implementação deste algoritmo para o LCP côncavo de grande dimensão, é importante notar que toda a informação necessária pode ser encontrada resolvendo sistemas com a matriz M_{FF}. Assim, da definição de operação pivotal principal apresentada no capítulo 1, o seguinte processo permite determinar o vector $\bar{q}$:

Resolver	$M_{FF}\,\bar{q}_F = -\,q_F$
Calcular	$\bar{q}_T = \bar{q}_T + M_{FT}\,\bar{q}_F$

Além disso para determinar $\bar{m}_{rj}$ no Passo 1, tem-se:

Resolver	$M_{FF}^T\,\alpha = M_{rF}^T$
Calcular	$\bar{m}_{rj} = m_{rj} - M_{Fj}^T\,\alpha$

Como $M_{FF} \in NSD_+$, então pode resolver-se o sistema sem usar escolha parcial de pivot [FUPL81]. Se a matriz M é simétrica, então a implementação descrita em [JUPI89b] pode ser usada para resolver o LCP côncavo de grandes dimensões com estrutura esparsa através deste método polinomial.

Nessa implementação é usada uma Fase de Análise, na qual é escolhida uma permutação principal da matriz M, de tal modo que não ocorram muitos enchimentos no processo de decomposição LDL^T. Para esse fim é conveniente usar um algoritmo de

reordenação das linhas e colunas de M. O algoritmo de grau mínimo [GELI81, cap. 5] é um bom processo e deve por isso ser incorporado nessa implementação.

Na Fase de Análise, é guardado espaço para a estrutura de dados das matrizes L e D de qualquer submatriz M_{FF} de M usada pelo algoritmo pivotal principal.

Como apenas ocorrem em F modificações de um elemento então é vantajoso actualizar a decomposição LDL^T de uma matriz M_{FF} a partir da decomposição da iteração anterior. Tal é conseguido usando um processo baseado numa versão para matrizes esparsas [LA85] do algoritmo de Bennett [BE65]. Para uma melhor descrição do processo de implementação sugerimos [JUPI88]. A nossa experiência computacional mostrou ser conveniente escolher os índices s e r no método polinomial de acordo com a ordenação da Fase de Análise. Com esta escolha não só a actualização da decomposição se faz de um modo mais eficiente como também há normalmente um menor número de passagens pelo Passo 2.

Apesar da nossa experiência computacional se reportar apenas ao caso de M ser simétrica, o processo de implementação discutido pode ser estendido de um modo simples para uma matriz não simétrica. Para isso a Fase de Análise deve ser aplicada à estrutura da matriz $M + M^T$ e decomposições LDU das matrizes M_{FF} devem ser usadas.

5 - Experiência Computacional

Nesta secção iremos apresentar alguma experiência computacional de resolução de LCPs côncavos com matrizes NSD gerais e NSD_+. Em relação ao primeiro caso testámos a eficiência do método enumerativo híbrido na resolução de LCPs côncavos. Duas versões deste algoritmo foram usadas que diferem entre si apenas na versão do método de gradiente reduzido modificado. Assim o algoritmo HYBEN incorpora a versão SMRG, enquanto que como anteriormente o método ALKHYBEN usa o algoritmo MRG na sua forma usual.

No quadro 3.1 apresenta-se a experiência computacional na resolução de onze problemas-teste de diferentes fontes, efectuada num CDC CYBER 180-830 da Universidade do Porto. Esses problemas são apresentados a seguir:

i) As matrizes dos problemas-teste PT1, PT2 e PT3 são tridiagonais simétricas, com os elementos gerados aleatoriamente, sendo os elementos da diagonal determinados de modo a que a matriz seja NSD.

ii) As matrizes dos problemas-teste PT4 e PT5 são simétricas e foram geradas aleatoriamente por um esquema semelhante ao descrito em [YATO85].

iii) Os problemas-teste PT6 e PT7 constituem as condições de Kuhn-Tucker dos problemas quadráticos de estrutura semelhante aos apresentados em [GO86]. Contudo a função objectivo foi multiplicada por –1 e os limites superiores nas variáveis foram substituídos por uma restrição do tipo $e^T x \leq \tau$ (onde $e^T = (1,\ldots, 1)$).

iv) Os problemas-teste PT8, PT9, PT10 e PT11 são as condições de Kuhn-Tucker do problema quadrático, cuja função objectivo é definida pelas matrizes usadas em PT4 (PT8 e PT9), PT1 (PT10) e PT5 (PT11). As restrições foram geradas de um modo semelhante ao usado para gerar as restrições lineares dos LCPs indefinidos referidos no capítulo anterior. O vector q foi gerado segundo a técnica de Ramarao e Shetty [RASH84] referida no capítulo anterior.

Para testar melhor a eficiência do método enumerativo resolvemos todos os problemas pelos métodos de Keller [JUFA90], Lemke e por uma modificação do método de Lemke [MU88] capaz de obter um ponto de Kuhn-Tucker de um programa quadrático com solução óptima. O método de Lemke foi incapaz de resolver qualquer dos LCPs, tendo a terminação em aresta ilimitada ocorrido sempre. Daí nos referirmos, na coluna LEMKE do quadro 3.1, à versão modificada do método de Lemke.

No quadro 3.1 usaram-se as seguintes notações:

n = dimensão do LCP = ordem da matriz M

ℓ = número das variáveis do problema quadrático

m = número de restrições do problema quadrático

NI = número de operações pivotais

ND = número de nós

T = tempo de CPU em segundos

UN = o método foi incapaz de obter solução do LCP

C = o método entrou em ciclo

PT	n	ℓ	m		HYBEN	ALKHYBEN	LEMKE	KELLER
1	500	500	—	NI ND T	160 53 27.8	160 53 52.5	UN	UN
2	1000	1000	—	NI ND T	325 103 92.1	325 103 263.6	UN	UN
3	2000	2000	—	NI ND T	699 243 422.4	699 243 1989.	UN	UN
4	500	500	—	NI ND T	189 65 35.9	189 65 71.	UN	UN
5	600	600	—	NI ND T	531 87 115.9	531 87 298.7	UN	UN
6	1199	799	400	NI ND T	325 23 109.	287 13 280.4	19 — 5.2	4 — 0.5
7	1499	999	500	NI ND T	124 1 42.8	124 1 152.8	C	9 — 1.2
8	520	500	20	NI ND T	136 7 33.3	136 7 87.1	7 — 1.3	30 — 1.16
9	550	500	50	NI ND T	1355 23 427.2	1354 23 1124.	307 — 38.7	668 — 72.2
10	550	500	50	NI ND T	3613 25 1091.	3709 23 2492.	C	821 — 88.7
11	650	600	50	NI ND T	1905 45 729.2	1905 45 2064.	683 — 120.9	764 — 104.

Quadro 3.1

Os resultados computacionais permitem concluir que o método enumerativo híbrido HYBEN resolve todos os problemas-teste de um modo eficiente. O facto de usar a versão simples SMRG do algoritmo de gradiente reduzido modificado no método enumerativo híbrido faz aumentar drasticamente a sua eficiência. Com efeito, apesar dos números de iterações e nós dos algoritmos HYBEN e ALKHYBEN serem sensivelmente os mesmos, os tempos são muitos inferiores para a versão HYBEN. Isto é explicado pela não necessidade de pesquisa de direcções descendentes no método HYBEN. Com efeito qualquer direcção associada a uma variável não básica com custo reduzido negativo gera uma direcção descendente.

Os métodos de Keller e a versão modificada de Lemke mostraram-se incapazes de resolver LCPs côncavos em todos os casos. No entanto são bastante mais eficientes que o método HYBEN quando conseguem obter essa solução. Tais resultados são de certo modo esperados. Com efeito esses algoritmos não necessitam de qualquer pesquisa em árvore para obter uma solução. É ainda interessante notar que no método de Lemke ocorreu um ciclo em dois dos problemas-teste. Isso mostra que, contrariamente aos LCPs convexos [CHG79], a regra de Bland não é suficiente para evitar ciclos no método de Lemke quando a matriz do LCP é NSD.

Seguidamente apresenta-se alguma experiência computacional com o algoritmo polinomial (POLYN) na solução de LCPs côncavos com matrizes NSD_+ simétricas. Para gerar matrizes NSD_+, considerámos problemas com matrizes $PD \cap Z$ e multiplicámos todos os seus elementos por (–1). Assim nos problemas-teste PT14 e PT15 foram consideradas matrizes $PD \cap Z$ associadas à resolução de equações diferenciais parciais com o método das diferenças finitas. Essa matrizes têm uma estrutura tridiagonal por blocos em que cada bloco diagonal é uma matriz tridiagonal. Essas matrizes tridiagonais foram usadas na geração dos problemas-teste PT12 e PT13. Finalmente nos problemas PT16 e PT17 as matrizes foram geradas de um modo aleatório em que a estrutura da matriz é gerada de um modo semelhante à da técnica exposta no capítulo anterior. Tal como anteriormente o vector q foi gerado segundo a técnica de Ramarao e Shetty [RASH84]. Para testar melhor a eficiência do método polinomial resolvemos alguns dos problemas-teste com o método enumerativo híbrido HYBEN.

No quadro 3.2, que apresenta os resultados das experiências, além dos parâmetros n, ND, NI e T usados anteriormente, usam-se também o parâmetro BVZ, que indica o número de variáveis básicas z_i da solução obtida pelo método enumerativo HYBEN, e o parâmetro NEG que indica o número de componentes negativas do vector q.

Para cada problema-teste, foram gerados dois problemas que diferem apenas no vector q. Isso permite estudar o comportamento do método para diferentes valores de NEG. Essa quantidade normalmente influencia o comportamento dos métodos pivotais

principais [JUPI88]. Os resultados do quadro 3.2 mostram que, neste caso, isso também acontece para o algoritmo POLYN, mas o aumento de NI com NEG revela-se pequeno.

Pela análise do quadro, pode concluir-se que o algoritmo POLYN tem um bom comportamento e é muito mais eficiente do que o método HYBEN. Além disso o algoritmo obtém uma solução do LCP com mais variáveis básicas z_i do que o algoritmo HYBEN. Esta conclusão baseia-se no facto de BVZ ser sempre inferior a NI do método POLYN e NI indicar o número dessas variáveis z_i na solução obtida por este último algoritmo. Neste momento ainda não encontrámos uma explicação para tal comportamento.

			POLYN		HYBEN			
PT	n	NEG	NI	T	ND	NI	T	BVZ
12	500	130	484	19.6	83	589	67.4	233
		166	497	20.1	105	623	80.4	344
13	1000	305	987	81.9	195	1349	289.	528
		336	992	90.6	189	1129	261.	656
14	512	223	466	41.6	5	388	80.5	300
		251	503	57.6	5	570	268.	421
15	1008	480	987	245.4				
		498	996	284.8				
16	500	202	424	39.2				
		227	434	47.5				
17	1000	258	772	73.6				
		337	781	78.6				

Quadro 3.2

Como conclusão final deste estudo computacional, podemos afirmar que o método enumerativo híbrido HYBEN é um método capaz de resolver eficientemente LCPs côncavos de grandes dimensões. Os algoritmos directos de KELLER e LEMKE são bastante mais eficientes que HYBEN, mas em geral são incapazes de obter uma solução do LCP côncavo. Se a matriz pertencer à classe NSD_+, então o método polinomial desenvolvido neste capítulo é altamente recomendável. Parece-nos uma área de investigação importante a procura de outras classes de LCPs côncavos para as quais seja possível desenvolver métodos directos ou iterativos para a resolução do LCP.

CAPÍTULO 4

RESOLUÇÃO DO PROBLEMA LINEAR COMPLEMENTAR COM LIMITES SUPERIORES (BLCP)

1 - Propriedades do BLCP

Tal como referimos no capítulo 1, o BLCP consiste em encontrar vectores $z \in R^n$ e $w \in R^n$ tais que

$$w = q + M z$$

$$\left.\begin{array}{l} a_i \le z_i \le b_i \\ z_i = a_i \Rightarrow w_i \ge 0 \\ z_i = b_i \Rightarrow w_i \le 0 \\ a_i < z_i < b_i \Rightarrow w_i = 0 \end{array}\right\} \quad i = 1,\ldots, n \qquad (4.1)$$

onde $q \in R^n$ e $M \in R^{nxn}$ são dados e $-\infty \le a_i < b_i \le +\infty$ para todo i = 1,..., n. Se M é simétrica então o BLCP representa as condições de Kuhn-Tucker do programa quadrático [GIMUWR81]

$$\begin{array}{ll} \text{minimizar} & q^T z + \frac{1}{2} z^T M z \\ \text{sujeito a} & z \in S \end{array} \qquad (4.2)$$

com

$$S = \{ z \in R^n : a_i \le z_i \le b_i , i = 1,\ldots, n \} \qquad (4.3)$$

Em geral o BLCP é equivalente ao seguinte Problema de Desigualdades Variacionais Lineares [HAPA90]

Encontrar $\bar{z} \in S$ tal que

$$(q + M \bar{z})^T (z - \bar{z}) \ge 0 \text{ para todo } z \in S \qquad (4.4)$$

com S o conjunto definido por (4.3). Como este problema tem solução no caso de S ser um conjunto convexo e compacto [HAPA90], então podemos enunciar o seguinte resultado:

Teorema 4.1 – O BLCP tem solução se todos os limites inferiores e superiores forem finitos.

Neste caso particular, se considerarmos sem perda de generalidade que todos os limites inferiores são iguais a zero, então o BLCP é equivalente ao LCP [AH83]

$$\begin{bmatrix} v \\ y \end{bmatrix} = \begin{bmatrix} q \\ b \end{bmatrix} + \begin{bmatrix} M & I_n \\ -I_n & 0 \end{bmatrix} \begin{bmatrix} z \\ \beta \end{bmatrix} \qquad (4.5)$$

$$y, \beta, z, v \geq 0 \quad , \quad z^T v = y^T \beta = 0$$

Consideremos agora um BLCP na sua forma mais geral e suponhamos que, de igual modo, todos os limites inferiores finitos são nulos. Então podemos considerar uma partição de $\{1,\ldots, n\}$ nos seguintes conjuntos:

$$\begin{aligned} G &= \{i : a_i = -\infty \text{ e } b_i = +\infty\} \\ J_1 &= \{i : a_i = 0 \text{ e } b_i < +\infty\} \\ J_2 &= \{i : a_i = 0 \text{ e } b_i = +\infty\} \\ J_3 &= \{i : a_i = -\infty \text{ e } b_i < +\infty\} \end{aligned} \qquad (4.6)$$

Suponhamos inicialmente que $G = \emptyset$. Nessas condições, considerando a equivalência referida anteriormente e que para cada $i \in J_3$ podemos fazer uma mudança de variável

$$y_i = b_i - z_i \quad , \quad 0 \leq y_i \leq +\infty \quad , \quad i \in J_3$$

então o BLCP é equivalente a um LCP da forma

$$\begin{bmatrix} v_{J_1} \\ v_{J_2} \\ \beta_{J_3} \\ y_{J_1} \end{bmatrix} = \begin{bmatrix} q_{J_1} + M_{J_1 J_3} b_{J_3} \\ q_{J_2} + M_{J_2 J_3} b_{J_3} \\ q_{J_3} + M_{J_3 J_3} b_{J_3} \\ b_{J_1} \end{bmatrix} + \begin{bmatrix} M_{J_1 J_1} & M_{J_1 J_2} & -M_{J_1 J_3} & I_{|J_1|} \\ M_{J_2 J_1} & M_{J_2 J_2} & -M_{J_2 J_3} & 0 \\ -M_{J_3 J_1} & -M_{J_3 J_2} & M_{J_3 J_3} & 0 \\ -I_{|J_1|} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_{J_1} \\ z_{J_2} \\ y_{J_3} \\ \beta_{J_1} \end{bmatrix} \qquad (4.7)$$

$$v, z, y, \beta \geq 0 \qquad z^T v = \beta^T y = 0$$

com $z, v \in R^{|J_1 \cup J_2|}$, $\beta, y \in R^{|J_1 \cup J_3|}$, e $|J|$ o número de elementos do conjunto J. Além disso tem-se

Teorema 4.2 – Se o BLCP é convexo (côncavo), então o LCP (4.7) é convexo (côncavo).

Demonstração: Podemos escrever a matriz do LCP (4.7) na seguinte forma:

$$H = \left[\begin{array}{cc|c} A & -B & \\ & & -E^T \\ -C & D & \\ \hline & E & 0 \end{array}\right] \qquad (4.8)$$

com

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

Para qualquer vector $x = (x^1, x^2, x^3)$ tem-se

$$x^T H x = \begin{bmatrix} x^1 \\ x^2 \end{bmatrix}^T \begin{bmatrix} A & -B \\ -C & D \end{bmatrix} \begin{bmatrix} x^1 \\ x^2 \end{bmatrix}$$

$$= (x^1)^T A x^1 - (x^1)^T B x^2 - (x^2)^T C x^1 - (x^2)^T D x^2$$

$$= (x^1)^T A x^1 + (x^1)^T B (-x^2) + (-x^2)^T C x^1 + (x^2)^T D (-x^2)$$

$$= \begin{bmatrix} x^1 \\ -x^2 \end{bmatrix}^T \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x^1 \\ -x^2 \end{bmatrix}$$

Donde a matriz do LCP (4.7) é PSD (NSD) se e só se o BLCP é convexo (côncavo). ♦

Consideremos agora o caso em que o conjunto G definido por

$$G = \{ i : a_i = -\infty \text{ e } b_i = +\infty \}$$

é não vazio, isto é, existem variáveis sem restrição de sinal. Então as variáveis w_i, $i \in G$, têm que ser nulas em qualquer solução do BLCP. Essas variáveis podem ser eliminadas usando um algoritmo descrito em [JU82]. Esse processo usa apenas soluções básicas do BLCP e operações pivotais principais. Uma solução básica para o BLCP é definida de modo semelhante ao usado no LCP, mas as variáveis não básicas podem tomar valores iguais a um dos limites inferiores ou superiores. Considerando os conjuntos G, J_i ($i = 1, 2, 3$) definidos por (4.6), uma solução básica inicial para o BLCP vem dada por

$$z_i = 0 \ , \ i \in G \cup J_1 \cup J_2 \ , \ z_i = b_i \ , \ i \in J_3$$

$$w_i = q_i + \sum_{j \in J_3} m_{ij} b_j = \eta_i \qquad i = 1,\ldots, n$$

Se |G| é o número de elementos do conjunto G e M é uma matriz PSD ou NSD, então o seguinte processo pode ser usado para obter uma solução básica com $w_G = 0$ ou mostrar que o BLCP não tem solução:

ALGORITMO FASE 1

Passo 0 - Seja $\bar{M} = M$, $\bar{q} = \eta$ e $\bar{G} = G$.

Passo 1 - Se $\bar{G} = \emptyset$ termine. De outro modo seja

$$r = \min \{i \in \bar{G}\}$$

Passo 2 - Se $\bar{m}_{rr} = 0$ vá para Passo 3. De outro modo efectue uma operação pivotal principal com pivot $\bar{m}_{rr}$. Faça $\bar{G} = \bar{G} - \{r\}$ e vá para Passo 1.

Passo 3 - Se existir um $s \in \bar{G} \cup J_1 \cup J_2 \cup J_3$ tal que $\bar{m}_{rs} \neq 0$ vá para Passo 4. De outro modo calcule $\bar{q}_r$. Se $\bar{q}_r \neq 0$, então o BLCP não tem solução. De outro modo a linha r é redundante, faça $\bar{G} = \bar{G} - \{r\}$ e vá para Passo 1.

Passo 4 - Efectue uma operação pivotal dupla com pivots $\bar{m}_{rs}$ e $\bar{m}_{sr} = -\bar{m}_{rs}$. Faça

$$\bar{G} = \begin{cases} \bar{G} - \{s, r\} & \text{se } s \in \bar{G} \\ \bar{G} - \{r\} & \text{se } s \notin \bar{G} \end{cases}$$

Vá para Passo 1.

Como é discutido em [JU82], é sempre possível usar este algoritmo quando $M \in$ PSD ou NSD. Com efeito $\bar{M}$ é PSD ou NSD em cada iteração, pois apenas operações pivotais principais são usadas [MU88]. Além disso se $\bar{m}_{rr} = 0$ e $\bar{m}_{rs} \neq 0$, então $\bar{m}_{sr} = -\bar{m}_{rs}$ [MU88] e portanto a operação dupla pode ser efectuada no Passo 4.

Após a aplicação deste algoritmo, as variáveis z_i , $i \in G$, são básicas ou não básicas numa coluna nula e as variáveis w_i , $i \in G$, são não básicas ou básicas numa linha nula. Como as variáveis z_i , $i \in G$, não têm restrição de sinal, então não são importantes na resolução do BLCP. Nessa medida podemos sem perda de generalidade assumir $G = \emptyset$

nos BLCPs convexo e côncavo. Neste capítulo iremos desenvolver algoritmos para BLCPs convexos e côncavos e para terminar discutiremos um possível algoritmo capaz de resolver o BLCP em todos os casos.

2 - Extensão do Método de Lemke

2.1 – Descrição do Algoritmo

Tal como no método de Lemke discutido no capítulo 1, a extensão [JUFA91c] a propor nesta secção considera uma variável artificial $0 \leq z_0 \leq +\infty$ com vector p associado de modo a obter um sistema alargado da forma

$$w = q + z_0 p + Mz \tag{4.9}$$

O vector p é definido de modo a que este sistema tenha uma solução satisfazendo

$$\left.\begin{array}{l} z_0 \geq 0 \\ a_i \leq z_i \leq b_i \\ z_i = a_i \Rightarrow w_i \geq 0 \\ z_i = b_i \Rightarrow w_i \leq 0 \\ a_i < z_i < b_i \Rightarrow w_i = 0 \end{array}\right\} \quad i = 1,\ldots, n \tag{4.10}$$

Se $G = \emptyset$, então os conjuntos J_i $(i = 1, 2, 3)$ definidos por (4.6) formam uma partição de $\{1,\ldots, n\}$. Para satisfazer as condições (4.10) é necessário que as componentes p_i do vector p satisfaçam as seguintes condições:

$$\begin{array}{l} i \in J_1 \cup J_2 \Rightarrow \left\{\begin{array}{l} p_i \geq 0 \\ q_i < 0 \Rightarrow p_i > 0 \end{array}\right. \\ \\ i \in J_3 \Rightarrow \left\{\begin{array}{l} p_i \leq 0 \\ q_i > 0 \Rightarrow p_i < 0 \end{array}\right. \end{array} \tag{4.11}$$

Considerando um vector p nessas condições, calcule-se

$$\mu_1 = \left\{\begin{array}{ll} 0 & \text{se } \{i \in J_1 \cup J_2 : q_i < 0\} = \emptyset \\ \max \{-\frac{q_i}{p_i} : i \in J_1 \cup J_2 \text{ e } q_i < 0\} & \end{array}\right.$$

$$\mu_2 = \begin{cases} 0 & \text{se } \{i \in J_3 : q_i > 0\} = \emptyset \\ \max \{-\frac{q_i}{p_i} : i \in J_3 \text{ e } q_i > 0\} \end{cases}$$

e seja

$$\mu_0 = \max \{\mu_1, \mu_2\}$$

Se r é a linha onde o valor μ_0 é atingido, então efectuando uma operação pivotal com pivot p_r obtém-se um quadro da forma:

		$w_{\bar{F}}$	$z_{\bar{T}_1}$	$z_{\bar{T}_2}$
$z_F =$	g_F	$A_{F\bar{F}}$	$A_{F\bar{F}_1}$	$A_{F\bar{T}_2}$
$w_{T_1} =$	g_{T_1}	$A_{T_1\bar{F}}$	$A_{T_1\bar{T}_1}$	$A_{T_1\bar{T}_2}$
$w_{T_2} =$	g_{T_2}	$A_{T_1\bar{F}}$	$A_{T_2\bar{T}_1}$	$A_{T_2\bar{T}_2}$

(4.12)

com

$$F = \{i : z_i \text{ é básica }, \ i = 0,1,\ldots,n\}$$

$$T_1 = \{i : w_i \text{ é básica e } z_i = 0\}$$

$$T_2 = \{i : w_i \text{ é básica e } z_i = b_i\}$$

$$\bar{F} = F - \{0\} \cup \{r\} \qquad (4.13)$$

$$\bar{T}_1 = \begin{cases} T_1 & \text{se } z_r = b_r \\ T_1 \cup \{r\} & \text{se } z_r = 0 \end{cases}$$

$$\bar{T}_2 = \begin{cases} T_2 & \text{se } z_r = 0 \\ T_2 \cup \{r\} & \text{se } z_r = b_r \end{cases}$$

Além disso, a solução básica dada pelo quadro (4.12) satisfaz as seguintes propriedades:

(i) Existe um par (z_r, w_r) de variáveis não básicas complementares;

(ii) Para cada $i \neq r$

z_i básica $\Leftrightarrow$ w_i não básica

z_i não básica $\Leftrightarrow$ w_i básica;

(iii) z_0 é básica;

(iv) Se η é o vector definido por

$$\eta_i = g_i + \sum_{j \in \bar{T}_2} a_{ij}\, b_j \qquad i = 1,\ldots, n \tag{4.14}$$

então

$$a_i \leq \eta_i \leq b_i \ , i \in F \ , \eta_{T_1} \geq 0 \ \text{ e } \ \eta_{T_2} \leq 0 \tag{4.15}$$

Como no método de Lemke, chamamos **quase-complementar** a uma solução básica com estas características. A extensão do método de Lemke contém dois critérios de escolha da variável não básica de entrada e da variável básica que sai da base que são apresentados a seguir.

Critério (C1) para a variável de entrada

A variável de entrada é a complementar da variável que se torna não básica ou que mudou de um limite para o outro na iteração anterior.

Critério (C2) para a variável de saída

Seja η o vector definido por (4.14), T_1 e T_2 os conjuntos dados por (4.13). Além disso consideremos a partição do conjunto F definido por

$$\begin{aligned} F_1 &= \{i \in F \ : \ a_i = 0 \ \text{ e } \ b_i < +\infty\} \\ F_2 &= \{i \in F \ : \ a_i = 0 \ \text{ e } \ b_i = +\infty\} \\ F_3 &= \{i \in F \ : \ a_i = -\infty \ \text{ e } \ b_i < +\infty\} \end{aligned} \tag{4.16}$$

Seja $A_{.r}$ a coluna do quadro (4.12) associada à variável de entrada escolhida pelo critério (C1). Para cada um dos seguintes conjuntos K apresentados atrás, considerem-se os conjuntos K_i, i = 1, 2, definidos por:

$$K_1 = \{i \in K : a_{ir} > 0\} \qquad K_2 = \{i \in K : a_{ir} < 0\}$$

Então há quatro casos possíveis que se apresentam a seguir.

Caso 1 - w_r é a variável de entrada e $z_r = 0$ $(r \in \bar{F})$

Calcule

$$\mu_0 = \begin{cases} +\infty \quad \text{se } J_4 = F_{11} \cup F_{12} \cup F_{22} \cup F_{31} \cup T_{12} \cup T_{21} = \emptyset \\ \min \left\{ \dfrac{\alpha_i - \eta_i}{a_{ir}} : i \in J_4 \right\} \end{cases} \qquad (4.17)$$

com

$$\alpha_i = \begin{cases} b_i \quad \text{se } i \in F_{11} \cup F_{31} \\ 0 \quad \text{se } i \in J_4 - (F_{11} \cup F_{31}) \end{cases}$$

Se $\mu_0 = +\infty$, a terminação em aresta ilimitada ocorre. Caso contrário, faça $T_1 = T_1 \cup \{r\}$, $\bar{F} = \bar{F} - \{r\}$ e seja t o indíce da variável básica para a qual μ_0 é atingido. Então:

(i) se $t \in F_{11} \cup F_{31}$, z_t passa a não básica com valor b_t e faça

$$F = F - \{t\} \text{ e } \bar{T}_2 = \bar{T}_2 \cup \{t\}$$

(ii) se $t \in F_{12} \cup F_{22}$, z_t passa a não básica com valor 0 e $F = F - \{t\}$. Se $t = 0$, páre: a solução do BLCP foi encontrada. Caso contrário, faça $\bar{T}_1 = \bar{T}_1 \cup \{t\}$.

(iii) se $t \in T_{12}$, w_t passa a não básica. Faça

$$T_1 = T_1 - \{t\} \text{ e } \bar{F} = \bar{F} \cup \{t\}$$

(iv) se $t \in T_{21}$, w_t passa a não básica. Faça

$$T_2 = T_2 - \{t\} \text{ e } \bar{F} = \bar{F} \cup \{t\}$$

Caso 2 - w_r é a variável de entrada e $z_r = b_r$ $(r \in \bar{F})$

Calcule

$$\mu_0 = \begin{cases} -\infty \quad \text{se } J_5 = F_{11} \cup F_{12} \cup F_{21} \cup F_{32} \cup T_{11} \cup T_{22} = \emptyset \\ \max \left\{ \dfrac{\alpha_i - \eta_i}{a_{ir}} : i \in J_5 \right\} \end{cases} \qquad (4.18)$$

com

$$\alpha_i = \begin{cases} b_i & \text{se } i \in F_{32} \cup F_{12} \\ 0 & \text{se } i \in J_5 - (F_{32} \cup F_{12}) \end{cases}$$

Se $\mu_0 = -\infty$, a terminação em aresta ilimitada ocorre. Caso contrário, faça $T_2 = T_2 \cup \{r\}$, $\bar{F} = \bar{F} - \{r\}$ e seja t o índice da variável básica para a qual μ_0 é atingido. Então:

(i) se $t \in F_{11} \cup F_{21}$, z_t passa a não básica com valor 0 e $F = F - \{t\}$. Se $t = 0$, páre: a solução do BLCP foi encontrada. Caso contrário, faça $\bar{T}_1 = \bar{T}_1 \cup \{t\}$.

(ii) se $t \in F_{32} \cup F_{12}$, z_t passa a não básica com valor b_t e faça

$$F = F - \{t\} \text{ e } \bar{T}_2 = \bar{T}_2 \cup \{t\}$$

(iii) se $t \in T_{11}$, w_t passa a não básica e faça

$$T_1 = T_1 - \{t\} \text{ e } \bar{F} = \bar{F} \cup \{t\}$$

(iv) se $t \in T_{22}$, w_t passa a não básica e faça

$$T_2 = T_2 - \{t\} \text{ e } \bar{F} = \bar{F} \cup \{t\}$$

Caso 3 - z_r é a variável de entrada e $z_r = 0$ $(r \in \bar{T}_1)$.
Calcula-se μ_0 por (4.17). Se $b_r < +\infty$ e $\mu_0 \geq b_r$, faz-se $z_r = b_r$, isto é, $\bar{T}_1 = \bar{T}_1 - \{r\}$ e $\bar{T}_2 = \bar{T}_2 \cup \{r\}$. Se $\mu_0 = +\infty$, a terminação em aresta ilimitada ocorre. Caso contrário, faz-se $F = F \cup \{r\}$ e $\bar{T}_1 = \bar{T}_1 - \{r\}$. Além disso, a actualização dos conjuntos dos índices é realizada como no caso 1.

Caso 4 - z_r é a variável de entrada e $z_r = b_r$ $(r \in \bar{T}_2)$.
Calcula-se μ_0 por (4.18). Se $a_r = 0$ e $-\mu_0 \geq b_r$, faz-se $z_r = 0$, isto é, $\bar{T}_2 = \bar{T}_2 - \{r\}$ e $\bar{T}_1 = \bar{T}_1 \cup \{r\}$. Se $\mu_0 = -\infty$, a terminação em aresta ilimitada ocorre. Caso contrário, faz-se $F = F \cup \{r\}$ e $\bar{T}_2 = \bar{T}_2 - \{r\}$ e actualiza-se os conjuntos dos índices como no caso 2.

Da descrição dos passos do algoritmo chega-se à conclusão que ou uma solução é obtida ou ocorre a terminação em aresta ilimitada. À semelhança do método de Lemke,

este último tipo de terminação pode significar que o BLCP não tem solução, mas também pode ocorrer para BLCPs que têm solução. Seguidamente iremos estabelecer a convergência do método para $M \in PSD$ e mostrar neste caso que a ocorrência da terminação em aresta ilimitada significa que o BLCP não tem solução. Além disso também provaremos que se todos os limites inferiores ou superiores forem finitos, então a terminação em aresta ilimitada não ocorre e obtém-se sempre uma solução do BLCP [JUMAFA91].

2.2 - Convergência do Algoritmo

Consideremos o BLCP (4.1) e seja

$$G = \{i : a_i = -\infty \ , \ b_i = +\infty\} = \emptyset$$

Como vimos anteriormente o BLCP é equivalente ao LCP (4.7). Seja p o vector auxiliar usado na extensão do algoritmo de Lemke proposto neste capítulo e consideremos o seguinte vector:

$$\delta = [\, p_{J_1}, p_{J_2}, -p_{J_3}, 0 \,]^T \qquad (4.19)$$

Então é fácil de ver que o passo inicial da extensão do método de Lemke, EXTLEMKE, corresponde ao passo inicial do método de Lemke para a resolução do LCP (4.7) com δ o vector auxiliar neste último processo. Tal como no método de Lemke iremos supor que todas as soluções básicas usadas pelo processo EXTLEMKE são não degeneradas, isto é,

$$\begin{cases} z_i \text{ básica } \Rightarrow 0 < z_i < b_i \\ w_i \text{ básica } \Rightarrow w_i > 0 \text{ ou } w_i < 0 \end{cases} \qquad (4.20)$$

Nessas condições é possível provar que os quatro casos do processo EXTLEMKE correspondem aos seguintes casos no método de Lemke aplicado ao LCP (4.7):

Caso 1 - v_r é a variável de entrada. Além disso as ocorrências (i), (ii), (iii) e (iv) significam que a variável a passar a não básica é y_t, z_t, v_t, β_t respectivamente.

Caso 2 - β_r é a variável de entrada. Além disso as ocorrências (i), (ii), (iii) e (iv) significam que a variável a passar a não básica é z_t, y_t, v_t, β_t respectivamente.

Caso 3 - z_r é a variável de entrada. Então (i), (ii), (iii) e (iv) são iguais ao caso 1. O procedimento para a ocorrência de $\mu_0 \geq b_r$ corresponde a (i) com $t = r$.

Caso 4 - y_r é a variável de entrada. Então (i), (ii), (iii) e (iv) são iguais ao caso 2. O procedimento para a ocorrência de $-\mu_0 \geq b_r$ corresponde a (i) com $t = r$.

A demonstração de equivalência entre o método EXTLEMKE para o BLCP e o método de Lemke aplicado ao LCP (4.7) é bastante técnica e não será apresentada neste trabalho. Como consequência desta equivalência é possível estabelecer dois resultados em relação à terminação do método EXTLEMKE.

Teorema 4.3 - Se o BLCP é convexo ($M \in PSD$) e tem pelo menos uma solução e se as soluções usadas pelo método EXTLEMKE são não degeneradas, então o método obtém uma solução do BLCP.

Demonstração: Devido ao teorema 4.2, o LCP (4.7) é convexo. Se aplicarmos o método de Lemke a esse LCP ou se obtém uma solução ou então o LCP não tem solução. Dada a equivalência entre o algoritmo de Lemke aplicado ao LCP (4.7) e o processo EXTLEMKE aplicado ao BLCP e a equivalência desses dois problemas, o resultado é imediato. ♦

Teorema 4.4 - Se todos os limites inferiores e superiores são finitos e as soluções básicas são todas não degeneradas, então o método EXTLEMKE obtém uma solução do BLCP.

Demonstração: Devido à equivalência entre os algoritmos e os problemas BLCP e LCP (4.5), basta provar que a terminação ilimitada não pode ocorrer se aplicarmos o método de Lemke ao LCP (4.5). Como $b_i > 0$ e $0 \leq z_i \leq b_i$ para todo o i, então também $0 \leq y_i \leq b_i$ para todo $i = 1,\ldots, n$. Portanto a terminação em aresta ilimitada só pode ocorrer em um dos dois seguintes casos:

(i) $\beta_i \rightarrow +\infty$. Neste caso também $v_i \rightarrow +\infty$ e portanto v_i tem de ser básica pela hipótese de não degenerescência. Então $y_i = b_i > 0$ é básica e β_i não pode ser a variável não básica a ser escolhida para passar a básica. Portanto β_i não pode tender para $+\infty$;

(ii) $v_i \rightarrow +\infty$. Então z_i é não básica ($z_i = 0$) e portanto $y_i = b_i > 0$. Logo β_i é não básica devido à hipótese de não degenerescência. Mas

$$v_i = q_i + z_0 \delta_i + \sum_{j=1}^{n} m_{ij} z_j$$

com δ vector auxiliar do método de Lemke dado por (4.19). Como $0 \leq z_j \leq b_j$ para todo o j, então $z_0 \rightarrow +\infty$. Donde a aresta ilimitada é a primeira e isso é impossível. Logo v_i também não pode tender para $+\infty$.

Portanto a terminação em aresta ilimitada não pode ocorrer e a extensão do método de Lemke termina numa solução do BLCP. ♦

Provámos assim que o método EXTLEMKE é capaz de resolver BLCPs convexos ($M \in PSD$) ou BLCPs em que todos os limites inferiores e superiores sejam finitos, desde que todas as soluções básicas sejam não degeneradas. A experiência computacional com este tipo de BLCPs mostrou que o uso da regra de Bland (isto é, a escolha do primeiro índice no critério da variável a passar a não básica) é suficiente para resolver BLCPs dos tipos citados anteriormente quando algumas das soluções são degeneradas. O desenvolvimento teórico de processos capazes de evitar possíveis ciclos em casos degenerados, bem como o estudo de classes de matrizes para as quais o método EXTLEMKE possa resolver o BLCP, serão certamente áreas de investigação futura.

2.3 - Implementação do Algoritmo para BLCP de Estrutura Esparsa

Da descrição do processo EXTLEMKE conclui-se que é necessário conhecer em cada iteração os conjuntos F_1, F_2, F_3, G, T_1 e T_2 e, actualizados, a coluna pivotal $d = A_{.r}$ e o vector η (4.14) dos termos independentes.

Uma estrutura de dados simples é usada para armazenar toda a informação relacionada com os conjuntos de variáveis mencionados anteriormente. Nessa estrutura de dados as n variáveis $z_1,\ldots, z_n$ $(w_1,\ldots,w_n)$ são representadas por inteiros $1,\ldots, n$ $(n+1,\ldots, 2n)$ respectivamente. O inteiro $(2n+1)$ corresponde à variável artificial z_0. Usando esta representação, duas variáveis i e j são complementares se e só se $|i - j| = n$ e isto é explorado na escolha da variável de entrada em cada iteração. A estrutura de dados consiste em três vectores IVAR, IPT e MARCA e é definida do seguinte modo:

(i) o vector IVAR tem dimensão $(2n+1)$ e armazena nas suas primeiras NG componentes as variáveis z_i correspondentes ao conjunto G. Todas as restantes variáveis básicas são armazenadas nas $n - NG$ posições seguintes de IVAR. As nvb variáveis não básicas z_i que estão no limite superior b_i são armazenadas nas componentes $(n+1),\ldots, (n+nvb)$ de IVAR. As últimas

(2n+1) – (n+nvb) componentes do vector IVAR não são importantes e podem conter quaisquer números inteiros.

(ii) O vector IPT tem dimensão n e satisfaz

$$IPT(i) = j \Leftrightarrow IVAR(j) = i$$

para todo $j \in \{n+1,\ldots, n+nvb\}$. Todas as outras componentes deste vector são iguais a zero.

(iii) O vector MARCA tem dimensão n e é definido do seguinte modo

$$MARCA(i) = \begin{cases} 1 \Leftrightarrow IVAR(i) \in T_1 \\ -1 \Leftrightarrow IVAR(i) \in T_2 \\ 2 \Leftrightarrow IVAR(i) \in F_1 \cup F_3 \\ 3 \Leftrightarrow IVAR(i) \in F_2 \end{cases}$$

Estes vectores são modificados em cada iteração por um processo simples e rápido. Além disso esta estrutura de dados satisfaz completamente os nossos propósitos de representar os conjuntos F_i e T_i e escolher a variável de entrada em cada iteração de um modo eficiente.

O sistema $w = q + z_0 p + Mz$ pode ser escrito na forma

$$Ax = q \qquad (4.21)$$

onde

$$A = [I_n : -p : -M] \quad , \quad x = [w : z_0 : z]^T$$

Como apenas são usadas soluções básicas associadas com o sistema (4.21), então em cada iteração o vector η e a coluna pivotal d podem ser obtidos a partir de

$$B\eta = q + M_{.\overline{T}_2} b_{\overline{T}_2}$$

e

$$B\,d = \begin{cases} -I_{.r} & \text{se } w_r \text{ é a variável escolhida em (C1)} \\ M_{.r} & \text{se } z_r \text{ é a variável escolhida em (C1)} \end{cases}$$

onde B é a matriz base associada com a solução básica. Tal como anteriormente a decomposição LU da matriz B é usada para resolver estes sistemas e é actualizada usando o esquema descrito no capítulo 2.

É evidente que o método de Lemke é um caso particular do processo EXTLEMKE quando todos os limites inferiores são nulos e os limites superiores são todos infinitos. Por isso este processo pode também ser usado para resolver LCPs. Neste último caso as estruturas de dados ficam mais simplificadas.

2.4 - Aplicação à Programação Quadrática Convexa

Consideremos o programa quadrático convexo (CQP) na sua forma mais geral

$$\begin{aligned} &\text{minimizar} && c^T x + \frac{1}{2} x^T Q x \\ &\text{sujeito a} && A_1 x = d^1 \ , \ A_2 x \geq d^2 \\ & && \alpha_i \leq x_i \leq \beta_i \ , \ i = 1, 2, \ldots, \ell \end{aligned} \qquad (4.22)$$

com $A_i \in R^{m_i x \ell}$, $d^i \in R^{m_i}$, $i = 1,2$, $c \in R^{\ell}$, $x \in R^{\ell}$ e $Q \in R^{\ell x \ell}$ é uma matriz simétrica PSD e $-\infty \leq \alpha_i < \beta_i \leq +\infty$ para todo $i = 1,\ldots,\ell$. Como a função é convexa, as suas condições de optimalidade de primeira ordem são necessárias e suficientes e constituem um BLCP [GIMUWR81] com

$$q = \begin{bmatrix} c \\ -d^1 \\ -d^2 \end{bmatrix}, \ M = \begin{bmatrix} Q & -A_1^T & -A_2^T \\ A_1 & 0 & 0 \\ A_2 & 0 & 0 \end{bmatrix}, \ w = \begin{bmatrix} y \\ v^1 \\ v^2 \end{bmatrix}, \ z = \begin{bmatrix} x \\ \theta^1 \\ \theta^2 \end{bmatrix}$$

e

$$a_i = \begin{cases} \alpha_i & i = 1,\ldots, \ell \\ -\infty & i = \ell+1,\ldots, \ell+m_1 \\ 0 & i = \ell+m_1+1,\ldots, \ell+m_1+m_2 \end{cases}$$

$$b_i = \begin{cases} \beta_i & i = 1,\ldots, \ell \\ +\infty & i = \ell+1,\ldots, \ell+m_1+m_2 \end{cases}$$

Devido a esta equivalência o método EXTLEMKE pode ser usado para obter o mínimo global de qualquer programa quadrático convexo. Se a terminação em aresta ilimitada ocorrer então ou o CQP é não admissível ou a função tende para $-\infty$ no conjunto de restrições. A implementação do método EXTLEMKE descrita anteriormente pode ser usada para CQPs de grandes dimensões desde que seja introduzida uma estrutura de dados para representar os limites inferiores e superiores. Um estudo computacional da eficiência do processo EXTLEMKE para a resolução de CQPs [JUFA91c] é

seguidamente apresentada. Todas as experiências foram efectuadas num CDC CYBER 180-830 da Universidade do Porto. Os resultados computacionais são apresentados nos quadros 4.1 e 4.2 onde são usadas as seguintes notações:

n = dimensão do BLCP = ordem da matriz

ℓ = número de variáveis do CQP.

m = número de restrições do CQP (4.22)

nup = número de limites superiores finitos nas variáveis x_i do CQP (4.22)

nonz = número de não zeros da matriz M do BLCP

nvz = número de variáveis básicas z_i na solução do BLCP

nvb = número de variáveis não básicas z_i igual ao limite superior na solução do BLCP

NI = número de iterações.

T = tempo de CPU em segundos.

O quadro 4.1 apresenta o comportamento do método EXTLEMKE na resolução de um certo número de CQPs em que as restrições consistem em apenas limites inferiores e superiores nas variáveis. As matrizes dos problemas-teste foram tiradas das seguintes fontes:

ZR = matrizes simétricas positivas definidas (SPD) com elementos não diagonais não positivos e gerados aleatoriamente [JUPI89a]

P = matrizes SPD pentadiagonais [JUPI89a]

PS = matrizes SPD associadas com problemas de "portfolio" descritos em [PAN80]

EPA = matriz SPD de uma aplicação de CQP em análise elastoplástica de estruturas [JUPI89a]

Todos os CQPs foram gerados com os limites inferiores iguais a zero.

Probl.	fonte	n	nonz	nup	nvz	nvb	NI	T
1	ZR	500	2156	500	125	125	376	52.7
					187	188	564	339.
				250	125	125	376	47.1
					183	67	318	39.2
2	ZR	1000	2842	1000	250	250	751	154.
					375	375	1126	319.
				500	250	250	751	142.
					372	128	629	116.
3	P	500	2494	500	125	125	376	46.8
					187	188	568	74.2
				250	125	125	380	40.1
					183	69	326	33.6
4	P	1000	4994	1000	250	250	757	171.
					375	375	1146	295.
				500	250	250	757	157.
					374	134	647	140.
5	PS	200	1922	200	50	50	151	8.8
					75	75	221	16.8
				100	50	50	151	8.7
					72	28	129	7.1
6	PS	600	9878	600	150	150	455	82.7
					225	225	676	193.
				300	150	150	454	70.4
					218	82	386	59.9
7	EPA	484	9920	484	121	121	390	93.9
					181	182	564	217.
				242	121	121	456	107.
					193	58	392	97.5

Quadro 4.1. – Solução do CQP só com limites

O quadro 4.2 apresenta os resultados da solução de CQPs descritos em [GO86] que contêm limites em todas as variáveis e igualdades.

Probl.	ℓ	m	n	nonz	nvz	nvb	NI	T
G11	99	49	148	439	95	2	161	12.2
G12	199	99	298	889	193	4	451	79.5
G13	299	149	448	1339	294	3	560	171.
G14	399	199	598	1789	393	3	851	368.
G15	499	249	748	2239	470	2	1032	476.
G21	99	49	148	586	49	49	107	5.8
G22	199	99	298	1186	99	99	203	20.6
G23	299	149	448	1786	149	149	303	42.6
G24	399	199	598	2386	199	199	403	73.7
G25	499	249	748	2986	249	249	503	119.

Quadro 4.2 – Solução do CQP com restrições de igualdade

Os resultados das experiências de resolução de CQPs de média e grande dimensão mostram que o processo EXTLEMKE resolve eficientemente todos os CQPs. Como é esperado deste tipo de métodos pivotais, o número de iterações depende do número (nvz + nvb) de variáveis que mudam o seu estatuto da solução inicial para a final. Estas conclusões, conjuntamente com a possibilidade do processo começar com uma qualquer solução básica, poderão ser exploradas no desenvolvimento de métodos do tipo Newton [JO79, MAT85] para a resolução de programas não lineares de grandes dimensões. Este será um dos tópicos de investigação futura.

3 - Métodos Pivotais Principais para Solução do BLCP Côncavo com Limites Finitos

3.1 - Os Algoritmos

Nesta secção vamos apresentar algoritmos que fornecem uma solução do BLCP quando M é uma matriz NSD e os limites são todos finitos, isto é, $a_i = 0$ e $b_i < +\infty$ para todo $i = 1,\ldots, n$. Como vimos anteriormente, o BLCP é equivalente ao Problema de Desigualdades Variacionais Lineares (4.4) com

$$S = \{z \in R^n : 0 \le z \le b\}$$

Como S é limitado, o BLCP tem solução para cada $q \in R^n$. Além disso, se M é simétrica, o BLCP representa as condições de Kuhn-Tucker do Programa Quadrático (4.2), isto é, qualquer solução do BLCP é um ponto estacionário desse programa.

Como todos os limites são finitos, o BLCP é equivalente a um LCP do tipo

$$\begin{aligned} & w = q + Mz + I_n \beta \\ & v = b - I_n z \\ & w, v, z, \beta \ge 0 \\ & v^T \beta = z^T w = 0 \end{aligned} \qquad (4.23)$$

Os algoritmos que vamos discutir são métodos pivotais principais que em cada iteração usam soluções básicas complementares inadmissíveis do LCP (4.23) até obter uma solução complementar admissível.

Dada a solução básica ($z = 0$, $\beta = 0$) do LCP (4.23) podemos considerar os conjuntos $F = \{i : q_i < 0\}$ e $T = \{i : q_i \ge 0\}$ e escrever as igualdades do LCP (4.24) na seguinte forma:

	1	z_F	z_T	β_F	β_T
$w_F =$	q_F	M_{FF}	M_{FT}	$I_{\|F\|}$	0
$w_T =$	q_T	M_{TF}	M_{TT}	0	$I_{\|T\|}$
$v_F =$	b_F	$-I_{\|F\|}$	0	0	0
$v_T =$	b_T	0	$-I_{\|T\|}$	0	0

(4.24)

Se efectuarmos uma operação pivotal bloco principal com pivot

$$\begin{bmatrix} M_{FF} & I_{|F|} \\ -I_{|F|} & 0 \end{bmatrix}$$

obtém-se o seguinte quadro contraído:

	1	v_F	z_T	w_F	β_T
$\beta_F =$	$\bar{q}_F$	M_{FF}	$-M_{FT}$	$I_{\|F\|}$	0
$w_T =$	$\bar{q}_T$	$-M_{TF}$	M_{TT}	0	$I_{\|T\|}$
$z_F =$	b_F	$-I_{\|F\|}$	0	0	0
$v_T =$	b_T	0	$-I_{\|T\|}$	0	0

(4.25)

com

$$\begin{cases} \bar{q}_F = -(q_F + M_{FF}\, b_F) \\ \bar{q}_T = q_T + M_{TF}\, b_F \end{cases} \tag{4.26}$$

Consideremos agora o quadro (4.25). Se $\bar{q} = (\bar{q}_F, \bar{q}_T) \geq 0$, então $z = (b_F, 0)$ é uma solução do BLCP. De outro modo existe pelos menos um $i \in FUT$ tal que $\bar{q}_i < 0$. Se efectuarmos uma operação pivotal com pivot

$$\begin{bmatrix} m_{ii} & 1 \\ -1 & 0 \end{bmatrix}$$

$\bar{q}_i$ é transformado em

$$\bar{\bar{q}}_i = -(\bar{q}_i + m_{ii}\, b_i)$$

Como $m_{ii} \leq 0$, $b_i > 0$ e $\bar{q}_i < 0$, então $\bar{\bar{q}}_i > 0$ e a inadmissibilidade desaparece. Além disso após essa operação pivotal obtém-se um quadro da mesma forma, com os conjuntos F e T actualizados da seguinte forma:

$$\begin{cases} i \in F \Rightarrow F = F - \{i\} \quad e \quad T = T \cup \{i\} \\ i \in T \Rightarrow F = F \cup \{i\} \quad e \quad T = T - \{i\} \end{cases}$$

Portanto este tipo de processo pode ser repetido sucessivamente. Assim se obtém um algoritmo pivotal principal, que pode ser usado para a resolução de um BLCP com uma matriz NSD, simétrica ou não. Contudo a convergência teórica deste processo para

matrizes não simétricas ainda não foi estabelecida. Para matrizes simétricas a convergência está assegurada. Com efeito basta notar que o algoritmo pivotal principal é exactamente igual ao método de Keller, em que a solução inicial é $z = (b_F, 0)$, com

$$F = \{i : q_i < 0\}$$

Esta escolha é conveniente, pois o valor da função f do programa quadrático (4.2) com $z = (b_F, 0)$ é inferior a f(0). Com efeito

$$f(z) = q_F^T b_F + \frac{1}{2} b_F^T M_{FF} b_F < 0 = f(0)$$

pois $M_{FF} \in NSD$, $q_F < 0$ e $b_F > 0$.

Portanto o seguinte algoritmo pivotal principal resolve o BLCP quando M é uma matriz NSD simétrica e $a_i = 0$, $b_i < +\infty$ para todo $i = 1,\ldots, n$.

ALGORITMO PRPIVT

Passo 0 – Faça $F = \{i : q_i < 0\}$ e $T = \{i : q_i \geq 0\}$.

Passo 1 – Calcule $\bar{q}$ por (4.26).

Passo 2 – Se $\bar{q} = (\bar{q}_F, \bar{q}_T) \geq 0$, páre; $z = (b_F, 0)$ é uma solução do BLCP. De outro modo seja

$$r = \min\{i : \bar{q}_i < 0\}$$

Faça

$$F = \begin{cases} F - \{r\} & \text{se } r \in F \\ F \cup \{r\} & \text{se } r \notin F \end{cases}$$

e $T = \{1,\ldots, n\} - F$. Vá para Passo 1.

Para implementar este algoritmo, basta considerar um vector de inteiros MARCA de ordem n definido por

$$MARCA(i) = \begin{cases} -1 & \text{se } i \in F \\ 1 & \text{se } i \in T \end{cases}$$

e um vector real Q de ordem n. Esse vector é actualizado em cada operação pivotal da seguinte forma:

$$Q(i) = \begin{cases} Q(i) + m_{ir}\, b_r & \text{se } r \in T \\ Q(i) - m_{ir}\, b_r & \text{se } r \in F \end{cases} \qquad (4.27)$$

Além disso o vector $\bar{q}$ dos termos independentes vem dado por

$$\bar{q}_i = \text{MARCA}(i).Q(i)\ ,\ i = 1,\ldots, n$$

Cada iteração do algoritmo consiste numa passagem pelas n componentes dos vectores Q e MARCA.

Sempre que $\bar{q}_i < 0$, os vectores MARCA e Q são actualizados por

$$\text{MARCA}(i) = -\text{MARCA}(i)$$

e pelas fórmulas (4.27), respectivamente. Na prática é conveniente começar com $F = \emptyset$, isto é, com MARCA(i) = 1, i = 1,..., n, pois poupa-se uma passagem pelas n componentes do vector q. A implementação do algoritmo PRPIVT tem então a seguinte forma:

IMPLEMENTAÇÃO DE PRPIVT

Passo 0 – Faça MARCA(i) = 1 e $Q(i) = q_i$, i = 1,..., n e NUPD = 0.

Passo 1 – Para i = 1,..., n
Calcule VAL = MARCA(i).Q(i)
Se VAL < 0 faça
Para j = 1,..., n
$\quad Q(j) = Q(j) + m_{ji}\, b_i$.MARCA(i)
MARCA(i) = −MARCA(i)
NUPD = NUPD + 1

Passo 2 – Se NUPD = 0 vá para EXIT. De outro modo faça NUPD = 0 e vá para Passo 1.

EXIT – O vector

$$z_i = \begin{cases} 0 & \text{se MARCA}(i) = 1 \\ b_i & \text{se MARCA}(i) = -1 \end{cases}$$

é uma solução do BLCP.

Como facilmente se constata pela descrição dos seus passos, o algoritmo PRPIVT é extremamente simples. A complexidade deste processo não é conhecida e está relacionada com o número de vezes que o Passo 1 é usado.

Como cada operação bloco principal com pivot

$$\begin{bmatrix} M_{KK} & I_{|K|} \\ -I_{|K|} & 0 \end{bmatrix}$$

é equivalente a |K| operações pivotais com pivots

$$\begin{bmatrix} m_{ii} & 1 \\ -1 & 0 \end{bmatrix}$$

então também o seguinte algoritmo bloco principal pivotal é capaz de resolver o BLCP quando M∈ NSD.

ALGORITMO BLOCK

Passo 0 – Faça $F = \{i : q_i < 0\}$ e $T = \{i : q_i \geq 0\}$.

Passo 1 – Calcule $\bar{q}$ por (4.26).

Passo 2 – Determine o conjunto de inadmissibilidades

$$K = \{i : \bar{q}_i < 0\}$$

Se $K = \emptyset$, páre ; $z = (b_F, 0)$ é solução do BLCP. De outro modo vá para Passo 3.

Passo 3 – Sejam

$$K_F = K \cap F \text{ e } K_T = K \cap T$$

Faça

$$F = F - K_F \cup K_T \,, \quad T = T - K_T \cup K_F$$

e vá para Passo 1.

Na implementação deste algoritmo utiliza-se um vector INADM de dimensão nadm ≤ n, que em cada iteração armazena as inadmissibilidades (índices do conjunto K). O processo tem a seguinte forma:

IMPLEMENTAÇÃO DE BLOCK

Passo 0 – Faça MARCA(i) = 1 e Q(i) = q_i, i = 1,..., n.

Passo 1 – Actualização do conjunto de Inadmissibilidades

nadm = 0

Para i = 1,..., n

| Se MARCA(i).Q(i) < 0 faça
| nadm = nadm + 1
| INADM(nadm) = i

Se nadm = 0, vá para EXIT. De outro modo vá para Passo 2.

Passo 2 – Para i = 1,..., nadm

k = INADM(i)

Para j = 1,..., n

| Q(j) = Q(j) + m_{jk} b_k.MARCA(k)

MARCA(k) = – MARCA(k)

e vá para Passo 1.

EXIT – O vector

$$z_i = \begin{cases} 0 & \text{se MARCA(i)} = 1 \\ b_i & \text{se MARCA(i)} = -1 \end{cases}$$

é uma solução do BLCP.

Da descrição dos passos da implementação do método BLOCK facilmente se conclui que o algoritmo PRPIVT necessita de um menor esforço computacional. Na última parte desta secção apresentaremos alguma experiência computacional com estes dois algoritmos, que mostra, para a maior parte dos casos, a superioridade do método PRPIVT.

Tal como o algoritmo PRPIVT também a complexidade do método BLOCK não é conhecida. Contudo é possível estabelecer que o método BLOCK é polinomial nos seguintes casos:

(i) M é não positiva

(ii) Os elementos não diagonais de M são não negativos.

Além disso a matriz M pode ser não simétrica em ambos os casos.

Caso 1 – M é uma matriz não positiva

Se

$$F_1 = F - K_F\,,\ T_1 = T - K_T$$

então as componentes do vector $\bar{q}$ são transformadas de acordo com as seguintes fórmulas:

$$\begin{aligned}
\bar{q}_{F_1} &\rightarrow \bar{q}_{F_1} + \bar{M}_{F_1K_F} b_{K_F} - M_{F_1K_T} b_{K_T} \\
\bar{q}_{K_F} &\rightarrow -\bar{q}_{K_F} - \bar{M}_{K_FK_F} b_{K_F} + M_{K_FK_T} b_{K_T} \\
\bar{q}_{K_T} &\rightarrow -\bar{q}_{K_T} + \bar{M}_{K_TK_F} b_{K_F} - M_{K_TK_T} b_{K_T} \\
\bar{q}_{T_1} &\rightarrow \bar{q}_{T_1} - \bar{M}_{T_1K_F} b_{K_F} + M_{T_1K_T} b_{K_T}
\end{aligned} \qquad (4.28)$$

Se $M \leq 0$, então as fórmulas (4.26) implicam que $\bar{q}_F \geq 0$ após o Passo 0 do algoritmo BLOCK. Portanto $K_F = \emptyset$ e na primeira iteração do algoritmo tem-se

$$F = F \cup K_T = F_1 \cup K_T$$

Além disso as componentes do vector $\bar{q}_F$ após esta iteração vêm dadas por

$$\begin{cases} i \in F_1 \Rightarrow \bar{q}_i = \bar{q}_i - M_{iK_T} b_{K_T} \\ i \in K_T \Rightarrow \bar{q}_i = -\bar{q}_i - M_{iK_T} b_{K_T} \end{cases}$$

Como $M \leq 0$, então $\bar{q}_F \geq 0$ e portanto o conjunto de inadmissibilidades foi reduzido na segunda iteração. Além disso não há inadmissibilidades no conjunto F, pelo que sempre que uma variável z_i toma o valor b_i permanece até ao fim com esse valor. Isto mostra que ambos os métodos BLOCK e PRPIVT têm no máximo n iterações e são por isso polinomiais. Além disso as convergências dos algoritmos são baseadas apenas na redução do número de inadmissibilidades, pelo que se mantêm convergentes quando M é não simétrica.

Como não há inadmissibilidades no conjunto F, ambas as implementações dos algoritmos PRPIVT devem ser modificadas de modo a explorar essa propriedade. Assim no método PRPIVT é introduzido um vector INADM de dimensão nadm $\leq$ n, que é actualizado segundo um processo semelhante ao usado no método BLOCK e que contém os índices que pertencem ao conjunto T. No Passo 1 há que investigar apenas o sinal das nadm componentes do vector Q cujos índices pertencem ao vector INADM. Além disso

apenas essas nadm componentes do vector Q têm de ser actualizadas. O mesmo se passa no que diz respeito ao algoritmo BLOCK, onde a investigação das inadmissibilidades no Passo 1 e a actualização do vector Q no Passo 2 são feitas apenas em relação a esses índices que pertencem ao vector INADM. Se compararmos os passos dos processos PRPIVT e BLOCK neste caso, facilmente concluímos que o primeiro continua a ter um menor esforço computacional por iteração. Além disso a estratégia usada no algoritmo PRPIVT permite fixar mais variáveis z_i no valor b_i. Isso vai obviamente reduzir a pesquisa na iteração seguinte. A experiência computacional apresentada no final desta secção mostra que o algoritmo PRPIVT é realmente mais eficiente.

Caso 2 – Os elementos não diagonais de M são não negativos

Para estabelecer que o método BLOCK é polinomial neste caso, precisamos do seguinte resultado

Teorema 4.5 – Considere-se o quadro

		z	β
w =	q	M	I_n
v =	b	$-I_n$	0

com $q < 0$ e $b > 0$. Se efectuarmos uma operação pivotal principal com pivot

$$\begin{bmatrix} M & I_n \\ -I_n & 0 \end{bmatrix}$$

então o transformado do vector q tem pelo menos uma componente não negativa.

Demonstração: Se efectuarmos a referida operação pivotal, obtemos um quadro de forma

		v	w
β =	$\bar{q}$	M	I_n
z =	b	$-I_n$	0

com

$$\bar{q} = -(q + Mb)$$

Se $\bar{q} < 0$, então $q + Mb > 0$ e portanto $q^T b + b^T Mb > 0$. Isso é impossível por $q < 0$, $b > 0$ e $M \in NSD$. Portanto existe pelo menos um i tal que $\bar{q}_i \geq 0$. ♦

Consideremos então a aplicação do método BLOCK para a resolução do BLCP com M NSD (simétrica ou não simétrica) de elementos não diagonais não negativos. Se usarmos a notação J^i para representar o conjunto J na iteração i, então após o Passo 0 tem-se, por (4.26)

$$\bar{q}_{F^1} = -(q_{F^1} + M_{F^1F^1}\, b_{F^1})$$

$$\bar{q}_{T^1} = q_{T^1} + M_{T^1F^1}\, b_{F^1} \geq 0$$

Portanto

$$K_T^1 = \emptyset\,, \quad K^1 = K_F^1 \subset F^1$$

e os conjuntos F e T são actualizados da seguinte forma:

$$F^2 = F^1 - K_F^1\,, \quad T^2 = T^1 \cup K_F^1$$

Na segunda iteração tem-se, por (4.28),

$$\bar{q}_{F^2} = -(q_{F^2} + M_{F^2F^1}\, b_{F^1}) + M_{F^2K_F^1}\, b_{K_F^1}$$

Como os índices de F^2 correspondem a variáveis com valores não negativos, então

$$-(q_{F^2} + M_{F^2F^1}\, b_{F^1}) \geq 0$$

e portanto

$$\bar{q}_{F^2} \geq 0$$

Além disso, as novas componentes $\bar{q}_T$ do vector $\bar{q}_{T^2}$ vêm dadas por

$$\bar{q}_{K_F^1} = (q_{K_F^1} + M_{K_F^1F^1}\, b_{F^1}) - M_{K_F^1K_F^1}\, b_{K_F^1}$$

$$\bar{q}_{T^1} = (q_{T^1} + M_{T^1F^1}\, b_{F^1}) - M_{T^1K_F^1}\, b_{K_F^1}$$

Mas $q_{T^1} \geq 0$ e

$$M_{T^1F^1}\, b_{F^1} - M_{T^1K_F^1}\, b_{K_F^1} \geq 0$$

por $K_F^1 \subset F^1$. Donde

$$\bar{q}_{T^1} \geq 0$$

Então ou $\bar{q}_{K_F^1} \geq 0$ e obtém-se uma solução do BLCP ou

$$K^2 = K_T^2 \subseteq K_F^1 = K^1$$

Mas se $K_T^2 = K_F^1$, isso significa que todas as componentes $\bar{q}_i$, $i \in K_F^1$, se mantiveram negativas, o que é impossível pelo teorema 4.5. Portanto $K^2 \subset K^1$.

Na terceira iteração, os conjuntos F e T são então actualizados da seguinte forma:

$$F^3 = F^2 \cup K_T^2, \quad T^3 = T^2 - K_T^2$$

Mas após essas mudanças tem-se, por (4.28),

$$\bar{q}_{T^3} = q_{T^3} + M_{T^3F^2}\, b_{F^2} + M_{T^3K_T^2}\, b_{K_T^2} \geq 0$$

pois $q_{T^3} + M_{T^3F^2}\, b_{F^2} \geq 0$ por as inadmissibilidades de T^2 não pertencerem a T^3. Além disso as novas componentes $\bar{q}_{F^2}$ do vector $\bar{q}_{F^3}$ vêm dadas por

$$\bar{q}_{F^2} = -\left(q_{F^2} + M_{F^2F^2}\, b_{F^2} + M_{F^2K_T^2}\, b_{K_T^2}\right)$$

Mas $K_T^2 \subset K_F^1$ e portanto

$$F^2 \cup K_T^2 \subset F^2 \cup K_F^1 = F^1$$

Se $L = F^1 - (F^2 \cup K_T^2)$, então vem

$$M_{F^2F^2}\, b_{F^2} + M_{F^2K_T^2}\, b_{K_T^2} = M_{F^2F^1}\, b_{F^1} - M_{F^2L}\, b_L$$

Donde

$$\bar{q}_{F^2} = -\left(q_{F^2} + M_{F^2F^1}\, b_{F^1}\right) + M_{F^2L}\, b_L \geq 0$$

Portanto ou as componentes $\bar{q}_{K_T^2}$ de $\bar{q}_{F^3}$ são não negativas e obtém-se uma solução do BLCP ou então

$$K^3 = K_F^3 \subset K_T^2 = K^2$$

pois, como anteriormente, a igualdade $K_F^3 = K_T^2$ não se pode verificar pelo teorema 4.5.

De modo semelhante se provava por indução que para qualquer iteração $i \geq 3$ se tem

$$K^{i+1} \subset K^i$$

e

$$\begin{cases} i \text{ ímpar} \Rightarrow K^i = K_F^i \\ i \quad \text{par} \Rightarrow K^i = K_T^i \end{cases}$$

Provámos assim que o método de BLOCK tem no máximo n iterações e que as inadmissibilidades ocorrem alternadamente em F e T à medida que o processo vai prosseguindo. Esta propriedade pode ser explorada na implementação do algoritmo. Com efeito a actualização do conjunto de inadmissibilidades pode ser implementada de modo a considerar apenas as inadmissibilidades da iteração anterior.

A complexidade do algoritmo PRPIVT não é conhecida. Na última parte desta secção iremos apresentar experiência computacional com ambos os algoritmos na resolução de BLCPs com este tipo de matrizes que mostra uma superioridade do método BLOCK neste caso.

3.2 - Experiência Computacional

Nesta secção apresentamos alguma experiência computacional com os algoritmos BLOCK e PRPIVT para os três casos analisados. As experiências foram efectuadas usando um computador CYBER CDC 180-830 da Universidade do Porto. Para gerar as matrizes NSD, considerámos problemas com matrizes PSD e multiplicámos todos os seus elementos por (–1), usando técnicas já apresentadas anteriormente. Assim gerámos problemas-teste com matrizes não positivas (PT1, PT2), com matrizes de elementos não diagonais não negativos (PT3, PT4) e com matrizes de elementos não diagonais de sinal arbitrário (PT5, PT6). As matrizes –M dos problemas-teste PT2, PT3 e PT4 foram geradas de modo aleatório, segundo técnica apresentada em [JUPI89a]. Em PT1 a matriz –M é tridiagonal por blocos obtida na resolução de problemas de equações diferenciais parciais pelo método das diferenças finitas. Finalmente, as matrizes –M dos problemas

PT5 e PT6 correspondem a problemas de "portfolio" descritos em [PAN80]. Os limites superiores de todos os problemas são iguais a 1. Para cada problema PTi, i = 1,..., 6, foram considerados cinco problemas-teste que apenas diferem nos termos independentes. Esses elementos foram gerados aleatoriamente de modo a que sejam alguns negativos.

No quadro 4.3 apresenta-se o comportamento dos dois algoritmos na resolução dos cinco problemas de cada PT_i, mostrando, em termos de tempo de CPU, o resultado melhor (B), o pior (P) e o médio (M). Nesse quadro, usaram-se as seguintes notações:

n = número de variáveis z = ordem da matriz M
NZ = número de não zeros de M
NEGMIN = número mínimo de q_i negativos
NEGMAX = número máximo de q_i negativos
NI = número de iterações = número de passagens pelo Passo 1
T = tempo de CPU em segundos

						PRPIVT		BLOCK	
Probl.	n	NZ	NEGMIN	NEGMAX		NI	T	NI	T
PT1	3000	11660	505	2240	B	2	.08	2	.08
					M	2	.10	2	.10
					P	2	.12	2	.12
PT2	3000	13108	518	1950	B	5	.17	6	.20
					M	5	.18	8	.24
					P	7	.22	13	.34
PT3	3000	13532	505	3000	B	2	.12	2	.06
					M	2	.21	2	.15
					P	3	.34	3	.27
PT4	3000	15902	505	3000	B	2	.13	2	.07
					M	2	.25	3	.21
					P	3	.36	3	.30
PT5	3000	24070	243	3000	B	4	.37	7	.47
					M	5	.43	7	.54
					P	8	.58	6	.64
PT6	3000	28062	243	3000	B	5	.40	7	.48
					M	5	.46	8	.59
					P	7	.63	7	.70

Quadro 4.3

Os resultados computacionais permitem concluir que os algoritmos PRPIVT e BLOCK são extremamente eficientes na resolução de BLCPs de grandes dimensões. A eficiência dos métodos é afectada um pouco com o aumento da densidade problema, mas tal redução não é considerável. Assim o ligeiro agravamento dos resultados nos problemas-teste PT5 e PT6 é justificado mais por um aumento de densidade do que pelo facto de se tratar de problemas gerais onde o comportamento polinomial dos algoritmos não está provado teoricamente. Os algoritmos são bastante robustos, dado que não se verifica grande alteração entre o melhor e o pior resultado. O número de elementos negativos do vector q não parece ter grande importância na eficiência dos algoritmos. Apesar das soluções obtidas pelos dois métodos nem sempre serem coincidentes, as diferenças significativas encontradas para os respectivos esforços computacionais indica que o algoritmo PRPIVT é mais eficiente para $M \leq 0$ e para as matrizes gerais. De um modo geral nestes dois casos o algoritmo PRPIVT necessita de um menor número de iterações para obter uma solução do BLCP. Para matrizes NSD de elementos não diagonais não negativos o método BLOCK é mais eficiente, pois a actualização do conjunto de inadmissibilidades neste algoritmo requer um esforço computacional bastante inferior.

4 - Extensão do Método de Keller a BLCP Bissimétrico

4.1 - Algoritmo

No capítulo 1 apresentámos sumariamente o método de Keller para a resolução de problemas de programação quadrática com apenas restrições de desigualdade. Se o programa quadrático é côncavo então é possível estender este método para restrições de igualdade e limites superiores nas variáveis explorando devidamente as características do problema. Considere-se então o seguinte programa quadrático côncavo:

$$\underset{x \in S}{\text{minimizar}} \quad c^T x + \frac{1}{2} x^T Q x \tag{4.29}$$

com

$$S = \{x \in R^n : Ax = d \ \text{ e } \ 0 \leq x \leq b\}$$

As condições de Kuhn-Tucker de 1ª ordem deste programa quadrático constituem o seguinte BLCP:

$$\begin{aligned} y &= c + Qx - A^T\theta \\ 0 &= -d + Ax \end{aligned} \qquad (4.30)$$

$$\left.\begin{array}{l} x_i = 0 \Rightarrow y_i \geq 0 \\ x_i = b_i \Rightarrow y_i \leq 0 \\ 0 < x_i < b_i \Rightarrow y_i = 0 \\ 0 \leq x_i \leq b_i \end{array}\right\} \quad i = 1,\ldots, n$$

A extensão do método de Keller aqui proposta é bastante semelhante ao método simplex com limites superiores [MU76, cap. 10]. Assim começa-se por encontrar uma solução admissível do conjunto S. Para obter essa solução usa-se o processo Fase 1 com bases de trabalho descrito em [MU76].

Sejam B, N^1 e N^2 respectivamente as submatrizes de A associadas às variáveis básicas, variáveis não básicas com valor zero e variáveis não básicas com valor no limite superior. Então A = [B : N] e N = [N^1 : N^2]. Se F, T_1 e T_2 ($T_1 \cup T_2 = \{1,\ldots, n\} - F$) forem os conjuntos dos índices das colunas associadas a B, N^1 e N^2, a solução admissível obtida na Fase 1 do algoritmo tem a forma

$$\bar{x}_F = \bar{q} \ , \ \bar{x}_{T_1} = 0 \ \text{ e } \ \bar{x}_{T_2} = b_{T_2}$$

Então pode associar-se a esta solução um quadro da forma

		x_{T_1}	x_{T_2}	ψ	y_F
y_{T_1} =	$\bar{p}_{T_1}$	$\bar{Q}_{T_1T_1}$	$\bar{Q}_{T_1T_2}$	$\bar{Q}_{T_1F}$	$-(\bar{N}^1)^T$
y_{T_2} =	$\bar{p}_{T_2}$	$\bar{Q}_{T_2T_1}$	$\bar{Q}_{T_2T_2}$	$\bar{Q}_{T_2F}$	$-(\bar{N}^2)^T$
θ =	$\bar{p}_F$	$\bar{Q}_{FT_1}$	$\bar{Q}_{FT_2}$	$\bar{Q}_{FF}$	$-\bar{B}^T$
x_F =	$\bar{q}$	$\bar{N}^1$	$\bar{N}^2$	$\bar{B}$	0

(4.31)

com ψ um vector de variáveis artificiais.

Este quadro é obtido de (4.30) por uma operação pivotal principal com pivot

$$\begin{bmatrix} Q_{FF} & -B^T \\ B & 0 \end{bmatrix}$$

Esta matriz é não singular pois B é não singular. Como θ é um vector de variáveis sem restrição de sinal e $\bar{x}$ é uma solução admissível de S, então $\bar{q} \geq 0$ e

$$\bar{y}_{T_1} = \bar{p}_{T_1}, \bar{y}_{T_2} = \bar{p}_{T_2}, \bar{\theta} = \bar{p}_F, \bar{x}_F = \bar{q}, \bar{x}_{T_2} = b_{T_2}$$

$$\bar{y}_F = \bar{x}_{T_1} = 0$$

é solução do BLCP (4.30) desde que $\bar{p}_{T_1} \geq 0$ e $\bar{p}_{T_2} \leq 0$. Caso contrário, determina-se a variável a entrar x_r pelo critério

$$|\bar{p}_r| = \max \{ |\bar{p}_i| : (i \in T_1 \text{ e } \bar{p}_i < 0), (i \in T_2 \text{ e } \bar{p}_i > 0) \}$$

O valor da variável x_r é então incrementado a partir de zero ($r \in T_1$) ou reduzido a partir de b_r ($r \in T_2$). Como $Q \in NSD$ então, pelo lema 3.1, também $\bar{Q} \in NSD$ e esse incremento ou decréscimo só pode ser bloqueado por uma variável x_s ($s \in F$) que entretanto tenha atingido um dos limites.

Assim se designarmos por

$$F_1 = \{ i \in F : \bar{n}_{ir} > 0 \}$$

$$F_2 = \{ i \in F : \bar{n}_{ir} < 0 \}$$

o critério para determinar x_s terá dois aspectos:

Caso 1 - Se $r \in T_1$, calcula-se s a partir de

$$\mu_0 = \begin{cases} \min \{ \dfrac{\alpha_i - \bar{q}_i}{\bar{n}_{ir}} : i \in F_1 \cup F_2 \} \\ +\infty \qquad \text{se } F_1 \cup F_2 = \emptyset \end{cases}$$

com

$$\alpha_i = \begin{cases} b_i \text{ se } i \in F_1 \\ 0 \text{ se } i \in F_2 \end{cases}$$

Se $\mu_0 \geq b_r$, então a variável x_r muda para o limite superior e faz-se

$$T_1 = T_1 - \{r\} \text{ e } T_2 = T_2 \cup \{r\}.$$

Caso contrário, efectua-se uma operação pivotal com pivot

$$\begin{bmatrix} \bar{q}_{rr} & -\bar{n}_{sr} \\ \bar{n}_{sr} & 0 \end{bmatrix} \tag{4.32}$$

e dois casos podem ocorrer:

(i) se $s \in F_1$, x_s passa a não básica com valor b_s e tem-se

$$F = F - \{s\} \cup \{r\}\ ,\ T_1 = T_1 - \{r\} \text{ e } T_2 = T_2 \cup \{s\}$$

(ii) se $s \in F_2$, x_s passa a não básica com valor 0, e tem-se

$$F = F - \{s\} \cup \{r\} \text{ e } T_1 = T_1 - \{r\} \cup \{s\}$$

Caso 2 - Se $r \in T_2$, calcula-se s a partir de:

$$\mu_0 = \begin{cases} \max \{ \dfrac{\bar{q}_i - \alpha_i}{-\bar{n}_{ir}} : i \in F_1 \cup F_2 \} \\ -\infty \quad \text{se } F_1 \cup F_2 = \emptyset \end{cases}$$

com

$$\alpha_i = \begin{cases} 0 & \text{se } i \in F_1 \\ b_i & \text{se } i \in F_2 \end{cases}$$

Se $-\mu_0 \geq b_r$, então a variável x_r muda para o limite inferior e faz-se

$$T_1 = T_1 \cup \{r\} \text{ e } T_2 = T_2 - \{r\}$$

Caso contrário, efectua-se uma operação pivotal principal com pivot (4.32) e dois casos podem ocorrer:

(i) se $s \in F_1$, x_s passa a não básica com valor 0 e tem-se

$$F = F - \{s\} \cup \{r\}\ ,\ T_1 = T_1 \cup \{r\} \text{ e } T_2 = T_2 - \{s\}$$

(ii) se $s \in F_2$, x_s passa a não básica com valor b_s, e tem-se

$$F = F - \{s\} \cup \{r\} \text{ e } T_2 = T_2 - \{r\} \cup \{s\}$$

Depois desta iteração ter sido executada, um novo quadro da forma (4.31) é obtido e o processo é repetido.

Se $Q \in NSD$ é simétrica e se apenas pivotagens não degeneradas forem usadas (isto é, $\bar{q}_s > 0$) então o valor da função quadrática do programa quadrático (4.29) decresce em cada iteração e o método termina num número finito de iterações. No caso de haver uma pivotagem degenerada, é fixada uma ordem para as variáveis e uma modificação da regra de Bland [GO86] para a escolha das variáveis x_r e x_s, segundo essa ordem, é suficiente para garantir a convergência.

Se $Q \in NSD$ é não simétrica, então o BLCP (4.30) é equivalente a um problema de desigualdades variacionais (LVI) do tipo:

Encontrar $\bar{x} \in S$ tal que

$$(x - \bar{x})^T (c + Q\bar{x}) \geq 0 \quad \text{para todo} \quad x \in S$$

desde que S seja limitado e não vazio [KIST80].

Neste caso o argumento que nos garante a convergência no caso simétrico não é válido. Contudo o método de Keller funciona exactamente da mesma maneira e conjecturamos a convergência do algoritmo para uma solução do LVI em todos os casos. Aliás a experiência computacional apresentada mais adiante confirma exactamente esta nossa conjectura.

4.2 - Implementação do Método de Keller

Tal como se viu na secção anterior, cada iteração do método de Keller tem associado um quadro da forma (4.31) com $\bar{Q} \in NSD$ e $\bar{q} \geq 0$. Contudo, para a aplicação do método é apenas necessário conhecer os vectores $\bar{q}$, $\bar{p}_{T_1}$ e $\bar{p}_{T_2}$ e a coluna $\bar{N}_{.r}$, da matriz $\bar{N}$, associada à variável x_r que entra na base. Mas todos os elementos do quadro (4.31) podem ser obtidos do BLCP (4.30) por uma operação pivotal principal cujo pivot é a matriz

$$\begin{bmatrix} Q_{FF} & -B^T \\ B & 0 \end{bmatrix}$$

Com efeito tem-se

$$\begin{bmatrix} \bar{q} \\ \bar{p}_F \end{bmatrix} = -\begin{bmatrix} Q_{FF} & -B^T \\ B & 0 \end{bmatrix}^{-1} \begin{bmatrix} c_F + Q_{FT_2} b_{T_2} \\ -d + N^2 b_{T_2} \end{bmatrix}$$

Assim, os vectores $\bar{q}$ e $\bar{p}_F$ são calculados resolvendo o sistema

$$\begin{bmatrix} Q_{FF} & -B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \bar{q} \\ \bar{p}_F \end{bmatrix} = - \begin{bmatrix} c_F + Q_{FT_2}\, b_{T_2} \\ -d + N^2\, b_{T_2} \end{bmatrix}$$

que é equivalente a resolver:

$$\begin{cases} B\ \bar{q} = -\ (-d + N^2\, b_{T_2}) \\ B^T\, \bar{p}_F = c_F + Q_{FT_2}\, b_{T_2} + Q_{FF}\, \bar{q} \end{cases} \tag{4.33}$$

Então

$$\begin{bmatrix} \bar{p}_{T_1} \\ \bar{p}_{T_2} \end{bmatrix} = \bar{p}_T = (c_T + Q_{TT_2}\, b_{T_2}) + [Q_{TF} : -N^T] \begin{bmatrix} \bar{q} \\ \bar{p}_F \end{bmatrix}$$

isto é

$$\bar{p}_T = c_T + Q_{TT_2}\, b_{T_2} + Q_{TF}\, \bar{q} - N^T\, \bar{p}_F \tag{4.34}$$

Finalmente para determinar a coluna $\bar{N}_{.r}$, tem-se

$$\bar{N}_{.r} = -\ B^{-1}\ N_{.r}$$

ou seja, $\bar{N}_{.r}$ é a solução do seguinte sistema linear:

$$B\ \bar{N}_{.r} = -\ N_{.r} \tag{4.35}$$

A implementação proposta consiste em considerar em cada iteração a decomposição LU da matriz B, usando um processo idêntico ao indicado para as implementações anteriores. Como cada operação pivotal consiste simplesmente em trocar uma coluna de B por uma coluna de N, o processo de actualização da decomposição LU, descrito anteriormente, é utilizado.

Como em cada iteração temos que ter acesso rápido a $N_{.r}$, a matriz esparsa A é armazenada por colunas na forma compacta [PI84]. Para a matriz Q é usado o mesmo esquema por colunas para facilitar os produtos (4.33) e (4.34). Note-se que com este esquema a matriz Q não é decomposta, o que permite resolver problemas de grandes dimensões de modo bastante eficiente.

O tipo de implementação descrita é particularmente recomendável para restrições da forma $Ax = d$. Se no conjunto S existirem t restrições de desigualdade $Ex \geq g$ ($E \in R^{txn}$, $g \in R^t$), estas são transformadas em igualdades introduzindo t variáveis de folga z_i. Então obtém-se um BLCP da forma

	1	x	z	θ_L	θ_T
y =	c	Q	0	$-A^T$	$-E^T$
w =	0	0	0	0	I
0 =	-d	A	0	0	0
0 =	-g	E	-I	0	0

(4.36)

com $z \geq 0$ e x a satisfazer as restrições (4.30). Esta transformação implica um aumento da dimensão do BLCP. Contudo, o armazenamento desta nova matriz esparsa é feita do mesmo modo que a anterior, pois como veremos, as colunas referentes às variáveis de folga não precisam de ser armazenadas.

Nesse processo de implementação as variáveis x são numeradas de 1 até n. Na Fase 1 são introduzidas as variáveis n+1 até n+m+t, onde t é o número de variáveis de folga e m o de variáveis artificiais (que serão eliminadas se possível ao longo dessa Fase 1). Com este processo de numeração das variáveis, é possível gerar a informação necessária do quadro (4.36). Se a variável tem a numeração $i \leq n$, temos que considerar a coluna i de Q, A e E. Se a variável tem a numeração $i > n$, a única componente diferente de zero (= -1) é a correspondente à restrição $i - n$ do conjunto S. Isto permite que as restrições de igualdade e desigualdade estejam dispostas por qualquer ordem, existindo um vector $IS \in R^{m+t}$ para identificar o tipo de restrição. Se IS(i) = 0 a restrição i é de igualdade e se $IS(i) \neq 0$ a restrição i é do tipo $\leq$.

Tal como anteriormente é usado um vector IVAR para guardar a informação das variáveis básicas e não básicas. As variáveis básicas estão guardadas nas primeiras M = m+t componentes de IVAR e as variáveis não básicas estão guardadas nas componentes M+1 até M+NC (se não existirem variáveis artificiais básicas, isto é, não houver restrições redundantes, M+NC = t + n). As últimas NB componentes de IVAR correspondem a variáveis não básicas com valores iguais aos limites superiores. É assim possível percorrer rapidamente as componentes pertencentes a T_2, o que facilita a obtenção da informação necessária ao cálculo das fórmulas (4.33 - 4.35). Para uma maior facilidade nesses cálculos, considera-se ainda um vector IPT com n componentes, tal que IPT(i) representa a componente de IVAR que guarda a variável x_i. Este vector é bastante

útil para identificar se uma variável x é ou não básica e sendo não básica se está no limite inferior ou superior. Assim se IPT(i) $\leq$ M então i $\in$ F e se IPT(i) > M tem-se i $\in$ T. Neste último caso i $\in T_2$ se IPT(i) > M + NC – NB e i $\in T_1$ no caso contrário. Usando estes vectores é possível implementar os produtos das fórmulas (4.33 - 4.35) de modo a evitar que os blocos nulos sejam considerados. Isso torna o esforço computacional idêntico para os casos das igualdades e desigualdades.

4.3 - Experiência Computacional

Nesta secção apresentamos alguma experiência computacional do método de Keller na resolução de QPs e LVIs côncavos com limites superiores nas variáveis.

Dois tipos de problemas-teste foram considerados. No primeiro grupo de testes, PT1, PT2 e PT3 foram retirados de [GO86], multiplicando a matriz da forma quadrática por –1 para que Q $\in$ NSD. Os outros problemas-teste PT4, PT5, PT6 e PT7 foram construídos usando a técnica descrita em [ST80] para gerar a matriz –Q, enquanto que a matriz esparsa A $\in R^{mxn}$ das restrições do conjunto S é gerada de modo idêntico ao descrito na secção 2.8. Os limites superiores foram gerados aleatoriamente.

Para construir LVIs com matrizes NSD não simétricas alguns dos elementos simétricos a_{ij} e a_{ji} dos problemas-teste PT1-PT7 foram modificados de modo a que $a'_{ij} + a'_{ji} = a_{ij} + a_{ji}$, mas com $a'_{ij} \neq a'_{ji}$.

Os testes foram executados num CDC CYBER 180-830 da Universidade do Porto, apresentando-se os resultados no quadro 4.4. Nesse quadro usam-se as seguintes notações:

m = número de restrições

n = número de variáveis

$$r(A) = \frac{\text{número de não zeros de A}}{n}$$

$$r(Q) = \frac{\text{número de não zeros de Q}}{n}$$

NI = número de iterações

T = tempo de CPU em segundos

Os resultados apresentados neste quadro mostram a eficiência da implementação para resolver QPs e LVIs côncavos com matrizes simétricas e não simétricas.

				Fase 1		Método de Keller					
						Q simétrica			Q não simétrica		
PT	m	n	r(A)	NI	T	r(Q)	NI	T	r(Q)	NI	T
1	299	599	1.5	299	15.2	3.	300	20.2	3.	594	51.4
2	399	799	1.5	399	27.2	3.	398	36.5	3.	794	90.
3	499	999	1.5	499	42.	3.	498	56.	3.	994	142.
4	50	500	6.6	455	49.4	6.2	630	120.	4.9	644	119.
5	80	500	4.9	637	90.	6.2	877	216.	4.9	850	210.
6	50	600	6.6	602	77.4	9.3	1102	244.	7.2	1076	230.
7	80	600	4.9	786	122.	9.3	1348	393.	7.2	1284	357.

Quadro 4.4

Como seria de esperar constata-se que um aumento da densidade e dimensão da matriz A é mais importante na eficiência do processo do que um tal aumento na matriz Q. Isto confirma a vantagem desta implementação na resolução de QPs e LVIs com um grande número de variáveis e um número razoável de restrições. Além disso, como com este tipo de implementação a matriz Q não necessita de ser decomposta, é de esperar o desenvolvimento de outras implementações para programas quadráticos e LVIs côncavos com restrições de estruturas especiais, tais como problemas de redes e de transportes [MU83]. Uma outra aplicação deste tipo de processo é o da resolução de programas ou desigualdades variacionais côncavos não lineares por uma sequência de programas quadráticos ou LVIs côncavos [GIMUWR81, MAT85]. Se as restrições desses problemas não lineares forem lineares, então esses QPs e LVIs têm o mesmo conjunto de restrições. Por isso a matriz Base correspondente a uma solução da sequência pode ser usada como solução inicial do problema seguinte. Elimina-se assim a necessidade da resolução da Fase 1 que, como se pode ver no quadro 4.4, exige um esforço computacional elevado.

5 - Método Enumerativo

Nesta secção iremos apresentar algumas propostas para um possível método enumerativo capaz de resolver o BLCP em todos os casos. Como vimos anteriormente, se todos os limites inferiores e superiores forem finitos então o BLCP é equivalente ao LCP

$$\begin{aligned} w^1 &= q + Mz + I_n\, w^2 \\ v &= b - I_n \\ & v,\ w^1,\ z,\ w^2 \geq 0 \\ & z^T w^1 = v^T w^2 = 0 \end{aligned} \qquad (4.37)$$

O método enumerativo descrito no capítulo 2 pode então ser aplicado a esse LCP. Contudo, devido à natureza especial da matriz do LCP, várias simplificações são possíveis nesse algoritmo.

Considere-se a seguinte partição de $\{1,\ldots, n\}$

$$F = \{i:\ 0 < z_i < b_i\}$$

$$T_1 = \{i:\ z_i = 0\}$$

$$T_2 = \{i:\ z_i = b_i\}$$

Para obter uma solução básica inicial podemos simplesmente escrever

$$T_1 = \{i:\ q_i \geq 0\}\ ,\ T_2 = \{i:\ q_i < 0\}\ \text{ e }\ F = \emptyset$$

Depois calcula-se

$$\bar{q} = q + \sum_{i \in T_2} M_{.i}\, b_i$$

e faz-se:

$$\begin{cases} \text{se } \bar{q}_j < 0 \text{ então } w_j^2 \text{ é básica} \\ \text{se } \bar{q}_j \geq 0 \text{ então } w_j^1 \text{ é básica} \end{cases} \qquad (4.38)$$

Como $w = q + Mz$ e $0 \leq z \leq b$, este algoritmo fornece uma solução admissível, tanto do BLCP, como do LCP (4.37) que não é complementar, nos seguintes casos:

(i) existe $j \in T_1$ tal que $\bar{q}_j < 0$

(ii) existe $j \in T_2$ tal que $\bar{q}_j > 0$

Sempre que uma solução admissível do LCP (4.37) não é complementar, teremos, tal como anteriormente, que obrigar a que uma das variáveis complementares seja nula.

Como neste caso existem relações simples entre as variáveis z_i, v_i e w_i^1, w_i^2 convém estudar as marcações de z_i, w_i^1 e v_i, w_i^2 simultaneamente e ver o que é que isso significa no BLCP.

Notemos que se verifica a seguinte correspondência entre o LCP (4.37) e o BLCP:

$$\begin{array}{ll} w_i^1 = 0 & w_i \leq 0 \\ z_i = 0 & z_i = 0 \\ v_i = 0 & z_i = b_i \\ w_i^2 = 0 & w_i \geq 0 \end{array}$$

Portanto os ramos da árvore para a resolução do LCP

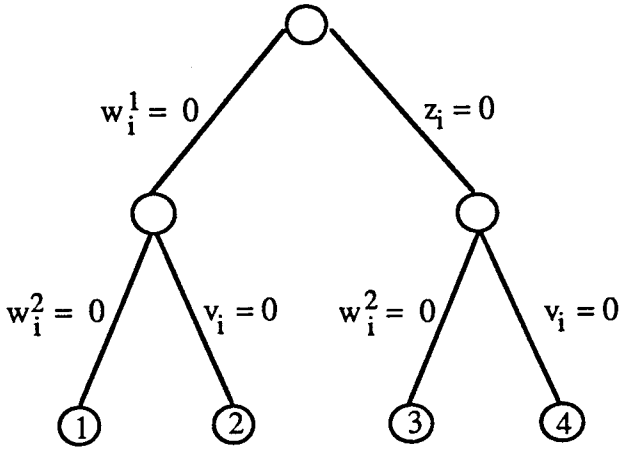


Figura 4.1

têm a seguinte forma para o BLCP:

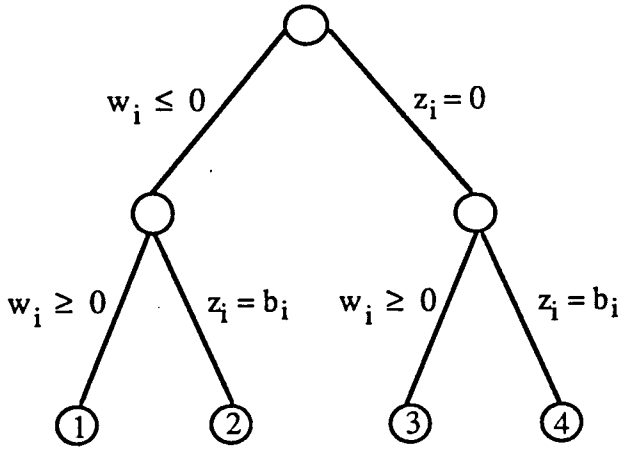


Figura 4.2

Mas pela análise da figura 4.2 e como $b_i \neq 0$, o ramo 4 é inadmissível. Portanto cada nó dá lugar aos três seguintes ramos:

$$R1 — w_i = 0$$
$$R2 — z_i = b_i \text{ e } w_i \leq 0$$
$$R3 — z_i = 0 \text{ e } w_i \geq 0$$

A questão que agora se põe é como se vão obter cada uma das soluções admissíveis destes três ramos. Isso deverá obrigar a tomar diferentes vias consoante o tipo de solução básica associada a cada nó e ramo pretendido. Esses caminhos obrigarão a maximizar, ou minimizar z_i ou w_i. Nalguns desses subproblemas w_i passa a ser uma variável com limites de um dos tipos $w_i \geq 0$ ou $w_i \leq 0$. Além disso $0 \leq z_i \leq b_i$ para todo $i = 1,\ldots, n$. Isto obrigará a marcações um pouco mais complicadas do que as utilizadas no método enumerativo para solução do LCP, pois os programas lineares a resolver terão que ter em conta esses limites.

Quanto à determinação da variável não básica escolhida para passar a básica usar-se-á uma técnica semelhante à do método simplex com limites superiores [MU76, cap. 10]. Nesse processo uma variável não básica só será candidata a entrar na base se está no limite inferior e esse custo reduzido for negativo ou se está no limite superior e o custo reduzido for positivo.

Um possível método enumerativo para o BLCP terá obviamente que incorporar um algoritmo do tipo MRG para diminuir a pesquisa em árvore. Esse é um dos pontos de investigação futura e será determinante no desenvolvimento de um algoritmo eficiente para a resolução do BLCP em todos os casos.

CAPÍTULO 5

RESOLUÇÃO DO PROBLEMA LINEAR COMPLEMENTAR COM UMA FUNÇÃO LINEAR A MINIMIZAR (MLCP)

1 - Definição do MLCP

O Problema Linear Complementar com uma função linear a minimizar (MLCP) foi introduzido por Kirchgässner [KI62] tendo sido depois estudado com maior detalhe em [JU82]. Como referimos no capítulo 1 este problema pode ser definido da seguinte forma

$$\begin{aligned} &\text{minimizar} && g(x, y) = c^T z + d^T y \\ &\text{sujeito a} && w = a + Mz + Ny \\ & && v = b + Ez + Dy \\ & && z, w, y, v \geq 0 \\ & && z^T w = 0 \end{aligned} \qquad (5.1)$$

onde $a, c, z, w \in R^n$, $d, y \in R^t$, $v, b \in R^{\ell}$, $M \in R^{n \times n}$, $N \in R^{n \times t}$, $E \in R^{\ell \times n}$ e $D \in R^{\ell \times t}$. Portanto este tipo de problema consiste na minimização de uma função linear sujeita a um conjunto de restrições que constituem um LCP ou GLCP. Como o conjunto de restrições é não convexo é bastante difícil determinar uma solução óptima deste problema. Aliás o MLCP é em geral um problema NP-completo, pois o mesmo acontece ao GLCP. A importância deste tipo de problema advém do facto de vários problemas de optimização global se poderem resolver mais ou menos eficientemente explorando a sua equivalência a MLCPs. Esses problemas serão discutidos em capítulos posteriores desta tese.

É interessante notar que o MLCP pode ser um problema bastante simples em alguns casos especiais. Assim, por exemplo, se $t = \ell = 0$ e $M \in PD$, então o MLCP pode ser resolvido polinomialmente. Com efeito neste caso o conjunto de restrições do MLCP consiste apenas de um ponto, que como vimos anteriormente pode ser determinado por

um método polinomial. Em geral o conjunto de restrições é constituído por mais do que um ponto e é apenas finito em alguns casos especiais. Contudo, mesmo nesses casos o esforço computacional para determinar todas as soluções do GLCP é proibitivo. Daí a necessidade de desenvolver métodos que permitam resolver o MLCP sem tentar obter todas as soluções do correspondente GLCP. Nesta tese iremos apresentar dois processos enumerativos de pesquisa em árvore que permitem resolver pelo menos em teoria o MLCP em todos os casos. Esses processos serão depois usados em capítulos posteriores para obter soluções óptimas de alguns problemas de optimização global bastante conhecidos na literatura. Como nota final desta introdução gostaríamos de salientar a importância do desenvolvimento de algoritmos directos ou iterativos capazes de resolver o MLCP em alguns casos particulares. Essa será certamente uma das áreas de investigação futura.

2 - Método de Pesquisa em Árvore com Limites

Este processo foi inicialmente proposto por Ibaraki [IB71] para um caso especial do MLCP equivalente a um programa linear 0 –1, tendo depois sido desenvolvido em [JU82] para o MLCP geral. Nesse algoritmo é primeiramente obtida uma solução óptima do programa linear RLP, que se obtém do MLCP por relaxação da condição de complementaridade. Usando depois soluções primais ou duais admissíveis do RLP, uma árvore binária do tipo da figura 2.1 do capítulo 2 é explorada até se obter uma solução óptima do MLCP. Sempre que uma solução complementar $(\overline{z}, \overline{y})$ do MLCP (isto é, uma solução do seu conjunto de restrições) é obtida, tal solução é denominada incumbente e é armazenada. Todos os nós da árvore para os quais o valor da função linear é superior ou igual ao valor da função da incumbente $g_{inc} = g(\overline{z}, \overline{y})$ são então desprezados. O processo continua obtendo sucessivamente novas incumbentes e eliminando nós segundo este processo, até se ter a garantia que não há nós a investigar. Os passos do algoritmo de pesquisa em árvore com limites são os seguintes:

ALGORITMO DE PESQUISA EM ÁRVORE COM LIMITES

Passo 0 – Seja $k = 0$ e $g_{inc} = +\infty$.

Passo 1 – Resolva o programa linear

$$\begin{aligned} &\text{minimizar} && c^T z + d^T y \\ &\text{sujeito a} && w = a + Mz + Ny \\ & && v = b + Ez + Dy \\ & && z, w, y, v \geq 0 \\ & && z_{i_t} = 0 \text{ ou } w_{i_t} = 0 \quad \text{para} \quad t = 1,\ldots, k-1 \end{aligned} \tag{5.2}$$

Se o programa linear é não admissível, vá para Passo 5. De outro modo seja $x^k = (z^k, w^k, y^k, v^k)$ a solução óptima desse programa linear.

Passo 2 – Se $g(z^k, y^k) \geq g_{inc}$, vá para Passo 5. De outro modo, vá para Passo 3.

Passo 3 – Se x^k é complementar ($z_i^k w_i^k = 0$, $i = 1,\ldots, n$), então (z^k, y^k), é uma solução incumbente. Actualize

$$g_{inc} = g(z^k, y^k)$$

e vá para Passo 5. De outro modo, vá para Passo 4.

Passo 4 – Seja (z_{s_k}, w_{s_k}) um par de variáveis complementares positivas e ramifique a árvore, minimizando as variáveis z_{s_k} e w_{s_k} sujeitas ao conjunto de restrições do programa linear (5.2). Se o mínimo de alguma dessa variáveis é positivo, então o nó respectivo não pode ser gerado.

Passo 5 – Se todos os nós gerados anteriormente já tiverem sido explorados, vá para EXIT. De outro modo escolha um nó do nível k da árvore e vá para Passo 1.

EXIT – Se $g_{inc} = +\infty$, o MLCP não tem solução. De outro modo a última solução incumbente é a solução óptima do MLCP.

A experiência computacional [JUFA91b, JUFA91d] com este algoritmo sugeriu que no Passo 5 se deve escolher o nó para o qual $g(z^k, w^k)$ é mínimo. Este tipo de regra heurística conduz normalmente à obtenção de "boas" soluções incumbentes que vão permitir podar vários ramos, tornando assim mais reduzida a pesquisa em árvore. Além

disso no Passo 4 é conveniente escolher o par (z_s, w_s) para o qual é máximo o valor de $z_i w_i$.

A resolução do programa linear (5.2) pode ser feita usando os métodos primal ou dual simplex. O uso destes dois tipos de processos é discutido em [JU82], tendo sido mostrado que a escolha do método dual simplex é normalmente vantajosa pelo menos para os problemas-teste resolvidos nesse trabalho. Com efeito, a geração de nós é muito mais simples se o método dual simplex for usado. Se o método primal simplex é escolhido, então o processo MINVAR descrito no capítulo 2 é normalmente usado para gerar os nós no Passo 4.

Em alguns casos de utilização do MLCP em optimização global, o problema linear RLP é admissível mas não tem solução óptima, isto é, a função linear tende para $-\infty$. Essa desvantagem pode ser ultrapassada introduzindo uma restrição

$$e^T z + e^T y \leq \tau \qquad (5.3)$$

com $e = (1,\ldots, 1)^T$ e τ um número bastante grande [JU82]. Contudo a inclusão desse tipo de restrição acarreta dificuldades computacionais na resolução do MLCP [JU82]. Um outro processo de ultrapassar este problema consiste em obter um limite inferior LB para o MLCP [JUFA91d] e introduzir a restrição

$$g(z, y) \geq LB \qquad (5.4)$$

Veremos mais adiante como é possível determinar esse limite inferior em algumas aplicações do MLCP na resolução de problemas de optimização global.

3 - Método Sequencial LCP (SLCP)

Este processo foi inicialmente introduzido por Bialas e Karwan [BIKA84] para um caso especial do problema de dois níveis. Neste método é introduzido um parâmetro λ e a função objectivo é substituída pela restrição

$$c^T z + d^T y \leq \lambda \qquad (5.5)$$

obtendo-se o seguinte problema linear complementar

LCP(λ)

$$\begin{bmatrix} w \\ v \\ v_0 \end{bmatrix} = \begin{bmatrix} a \\ b \\ \lambda \end{bmatrix} + \begin{bmatrix} M & N \\ E & D \\ -c^T & -d^T \end{bmatrix} \begin{bmatrix} z \\ y \end{bmatrix} \qquad (5.6)$$

$$w, v, v_0, z, y \geq 0 \qquad w^T z = 0$$

O algoritmo sequencial consiste em encontrar uma solução do LCP($\bar{\lambda}$), onde $\bar{\lambda}$ é o menor valor de λ, tal que LCP(λ) tem uma solução. Para encontrar essa solução, o método começa por resolver o LCP(λ_0) obtido do LCP(λ) (5.6), omitindo a restrição $c^T z + d^T y \leq \lambda$. Seja (z^0, y^0) a solução do LCP(λ_0) e $\lambda_0 = g(z^0, y^0) = c^T z^0 + d^T y^0$. O algoritmo resolve então uma sequência de LCPs(λ_k), onde $\{\lambda_k\}$ é uma sucessão decrescente definida por

$$\lambda_k = g(z^{k-1}, y^{k-1}) - \gamma\, |\, g(z^{k-1}, y^{k-1}) \,| \qquad (5.7)$$

com (z^k, y^k) solução do LCP(λ_k) e γ um número positivo pequeno. O método termina numa iteração k para a qual LCP(λ_k) não tem solução. Quando isso acontece, a solução (z^{k-1}, y^{k-1}) do LCP(λ_{k-1}) satisfaz

$$0 \leq g(z^{k-1}, y^{k-1}) - g(\bar{z}, \bar{y}) \leq \gamma\, |\, g(z^{k-1}, y^{k-1}) \,| \qquad (5.8)$$

onde $(\bar{z}, \bar{y}, \bar{w}, \bar{v})$ é a solução óptima do MLCP (5.1).

Assim, se o conjunto de soluções admissíveis do MLCP é não vazio e limitado, o algoritmo SLCP encontra uma solução ε-óptima do MLCP com

$$\varepsilon = \gamma\, |\, g(z^{k-1}, y^{k-1}) \,| \qquad (5.9)$$

Na prática, se γ for suficientemente pequeno, a solução (z^{k-1}, y^{k-1}) do último LCP(λ_{k-1}) é normalmente a solução óptima do MLCP. Em capítulos posteriores discutiremos o valor que γ deve tomar na prática.

Os passos do algoritmo SLCP são os seguintes:

ALGORITMO SLCP

Passo 0 – Faça $k = 0$

Passo Geral – Resolva o LCP(λ_k). Se LCP(λ_k) não tem solução, vá para EXIT. Caso contrário, seja (z^k, y^k, w^k, v^k) a solução do LCP. Faça

$$\lambda_{k+1} = g(x^k, y^k) - \gamma \mid g(z^k, y^k) \mid$$

onde γ é um valor previamente fixado. Faça $k = k + 1$ e repita o processo.

EXIT – Se $k = 0$, o MLCP não tem solução. Caso contrário, (z^{k-1}, y^{k-1}) é uma solução ε-óptima do MLCP, onde ε é dado por (5.9).

A eficiência do algoritmo SLCP depende essencialmente do algoritmo para resolver LCP(λ_k). O método enumerativo híbrido descrito no capítulo 2 serve perfeitamente os nossos propósitos. Em alguns casos de aplicação do MLCP em optimização global é possível usar um método directo ou iterativo para resolver o LCP(λ_0). Esse assunto será discutido em capítulos posteriores.

É de notar que, quando se pretende resolver o LCP(λ_k), a solução do LCP(λ_{k-1}) já é conhecida e pode por isso ser usada como solução inicial para o LCP(λ_k). Com efeito LCP(λ_{k-1}) e LCP(λ_k) só diferem na última componente do vector dos termos independentes

$$q = \begin{bmatrix} a \\ b \\ \lambda_k \end{bmatrix}$$

Por isso se B é a base associada à solução $(z^{k-1}, y^{k-1}, w^{k-1}, v^{k-1}, v_0^{k-1})$ do LCP(λ_{k-1}) e

$$\bar{q} = B^{-1}q$$

então $\bar{q}_j \geq 0$ para todo $j = 1, \ldots, n + \ell$ e $\bar{q}_{n+\ell+1} < 0$. Portanto, para a obtenção da solução inicial admissível, o método enumerativo híbrido tem de resolver o problema

minimizar z_0

sujeito a

$$\begin{bmatrix} w \\ v \\ v_0 \end{bmatrix} = \begin{bmatrix} a \\ b \\ \lambda_k \end{bmatrix} + \begin{bmatrix} M & N \\ E & D \\ -c^T & -d^T \end{bmatrix} \begin{bmatrix} z \\ y \end{bmatrix} + p\, z_0 \qquad (5.10)$$

no qual o vector p é definido por

$$p = B \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad (5.11)$$

onde $0 \in R^{n+\ell}$ é o vector nulo.

A escolha desta solução básica inicial provoca normalmente uma grande redução no esforço computacional do método enumerativo híbrido. De facto, o algoritmo requer com frequência um reduzido número de operações pivotais para encontrar a solução do LCP(λ_k) [JUFA88b, JUFA88c].

Para aumentar a eficiência deste algoritmo, foram ainda introduzidas duas modificações ao método sequencial atrás descrito e que são discutidas a seguir.

Quando uma solução do LCP(λ_k) é encontrada, procura-se uma nova solução do LCP(λ_k) tal que $c^T z + d^T y$ tenha um valor inferior. Para isso procura maximizar-se a variável v_0 de tal modo que a condição de complementaridade $z^T w = 0$ se mantenha satisfeita em cada iteração. Isso é realizado com a maximização de v_0 usando o método BRES descrito no capítulo 2. Assim, o método simplex é usado para maximizar v_0, mas em cada iteração a variável a entrar na base é escolhida de modo a que a sua complementar seja também não básica. Este algoritmo, que denotaremos por MAXVAR, pode terminar de dois modos possíveis:

i) se foi encontrado o valor máximo da variável v_0, sujeito às restrições lineares do LCP(λ_k), obteve-se a solução óptima global do MLCP e o algoritmo SLCP termina (TERM = 1);

ii) a variável v_0 pode ainda ser incrementada mas com a destruição da condição de complementaridade, isto é, são básicas todas as complementares das variáveis não básicas com custos reduzidos

positivos. Neste caso, λ é redefinido por (5.7) e outro LCP tem que ser resolvido no algoritmo SLCP (TERM = 2).

Como foi discutido em [JUFA88c] e pode ser confirmado por toda a experiência computacional apresentada nos próximos capítulos, a resolução dos últimos LCPs da sucessão exige geralmente um esforço computacional muito superior aos restantes. Com a inclusão do algoritmo MAXVAR consegue-se normalmente melhorar o comportamento global do método [JUFA91d] mas sem afectar a resolução do último LCP. Assim, a maior parte do esforço computacional de todo o método é gasto na resolução desse último LCP. Para alguns tipos de problemas [JUFA91b] uma forma de ultrapassar este problema consiste em reduzir o valor de λ um pouco mais e aplicar novamente o algoritmo enumerativo híbrido. Na prática isso é conseguido limitando o número de operações pivotais que a solução do LCP pode exigir através de uma quantidade NMAXPV fornecida pelo utilizador. Se esse número for atingido, λ_k é modificado para

$$\lambda_k = g(z^{k-1}, w^{k-1}) - \bar{\gamma} \mid g(z^{k-1}, w^{k-1}) \mid \qquad (5.12)$$

onde $\bar{\gamma}$ é uma constante positiva maior do que γ. Então resolve-se o LCP(λ_k), começando com a solução básica corrente, mas sem limite no número de pivotagens. Se este LCP não tiver solução, a última solução obtida pelo algoritmo SLCP é uma solução $\bar{\varepsilon}$-óptima, onde

$$\bar{\varepsilon} = \bar{\gamma} \mid g(z^{k-1}, y^{k-1}) \mid \qquad (5.13)$$

Assim, o algoritmo que se aplica em cada iteração do processo SLCP tem os seguintes passos:

Passo 1 – Seja NMAX = NMAXPV, onde NMAXPV é um número positivo escolhido pelo utilizador.

Passo 2 – Aplique o método enumerativo híbrido para resolver LCP(λ_k). Se o número de operações pivotais requeridas pelo método enumerativo ultrapassar NMAX, actualize λ_k a partir (5.12) e faça NMAX = $+\infty$. Caso contrário, o método enumerativo termina e vá para o Passo 3.

Passo 3 – Se LCP(λ_k) não tem solução, faça TERM = 3 e vá para EXIT. Caso contrário, aplique o algoritmo MAXVAR e vá para EXIT.

EXIT – Se TERM = 1, a solução (obtida pelo algoritmo MAXVAR) é um óptimo global do MLCP. Se TERM = 2, faça k = k + 1 e nova iteração do

algoritmo SLCP é executada com λ_k definido por (5.7). Se TERM = 3, a solução obtida na iteração k–1 do algoritmo SLCP é uma solução ε ou $\bar{\varepsilon}$-óptima para MLCP, consoante NMAX = NMAXPV ou NMAX > NMAXPV, respectivamente.

A experiência computacional que é apresentada em capítulos posteriores mostra que estas modificações tornam o método SLCP mais robusto e capaz de resolver de modo eficiente alguns problemas de optimização global explorando a sua redução a um MLCP.

Como afirmámos anteriormente os dois algoritmos discutidos neste capítulo são bastante gerais, pois permitem resolver o MLCP sem qualquer tipo de propriedade especial das matrizes envolvidas. Nos capítulos posteriores iremos discutir a eficiência desses processos na resolução de alguns MLCPs que são reformulações de problemas de optimização global bastante conhecidos.

CAPÍTULO 6

PROGRAMAÇÃO BILINEAR

1 - Definição do Problema

O Problema de Programação Bilinear (BLP) é normalmente definido da seguinte forma:

$$\begin{aligned} &\text{minimizar} \quad c^T x + d^T y + x^T C y \\ &\text{sujeito a} \quad x \in S_1,\ y \in S_2 \end{aligned} \tag{6.1}$$

com

$$\begin{aligned} S_1 &= \{x \in R^{n_1} : Ax \geq a,\ x \geq 0\} \\ S_2 &= \{y \in R^{n_2} : Dy \geq g,\ y \geq 0\} \end{aligned} \tag{6.2}$$

e $x, c \in R^{n_1}$, $d, y \in R^{n_2}$, $A \in R^{r_1 \times n_1}$, $D \in R^{r_2 \times n_2}$ e $C \in R^{n_1 \times n_2}$. Este problema foi inicialmente estudado em [KON71] onde foram apresentadas algumas das suas mais importantes aplicações nos domínios da engenharia, economia e na resolução de outros problemas de optimização. Uma característica interessante deste problema reside no facto das restrições nas variáveis x e y da função bilinear estarem separadas em dois conjuntos S_1 e S_2 respectivamente. Além disso é possível provar o seguinte resultado [KON76]:

Teorema 6.1 – Se o BLP tem uma solução óptima, então existe uma solução óptima (x^*, y^*) tal que x^* e y^* são pontos extremos de S_1 e S_2 respectivamente.

Baseado neste teorema é possível desenvolver um algoritmo simplex para a determinação de um ponto estacionário de um BLP. Esse algoritmo é devido a Konno [KON76] e consiste em resolver uma sucessão de programas lineares em x e em y. Os passos desse processo são apresentados a seguir.

ALGORITMO SPL

Passo 0 – Seja $\overline{y}$ uma solução admissível de S_2. Se $\overline{y}$ não existir o BLP é não admissível e termine.

Passo 1 – Resolva o programa linear

$$\begin{aligned} \text{PL1:} \quad & \text{minimizar} \quad (c + C\overline{y})^T x \\ & \text{sujeito a} \quad x \in S_1 \end{aligned} \tag{6.3}$$

Se PL1 é não admissível então o BLP é não admissível e termine. Se PL1 é ilimitado o mesmo acontece ao BLP e termine. Se os casos anteriores não ocorrerem seja x* uma solução óptima de PL1.

Passo 2 – Se $\overline{x}$ não está definido faça $\overline{x} = x^*$ e vá para Passo 3.
Se $x^* = \overline{x}$ então $(\overline{x}, \overline{y})$ é um ponto estacionário do BLP e termine. Se $x^* \neq \overline{x}$ faça $\overline{x} = x^*$ e vá para Passo 3.

Passo 3 – Resolva o programa linear

$$\begin{aligned} \text{PL2:} \quad & \text{minimizar} \quad (d + C^T\overline{x})^T y \\ & \text{sujeito a} \quad y \in S_2 \end{aligned} \tag{6.4}$$

Se PL2 é ilimitado o mesmo acontece com o BLP e termine. De outro modo seja y* uma solução óptima desse programa linear.

Passo 4 – Se $y^* = \overline{y}$, então $(\overline{x}, \overline{y})$ é um ponto estacionário do BLP e termine. De outro modo faça $\overline{y} = y^*$ e volte ao Passo 1.

A resolução dos programas lineares PL1 e PL2 pode ser feita usando o método simplex ou um método de pontos-interiores. Parece-nos contudo mais razoável usar o primeiro tipo de processos, porque na sucessão de programas lineares PL1 e PL2 os conjuntos de restrições são os mesmos. Assim, a solução básica óptima do programa PL1(PL2) numa iteração pode ser a solução inicial para o mesmo tipo de programa na iteração a seguir. Esta característica de usar uma Base Avançada não é utilizável pelos métodos de pontos-interiores e daí a nossa opção pelo método simplex.

Se a determinação de um ponto estacionário de um BLP é um problema relativamente simples, já o mesmo não se pode dizer da obtenção de um seu mínimo global. Com efeito o BLP é um problema NP-Completo [HAJASA89] a não ser em alguns casos especiais. Um desses casos é aquele em que o conjunto S_1 ou S_2 é um simplex generalizado, onde é possível obter o mínimo global do BLP em tempo polinomial, conforme se mostra a seguir.

Seja S_1 um simplex generalizado da forma

$$S_1 = \{x \in R^{n_1} : e^T x \leq a_0,\ x \geq 0\} \tag{6.5}$$

com a_0 um número positivo e $e = (1,\ldots, 1)^T$. Então S_1 contém apenas $n_1 + 1$ pontos extremos x^k. Portanto se resolvermos os $n_1 + 1$ LPs da forma

$$\begin{aligned} \text{PL(k):}\quad & \text{minimizar} \quad c^T x^k + (d + C^T x^k) y \\ & \text{sujeito a} \quad y \in S_2 \end{aligned} \tag{6.6}$$

obtemos um mínimo global do BLP. Como cada LP(k) pode ser resolvido em tempo polinomial [KA84], então concluímos da veracidade da nossa afirmação.

Este tipo de BLP [JUFA91d] é designado por BLP no simplex generalizado (GSBLP). Um processo simples de obter esta solução baseia-se no uso sequencial do método simplex, aproveitando a base de um dado LP para base de arranque do LP seguinte, conforme ilustra o algoritmo seguinte:

ALGORITMO BLSIMP

Passo 0 – Seja B_0 uma base admissível de S_2. Se B_0 não existe, faz-se TERM = 0 e vá para EXIT. Caso contrário, faça VAL =+∞ e k = 1.

Passo Geral – Seja x^k o ponto extremo do conjunto S_1 (6.5), obtido fixando $x_j = 0$ para todo $j \neq k$. Resolva o PL(k) usando o método simplex com base inicial B_{k-1}. Seja B_k a base da solução óptima y^k obtida e

$$VAL(k) = c^T x^k + (d + C^T x^k)^T y^k$$

Se VAL(k) < VAL, então VAL = VAL(k) e $(\bar{x} = x^k, \bar{y} = y^k)$. Se $k = n_1 + 1$, vá para EXIT. Caso contrário, faça k = k + 1 e repita o passo geral.

EXIT – Se TERM = 0, o GSBLP é não admissível. De outro modo $(\bar{x}, \bar{y})$ é a solução óptima do BLP e VAL é o valor da função objectivo nesse ponto.

Os algoritmos mais recomendados na literatura para a determinação de um mínimo global do BLP são processos de planos de corte e de pesquisa de pontos extremos [GAUL77, KON76, SHSH80, VASH77]. O algoritmo considerado mais importante é devido a Konno [KON76]. Trata-se de um método sequencial, que garante a obtenção de um ε-óptimo, ponto extremo de S_1 e S_2, num número finito de passos. Para tal, vai gerando sucessivamente cortes de três tipos

$$\text{i)} \quad b^T x \le \alpha_x$$

$$\text{ii)} \quad p^T y \le \alpha_y$$

$$\text{iii)} \quad b^T x + p^T y \le \alpha$$

que eliminam a solução previamente obtida. As restrições do tipo i) e ii), conjuntamente com as que definem S_1 e S_2, são utilizadas para determinar um ponto estacionário pelo algoritmo SPL. Faz-se então uma busca entre os vértices adjacentes dum chamado "Local Maximum Vertex". Sempre que novos cortes são gerados é feito um teste de optimalidade que lhe permite identificar soluções ε-óptimas do BLP. Apesar de prever a eliminação de cortes redundantes, o número de cortes vai aumentando, e o método, tal como os outros métodos deste tipo, tem tendência a perder eficiência e a tornar-se pouco estável. Assim, estes algoritmos só funcionam relativamente bem para BLPs de dimensões muito reduzidas, não se encontrando, em nenhum dos trabalhos referidos atrás, experiência computacional com BLPs de dimensões razoáveis. Em [JU82] foi pela primeira vez utilizada a complementaridade linear para obter o mínimo global do BLP. Neste capítulo iremos apresentar os resultados das nossas tentativas em seguir essa via.

2 - Redução de um BLP a um MLCP

Tal como é referido em [JUMI88b], o BLP formulado em (6.1) pode ser escrito na forma:

$$\min_y \{ d^T y + g(y) : y \in S_2 \} \tag{6.7}$$

com

$$g(y) = \min_x \{ (c + Cy)^T x : x \in S_1 \} \tag{6.8}$$

O dual do programa linear (6.8) tem a seguinte forma:

$$\begin{aligned} \max_{u} \quad & a^T u \\ \text{sujeito a} \quad & A^T u \le c + Cy \\ & u \ge 0 \end{aligned} \tag{6.9}$$

Se o programa linear (6.8) tem uma solução óptima para qualquer $y \in S_2$, então, da teoria da dualidade da programação linear, x é uma solução óptima do programa (6.8) se e só se se verificam as seguintes condições:

$$\alpha = c + Cy - A^T u$$

$$\beta = -a + Ax$$

$$\alpha, \beta, u, y, x \ge 0$$

$$\alpha^T x = \beta^T u = 0$$

Além disso

$$g(y) = a^T u$$

Portanto o BLP (6.1) é equivalente ao MLCP

$$\text{minimizar} \quad h(y, u) = d^T y + a^T u$$

$$\text{sujeito a} \quad \begin{bmatrix} \alpha \\ \beta \\ v \end{bmatrix} = \begin{bmatrix} c \\ -a \\ -g \end{bmatrix} + \begin{bmatrix} 0 & -A^T & C \\ A & 0 & 0 \\ 0 & 0 & D \end{bmatrix} \cdot \begin{bmatrix} x \\ u \\ y \end{bmatrix} \tag{6.10}$$

$$\alpha, \beta, v, x, u, y \ge 0 \qquad \alpha^T x = \beta^T u = 0$$

A equivalência estabelecida apenas exige que g(y) exista para qualquer $y \in S_2$. Para isso basta que o conjunto S_1 seja limitado.

Devido à equivalência apresentada nesta secção, a determinação de um mínimo global do BLP reduz-se à resolução do MLCP (6.10) pelo método de pesquisa em árvore com limites ou pelo algoritmo SLCP descrito no capítulo 5. Nas próximas secções iremos discutir o uso desses processos para o BLP. Finalmente uma comparação computacional entre esses algoritmos será apresentada na última secção deste capítulo.

3 - Método de Pesquisa em Árvore com Limites para o BLP

Tal como foi referido no capítulo 5, neste algoritmo começa por resolver-se o programa linear RLP obtido do MLCP (6.10) por relaxação das condições de complementaridade $\alpha^T x = \beta^T u = 0$. Se S_1 e S_2 são conjuntos limitados então é possível provar [JU82] que a função linear desse programa tende para $-\infty$ no conjunto de restrições. Para ultrapassar essa dificuldade foi sugerido em [JU82] a introdução de uma restrição do tipo

$$e^T x + e^T u + e^T y \leq \tau \qquad (6.11)$$

com e um vector de uns e τ um número positivo suficientemente grande.

Experiência computacional com o método de pesquisa em árvore com limites descrito no capítulo anterior, mostrou que, para o MLCP definido por (6.10) e (6.11), o método dual simplex é preferível para ser incorporado no Passo 1 do processo para gerar limites inferiores [JU82, JUMI88b]. Contudo o método não se mostrou particularmente eficiente para os problemas-teste resolvidos, pois algumas dificuldades computacionais aparecem devido à introdução da restrição (6.11). A situação torna-se muito mais grave quando a dimensão do BLP aumenta. Nesse sentido foi sugerido em [JUFA91d] introduzir uma restrição do tipo

$$d^T y + a^T u \geq LB \qquad (6.12)$$

onde LB é um limite inferior para o BLP. Uma maneira simples de determinar um limite inferior é baseado na desigualdade

$$\min_{x \in S_1, y \in S_2} d^T y + c^T x + x^T C y \geq \min_{y \in S_2} (d + \theta)^T y \qquad (6.13)$$

onde θ é um vector cujas componentes θ_k satisfazem

$$\text{LP(k):} \quad \theta_k = \min_{x \in S_1} (c + C_{.k})^T x \qquad (6.14)$$

com $C_{.k}$ a coluna k de C. Segundo esse processo é necessário resolver n_2 programas lineares da forma (6.14) para determinar o vector θ. Esses programas têm soluções óptimas, pois S_1 é limitado, e podem ser resolvidos usando uma técnica semelhante à do algoritmo SPL descrito na secção 1 deste capítulo. Após a obtenção do vector θ, LB é calculado a partir de

$$LB = \min_{y \in S_2} (d + \theta)^T y \qquad (6.15)$$

Portanto é necessário resolver $(n_2 + 1)$ programas lineares para obter o limite inferior. Note-se que esse número pode ser pequeno mesmo para programas bilineares de grandes dimensões, desde que um dos vectores x ou y tenha dimensão reduzida.

O algoritmo que permite determinar o valor do limite inferior LB será denotado por LOWBND e consiste então nos seguintes passos:

ALGORITMO LOWBND

Passo 0 – Determine duas soluções básicas admissíveis referentes aos conjuntos de restrições S_1 e S_2. Se tal não for possível faça TERM = 0 e vá para EXIT. De outro modo, faça k = 1 e sejam B_0 e $\bar{B}_0$ as matrizes bases associadas a essas soluções.

Passo 1 – Resolva o programa linear LP(k) definido por (6.14) pelo método simplex começando com a base inicial B_{k-1}. Seja B_k a base associada à solução óptima x^k e θ_k o valor óptimo do LP(k). Se $k = n_2$ vá para o Passo 2. De outro modo faça $k = k + 1$ e repita o Passo 1.

Passo 2 – Resolva o programa linear (6.15) usando o método simplex começando com a base inicial $\bar{B}_0$. Então LB é o valor óptimo desse programa e vá para EXIT com TERM = 1.

EXIT – Se TERM = 0, o BLP é não admissível. De outro modo o limite inferior LB do BLP foi calculado.

A introdução da restrição (6.12) nas restrições MLCP (6.10) origina soluções duais altamente degeneradas. Com efeito, os coeficientes desta restrição são iguais aos coeficientes de custo da função objectivo do MLCP. Por isso não é aconselhável usar o método dual simplex para resolver os programas lineares que são necessários para determinar limites inferiores no método de pesquisa em árvore com limites. Daí termos optado pelo método primal simplex para a resolução desses programas lineares.

Tal como referimos no capítulo 5, o método de pesquisa em árvore com limites permite, pelo menos teoricamente, obter um mínimo global do BLP. Na última secção deste capítulo apresentaremos alguma experiência computacional com este algoritmo na

resolução de alguns BLPs de média dimensão. Esses resultados mostram que a eficiência do método é bastante deteriorada pelo aumento da dimensão do BLP.

4 - Método SLCP para o BLP

Como foi referido no capítulo anterior, neste processo é introduzido um parâmetro λ e a função objectivo do MLCP (6.10) é substituída pela restrição $d^T y + a^T u \leq \lambda$, de modo a obter o seguinte LCP:

LCP(λ)

$$\begin{bmatrix} \alpha \\ \beta \\ v \\ v_0 \end{bmatrix} = \begin{bmatrix} c \\ -a \\ -g \\ \lambda \end{bmatrix} + \begin{bmatrix} 0 & -A^T & C \\ A & 0 & 0 \\ 0 & 0 & D \\ 0 & -a^T & -d^T \end{bmatrix} \begin{bmatrix} x \\ u \\ y \end{bmatrix} \qquad (6.16)$$

$$x, y, u, \alpha, \beta, v, v_0 \geq 0 \qquad \alpha^T x = \beta^T u = 0$$

O algoritmo consiste em resolver uma sucessão de LCPs(λ_k), com $\{\lambda_k\}$ uma sucessão de números reais decrescente definida a partir de

$$\lambda_k = h(y^{k-1}, u^{k-1}) - \gamma \, | \, h(y^{k-1}, u^{k-1}) \, |$$

onde $h(y, u) = d^T y + c^T u$, $(x^{k-1}, u^{k-1}, y^{k-1})$ é a solução do LCP(λ_{k-1}) e γ é um número real positivo pequeno.

O método termina numa iteração k tal que LCP(λ_k) não tenha solução. Nesse caso, $(x^{k-1}, u^{k-1}, y^{k-1})$ é uma solução ε-óptima com

$$\varepsilon = \gamma \, | \, h(y^{k-1}, u^{k-1}) \, |$$

Seguidamente iremos discutir os processos de solução dos LCP(λ_k).

(i) Solução do LCP(λ_0)

Neste LCP a restrição $d^T y + a^T u \leq \lambda$ é omitida, dando assim lugar a um problema da forma:

$$\begin{bmatrix} \alpha \\ \beta \\ v \end{bmatrix} = \begin{bmatrix} c \\ -a \\ -g \end{bmatrix} + \begin{bmatrix} 0 & -A^T & C \\ A & 0 & 0 \\ 0 & 0 & D \end{bmatrix} \begin{bmatrix} x \\ u \\ y \end{bmatrix} \qquad (6.17)$$

$$\begin{bmatrix} x \\ u \end{bmatrix} \geq 0, \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \geq 0, \; y \geq 0, \; v \geq 0, \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = 0$$

Devido à sua estrutura especial, a solução deste LCP pode ser obtida do seguinte modo:

i) Encontrar uma solução $\bar{y} \in S_2$

ii) Resolver o LCP

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} c + C\bar{y} \\ -a \end{bmatrix} + \begin{bmatrix} 0 & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \qquad (6.18)$$

$$\begin{bmatrix} x \\ u \end{bmatrix} \geq 0, \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \geq 0, \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = 0$$

Mas, pela teoria de dualidade da programação linear [MU76, cap. 4], resolver o problema (6.18) é equivalente a encontrar a solução óptima do seguinte programa linear:

$$\begin{aligned} &\text{minimizar} && (c + C\bar{y})^T x \\ &\text{sujeito a} && Ax \geq a \\ & && x \geq 0 \end{aligned} \qquad (6.19)$$

Por outro lado, como a escolha da solução admissível $\bar{y} \in S_2$ é arbitrária e tendo em conta que o fim em vista é a determinação de um mínimo global do BLP, podemos calcular $\bar{y}$ a partir da solução do seguinte programa linear:

$$\begin{aligned} &\text{minimizar} \quad (d + C^T\bar{x})^T y \\ &\text{sujeito a} \quad y \in S_2 \end{aligned} \tag{6.20}$$

com $\bar{x} = 0$.

Se x^0 e y^0 são as soluções óptimas dos programas lineares (6.19) e (6.20) respectivamente e se

$$\lambda_0 = (c + C\,y^0)^T x^0 + d^T y^0 = (d + C^T x^0)^T y^0 + c^T x^0 \tag{6.21}$$

então (x^0, y^0) é uma solução do LCP(λ_0). Como se pretende determinar um mínimo global do MLCP (6.10), então λ_0 deverá ser o mais pequeno possível. Para isso, podemos usar o algoritmo SPL descrito na secção 1 e assim determinar um ponto estacionário do BLP. Se (x^0, y^0) for esse ponto então (x^0, y^0) é solução do LCP(λ_0) com λ_0 dado por (6.21).

Chegamos assim à conclusão que o algoritmo SPL pode ser usado como processo para resolver o primeiro LCP. Em [JUFA88c] foi investigada não só a eficiência do algoritmo SPL (isto é, o número de operações pivotais e o tempo de CPU) mas também se o valor λ_0 obtido pelo processo SPL está suficientemente perto do valor óptimo do BLP. No quadro 6.1 são apresentados alguns resultados computacionais que dão alguma ideia do comportamento do algoritmo. Os problemas-teste foram gerados de acordo com o processo descrito na última secção deste capítulo. Nesse quadro usamos as seguintes notações:

NVX = número de variáveis x do BLP

NVY = número de variáveis y do BLP

NRX = número de restrições x do BLP

NRY = número de restrições y do BLP

λ_0 = valor de λ_0 após algoritmo SPL

λ_f = valor de λ correspondente à solução óptima do BLP (ou melhor solução encontrada)

NLP = número de programas lineares resolvidos pelo algoritmo SPL

NI = número de operações pivotais do algoritmo SPL

T = tempo CPU em segundos

Estes resultados não permitem tirar conclusões seguras quanto à obtenção de um valor para λ_0 próximo do valor óptimo do BLP. Com efeito há alguns problemas para os quais λ_0 está muito próximo de λ_f (até é igual em dois casos), mas também há outros BLPs para os quais

$$|\lambda_f - \lambda_0|$$

é grande. Contudo o método SPL é bastante eficiente. Com efeito não só é reduzido o número de programas lineares a resolver, como também o número de operações pivotais e o tempo de CPU são bastante pequenos. Daí recomendarmos o método SPL como o processo a usar para a resolução do LCP(λ_0).

Prob.	Dimensão				SPL				
	NVX	NVY	NRX	NRY	NLP	NI	T	λ_0	λ_f
1	30	200	1	30	4	148	3.7	–294.9	–321.3
2					4	229	5.8	–255.1	–343.6
3	30	300	1	30	4	231	7.3	–562.7	–562.7
4					4	210	6.9	–461.2	–552.7
5	30	100	1	50	4	308	10.9	–92.48	–132.8
6					4	320	11.9	–90.08	–109.2
7	30	200	1	50	4	393	17.1	–292.1	–292.1
8					4	360	15.1	–250.4	–340.7
9	10	100	10	30	4	117	1.5	–38.26	–38.51
10					4	112	1.5	–43.72	–45.08
11	10	200	10	30	4	114	2.1	–85.35	–126.0
12					7	137	2.4	–86.25	–89.27

Quadro 6.1

(ii) Solução do LCP(λ_k), $k \geq 1$

Como foi referido anteriormente, pode usar-se o método enumerativo híbrido descrito no capítulo 2 para resolver os LCPs(λ_k) com $k \geq 1$. Nesse método enumerativo, em cada nó é aplicado o algoritmo de gradiente reduzido modificado para determinar um ponto de Kuhn-Tucker ou LSM para a função

$$g(\alpha, x, \beta, u, y, v, v_0) = \alpha^T x + \beta^T u$$

referente ao conjunto de restrições lineares do MLCP.

Como a estrutura do LCP(λ_k) é separada nas variáveis complementares, então o algoritmo SMRG pode ser usado em cada um dos nós gerados pelo método enumerativo híbrido. Recordemos que nesse processo uma direcção descendente é identificada apenas por um custo reduzido negativo de uma determinada variável não básica.

5 - Experiência Computacional

Nesta secção é apresentada alguma experiência computacional com os algoritmos descritos neste capítulo na resolução de BLPs de pequena e média dimensão. Tal como anteriormente a experiência foi desenvolvida num CDC CYBER 180-830 da Universidade do Porto e os resultados são apresentados nos quadros 6.2 a 6.5. Nestes quadros além das notações referidas na secção anterior ainda consideramos os seguintes parâmetros:

NC = número de pares de variáveis complementares do MLCP (6.10)

NRW = número de linhas da matriz do MLCP (6.10)

NCL = número de colunas da matriz do MLCP (6.10)

NLCP = número de LCPs(λ_k) a serem resolvidos pelo algoritmo SLCP (incluíndo o LCP(λ_0))

ND = número de nós

NS = o algoritmo não terminou após 20000 operações pivotais

BVAL = valor da função objectivo correspondente à melhor solução encontrada pelo algoritmo SLCP ou pelo método de pesquisa em árvore com limites.

LVAL = valor do limite inferior obtido pelo algoritmo LOWBND

Os problemas-teste apresentados nos quadros 6.2 e 6.3 foram gerados do seguinte modo:

i) cada problema-teste PTi tem associados determinados valores de NRX, NVX, NRY, NVY;

ii) para cada PTi são considerados cinco problemas-teste nos quais as matrizes A, D e C são geradas aleatoriamente, por uma técnica semelhante à descrita no capítulo 2 e onde A e D têm uma linha densa de uns de modo a tornar limitados os conjuntos S_1 e S_2 do BLP.

Os quadros 6.2 e 6.3 contêm a informação do comportamento dos algoritmos na resolução dos cinco problemas de cada PTi, mostrando, em termos de operações pivotais, o resultado melhor (B), pior (P) e médio (M). Na coluna encabeçada por OPTIMO apresentamos o esforço computacional dos algoritmos para determinar a melhor solução do BLP (solução óptima em alguns casos). Como referimos no capítulo anterior, o último LCP normalmente acarreta um muito maior esforço que todos os outros. Para ilustrar um tal procedimento resolvemos apresentar os resultados computacionais referentes a este LCP numa coluna à parte encabeçada por ULTIMO.

Na experiência computacional na resolução de BLPs começámos por testar a influência do processo MAXVAR na eficiência do algoritmo SLCP. Nessa medida resolvemos os problemas-teste da colecção de BLPs usando o algoritmo SLCP (com $\gamma = 10^{-4}$) incorporando o processo MAXVAR ou não. Os resultados computacionais apresentados no quadro 6.2 mostram que o uso do processo MAXVAR implica uma razoável redução do número de LCPs (λ_k) a resolver. Isso acarreta por sua vez reduções no número de operações pivotais e no tempo de CPU, mas os decréscimos nesses valores não acompanham as elevadas reduções no número de LCPs a resolver. Daí também se poder concluir a grande eficiência do método enumerativo híbrido quando uma boa base inicial é conhecida.

O uso do algoritmo MAXVAR aumenta também a robustez do método SLCP, pois é normalmente menor a diferença entre a melhor e a pior eficiência do processo para cada um dos problemas-teste PTi. Além disso as comparações dos resultados dos quadros 6.2 e 6.3 mostram que o algoritmo SLCP não é muito influenciável pelo valor do parâmetro γ, mas normalmente o valor de $\gamma = 10^{-3}$ é mais recomendável.

Probl.	DIMENSÃO								SLCP com MAXVAR					SLCP sem MAXVAR				
									OPTIMO			ULTIMO		OPTIMO			ULTIMO	
	NVX	NVY	NRX	NRY	NC	NRW	NCL		NLCP	NI	T	NI	T	NLCP	NI	T	NI	T
1	30	200	1	30	31	61	231	B M P	2 4 5	304 777 1632	12.0 37.5 85.3	1338 1820 2300	70.1 101. 132.	34 60 153	368 1344 4122	13.5 65.3 214.	1351 1807 2335	73.4 99.4 133.
2	30	300	1	30	31	61	331	B M P	1 3 4	231 641 967	7.5 37.1 60.3	1683 1959 2269	118. 136. 162.	1 56 82	231 912 1927	7.5 51.4 120.	1468 1998 2470	100. 138. 166.
3	30	100	1	50	31	81	131	B M P	4 3 4	702 2431 4562	34.7 154. 295.	2998 3880. 4501.	183. 254. 299.	127 87 72	892 2563 4611	39.8 157. 290.	2737 3779 4273	166. 245. 303.
4	30	200	1	50	31	81	231	B M P	1 3 3	393 3080 5318	17.1 224. 411.	5434 6228 6781	408. 483. 538.	1 186 286	393 3415 5931	17.1 233. 432.	5198 6290 7129	383. 484. 541.
5	10	100	10	30	20	50	120	B M P	2 1 2	123 146 173	1.7 2.5 4.0	536 3219 11755	20.6 112. 404.	3 2 4	125 147 177	1.7 2.5 4.1	536 3219 11755	20.6 112. 404.
6	10	200	10	30	20	50	220	B M P	3 3 3	149 217 365	3.6 6.6 15.6	857 3184 11316	42.7 156. 557.	1 7 3	163 225 364	3.2 6.7 15.5	858 3088 11115	42.6 154. 547.
7	30	100	10	30	40	70	140	B M P	5 6 4	353 1741 5266	11.6 73.8 231.	NS NS NS		25 39 37	393 1815 5356	12.6 76.0 235.	NS NS NS	
8	30	200	10	30	40	70	240	B M P	3 5 5	231 3656 9935	8.7 208. 576.	NS NS NS		29 69 23	282 3771 9971	10.0 211. 577.	NS NS NS	

Quadro 6.2 – Algoritmo SLCP com $\gamma = 10^{-4}$

Prob.		LOWBND		SLCP (MAXVAR e $\gamma = 10^{-3}$)							BRANCH-AND-BOUND			INCUMBENTE B.B.		
				OPTIMO			ULTIMO									
							S/CORTE		C/CORTE							
		NI	T	NLCP	NI	T	NI	T	NI	T	ND	NI	T	ND	NI	T
1	B	2291	60.6	2	304	11.7	1337	70.0	1297	70.4	63	3547	226.	63	1580	95.9
	M	2715	78.6	4	790	37.6	1827	99.6	1828	105.	63	4140	268.	63	2514	156.
	P	2979	90.2	5	1632	82.4	2300	130.	2238	131.	63	4896	322.	63	3455	230.
2	B	2164	71.2	1	231	7.3	1683	114.	1613	114.	63	4519	353.	63	1932	141.
	M	2871	95.7	3	641	36.2	1959	133.	2041	146.	63	5184	410.	63	2355	180.
	P	3605	119.	4	967	58.2	2269	158.	2538	162.	63	6222	495.	63	2638	206.
3	B	4809	197.	4	730	35.5	2982	179.	2942	179.	63	6300	491.	63	4245	264.
	M	5109	223.	3	2436	149.	3877	245.	3813	249.	63	6985	561.	63	5381	367.
	P	5678	293.	4	4562	282.	4378	279.	4262	306.	63	7884	654.	63	6181	457.
4	B	7975	398.	1	393	17.2	4704	349.	5194	406.	63	10349	941.	63	5910	532.
	M	8317	434.	3	2608	205.	6093	466.	6248	497.	63	11635	1132.	63	6922	576.
	P	8810	478.	3	5318	400.	7121	550.	6970	592.	63	12505	1247.	63	8810	734.
5	B	426	8.0	2	123	1.8	526	20.4	561	22.1	125	3465	122.	73	2783	113.
	M	559	10.6	1	146	2.5	3231	111.	1107	40.1	114	4440	172.	93	3304	133.
	P	696	15.0	2	173	4.0	11765	397.	11496	429.	NS			NS		
6	B	882	25.1	3	149	3.6	748	38.1	748	38.9	107	5310	280.	77	2720	141.
	M	951	25.7	3	202	5.6	3177	155.	3132	158.	118	6230	335.	225	5903	322.
	P	1006	26.0	3	292	10.4	11317	552.	11054	559.	NS			809	15151	815.
7	B	1594	31.5	4	388	12.8	NS		NS		NS			NS		
	M	1856	35.9	6	1694	70.2	NS		NS		NS			NS		
	P	2149	41.4	4	5288	228.	NS		NS		NS			NS		
8	B	2307	58.8	3	231	8.7	NS		NS		NS			NS		
	M	2672	72.8	5	3664	202.	NS		NS		NS			NS		
	P	2994	87.4	5	10064	569.	NS		NS		NS			NS		

Quadro 6.3 – Comparação de algoritmos para a resolução do BLP

Os resultados computacionais apresentados nos quadros 6.2 e 6.3 mostram que o algoritmo SLCP é bastante eficiente para encontrar um mínimo global dum GSBLP. Todavia, o tempo e o esforço computacional exigidos para estabelecer que tal solução foi encontrada são superiores aos necessários para a obter. Este facto deve-se ao grande número de nós que o método enumerativo necessita de explorar quando o LCP(λ_k) não tem solução. A mesma conclusão pode ser tirada para os restantes problemas, sendo nesses casos ainda mais notória a ineficiência do método enumerativo para mostrar que o último LCP não tem solução.

A eficiência do algoritmo SLCP, com ou sem MAXVAR, depende da dimensão do BLP, sendo evidentes a maior influência do número NC de pares de variáveis complementares do correspondente MLCP e a menor influência do número NCL quando o seu acréscimo se deve apenas ao aumento de variáveis y. Como na formulação de um BLP no MLCP (6.10) não importa quais das variáveis se vai identificar por x ou y, recomenda-se a escolha das variáveis x quando

$$NRX + NVX < NRY + NVY$$

No quadro 6.3 é também apresentada a eficiência do método de pesquisa em árvore com limites (BRANCH-AND-BOUND) e de dois algoritmos referenciados como INCUMBENTE B.B. e SLCP C/CORTE. No primeiro destes dois últimos algoritmos, uma solução incumbente é encontrada inicialmente pelo algoritmo SLCP, sendo depois usado o método de pesquisa em árvore com limites, resultando o esforço computacional na soma dos dois métodos. No segundo algoritmo, é calculado um limite inferior da função objectivo pelo algoritmo LOWBND e a restrição (6.12) é acrescentada ao último LCP(λ_k) do SLCP.

Os resultados apresentados neste quadro 6.3 revelam que o método de pesquisa em árvore com limites não é competitivo com o método SLCP. O uso de uma incumbente dada pelo algoritmo SLCP melhora bastante a eficiência do método de pesquisa em árvore, mas mesmo assim o processo não é competitivo com o algoritmo SLCP. Além disso, não se detectaram melhorias pelo facto de se usar o corte (6.12) no método SLCP. Se se tiver em conta, entretanto, que para contabilizar o esforço computacional se deve incluir o relativo ao LOWBND, é ainda mais clara a desvantagem de se utilizar uma tal variante no método SLCP.

Probl.	LOWBND			SLCP (MAXVAR)				BRANCH-AND-BOUND			
	NI	T	LVAL	NLCP	NI	T	BVAL	ND	NI	T	BVAL
7-1	2149	41.4	-177.3	4	5266	231.	-128.8	337	16969	1027.	-59.4
7-2	1873	35.1	-168.2	5	353	11.6	-110.7	23	1012	54.8	-52.1
7-3	1594	31.4	-177.1	9	561	22.9	-138.3	111	5135	290.	-97.12
7-4	1803	36.6	-172.9	9	913	37.4	-101.0	285	13496	882.	-76.84
7-5	1861	35.0	-172.2	3	1610	65.8	-136.2	193	13018	853.	-44.45
8-1	2307	58.8	-374.1	6	440	18.9	-300.6	371	19982	1546.	-100.5
8-2	2571	65.8	-340.9	3	231	8.7	-244.7	359	11972	820.	-123.1
8-3	2994	87.4	-357.7	5	9935	576.	-242.8	351	16422	1219.	-113.0
8-4	2911	86.6	-346.8	4	7391	426.	-289.9	235	20051	1469.	-142.4
8-5	2578	65.6	-358.6	9	285	11.3	-303.9	269	19772	1436.	-184.6

Quadro 6.4 – Valores das melhores soluções obtidas pelo algoritmo SLCP e método de pesquisa em árvore

No quadro 6.4 compara-se o limite inferior encontrado pelo algoritmo LOWBND com os melhores valores obtidos pelo algoritmo SLCP e o método de pesquisa em árvore com limites para os problemas em que o algoritmo sequencial não terminou. Os resultados mostram que os valores encontrados pelo algoritmo SLCP são muito mais pequenos que os obtidos pelo método de pesquisa em árvore. Além disso, o esforço computacional para obter a melhor solução é muito menor para o algoritmo SLCP. A diferença entre o valor BVAL encontrado por este último método e o limite inferior LVAL explica a dificuldade do SLCP C/ CORTE em terminar. Estes resultados também mostram que o algoritmo LOWBND não fornece, geralmente, um limite inferior preciso.

Probl.	DIMENSÃO			BLSIMP		SLCP (MAXVAR)				
						OPTIMO			ULTIMO	
	NVX	NRY	NVY	NI	T	NLCP	NI	T	NI	T
1	5	30	150	134	3.2	1	137	2.7	71	2.5
2	10	30	150	336	7.5	3	150	3.5	232	8.0
3	15	30	150	626	14.5	1	163	3.0	656	25.3
4	20	30	150	1076	27.3	1	129	2.4	888	36.8
5	30	30	150	1536	36.9	1	213	4.2	1199	53.7
6	40	30	150	2840	72.7	3	247	8.3	2266	116.
7	30	30	300	2820	105.	5	621	35.8	1997	137.
8	30	50	100	4853	201.	4	702	35.4	4501	298.
9	30	50	150	5834	271.	4	5459	359.	5339	359.
10	30	50	200	7049	366.	3	2313	169.	5597	428.

Quadro 6.5 – Solução de GSBLPs

O quadro 6.5 mostra que o algoritmo SLCP é, em muitos casos, competitivo relativamente ao método BLSIMP na resolução de GSBLPs. O tempo é sempre menor para este último algoritmo, porque ele trabalha com matrizes base de dimensões menores.

Como conclusão final deste estudo computacional, pode afirmar-se que o algoritmo SLCP é bastante eficiente na obtenção de um mínimo global (ou pelo menos de uma solução bastante próxima do óptimo) para um BLP de dimensões razoáveis. No entanto, o método é incapaz, geralmente, de estabelecer que a solução foi encontrada. O uso do algoritmo SLCP como método de obtenção do mínimo global depende assim da escolha de procedimentos capazes de encontrar limites inferiores precisos. O algoritmo LOWBND, pode servir em certos casos, mas em geral não é um método eficiente para o objectivo em causa. É de crer que o uso de um procedimento que gere limites inferiores precisos, conjuntamente com o método SLCP, possa conduzir a um algoritmo poderoso para a resolução de BLPs.

CAPÍTULO 7

PROGRAMAÇÃO DE DOIS NÍVEIS

1 - Definição e Propriedades

O problema de programação matemática de dois níveis (BP) é na sua forma geral definido do seguinte modo:

$$\begin{array}{ll} \underset{x\in R^{n_2},\, y\in Y}{\text{minimizar}} & f_1(x, y) \\ \text{sujeito a} & g_i(x, y) \geq 0 \ , \ i = 1,\ldots, m_1 \end{array} \qquad (7.1)$$

com x definido implicitamente como a solução óptima do programa

$$\begin{array}{ll} \underset{x\in X}{\text{minimizar}} & f_2(x, y) \\ \text{sujeito a} & h_i(x, y) \geq 0 \ , \ i = 1,\ldots, m_2 \end{array} \qquad (7.2)$$

Nesta formulação $y \in R^{n_1}$, $x \in R^{n_2}$, Y e X são subconjuntos convexos de R^{n_1} e R^{n_2} respectivamente e f_1, f_2, g_i e h_i são funções de $R^{n_1+n_2}$ em R.

O BP é assim um problema de optimização hierárquica, onde existe um primeiro nível de decisão que controla as variáveis x_i, y_i, e um segundo nível correspondente às variáveis x_i. Este problema tem sido bastante estudado nos últimos anos devido ao seu elevado número de aplicações [BEBLBO88, KOL85, KOLA86, SA89]. Vários algoritmos têm sido desenvolvidos para o BP, principalmente para a sua versão mais simples em que todas as funções são lineares (BLLP) [BIKA82, BARD83, JUFA88b, JUFA91b, ANWH88, HAJASA89, BAMO90]. Alguns desses processos podem ser estendidos ao caso em que a função f_2 é quadrática (BLQP) [BAMO90, JUFA91e, HAJASA89]. Mais recentemente têm sido propostos algoritmos para a resolução de BPs não lineares [EDBA91, SAGA 90] e de BPs em que algumas das variáveis são inteiras [MOBA90]. Neste trabalho iremos estudar essencialmente problemas BLLP e BLQP.

Nesta secção iremos começar por apresentar algumas definições e propriedades que têm sido propostas na literatura da programação de dois níveis e que nos ajudam a entender melhor o problema. Assim, se considerarmos as funções

$$g(x, y) = (g_1(x, y), \ldots, g_{m_1}(x, y))$$

$$h(x, y) = (h_1(x, y), \ldots, h_{m_2}(x, y))$$

então podemos definir os seguintes conjuntos:

(i) Espaço de Soluções Relaxado

$$\Omega = \{ (x, y) \in X \times Y: \; g(x, y) \geq 0 \;, \; h(x, y) \geq 0 \}$$

(ii) Projecção de Ω no 1º nível de decisão

$$\Omega(Y) = \{ y \in Y: \text{existe } x \in R^{n_2} \text{ tal que } h(x, y) \geq 0 \}$$

(iii) Conjunto de Reacçãol do 2º nível para y fixo

$$M(y) = \{ x \in X: h(x, y) \geq 0 \text{ e } x \in \operatorname{argmin} f_2(x, y) \}$$

(iv) Região Induzida

$$IR = \{ (x, y) \in X \times Y : \; (x, y) \in \Omega \;, \; x \in M(y) \}$$

Se não existirem restrições no 1º nível, IR aparece com a definição mais comum:

$$IR = \{ (x, y) \in X \times Y : \; y \in \Omega(Y) \,, \; x \in M(y) \}$$

No caso de haver tais restrições, então podemos escrever

$$IR = \{ (x, y) \in X \times Y : \; y \in \Omega(Y) \,, \; x \in M(y) \text{ e } g(x, y) \geq 0 \}$$

Um dado ponto $(x^*, y^*) \in \Omega$ é uma **solução óptima** do BP se e só se $(x^*, y^*) \in IR$ e

$$f_1(x^*, y^*) \leq f_1(x, y)$$

para todos os pontos $(x, y) \in IR$.

A resolução do BP é normalmente feita explorando formulações equivalentes. Assim, se Ω é compacto e M (y) é unívoca, então o BP é equivalente a um problema com um só nível da forma [SA89]

$$\begin{aligned} &\underset{y}{\text{minimizar}} && f_1(x(y), y) \\ &\text{sujeito a} && y \in \Omega(Y) \\ & && g(x(y), y) \geq 0 \end{aligned} \qquad (7.3)$$

onde x (y) = M (y).

Como veremos mais adiante, alguns dos métodos de resolução usam esta formulação. No entanto não é em geral fácil obter um óptimo global, pois o problema não é convexo. Além disso, esta formulação não é simples já que exige o conhecimento de x (y) e Ω (Y) que não estão definidos explicitamente.

Outros processos de solução do BP exploram uma outra formulação em que se substitui o problema do 2º nível pelas suas condições necessárias e suficientes de optimalidade. Com efeito, se o problema do 2º nível for convexo, então o BP é equivalente ao seguinte problema de optimização:

$$\begin{aligned} &\underset{(x,y)\in X\times Y}{\text{minimizar}} && f_1(x, y) \\ &\text{sujeito a} && g(x, y) \geq 0 \\ &&& \nabla_x f_2(x, y) - \lambda \nabla_x h(x, y) = 0 \\ &&& h(x, y) \geq 0 \\ &&& \lambda h(x, y) = 0 \\ &&& \lambda \geq 0 \end{aligned} \qquad (7.4)$$

Se a função do 2º nível for quadrática, então as restrições do problema (7.4) são lineares. Se além disso f_1 é uma função linear, o BP é equivalente a um MLCP e portanto podemos enunciar o seguinte resultado:

Teorema 7.1 - Se o BLQP tem uma solução óptima, então alcança o óptimo num ponto extremo de Ω.

No entanto, como M (y) não é convexo, o BP é um problema de programação não convexa, podendo demonstrar-se [JE85] que se trata de um problema NP-completo.

2 - Métodos de Solução

Os algoritmos para a resolução do BP podem ser classificados em três categorias [KOL85]. Assim existem métodos descendentes, que exploram a equivalência do BP com o problema (7.3). Este tipo de métodos é muito usado na resolução de problemas de programação matemática não linear, mas não merece igual aceitação na resolução de problemas de dois níveis. Com efeito [SA89] o programa (7.3) não é convexo e normalmente tem vários óptimos locais. Como o método descendente apenas garante a existência de um mínimo local, a determinação do óptimo global fica dependente da escolha do ponto inicial. Além disso é difícil calcular a função implícita x (y) e, na maior

parte dos casos, essa função não é diferenciável em todo o domínio. Essas razões são mais do que suficientes para preterirmos este tipo de metodologia.

Numa segunda categoria incluem-se os métodos de enumeração implícita de pontos extremos. Tais processos baseiam-se no teorema 7.1, mas tornam-se impraticáveis quando a dimensão do BP aumenta [CATO82, BIKA84, PAP82].

Finalmente as metodologias mais usadas baseiam-se na equivalência do BP com o problema (7.4), que se obtém a partir das condições de optimalidade do problema do 2º nível. Esses são os únicos métodos a serem discutidos nesta tese.

Em relação ao BLLP, o MLCP equivalente tem a forma

$$\text{minimizar} \quad c^T x + d^T y$$

$$\text{sujeito a:} \quad \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} -b \\ a \end{bmatrix} + \begin{bmatrix} 0 & A_1 & A_2 \\ -A_1^T & 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ x \\ y \end{bmatrix} \qquad (7.5)$$

$$x, y, u, \alpha, \beta \geq 0 \quad , \quad x^T\beta = u^T\alpha = 0$$

Como vimos anteriormente este problema pode ser resolvido por um método de pesquisa em árvore com limites ou por um algoritmo SLCP. O primeiro trabalho a propor a resolução do BLLP através do método SLCP foi apresentado em Bialas e Karwan [BIKA84]. O processo aí proposto é bastante semelhante ao algoritmo SLCP, mas é sugerido o algoritmo BRES descrito no capítulo 2 para resolver os vários LCP (λ_k). Os autores concluem erradamente que, se o algoritmo BRES terminar sem obter solução, então o LCP (λ_k) é inadmissível. Com efeito o seguinte exemplo apresentado em [JUFA88b] mostra que tal não é verdadeiro.

Considere-se o BLLP:

$$\min_{x,y} \; -x$$

com x a solução óptima de

$$\min_{x} \; x$$

$$\text{sujeito a:} \; -y - 2x \leq -10$$
$$y - 2x \leq 6$$
$$2y - x \leq 21$$
$$y + 2x \leq 38$$
$$-y + 2x \leq 18$$
$$x, y \geq 0$$

$$y + 2x \leq 38$$
$$-y + 2x \leq 18$$
$$x, y \geq 0$$

Para resolver o LCP (0) usando o algoritmo BRES introduz-se uma variável artificial z_0 com coluna $p = (1, 0, 0, 0, 0, 0, 0)^T$ e obtém-se o seguinte quadro inicial:

		z_0	z_1	z_2	z_3	z_4	z_5	z_6	y
$w_1 =$	-10	(1)	0	0	0	0	0	2	1
$w_2 =$	6	0	0	0	0	0	0	2	-1
$w_3 =$	21	0	0	0	0	0	0	1	-2
$w_4 =$	38	0	0	0	0	0	0	-2	1
$w_5 =$	18	0	0	0	0	0	0	-2	1
$w_6 =$	1	0	-2	-2	-1	2	2	0	0
$v_0 =$	0	0	0	0	0	0	0	1	0

onde $z = (u, x)$ e $w = (\alpha, \beta)$. Inicialmente z_0 é introduzida na base efectuando uma operação pivotal com o pivot indicado por um círculo no quadro. O quadro associado com esta solução básica é apresentado a seguir:

		w_1	z_1	z_2	z_3	z_4	z_5	z_6	y
$z_0 =$	10	1	0	0	0	0	0	-2	-1
$w_2 =$	6	0	0	0	0	0	0	2	(-1)
$w_3 =$	21	0	0	0	0	0	0	1	-2
$w_4 =$	38	0	0	0	0	0	0	-2	-1
$w_5 =$	18	0	0	0	0	0	0	-2	1
$w_6 =$	1	0	-2	-2	-1	2	2	0	0
$v_0 =$	0	0	0	0	0	0	0	1	0

Como se pretende minimizar a variável z_0, as variáveis z_6 e y são elegíveis para entrarem na base. Contudo z_6 não pode entrar pois w_6 é básica. Por isso y é a variável escolhida para entrar na base. Depois de se encontrar o pivot (assinalado por um círculo no quadro) pelo critério usual do quociente mínimo e de se efectuar a correspondente operação pivotal, obtém-se uma outra solução básica complementar admissível que é dada pelo seguinte quadro:

		w_1	z_1	z_2	z_3	z_4	z_5	z_6	w_2
z_0 =	4	1	0	0	0	0	0	-4	1
y =	6	0	0	0	0	0	0	2	-1
w_3 =	9	0	0	0	0	0	0	-3	2
w_4 =	32	0	0	0	0	0	0	-4	1
w_5 =	24	0	0	0	0	0	0	0	-1
w_6 =	1	0	-2	-2	-1	2	2	0	0
v_0 =	0	0	0	0	0	0	0	1	0

Nesta iteração apenas z_6 é elegível para entrar na base. No entanto, como w_6 é básica, esta variável não pode ser candidata e o algoritmo BRES termina com TERM=3. Contudo, o LCP (0) tem pelo menos a solução $z = (0.5, 0, 0, 0, 0, 1)$, $y = 8$.

Este exemplo mostra que o método descrito em [BIKA84] não assegura a determinação de um óptimo local nem global, pelo que esse algoritmo não vai ser considerado na nossa discussão.

Para resolver o MLCP (7.5) Fortuny-Amat e McCarl [FOMC81] propõem a substituição da condição de complementaridade $w_i z_i = 0$ pelas restrições

$$w_i \leq \tau \eta_i$$
$$z_i \leq \tau (1 - \eta_i)$$
$$\eta_i \in \{0, 1\}$$

para cada $i = 1,\ldots, m_2 + n_2$, onde τ é um número suficientemente grande. Portanto o MLCP é transformado num programa linear com algumas variáveis inteiras. No entanto este problema é também de difícil resolução. Além disso a dimensão do problema aumenta drasticamente com essa transformação. Também os autores não apresentam quaisquer resultados computacionais que mostrem a validade dessa proposta.

O método de Bard e Falk [BAFA82] tenta contornar a dificuldade da condição de complementaridade, transformando o MLCP (7.5) num problema de programação com variáveis separadas. Para tal introduz as variáveis ψ_i e substitui a condição de complementaridade $w^T z = 0$ pelas restrições

$$\sum_{j=1}^{m_2+n_2} (\min \{0, \psi_i\} + z_i) = 0$$

$$w_i - z_i = \psi_i$$

No entanto não é apresentada qualquer experiência computacional com tal algoritmo e não nos parece que o processo seja eficiente devido à não convexidade das suas restrições adicionais.

O método apresentado em [HAJASA89] pode também ser considerado como um outro processo de explorar as condições de optimalidade do problema do 2º nível, eliminando algumas variáveis desse 2º nível num método de pesquisa em árvore.

O princípio básico do método recai no facto de, na solução óptima, algumas das desigualdades do 2º nível, incluindo as restrições de sinal nas variáveis x, serem **activas**, isto é, serem satisfeitas como igualdades. Isso vai permitir eliminar variáveis do 2º nível associando a cada restrição uma variável inteira $\alpha_i \in \{0, 1\}$ e gerando uma árvore binária em que em cada ramo um dos α_i é fixado com o valor 0 ou 1 com o significado

$$\alpha_i = \begin{cases} 1 \text{ se a restrição i é activa} \\ 0 \text{ no caso contrário} \end{cases}$$

No entanto, para tornar mais eficiente a pesquisa dessa árvore, as ramificações são feitas de modo racional só se considerando as ramificações a que a solução óptima tem que satisfazer. Para tal, criam-se condições necessárias de optimalidade nas variáveis α_i obtidas à custa do teorema de dualidade aplicado ao problema do 2º nível. Assim, se se considerar que o problema do 2º nível, depois de fixadas algumas variáveis, é dado por

$$\begin{aligned} \min_{\tilde{x}} \quad & \tilde{a}^T \tilde{x} \\ \text{sujeito a} \quad & \tilde{A}_1 \tilde{x} + \tilde{A}_2 y - v = \tilde{b} \\ & \tilde{x} \geq 0 \;\; v \geq 0 \end{aligned}$$

então, para a solução óptima $(\tilde{x}^*, y^*)$, existe um vector β e s tal que

$$\tilde{A}_1^T \beta + s = \tilde{a}$$

$$\beta \geq 0 \;\; , \;\; s \geq 0$$

$$\beta^T v = \tilde{x}^T s = 0$$

Logo, se $\tilde{a}_j < 0$, tem que existir pelo menos uma variável $\beta_i > 0$ correspondente a um $\tilde{a}^1_{ij} < 0$ ($\tilde{a}^1_{ij}$ é um elemento de $\tilde{A}_1$). No entanto, por complementaridade, $v_i = 0$ e portanto a restrição i tem que ser activa. Se, pelo contrário, $\tilde{a}_j > 0$ então ou é $s_j > 0$ e portanto $x_j = 0$ ou existe pelo menos um $\beta_i > 0$ correspondente a um $\tilde{a}^1_{ij} > 0$ e portanto é $v_i = 0$.

Estas condições necessárias de optimalidade obrigam às seguintes relações lógicas:

$$\tilde{a}_j < 0 \Rightarrow \sum_{i=1|\tilde{a}^1_{ij}<0}^{\tilde{n}_2} \alpha_i \geq 1$$

$$\tilde{a}_j > 0 \Rightarrow \sum_{i=1|\tilde{a}^1_{ij}>0}^{\tilde{n}_2} \alpha_i + \alpha_{j+\tilde{n}_2} \geq 1 \qquad (7.6)$$

onde $\alpha_{j+\tilde{n}_2}$ está associada à desigualdade $\tilde{x}_j \geq 0$.

Assim, quando se faz uma ramificação, é usada uma das restrições (7.6) anteriores, gerando-se novos ramos em que em cada um deles se impõem valores zero ou um para as várias variáveis da restrição lógica adicional.

Para tornar ainda mais eficiente o método, são verificados em cada nó testes de admissibilidade e de optimalidade que vão permitir podar ramos, melhorar soluções incumbentes e fixar variáveis sem necessidade de ramificar.

Tal como em programação inteira mista, são calculadas penalidades p_i associadas à restrição i não activa que medem o custo na função objectivo do 1º nível em obrigar a que seja nula a variável de afastamento da restrição i no problema relaxado. Essa penalidade vai permitir também podar ramos ou mostrar que determinadas restrições não podem ser activas para que a solução óptima desse subproblema não seja pior do que a solução incumbente. Além disso, fornece um bom critério para escolha do primeiro nó a ser explorado. Para uma descrição completa do algoritmo aconselhamos [HAJASA89].

A experiência computacional apresentada nesse artigo indica que o algoritmo é normalmente superior a um algoritmo de pesquisa em árvore com limites para a resolução do MLCP, quer no número de nós a explorar, quer em termos de tempo de CPU. Tais conclusões são mais evidentes para problemas mais esparsos.

No entanto trata-se de um método pouco robusto no sentido de que existe uma grande variação do esforço computacional na resolução do melhor e do pior problema-teste. De facto, tal como é referido em [HAJASA89], se muitos problemas são resolvidos sem ramificar, para outros são muitas vezes necessários mais nós do que os usados no algoritmo de pesquisa em árvore com limites. Por esse facto e porque este algoritmo exigia "software" bastante diferente do que o que havia disponível, optou-se por testar

apenas os algoritmos de pesquisa em árvore com limites e SLCP. Como afirmámos anteriormente, esses processos resolvem o MLCP directamente. Como esses métodos são também possíveis de utilizar para o BLQP e o BLLP é um caso particular desse problema, iremos considerar a aplicação do algoritmo ao BLQP. Esse problema pode ser escrito na sua forma mais geral do seguinte modo:

$$\begin{aligned} \underset{x\in R^{n_2},\, y\in R^{n_1}}{\text{minimizar}} \quad & f_1(x, y) = c^T x + d^T y \\ \text{sujeito a :} \quad & E_1 x + E_2 y \geq g \\ & y \geq 0 \end{aligned} \tag{7.7}$$

e $x \in R^{n_2}$ é a solução óptima de

$$\begin{aligned} \underset{x\in R^{n_2}}{\text{minimizar}} \quad & a^T x + y^T C x + \frac{1}{2} x^T Q x \\ \text{sujeito a :} \quad & A_1 x + A_2 y \geq b \\ & x \geq 0 \end{aligned} \tag{7.8}$$

onde $E = [E_1 : E_2] \in R^{m_1 x\,(n_1+n_2)}$, $A = [A_1 : A_2] \in R^{m_2 x\,(n_1+n_2)}$, $C \in R^{n_1 \times n_2}$, $g \in R^{m_1}$, $b \in R^{m_2}$ e $Q \in R^{n_2 \times n_2}$ é uma matriz simétrica positiva semi-definida.

Se o conjunto de restrições do problema quadrático convexo do 2º nível (7.8) é limitado e não vazio, as condições de Kuhn-Tucker desse problema originam o seguinte MLCP equivalente:

$$\begin{aligned} \text{minimizar} \quad & c^T x + d^T y \\ \text{sujeito a :} \quad & \alpha = -b + A_1 x + A_2 y \\ & \beta = a + Q x + C y - A_1^T u \\ & v = -g + E_1 x + E_2 y \\ & x, y, u, \alpha, \beta, v \geq 0 \quad , \; x^T\beta = u^T\alpha = 0 \end{aligned} \tag{7.9}$$

Esse problema pode ser resolvido pelo método de pesquisa em árvore com limites discutido no capítulo 5. Aliás Bard e Moore [BAMO90] propuseram este tipo de algoritmo sem o relacionar com a complementaridade. Daí se referir erradamente na literatura esse método como devido a Bard e Moore.

Um outro processo de resolução do MLCP é o algoritmo SLCP descrito no capítulo 5. Tal como foi referido anteriormente, em cada iteração do método SLCP é resolvido um LCP (λ) que se obtém substituindo a função objectivo pela restrição $c^T x + d^T y \leq \lambda$. Portanto:

LCP (λ)

$$\begin{bmatrix} \alpha \\ \beta \\ v \\ v_0 \end{bmatrix} = \begin{bmatrix} -b \\ a \\ -g \\ \lambda \end{bmatrix} + \begin{bmatrix} 0 & A_1 & A_2 \\ -A_1^T & Q & C \\ 0 & E_1 & E_2 \\ 0 & -c^T & -d^T \end{bmatrix} \begin{bmatrix} u \\ x \\ y \end{bmatrix} \qquad (7.10)$$

$$\alpha, \beta, v, v_0, u, x, y \geq 0 \quad , \quad u^T\alpha = x^T\beta = 0$$

No processo SLCP, λ toma uma sucessão de valores decrescentes $\{\lambda_k\}$ definidos por

$$\lambda_k = f_1(x^{k-1}, y^{k-1}) - \gamma \, | f_1(x^{k-1}, y^{k-1}) |$$

com (x^{k-1}, y^{k-1}) solução do LCP (λ_{k-1}) e γ um valor pequeno positivo. O método termina numa iteração k tal que LCP (λ_k) não tem solução. Nesse caso (x^{k-1}, y^{k-1}) é uma solução ε-óptima do BLQP com

$$\varepsilon = \gamma \, | f_1(x^{k-1}, y^{k-1}) |$$

O LCP (λ_k) (7.10) pode escrever-se na seguinte forma mais geral:

$$w = q + Mz + Ny \quad w \geq 0, z \geq 0, y \geq 0 \qquad (7.11)$$

$$z_i . w_i = 0 \ , \ i = 1, \ldots, m_2 + n_2$$

onde

$$w = \begin{bmatrix} \alpha \\ \beta \\ v \\ v_0 \end{bmatrix} \in R^{m_2+n_2+m_1+1} \ , \ q = \begin{bmatrix} -b \\ a \\ -g \\ \lambda_k \end{bmatrix} \in R^{m_2+n_2+m_1+1}$$

$$z = \begin{bmatrix} u \\ y \end{bmatrix} \in R^{m_2+n_2}$$

$$M = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix} = \begin{bmatrix} 0 & A_1 \\ -A_1^T & Q \\ 0 & E_1 \\ 0 & -c^T \end{bmatrix} \in R^{(m_2+n_2+m_1+1) \times (m_2+n_2)}$$

$$N = \begin{bmatrix} A_2 \\ C \\ E_2 \\ -d^T \end{bmatrix} \in R^{(m_2+n_2+m_1+1) \times n_1}$$

e $M_1 \in R^{(m_2+n_2) \times (m_2+n_2)} \in$ PSD:

Para resolver cada um destes LCP (λ_k) é usado o método enumerativo híbrido referido no capítulo 2. Note-se que, como a matriz M_1 é PSD, podemos usar a simplificação do método MRG exposta na secção 2.6. Como referimos anteriormente esse processo torna o método enumerativo bastante mais eficiente. Todas as outras metodologias usadas no algoritmo SLCP e discutidas no capítulo 5 devem também ser usadas neste caso.

No caso do BLLP, tem-se $C = 0$ e $Q = 0$ e é fácil de ver que o LCP (7.10) tem uma estrutura separada. Nesse caso o algoritmo SMRG deve ser usado em vez do processo MRG simplificado.

Para a solução de LCP (λ_0) é resolvido o LCP (7.10) sem a restrição $c^T x + d^T y \leq \lambda_0$

$$\begin{bmatrix} \alpha \\ \beta \\ v \end{bmatrix} = \begin{bmatrix} -b \\ a \\ -g \end{bmatrix} + \begin{bmatrix} 0 & A_1 & A_2 \\ -A_1^T & Q & C \\ 0 & E_1 & E_2 \end{bmatrix} \begin{bmatrix} u \\ x \\ y \end{bmatrix} \qquad (7.12)$$

$$\alpha, \beta, v, u, x, y \geq 0 \quad , \quad u^T\alpha = x^T\beta = 0$$

usando o método enumerativo híbrido.

No entanto, como este LCP tem várias soluções, nem sempre a primeira solução encontrada por esse método enumerativo é uma "boa" solução no sentido de o valor de λ_0 estar muitas vezes bastante afastado do valor óptimo $\bar{\lambda}$.

No caso do BLLP é possível explorar a separabilidade das variáveis complementares e usar o processo SPL para obter uma solução do LCP (λ_0). Como $Q = C = 0$ no LCP (7.12), então resolver esse LCP é equivalente a obter a solução óptima do programa

$$\begin{aligned} &\text{minimizar} && u^T \alpha + x^T \beta \\ &\text{sujeito a :} && \alpha = -b + A_1 x + A_2 y \\ &&& \beta = a - A_1^T u \\ &&& v = -g + E_1 x + E_2 y \\ &&& \alpha, \beta, v, u, x, y \geq 0 \end{aligned} \qquad (7.13)$$

que tem valor zero. Mas a função objectivo desse programa pode escrever-se na forma

$$u^T \alpha + \beta^T x = -b^T u + u^T A_1 x + u^T A_2 y + a^T x - u^T A_1 x = a^T x - b^T u + u^T A_2 y$$

Portanto o problema (7.13) é um BLP ao qual se pode aplicar o algoritmo SPL descrito no capítulo anterior. No entanto, como este algoritmo apenas fornece um ponto estacionário do problema (7.13) pode acontecer que a solução obtida não satisfaça a

condição de complementaridade (o valor da função objectivo de (7.13) ser diferente de zero). Por isso, a solução obtida, além de não ser complementar, pode também não ser uma "boa" solução inicial para o SLCP. Com o intuito de melhorar essa solução, introduz-se a função objectivo do problema do 1º nível e obtém-se o BLP

$$\text{minimizar} \quad \begin{bmatrix} c + a \\ d \end{bmatrix}^T \begin{bmatrix} x \\ y \end{bmatrix} - b^T u + u^T \begin{bmatrix} 0 & A_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (7.14)$$

$$\text{sujeito a} \quad A_1 x + A_2 y \geq b \qquad A_1^T u \leq a$$

$$E_1 x + E_2 y \geq g \qquad u \geq 0$$

$$x, y \geq 0$$

que, tal como o anterior, pode ser resolvido pelo algoritmo SPL. Neste caso o algoritmo pode ter duas terminações:

(i) a solução (x*, y*, u*) obtida por esse algoritmo satisfaz a condição de complementaridade

$$u^{*T}(-b + A_1 x^* + A_2 y^*) = 0$$

$$x^{*T}(\alpha - A_1^T u^*) = 0$$

Então (x*, y*, u*) é uma solução do LCP (7.12) e o método SLCP pode ser inicializado definindo λ_0 por

$$\lambda_0 = c^T x^* + d^T y^*$$

(ii) caso contrário, a solução obtida por esse algoritmo apenas fornece uma solução inicial para o método enumerativo híbrido. Seguidamente este último processo é aplicado para a solução do LCP (7.12).

Alguma experiência computacional com este algoritmo é apresentada na secção seguinte.

3 - Experiência Computacional

Nesta secção é apresentada alguma experiência computacional com o algoritmo SLCP, comparando-a em alguns casos com o método de pesquisa em árvore com limites [BAMO 90] que foi implementado de acordo com o descrito no capítulo 5.

Os problemas-teste são de três tipos diferentes. Num primeiro grupo são usados problemas BLLP com restrições apenas no 2º nível, sendo a matriz $A = [A_1 : A_2]$ construída do mesmo modo que as matrizes A e D dos problemas bilineares descritos no capítulo anterior. Para este tipo de problemas, a cada matriz A foram associados dois problemas-teste que apenas diferem nos coeficientes de custo. Nos problemas identificados com PTN, todos os coeficientes de custo c, d e a das duas funções objectivo são não negativos. Nos problemas identificados por PTG, os coeficientes de custo c e d da função objectivo do 1º nível são não negativos, mas a função objectivo do 2º nível tem alguns dos coeficientes a negativos. Com isto pretendeu simular-se BLLPs com objectivos conflituosos (problemas PTG) e não conflituosos (PTN).

Num segundo grupo de problemas-teste PTL, foram gerados problemas BLLPs com as características descritas em [HAJASA89], onde as matrizes A e $E = [E_1 : E_2]$ têm dimensões fixas, sendo geradas matrizes esparsas com uma percentagem de zeros fixada previamente, respectivamente 17% e 40%. Os elementos diferentes de zero são uniformemente distribuídos entre – 15 e 45 e os coeficientes das duas funções objectivo variam uniformemente entre – 20 e 20. Fez-se apenas uma modificação na determinação dos termos independentes, para que a maior parte destes problemas não fossem rejeitados pelo facto de não serem limitados ou não terem solução. Por essa razão, em vez de se gerarem termos independentes aleatoriamente entre 0 e 50 e aleatoriamente 70% de restrições de $\leq$ e 30% de $\geq$, foi introduzida, tal como nos problemas-teste anteriores, uma restrição com todos os coeficientes iguais a um e os termos independentes foram gerados de modo a que o domínio Ω admitisse a solução $x_i = y_i = 50/N$, onde $N = n_1 + n_2$.

Para um terceiro grupo de problemas-teste PTQ, além das matrizes A e E geradas pela técnica anterior com 8% de não zeros, foram ainda geradas matrizes $Q \in$ PSD e C quaisquer, obtidas a partir de matrizes PSD de dimensão $n_1 + n_2$, já usadas anteriormente e definidas em [PAN80], conservando-se as n_2 primeiras linhas dessa matriz.

As características de cada problema-teste são apresentadas em cada quadro. Todos os testes foram executados num CDC CYBER 180-830 da Universidade do Porto. Os parâmetros incluídos nos quadros têm os seguintes significados:

n = número de variáveis x do problema do 2º nível

m = número de variáveis y do problema do 1º nível

ℓ = número de restrições do 1º nível = nº de linhas de E

r = número de restrições do 2º nível = nº de linhas de A

$$\text{rsp} = \frac{\text{número de não zeros de A}}{\text{n+m}}$$

$$\text{psp} = \frac{\text{número de não zeros de A}}{\text{r x (n+m)}} \text{ x } 100$$

NC = número de variáveis complementares de MLCP (7.10) = r+n

NRW = número de linhas do MLCP (7.10) = r +ℓ+ n

NCL = número de colunas do MLCP (7.10) = r +n+ m

NLCP = número de LCPs (λ_k) resolvidos pelo algoritmo SLCP

NLP = número de LPs resolvidos pelo algoritmo SLP

NI = número total de operações pivotais

ND = número de nós requeridos pelo método de pesquisa em árvore com limites (BRANCH-AND-BOUND)

T = tempo de CPU em segundos

NS = o algoritmo não terminou depois de 20000 operações pivotais.

O primeiro objectivo deste estudo foi investigar a importância das diferentes alternativas do algoritmo SLCP. Para tal, foi utilizado o primeiro grupo de problemas-teste PTG e PTN. No algoritmo designado por SLCPMET0 é usada uma primeira versão [JUFA88b], que não recorre ao processo MAXVAR, nem ao parâmetro NMAXPV, sendo o valor de γ igual 0.01. Para os restantes algoritmos é usada a subrotina MAXVAR e o valor NMAXPV=500, designando-se por SLCPMET1 o algoritmo em que $\gamma = 0.01$ e $\bar{\gamma} = 0.05$ e por SLCPMET2 o algoritmo em que $\gamma = 0.001$ e $\bar{\gamma} = 0.01$.

A comparação entre o SLCPMET0 e o SLCPMET1 apresentada no quadro 7.1 mostra que, dum modo geral, o método SLCPMET1 é mais eficiente, fazendo diminuir significativamente o número NLCP de LCPs resolvidos e melhorando também, apesar de um modo não tão significativo, o número de pivotagens NI. Isso deve-se ao facto dos LCP (λ_k) que são evitados pelo uso da MAXVAR serem aqueles que são resolvidos pelo algoritmo BRES, gastando um número mínimo de operações pivotais (normalmente 3).

Este facto também se reflecte no tempo de execução, que algumas vezes é menor com o algoritmo SLCPMET0, mesmo quando o número de pivotagens é ligeiramente maior, pois o algoritmo BRES é mais rápido do que o MAXVAR.

Como afirmámos anteriormente, o algoritmo SLCP só garante soluções ε-óptimas do BLLP. No entanto, se o algoritmo SLCP terminar com o algoritmo MAXVAR, então um mínimo global é encontrado. Por isso é interessante investigar a influência que a MAXVAR e a redução do valor de ε (γ ou $\bar{\gamma}$) têm na eficiência do método SLCP. O SLCPMET2 foi considerado com esse propósito. Foi acrescentada no quadro 7.1 uma coluna encabeçada por OPT, na qual o valor α significa que o método SLCP pode assegurar pelo menos α% do valor óptimo ($\alpha = 0$ significa que o mínimo global foi encontrado). No caso de $\alpha \neq 0$, é importante verificar quando a solução obtida pelo método SLCP é um mínimo global. Para tal resolvemos os problemas pelo método de pesquisa em árvore com limites e escrevemos S (N) quando as soluções dos dois métodos coincidem (não coincidem) isto é, quando o algoritmo SLCP encontrou (não encontrou) o óptimo global. O algoritmo de pesquisa em árvore com limites não conseguiu resolver em menos de 20000 operações pivotais os BLLPs de dimensões maiores e por isso não se pode saber nesses casos se a solução obtida pelo método SLCP é um mínimo global para esses BLLPs. Por isso usou-se a notação ? na coluna OPT para esses problemas-teste.

Os resultados apresentados no quadro 7.1 permitem tirar as seguintes conclusões:

(i) todas as soluções obtidas pelo método SLCP, com excepção de um único caso para o SLCPMET0, são ε-óptimas;

(ii) a inclusão do algoritmo MAXVAR faz com que o algoritmo SLCP encontre com mais frequência um óptimo global, podendo, na maior parte desse casos, garantir esse óptimo global;

(iii) o processo SLCPMET1 parece mais eficiente do que o método SLCPMET2. No entanto, a diferença não é muito grande e o SLCPMET2 tem a vantagem de poder assegurar que a solução está mais próxima do óptimo global;

(iv) para todos os problemas-teste, excepto um, o algoritmo SLCPMET2 encontrou um óptimo global mesmo quando $\alpha \geq 0.1$. O SLCPMET1 falhou em três casos tendo, no entanto, encontrado um óptimo global no problema-teste em que a versão SLCPMET2 falhou;

	DIMENSÃO						SLCPMETØ				SLCPMET1				SLCPMET2			
Prob.	n	m	r	rsp	nrow	ncol	NLCP	NI	T	OPT	NLCP	NI	T	OPT	NLCP	NI	T	OPT
PTN1	30	50	30	3.5 —— 5.0	60	110	14	54	.96	1,S	5	35	0.8	0,S	5	37	0.9	0,S
PTG1							55	249	5.5	1,S	7	130	3.4	0,S	8	136	3.7	0,S
PTN2							27	315	10.3	1,S	7	259	6.8	1,S	9	400	11.5	0.1,S
PTG2							80	475	15.0	1,S	10	264	8.5	0,S	12	293	10.8	0,S
PTN3	50	120	50	5.2 —— 7.5	100	220	44	230	9.8	1,S	7	132	6.1	0,S	10	280	14.1	0,S
PTG3							124	529	32.5	1,S	6	304	18.7	0,S	9	348	23.9	0,S
PTN4							120	964	59.4	1,N	15	729	45.3	1,N	17	701	47.2	0.1,S
PTG4							163	1547	124.	1,N	26	1215	90.6	1,S	26	1068	86.1	1,S
PTN5	100	300	100	5.3 —— 7.1	200	500	139	1009	107.	1,N	11	655	66.2	0,S	23	1115	119.	0.1,N
PTG5							247	2207	308.	1,N	14	1031	141.	1,N	21	1775	284.	0,S
PTN6							126	682	79.9	1,N	4	585	86.8	0,S	5	595	88.7	0,S
PTG6							268	1899	380.	1,S	8	1744	396.	0,S	5	1298	287.	0,S
PTN7	150	250	150	5.3 —— 7.1	300	550	77	1934	128.	1,S	19	2183	278.	5,S	31	6657	856.	1,S
PTG7							214	3775	889.	1,?	24	2653	470.	5,?	39	4257	776.	1,?
PTN8							127	4740	726.	1,N	17	2332	352.	5,N	40	6910	1040.	1,?
PTG8							NS				30	6378	1571	5,?	69	8843	2123.	5,?

Quadro 7.1 – Comparação de várias versões do algoritmo SLCP

(v) o valor $\bar{\gamma} = 0.05$ teve que ser usado no SLCPMET2 para que o problema-teste PTG8 terminasse em menos de 20000 operações pivotais.

Estas conclusões permitem-nos recomendar o uso da versão SLCPMET2. Contudo, quando (r+n) é grande é conveniente considerar $\bar{\gamma} = 0.05$.

O valor NMAXPV = 500 foi usado em todos os ensaios apresentados no quadro 7.1. Para analisar o efeito dessa quantidade no método SLCP, foram executados, para diferentes valores de NMAXPV, todos os problemas-teste cuja versão SLCPMET2 só tinha garantido uma percentagem do óptimo superior a 0.1, levando-se essa análise até se encontrar a solução ε-óptima.

Prob.	r+n	NMAXPV	TOTAL			ÓPTIMO		OPT
			NLCP	NI	T	NI	T	
PTG4	100	500	26	1068	86.1	730	62.4	1,S
		1500	25	2212	163.	730	62.4	0.1,S
		3000	25	2212	163.	730	62.4	0.1,S
		4500	25	2212	163.	730	62.4	0.1,S
PTN7	300	500	31	6657	856.	4763	618.	1,S
		1500	29	5404	691.	2541	332.	1,S
		3000	28	4979	638.	2541	332.	0.1,S
		4500	28	4979	638.	2541	332.	0.1,S
PTG7	300	500	39	4257	776.	2498	551.	1,?
		1500	39	5084	871.	2498	551.	1,?
		3000	39	6574	1051.	2498	551.	1,?
		4500	38	7018	1090.	2498	551.	0.1,?
PTN8	300	500	40	6910	1040.	2380	382.	1,N
		1500	43	10621	1568.	5166	777.	1,?
		3000	43	12161	1789.	5166	777.	1,?
		4500	43	9741	1430.	5166	777.	1,?
PTG8	300	500	69	8843	2123.	7879	1980.	5,?
		1500	69	9854	2340.	7879	1980.	5,?
		3000	69	11542	2660.	7879	1980.	5,?
		4500	69	13016	2952.	7879	1980.	5,?

Quadro 7.2 – Influência do NMAXPV no SLCP

Os resultados no quadro 7.2 mostram que valores demasiado grandes de NMAXPV não são os mais apropriados em termos de velocidade. Contudo NMAXPV = 500 não é a melhor escolha para o problema PTN7 cujo número (r+n) de variáveis complementares é igual a 300. Assim, se se estiver interessado em obter uma boa solução rapidamente, então um valor relativamente pequeno para NMAXPV (≤ 10 (r+n)) deve ser usado e quando (r+n) é grande pode considerar-se $\bar{\gamma} = 0.05$. Se, pelo contrário, se estiver mais interessado na precisão do resultado, um valor maior para NMAXPV deve ser considerado e $\bar{\gamma} = 0.01$ é uma boa escolha.

Assim, para obter os restantes resultados, optou-se pela versão SLCPMET2 com NMAXP=10(r+n) e $\bar{\gamma} = 0.05$ quando (r+n) é maior.

Nos quadros seguintes esta versão do método SLCP é comparada com o método de pesquisa em árvore com limites. Nos quadros 7.3 e 7.4 é analisada também uma outra versão do método de pesquisa em árvore (INCUMBENTE B.B.) cuja solução incumbente inicial é dada pelo algoritmo SLCP. Para cada tipo de problemas foram considerados cinco problemas-teste, e o comportamento dos algoritmos a resolver estes cinco problemas é analisado apresentando os resultados médios (M), o pior (P) e o melhor (B) em termos de operações pivotais NI. Se para um PTi um algoritmo for incapaz de resolver, pelo menos um dos problemas, em menos de 20000 operações pivotais, então na linha encabeçada por (M) é escrito apenas o número de vezes que o algoritmo terminou com sucesso.

Nos quadros 7.3 e 7.4 é apresentada uma coluna OPT com informação semelhante à anterior nas linhas correspondentes a (P) e (B), enquanto que a linha associada a (M) apresenta o número de vezes que o algoritmo SLCP terminou com um mínimo global. Esses resultados conduzem às seguintes conclusões:

(i) o algoritmo SLCP é bastante eficiente para encontrar uma solução ε-óptima (ver coluna OPTIMO). Contudo, quando a dimensão aumenta, torna-se difícil garantir essa optimalidade e grande parte do esforço computacional é gasto a resolver o último LCP;

(ii) o método de pesquisa em árvore com limites é competitivo com o método SLCP para BLLPs de pequena dimensão, sendo nalguns casos até mais eficiente. Contudo é significativamente menos eficiente à medida que a dimensão do BLLP aumenta;

Prob.	DIMENSÃO								SLCP						BRANCH-AND--BOUND			INCUMBENTE B.B			
									TOTAL			OPTIMO									
	n	m	r	rsp	nrow	ncol	NMAXPV		NLCP	NI	T	NI	T	OPT	ND	NI	T	NLCP	ND	NI	T
PTN2	30	50	30	5.0	60	110	600	B	4	75	2.0	75	2.0	0,S	9	108	2.7	4	0	75	2.0
								M	8	299	8.7	262	8.3	3	41	427	12.8	8	5	318	9.4
								P	10	774	23.3	774	23.3	0,S	99	1239	38.7	10	0	774	23.3
PTG2	30	50	30	5.0	60	110	600	B	13	205	6.7	203	6.6	0.1,S	13	138	4.2	9	7	238	7.8
								M	15	406	13.3	405	13.3	4	49	373	11.4	15	4	420	13.8
								P	29	1032	32.3	1032	32.3	0,S	173	1023	31.6	29	0	1032	32.3
PTN4	50	120	50	7.5	100	220	1000	B	3	99	5.5	99	5.5	0,S	3	93	4.8	3	0	99	5.5
								M	11	1719	111.	662	43.2	5	135	4309	290.	11	107	3119	201.
								P	17	4116	270.	1428	93.9	1,S	399	15284	1045.	17	295	8323	551.
PTG4	50	120	50	7.5	100	220	1000	B	9	435	39.1	435	39.1	0,S	11	478	36.8	9	0	435	39.1
								M	18	2681	190.	651	54.3	5	159	3744	261.	18	126	4411	302.
								P	17	8275	565.	735	57.1	1,S	407	10245	738.	17	351	13232	891.
PTN6	100	300	100	7.1	200	500	2000	B	5	517	67.0	517	67.0	0,S	7	355	44.7	5	0	517	67.0
								M	21	5828	746.	3318	431.	4		3			4		
								P	36	14799	1873.	6737	872.	1,?		NS			NS		
PTG6	100	300	100	7.1	200	500	2000	B	5	1298	287.	1298	287.	0,S	5	789	112.	5	0	1298	287.
								M	23	6664	943.	3907	604.	4		4			4		
								P	32	25590	3266.	11970	1588.	1,?		NS			NS		
PTN8	150	250	150	7.1	300	550	3000	B	43	8289	1240.	5166	783.	5,?		NS			NS		
								M			3					0			0		
								P		NS		4015	702.	—		NS			NS		
PTG8	150	250	150	7.1	300	550	3000	B	40	5444	1500.	3969	1248.	0.1,N	129	9171	1803.	40	131	11179	2428.
								M	44	10140	2282.	7454	1752.	—		1			1		
								P	41	16779	3410.	13481	2828.	5,?		NS			NS		

Quadro 7.3 – Solução de BLLPs sem restrições no 1º nível

PTL	DIMENSÃO								SLCP						BRANCH-AND--BOUND			INCUMBENTE B.B			
									TOTAL			OPTIMO									
	n	m	r	psp	nrow	ncol	NMAXPV		NLCP	NI	T	NI	T	OPT	ND	NI	T	NLCP	ND	NI	T
1	20	100	28	17	48	148	480	B	3	94	3.5	94	3.5	0,S	5	68	2.2	3	0	94	3.5
								M	6	161	6.3	160	6.2	3	13	146	4.9	6	3	225	6.6
								P	7	225	9.0	223	9.0	0.1,N	17	210	7.4	6	3	233	9.4
2	20	100	28	40	48	148	480	B	5	173	10.2	173	10.2	0,S	9	153	7.9	5	0	173	10.2
								M	9	252	14.8	249	14.7	3	12	263	13.9	9	5	288	16.8
								P	13	312	20.0	305	19.7	0.1,N	23	491	25.8	12	21	465	28.4
3	40	60	40	17	80	140	800	B	19	1237	59.7	956	46.8	0.1,N	65	1149	53.1	19	45	1772	83.4
								M	22	2337	117.	685	36.8	3	384	7425	329.	22	359	6276	307.
								P	21	2793	133.	623	34.7	1,S	825	11829	541.	27	541	9866	480.
4	40	60	40	40	80	140	800	B	22	864	82.0	626	58.1	0.1,S	55	1569	133.	22	67	1672	150.
								M	24	2709	206.	801	71.3	4		4		24	381	7403	528.
								P	18	6637	458.	641	55.1	1,S		NS		18	1115	18470	1220.
5	40	120	40	17	80	200	800	B	18	935	55.1	777	47.0	0.1,N	115	2081	109.	18	51	1362	78.1
								M	18	2096	125.	1171	72.0	4	486	7089	386.	18	282	4878	287.
								P	16	3435	204.	2226	134.	1,S	925	15900	871.	16	499	8134	475.
6	40	120	40	40	80	200	800	B	14	507	59.0	507	59.0	0,S	5	264	23.5	14	0	507	59.0
								M	17	1106	114.	600	68.2	3	114	2502	215.	17	86	2118	202.
								P	22	1665	153.	637	69.0	1,S	251	5350	487.	15	213	4526	438.
7	70	80	70	17	140	220	1400	B	38	3154	417.	1634	259.	5,?		NS			NS		
								M	35	3624	477.	1901	284.	—		0			0		
								P	29	4196	623.	2514	355.	5,?		NS			NS		
8	70	80	70	40	140	220	1400	B	18	923	259.	907	255.	0.1,N	11	1041	240.	18	3	985	270.
								M	38	3781	930.	2380	585.	—		1			1		
								P	48	4851	1110.	3186	741.	5,?		NS			NS		

Quadro 7.4 – Solução de BLLPs com restrições no 1º nível

Prob.	DIMENSÃO								SLCP					BRANCH AND BOUND		
									TOTAL			OPTIMO				
	n	m	ℓ	r	NC	NRW	NCL		NLCP	NI	T	NI	T	ND	NI	T
1	35	35	7	21	56	63	91	B M P	4 6 6	148 673 1389	6.2 24.2 45.6	107 297 538	4.6 11.7 18.3	9 56 135	475 1096 4183	24. 67.9 137.
2	30	60	9	27	57	66	117	B M P	4 5 8	225 506 763	7.7 18.9 30.	174 219 344	5.8 8.1 13.8	27 47 67	459 1334 2232	15.4 49.7 88.2
3	45	45	9	27	72	81	117	B M P	4 7 9	390 2494 4551	18.2 109. 205.	228 910 2288	11.4 39.9 102.	21	1064 4 NS	49.4
4	30	70	10	30	60	70	130	B M P	7 7 7	375 694 958	17.7 30.7 41.1	260 397 661	12.8 17.8 28.2	135 252 701	3473 7321 17990	142. 321 792.
5	40	60	10	30	70	80	130	B M P	7 9 8	1235 1783 2438	54.2 76.6 110.	929 855 868	43.0 38.9 41.5	217	7519 3 NS	343.
6	50	50	10	30	80	90	130	B M P	10 9 9	1875 3779 5686	94.5 181. 287.	865 1674 2925	43.7 82.3 153.	323	7641 4 NS	421.
7	50	70	12	36	86	98	156	B M P	9 11 9	584 3252 10660	36. 180. 590.	457 1001 2249	28.6 59.1 134.	NS — NS		

Quadro 7.5 – Solução de BLQPs com restrições no 1º nível

(iii) o uso da solução do SLCP como incumbente no método de pesquisa em árvore aumenta a eficiência deste método para problemas de dimensão maior. Contudo, mesmo neste caso, o método de pesquisa em árvore com limites não é competitivo com o algoritmo SLCP;

(iv) o algoritmo SLCP encontrou o mínimo global do BLLP em mais de 60% dos problemas-teste.

No quadro 7.5 é apresentada alguma experiência computacional para BLQPs com restrições no problema do 1º nível. As seguintes conclusões devem ser apresentadas:

(i) tal como nos outros casos o algoritmo SLCP é bastante eficiente para encontrar um óptimo global do BLQP (ver coluna OPTIMO), mas o esforço computacional para estabelecer essa optimalidade é em muitos casos maior que o requerido para obter a solução. A dificuldade deste último LCP (λ_k) é maior para este tipo de problemas BLQPs do que a verificada para BLLPs, como se pode ver comparando com os quadros anteriores;

(ii) o método SLCP é um algoritmo robusto no sentido de ser pequena a variação entre o problema melhor e pior para cada problema PTQi (i = 1,...,7);

(iii) a eficiência do algoritmo SLCP depende da dimensão do BLQP, mas é principalmente influenciada pelo número NC de pares de variáveis complementares do MLCP (7.10);

(iv) o algoritmo SLCP é mais eficiente do que o método de pesquisa em árvore com limites, aumentando significativamente a diferença com o aumento da dimensão do BLQP. Além disso o método de pesquisa em árvore não é robusto no sentido explicado anteriormente.

No quadro 7.6 é analisada a eficiência do método na resolução do BLQP com e sem restrições no 1º nível, e do BLLP com restrições no 1º nível. Para tal fixou-se NC em 60 e 80 e gerou-se para cada NC três problemas:

(i) BLQP com restrições no 1º nível = PTQ4 e PTQ6;

(ii) BLQP sem restrições no 1º nível = PTQ8 e PTQ9;

(iii) BLLP com restrições no 1º nível = PTL9 e PTL10.

Prob.	DIMENSÃO								SLCP					BRANCH AND		
									TOTAL			OPTIMO		BOUND		
	n	m	ℓ	r	NC	NRW	NCL		NLCP	NI	T	NI	T	ND	NI	T
PTQ4	30	70	10	30	60	70	130	B	7	375	17.7	260	12.8	135	3473	142.
								M	7	694	30.7	397	17.8	252	7321	321.
								P	7	958	41.1	661	28.2	701	17990	792.
PTQ8	30	70	0	30	60	60	130	B	5	119	4.1	116	3.9	15	373	12.1
								M	5	308	10.9	153	5.3	55	826	28.0
								P	5	480	17.6	106	3.4	87	1229	43.6
PTL9	30	70	10	30	60	70	130	B	10	241	11.3	171	7.1	53	197	7.2
								M	11	445	18.3	272	10.6	216	1425	104.
								P	15	889	35.2	527	20.	443	4593	167.
PTQ6	50	50	10	30	80	90	130	B	10	1875	94.5	865	43.7	323	7641	421.
								M	9	3779	181.	1674	82.3	4		
								P	9	5686	287.	2925	153.	NS		
PTQ9	50	50	0	30	80	80	130	B	6	251	11.7	203	9.6	61	1675	68.6
								M	8	412	17.4	227	9.7	91	2209	87.3
								P	4	628	26.6	154	6.7	135	3722	146.
PTL10	50	50	10	30	80	90	130	B	6	430	18.8	80	3.1	109	451	18.9
								M	8	852	36.8	231	9.6	575	3705	145.
								P	9	1101	49.1	353	15.	1775	12618	490.

Quadro 7.6 – Comparação entre BLLPs e BLQPs com e sem restrições no 1º nível

Os resultados apresentados no quadro 7.6 mostram que:

(i) os problemas BLQP são mais difíceis de resolver do que o BLLP e a diferença no esforço computacional aumenta significativamente com o aumento do número NC de pares de variáveis complementares do MLCP (7.10);

(ii) a existência de restrições no 1º nível torna o problema de mais difícil resolução, sendo até o BLQP sem restrições no 1º nível muito mais simples de resolver do que o BLLP com restrições no 1º nível. Tal como anteriormente, a diferença de complexidade dos dois tipos de BLQPs aumenta significativamente com um pequeno aumento do número NC.

O estudo computacional apresentado nesta secção mostra que o método de pesquisa em árvore com limites [BAMO90] não é muito eficiente para resolver BLLPs e BLQPs de dimensão razoável. A eficiência deste algoritmo pode no entanto ser melhorada se se usar a solução do SLCP como solução inicial incumbente. Para tal é necessário decidir quando se deve abandonar o método SLCP e começar com o método de pesquisa em árvore com limites. Como normalmente só nos últimos LCP(λ_k) o algoritmo SLCP necessita de um número grande de operações pivotais, um dos critérios que poderá ser utilizado é o que se baseia na quantidade NMAXPV já definida. Assim, se na resolução dum LCP(λ_k) com $k \geq 1$ o método enumerativo híbrido requerer mais de NMAXPV operações pivotais, a solução obtida na iteração $k - 1$ vai ser a solução incumbente para o método de pesquisa em árvore com limites. Tal como já se viu atrás, a experiência mostra que o valor de NMAXPV não deve ser nem muito grande, nem muito pequeno. NMAXPV = 10 (r + n) é possivelmente uma boa escolha.

No entanto, a experiência computacional apresentada com este método para BLLPs de dimensão maior mostra que mesmo assim o método é pouco robusto. Parece importante a investigação de bons limites inferiores para o método de pesquisa em árvore.

A grande dificuldade do algoritmo SLCP reside na necessidade de resolver o último LCP. Por isso este método poderá ser bastante útil na resolução de problemas não lineares por uma sequência de problemas de dois níveis lineares em que não é necessária a obtenção da solução óptima mas apenas uma boa solução ε–óptima.

Prob.	DIMENSÃO								SLCP c/ SPL						SLCP normal				
									SPL			OPTIMO (total)			LCP(λ_0)		OPTIMO (total)		
	n	m	ℓ	r	NC	NRW	NCL		NLP	NI	T	NLCP	NI	T	NI	T	NLCP	NI	T
PTN4	50	120	0	50	100	100	220	B	4	130	4.9	1	130	4.9	74	4.1	3	99	5.5
								M	4	163	7.1	8	689	39.2	136	9.7	10	662	43.2
								P	5	154	7.0	6	1475	88.7	178	14.0	15	1428	93.9
PTG4	50	120	0	50	100	100	220	B	6	383	15.7	3	407	17.1	300	29.9	9	435	39.1
								M	5	475	25.1	10	923	52.8	301	30.2	17	651	54.3
								P	4	433	22.4	10	1807	107.	342	35.4	22	789	63.4
PTL5	40	120	0	40	80	80	200	B	6	189	8.2	9	344	17.9	166	11.9	17	587	40.8
								M	6	236	9.6	7	759	38.3	212	15.2	16	1171	72.0
								P	7	302	14.5	8	1433	79.1	207	14.3	14	2226	134
PTL6	40	120	0	40	80	80	200	B	6	287	21.4	6	380	30.4	192	26.4	13	492	58.3
								M	6	352	28.4	8	541	46.0	222	27.5	15	600	68.2
								P	10	458	41.0	7	742	66.5	259	33.5	16	790	88.8
PTL9	30	70	10	30	60	70	130	B	5	88	1.5	1	88	1.5	41	1.2	7	138	5.2
								M	5	99	2.0	5	224	6.9	54	1.7	10	272	10,6
								P	6	109	2.2	10	463	15.4	78	2.5	14	527	20.0
PTL10	50	50	10	30	80	90	130	B	4	76	1.4	2	99	2.4	25	0.7	5	80	3.1
								M	6	115	2.6	5	274	9.3	54	1.9	7	231	9.6
								P	9	119	2.9	6	457	16.0	64	2.6	8	353	15.0

Quadro 7.7 – Solução do LCP(λ_0) em BLLPs

Tal como já referimos anteriormente, é apresentado em [HAJASA89] um algoritmo tipo pesquisa em árvore que elimina as variáveis x do problema do 2º nível de modo a reduzir a complexidade do BLLP. Apesar de não compararmos totalmente o método SLCP com este método, a experiência computacional indica a superioridade do método SLCP em geral, que além do mais se mostra muito mais robusto no sentido referido anteriormente.

Tal como se referiu atrás o programa bilinear pode ser explorado de modo a obter uma "boa" solução inicial para o BLLP. Os resultados obtidos com este processo SPL são apresentados no quadro 7.7. Os problemas-teste escolhidos foram os problemas de tamanho médio apresentados nos quadros 7.3, 7.4 e 7.6, pois para esses problemas é mais natural que a escolha da solução inicial influencie mais o resultado final.

Os resultados apresentados no quadro 7.7 conduzem às seguintes conclusões:

(i) a inclusão do algoritmo SPL fornece uma melhor solução, evitando por isso a resolução de um número significativo de LCPs;

(ii) o número total de iterações nem sempre é inferior, pois a resolução do LCP(λ_0) usando o algoritmo SPL normalmente requer um maior número de operações pivotais;

(iii) o tempo de execução do SLCP com SPL é normalmente inferior, mesmo quando o número de pivotagens é superior, pois o esforço computacional por operação pivotal no SPL é bastante inferior ao do método enumerativo híbrido.

Pode concluir-se por isso que, apesar do número de pivotagens nem sempre ser melhor, o esforço computacional do método SLCP normalmente diminui se se incorporar o algoritmo SPL. Convém também referir que apesar de o algoritmo SPL não permitir garantir que a solução obtida seja complementar, uma tal solução foi obtida para a quase totalidade dos problemas sem restrições no 1º nível. O mesmo já não se verifica nos problemas com restrições no 1º nível, onde a maior parte das soluções obtidas são não complementares exigindo por isso a utilização do método enumerativo híbrido para obter a solução inicial. No entanto, ao iniciar o método enumerativo híbrido com a solução obtida pelo algoritmo SPL, o método enumerativo é extremamente rápido e fornece uma solução

sem piorar muito o valor de λ_0. Podemos por isso recomendar o uso do algoritmo SPL como processo inicial do método SLCP para o BLLP.

Como conclusão final deste estudo podemos afirmar que o método LCP é um processo bastante eficiente para obter uma solução ε–óptima dos problemas BLLP e BLQP. Tal como no caso do BLP a necessidade de mostrar que o último LCP não tem solução é a parte menos convidativa. Contudo este defeito é menos pronunciado para problemas de dois níveis do que para programas bilineares.

CAPÍTULO 8

PROGRAMAÇÃO QUADRÁTICA NÃO CONVEXA

1 - Introdução

Um Programa Quadrático (QP) é um problema de optimização que consiste em minimizar uma função quadrática sujeita a restrições lineares. Neste capítulo iremos considerar apenas restrições de desigualdade e portanto o problema QP tem a forma

$$\begin{aligned} &\text{minimizar} \quad f(x) = c^T x + \frac{1}{2} x^T Q x \\ &\text{sujeito a} \quad Ax \geq b \\ &\qquad\qquad x \geq 0 \end{aligned} \tag{8.1}$$

onde $c,x \in R^n$, $A \in R^{m \times n}$, $b \in R^m$ e $Q \in R^{n \times n}$ é uma matriz simétrica. Como é usual em problemas de optimização, podemos considerar QPs Convexos e Não Convexos, dependendo da função f ser convexa ou não. Como essa propriedade está associada com o facto de Q ser PSD ou não, então podemos afirmar que

$$\text{QP é Convexo} \Leftrightarrow Q \in \text{PSD}$$

$$\text{QP é Não Convexo} \Leftrightarrow Q \notin \text{PSD}$$

Se um programa quadrático é convexo, então todo o mínimo local é global e pode ser calculado resolvendo o problema linear complementar

$$\begin{bmatrix} w \\ \theta \end{bmatrix} = \begin{bmatrix} c \\ -b \end{bmatrix} + \begin{bmatrix} Q & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \tag{8.2}$$

$$x, u, w, \theta \geq 0 \quad , \quad x^T w = u^T \theta = 0$$

Este problema pode ser resolvido em tempo polinomial [KOMIYO89] e existem vários algoritmos directos e iterativos bastante eficientes para a sua solução [MU88].

À parte uns casos muito especiais em que a solução do LCP (8.2) é um mínimo global do QP não convexo (8.1) [MAR75, cap. 9], a resolução deste tipo de problema é bastante complexa. Se a matriz Q é NSD, então a função f é côncava e o QP diz-se Côncavo. Nesse caso é possível provar que o mínimo global é um ponto extremo do conjunto de restrições do QP

$$S = \{x \in R^n : Ax \geq b, x \geq 0\} \qquad (8.3)$$

Contudo, mesmo neste caso, o QP é um problema NP-completo [PAROS87] e portanto de difícil resolução. Um programa quadrático não convexo e não côncavo diz-se Indefinido, pois o mesmo acontece à sua matriz Q. Nesse caso o mínimo global do QP pode ser alcançado num ponto fronteira de S que não seja ponto extremo [PAROS87].

Existem várias aplicações da programação quadrática não convexa, tanto em problemas quadráticos como na resolução de outros problemas de optimização [PAROS87]. Neste capítulo iremos estudar esse problema com algum detalhe. Após uma descrição sumária de alguns processos que têm sido propostos para a sua resolução, iremos debruçar-nos sobre a utilização do problema MLCP para a obtenção do mínimo global de um programa quadrático não convexo. Alguma experiência computacional indicará as vantagens e desvantagens desse tipo de processo. Finalmente apresentaremos uma aplicação concreta da programação quadrática não convexa na gestão de pequenos aproveitamentos hidroeléctricos.

2 - Métodos de Solução

Como afirmámos anteriormente o mínimo global de um QP côncavo é atingido num ponto extremo do seu conjunto de restrições. Por essa razão os primeiros algoritmos para a resolução deste tipo de problemas basearam-se na busca de pontos extremos. O método mais conhecido desta categoria é devido a Cabot e Francis [CAFR70]. No primeiro passo desse processo uma função linear g(x) é construída de modo a satisfazer

$$g(x) \leq f(x) \qquad \text{para todo } x \in S$$

Para obter essa função resolvem-se n programas lineares da forma

$$\underset{x \in S}{\text{minimizar}} \quad Q_{\cdot j}^T x \qquad (8.4)$$

com $j = 1, 2, \ldots, n$ e $Q_{\cdot j}$ a coluna j de Q.

Se o conjunto S é limitado, cada um destes programas lineares tem um valor finito θ_j e a função g(x) é dada por

$$g(x) = \sum_{j=1}^{n} (c_j + \frac{1}{2}\theta_j)x_j \qquad (8.5)$$

Se x_0 é a solução óptima do programa linear

$$\underset{x \in S}{\text{minimizar}} \quad g(x) \qquad (8.6)$$

e x* é um mínimo global do QP (8.1), então

$$f_\ell = g(x_0) \leq f(x^*) \leq f(x_0) = f_u$$

Portanto x_0 é mínimo global do QP (8.1) se $f(x_0) = g(x_0)$. Caso contrário f_ℓ e f_u são respectivamente um limite inferior e superior da função f. Usando o método de pesquisa de pontos extremos de Murty [MU76, p.142] obtém-se uma sucessão de pontos extremos $\{x_k\}$ de S tais que

$$g(x_{k-1}) \leq g(x_k) \quad \text{para todo k}$$

Além disso sempre que $f(x_k) < f_u$ faz-se $f_u = f(x_k)$. O processo termina quando se encontra um ponto x_t tal que $g(x_t) > f_u$. Nesse caso o ponto extremo x_k tal que $f(x_k) = f_u$ é o mínimo global do QP (8.1).

Da descrição do algoritmo facilmente se conclui que este processo é muito pouco eficiente quando a dimensão do QP aumenta. É ainda de notar que este algoritmo não pode ser aplicado a QPs não convexos indefinidos, por não haver a certeza do mínimo global ser atingido num ponto extremo.

Algoritmos de planos de corte [PAROS87] foram também propostos por vários autores para a resolução de QPs não convexos. A filosofia deste tipo de processos é semelhante à referida no capítulo 6 para a resolução do programa bilinear. Tal como então referimos, esses algoritmos possuem algumas desvantagens que os tornam mais ou menos impraticáveis para QPs não convexos de dimensões razoáveis.

O método mais popular para a resolução de QPs não convexos é devido a Pardalos e Rosen [PAROS87]. Nesse processo o programa quadrático é escrito numa forma um pouco mais geral que consiste em introduzir variáveis y_i que apenas intervêm linearmente no problema. Assim, considera-se o seguinte programa quadrático:

$$\underset{(x,\,y)\in \overline{S}}{\text{minimizar}} \quad f(x,y) = c^T x + \frac{1}{2} x^T Q x + d^T y \tag{8.7}$$

com

$$\overline{S} = \{(x,y): A_1 x + A_2 y \le b,\ x \ge 0,\ y \ge 0\}$$

e $c,x \in R^{n_1}$, $d,y \in R^{n_2}$, $b \in R^m$, $A_1 \in R^{mxn_1}$, $A_2 \in R^{mxn_2}$. O primeiro passo do algoritmo consiste em determinar a decomposição espectral de Q (isto é, os seus valores próprios e vectores próprios associados) e transformar o QP num programa quadrático de variáveis separadas da forma

$$\begin{aligned} &\text{minimizar} && f(x,y) = d^T y + \sum_{i=1}^{n_1} (h_i u_i + \frac{1}{2} \chi_i u_i^2) \\ &\text{sujeito a} && Bu + A_2 y \le b \\ & && y \ge 0 \end{aligned} \tag{8.8}$$

onde $\chi_1,\ldots, \chi_{n_1}$ são os valores próprios de Q associados aos vectores próprios p^i e

$$P = [p^1,\ldots,p^{n_1}],\ B = A_1 P,\ h = P^T c,\ P^T x = u$$

É agora possível encontrar um conjunto Ω^0 que contenha o conjunto de restrições do QP (8.7) se se resolverem 2n programas lineares da forma

$$\underset{(u,y)\in \Omega}{\text{minimizar}}\ u_i \ ,\ \underset{(u,y)\in \Omega}{\text{maximizar}}\ u_i \quad i = 1, 2,\ldots, n_1$$

com Ω conjunto de restrições do programa (8.8). Com efeito, se α_i e β_i são os valores óptimos desses programas lineares, então

$$\Omega^0 = \{(u, y)\in \Omega \ : \alpha_i \le u_i \le \beta_i,\ i = 1, 2,\ldots, n_1\}$$

Considerando as quantidades

$$\psi_i = \begin{cases} \alpha_i & \text{se } \chi_i \ge 0 \\ \beta_i & \text{se } \chi_i < 0 \end{cases} \qquad i = 1,\ldots, n_1$$

a função linear

$$\varphi(u, y) = d^T y + \sum_{i=1}^{n} (h_i + \frac{1}{2} \psi_i \chi_i)\, u_i$$

satisfaz

$$\varphi(u, y) \le f(x, y) \text{ para todo } (x, y)\in \overline{S}$$

Seja (u_0, y_0) o mínimo do programa linear

$$\begin{array}{ll} \text{minimizar} \\ (u,y)\in \Omega^0 \end{array} \quad \varphi(u, y)$$

Então, se (x^*, y^*) é o mínimo global do QP (8.7), tem-se

$$f_{\ell} = \varphi(u_0, y_0) \le f(x^*, y^*) \le f(x_0, y_0) = f_u$$

Se $\varphi(u_0, y_0) = f(x_0, y_0)$, (x_0, y_0) é o mínimo global do QP (8.7). De outro modo é desenvolvido um método de pesquisa em árvore que vai obtendo conjuntos da forma:

$$\Omega^k = \{(u, y)\in \Omega : \alpha_i^k \le u_i \le \beta_i^k,\ i = 1, 2,\ldots, n_1\}$$

com

$$(\beta_i^k - \alpha_i^k) \le (\beta^{k-1}_i - \alpha^{k-1}_i) \text{ para todo k}$$

O processo termina quando se obtiver um conjunto em que $(\beta_i^k - \alpha_i^k)$ seja quase nulo. Nesse caso a solução do programa

$$\begin{array}{ll} \text{minimizar} \\ (u,y)\in \Omega\cap\Omega^k \end{array} \quad \varphi(u, y)$$

é uma solução ε-óptima do QP (8.7).

Experiência computacional apresentada em [PAROS88] para QPs de variáveis separadas, com n_1 relativamente pequeno e n_2 elevado, mostra que o algoritmo é eficiente para n_1 bastante pequeno, mas o seu comportamento piora drasticamente com um aumento de n_1. É ainda de notar que no caso do QP não ser de variáveis separadas é necessário determinar a decomposição espectral da matriz Q e obter o QP (8.8). Essa é uma outra desvantagem deste tipo de processo.

Uma outra abordagem foi estudada em [JU82] e consiste em explorar a complementaridade linear. Com efeito [JUMI88b, JU82], o programa quadrático (8.1) é equivalente ao seguinte MLCP:

$$\begin{array}{ll} \text{minimizar} & c^T x + b^T u \\ \text{sujeito a} & w = \ c + Qx - A^T u \\ & \theta = -\, b + Ax \\ & x,\ u,\ w,\ \theta \ge 0\ ,\ x^T w = u^T \theta = 0 \end{array} \qquad (8.9)$$

Este MLCP pode ser resolvido por um dos dois algoritmos referidos no capítulo 5. Como referimos anteriormente o processo de pesquisa em árvore com limites é normalmente o menos eficiente dos dois processos. Neste caso, e à semelhança da

programação bilinear, é possível provar [JU82] que o programa linear relaxado da condição de complementaridade linear é ilimitado. Para ultrapassar esse inconveniente é obtido um limite inferior a partir da resolução do programa linear (8.6). Se LB é o valor óptimo desse programa, introduz-se a restrição

$$c^T x + b^T u \geq LB$$

Por essa razão e tal como afirmámos anteriormente, é preferível usar o método primal simplex para resolver os vários programas lineares de que o método de pesquisa em árvore necessita.

Uma outra hipótese de resolver o MLCP (8.9) é o algoritmo SLCP descrito no capítulo 5 . É de notar que, tal como afirmámos no capítulo 3, o algoritmo SMRG pode ser usado no caso do QP ser côncavo, enquanto que o algoritmo MRG tem de ser usado na sua forma geral para QP indefinidos. Isto mostra que o método SLCP é mais simples de implementar para QPs côncavos, o que não quer dizer que os QPs côncavos sejam mais fáceis de resolver que os QPs indefinidos.

É ainda importante mencionar que os QPs côncavos podem ser resolvidos a partir de programas bilineares. Com efeito, se $Q \in NSD$, o QP (8.1) é equivalente ao programa bilinear

$$\begin{array}{lll} \text{minimizar} & 2\,f(x, y) = c^T x + c^T y + x^T Q\, y & \\ \text{sujeito a} & Ax \geq b \qquad Ay \geq b & (8.10) \\ & x \geq 0 \qquad y \geq 0 & \end{array}$$

Essa redução foi obtida por Konno [KON76], que propôs o seu algoritmo de planos de corte para programas bilineares para resolver QPs côncavos. Na próxima secção iremos apresentar alguma experiência computacional da resolução de QPs côncavos usando o algoritmo SLCP para a resolução do MLCP (8.9) e do que se obtém do programa bilinear (8.10).

Como referimos no capítulo 1, vários autores têm proposto a solução do LCP a partir da resolução de programas não convexos equivalentes. Assim é sabido que o LCP é equivalente ao programa quadrático

$$\begin{array}{lll} \text{minimizar} & q^T z + \frac{1}{2} z^T (M + M^T)\, z = h(z) & \\ \text{sujeito a} & Mz \geq -q & (8.11) \\ & z \geq 0 & \end{array}$$

e além disso $\bar{z}$ é solução do LCP se e só se $h(\bar{z}) = 0$. Como vimos anteriormente, a grande desvantagem do método SLCP está no último passo onde é necessário provar que um LCP não tem solução. O facto de se saber o valor da função no mínimo global torna desnecessário este último passo. Será por isso interessante atestar da eficiência do método SLCP para a resolução de LCP côncavos e indefinidos. Na próxima secção iremos também discutir a eficiência deste tipo de processo.

3 - Experiência Computacional

Nesta secção iremos apresentar alguma experiência computacional com o método SLCP para a resolução de programas quadráticos côncavos. Para testar a eficiência do método SLCP foram executados num CDC CYBER 180-830 da Universidade do Porto problemas-teste de fontes conhecidas ou gerados aleatoriamente.

A primeira experiência consiste em comparar a eficiência do método enumerativo HYBEN descrito no capítulo 3 com o método SLCP (SLCPQP) para a resolução de LCPs côncavos de média dimensão. Tal como foi afirmado anteriormente, o método SLCP é aplicado à resolução do QP côncavo (8.11) e o processo termina quando $h(z) = 0$. Um outro algoritmo SLCP (SLCPBLP) é também considerado neste estudo computacional, consistindo em aplicar o algoritmo SLCP ao programa bilinear equivalente ao QP côncavo e construído de acordo com a redução explicitada em (8.10).

As matrizes dos problemas-teste foram obtidas considerando matrizes simétricas positivas definidas de fontes conhecidas e multiplicando todos os seus elementos por (–1) para obter matrizes NSD. A técnica de Pang [PAN80] foi usada para gerar os problemas-teste PT1 e PT2. Para gerar PT3, foi utilizada uma matriz associada à solução de equações diferenciais elípticas pelo método das diferenças finitas. Finalmente, para gerar PT4, usou-se uma matriz tridiagonal do bloco diagonal da matriz do tipo anterior. Foram gerados aleatoriamente cinco termos independentes para cada matriz M de cada problema-teste. No quadro 8.1 é apresentado o comportamento dos algoritmos, mostrando o melhor (B), o médio (M) e o pior (P) em termos de operações pivotais. Neste quadro, tal como nos restantes, usam-se as seguintes notações:

n = dimensão do LCP = ordem da matriz

ℓ = número de variáveis do programa quadrático

NI = número de operações pivotais

ND = número de nós

T = tempo de CPU em segundos

NLCP = número de LCPs (λ_k) resolvidos pelo algoritmo SLCP.

Em ambos os métodos SLCP foi utilizado o valor de $\gamma = 10^{-3}$ já recomendado em capítulos anteriores.

			SLCPBLP			SLCPQP			HYBEN		
PT	n		NLCP	NI	T	NLCP	NI	T	TND	NI	T
		B	1	63	1.9	1	20	.69	1	21	.67
1	100	M	4	163	14.	3	100	9.	4	32	1.6
		P	9	255	27.6	7	332	34.3	5	54	3.
		B	1	34	2.8	1	10	.88	1	9	.69
2	300	M	2	67	10.2	2	80	19.	2	18	1.7
		P	4	132	21.7	3	189	50.6	3	36	3.8
		B	7	290	26.1	4	131	11.2	3	96	4.4
3	100	M	5	320	31.8	5	156	26.1	3	126	7.2
		P	3	363	30.3	7	198	56.1	3	138	8.7
		B	1	18	.43	1	14	.36	1	3	.06
4	100	M	2	66	5.	2	64	5.	2	8	.36
		P	4	219	20.6	5	258	23.5	7	23	1.4

Quadro 8.1 - Comparação de SLCP e HYBEN para LCPs côncavos de dimensão média

As seguintes conclusões podem ser tiradas destes resultados:

(i) o algoritmo HYBEN é mais eficiente do que os algoritmos SLCP.

(ii) O algoritmo SLCPBLP parece ser mais robusto do que o algoritmo SLCPQP, já que apresenta uma menor diferença entre o melhor e o pior resultado.

(iii) o algoritmo HYBEN trabalha com LCPs de dimensão menor (e portanto com bases mais pequenas) do que os algoritmos SLCP. Tal facto contribui para um aumento do tempo de execução por operação pivotal nos métodos SLCP.

(iv) Apesar de o algoritmo SLCPBLP trabalhar com bases ligeiramente maiores do que o SLCPQP, o esforço computacional por operação pivotal não é

muito agravado, dado que essas bases são mais esparsas. Além disso verifica-se que de um modo geral o aumento do número de operações pivotais não é muito significativo, pois é igual o número de pares de variáveis complementares em ambos os algoritmos SLCP.

Pode assim concluir-se que para a resolução de LCPs côncavos de grande dimensão se deve usar o algoritmo HYBEN, em vez de se explorar a sua equivalência com problemas de optimização não convexa. Esses resultados permitem também mostrar a grande eficiência dos algoritmos SLCP na resolução de problemas cujo valor óptimo é conhecido, já que não há necessidade de resolver o último LCP inadmissível, cujo esforço é normalmente elevado.

A segunda experiência consistiu em utilizar o método SLCP na resolução dos QPs côncavos descritos em [FLPA90]. As seguintes versões do método SLCP foram consideradas:

(i) SLCPQP, que procura resolver o MLCP da forma (8.9).

(ii) SLCPBLP, onde os QPs côncavos são transformados em programas bilineares equivalentes que são resolvidos pelo algoritmo SLCP descrito no capítulo 6.

(iii) SLCPLBQP – nesta versão o limite inferior LB é primeiramente obtido resolvendo os programas lineares (8.4) e (8.6) segundo um algoritmo semelhante ao LOWBND descrito no capítulo 6. O plano de corte

$$c^T x + b^T u \geq LB$$

é então introduzido no MLCP que é resolvido pelo algoritmo SLCPQP.

(iv) SLCPLBBLP - o mesmo procedimento que em iii), mas usando o algoritmo SLCPBLP.

Probl	ℓ	m	SLCPBLP					SLCPQP							SLCPLBBLP					SLCPLBQP				
			OPTIMO			ULTIMO		OPTIMO			ULTIMO		LOWBND		OPTIMO			ULTIMO		OPTIMO			ULTIMO	
			NLCP	NI	T	NI	T	NLCP	NI	T	NI	T	NI	T	NLCP	NI	T	NI	T	NLCP	NI	T	NI	T
1	5	6	3	22	.15	214	2.6	5	95	1.0	125	1.8	12	.03	3	21	.15	248	3.3	4	167	2.1	165	2.3
2	6	7	1	12	.03	72	.87	1	6	.02	56	.61	22	.05	1	12	.03	0	0	1	6	.01	0	0
3	13	19	5	91	1.2	2062	69.3	3	72	.94	2295	63.4	33	.14	5	91	1.2	2056	70.3	3	67	.89	2260	62.9
4	6	8	1	16	.06	239	3.6	1	8	.03	185	2.1	18	.06	1	16	.06	239	3.8	1	8	.03	185	2.3
5	10	21	1	22	.21	2078	61.0	1	11	.10	1439	27.7	32	.28	1	22	.21	1860	54.0	1	11	.10	1372	27.2
6	10	15	5	271	4.3	11300	224.	6	397	5.4	4902	76.4	34	.21	5	282	4.7	10667	223.	5	281	4.1	4723	79.1
7.1	20	10	24	16786	467.	NS		20	31853	750.	NS		347	3.9	24	16782	480.	NS		20	31599	763.	NS	
7.2	20	10	18	18259	506.	NS		16	34142	799.	NS		348	3.9	18	18258	520.	NS		16	34625	839.	NS	
7.3	20	10	9	13594	351.	NS		12	32654	690.	NS		346	3.9	9	13590	360.	NS		12	32877	701.	NS	
7.4	20	10	5	37529	951.	NS		5	20493	492.	NS		333	3.8	5	37528	974.	NS		5	20614	506.	NS	
7.5	20	10	18	428	10.8	NS		38	11685	281.	NS		342	3.8	18	428	11.2	NS		38	12224	300.	NS	
8	24	20	7	148	3.0	NS		9	940	19.7	NS		143	.95	8	125	2.2	NS		7	327	7.0	23368	517.

Quadro 8.2 – Solução de QP Côncavos de Floudas e Pardalos [FLPA90]

Os resultados das experiências aparecem no Quadro 8.2 onde se usa a notação NS quando o método enumerativo híbrido não consegue resolver o LCP (λ_k) (encontrar a solução ou estabelecer que não existe solução) em menos de 30000 operações pivotais. Nas colunas OPTIMO e ULTIMO regista-se respectivamente o esforço computacional necessário para encontrar a solução óptima e o necessário para mostrar que o LCP não tem solução. O desempenho do algoritmo LOWBND é apresentado nas colunas sob essa designação. Os resultados apresentados no Quadro 8.2 conduziram às seguintes conclusões:

(i) Os quatro algoritmos foram capazes de obter os mínimos globais (ou as melhores soluções) de todos os problemas referidos em [FLPA90]. Nos seis primeiros problemas-teste os processos conseguiram estabelecer que os mínimos globais foram realmente encontrados. Tal não aconteceu para os restantes problemas, tendo apenas o algoritmo SLCPLBQP conseguido mostrar uma única vez que o último LCP (λ_k) não tem solução. Além disso no problema 7.5 os algoritmos obtiveram uma solução melhor do que a referida em [FLPA90]. Esta solução x* é dada por

$$x_3 = 1.0429 \qquad x_{11} = 1.7467 \qquad x_{13} = 0.43147$$
$$x_{16} = 4.4331 \qquad x_{18} = 15.859 \qquad x_{20} = 16.487$$
$$x_i = 0 \ , \ i \neq 3, 11, 16, 18, 20$$

e o valor da função objectivo é – 4150.4102.

(ii) A eficiência dos algoritmos SLCPQP e SLCPBLP na determinação da solução óptima não parece ser muito influenciada pelo uso do algoritmo LOWBND e corte respectivo. Contudo, o desempenho do método enumerativo híbrido para o último LCP (λ_k) é em geral melhorado pelo uso deste corte.

(iii) Em muitos problemas, os algoritmos SLCP são bastante eficientes na determinação da melhor solução (ou mínimo global). Há no entanto alguns problemas-teste nos quais os algoritmos SLCP encontram algumas dificuldades em cumprir aquele objectivo. É importante referir que uma boa solução para estes problemas pode ser encontrada com esforço relativamente pequeno, mas é muito elevado o esforço computacional para determinar a solução óptima.

(iv) O algoritmo SLCPBLP é geralmente mais eficiente que o método SLCPQP para achar a solução óptima (ou mínimo global). No entanto, o SLCPQP comporta-se muito melhor no problema-teste 7.4.

No quadro 8.3 são apresentados resultados para QP côncavos com variáveis separadas do tipo:

$$\text{minimizar} \quad \frac{\rho_1}{2} \sum_{i=1}^{\ell} \chi_i (x_i - \bar{w}_i)^2 + \rho_2 \sum_{i=\ell+1}^{n} c_i x_i$$

$$\text{sujeito a} \quad x \in S$$

onde $\rho_1, \rho_2 \in R$, $\chi, \bar{w} \in R^{\ell}$ e $\chi = (\chi_1, \ldots, \chi_{\ell}) > 0$.

Estes problemas foram gerados aleatoriamente, de acordo com o processo indicado em [PHRO88] para gerar problemas "difíceis". Para cada dimensão foram gerados cinco problemas, dois deles com $\bar{w}$ exterior a S (problemas mais fáceis) e três com $\bar{w}$ pertencente ao interior de S. Tal como foi indicado em [PHRO88] fizemos $\rho_1 = -0.001$ e $\rho_2 = 0.1$ nas nossas experiências.

Para testar a eficiência dos métodos SLCP foi utilizado o algoritmo SLCPBLP que anteriormente se revelara o mais robusto. Assim, cada QP côncavo é reformulado e transformado num BLP do tipo (8.10). O algoritmo LOWBND foi aplicado a cada problema para encontrar um limite inferior LVAL, que é utilizado apenas como critério de paragem quando $\lambda_k \leq$ LVAL.

No quadro 8.3 são ainda introduzidas as seguintes notações:

NC = número de pares de variáveis complementares

NRW = número de linhas do LCP(λ_k)

NCL = número de colunas do LCP (λ_k)

BVAL = melhor solução obtida pelo SLCPBLP

Probl.	DIMENSÃO						LOWBND			SLCP BLP			
	m	n	ℓ	NC	NRW	NCL	NI	T	LVAL	NLCP	NI	T	BVAL
1	20	25	25	45	65	70	503	10.2	-34.7	10	318	8.5	-13.1
							598	12.1	-183.5	1	38	0.4	-183.3
							729	14.6	-172.3	1	30	0.4	-172.0
							665	13.5	-44.5	12	2490	70.1	-26.9
							515	10.3	-42.7	3	31	0.6	-15.6
2	20	125	25	145	165	270	1073	29.0	-193.8	1	42	1.2	-193.8
							1108	44.8	-223.2	1	64	1.6	-223.2
							1074	14.6	-193.2	1	30	1.7	-193.2
							1077	13.5	-212.2	1	2490	2.9	-212.2
							1101	10.3	-191.4	1	31	1.7	-191.4
3	20	225	25	245	265	470	863	49.6	-372.	1	90	3.9	-372.0
							861	25.6	-407.6	1	86	3.6	-407.6
							965	55.6	-361.5	1	66	2.5	-361.5
							991	39.8	-394.3	1	104	4.4	-394.3
							830	15.6	-401.2	1	70	2.8	-401.2

Quadro 8.3 - Solução de QP côncavos de Philips e Rosen [PHRO88]

Para todos os problemas-teste com $\ell < n$, o algoritmo SPL incorporado no método SLCPBLP (ver capítulo 6), encontra o mínimo global para o QP côncavo (NLCP = 1). Para estes problemas, o algoritmo LOWBND é bastante eficiente, pois fornece o valor desse óptimo (LVAL = BVAL). Por isso a solução óptima, é facilmente reconhecida e o método SLCP é muito mais eficiente do que o método de Pardalos e Rosen [PHRO88].

O mesmo tipo de comportamento ocorreu para dois dos QP côncavos puros ($\ell = n$). Tais problemas correspondem a problemas com $\bar{w}$ exterior a S, que são reconhecidos por [PHRO88] como problemas simples. Contudo, para os restantes QP côncavos em que $\bar{w}$ é interior a S, o algoritmo SLCPBLP já requer mais do que um LCP para o resolver e não consegue terminar antes das 20000 operações pivotais. Este tipo de problemas é considerado em [PHRO88] como o de mais difícil resolução, com o método aí descrito a necessitar de 60000 operações pivotais para encontrar um valor ε-óptimo com $\varepsilon = 0.001$.

Podemos por isso considerar que o algoritmo SLCP é superior ao de Pardalos e Rosen, pelo menos para este tipo de problemas-teste.

Convém no entanto referir que, em qualquer dos casos, se o algoritmo LOWBND não tivesse sido utilizado, o algoritmo SLCP seria incapaz de estabelecer em menos de 20000 operações pivotais que o óptimo global tinha sido encontrado.

Como conclusão final, verifica-se que os resultados apresentados neste capítulo vêm confirmar os já obtidos em capítulos anteriores mostrando que o algoritmo SLCP é bastante promissor como método global. Permanece contudo a necessidade de desenvolver processos eficientes que permitam acelerar o método SLCP na parte final de alguns problemas difíceis e também de eliminar o elevado custo do último LCP.

4 - Uma Aplicação da Programação Quadrática Não Convexa

Nesta secção apresenta-se uma aplicação da programação quadrática não convexa num modelo de gestão integrada de pequenas centrais hidroeléctricas (mini-hídricas) que se situam numa mesma linha de água [SOFA90]. Trata-se de um instrumento de análise à viabilidade e interesse em se gerir a capacidade de regularização conjunta de uma série de pequenos reservatórios em cascata.

O desenvolvimento deste modelo gera um programa quadrático não convexo, com uma condição adicional de complementaridade, que é resolvido iterativamente pelo método SLCP, usando o método enumerativo híbrido descrito no capítulo 2 desta tese.

A reduzida capacidade de regularização, geralmente associada a pequenos aproveitamentos hidroeléctricos, conduz a modelos de exploração do tipo "fio-de-água" com arranque e paragem das turbinas reguladas pelo nível da albufeira. Em alternativa, procurou formular-se um modelo que se aplicasse a sistemas de mini-hídricas no mesmo curso de água, aceitando a sua interdependência e tirando partido da capacidade de regularização conjunta de modo a optimizar a produção energética do sistema.

A configuração-base do sistema de mini-hídricas que foi objecto de modelação prevê as seguintes características fundamentais:

- açude galgável;
- central no pé do açude com restituição próxima;
- descarga de fundo, como solução mais corrente de descarga auxiliar.

Com esta configuração, designadamente na consideração de vazão sobre o açude, procura encontrar-se uma situação abrangente relativamente a outras configurações de mini-hídricas, onde estruturas tais como câmaras de carga retiram a importância daquela vazão na definição da queda útil do aproveitamento.

Foi assim criado um modelo matemático que aproxima suficientemente bem a realidade e que, tendo como objectivo primeiro a optimização da energia produzida por um sistema numa dada sequência de unidades de tempo, pode também ser aplicado a uma mini-hídrica funcionando isoladamente, e permite a comparação dos respectivos resultados com os modelos de exploração mais vulgarmente utilizados.

4.1 - Construção do Modelo de Optimização

I - Variáveis do problema

O estudo de um sistema de mini-hídricas envolve os seguintes caudais integrais, com i e j designando respectivamente a posição da mini-hídrica no sistema e o número de ordem do intervalo de tempo em estudo:

- volumes afluentes Q(i, j);
- volumes DS(i, j) descarregados pelo açude;
- volumes DA(i, j) descarregados pela descarga auxiliar;
- volumes T(i, j) turbinados;
- volumes V(i, j) na albufeira i no final do período de tempo j.

Sendo Q(i, j) um dado do problema, as restantes quatro variáveis contêm um grau de dependência traduzido pela equação de balanço hídrico

$$V(i,j) = V_{i,0} + \sum_{k=1}^{j} Q(i,k) - \sum_{k=1}^{j} [DS(i,k) + DA(i,k) + T(i,k)] + \sum_{k=1}^{j} [DS(i-1,k) + DA(i-1,k) + T(i-1,k)] \qquad (8.12)$$

onde $V_{i,0}$ representa o volume inicial no reservatório i.

As variáveis do problema, para um estudo que envolva n mini-hídricas e m períodos de tempo, serão em número de 3 n m e dadas por DS(i, j), DA(i, j) e T(i, j).

II - Função Objectivo

A energia produzida por n mini-hídricas é dada por

$$E = \sum_{i=1}^{n} ETA_i \, \mu \, g \, h_i \, T_i \qquad (8.13)$$

com μ e g valores constantes, ETA_i o rendimento do conjunto turbina-alternador, h_i a queda útil média explorável e T_i o volume turbinado.

Dado que é h_i função do volume médio da albufeira no intervalo de tempo em estudo, da carga sobre o açude devida a DS e do nível de água na restituição dependente de DS + DA + T, e como o rendimento é também função dos três tipos de variáveis, a equação (8.13) pode apresentar a formulação geral

$$E = \mu g \, f(DS, DA, T) \; H(DS, DA, T) \, T \qquad (8.14)$$

Sendo perfeitamente definida a contribuição do volume T como factor, optou-se por linearizar toda a restante expressão para que a função objectivo pudesse ser quadrática. Para tal, adoptaram-se aproximações da curva do rendimento (por patamares) e das curvas que contribuem para a queda útil (com interpolações lineares segmentadas), consideradas preferíveis às simplificações de natureza matemática em que se incorreria se se optasse pela resolução de um problema de programação de grau superior ao quadrático.

No que diz respeito à curva de rendimentos f(DS, DA, T), modificada para introdução das perdas de carga no circuito hidráulico, o intervalo de discretização adoptado determina o número de patamares que são convenientemente indexados.

Quanto à queda útil H(DS, DA, T), cada uma das rectas de interpolação das curvas h(V) (de capacidade do reservatório), h(Q) (da vazão sobre o açude) e h(A) (da altura de água na restituição) foi assimilada a uma expressão do tipo

$$y = k_1 + k_2 \, x$$

sendo os parâmetros k_1 e k_2 indexados sob a forma k(t, i, ℓ) o que permite referenciá-los ao tipo de curva e ordem do parâmetro, ao reservatório a que se aplica, e à sua localização na sequência de rectas de cada interpolação.

A expressão (8.14) que constitui a função objectivo do problema, pode ser apresentada na forma

$$c^T x + \frac{1}{2} x^T Q x \qquad (8.15)$$

em que o vector x terá 3nm variáveis, correspondentes a valores de DS(i, j), DA(i, j) e T(i, j) para cada reservatório e cada unidade de tempo. O vector c tem apenas componentes não nulas associadas a T(i, j) e a matriz Q terá elementos não nulos associados a $T(i, j)^2$ e aos produtos de T(i, j) com DS(i, j) e DA(i, j), que são definidos à custa das leis de vazão do açude e de descarga de fundo.

III - Condições de Restrição

As condições de restrição exigidas pelo problema são do tipo $x \geq 0$ (todas as variáveis só podem tomar valores não negativos) e ainda da forma $Ax \geq b$, resultantes de outros limites a impor. Assim

$$V_{m,i} \leq V(i, j) \leq V_{M,i} \qquad (8.16)$$

determinam os volumes mínimo e máximo no reservatório i. O volume máximo de turbinamento para o intervalo de tempo em estudo é imposto pela condição

$$T(i, j) \leq T_{M,i} \qquad (8.17)$$

e as condições de volume máximo ou mínimo a ser produzido pela descarga auxiliar no mesmo intervalo de tempo são dadas pelas expressões

$$DA(i, j) \leq DA_{M,i}$$

e/ou (8.18)

$$DA(i, j) \geq DA_{m,i}$$

Com uma destas últimas condições (8.18) e as condições (8.16) e (8.17) desenvolvidas de acordo com a expressão (8.12), o modelo gera a matriz das restrições A de ordem 4nm por 3nm e o vector b com 4nm componentes, função dos limites das condições de restrição.

IV - Condição Adicional de Complementaridade

Podem ocorrer situações em que a transferência de água de uma mini-hídrica para a situada imediatamente a jusante seja exigida pelas condições de optimização energética. Nos casos em que a indisponibilidade da central não seja total, o modelo poderia recorrer

à descarga no açude, por ser mais favorável em termos de queda útil, sem que o reservatório estivesse cheio e violando a realidade física.

Porque havia de remeter essa transferência, em tais casos, para a descarga de fundo, houve que recorrer a uma condição adicional de complementaridade, entre o volume descarregado pelo açude e o volume livre da albufeira, do tipo

$$s(i, j) = DS(i, j) = 0$$

onde s(i, j) é uma variável de afastamento dada por

$$s(i, j) = V_{M,i} - V(i, j) \tag{8.19}$$

Impõe-se assim que só haja descarga pelo açude (DS(i, j) > 0) quando o reservatório estiver cheio (s(i, j) = 0) e que, por outro lado, estando este abaixo do nível máximo (s(i, j) > 0) terá a descarga superior de ser nula (DS(i, j) = 0).

Passaremos então a ter um problema quadrático com uma condição de complementaridade cuja resolução é feita à custa de um problema equivalente do tipo

$$\begin{aligned} &\text{minimizar} \quad -(c^T x + \frac{1}{2} x^T Q x) + \gamma\, s^T x_D \\ &\text{sujeito a} \quad Ax \geq b \\ &\qquad\qquad x \geq 0 \end{aligned} \tag{8.20}$$

onde x_D é um sub-vector de x correspondendo às componentes do tipo DS(i, j), s o vector das variáveis s(i, j) definidas em (8.19) e γ um escalar positivo e suficientemente grande para que a minimização de $s^T x_D$ ocorra para o valor zero.

Caso o problema quadrático seja resolvido satisfazendo em si mesmo a condição de complementaridade, torna-se desnecessária a intervenção de γ. No entanto, no caso contrário, este parâmetro terá um efeito de penalização sobre a nova parcela da função objectivo e será incrementado, tanto quanto necessário, até que a contribuição dessa parcela se torne nula.

O desenvolvimento de $s^T x_D$ nas variáveis do problema conduz a uma expressão com a forma

$$s^T x_D = v^T x + \frac{1}{2} x^T C x$$

onde C é uma nova matriz quadrada de ordem 3nm e v um vector de 3nm componentes [SOFA90].

4.2 - Resolução do Problema de Optimização

Como se infere dos pontos anteriores, o problema quadrático (8.20) passa a ser do tipo

$$\begin{aligned} \text{minimizar} \quad & -(c^T x + \frac{1}{2} x^T Q\, x) + \gamma\, (v^T x + \frac{1}{2} x^T C\, x) \\ \text{sujeito a} \quad & Ax \geq b \\ & x \geq 0 \end{aligned}$$

onde nenhuma das matrizes Q e C é positiva semi-definida. A resolução do problema é então equivalente a um MLCP do tipo

$$\begin{aligned} \text{minimizar} \quad & (-c^T + \gamma\, v)^T x + b^T u \\ \text{sujeito a} \quad & w = (-c^T + \gamma\, v) + (-Q + \gamma\, C)\, x - A^T u \\ & \theta = -b + A\, x \\ & x, u, w, \theta \geq 0 \\ & w^T x = u^T \theta = 0 \end{aligned} \qquad (8.21)$$

onde Q não é constante, dependendo da combinação dos intervalos relativos às interpolações lineares das curvas que definem a queda útil. O problema quadrático terá por isso de ser resolvido iterativamente até haver coincidência entre todos os intervalos de entrada (cada um deles função de variáveis que designamos por VS) e os intervalos correspondentes à última solução do problema.

Considerando os novos vector e matriz

$$\bar{c} = -c + \gamma\, v \;,\; \bar{Q} = -Q + \gamma\, C$$

o algoritmo do problema é então como se segue:

Passo 1 - Fixe $\bar{Q}$ inicialmente, de acordo com o conjunto de arranque das variáveis VS (abcissas das curvas que definem H) e com $\gamma = 0$ (não impondo a condição adicional de complementaridade).

Passo 2 - Resolva o MLCP (8.21). Se a solução obtida satisfizer a condição adicional de complementaridade vá para o Passo 4. Caso contrário vá para o Passo 3.

Passo 3 - Altere $\bar{Q}$, com incremento do tipo

$$\gamma = \gamma_i = \gamma_{i-1} + \gamma_{inc}$$

(onde γ_{inc} é o incremento de γ) para forçar a complementaridade a ser satisfeita, e volte ao Passo 2.

Passo 4 - Construa um novo conjunto de VS a partir da solução obtida, e verifique a correspondência entre os intervalos de interpolação ligados àquele conjunto e os referentes aos valores de VS que intervieram na construção de $\bar{Q}$. Se tal se verificar, o processo termina e obteve-se a solução do problema. Caso contrário, $\bar{Q}$ é redefinida com as VS actualizadas e regressa-se ao Passo 2, tomando-se sempre que possível para solução inicial uma solução básica cujas variáveis básicas e não básicas têm índices iguais às variáveis básicas e não básicas da última solução obtida.

Para a resolução do MLCP (8.21) utiliza-se o algoritmo SLCP descrito neste capítulo. Como se trata de um programa indefinido, o método enumerativo híbrido incorpora o método de gradiente reduzido modificado MRG para a busca de direcções descendentes.

4.3 - Experiência Computacional

O modelo foi ensaiado para situações existentes, com dados de exploração conhecidos, e ainda para sistemas múltiplos de mini-hídricas e diversas sequências de caudais, com diferentes intervalos de tempo.

Os ensaios confirmaram qualitativamente as expectativas, com a obtenção das seguintes conclusões:

(i) a gestão integrada é, do ponto de vista energético, mais favorável que a gestão convencional, permitindo ainda a racionalização dos recursos no cumprimento de outros objectivos (abastecimento de água, regas, controle de poluição, etc.);

(ii) em termos percentuais, a gestão integrada assume um maior interesse para maiores relações "volume total dos reservatórios-caudais afluentes";

(iii) a capacidade de gestão integrada é maior quando os maiores reservatórios se situam a montante;

(iv) a maior eficiência de gestão integrada não é obtida à custa de perdas significativas para qualquer das mini-hídricas do sistema;

(i) o modelo de gestão é ainda, quando aplicado a mini-hídricas isoladas, mais favorável que o modelo de gestão convencional.

Do ponto de vista computacional, o modelo revelou-se consistente, com tempos de execução que tornam possível a operação em tempo real e, embora sem ter sido demonstrado, sempre convergente no processo iterativo resultante da utilização de interpolações segmentadas. Além disso está adequado à variação de todos os parâmetros envolvidos no problema, podendo ser simulados graus de discretização no tempo sem grandes limitações. Os reduzidos tempos de execução resultam em parte do facto de se ter deixado de exigir a resolução do LCP (λ_k) inadmissível sempre que $\gamma = 0$, já que se verificou em todos os ensaios realizados que a solução óptima ocorria para o LCP (λ_0). Na verdade, a intervenção da condição adicional de complementaridade só em condições excepcionais de exploração poderá ser exigida e conduzir portanto à extensão desses tempos de execução. Poderá isto querer dizer que, nas situações correntes de funcionamento do sistema, a solução óptima global é única.

CAPÍTULO 9

PROGRAMAÇÃO QUADRÁTICA CÔNCAVA COM APENAS LIMITES NOS VALORES DAS VARIÁVEIS

1 - Introdução

Neste capítulo iremos estudar programas quadráticos (BCQP) da forma

$$\begin{aligned} &\text{minimizar} \quad f(x) = q^T x + \frac{1}{2} x^T M x \\ &\text{sujeito a} \quad 0 \le x \le b \end{aligned} \tag{9.1}$$

com $q, x \in R^n$, $b = (b_1,\ldots, b_n)^T$, $0 < b_i < +\infty$ para todo $i = 1,\ldots, n$ e M uma matriz simétrica negativa semi-definida (NSD) de ordem n. Este problema é NP-completo [PAROS87] e portanto de difícil resolução. Além disso o mínimo global deste programa quadrático ocorre num ponto extremo do hiper-rectângulo

$$\Omega = \{x \in R^n : \ 0 \le x \le b\} \tag{9.2}$$

e pode por isso ser obtido a partir da resolução do seguinte programa quadrático de vairáveis 0–1 (QP 0–1):

$$\begin{aligned} &\text{minimizar} \quad c^T y + \frac{1}{2} y^T Q y \\ &\text{sujeito a} \quad y_i \in \{0, 1\} \ , \quad i = 1,\ldots, n \end{aligned} \tag{9.3}$$

onde $y_i = \dfrac{x_i}{b_i}$, $c_i = q_i b_i$ e $q_{ij} = m_{ij}\, b_i\, b_j$, $i, j = 1,\ldots, n$.

Inversamente qualquer QP 0–1 da forma (9.3) pode ser transformado num QP côncavo da forma

$$\begin{aligned} &\text{minimizar} \quad c^T y + \frac{1}{2} y^T Q y + \tau\, y^T (e - y) \\ &\text{sujeito a} \quad 0 \le y \le e \end{aligned} \tag{9.4}$$

com $e = (1,\ldots, 1)^T \in R^n$ e τ um número suficientemente grande de modo a que $(Q - 2\tau I_n) \in NSD$.

A determinação de um mínimo global do QP côncavo com apenas limites nos valores das variáveis tem suscitado grande interesse nos últimos anos. Com efeito, além da importância do problema em si próprio, há a considerar vários problemas combinatórios que têm formulações do tipo QP 0–1. Sugerimos [PAROD89] para uma revisão das aplicações mais importantes deste tipo de problemas.

Devido à sua complexidade, apenas um método de pesquisa em árvore é capaz de resolver o QP côncavo em todos os casos. Vários processos heurísticos têm vindo a ser desenvolvidos de modo a tornar mais reduzida a pesquisa de um mínimo global [GUGUMI84, BAJURE89, PAROD90, PAR89]. Neste capítulo iremos desenvolver técnicas para esse fim e discutiremos as suas implementações e incorporações num processo do tipo pesquisa em árvore com limites. Se os elementos não diagonais da matriz M são não positivos, então o BCQP pode ser reduzido a um problema de corte mínimo de uma rede [PIRA74]. As técnicas referidas anteriormente podem neste caso reduzir a dificuldade da resolução de um tal problema, pois permitem marcar algumas variáveis e reduzir a dimensão da rede. Esse assunto será também discutido neste capítulo.

2 - Métodos de Resolução

Como referimos na secção anterior existem dois processos para resolver o BCQP, nomeadamente o método de pesquisa em árvore com limites e a sua redução a um problema de redes. Seguidamente iremos discutir sumariamente estes dois tipos de processos.

2.1 - Algoritmo de Pesquisa em Árvore com Limites

Como já foi discutido em capítulos anteriores, o método de pesquisa em árvore com limites é um processo bastante geral que vai decompondo o problema a resolver em subproblemas de menor dificuldade que se obtêm fixando os valores de algumas variáveis. Esses subproblemas são os ramos de uma árvore que vai assim sendo explorada até se obter o mínimo global. Em relação ao BCQP, em cada nível da árvore são considerados dois subproblemas que se definem fixando uma dada variável x_i a um

dos limites 0 ou b_i. Para uma descrição do algoritmo consideremos numa dada iteração os conjuntos

$$J_0^k = \{i : x_i = 0\} \ , \ J_1^k = \{i : x_i = b_i\}$$

e seja $J_\ell^k = \{1,\ldots, n\} - (J_0^k \cup J_1^k)$ o conjunto das variáveis livres (isto é, não fixadas a nenhum dos limites inferior ou superior). Então podemos apresentar os passos do algoritmo na seguinte forma:

ALGORITMO DE PESQUISA EM ÁRVORE COM LIMITES

Passo 0 - Seja $k = 0$, $J_0^k = J_1^k = \emptyset$ e $J_\ell^k = \{1,\ldots, n\}$. Faça

$$\text{VINC} = \begin{cases} f(\bar{x}) \text{ se } \bar{x} \text{ é um mínimo local conhecido} \\ +\infty \text{ de outro modo} \end{cases}$$

Passo 1 - (i) Se $J_\ell^k = \emptyset$, seja $\bar{x}$ a solução definida por J_0^k e J_1^k. Se $f(\bar{x}) < \text{VINC}$, então $\bar{x}$ é a nova solução incumbente. Faça $\text{VINC} = f(\bar{x})$ e vá para Passo 3.

(ii) Se $J_\ell^k \neq \emptyset$, escolha uma variável x_r, $r \in J_\ell^k$ para ser fixada e vá para Passo 2.

Passo 2 - Crie dois novos nós, retirando r de J_ℓ^k e acrescentando-o a J_0^k e J_1^k respectivamente.

Passo 3 - Se não houver nós a explorar, a última solução incumbente é um mínimo global do BCQP. De outro modo escolha um dos nós por explorar (nó k) e vá para Passo 1.

Este algoritmo determina o mínimo global de um BCQP explorando os $2^{n+1} - 1$ nós da árvore binária. No entanto, a pesquisa exaustiva desta árvore é proibitiva mesmo para valores de n suficientemente pequenos. O grande objectivo dos utilizadores deste tipo de técnicas consiste por isso em desenvolver processos que reduzam drasticamente a pesquisa da árvore binária. Na secção 3 iremos apresentar um algoritmo bastante eficiente para esse fim.

A redução da pesquisa também depende da maneira como se escolhe a variável no Passo 1 e o nó no Passo 3. Seguidamente iremos descrever sumariamente como são feitas essas escolhas. Esses processos são baseados em algumas das ideias apresentadas em [PAROD89, PAROD90, GUGUMI84].

Assim, no que se refere ao Passo 3 e contrariamente ao que é proposto no capítulo 2, o nó a escolher é o último nó gerado. Esta regra é de muito mais fácil implementação do que a usada no método enumerativo do capítulo 2. Essa outra regra justificava-se pelo facto deste último algoritmo apenas procurar uma solução complementar. No caso do algoritmo desta secção, a determinação de um ponto estacionário do BCQP não implica de modo nenhum a sua terminação. Daí o esforço computacional adicional que uma regra mais elaborada acarreta não compensar neste caso.

Para a escolha do índice r no Passo 2, usamos a seguinte regra sugerida em [PAR89]:

$$\delta_r = \max \{\delta_i = \min \{-\ell_i^k, u_i^k\}, i \in J_\ell^k\} \tag{9.5}$$

com

$$\ell_i^k = \bar{q}_i + \frac{1}{2} m_{ii} b_i + \sum_{\substack{j \in J_\ell^k \\ j \neq i}} \bar{m}_{ij} b_j$$

$$u_i^k = \bar{q}_i + \frac{1}{2} m_{ii} b_i + \sum_{\substack{j \in J_\ell^k \\ j \neq i}} m_{ij}^+ b_j \tag{9.6}$$

e

$$\bar{q}_i = q_i + \sum_{j \in J_1^k} m_{ij} b_j \tag{9.7}$$

$$\bar{m}_{ij} = \min \{m_{ij}, 0\} \ , \ m_{ij}^+ = \max \{m_{ij}, 0\}$$

Para além destas duas regras heurísticas, usamos em cada nó k um procedimento que foi sugerido em [PAR89] e que permite em muitos casos podar ramos da árvore. Para a sua descrição, sejam J_0^k, J_1^k e J_ℓ^k os conjuntos mencionados anteriormente e P^k o problema a resolver no nó k. Seja ainda VINC o valor da incumbente e consideremos uma função g que satisfaz às seguintes condições:

$$\begin{cases} g(P^k) \leq f(P^k) \\ g(P^k) = f(P^k) \text{ se } J_\ell^k = \emptyset \\ g(P^k) \geq g(P^i) \text{ se } P^k \text{ é um subproblema de } P^i \end{cases}$$

onde f é a função do BCQP. Então $g(P^k) \geq$ VINC implica que nenhum dos nós abaixo de P^k precisa de ser investigado, por não conduzir a qualquer ponto estacionário do BCQP com valor inferior a VINC. Para a construção da função limite inferior g podemos usar o valor LB obtido pelo algoritmo LOWBND aplicado ao BCQP. Assim tem-se

$$g(P^0) = \sum_{i=1}^{n} (q_i + \frac{1}{2} m_{ii} b_i + \frac{1}{2} \sum_{\substack{j=1 \\ j \neq i}}^{n} m_{ij}^{-} b_j)^{-} b_j$$

e para $k \geq 1$

$$g(P^k) = f(P^k) + \sum_{j \in J_\ell^k}^{n} (\bar{q}_i + \frac{1}{2} m_{ii} b_i + \frac{1}{2} \sum_{\substack{j \in J_\ell^k \\ j \neq i}}^{n} m_{ij}^{-} b_j)^{-} b_i$$

onde $\rho^{-} = \min \{\rho, 0\}$ e $\bar{q}_i$ é dado por (9.7). É de notar que $g(P^k)$ é um melhor limite inferior para $f(P^k)$ do que o usado em [PAR89].

2.2 - Redução a um Problema de Corte Mínimo de uma Rede

Uma rede é representada por G = (V, A), onde V representa o conjunto de n+2 nós $v_0 \ldots, v_{n+1}$ e A o conjunto dos arcos que ligam esses nós. Tal como é referido em [PIRA74], numa rede podemos associar a cada par de nós v_i e v_j os seguintes arcos:

(i) um arco orientado no sentido de v_i para v_j com capacidade c_{ij}^d.

(ii) um arco orientado no sentido de v_j para v_i com capacidade c_{ji}^d.

(iii) um arco não orientado com capacidade $c_{ij}^u = c_{ji}^u$.

Os nós v_0 e v_{n+1} são normalmente designados por **fonte** e **destino**. Um **corte** $(S, \bar{S})$ separando os nós v_0 e v_{n+1} é uma partição dos nós de V tal que $v_0 \in S$ e $v_{n+1} \in \bar{S}$. Chama-se **capacidade** de $(S, \bar{S})$ à soma das capacidades dos arcos que unem um nó de S a um nó de $\bar{S}$. Assim

$$C(S, \bar{S}) = \sum_{i \in I} \sum_{j \in \bar{I}} (c_{ij}^d + c_{ij}^u) \qquad (9.8)$$

com

$$I = \{i: v_i \in S\} \ , \ \bar{I} = \{i: v_i \in \bar{S}\}$$

Finalmente um **corte mínimo** separando os nós v_0 e v_{n+1} é o corte que separa v_0 e v_{n+1} e tem capacidade mínima.

Tal como é referido em [PIRA74], a cada corte $(S, \bar{S})$ podemos associar um vector $(1, x_1, x_2, \ldots, x_n, 0)$ com $x_i \in \{0, 1\}$ tal que $S = \{v_i : x_i = 1\}$ e $\bar{S} = \{v_i : x_i = 0\}$. Usando esta notação, a capacidade do corte $(S, \bar{S})$ vem dada por

$$C(S, \overline{S}) = \sum_{i=0}^{n+1} \sum_{j=0}^{n+1} c_{ij}^{d} x_i (1 - x_j) + \frac{1}{2} \sum_{i=0}^{n+1} \sum_{j=0}^{n+1} c_{ij}^{u} (x_i - x_j)^2$$

Fazendo $x_0 = 1$, $x_{n+1} = 0$ e atendendo a que $x_i \in \{0, 1\}$ para todo $i = 1,\ldots, n$ e $c_{ij}^{u} = c_{ji}^{u}$, podemos escrever

$$C(S, \overline{S}) = \sum_{j=1}^{n+1} c_{0j} + \sum_{i=1}^{n} (c_{i,n+1} - c_{0i} + \sum_{j=1}^{n} c_{ij})\, x_i - \\ - \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_i\, x_j$$

onde $c_{ij} = c_{ij}^{d} + c_{ij}^{u}$. Assim o problema de determinação de um corte mínimo pode ser formulado num QP 0–1 da forma

$$\begin{aligned} &\text{minimizar} \quad \rho + b^T x + x^T Q\, x \\ &\text{sujeito a} \quad x_i \in \{0, 1\}\ ,\ i = 1,\ldots, n \end{aligned} \tag{9.9}$$

com

$$\rho = \sum_{j=1}^{n+1} c_{0j}\ ,\quad b_i = c_{i,n+1} - c_{0i} - \sum_{j=1}^{n} c_{ij}\, x_j\ ,\quad q_{ij} = -\, c_{ij}$$

Inversamente, dado um QP 0–1 (9.9), é possível encontrar uma rede orientada ou não orientada cujo corte mínimo fornece um mínimo global para o programa quadrático, sendo o seu valor óptimo igual à capacidade desse corte mínimo. Assim [PIRA74], o QP 0–1 (9.9) é equivalente à determinação do corte mínimo da rede não orientada cujas capacidades são definidas por

$$c_{ij}^{u} = c_{ij} = -\, q_{ij}$$

$$c_{i,n+1}^{u} = c_{i,n+1} = \max \{\sum_{j=1}^{n} q_{ij} + b_i\ ,\ 0\}$$

$$c_{0i}^{u} = c_{0i} = \max \{-\sum_{j=1}^{n} q_{ij} - b_i\ ,\ 0\}$$

Além disso a determinação do corte mínimo dessa rede reduz-se a um problema de fluxo máximo, pois verifica-se o seguinte teorema [PAST82, pp 119-120]:

Teorema 9.1 – O fluxo máximo de v_0 para v_{n+1} é igual à capacidade do corte mínimo. Além disso o fluxo f e o corte $(S, \overline{S})$ são óptimos se e só se

$$f_{ij} = 0 \quad \text{se } v_i \in \overline{S} \text{ e } v_j \in S$$
$$f_{ij} = c_{ij} \quad \text{se } v_i \in S \text{ e } v_j \in \overline{S}$$

Se as capacidades c_{ij} são de sinal arbitrário, o problema de fluxo máximo é muitas vezes impossível de resolver [PIRA74]. Contudo, no caso de todas as capacidades serem não negativas, o problema do fluxo máximo pode ser resolvido em tempo polinomial [PAST82]. Após a obtenção do fluxo máximo, o corte mínimo pode também ser obtido em tempo polinomial usando o método de marcação de Ford-Fulkerson [PAST82, pp. 120]. Daqui se conclui que o BCQP e o QP 0–1 podem ser resolvidos em tempo polinomial desde que os elementos não diagonais de matriz sejam todos não positivos. A experiência computacional tem indicado que o método simplex é normalmente mais eficiente que os métodos polinomiais existentes para o problema do fluxo máximo [GLKLMOWH79], principalmente quando o número de nós da rede aumenta. Na última secção deste capítulo apresentaremos alguma experiência computacional na resolução de alguns QP 0–1 usando este tipo de abordagem. O problema BCQP pode também ser resolvido de modo semelhante desde que seja previamente transformado num QP 0–1, usando a redução (9.3).

3 - Algoritmo para a Determinação de uma Boa Solução Incumbente e Marcação de Variáveis

Como referimos no capítulo 8, o BCQP é equivalente a um MLCP da forma

$$\begin{aligned} \text{minimizar} \quad & z_0 = q^T x - b^T \beta \\ \text{sujeito a} \quad & w = q + Mx - I_n \beta \\ & v = b - I_n x \\ & w, v, x, \beta \geq 0 \ , \ x^T w = v^T \beta = 0 \end{aligned} \tag{9.10}$$

Na secção 3 do capítulo 4 foram introduzidos algoritmos pivotais principais capazes de obter uma solução do LCP (9.10) em que todas as variáveis x_i têm valores iguais a um dos limites 0 ou b_i. Tal solução é em geral um mínimo local do BCQP. Nesta secção iremos descrever uma técnica bastante simples que permite obter um LSM, isto é, um ponto extremo $\bar{x}$ do conjunto Ω de restrições do BCQP, tal que $f(\bar{x}) \leq f(x)$ para todo o ponto extremo de Ω adjacente a $\bar{x}$. Iremos também introduzir um processo de marcar variáveis x_i num dos limites inferior ou superior. Esse processo vai reduzir a dimensão do BCQP, proporcionando uma muito mais fácil utilização do algoritmo de pesquisa em árvore com limites ou de fluxo máximo.

Consideremos uma solução básica $\bar{x}$ do BCQP geral, isto é, um ponto extremo do conjunto Ω definido por (9.2). Se considerarmos os conjuntos

$$F = \{i : \bar{x}_i = b_i\},\ T = \{i : \bar{x}_i = 0\}$$

então podemos escrever as relações lineares do MLCP (9.10) na seguinte forma:

	1	x_F	x_T	β_F	β_T
$z_0 =$	α	q_F	q_T	$-b_F$	$-b_T$
$w_F =$	q_F	M_{FF}	M_{FT}	$I_{\|F\|}$	0
$w_T =$	q_T	M_{TF}	M_{TT}	0	$I_{\|T\|}$
$v_F =$	b_F	$-I_{\|F\|}$	0	0	0
$v_T =$	b_T	0	$-I_{\|T\|}$	0	0

(9.11)

onde $\alpha = 0$ é o valor da função objectivo do BCQP. À solução básica $\bar{x}$ referida atrás podemos associar um quadro semelhante, que se obtém de (9.11) por uma operação pivotal principal com pivot

$$\begin{bmatrix} M_{FF} & I_{|F|} \\ -I_{|F|} & 0 \end{bmatrix}$$

Esse quadro tem a forma

	1	v_F	x_T	w_F	β_T
$z_0 =$	$\bar{\alpha}$	$\bar{q}_F$	$\bar{q}_T$	$-b_F$	$-b_T$
$\beta_F =$	$\bar{q}_F$	M_{FF}	$-M_{FT}$	$I_{\|F\|}$	0
$w_T =$	$\bar{q}_T$	$-M_{TF}$	M_{TT}	0	$I_{\|T\|}$
$v_F =$	b_F	$-I_{\|F\|}$	0	0	0
$v_T =$	b_T	0	$-I_{\|T\|}$	0	0

(9.12)

com

$$\bar{q}_F = -(q_F + M_{FF}\, b_F),\ \bar{q}_T = q_T + M_{TF}\, b_F$$

$$\bar{\alpha} = \alpha + 2b_F^T q_F + b_F^T M_{FF}\, b_F \qquad (9.13)$$

Se $\bar{q} = (\bar{q}_F, \bar{q}_T) \geq 0$, então a solução básica dada pelo quadro (9.12) é em geral um mínimo local do BCQP. Contudo podem existir pontos extremos adjacentes de $\bar{x}$ para os quais o valor da função objectivo é menor que $\bar{\alpha}$. Se verificarmos atentamente a expressão de $\bar{\alpha} - \alpha$, facilmente concluímos que tal não acontece se

$$2\bar{q}_i + m_{ii}\, b_i \geq 0 \quad \text{para todo} \quad i = 1,\ldots, n \qquad (9.14)$$

Se existe um $r \in \{1,\ldots, n\}$ tal que $2\bar{q}_r + m_{rr}\, b_r < 0$, então a função objectivo $f(x)$ tem um valor menor que $\bar{\alpha}$ no ponto extremo adjacente de $\bar{x}$ que se obtém mudando o valor da variável x_r de um limite para o outro. A esse ponto extremo podemos associar um quadro da forma anterior que se obtém de (9.12) por uma operação pivotal principal com pivot

$$\begin{bmatrix} m_{rr} & 1 \\ -1 & 0 \end{bmatrix}$$

O processo é então repetido e terminará numa solução básica satisfazendo a condição (9.14). Como foi discutido no capítulo 2, uma tal solução é denominada um **ponto LSM**. A terminação do algoritmo é consequência do facto do valor da função decrescer em cada iteração. Os passos do processo são apresentados a seguir.

ALGORITMO LSMBCQP

Passo 0 - Faça NUPD = 0 , F = ∅ e T = {1,..., n}

Passo 1 - Para i = 1,..., n faça

Se $2\bar{q}_i + m_{ii}\, b_i < 0$ faça

(i) NUPD = NUPD + 1

(ii) Actualize F e T a partir de

$$F = \begin{cases} F - \{i\} & \text{se } i \in F \\ F \cup \{i\} & \text{se } i \notin F \end{cases}$$

e T = {1,..., n} - F

(iii) Actualize $\bar{q}$ a partir de (9.13)

Passo 2 - Se NUPD = 0, páre. ($x_F = b_F$, $x_T = 0$) é um ponto LSM. De outro modo faça NUPD = 0 e vá para Passo 1.

A implementação deste algoritmo é semelhante à do método PRPIVT discutido na secção 3 do capítulo 4.

Da descrição do algoritmo facilmente se conclui que este processo pode ser usado para o QP 0–1 sem necessidade de converter este tipo de problema num BCQP. Experiência computacional apresentada na última secção deste capítulo mostra que o LSM obtido por este processo é normalmente uma boa solução incumbente, isto é, uma solução para a qual o valor da função objectivo não está muito longe do valor óptimo da função. É ainda

interessante notar que, para o caso do QP 0–1, o algoritmo LSMBCQP é equivalente ao apresentado em [GUGUMI84] para o mesmo fim. No caso do BCQP é vantajoso utilizar o algoritmo descrito no capítulo 4 primeiramente, principalmente quando os elementos fora da diagonal são não negativos, e começar com os conjuntos F e T dados por esse processo.

Seguidamente iremos descrever um processo que procura marcar algumas variáveis num dos limites inferior ou superior, diminuindo assim a dimensão do BCQP ou QP 0–1 a resolver pelo método de pesquisa em árvore com limites ou de corte mínimo. Para isso, e tal como foi feito no capítulo 4, sejam K_F e K_T os conjuntos dos índices das variáveis x_i que vão mudar do limite superior para o inferior e inversamente respectivamente. Então tem-se

$$F = K_F \cup F_1 \ , \ T = K_T \cup T_1$$

onde F_1 e T_1 são os conjuntos dos índices das variáveis que permanecem sem alteração. Se na solução básica dada por (9.12) as variáveis x_i, $i \in K_F \cup K_T$ mudarem de um limite para o outro, então obtém-se uma solução básica com valores dados por

$$\begin{aligned}
\bar{\bar{q}}_{F_1} &= \bar{q}_{F_1} + M_{F_1K_F} b_{K_F} - M_{F_1K_T} b_{K_T} \\
\bar{\bar{q}}_{K_F} &= -\bar{q}_{K_F} - M_{K_FK_F} b_{K_F} + M_{K_FK_T} b_{K_T} \\
\bar{\bar{q}}_{K_T} &= -\bar{q}_{K_T} + M_{K_TK_F} b_{K_F} - M_{K_TK_T} b_{K_T} \\
\bar{\bar{q}}_{T_1} &= \bar{q}_{T_1} - M_{T_1K_F} b_{K_F} + M_{T_1K_T} b_{K_T}
\end{aligned} \tag{9.15}$$

Além disso o valor $\bar{\bar{\alpha}}$ da função objectivo associado a essa solução básica satisfaz a igualdade

$$\begin{aligned}
\bar{\bar{\alpha}} - \bar{\alpha} = b_{K_F}^T (2\,\bar{q}_{K_F} + M_{K_FK_F} b_{K_F} - M_{K_FK_T} b_{K_T}) + \\
+ b_{K_T}^T (2\,\bar{q}_{K_T} + M_{K_TK_T} b_{K_T} - M_{K_TK_F} b_{K_F})
\end{aligned} \tag{9.16}$$

Tendo em conta que a matriz M é simétrica, tem-se $M_{K_FK_T} = M_{K_TK_F}^T$ e podemos escrever a expressão (9.16) numa das seguintes formas:

$$\begin{aligned}
\bar{\bar{\alpha}} - \bar{\alpha} = b_{K_F}^T (2\,\bar{q}_{K_F} + M_{K_FK_F} b_{K_F} - 2M_{K_FK_T} b_{K_T}) + \\
+ b_{K_T}^T (2\,\bar{q}_{K_T} + M_{K_TK_T} b_{K_T})
\end{aligned} \tag{9.17}$$

ou

$$\bar{\alpha} - \alpha = b_{K_T}^T (2\,\bar{q}_{K_T} + M_{K_T K_T} b_{K_T} - 2M_{K_T K_F} b_{K_F}) + \\ + b_{K_F}^T (2\,\bar{q}_{K_F} + M_{K_F K_F} b_{K_F}) \tag{9.18}$$

Podemos ainda isolar a contribuição de uma variável x_i no valor da função objectivo quando muda de um limite para o outro. Assim, se $i \in K_F$, a sua contribuição é

$$b_i (2\bar{q}_i + m_{ii} b_i + 2 \sum_{\substack{j \in K_F \\ j \neq i}} m_{ij} b_j - 2 \sum_{j \in K_T} m_{ij} b_j)$$

Por outro lado, se $i \in K_T$, então a contribuição de x_i será dada por

$$b_i (2\bar{q}_i + m_{ii} b_i + 2 \sum_{\substack{j \in K_T \\ j \neq i}} m_{ij} b_j - 2 \sum_{j \in K_F} m_{ij} b_j)$$

Se essas quantidades são não negativas para quaisquer escolhas de K_F e K_T, então a variável x_i pode ser fixada no conjunto a que pertence, isto é, será marcada em zero ou b_i, dependendo de $i \in K_T$ ou $i \in K_F$ respectivamente. Na prática esta propriedade pode ser verificada de um modo simples, se introduzirmos as quantidades

$$\Delta_F^i = 2\bar{q}_i + m_{ii} b_i + 2 \sum_{\substack{j \in K_F \\ j \neq i}} m_{ij}^- b_j - 2 \sum_{j \in K_T} m_{ij}^+ b_j \tag{9.19}$$

$$\Delta_T^i = 2\bar{q}_i + m_{ii} b_i + 2 \sum_{\substack{j \in K_T \\ j \neq i}} m_{ij}^- b_j - 2 \sum_{j \in K_F} m_{ij}^+ b_j \tag{9.20}$$

onde

$$m_{ij}^- = \min\{m_{ij}, 0\}\ ,\ m_{ij}^+ = \max\{m_{ij}, 0\}$$

Com efeito, como $x_i \in \{0, b_i\}$ e $0 < b_i$ para todo $i = 1, \ldots, n$, então verificam-se as seguintes implicações:

$$\begin{cases} \Delta_F^i \geq 0 \ \Rightarrow \ x_i = b_i \ \ (i \in F) \\ \Delta_T^i \geq 0 \ \Rightarrow \ x_i = 0 \ \ (i \in T) \end{cases} \tag{9.21}$$

Este tipo de marcação de variáveis pode ser aplicada a um qualquer ponto extremo do conjunto Ω de restrições do BCQP. É evidentemente boa política aplicá-lo ao LSM obtido pelo algoritmo LSMBCQP. Suponhamos que tal aconteceu e sejam K_F e K_T os conjuntos das variáveis não marcadas por este tipo de processo e

$$K = K_F \cup K_T$$

Se o número de elementos $|K|$ de K é menor ou igual do que um, então o LSM é um mínimo global do BCQP. Isso é consequência imediata da definição do ponto LSM.

Seja agora $|K| > 1$ e suponhamos que o quadro associado ao ponto LSM tem a forma (9.12). Se calcularmos, a partir de (9.16), o valor $\bar{\bar{\alpha}}$ da função objectivo correspondente à solução básica que se obteria se mudássemos todas as variáveis x_i, $i \in K$, de um limite para o outro, então podem acontecer dois casos, que serão discutidos a seguir.

Caso 1 – Se $\bar{\bar{\alpha}} \geq \bar{\alpha}$, então o novo ponto extremo não deve ser considerado. Nesse caso o LSM é um mínimo global se $|K| \leq 2$. Com efeito apenas mudanças de um elemento poderiam dar lugar a melhorias no valor da função objectivo e tal é impossível por definição de ponto LSM. Se $|K| > 2$, então o LSM é uma solução incumbente para o método de pesquisa em árvore com limites. Além disso esse processo deve ser utilizado apenas para o BCQP correspondente às variáveis não marcadas x_i, $i \in K$, que terá por isso dimensão igual ao número de elementos do conjunto K.

Caso 2 – Se $\bar{\bar{\alpha}} < \bar{\alpha}$, então todas as variáveis x_i, $i \in K$, devem mudar os seus valores de um limite para o outro. Se $|K| \leq 2$, o novo ponto extremo é um ponto LSM, pois

$$\begin{cases} 2\bar{\bar{q}}_i + m_{ii}\, b_i = -\Delta_F^i > 0 & \text{se } i \in K_F \\ 2\bar{\bar{q}}_i + m_{ii}\, b_i = -\Delta_T^i > 0 & \text{se } i \in K_T \end{cases}$$

Tal como no Caso 1 esse ponto LSM é um mínimo global do BCQP.

Se $|K| > 2$ então a nova solução $\bar{\bar{x}}$ satisfaz as seguintes propriedades:

Teorema 9.2 – (i) $\bar{\bar{x}}$ pode não ser um ponto LSM

(ii) o conjunto K das variáveis não marcadas não é alterado.

Demonstração: Sejam $\bar{q}$ e $\bar{\bar{q}}$ os vectores transformados de q correspondentes às soluções básicas $\bar{x}$ e $\bar{\bar{x}}$. Se designarmos por J^1 e J^2 os conjuntos que caracterizam essas soluções básicas $\bar{x}$ e $\bar{\bar{x}}$, tem-se:

$$K = K_F^1 \cup F_T^1$$

$$F^2 = F^1 - K_F^1 \cup K_T^1, \quad T^2 = T^1 - K_T^1 \cup K_F^1$$

e

$$K_F^2 = K_T^1, \quad K_T^2 = K_F^1$$

Para demonstrar (i) teremos que provar que existe pelo menos um $i \in K$ tal que $2\bar{\bar{q}}_i + m_{ii}\, b_i$ não é necessariamente não negativo. Mas por (9.15) e (9.19) tem-se para $i \in K_F^1$

$$2\bar{\bar{q}}_i + m_{ii}\, b_i = 2\,(-\bar{q}_i - m_{ii}\, b_i - \sum_{\substack{j \in K_F^1 \\ j \neq i}} m_{ij}\, b_j + \sum_{j \in K_T^1} m_{ij}\, b_j) + m_{ii}\, b_i$$

$$= -\Delta_F^i 1 - 2 \sum_{\substack{j \in K_F^1 \\ j \neq i}} m_{ij}^+ b_j + 2 \sum_{j \in K_T^1} m_{ij}^- b_j$$

Por outro lado, se $i \in K_T^1$, por (9.15) e (9.20) vem

$$2\bar{\bar{q}}_i + m_{ii}\, b_i = 2\,(-\bar{q}_i - m_{ii}\, b_i - \sum_{\substack{j \in K_T^1 \\ j \neq i}} m_{ij}\, b_j + \sum_{j \in K_F^1} m_{ij}\, b_j) + m_{ii}\, b_i$$

$$= -\Delta_T^i 1 - 2 \sum_{\substack{j \in K_T^1 \\ j \neq i}} m_{ij}^+ b_j + 2 \sum_{j \in K_F^1} m_{ij}^- b_j$$

Então $2\bar{\bar{q}}_i + m_{ii}\, b_i$ não é necessariamente não negativo e isso prova a propriedade (i).

Para demonstrar (ii) basta provar que $\Delta_F^i 2 < 0$ e $\Delta_T^i 2 < 0$ para todo $i \in K$. Como $\bar{x}$ é um ponto LSM,

$$2\bar{q}_i + m_{ii}\, b_i \geq 0 \qquad \text{para todo } i \in K$$

Além disso, por (9.15), (9.19) e (9.20), tem-se

$$\Delta_F^i 2 = 2\bar{\bar{q}}_i + m_{ii}\, b_i + 2 \sum_{\substack{j \in K_F^2 \\ j \neq i}} m_{ij}^- b_j - 2 \sum_{j \in K_T^2} m_{ij}^+ b_j$$

$$= 2\,(-\bar{q}_i - m_{ii}\, b_i - \sum_{\substack{j \in K_T^1 \\ j \neq i}} m_{ij}\, b_j + \sum_{j \in K_F^1} m_{ij}\, b_j) +$$

$$+ m_{ii}\, b_i + 2 \sum_{\substack{j \in K_T^1 \\ j \neq i}} m_{ij}^- b_j - 2 \sum_{j \in K_F^1} m_{ij}^+ b_j$$

$$= -\,(2\bar{q}_i + m_{ii}\, b_i + 2 \sum_{\substack{j \in K_T^1 \\ j \neq i}} m_{ij}^+ b_j - \sum_{j \in K_F^1} m_{ij}^- b_j) < 0$$

e

$$\Delta_T^i 2 = 2\bar{\bar{q}}_i + m_{ii}\, b_i + 2 \sum_{\substack{j \in K_T^2 \\ j \neq i}} m_{ij}^- b_j - 2 \sum_{j \in K_F^2} m_{ij}^+ b_j$$

$$= 2 \left(-\bar{q}_i - m_{ii}\, b_i - \sum_{\substack{j \in K_F^1 \\ j \neq i}} m_{ij}\, b_j + \sum_{j \in K_F^1} m_{ij}\, b_j\right) +$$

$$+ m_{ii}\, b_i + 2 \sum_{\substack{j \in K_F^1 \\ j \neq i}} m_{ij}^- b_j - 2 \sum_{j \in K_T^1} m_{ij}^+ b_j$$

$$= -\left(2\bar{q}_i + m_{ii}\, b_i + 2 \sum_{\substack{j \in K_F^1 \\ j \neq i}} m_{ij}^+ b_j - \sum_{j \in K_T^1} m_{ij}^- b_j\right) < 0. \quad \blacklozenge$$

Por este teorema há toda a vantagem em aplicar o algoritmo LSMBCQP após a obtenção da solução $\bar{\bar{x}}$. Se tal acontecer, então o conjunto $\tilde{K}$ das variáveis não marcadas correspondente ao novo ponto LSM $\tilde{x}$ obtido pelo método LSMBCQP é igual a K. Com efeito, basta notar que $\tilde{x}$ pode ser obtido do ponto LSM $\bar{x}$ por uma operação pivotal principal e usar a demonstração do teorema 9.2 (ii). Portanto, após a aplicação do algoritmo LSMBCQP, não há qualquer necessidade de investigar se algumas variáveis podem ainda ser marcadas.

Estas considerações sugerem o seguinte algoritmo que permite determinar uma boa solução incumbente e ao mesmo tempo marcar algumas variáveis do BCQP:

ALGORITMO MINBCQP

Passo 0 - Aplique o algoritmo BLOCK ou PRPIVT da secção 3 do capítulo 4. Sejam F e T os conjuntos correspondentes à solução obtida por esse algoritmo. Faça $K_F = F$, $K_T = T$ e $NK = n$.

Passo 1 - Aplique o algoritmo LSMBCQP. Faça NMARK = 0.

Passo 2 - Para $i = 1, \ldots, NK$ faça

(i) Se $i \in K_F$, calcule Δ_F^i a partir de (9.19). Se $\Delta_F^i \geq 0$ faça $K_F = K_F - \{i\}$ e NMARK = NMARK+1.

(ii) Se $i \in K_T$, calcule Δ_T^i a partir de (9.20). Se $\Delta_T^i \geq 0$ faça $K_T = K_T - \{i\}$ e NMARK = NMARK+1.

Passo 3 - Se NMARK = 0, vá para Passo 4. De outro modo faça NK = NK – NMARK e vá para Passo 2 ou Passo 4, dependendo de NK > 2 ou NK ≤ 2 respectivamente.

Passo 4 - Se NK ≤ 1, faça TERM = 1 e vá para EXIT. De outro modo calcule $\bar{\alpha}$ a partir de (9.16).

(i) Se $\bar{\alpha} \leq \bar{\bar{\alpha}}$, vá para EXIT com TERM = 1 (se NK ≤ 2) ou TERM = 2 (se NK > 2).

(ii) Se $\bar{\alpha} > \bar{\bar{\alpha}}$, faça $F = F - K_F \cup K_T$, $T = T - K_T \cup K_F$. Se NK ≤ 2, faça TERM = 1 e vá para EXIT. De outro modo actualize $\bar{q}$ a partir de (9.15) e $\bar{q} = \bar{\bar{q}}$. Aplique o algoritmo LSMBCQP, faça TERM = 2 e vá para EXIT.

EXIT - Se TERM = 1, a solução

$$x_i = b_i \ , \ i \in F \ ; \ x_i = 0 \ , \ i \in T \qquad (9.22)$$

é um mínimo global do BCQP. Se TERM = 2, a solução (9.22) é uma solução incumbente para o BCQP. Além disso há apenas a necessidade de resolver um BCQP de dimensão NK.

Este algoritmo pode também ser utilizado para a solução do QP 0–1 desde que se altere o Passo 0. Com efeito, os métodos pivotais principais BLOCK ou PRPIVT só podem ser utilizados para o BCQP côncavo. Deste modo pode começar-se o Passo 1 com $F = \emptyset$ e $T = \{1,\ldots, n\}$.

O algoritmo MINBCQP pode também ser aplicado em cada nó da árvore gerada pelo algoritmo de pesquisa em árvore com limites. Nesse caso se TERM = 1 ocorre, nenhum dos nós abaixo do nó corrente necessita de ser investigado.

O algoritmo MINBCQP é muito mais eficiente no caso em que a matriz M é não positiva. Com efeito verifica-se o seguinte teorema:

Teorema 9.3 – Seja $M \leq 0$ e $x = (b_F, 0)$ a solução obtida pelo algoritmo LSMBCQP, com $F = \emptyset$ inicialmente. Então as variáveis $x_i = b_i$ podem ser marcadas, isto é, NK = n – | F |, com | F | número de elementos do conjunto F.

Demonstração: Como $F = \emptyset$, então também é vazio o conjunto K_F das variáveis x_i a mudar de valor de b_i para 0. Se $x = 0$ não é um ponto LSM, tem de existir pelo menos um $i \in T$ tal que

$$2q_i + m_{ii}\, b_i < 0$$

Então o algoritmo LSMBCQP faz

$$F = \{i\} \quad \text{e} \quad T = \{1,\ldots, n\} - \{i\}$$

Após esta mudança $K_F = \{i\}$ e portanto

$$\Delta_F^i = 2\bar{q}_i + m_{ii}\, b_i + 2 \sum_{\substack{j \in K_F \\ j \neq i}} m_{ij}\, b_j$$

$$= -2(q_i + m_{ii}\, b_i) + m_{ii}\, b_i$$

$$= -(2q_i + m_{ii}\, b_i) > 0$$

Então $x_i = b_i$ pode ser marcada e $K_F = \emptyset$.

Suponhamos agora que numa iteração k do algoritmo LSMBCQP se tem uma solução $\bar{x}$ tal que todas as variáveis $x_i = b_i$, $i \in F^k$ estão marcadas. Donde $K_F = \emptyset$ e, se $\bar{x}$ não é um ponto LSM, existe pelo menos um $i \in T$ tal que

$$2\bar{q}_i + m_{ii}\, b_i < 0$$

Obtém-se uma nova solução $\bar{\bar{x}}$ cujo conjunto F^{k+1} satisfaz a

$$F^{k+1} = F^k \cup \{i\}$$

Além disso $K_F = \{i\}$ e portanto

$$\Delta_F^i = 2\bar{\bar{q}}_i + m_{ii}\, b_i + \sum_{j \in K_F} m_{ij}\, b_j$$

$$= -2(q_i + \sum_{j \in F^{k+1}} m_{ij}\, b_j) + m_{ii}\, b_i$$

$$= -2(q_i + \sum_{j \in F^k} m_{ij}\, b_j) - m_{ii}\, b_i$$

$$= -(2\bar{q}_i + m_{ii}\, b_i) > 0$$

Então x_i pode ser marcada em b_i e o resultado fica demonstrado por indução. ♦

Por este teorema, sempre que uma variável toma o valor b_i pode ser imediatamente marcada. Por isso no Passo 2 apenas há a necessidade de investigar as quantidades Δ_T^i.

4 - Experiência Computacional

Nesta secção apresentamos alguma experiência computacional da resolução do QP 0–1 no caso particular de $m_{ij} \leq 0$ se $i \neq j$. Para testar a eficiência dos métodos geraram-se aleatoriamente alguns problemas-teste que foram resolvidos usando um computador CDC CYBER 180-830 da Universidade do Porto. Como referimos anteriormente, o método de pesquisa em árvore com limites não é o mais indicado para este tipo de problemas. A redução a um problema de corte mínimo é bastante mais recomendada, pois esse problema é resolvido com bastante eficiência por um método de fluxo máximo e pelo método de marcação de Ford-Fulkerson. Por essa razão, apenas para os problemas de menor dimensão é apresentada alguma experiência com o método de pesquisa em árvore com limites (BRANCH–AND–BOUND), destacando-se na coluna encabeçada por OPTIMO o esforço computacional para obter o óptimo global. O algoritmo de pesquisa em árvore com limites incorpora numa fase inicial o algoritmo MINBCQP (com $F = \emptyset$ no Passo 0). Na informação referente a este algoritmo, indica-se na coluna OP o número de vezes que esse óptimo global é encontrado por esse processo. Para resolver o problema de corte mínimo usou-se um algoritmo muito semelhante ao apresentado em [MASA91]. Nos resultados apresentados esse algoritmo é aplicado ao grafo do problema completo (FLUXTO) ou apenas ao grafo do subproblema definido pelas NK variáveis não marcadas pelo algoritmo MINBCQP (FLUXINC) e que tem por isso NK+2 nós.

As matrizes dos problemas-teste foram obtidas segundo a técnica apresentada em [PAROD90], de modo a que os elementos fora da diagonal tivessem uma percentagem predefinida de elementos inteiros negativos pertencentes a [–50, –1]. Os elementos da diagonal pertencem a [–50, 50] ou [–100, 100]. Esses elementos foram gerados aleatoriamente através de um gerador GGBUFS usado na IBM tal que

$$DS = (16807 \times DS) \bmod (2^{31} - 1)$$

$$GGBUFS = DS/2^{31}$$

onde DS é uma semente previamente fornecida em dupla precisão. Para cada matriz M foram gerados aleatoriamente cinco termos independentes positivos. Nos quadros 9.1 e 9.2 são apresentados os resultados dos diferentes algoritmos, mostrando o melhor (B), o médio (M) e o pior (P) em termos de tempo de execução. Nesses quadros usam-se as seguintes notações:

n = número de variáveis x do QP

NZ = número de não zeros fora da diagonal de M

NI = número de iterações do algoritmo MINBCQP = número de passagens pelo Passo 1 do algoritmo LSMBCQP + número de passagens pelo Passo 2 do algoritmo MINBCQP

T = tempo parcial de CPU em segundos

TT = tempo total de CPU em segundos

NK = número de variáveis não marcadas

NR = número de ramos explorados

Para cada problema fixámos um tempo máximo de 100 segundos para a aplicação do algoritmo de pesquisa em árvore com limites. Dado que o algoritmo não foi capaz de resolver alguns dos cinco problemas em menos de 100 segundos, optou-se por apresentar, na linha referente à notação (M), o número de vezes que o algoritmo teve sucesso. Além disso, na linha referente à notação (P) é apresentado o esforço computacional necessário para resolver o pior dos problemas em que o algoritmo terminou. Assim, numa mesma linha (P), as informações contidas nas colunas NR e TT e nas colunas enquadradas por OPTIMO (NR e T) não correspondem necessariamente ao mesmo problema.

				MINBCQP				BRANCH-AND-BOUND				FLUXTO	FLUXINC	
										OPTIMO				
Probl	n	NZ		NI	TT	NK	OP	NR	TT	NR	T	TT	TT	NK
			B	4	.06	0		0	.06	0	.06	.12	.06	0
PT1	500	5038	M	4	.16	279	5	3		0	.16	2.2	1.2	279
			P	5	.26	416		5	3.3	0	.26	4.1	2.6	480
			B	3	.08	0		0	.08	0	.08	.14	.08	0
PT2	500	7574	M	4	.25	293	5	3		0	.25	2.0	1.6	293
			P	10	.49	0		2	.52	0	.49	5.0	4.8	477
			B	4	.09	0		0	.09	0	.09	.73	.09	0
PT3	1000	7096	M	6	.29	320	4	3		2844	9.5	17.3	6.2	320
			P	6	.49	711		2	.11	14221	46.3	35.5	15.8	882
			B	3	.12	0		0	.12	0	.12	.21	.12	0
PT4	1000	10136	M	3	.25	393	5	4		0	.25	3.6	2.2	393
			P	4	.45	965		2	10.	0	.45	13.2	9.9	965

Quadro 9.1

A análise dos resultados apresentados no quadro 9.1 mostra que o método de pesquisa em árvore com limites não é robusto. Com efeito, o algoritmo é muito eficiente quando

consegue terminar, mas em geral é incapaz de em tempo útil garantir o mínimo global do BCQP. A solução incumbente obtida pelo método MINBCQP permite obter, de um modo geral, o óptimo global rapidamente.

O algoritmo MINBCQP é extremamente eficiente, mesmo para os problemas mais difíceis. O tempo de execução parece depender mais da densidade do problema do que da sua dimensão. Em geral, quanto mais esparso é o problema, menor é o tempo de execução e maior é o número de variáveis marcadas pelo algoritmo. O algoritmo FLUXTO parece ser mais sensível à dimensão do problema do que à sua densidade. Dos três métodos globais apresentados, o algoritmo FLUXINC mostrou-se o mais eficiente para resolver este tipo de problemas, apesar do algoritmo FLUXTO ser bastante eficiente. Isto deve-se ao facto do algoritmo MINBCQP ser extremamente rápido e portanto o seu tempo de execução pouco afectar o tempo total de execução do processo FLUXINC mesmo quando poucas variáveis são marcadas. Como, dum modo geral, grande parte das variáveis são marcadas pelo algoritmo MINBCQP e a dimensão do grafo afecta de modo significativo a eficiência do método de fluxo máximo, torna-se evidente a vantagem do algoritmo FLUXINC.

				MINBCQP				FLUXTO	FLUXINC	
Probl.	n	NZ		NI	TT	NK	OP	TT	TT	NK
PT1	500	5038	B	4	.06	0	5	.12	.06	0
			M	4	.16	279		2.2	1.2	279
			P	5	.26	416		4.1	2.6	480
PT3	1000	7097	B	4	.09	0	4	.73	.09	0
			M	6	.29	320		17.3	6.2	320
			P	6	.49	711		35.5	15.8	882
PT5	2000	12020	B	6	.17	2	3	4.4	.17	0
			M	10	.66	424		77.3	22.4	424
			P	22	1.8	531		240.	107.	1572
PT6	3000	13428	B	8	.23	6	2	24.2	.23	6
			M	9	.59	343		142.	15.1	343
			P	10	1.5	1405		340.	73.2	1405

Quadro 9.2

No quadro 9.2 são apresentados alguns ensaios com matrizes de dimensões maiores. Os resultados confirmam algumas das conclusões referidas anteriormente. É no entanto ainda mais evidente que o número de variáveis que são marcadas pelo algoritmo MINBCQP aumenta com a esparsidade do problema. Tal como anteriormente, o

algoritmo FLUXTO é mais sensível à variação da dimensão do que à variação da esparsidade. Daí se concluir a grande vantagem do método FLUXINC para BCQPs muito esparsos e de dimensões elevadas.

Como conclusão final deste estudo computacional, pode afirmar-se que a incorporação do algoritmo MINBCQP num método de resolução de corte mínimo é altamente recomendável. O algoritmo deve também ser incorporado num método de pesquisa em árvore com limites. Contudo, mesmo assim, este último método não é de modo algum competitivo com o outro tipo de técnica.

Se a matriz do BCQP tem pelo menos um elemento não diagonal positivo, então o método de corte mínimo não pode ser usado. Por outro lado, apesar do método MINBCQP ser igualmente eficiente para obter boas soluções incumbentes, é de admitir que não consiga marcar tantas variáveis. Por isso há que esperar que o método de pesquisa em árvore com limites ainda seja menos recomendável neste caso. Daí a necessidade de encontrar processos alternativos que possam resolver o BCQP no caso geral. Esses processos devem incorporar o algoritmo MINBCQP de modo a obter uma boa solução incumbente e até marcar algumas variáveis. Essa será certamente uma área muito importante da nossa investigação futura.

CONCLUSÃO

Nesta tese procurámos desenvolver algoritmos para o Problemas Linear Complementar (LCP) Não Convexo (em que a matriz não é positiva semi-definida) e algumas das sua generalizações (GLCP, BLCP e MLCP), aplicando esses processos na resolução de alguns problemas de optimização global. Todos esses algoritmos foram testados para problemas de médias e grandes dimensões, tendo sido desenvolvidas implementações eficientes para o efeito. Na sequência desse trabalho indicam-se, ao longo desta tese, várias pistas para investigação futura.

No capítulo 2 discutimos a eficiência de um método enumerativo para a resolução do LCP e GLCP. Após a introdução de várias técnicas heurísticas, debruçámo-nos sobre o algoritmo de gradiente reduzido modificado (MRG) de Al-Khayyal e propusemos algumas extensões e simplificações para certos casos particulares. Experiência computacional com LCPs não convexos, de dimensão média, mostrou que o algoritmo é capaz de resolver eficientemente esses problemas.

No capítulo 3 estudámos o LCP côncavo, isto é, o caso em que a sua matriz é negativa semi-definida. Mostrámos que o método enumerativo híbrido discutido no capítulo 2 é extremamente simplificado neste caso, pois o algoritmo MRG passa a ser um método tipo simplex. Nesse mesmo capítulo foi ainda proposto um método polinomial para o LCP côncavo quando os elementos não diagonais da sua matriz são não negativos. Experiência computacional indica que este último método é muito eficiente para resolver LCPs de grandes dimensões. Esses resultados sugerem a importância de investigar no futuro outras classes de LCPs côncavos para as quais seja possível desenvolver métodos directos e iterativos.

No capítulo 4 apresentámos extensões dos métodos de Lemke e Keller para o BLCP. Mostrámos que o primeiro processo é capaz de resolver pelo menos BLCPs convexos e BLCPs não convexos com os limites todos finitos. Além disso, a extensão do método de Keller pode ser usada para BLCPs côncavos bissimétricos. Experiência computacional com BLCPs côncavos um pouco mais gerais levou-nos a conjecturar a sua utilidade para esses problemas. Desenvolvemos ainda dois algoritmos pivotais principais muito eficientes para a resolução do BLCP côncavo quando todos os limites são finitos. O uso deste métodos para outros casos especiais do BLCP, assim como o desenvolvimento de um algoritmo enumerativo para o BLCP geral, são certamente áreas de investigação bastante importantes. Aliás, algumas ideias para este último tipo de processos foram apresentadas na parte final do capítulo.

No capítulo 5 apresentámos um algoritmo sequencial LCP (SLCP) para a resolução do MLCP. Nesse processo é resolvida uma sucessão de LCPs (ou GLCPs) usando o método enumerativo híbrido discutido em capítulos anteriores. O algoritmo termina quando aparecer um LCP (ou GLCP) sem solução. Esta é a grande desvantagem do algoritmo. Assim, o processo é bastante eficiente para terminar uma solução ε-óptima do MLCP, mas torna-se incapaz de terminar quando a dimensão é elevada. O estabelecimento de condições suficientes ou processos capazes de ultrapassar esta desvantagem terá certamente um grande impacte na optimização global. Com efeito, este método SLCP pode ser utilizado na resolução de alguns problemas de optimização global com as vantagens e desvantagens já referidas. As programações bilineares, de dois níveis e quadráticas não convexas são exemplos desses problemas e foram objecto de estudo nos capítulos 6, 7 e 8. A utilidade deste método SLCP para outros problemas de optimização global será certamente uma das nossas prioridades futuras.

No último capítulo estudámos a resolução de problemas de optimização quadrática côncava apenas com limites finitos nos valores das variáveis. Foi desenvolvida uma técnica bastante eficiente para aliviar a pesquisa de um mínimo global desse problema. Esses resultados têm implicações importantes na resolução de programas quadráticos 0–1. Acreditamos que outros processos poderão ser desenvolvidos para o mesmo fim, mostrando a importância de se considerar alguns problemas de optimização combinatória no contexto da optimização global.

REFERÊNCIAS

[AH83] - Ahn, B.H. Iterative methods for linear complementarity problems with upper-bounds on primary variables, Mathematical Programming 26 (1983) 295-315.

[AL86a] - Al-Khayyal, F.A., Linear, quadratic and bilinear programming approaches to the linear complementarity problem, European Journal of Operational Research 24 (1986) 216-227.

[AL86b] - Al-Khayyal, F.A., Further simplified characterizations of linear complementarity problems solvable by linear programs, Technical Repport PDRC 86-08, School of Industrial and Systems Engineering, Georgia Institute of Technology (1986).

[AL87] - Al-Khayyal, F.A., An implicit enumeration procedure for the general linear complementarity problem, Mathematical Programming Studies 31 (1987) 1-20.

[AL89] - Al-Khayyal, F.A., On characterizing linear complementarity problems as linear programs, Optimization 20 (1989) 153-159.

[ANWH88] - Anandalingam G. e White, D.I., A penalty function approach for solving bilevel linear programs, Working Paper, Department of Systems, University of Pennsylvania (1988).

[BAFA80] - Bard, J.F. e Falk, J.E., Computing equilibria via nonconvex progamming, Naval Research Logistics Quarterly 27 (1980) 233-255.

[BAFA82a] - Bard, J.F. e Falk, J.E., An explicit solution to the multi-level programming problem, Computers and Operations Research 9 (1982) 77-100.

[BAFA82b] - Bard, J.F. e Falk, J.E., A separable programming approach to the linear complementarity problem, Computers and Operations Research 9 (1982) 153-159.

[BAJURE89] - Barahona, F., Jünger, M. e Reinelt, G., Experiments in quadratic 0-1 programming, Mathematical Programming 44 (1989) 127-137.

[BAMO90] - Bard, J.F. e Moore, J.T., A branch and bound algorithm for the bilevel programming problem, SIAM Journal of Scientific and Statistical Computing 11 (1990) 281-292.

[BARA86] - Barahona, F., A solvable case of quadratic 0-1 programming, Discrete Applied Mathematics 13 (1986) 23-26.

[BARD83] - Bard, J.F., An efficient point algorithm for a linear two-stage optimization problem, Operations Research 31 (1983) 670-684.

[BE65] - Bennett, J.M., Triangular factors of modified matrices, Numerische Mathematik 7 (1965) 217-221.

[BEBLBO88] - Ben-Ayed, O., Blair, C.E. e Boyce D.E. Solving a real world highway network design problem using bilevel linear programming, BEBR Faculty Working Paper 1463, University of Illinois, Urbana Champaign, 1988.

[BIKA84] - Bialas, W.F. e Karwan, M.H., Two-level linear programming, Management Science 30 (1984) 1004-1020.

[BL77] - Bland, R.G., New finite pivoting rules for the simplex method, Mathematics of Operations Research 2 (1977) 103-107.

[CAFR70] - Cabot, A.V. e Francis, R.L., Solving certain nonconvex quadratic minimization problems by ranking the extreme points, Operations Research 18 (1970) 421-448.

[CATO82] - Candler, W. e Townsley, R., A linear two-level programming problem, Computers and Operations Research 9 (1982) 59-76.

[CHCO80] - Chang, Y.Y. e Cottle, R.W., Least-index resolution of degenerancy in quadratic programming, Mathematical Programming 18 (1980) 127-137.

[CHD70] - Chandrasekaran, R., A special case of the complementarity pivot problem, Opsearch 7 (1970) 263-268.

[CHG79] - Chang, Y.Y, Least-index resolution of degenerancy in linear complementarity problems, Technical Report 79-14, Department of Operations Research, Stanford University, California (1979).

[CO68] - Cottle, R.W., The principal pivoting method of quadratic programming, em "Mathematics of Decision Sciences", editado por G.B. Dantzig e A.F. Veinott Jr., American Mathematical Society, Providence(1968), pp.144-162.

[CO74] - Cottle, R.W., Manifestations of the Schur Complement, Linear Algebra and its Applications 8 (1974) 189-211.

[CO90] - Cottle, R.W., private communication (1990).

[CODA68] - Cottle, R.W. e Dantzig, G.B., Complementary pivot theory of mathematical programming, em "Mathematics of Decision Sciences", editado por G.B. Dantzig e A.F. Veinott Jr., American Mathematical Society, Providence (1968), pp. 55-73.

[CRHA69] - Crabtree, D.E. e Haynsworth, E.V., An identity for the Schur Complement of a matrix, Proceedings of American Mathematical Society 22 (1969) 364-366.

[DILI91] - Ding, J. e Li, T.Y., A polynomial-time predictor-corrector algorithm for a class of linear complementarity problems, SIAM Journal on Optimization 1 (1991) 83-92.

[EDBA91] - Edmunds, T.A. e Bard, J.F., Algorithms for nonlinear bilevel mathematical programs, IEEE Transactions on Systems, Management and Cybernetics 21 (1991) 83-89.

[ERGRLEPO85] - Erisman, A.M., Grimes, R.G., Lewis, J.G. e Poole, Jr.W.G., A structurally stable modification of Hellerman-Rarick's P4 algorithm for reordering unsymmetric sparse matrices, SIAM Journal of Numerical Analysis 22 (1985) 369-385.

[FLPA90] - Floudas, C.A. e Pardalos, P.M., "A Collection of Test Problems for Constrained Global Optimization Algorithms", Lecture Notes in Computers Science 455, Springer-Verlag, Berlin Heidelberg (1990).

[FOMC81] - Fortuny-Armat, J. e McCarl, B., A representation and economic interpretation of a two-level programming problem, Journal of the Operational Research Society 32 (1981) 783-792.

[FUPL81] - Funderlic, R.E. e Plemmons, R.J., LU decompositions of M-matrices by elimination without pivoting, Linear Algebra and its Applications 41 (1981) 99-110.

[GALE71] - Garcia, G.B. e Lemke, C.E., All solutions to linear complementarity problems by implicit search, comunicação apresentada no "39th Meeting of the Operations Research Society of America" (1971).

[GAUL77] - Gallo, G. e Ülkücü, A., Bilinear programming: an exact algorithm, Mathematical Programming 12 (1977) 173-194.

[GELI81] - George, A. e Liu, J.W.H., "Computer Solution of Large Sparse Positive Definite Systems", Prentice-Hall, Englewood Cliffs, New Jersey (1981).

[GIMUW81] - Gill, P.E., Murray, W. e Wright, M.H., "Practical Optimization", Academic Press, New York (1981).

[GLKLMOWH79] - Glover, F., Klingman, D., Mote, J. e Whitman, D., A primal simplex variant for the maximum flow problem, Technical Report CCS 362 Center for Cybernetic Studies, University of Texas, Austin (1979).

[GO86] - Gould, N.I.M., An algorithm for large scale quadratic programming, a aparecer em IMA Journal of Numerical Analysis.

[GUGUMI84] - Gulati, V.P. Gupta, S.K. e Miltal, A.K., Unconstrained quadratic bivalent programming problem, European Journal of Operations Research 15 (1984) 121-125.

[HAJASA89] - Hansen, P., Jaumard, B. e Savard, G., A variable elimination algorithm for bilevel linear programming, Rutcor Research Report 17-89, RUTCOR, Rutgers University (1989).

[HAPA90] - Harker, P.T. e Pang, J.S., Finite-dimensional variational inequality and nonlinear complementarity problem: a survey of theory, algorithms and applications, Mathematical Programming 48 (1990) 161-220.

[HERA71] - Hellerman, E. e Rarick, D., Reinversion with the preassigned pivot procedure, Mathematical Programming 1 (1971) 195-216.

[HERA72] - Hellerman, E. e Rarick, D., The partitioned preassigned pivot procedure (p4), em "Sparse Matrices and their Applications", editado por D.J. Rose e R.A. Willoughby, Plenum Press, New York (1972), pp. 67-76.

[IB71] - Ibaraki, T., Complementarity programming, Operations Research 19 (1971) 1523-1528.

[JE78] - Jeroslaw, R.G., Cutting-planes for complementarity constraints, SIAM Journal of Control and Optimization 16 (1978) 56-62.

[JE85] - Jeroslaw, R.G., The polynomial hierarchy and a simple model for competitive analysis, Mathematical Programming 32 (1985) 146-164.

[JO79] - Josephy, N.H., Newton's method for generalized equations, Technical Report 1966, Mathematics Research Center, University of Wisconsin - Madison (1979).

[JU82] - Júdice, J.J., A study of the linear complementarity problems, Ph.D. thesis, Department of Mathematics and Statistics, Brunel University (1982).

[JU83] - Júdice, J.J., Classes of matrices for the linear complementarity problem, artigo apresentado no 1º Encontro de Algebra Linear e Aplicações, Outubro 1982, resumo publicado em Linear Algebra and its Applications 54 (1983) 122-125.

[JU90] - Júdice, J.J., Alguns resultados para matrizes Q e Q_0, em Actas das XV Jornadas Luso-Espanholas de Matemática, Vol. IV (1990) 439-444.

[JUFA86] - Júdice, J.J. e Faustino, A.M., Complementaridade e modelos de equilíbrio económico, Documento nº 26, CEMAPRE, Instituto Superior de Economia de Lisboa (1986).

[JUFA88a] - Júdice, J.J. e Faustino, A.M., An experimental investigation of enumerative methods for the linear complementarity problem, Computers and Operations Research 15 (1988) 417-426.

[JUFA88b] - Júdice, J.J. e Faustino, A.M., The solution of the linear bilevel programming problem by using the linear complementarity problem, Investigação Operacional 8, nº 1 (1988) 77-95.

[JUFA88c] - Júdice, J.J. e Faustino, A.M., A sequential LCP algorithm for bilinear and concave quadratic programming, Investigação Operacional 8, nº 2 (1988) 67-87.

[JUFA90] - Júdice, J.J. e Faustino, A.M., An implementation of Keller's method for two concave optimization problems, Investigação Operacional 10, nº 2 (1990) 95-109.

[JUFA91a] - Júdice, J.J. e Faustino, A.M., Solution of the concave linear complementarity problem, "Recent Advances in Global Optimization", editado por C.A. Floudas e P.M. Pardalos, Princeton University (1991), pp. 76-101.

[JUFA91b] - Júdice, J.J. e Faustino, A.M., A sequential LCP method for bilevel linear programming, a aparecer em Annals of Operations Research.

[JUFA91c] - Júdice, J.J. e Faustino, A.M., Solution of large-scale convex quadratic programs by Lemke's method, a aparecer em Actas da 1ª Conferência em Estatística e Optimização.

[JUFA91d] - Júdice, J.J. e Faustino, A.M., A computational analysis of LCP methods for bilinear and concave quadratic programming, Computers and Operations Research 18 (1991) 645-654.

[JUFA91e] - Júdice, J.J. e Faustino, A.M., The linear-quadratic bilevel programming problem, a aparecer em INFOR.

[JUMAFA91] - Júdice, J.J., Machado, J. e Faustino, A.M., An extension of Lemke's method for the solution of a generalized linear complementarity problem, a aparecer em Proceedings of the 15th IFIP Conference on System Modelling and Optimization.

[JUMI88a] - Júdice, J.J. e Mitra, G., An enumerative method for the solution of linear complementarity problems, European Journal of Operational Research 36 (1988) 122-128.

[JUMI88b] - Júdice, J.J. e Mitra, G., Reformulation of mathematical programming problems as linear complementarity problems and investigation of their solution methods, Journal of Optimization Theory and Applications 57 (1988) 123-149.

[JUPI88] - Júdice, J.J. e Pires, F.M., Bard-type methods for the linear complementarity problem with symmetric positive definite matrices, IMA Journal of Mathematics Applied in Business and Industry 2(1988/89) 51-68.

[JUPI89a] - Júdice, J.J. e Pires, F.M., Direct methods for convex quadratic programs subject to box constraints, Investigação Operacional 9 (1989) 23-56.

[JUPI89b] - Júdice, J.J. e Pires, F.M., A comparison between direct and iterative methods for solving large-scale convex quadratic programs on the simplex, Pesquisa Operacional 9 (1989) 55-78.

[KA84] - Karmarkar, N., A new polynomial-time algorithm for linear programming, Combinatorica 4 (1984) 373-395.

[KAHA78] - Kaneko, I. e Hallman, W.P., An enumeration algorithm for a general linear complementarity problem, Technical Report WP 78-11, University of Wisconsin - Madison (1978).

[KE73] - Keller, E.L., The general quadratic optimization problem, Mathematical Programming 5 (1973) 311-337.

[KI62] - Kirchgässner, K., Ein verfahren zur maximierung linear funktionen in nichtkonvexen bereichen, Zeitschrift fur Angewardt Mathematik 42 (1962) 22-24.

[KIST80] - Kinderlehrer, D. e Stampachia, G., "An Introduction to Variational Inequalities and their Applications", Academic Press, New York (1980).

[KOL85] - Kolstad, C.D., A review of the literature on bilevel mathematical programming, Technical Report LA - 10284 - MS, Los Alamos National Laboratory (1985).

[KOLA86] - Kolstad, C.D. e Lasdon, L.S., Derivate evaluation and computational experience with large bilevel mathematical program, BEBR Faculty Working Paper 1266, College of Commerce and Business Administration, University of Illinois at Urbana-Champaign (1986).

[KOMIYO89] - Kojima, M., Mizuno, S. e Yoshise, A., A polynomial-time algorithm for a class of linear complementarity problems, Mathematical Programming 44 (1989) 1-26.

[KON71] - Konno, H., Bilinear programming: Part II – Applications of bilinear programming, Technical Report 71-10, Department of Operations Research, Stanford University (1971).

[KON76] - Konno, H., A cutting-plane algorithm for solving bilinear programs, Mathematical Progamming 11 (1976) 14-27.

[LA85] - Law, K.H, Sparse matrix factor modification in structural reanalysis, International Journal for Numerical Methods in Engineering 21(1985) 37-63.

[LE68] - Lemke, C.E., On complementarity pivot theory, em "Mathematics of Decision Sciences", editado por G.B. Dantzig e A.F. Veinott Jr., American Mathematical Society, Providence (1968), pp. 95-114.

[LE70] - Lemke, C.E., Recent results on complementarity problems, em "Nonlinear Programming", editado por Rosen, Mangasarian e Ritter, Academic Press, New York (1970), pp. 350-384.

[LIPA89] - Lin, Y.Y. e Pang, J.S., Iterative methods for large convex quadratic programs: a survey, SIAM Journal on Control and Optimization 25 (1987) 383-411.

[MAN79] - Mangasarian, O.L., Simplified characterizations of linear complementarity problems solvable as linear programs, Mathematics of Operations Research 4 (1979) 268-273.

[MAR75] - Martos, B., "Nonlinear Programming: Theory and Methods", North-Holland (1975).

[MASA91] - Martins, E. e Salgueiro, R.P., An incremental chain algorithm for the maximal flow problem, comunicação apresentada na Workshop on Large--Scale Optimization, Coimbra (1991).

[MAT85] - Mathiesen, L., Computation of economic equilibria by a sequence of linear complementarity problems, Mathematical Programming Study 23 (1985) 144-162.

[MITJA79] - Mitra, G. e Jahanshalou, G.R., Linear complementarity problem and a tree search algorithm for its solution, em "Survey of Mathematical Programming", editado por A. Prekopa, North-Holland, Amsterdam (1979), pp. 35-56.

[MOBA90] - Moore, J.T. e Bard J.F., The mixed integer linear bilevel programming problem, Operations Research 38 (1990) 911-921.

[MOTO88] - Moré, J.J. e Toraldo, G., Algorithms for bound constrained quadratic programming problems, Numerische Mathematik 55 (1989) 377-400.

[MOTO91] - Moré, J.J. e Toraldo, G., On the solution of large quadratic programming problems with bound constraints, SIAM Journal on Optimization 1 (1991) 93-113.

[MU70] - Murty, K.G., Algorithm for finding all the feasible complementarity bases for a linear complementary problem, Technical Report 70-7, Department of Industrial and Operation Engineering, University of Michigan (1970).

[MU74] - Murty, K.G., Note on a Bard-type scheme for solving the complementarity problem, Operations Research 11 (1974) 123-130.

[MU76] - Murty, K.G., "Linear and Combinatorial Programming", Wiley (1976).

[MU83] - Murty, K.G., "Linear Programming", John Wiley & Sons, New York (1983).

[MU88] - Murty, K.G., "Linear Complementarity, Linear and Nonlinear Programming", Heldermann Verlag, Berlin (1988).

[OW73] - Owen, G., Cutting planes for programs with disjunctive constraints, Journal of Optimization Theory and Applications 11 (1973) 49-55.

[PAKAHA] - Pang, J.S., Kaneko, I. e Hallman, W.P., On the solution of some (parametric) linear complementarity problems with applications to portfolio analysis, structural engineering and graduation, Mathematical Programming 16 (1979) 325-347.

[PAN80] - Pang, J.S., A new and efficient algorithm for a class of portfolio selection problems, Operations Research 28 (1980) 754-767.

[PAP82] - Papavassilopoulos, G.P., Algorithms for static stackelberg games with linear cost and polyhedra constraints, Proceeding of the 21st IEEE Conference on Decision and Control (1982) 647-652.

[PAR89] - Pardalos, P.M., Performance of parallel branch and bound algorithm for quadratic 0-1 programming (1989).

[PAROD89] - Pardalos, P.M. e Rodgers, G.P., A branch and bound algorithm for the maximum clique problem, a aparecer em Mathematical Programming.

[PAROD90] - Pardalos, P.M. e Rodgers, G.P., Computational aspects of a branch and bound algorithm for quadratic zero-one programming, a aparecer em Computing.

[PAROS87] - Pardalos, P.M. e Rosen, J.B., "Constrained Global Optimization: Algorithms and Applications", Lectures Notes in Computer Science 268, Springer Verlag, Berlin, Heidelberg (1987).

[PAROS88] - Pardalos, P.M. e Rosen, J.B., Global optimization approach to the linear complementarity problem, SIAM Journal on Scientific and Statistical Computing (1988) 341-353.

[PAST82] - Papadimitriou, C. e Steiglitz, K., "Combinatorial Optimization: Algorithms and Complexity", Prentice-Hall, Englewood Cliffs, New Jersey (1982).

[PAYEHA90] - Pardalos, P.M., Ye, Y. e Han, C.G., An interior-point algorithm for large-scale quadratic programming problems with bound constraints, SIAM Journal on Optimization 1 (1991) 91-113.

[PAYEHAKA90] - Pardalos, P.M., Ye, Y., Han, C.G. e Kalisky, J.A., Solution of P-matrix linear complementarity problems using a potential reduction algorithm, Technical Report C5-90-32, Department of Computer Science, Pennsylvania State University (1990).

[PHRO88] - Phillips, A.T. e Rosen, J.B., A parallel algorithm for constrained concave quadratic global minimization, Mathematical Programming 42 (1988) 421-448.

[PI84] - Pissanetzky, S., "Sparse Matrix Technology", Academic Press, London (1984).

[PIRA74] - Picard, J.C. e Ratliff, H.D., Minimum cuts and related problems, Networks 5 (1974) 357-370.

[RASH84] - Ramarao, B. e Shetty, C.M., Application of disjunctive programming to the linear complementarity problem, Naval Research Logistics Quarterly 31 (1984) 589-600.

[RE82] - Reid, J.K., A sparsity exploiting variant of the Bartels-Golub decomposition for linear programming bases, Mathematical Programming 24 (1982) 55-69.

[SA89] - Savard, G., Contributions à la programation mathématique à deux niveaux, PhD. thesis, École Polytechnique de Montréal, Canada (1989).

[SAGA90] - Savard, G e Gauvin, J., The steepest descent method for the nonlinear bilevel programming problem (1990).

[SHSH80] - Sherali, H.D. e Sheety, C.M, A finitely convergent algorithm for bilinear programming problem using polar cuts and disjuntive face cuts, Mathematical Programming 19 (1980) 14-31.

[SOFA90] - Sousa, V. e Faustino, A.M., Programação quadrática não convexa na gestão integrada de mini-hídricas, Investigação Operacional 10, nº 1 (1990) 85-96.

[ST80] - Stewart, G.W., The efficient generation of random orthogonal matrices with an application to condition estimators, SIAM Journal Numerical Analysis 17 (1980) 403-409.

[TO72] - Tomlin, J.A., Pivoting for size and sparsity in linear programming inversion routes, Journal of Institute Mathematics and Applications 10(1972) 289-295.

[VASH77] - Vaish, H. e Shetty, C.M., A cutting-plane algorithm for the bilinear programming problem, Naval Research Logistics Quarterly 24(1977) 83-94.

[WI76] - Wirth, N., "Algorithms + Data Structures = Programs", Prentice-Hall, Englewood Cliffs, New Jersey (1976).

[WO59] - Wolfe, P., The simplex method for quadratic programming, Econometrica 27 (1959) 382-398.

[YATO85] - Yang, E.K. e Tolle, J.W., A class of methods for solving large-scale convex quadratic programs subject to box constraints, Technical Report, University of Massachussets, Boston (1985).